

Real-time suggestions for improving engagement of social media posts using machine learning

Anders Michael Nielsen & Benjamin Dalsgaard Hughes
s093035, s093020

DTU



Kongens Lyngby, 2015-02-15
DTU Compute

Technical University of Denmark

DTU Compute

Department of Applied Mathematics and Computer Science

Matematiktorvet, Building 303 B, DK-2800 Kongens Lyngby, Denmark

Phone +45 4525 3031

compute@compute.dtu.dk

www.compute.dtu.dk

Abstract

This thesis attempted to apply predictive analytics on Facebook, and integrate it to an enterprise environment in collaboration with the Danish company Falcon Social. While doing so, a framework that documents the process was outlined, which will also function as a way of implementing new big data projects and extending the one made in this thesis. Regression and classification models such as Random Forest, Decision Tree, and K-Nearest Neighbours was applied and evaluated against each other. They all attempted to predict how a Facebook post would perform in terms of the number of achieved likes, consumptions and impressions. Through careful validation with an unseen test set, it was possible to predict significantly better than random guessing. Especially prediction of likes seemed to have almost 72 % accuracy on above/below average estimates. A Median Absolute Percentage Error (MdAPE) estimate of 52.80 % on likes showed that regression was applicable as well. Some predictions proved to be far off, even more so when a post had achieved an amount of likes which was much larger than the average likes that the page would normally get.

Overall, the predictions performed better than random and there is still room for improvement. Random Forest seemed to almost always outperform or match the other models with statistical significance.

Keywords: Machine Learning, Facebook, Social Networks, Predictive Analytics

Preface

This thesis was prepared at the Section of Cognitive Systems, DTU Compute at the Technical University of Denmark in fulfilment of the requirements for acquiring a M. Sc. in Digital Media Engineering. It was made in collaboration with Falcon Social in the duration from September 2014 to February 2015 corresponding to 30 ECTS credits.

We would like to take the opportunity to thank our supervisor Ole Winther, PhD, Associate Professor, for outstanding guidance, mentoring and patience not only throughout this thesis, but also in previous related projects.

We would also like to thank Falcon Social for the partnership, especially Laszlo Fogas for supervising at Falcon Social, and Dennis Green-Lieber (CTO) for creating this opportunity and for his encouragements. Also, Helle Tyllesen and May Laursen for providing excellent insights in their organisation, Jeppe Toustrup, Jess Alfredsen (Falcon co-supervisor), Esben Sonne, Patrick Jensen, Morten Hansen, and Justina Malciute for technical expertise about Falcon Socials platform and machine learning.

We would also like to thank David Kofoed Wind, PhD student under Ole Winther, for co-supervising the project and for sharing his expertise within machine learning with us. Anders Munk-Nielsen (PhD student at the Department of Economics - University of Copenhagen) for helping us with regression analysis and various issues regarding statistics.

Finally, we would like to thank our families, friends, and significant others for their support, proofreading, and listening to our programming and mathematics rambling for the entire span of our education.

Thank you.

Anders Michael Nielsen & Benjamin Dalsgaard Hughes

Contents

1	Introduction	1
1.1	Introduction	1
1.1.1	Predictive Analytics	1
1.1.2	Falcon Social	1
1.1.3	Thesis outline	2
2	Context	3
2.1	Falcon Socials platform	3
2.1.1	Describing Facebook metrics	4
2.2	Falcon Social Predictive Publishing	6
2.3	The changing landscape of social media	7
2.4	Social media in Enterprise	8
3	Related work	11
3.1	The Adobe Social platform	11
3.2	Kaggle competitions	12
4	Framework	13
4.1	Data Preparation	13
4.1.1	Extraction	14
4.1.2	Transformation	14
4.1.3	Load	15
4.2	Exploration of Data	15
4.2.1	Outliers	15
4.2.2	Visualisations	15
4.2.3	Correlations	17
4.3	Feature engineering	19
4.3.1	User centric feature creation	19
4.3.2	Data centric feature creation	19
4.3.3	Feature selection	23
4.4	Choosing what to predict	25
4.5	Choosing the right model	26

4.5.1	Setting the right parameters	26
4.5.2	Describing the training, validation and test set	26
4.5.3	Problem of overfitting	27
4.6	Evaluation	29
4.6.1	Dummy models	30
4.6.2	Classification models	30
4.6.3	Regression models	32
4.6.4	Performance evaluation	33
4.7	From machine learning to product in enterprise	35
4.7.1	Enterprise challenge	35
4.7.2	Enterprise requirements	35
4.7.3	What should be delivered	36
4.7.4	Where can machine learning help overcome the challenge	36
4.7.5	Who are the users of the system and what should they be able to do	36
4.7.6	Solution development in an agile process	37
5	Methods	39
5.1	Decision Trees	39
5.1.1	Decision tree prediction example	40
5.2	Random Forest	41
5.2.1	Feature importance	42
5.3	Nearest neighbours	42
5.3.1	Describing different distance measures	43
5.3.2	Example of computing distances using different similarity measures	44
5.4	Agglomerative hierarchical clustering	44
5.5	Latent Dirichlet Allocation	45
5.5.1	Natural number of topics in LDA	47
6	Analysis	49
6.1	Data Architecture	49
6.2	ETL	51
6.3	Introduction to the data	52
6.4	Cleaning data	52
6.4.1	Data issues and missing values	53
6.5	Data Exploration	54
6.5.1	Correlation matrix	54
6.5.2	Distribution of posts	55
6.5.3	Outliers	56
6.5.4	The problem of UTC time and an international business	57
6.5.5	Customer exploration	58

6.5.6	Likes over time	64
6.6	Feature Engineering	66
6.6.1	User centric feature creation	66
6.6.2	Data centric feature creation	71
6.6.3	Performing feature Selection	78
6.6.4	Iterative Process of Feature Engineering	80
6.7	Clustering	81
6.8	Choosing What to Predict	81
6.8.1	Metrics for post engagement	82
6.8.2	Different company sizes	82
6.8.3	Classification	85
6.9	Model Selection	86
6.9.1	Choosing the right model	86
6.9.2	Parameters	86
6.9.3	Training, Validation and Test	87
7	Results and Evaluation	89
7.1	Results	89
7.1.1	Classification	89
7.1.2	Regression	92
7.2	Evaluation of models	94
7.2.1	Classification	94
7.2.2	Regression	96
7.3	Testing with unseen data	96
7.3.1	Classification	96
7.3.2	Regression	98
7.3.3	Weighting posts by time	98
7.3.4	Example Predictions	99
8	Describing the Product	101
8.1	RESTful web service	101
8.1.1	Current prediction	103
8.1.2	Predictions over time	103
8.1.3	Suggestions	103
8.2	Describing the website	104
8.2.1	Describing users and use cases	104
8.2.2	Live prediction	106
8.2.3	Prototypes	106
8.2.4	Usability test	110

9 Conclusion	117
9.1 Future work	118
9.1.1 Features, features, features	118
9.1.2 Business verticals	118
9.1.3 Paid reach	119
9.1.4 More social networks	119
9.1.5 Improved interface	119
9.1.6 Improved post weight	119
Bibliography	121
A Falcon’s project proposal	125
B PCA	127
C Topic Models	129
D Clustering	131
D.1 Agglomerative Clustering	131
D.2 K-means Clustering	131
E Extended Results	133

Chapter 1

Introduction

"Prediction is very difficult, especially about the future."

—Niels Bohr, 1918.

1.1 Introduction

In the recent rise of social media, with Facebook and Twitter in the lead, data on the web has increased exponentially, and companies are seeking to draw advantages of data which they have been collecting and storing for years. Social media have provided companies a way to reach out to their segmented audience easily. A key factor of success of a business to consumer (B2C) corporation is to have a social strategy.

1.1.1 Predictive Analytics

Attempting to predict the future using historical data and facts have been widely used across industries to everything from forecast stocks to detecting fraud or spam. Even though prediction may sometimes be wrong - even if they are often wrong - a slightly better guess than "rolling a dice" can yield significant value to a business. [Siegler, 2013] labels this phenomenon as "The Prediction Effect" and elaborates a bit further with "Predicting better than pure guesswork, even if not accurately, delivers real value".

And that is exactly the mindset one should have when attempting to predict social data. The nature of social data is influenced by numerous factors, and they are constantly changing. Achieving high accuracy in social data is unlikely, but the results can prove to be valuable to companies like Falcon Social and their customers.

1.1.2 Falcon Social

Falcon Social is a Danish company that provides a platform to help its clients gain insights in social media. Their entire business evolves around social data, and when the social media landscape shifts its tectonic

plates, Falcon Social follows. Their platform offers, among many things, the possibility to publish social media posts through the platform.

Falcon Social wishes to be able to automatically provide useful advice and predictions to a customer using their publishing tool when making posts on social media networks.

1.1.3 Thesis outline

This thesis aims towards making such a tool for Falcon Social while also providing a framework for future analysis as well as documenting the work.

Chapter 2-3 will begin by providing some examples of social platforms and machine learning communities and give a context to the domain and situation of Falcon Social.

Afterwards the predictive analytics framework will be outlined and theory of relevant models will be briefly touched. This is done in chapter 4 and 5, respectively.

Chapter 6 will apply the framework in order to develop the predictive analytics tool in an enterprise environment. Finally, the results and product are reviewed, and followed up by a conclusion.

Chapter 2

Context

Contents

2.1	Falcon Socials platform	3
2.2	Falcon Social Predictive Publishing	6
2.3	The changing landscape of social media	7
2.4	Social media in Enterprise	8

2.1 Falcon Socials platform

The social platform of Falcon Social is primarily targeted towards those kind of companies who have multiple brands/channels and several employees who are dedicated to social media. The platform consists of seven main categories: Listen, engage, publish, build, manage, inspire and measure. Figure 2.1.1 provides an overview of the functionalities.

Briefly, they cover the following areas of social media:

- Listen: Monitor specific keywords or follow competitor to address business opportunities and customer management
- Engage: Managing who (internally in the corporation) handles comments and feedback from users.
- Publish: A toolbox to post to any linked social media platform across brands, while also maintaining an overview of existing content.
- Build: The publish section also contains an app builder, which is a tool that helps customers build an app directly to their social media page.
- Manage: A layer that handles permissions and audit trails of employees to supplement a more streamlined social media strategy.

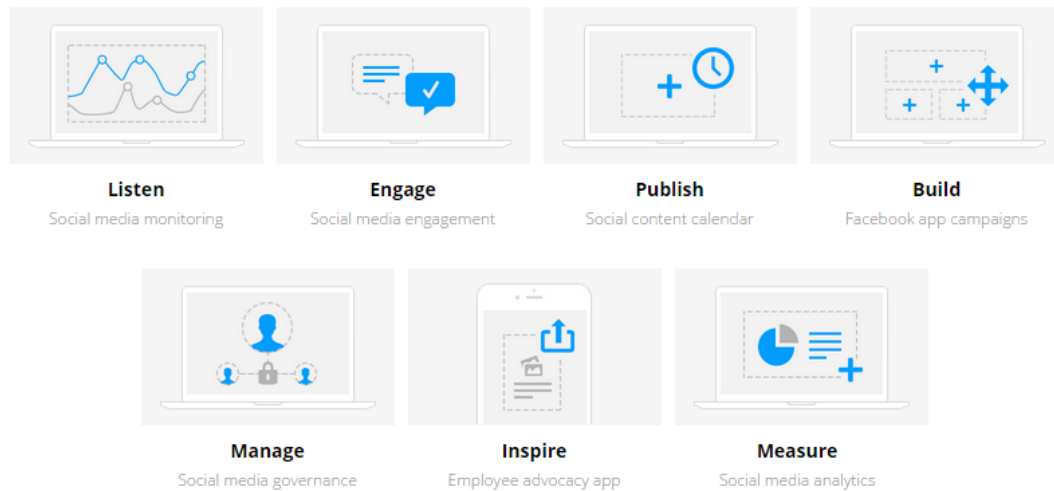


Figure 2.1.1 Falcon Socials platform categories

- **Inspire:** A mobile-only platform that is powered by their social platform. It helps promoting marketing content by letting employees share it.
- **Measure:** The part of the tools then help providing in-depth analytics of social data, like watching benchmarks and other performance metrics. It can also generate reports to support management decisions.

2.1.1 Describing Facebook metrics

To measure how successful a Facebook post is, it is important to have some metrics that scores a post. Facebook has more than 50 metrics, regarding the page, post, video, ad clicks, and more^{1 2 3 4}.

The most important metrics that will be used in this thesis are explained in this section.

- **Likes:** When posting a post on Facebook, users can click a like button, which indicates an interest in a post. Likes is referred to as the number of likes a single post will receive. Likes are quite common. See figure 2.1.2 at the top left red square.
- **Comments:** Upon seeing an interesting post, users can comment on this post. Comments refer to the number of comments created for a single post. Comments are not as common as likes. See figure 2.1.2 at the red square on the right hand side.

¹https://developers.facebook.com/docs/graph-api/reference/v2.2/object/comments?locale=da_DK

²https://developers.facebook.com/docs/graph-api/reference/v2.2/object/likes?locale=da_DK

³https://developers.facebook.com/docs/graph-api/reference/v2.2/insights?locale=da_DK

⁴https://developers.facebook.com/docs/graph-api/reference/v2.2/post?locale=da_DK

2.1. Falcon Socials platform



Figure 2.1.2 Example of a Roskilde post with 366 likes, 120 comments and 23 shares

- **Shares:** When seeing an interesting post, users can share that post. Sharing a post, takes the post and posts it on the users wall who pressed share. This indicates a strong interest in the post. Shares are less common than comments. See figure 2.1.2 at the red box to the bottom left.
- **Impressions:** Impressions are the number of times a user saw a post. Users can see the same post multiple times, for example in their news feed and later if their friend shares the same post. Both incidents will increment the overall impressions.
- **Reach:** Reach is the number of users who saw a specific post. This is also sometimes referred to as unique post impressions.
- **Story:** A story is an update showing some kind of interaction from a user's friends. A typical story is: "user1 commented on this", linked to a picture.
- **Consumptions:** Consumption is the number of clicks anywhere on the post, without generating a

story. This means clicking a link, starting a video, marking text - an interaction with the element.

2.2 Falcon Social: Predictive Publishing

The predictive analytics problem that this thesis will approach is a new feature to the publishing platform of Falcon Social. It was proposed by Falcon Social to create a tool that extends their current functionality in their platform.

While a user of the publishing platform is creating a post, a predictive publishing feature should come up with suggestions and prediction of post performance to improve the post.

Appendix A provides examples of such suggestions:

- "We can see you are writing about X. Consider posting this on a Friday as posts with this topics usually gets 18% more attention on Fridays."
- "You might want to throw in an image. People love images and usually retweets image posts 25% more than non-image posts"
- "We noticed that your last 12 posts have been around roughly the same topics. Time for a change maybe? People usually engage 31% more with posts that are different from the usual content."

The appendix also covers the full project proposal made by Falcon Social. They imagine such functionality, if possible, would provide great value to users of their platform. And that it would offer them a cutting edge feature advantage against their competitors.

The initial proposal describes a tool that can handle all popular social media (Facebook, Twitter, Google+, Instagram, etc). However, the scope for this thesis will be limited to Facebook only. This was a result of multiple factors:

- Handling all popular social media sites would require a lot of programming/data mining, taking up time that could be used for creating better models and a better website.
- Facebook has more insight metrics, e.g. consumptions and impressions compared to Twitter and Instagram, who currently lack those functionalities.
- Facebook posts in general have more text to analyse for natural language processing purposes such as sentiment analysis and lexical diversity.

2.3 The changing landscape of social media

Social networks are constantly changing at a rapid pace. In this section, examples of those changes are described.

From 2010-2012 Twitter had an increase in user growth of 100%, Facebook had a growth of 38% and LinkedIn had a growth of 60%. Snapchat was founded in 2011 and in May 2014, 700 million photos and videos per day were sent using Snapchat. From 2009-2012 people who uses social media in the age of 50-64 increased 8% point, but people in the age of 18-34 decreased 9% point. From 2011-2012 tablet ownership increased 12% point. From 2010-2012 adults who view video on their smartphone increased 11% point ⁵.

All these examples show that the landscape of social media is changing in lots of different ways. To summarise some trends:

- The different social media sites are changing in size and new ones are being created all the time.
- The people who use social media are changing.
- The way people view social media is changing.
- The things people view online are changing.

The social media sites themselves are also changing. They change business plans, design ⁶, features ⁷ and privacy settings ⁸.

As the scope of this report is Facebook, it is important to have a focus on the changes being made by Facebook. The algorithm that ranks and decides what shows up in a users news feed, consists of 100.000 weights ⁹, that is being changed from time to time ^{10 11}. All changes can be found at ¹². The changes mean that what worked when getting successful posts for companies one month ago might be outdated today.

The newest update for example factors in trending topics and looking at when people like or comment. When an update like this is applied to Facebook, a customer with a Facebook page needs to take the changes into account and maybe change their social media strategy accordingly.

In an interview with Helle Tyllesen [Helle Tyllesen, 2014], she also mentioned this change in the algorithms of the Facebook news feed. Helle gave an example of this: Getting as many likes as possible worked well for companies 2 years ago. This resulted in like-bait posts like the one seen in the figure below. Like-bait posts trick/compel users into liking a post. Now Facebook will decrease the exposure of posts like this ¹³.

⁵http://www.mediabistro.com/alltwitter/media-landscape_b37736

⁶<https://developers.facebook.com/blog/post/2013/03/07/get-to-know-the-new-design-for-news-feed/>

⁷<https://www.facebook.com/help/380049702048759>

⁸<http://cir.ca/news/facebook-privacy-policy-updates>

⁹<http://marketingland.com/edgerank-is-dead-facebooks-news-feed-algorithm-now-has-close-to-100k-weight-factors-55908>

¹⁰newsroom.fb.com/news/2014/09/news-feed-fyi-showing-more-timely-stories-from-friends-and-pages

¹¹<https://www.facebook.com/business/news/News-Feed-FYI-A-Window-Into-News-Feed>

¹²newsroom.fb.com/news/

¹³<http://mashable.com/2014/04/16/news-feed-changes/>



Figure 2.3.1 Example of a post that encourages like baiting

As the Facebook news feed is updated regularly, it is important that companies adapt to these changes. The algorithm/model created in this report has to also adapt to these changes of social media and news feed algorithm updates. It is not possible to fit a model and then expect this model to still work in six months after it has been fitted. The model has to be continuously updated according to all the changes that will happen in the eco system of social media.

2.4 Social media in Enterprise

Strategies within social media management vary across industries and size of corporations. While some companies just barely manage to publish a post on their favourite social media, others have employed social media managers to make sure certain key performance indicators (KPI) are achieved. An example of a KPI could be weekly reach. Of course, the last mentioned are typically common within larger organizations or companies whose product evolves around social media.

Companies also tend to budget for social media engagement by paying for a post to reach a broader audience. For instance, this allows a post to be injected onto a targeted audiences news feed on Facebook. Paid posts

are more successful in reaching a large number of potential customers, as this is what a customer pay for: Increased reach.

Reach is one way to measure if a post is successful, and one can argue that different businesses have different goals for social media posts. A post could also be aimed towards getting likes, comments, and/or shares - or a combination of those (this will be explained further in the next section). Wherever the focus is, the tools provided by Facebook (and soon Twitter) help companies to measure whether or not the goals (or KPI's) are achieved as well as constantly observe how the a company's social media page is performing.

Chapter 3

Related work

Contents

3.1	The Adobe Social platform	11
3.2	Kaggle competitions	12

With the continuous changes and increasing complexity in social media, it opens up the opportunity for companies to extend their current business models to include social media - and to have an actual social media strategy. A close competitor to Falcon Social is Adobe, who has developed a similar product. This section will cover the most relevant aspects of Adobe Social regarding this project. In addition, this section will also cover how machine learning is currently evolving by looking at competitive challenges within machine learning. Kaggle is a pioneer within machine learning competitions, and they have participants from all over the world. This also helps giving an insight to what problems have been solved using machine learning and predictive analytics.

3.1 The Adobe Social platform

Adobe has a social platform named Adobe Social. With Adobe Social comes lots of features like analytics, media optimization, and much more [Digital marketing | Adobe Marketing Cloud, 2014]. Adobe Social has in its latest update included predictive publishing. Predictive publishing is a new feature that helps marketers choose the right time to publish, and engagement predictions based on sentiment analysis, historic data and text mining [Adobe Social Unveils Predictive Publishing for Facebook | EON: Enhanced Online News, 2013]. In this way marketers can get an indication about how their content will perform. As of writing the feature only includes suggesting a different timing for a post [Adobe Social Unveils Predictive Publishing for Facebook | Business Wire, 2013].

3.2 Kaggle competitions

Kaggle is a site that hosts complex competitions within data science and collaborates with companies in order to solve real-life problems. Users compete against each other in creating the best solutions in reward of money, career opportunities and prestige.

There is another side of Kaggle: Their data-driven solutions targeting the energy industry to assist in increasing production while minimising its costs. This is achieved by looking at several big data sets and identify characteristics of acreage [*Energy Applications* | Kaggle, 2014] and optimisation possibilities.

Example competitions:

At the time of writing, 161 competitions have been hosted on Kaggle, and the topics of machine learning varies from optimisations, classification, text analysis and scoring problems.

Car Auction: Predict if a purchased car is a "kick"

This competition focuses on one of the biggest issues within auto dealership auction, where a purchased car sometimes have been tampered with or contains other issues that will prevent it from being sold to customers - a so-called "kick" purchase. Such purchases can be very expensive for a car dealer, so limiting such expenses is advantageous.

Prediction: Predict whether a mobile ad is going to be clicked or not.

Avazu, a leading advertising platform within cross-device advertising, has hosted a competition to predict whether a mobile ad will be clicked based on 11 months worth of data. The challenge also includes beating standard classification algorithms. The fields of the provided data contains information about the ad, such as banner position, information about the device, information about the app/site it has been shown on, and several other properties - most importantly a binary field determining if the advertisement has been clicked or not.

Chapter 4

Framework

Contents

4.1	Data Preparation	13
4.2	Exploration of Data	15
4.3	Feature engineering	19
4.4	Choosing what to predict	25
4.5	Choosing the right model	26
4.6	Evaluation	29
4.7	From machine learning to product in enterprise	35

This chapter will define a framework in which machine learning can be used regarding enterprise software development. There will be a special focus on predictive analytics.

The chapter will document the process all the way from data mining through analysis to how it can be implemented in a functioning company. It will also cover how the results can be evaluated.

4.1 Data Preparation

In order to be able to analyse a dataset and apply machine learning methods, it is necessary to prepare existing data. The discipline of Business Intelligence (BI) has defined this process as ETL - Extraction, Transformation and Load . BI has been used for decades, however mostly in enterprise, and its standards and definitions are well documented, such as in [Loshin, 2013]. Rather than reinventing the wheel and explaining a new process, ETL will cover the phase of making data available and present it in a meaningful format for future use.

4.1.1 Extraction

This part of the process grabs relevant data from available sources without any formatting whatsoever. Depending on the data sources, this step can prove to be the most challenging of the three, as it generates the foundation of future data quality. Normally during an extraction phase, it is important to ensure that no data is left behind (especially in enterprise or financial systems). In the domain of machine learning, missing an entry typically does not matter if the amount of data is sufficient so that it represents all aspects of the data. Especially if the amount of data to process is high.

4.1.2 Transformation

Transformation takes the extracted data and applies calculations and rules to the data. Additionally, this step also cleans the data (e.g. handling null values, missing data and units of measure).

Several other instances of data cleaning might also include:

- Decode/encode free-form mappings - i.e. codes that means something else than they suggest. This could be status codes or error codes. Or if codes have been implemented inconsistently across multiple platforms.
- Validate data - evaluate whether or not the data is correct, and if not, update it.
- Deriving new values - especially in machine learning this is a crucial step, where combinations of features can reveal important insights.
- Summarising values - this typically happens if there is a need for a row to have a count or sum of an item. For instance, the number of sales in a region or a store.

When this step is complete, the data should be ready to use in machine learning models and for faster queries - i.e. the data structure should also have been structured in a way that supports its future usage.

In general, transformation should aim towards satisfying the following criteria:

Accuracy: Describes how precise the data is and whether it is correct. Low accuracy could be influenced by performing systematic errors when the data is written to the database or random errors.

Completeness: A complete dataset is complete if all measured data is known. That is, if all columns and rows are filled with data and therefore contain no missing values.

Consistency: To what extent is the same data measured equivalent throughout the domain. For instance, if a customer is registered with two different ID's for the same subscription in a system. That would be an example of inconsistency, since it is unsure which of the entries that are correct.

Uniformity: Is the data measured with the same unit? Timestamps are an excellent example of this. If two

posts is published 1 pm and 3 pm in Denmark, but the timestamps are registered on a service located in USA and in Europe, which time zone should be used? Uniformity ensures that, for instance, all time zones are measured in UTC across the data regardless of its origin.

4.1.3 Load

Load is simply the discipline of loading the data to the end-target, such as a new database. This step should ensure data integrity - uniqueness, referential and not-null fields. This implies, for instance a social media post can not have more parents or exist multiple times.

In enterprise BI, data is often loaded to either a new database or a data warehouse.

4.2 Exploration of Data

In data analysis, it is key to get to know the data in detail. Through visualisations, plots and simple statistics most relationships can be identified. While also getting a somewhat intuitive feeling about the data set, exploration of data gives better understanding of how the data might behave and what to watch out for when trying to fit a model.

4.2.1 Outliers

Finding outliers is a great way of investigating if there are faulty or abnormal values in a dataset. Boxplots as well as minimum/maximum values can be helpful when detecting outliers. Outliers can occur by chance in a distribution, but sometimes it can be due to a wrong entry. This could happen when looking at GPS entries for a runner, and his speed is constantly around 15 km/h. If one entry is 50 km/h, then something most likely went wrong.

Also in the domain of social media, a post having 1 impression but 4 consumptions indicates that the impressions found might be wrong. Also, viral posts (i.e. posts that have for some reason gained extreme attention) are close to impossible to predict for. Examples of the applications of boxplots can be found in the visualisations section (section 4.2.2)

4.2.2 Visualisations

Choosing what to visualise should not be underestimated, especially when a data set is unfamiliar, and there is plenty of excellent literature on that subject. In this section, the most common (and simple) ways

are described.

Scatter plots

When searching for relationships between variables, possibly one of the easiest methods to use are scatter plots. Scatter plots will provide a quick overview of relationships between attributes - assuming the number of attributes and the data samples are not too high. In that case, scatter plots might be difficult to interpret.

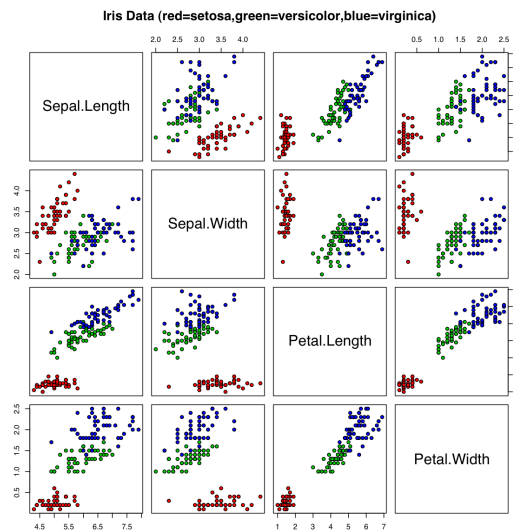


Figure 4.2.1 In the classic machine learning data set the Iris flower, scatter plots are an efficient way of getting an overview of the relationships between the features of the flower strains. Colouring each class helps identify possible correlations.

One data set that provides good insights about the data with scatter plots, is the Iris Flower data set [Anderson, 1936]. A quick glance at the scatter plots in figure 4.2.1 reveals seemingly high correlation between Petal Length and Petal Width as the points are separated nicely and it is easily imaginable that a straight line could be fitted here. On the other hand, Sepal Length and Sepal Width seems a bit more random, as the blue and green points are mixed together - in other words, they are harder to distinguish from each other.

Boxplots

Another visualization technique is boxplots. They are especially useful for revealing outliers and how the data is distributed. However, depending on how the boxplots are used, different conclusions/insights can be found. The boxplots on figure 4.2.2 provide an overview of the attributes in the iris data set.

The information gained from visualisations can lead to sudden insights that can be used at a later point. Looking at figure 4.2.3 it is possible to determine some intervals for the Petal Length of an Iris flower. Here the boxplots are visualised for each class rather than combined.

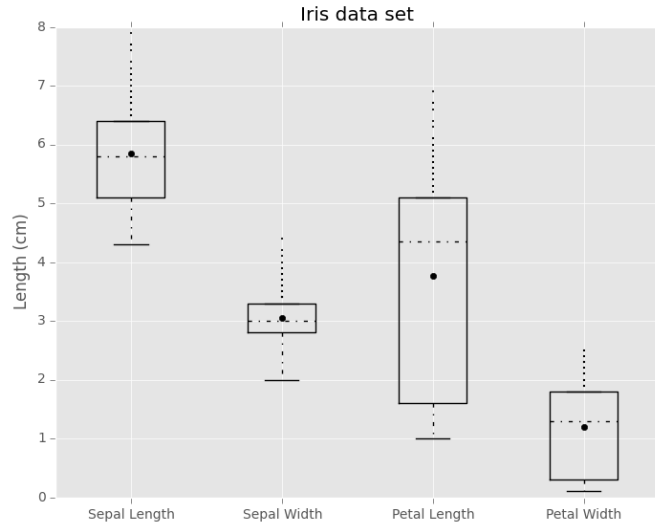


Figure 4.2.2 The boxplots visualise the attributes of The Iris Flower data set and makes it possible to identify potential outliers in the data.

This section about visualisation only covered some of the simple plots available. Plots such as histograms, 3D-plots, and, perhaps even pie charts are viable options as well. The boundaries of visualisations lies within the imagination of the creator, as creative visualisations can lead to interesting insights.

4.2.3 Correlations

In order to formulate relationships between features of the data set, correlations can be calculated. In order to do so, a correlation coefficient matrix can be calculated. First, the concept of covariance must be introduced to grasp the concept of correlation coefficient matrices:

$$C = cov(x, y) = \mathbb{E}[(x - \bar{x})(y - \bar{y})] \quad (4.2.1)$$

Where \mathbb{E} is the expected value of a number - or simply just the mean. For the values:

$$X = (x_1, x_2, \dots, x_N)^T \quad (4.2.2)$$

the covariance matrix the consists of the element C_{ij} in range of N. Therefore, the covariance of x_i and x_j and C_{ii} is the covariance of x_i and x_i . Since attribute values normally vary in data range, it generally makes

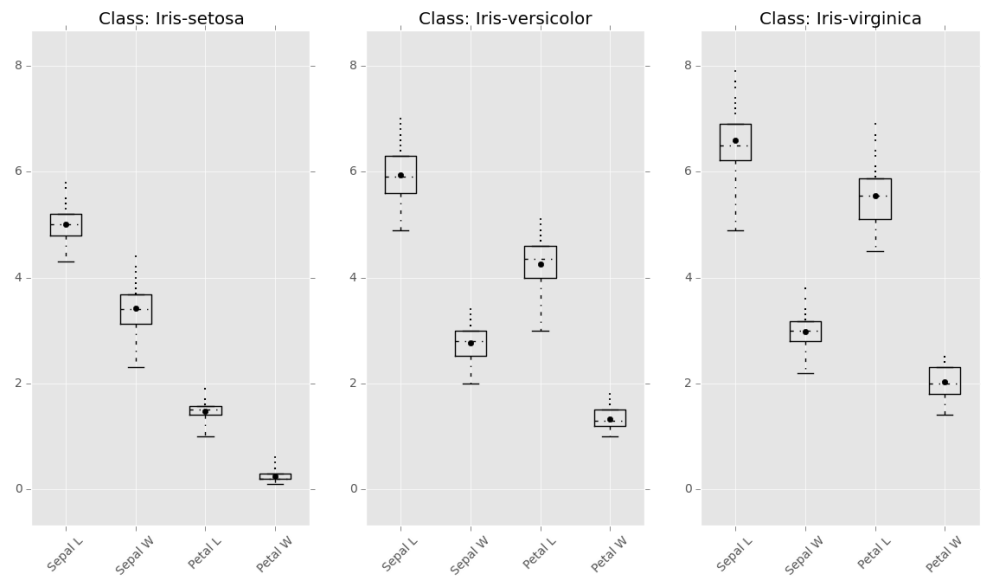


Figure 4.2.3 This combination of boxplots assist in determining relationships between types of flowers and its attributes. For instance, the petal length seems to be either between 1-2 cm or between 3-7 cm. This knowledge may become useful when trying to classify the data, and only a minimum of visualisation was actually needed to identify this.

sense to normalise the covariance matrix (the correlation coefficient matrix) where the element P then is:

$$P_{ij} = \frac{C_{ij}}{\sqrt{C_{ij} \cdot C_{jj}}} \quad (4.2.3)$$

If the The Iris Flower data set again is considered and the scatter plots in figure 4.2.1 are recalled, the correlation coefficient matrix is then listed in table 4.2.1.

Correlations	Sepal L	Sepal W	Petal L	Petal W
Sepal L	1	-0.11	0.87	0.82
Sepal W	-0.11	1	-0.42	-0.36
Petal L	0.87	-0.42	1	0.96
Petal W	0.82	-0.36	0.96	1

Table 4.2.1 Correlations between the attributes in the Iris data set. Especially, there seems to be high correlation between the Petal length and width. Note that the matrix is symmetric.

As discussed in section 4.2.2, Petal Width and Length seems to correlate well with each other. It can also be seen that the Sepal Length seems to be (positively) correlated with the Petal Width and Length (0.82 and 0.87) respectively.

4.3 Feature engineering

A feature is the specification of an attribute and its value [Ron Kohavi, 1998]. It is an attribute that is meaningful regarding the problem that is being solved. An example of a feature could be the length of the text in a Facebook post.

Feature engineering is the process of getting the most out of data for a predictive model to work with. Feature engineering takes raw data and transforms the data into meaningful features, that represent the underlying problem being solved [Jason Brownlee, 2014]. It is an exploratory process where new features are created and evaluated against each other.

Feature extraction is a subset of feature engineering where new features are found and created. This can be done in two ways - with a user centric approach or a data centric approach. The methods are explained in the rest of the section.

4.3.1 User centric feature creation

A user centric approach of feature creation is one where the data scientist manually creates the features. The data scientist analyses and thinks about the data, to find the underlying problems and expose them in the form of features. An example could be trying to predict the number of sold computers in an electronic store. One might think time of day has an impact regarding the number of sold computers, therefore a feature called time of day is created and evaluated against number of sold computers.

4.3.2 Data centric feature creation

Data centric feature creation is the process of creating new features using data with statistical methods and simple mathematics.

4.3.2.1 Principal Component Analysis

A widely used method for feature extraction is Principal Component Analysis (PCA). PCA works by projecting a high dimensional data set into a specific number of dimensions called components. The components are sorted so that the first component contains the most of the variability of the data set, the second component contains the second most variability and so on. The different components are orthogonal to each other.

PCA can be divided into 3 steps:

1. Firstly subtracting the mean from each attribute.

2. With the mean subtracted from each attribute, calculate the covariance matrix.
3. From the covariance matrix, the eigenvectors and eigenvalues are extracted.

The eigenvectors of the covariance matrix is the principal components, also called the loadings. The eigenvalues explain how much of the variance is contained within one component. To finish PCA, order the eigenvalues by size and get the corresponding eigenvector. Then project your original data onto the eigenvector.

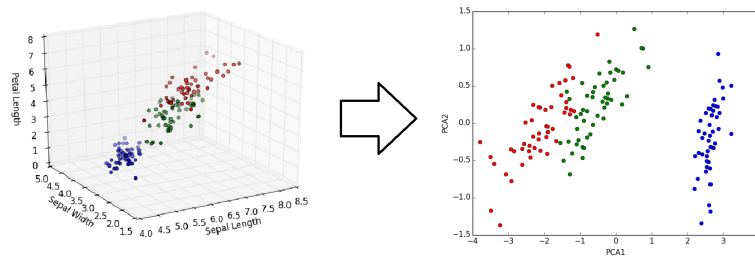


Figure 4.3.1 Example of a dataset being projected onto the two components that contain the most of the variation

To extract features, select a component, and look at the attribution from each attribute in the component. This can reveal features that would otherwise be difficult to find, and it can be seen how the first two principal components split the data nicely.

4.3.2.2 Natural Language Processing

The discipline of Natural Language Processing (NLP) spans over a variety of topics, and can help create additional features that are not always present in a data set. Often, a NLP process can be computationally heavy and mathematically complicated. So performing NLP on a data set might be preferable before fitting a model. This will help standardise the data and structure it conveniently, for instance in a bag-of-words representation.

Below are some examples of NLP than can be used in predictive publishing, and which will also be used later in this project:

Stemming, lemmatisation and stop words

Perhaps the most widely used method in the NLP toolbox is stemming and removal of stop words (words that provide no meaning). Stemming reduces a word to its root form - for example: "fishing", "fisher" and "fished" might be reduced to "fish". This reduces the overall vocabulary and therefore reduces the complexity of a corpus (a large, structured collection of documents or texts). Stemming does not take into account words that mean the same - for example, "happy" and "glad" basically means the same thing, but is viewed as different words. So while stemming tries to crop endings of a word in a heuristic process, lemmatisation

tries the same reduced a word to its base form - better known as a lemma. Therefore, when looking at the word "saw", stemming may return "s" while lemmatisation returns either "see" or "saw" based on the semantic context (the word being a noun or a verb in the context).

In addition to stemming and lemmatisation, it is also useful to remove words shorter than two characters and special characters. If the text contains links, it can be useful to remove them as they do not provide semantic meaning. Keeping track that a text contained a link can be useful, and might be used as a feature if it is relevant to the machine learning problem.

POS Tagging

POS tagging, short for part-of-speech tagging, is a method to identify words into categories of nouns, verbs, adjectives, etc. Splitting words into their right segment of speech can help to distinguish words of several meanings from each other (disambiguate homonyms). For instance, the sentence: "I like to fish fish"

NLTK has a library which performs the POS tagging using the Brown corpus¹. An example output could be:

```
>>> text = tokenize.word_tokenize("I like to fish fish")
>>> nltk.pos_tag(text)
[('I', 'PRP'), ('like', 'VBP'), ('to', 'TO'), ('fish', 'VB'), ('fish', 'JJ')]
```

Here "fish" is both tagged as VB (verb) and JJ (adjective).

Sentiment Analysis

Sentiment analysis tries to find the opinion/attitude of some text in order to identify the emotional state of a writer. It is widely used by companies who want to track the moods of their reviewers or articles written about them. One of the easiest ways to make a sentiment analysis is using a word list with an array of words and a rating that ranges from a negative to positive value, for instance -3 to 3. The rating is done by a human and is considered a ground truth for the analyser. The analyser then looks at a text and matches the words of the text with the word list. Consider the sentence "I am happy, but tired". "Happy" might be rated very positive with the rating 3, and tired is a little negative and might be rated -1. The overall sentiment of the sentence is then 2. Additionally, the use of emoticons (smileys) is also relevant to this approach.

A little more advanced way of approaching sentiment analysis is through the use of a machine learning algorithm such as a Naïve Bayes (NB) classifier[Tan et al., 2009]. NB can be trained on documents which are already classified with their sentiment - either as labels (positive, negative) or in a range from -1 to 1. There are many variations of the implementation of the NB classifier, and the Bernoulli NB classifier tends to perform well on shorter documents and a large corpus - in other words, ideal for social media posts such as Facebook and Twitter. For a positive and negative class, the classifier is given as:

$$P(c_i|d) = \frac{\prod P(x_i|c_j) \cdot P(c_j)}{P(d)} \quad (4.3.1)$$

Here x_i is the individual words of a document, c_j is the class label and d is the document.

¹<http://www.scs.leeds.ac.uk/ccalas/tagsets/brown.html>

There exist a broad number of implementations of sentiment analysis. Most worth mentioning might be NLTK² (for the Python programming language) which not only is excellent for sentiment analysis, but also for other NLP challenges.

Topic Modelling

Regarding topic modelling, methods such as Latent Dirichlet Allocation (LDA), Latent Semantic Indexing (LSI) and Non-Negative Matrix Factorisation (NNMF) are the most effective and widely accepted. While LDA is a probabilistic generative model (further explained in section 5.5), the other two are based on linear algebra.

- **LDA:** Generates a mixture of topics where each topic has the same of amount of words related, and a probability. The same word can exist in more topics, hereby overcoming the challenge of when a word have more meanings. Gensim (again for Python) provides a good implementation of LDA.
- **LSI:** LSI is a method that generates relationships between unstructured collections of text. It is based on Singular Value Decomposition (SVD), that uses eigenvectors to reduce dimensionality.
- **NNMF:** NNMF is based on matrix factorisation[Sra and Dhillon, 2006] and have, among plenty of applications, been used to cluster Wikipedia articles and scientific journals.

Lexical Diversity

It is sometimes useful to investigate the vocabulary of a writer or text. Lexical diversity is a measure that describes the variety of words of a writer. Someone who uses the same words constantly in a text instead of mixing it up is not considered lexically diverse, while someone who keeps using different words for the same thing ("approach", "strategy", "plan" and "program") is considered lexically diverse. A simple measure for lexical diversity can be the number of unique words divided by the total number of words in the text. A high score then indicates that a writer is lexically diverse.

Other

So far only a handful of NLP techniques have been mentioned, and it comes down to the specific problem of interest that the framework is used to solve. In general, regardless of the task at hand, it is useful to do some form of text processing. Especially dealing with content from the web such as HTML sites and RSS feeds. Along with the previously mentioned sentiment analysis implementation, NLTK provides tools for tokenisation and stemming.

Benefits of NLP

This section explained that NLP has a lot of application and opportunities in predictive analytics - and in machine learning in general. NLP is also a way of organising and analysing text in order to achieve insights about some text or whole collections of text.

²<http://www.nltk.org/>

4.3.3 Feature selection

Feature selection is a crucial part of feature engineering, where the number of features are decreased to contain the most informative features regarding what is being predicted. Feature selection addresses the problem of overfitting, performance of model and storage requirements [Zhai et al., 2012].

Feature selection can be done in 3 ways: filter methods, wrapper methods, and embedded methods.

4.3.3.1 Filter methods

Filter methods are independent to the learning method. They return a relevance index: $J(S|D)$. Given data D , what is the relevancy of feature subset S , Regarding a problem Y [Guyon and Steve Gunn, 2007]. The features are evaluated using for example mutual information [Zhai et al., 2012] regarding classification. The least relevant features regarding Y are filtered out.

4.3.3.2 Wrapper methods

Wrapper methods use a machine learning classifier to predict relative usefulness of subsets of features. To find the subsets of features is a problem that is NP-hard [Guyon and Elisseeff, 2003]. Therefore smart search strategies has been created such as best-first and simulated annealing [Guyon and Elisseeff, 2003].

4.3.3.3 Embedded methods

Embedded methods do feature selection as a part of the model creation. They embed the feature selection when fitting a model. An example is using the Lasso regression technique, and then select the features that have a non zero regression coefficient.

4.3.3.4 Contextual features

In machine learning there are three kinds of features [Turney, n.d.]:

- Primary features. These features are features that can predict a class when considered in isolation.
- Context features. These features need to be considered together with other features in order to classify an object.
- Irrelevant features. These features are not relevant for classifying a class.

Context based features have been shown to improve performance in machine learning models [Brezillon and Gonzales, 2014]. Some examples of context regarding machine learning datasets are light conditions

when doing image classification, accent when doing speech recognition and a patient's identity regarding heart disease diagnosis [Turney, n.d.].

4.3.3.5 Categorical features

Categorical features are objects which has the property to either belong to a specific category or not belonging to a category. This type of feature is also called a nominal feature. Examples of this could be a city name, media type or country.

To give a concrete example of 6 objects with the country names seen in table 4.3.1. Simply assigning a city name to a numerical value and then doing prediction is problematic. For example Copenhagen: 1, Stockholm: 2, New York: 3, London: 4, see table 4.3.2. If this is fed into a learning algorithm, the algorithm will assume that London is a larger value than Copenhagen. This does not make a lot of sense and is not the intention of encoding the feature.

Therefore one-hot-encoding is introduced. This technique divides every unique value of a feature into its own boolean feature as is seen in 4.3.3. Now a feature can be easily applied to a learning algorithm. The only problem with this approach is that the number of features rapidly becomes very large, if the categorical feature has a lot of values.

	City name
object 1	Copenhagen
object 2	Stockholm
object 3	New York
object 4	London
object 5	London
object 6	Copenhagen

Table 4.3.1 Posts with categorical features

	City name
object 1	0
object 2	1
object 3	2
object 4	3
object 5	3
object 6	0

Table 4.3.2 Posts with categorical feature, encoded simple

	City name = Copenhagen	City name = Stockholm	City name = New York	City name = London
object 1	1	0	0	0
object 2	0	1	0	0
object 3	0	0	1	0
object 4	0	0	0	1
object 5	0	0	0	1
object 6	1	0	0	0

Table 4.3.3 Posts with categorical feature, one hot encoding

4.4 Choosing what to predict

In machine learning the dependent variable is the output also known as the target variable. The dependent variable can be discrete or continuous. Examples of a dependent variable could be: If a person will develop cancer or not, if an ad will be clicked or not, price of a house, age of a person or temperature of a day.

The dependent variable tries to solve a specific use case, but the mapping between use case and prediction might not map directly. The examples of dependent variables given above are easily measured, but what if the use case is to predict the successfulness of an online ad campaign. The dependent variable could be lots of different variables: People who saw the ad, people who clicked the ad, people who bought a product after seeing the ad, or maybe a combination of multiple variables. Successfulness is subjective and will vary from each situation. The important thing is to go back to the use case and examine the problem further. What makes an online ad campaign successful? Try to interview people who have domain knowledge about the problem in order to gain deeper insights. Maybe the use case needs to be broken into parts, and different classifiers created for each part of the use case.

Choosing the right thing to predict should be done carefully, and just applying ones favourite machine learning model on the first metric that comes to mind may not yield optimal results.

4.5 Choosing the right model

When the dependent variable has been chosen, a model can be fitted to the data. Plenty of well-performing machine learning models have already been implemented and are publicly available. Choosing the right model and its optimal parameters should be done smartly and thoroughly.

The model depends on the dependent variable. If the dependent variable is a regression problem, regression models must be applied to the data, if the dependent variable is a classification problem, a classification model must be applied to the data. There are multiple models to try and for each model there are several parameters to set. The different models and their parameters will be described further in chapter 5.

The goal of choosing the right model is to pick the model that predicts new data best.

4.5.1 Setting the right parameters

To find the optimal parameters for a model, grid search can be performed. Grid search is a part of hyperparameter optimization. Hyperparameters are the parameters that can be tweaked in the model [Bergstra and Bengio, 2012]. An example of a hyperparameter is the depth in a tree classifier or the number of neighbours in K-nearest neighbours. Grid search works by initially specifying a set of parameters to be tested for a dataset. Grid search then works by simply fitting a model to the data and testing on held-out data [Sklearn grid search, 2014]. As every possible combination of configurations must be applied to grid search, it is quite computationally heavy.

4.5.2 Describing the training, validation and test set

When the right model and the right parameters for the model has been chosen, the data must be fitted to a model in order to predict new data. In the fitting process, the machine learning model learns from historical data.

When doing machine learning it is important to test how well a model is performing. This is done by splitting the available data, by simply dividing the dataset into three parts. A reasonable division of training, validation, and test set is 50%, 30% and 20% respectively . The exact distribution varies, but usually the training set is the largest, while the validation and test set are closer in size.

The training set is used to train/fit a model. This is where the model learns the underlying tendency of the data.

After the model has been trained it is appropriate to evaluate the model, with some new data in order to find an optimal model, which minimises both training and validation error. Here the new data is the validation set. For each data point in the validation set, the model predicts the dependent variable and this predicted value is compared to the real value of the dependent variable.

The splitting of the data set should be done accordingly regarding the nature of the problem that is being solved. For example for prediction, a good strategy is to validate with data that is newer than the training data. This gives a realistic validation error for a model, that needs to predict object into the future.

When a model has been chosen based on the lowest validation error, it still leaves some bias regarding the validation set, since the model is optimised to that set. This is where the test set is brought in, and the models final score is given using the predictions of the test set. It is important that the test set has remained unseen to the model during both the fitting and selection process. There cannot be any data leaking from the training or validation set into the test set.

4.5.2.1 Cross-validation

Training using a specific part of the data and predicting using another part can result in test errors that does not reflect the data.

Cross-validation is the process of dividing and testing a data set, but on different parts of the data. This process will ensure that the performance on the test set, will reflect the performance on the data set overall. There are different ways to do cross validation, which are listed here:

- **Holdout:** The simplest part of cross-validation. Choose a part of the the data set to be holdout. Train on the data that is not held out. Test on the data that has been held out.
- **Leave-one-out:** Leave-one-out takes N number of observations. The method holds out one observation which is used for testing, the remaining observations $N - 1$ is used for training.
- **K-fold:** K-fold cross-validation works by randomly splitting the dataset into k folds. For every fold, the data is split into a test and a training set, that is mutually exclusive from the previous folds. A three fold cross-validation method is showed in figure 4.5.2.

4.5.3 Problem of overfitting

Overfitting is the problem of fitting data too precisely. Instead of modelling the larger tendency of the data, the model fits exactly every point. This is illustrated on the right hand side of figure 4.5.3. Underfitting is the opposite problem of not modelling the underlying tendency of the data. The left hand side of figure 4.5.3 shows this.

³Slides from DTU course 02450

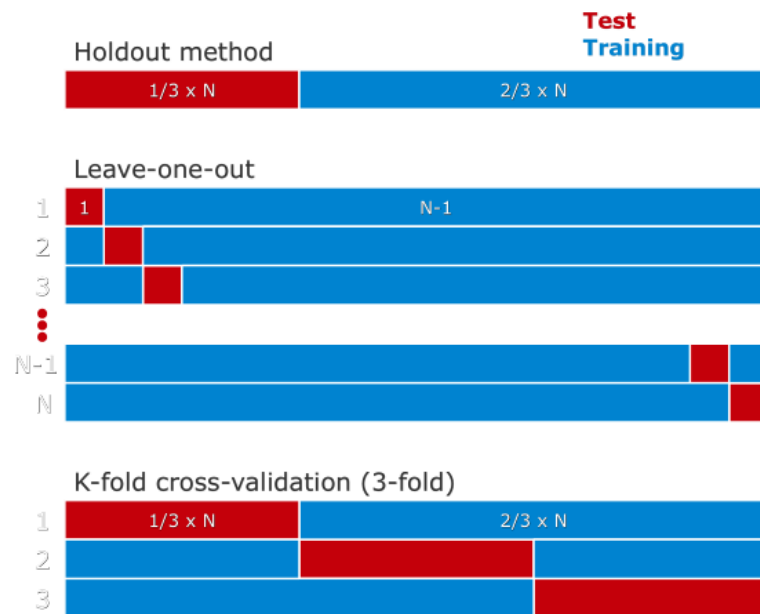
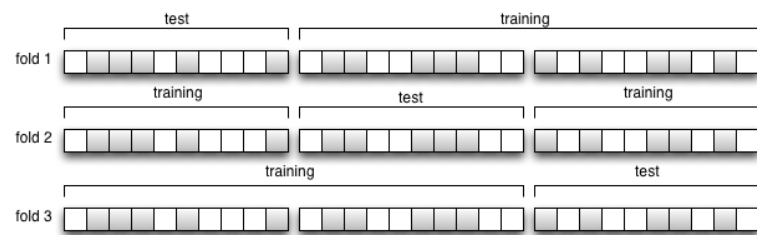
Figure 4.5.1 Figure showing the different cross-validation methods ³

Figure 4.5.2 Threefold cross validation

Overfitting can happen for a number of reasons[Hawkins, 2004]:

1. A model is too flexible or too complex.
2. A model uses too many features.
3. There are few data points.

The ratio between complexity of a model and the number of training data should not be too high. Figure 4.5.3 to the left shows a model that is too simple. Both the test error and the training error will be large, as the model has not learned anything about the data. This can also be seen in figure 4.5.4 to the left. This area is characterised by low variance, but high bias. The low variance comes from the model not being very complex, and therefore not varying too much. Bias measures how much in general the predicted value is

from the expected value. When model complexity is low, the model will keep predicting the same thing no matter the data, resulting in high bias.

As the model complexity increases, the training error and the test error both decrease as the model learns about the underlying tendencies of the data. This is the middle part of figure 4.5.3 and 4.5.4.

If the model complexity keeps increasing, the model starts overfitting. This means that the test error will increase again, because the model starts modelling the training data too precisely, so when tested with new data it will only have learned the tendency of the training data and not the test data. This also decreases the training error. This can be seen to the right in figure 4.5.3 and 4.5.4. This part is characterized by high variance from the large model complexity and low bias, as the model will be flexible to fit the data.

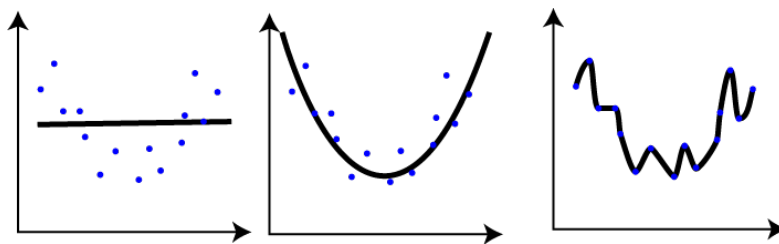


Figure 4.5.3 An example of first underfitting, optimal fitting and overfitting

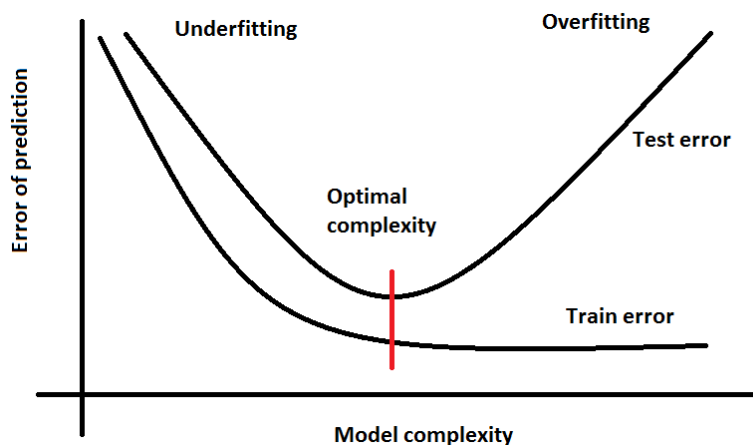


Figure 4.5.4 Plot that shows training error and test error, when model complexity increases. The red line represents the optimal model complexity.

4.6 Evaluation

Evaluating fitted models and performing sanity checks are crucial parts of predictive analytics. This section will go through the use of dummy models and how to evaluate regression and classification results. Finally, test statistics are introduced to choose between models that performs closely on validation sets.

4.6.1 Dummy models

Sometimes a sanity check of a model - regardless of classification and regression - can be performed by comparing it to a simple model (a so-called dummy model). A dummy model uses predicting strategies that are very simple, and can help determine if a model performs better than random/other simple strategies. The guesses vary a bit from classification and regression, but the general concept behind it remains the same:

- **Most frequent value:** The value which is most frequently present in the dataset
- **Mean/median:** A guess on the mean/median of the data set.
- **Uniform:** Used in classification where the model guesses uniformly across the array of classes in the data.
- **Constant:** The model performs a constant guess.
- **Stratified:** In classification, guessing randomly among the classes in the data while respecting the distribution of the classes.

4.6.2 Classification models

In classification, success (and failure) in prediction is straightforward to measure. Either the model's prediction is correct, otherwise incorrect. However, there are also some measures of a model's performance that allow more detailed insights. In a binary classification problem, a model can guess incorrectly in two ways:

- **False Positive (FP):** The model predicts a class (a cat) while it was actually a dog.
- **False Negative (FN):** The model wrongly mistakes a true class (dog) for being a cat.

And can similarly be correct in the following ways:

- **True Positive (TP):** Correctly predicted class. For instance, "dog" is predicted as "dog".
- **True Negative (TN):** Correctly predicted class, which is not "dog". If the class is "cat" and is labelled "cat", then it is a true negative.

Confusion Matrix:

A confusion matrix is a way of visualising the predictions of classification models, so that it is possible to distinguish actual labels against predicted labels. Below is a confusion matrix for the previously used The Iris Flower data set, where the species (setosa, versicolor, and virginica) are the classes:

		Iris		
		0	1	2
True class	0	13	0	0
	1	0	15	1
	2	0	0	9
		Predicted		

Table 4.6.1 Confusion matrix for The Iris Flower data set. The diagonal represents true predicted values. Here class 0, 1 and 2 is setosa, versicolor, and virginica respectively. Only one place did the model wrongly predict the flower species.

The diagonal of the matrix reveals the true predicted values - where the model predicted the correct species of the flower. Overall the model only classifies one flower incorrectly, which is impressive performance. This is probably one of the reasons why the Iris data set is a popular illustrative data set.

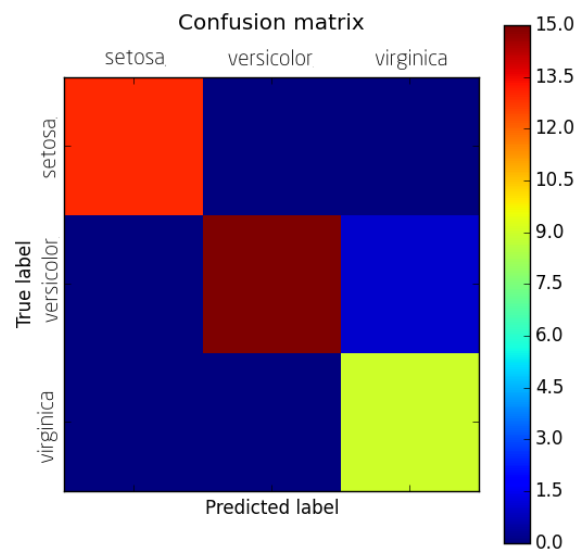


Figure 4.6.1 Visualisation of the confusion matrix for the Iris data set

The predictions regarding the species versicolor can be evaluated in the following way:

- True Positive: 15 (Correct classifications of versicolor)
- False Positive: 0 (Incorrect labelled as versicolor)
- False Negative: 1 (Versicolor which was wrongly labelled as a different flower)
- True Negative: 22 (Correct non-versicolor classifications)

Tables can sometimes be hard to relate to, so a convenient alternative can be made using a coloured version of the table, which is seen in figure 4.6.1.

4.6.3 Regression models

Evaluation of regression models is normally done by using additive measures of residual errors. Here the residuals are given as

$$r_i = y_i - \hat{y}_i \quad (4.6.1)$$

Where y_i is the i 'th test value of the test set \vec{y}_t and \hat{y}_i is the i 'th value of \hat{y} . Not all methods of regression evaluation summarises the accuracy of a regression model, and it is sometimes useful to have several measures of evaluation. Below are some of the most common measures, but other approaches [Rosset et al., 2006] have also been taken to, for instance, use a ranking based approach for performance estimation.

Mean Squared Error (MSE): MSE uses the average of the squares of errors on the predicted values, and is defined as:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (4.6.2)$$

While MSE can be a good evaluation factor for regression model, it is heavily influenced by outliers, as the difference is squared and therefore punishing outliers more.

Mean Absolute Error (MAE): MAE measures the error of a prediction, where the absolute error collected additively and averaged:

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (4.6.3)$$

Coefficient of Determination (R^2): This is the ratio between the residual sum of squares (RSS) and the total sum of squares (TSS). R^2 is then:

$$R^2 = 1 - \frac{RSS}{TSS} \quad (4.6.4)$$

Where TSS is the total sum of squares:

$$TSS = \sum_{i=1}^N (y_i - \bar{y})^2 \quad (4.6.5)$$

and RSS is mean difference from each value in the test set squared:

$$RSS = \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (4.6.6)$$

Mean Absolute Percentage Error (MAPE): Much like MAE, this measures the error of a prediction.

However, MAPE, as the name suggests, provides the error as a percentage, and thus giving a straight forward metric. It is given by:

$$MAPE = \frac{1}{N} \sum_{i=1}^N \frac{y_i - \hat{y}_i}{y_i} \quad (4.6.7)$$

Median Absolute Percentage Error (MdAPE): Instead of providing the mean of the percentage error, the median is provided instead. This is quite helpful when data has lots of outliers, as the median is robust regarding outliers [Armstrong and Collopy, 1992].

$$MdAPE = Median\left(\frac{y - \hat{y}}{y}\right) \quad (4.6.8)$$

Explained Variance (EV): The Explained Variance is a simple way of determining how much variance is explained by a model.

$$EV = 1 - \frac{Var(y) - \hat{y}}{Var(y)} \quad (4.6.9)$$

4.6.4 Performance evaluation

After selecting the best performing model using the above mentioned evaluation metrics, the model should be evaluated using a test set. If there are more than one model, it should be considered to do some test statistics to compare the models if their performance is close on the test set. Depending on the size of the test set, the test can vary, but for this section it is assumed the test set is large (above 30).

4.6.4.1 Classification Models

When evaluating the performance of classification models, it is likely that some may perform equally well - or close. A measure to compare two models can be done using the Welch's t-test [Welch, 1947], which is a variation of the more generally known Student's t-test. Welch's t-test takes into account that two sample populations might have unequal variances.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (4.6.10)$$

Since Welch's t-test requires two populations it is applicable when evaluating performance where cross-validation for a model has been used.

When, cross-validation is not a viable option for testing, a z-test can be used. A z-test is especially appropriate for populations larger than 30 and when the variance is unknown. Depending on the null-hypothesis, the z-test is given as

$$z = \frac{(\hat{p}_1 - \hat{p}_2) - d_0}{\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}}} \quad (4.6.11)$$

where the null-hypothesis states that sample mean of differences d_0 is greater than zero (suggesting that the one model is better than the other.)

Equation 4.6.11 is then a viable way of comparing two classification models - under the assumption that for a high enough number of training samples, the values can be approximated to follow a normal distribution.

4.6.4.2 Regression Models

Comparison of two regression models is fundamentally not as straightforward as comparing two classification models. In classification there is a specific measure of the correctly classified labels. That makes it easy to compute an error statistic simply by looking at relationship between the wrongly classified labels and the total number for samples.

First of all, since regression by its definition is continuous, measuring the distinction between a correct and wrong prediction must be based on an evaluation metric. Section 4.6.3 covers metrics such as MSE and R^2 - both of which are squared and can therefore not follow a normal distribution.

Looking at two models M_1 and M_2 , they are both trained and evaluated using the same test set, and chosen based on the lowest MSE. Their errors are then:

$$\hat{e}_i = \hat{y}_p - \hat{y}_t \quad (4.6.12)$$

where \hat{y}_p is the prediction for model i, and \hat{y}_t is the correct values of the test set.

Let \hat{e}_1 and \hat{e}_2 be the errors of M_1 and M_2 respectively. The normal approach here will be to do a t-test of the two error vectors. However, since they are chosen from MSE (which is squared) their distributions are not considered to follow the normal distribution. The difference between the two vectors can instead be found:

$$e = \hat{e}_1 - \hat{e}_2 \quad (4.6.13)$$

It can then be assumed that if the models performed equally well, the mean of e would be 0 and e will, conveniently, follow a normal distribution around 0. Now a t-test can be made for e with the null hypothesis that the mean μ is 0:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} \quad (4.6.14)$$

where n is the number of samples and s is the variation of the data.

Even though the evaluation metrics are given for both regression and classification, one should always look at the results critically. Sometimes a metric such as MSE might over- and underfit and still give a good results.

4.7 From machine learning to product in enterprise

Developing a product in enterprise is a task that can seem difficult to begin with. This section will discuss how a machine learning problem can be applied to an enterprise environment.

4.7.1 Enterprise challenge

The first thing to figure out, is the challenge of the enterprise. Here are some questions to ask in order to explore the problem further.

- What exactly is the their challenge?
- What makes them want to start a project?
- What do they do know?
- What have they tried before?

With these questions it should be possible to map out exactly what the challenge is and how far an enterprise is in their process of solving the challenge.

What comes out of this part is a goal for the project. This goal should be a short description that answers the following question: Why does this project need to be implemented? The challenge for this project can be seen in A. Regarding this specific challenge, the following goal was created: Help Falcon Social's customers create Facebook posts, in order to improve social engagement.

4.7.2 Enterprise requirements

It is important to gather requirements from the enterprise, to make sure the solution will work as intended and will meet their needs. It is important to talk about functional requirements. These include things like minimal response times, security and performance load.

Technical requirements also need to be discussed, such as how should the solution be developed. Should specific technologies be used?

4.7.3 What should be delivered?

Make sure that it is very clear what should be delivered to the enterprise and how. Is it a NoSQL based database, a MySQL database, a RESTful web service, a SOAP based web service, an excel file or a website. All this needs to be as clear as possible. Just deciding on a RESTful web service is not enough. Discuss what the interface between the solution and the enterprise is consisting of. What does the enterprise need to provide, when calling a web service and what should be returned. Especially agree on a data structure, so that dates have the same format, and attributes have the same types.

4.7.4 Where can machine learning help overcome the challenge?

As this is a framework for machine learning, this part will only focus on machine learning techniques as a solution for the challenge. Machine learning can not solve all problems that an enterprise faces, so do not use it unless it makes sense regarding the enterprise's problem.

The kind of problems machine learning can solve are classification, prediction, clustering and dimensionality reduction. The next section will discuss this further.

4.7.4.1 What kind of machine learning should be used?

Machine learning can be divided roughly into two groups. Supervised and unsupervised learning.

Supervised learning: This type of learning is characterised by having an object and its desired output. In supervised learning, the model learns from the inputs and outputs. If the output has one or more classes the type of learning is called classification, if the output is continuous, the learning type is called regression. The goal of machine learning is then to create a model that maps inputs to outputs.

Supervised learning should be used when the output is known and the output is what should be predicted.

Unsupervised learning: This type of learning is characterised by the model figuring out what underlying structures/clusters are in the data. The model gets no input and output, but only the inputs.

Unsupervised learning should be used when the output is not known, but clusters of objects is sought.

4.7.5 Who are the users of the system and what should they be able to do?

At first, the users must be identified. Will the system be used by the employees at the enterprise? Or are the users actually the customers of the enterprise? The following is taken from the Microsoft Solutions Framework (MSF) for Agile Software Development[Guang-yong, n.d.]: "To help this process of figuring out who the users are, create personas, that describe typical skills, tasks and backgrounds for a set of users".

With a set of typical users, create a list of typical use cases. A use case contains what the users should be able to do with system. It can be quite helpful to also prioritize the use cases, so that it is clear what is must have features and which and nice to have. Defining the use cases will help when developing the solution in an agile process.

4.7.6 Solution development in an agile process

The last thing to do is to create a MVP (minimal viable product). This is a product that has all the essential features and nothing more. Typically this will be the use cases with the highest priority. Let the enterprise test the MVP and provide feedback based on the tests. With this feedback, revise the solution while also adding more functionality. This process should be done in an iterative process, where the product gets more features and is customized to the enterprises needs. The process can be seen in figure 4.7.1.

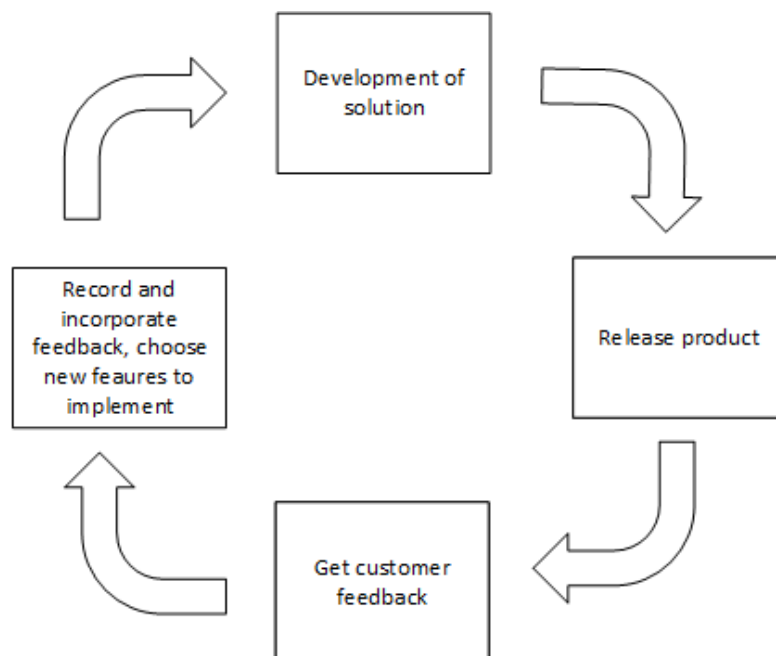


Figure 4.7.1 Figure of the iterative process

Chapter 5

Methods

Contents

5.1	Decision Trees	39
5.2	Random Forest	41
5.3	Nearest neighbours	42
5.4	Agglomerative hierarchical clustering	44
5.5	Latent Dirichlet Allocation	45

This section will describe the different machine learning models that is used for both predicting new Facebook posts, but also for doing topic modelling.

5.1 Decision Trees

Decision trees is a machine learning method that is used for both regression and classification. Decision trees works by creating a tree hierarchy of rules, based on the features of the dataset.

More specifically the decision tree model works by splitting the data into smaller subparts. The way the decision tree model does this, is by selecting a split that divides the data into the most pure subparts in regard to the dependent variable. This ensures that uncertainty regarding classification of an object is minimized. Calculating the most pure split can be done in a couple of different ways:

$$Entropy(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t) \quad (5.1.1)$$

$$Gini(t) = 1 - \sum_{i=0}^{c-1} (p(i|t))^2 \quad (5.1.2)$$

$$Class.error(t) = 1 - \max_i p(i|t) \quad (5.1.3)$$

To compute the actual gain of purity

$$\Delta = I(parent) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j) \quad (5.1.4)$$

A way of selecting the best split is by creating a lot of splits and simply selecting the split that reduces the impurity the most. The algorithm starts with all the data, split this data into two parts, using impurity. Now the data is divided into two parts, the data that match the split criterion and the data that does not. Now the algorithm continues creating splits of the new data, this is how the decision tree is created.

5.1.1 Decision tree prediction example

To classify a new object, you go through the tree of rules and end up with a value for a new object. To go through an example, a Facebook post from the Falcon Social database is used. The number of impressions will be predicted for the post in table 5.1.1. The tree used for prediction can be seen in figure 5.1.1.

days_since_last	day_of_week	message_length
0.3	1	17

Table 5.1.1 A data points features and values

With a new object, you go through each node, if the answer is yes, you go to the left child of the node, and vice versa. The questions for figure 5.1.1 will be the following:

1. Is the feature days_since_last smaller than or equal to 0.5 → yes
2. Is the day_of_week smaller than or equal to 0.5 → no
3. Is the message_length smaller than or equal to 19 → yes
4. Predicted impressions for this Facebook post: 48169.

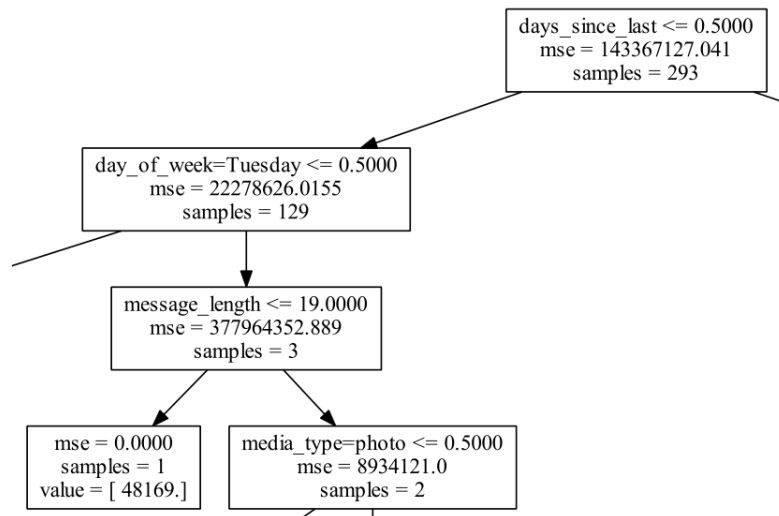


Figure 5.1.1 A visualisation of a trained decision tree and the rules within it.

5.2 Random Forest

Random forest is an ensemble machine learning method for both classification and regression, which consists of multiple decision trees. The trees are grown by using a subsample from the training data, where one third of the data is left out. A test set is hereby left out for each tree. This method of selecting data is called out-of-bag data (OOB or bagging) and is done by sampling with replacement. It is used to also give an unbiased estimate of the classification error as each tree is added to the forest. In other words, bagging is used to average noisy and unbiased models to create a model with low variance.

Consider a training set of $X = \vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$ and its respective classes $y = y_1, y_2, \dots, y_n$. Then, for each tree:

- Take n samples (with replacement) from X and y
- Train the tree using the samples

By sampling with replacement, the same entry can appear multiple times in the training data of a tree thus decreasing the variance of the model. When doing this for many trees, the training set for each tree becomes different. This results in the trees being de-correlated in contrast to using the same training set for each tree. Each tree will be vulnerable to noise in its training set but the average of the trees will not.

When all the trees in the Random Forest have been trained, the model uses each individual tree for classification/regression. Predicting a class for a new object works by majority voting, where the individual tree votes its prediction. The prediction which the majority of the trees agree on will be the prediction of the Random Forest.

Properties of Random Forest

From [Bleiman, 2001] the following properties of the Random Forest method can be listed:

1. The generalisation error almost always converges when the number of trees increases (Theorem 1.2)
2. Law of Large Number ensure that Random Forests do not overfit¹

Finally, [Bleiman, 2001] finds that on common machine learning methods, Random Forests generally tends to perform better than the popular algorithm AdaBoost [Freund and Schapire, 1999] while also being less susceptible to over-fitting and its generalisation error almost always converges.

5.2.1 Feature importance

In some cases of machine learning it is important to know the importance of the individual features. This can for example be used for feature selection.

Given M features and n objects for every feature: $m = 1, 2, \dots, M$, change the order of the corresponding object values for this feature and now do classification. Compute the error value after the objects for this specific feature has changed. Now perform classification without changing the order for the object for this feature. Compare the error rate to the error rate where the order was changed. If a feature has a high importance, the error rate will increase significantly when the object values are changed. If a feature has a small importance the error rate will not increase very much.

When calculating the feature importance, the output is the percentage increase in misclassification [Bleiman, 2001].

5.3 Nearest neighbours

Nearest neighbours is a group of learning algorithms of the type instance-based learning. Two of the most important nearest neighbour learning algorithms will be discussed here:

Method 1: K-nearest neighbours

K-nearest neighbours is a supervised learning algorithm. It takes as input a similarity measure and the number of nearest neighbours.

It works by calculating the similarity to all other neighbours by using the specified similarity measure. With all the similarities to the neighbours, the algorithm finds the k-nearest neighbours, the neighbours that

¹ According to the Law of Large Numbers, if the average of a large number of trials is close to the expected value, then doing an even larger number of trials will tend to come closer the expected value.

are most similar to an object. If the dependent variable is continuous a new object's predicted value is the mean of the k nearest neighbours' dependent variable. If the dependent variable is discrete, predicting a new object will work by majority voting.

Method 2: Radius based nearest neighbours

Radius based nearest neighbours works very similar to K -nearest neighbours. Instead of a specific number of neighbours, a radius is chosen and the neighbours which are inside the radius are chosen. If the machine learning problem is regression, a prediction is made from the mean of the chosen neighbours. If, on the other hand, it is classification problem, the majority vote is done to classify a new object.

Similarity measurement

Finding the most similar objects and finding the nearest objects is exactly the same. There are different ways of measuring similarity between two objects. Some of these methods will be discussed in the following section.

5.3.1 Describing different distance measures

Euclidean Distance: The most normal distance measure is called Euclidean Distance. It defines the length of line between two points. The formula is found using the Pythagorean formula.

$$d_2(x, y) = \sqrt{\sum_{n=1}^N (x_n - y_n)^2} \quad (5.3.1)$$

Cosine similarity:

$$\cos(x, y) = \frac{x^T y}{|x_2||y_2|} \quad (5.3.2)$$

Jaccard coefficient:

$$J(x, y) = \frac{f_{11}}{K - f_{00}} \quad (5.3.3)$$

Simple matching coefficient (SMC):

$$SMC(x, y) = \frac{f_{00} + f_{11}}{K} \quad (5.3.4)$$

K : Total number of attributes

f_{00} : Number of attributes where $x_k = y_k = 0$

f_{11} : Number of attributes where $x_k = y_k = 1$

5.3.2 Example of computing distances using different similarity measures

To give an example of using a nearest neighbours algorithm, K-nearest neighbours is chosen.

The objective is to decide if Ben is either overweight or not overweight. To do this the most similar people needs to be found first.

Using table 5.3.2 the most similar person to Ben using Jaccard coefficient is Jim. Using SMC the most similar person is Jim, using Cosine similarity it is Jim again, and using the Euclidean Distance it is Pete.

To actually classify Ben's overweight feature, first a distance measure is chosen: SMC and a number of nearest neighbours: 1. The nearest neighbour to Ben is the Jim who is not overweight. Ben will therefore be classified as not being overweight.

id	Owens dog	Owens car	Walks every day	Is overweight
Ben	1	1	0	?
Pete	0	0	0	1
Jim	1	0	1	0

Table 5.3.1 Objects to measure similarity from

Distance	Jaccard	SMC	Cos	Euclidean
$d(Ben, Pete)$	$\frac{1}{4-1} = \frac{1}{3}$	$\frac{2}{4} = \frac{1}{2}$	$\frac{1}{3*1} = \frac{1}{3}$	$\sqrt{(1-0)^2 + (1-0)^2 + (0-0)^2 + (0-0)^2} = \sqrt{2}$
$d(Ben, Jim)$	$\frac{1}{4-0} = \frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{3*\sqrt{4}} = \frac{1}{6}$	$\sqrt{3}$

Table 5.3.2 Different distance measures

5.4 Agglomerative hierarchical clustering

Agglomerative hierarchical clustering is an unsupervised learning algorithm. It works by using the following algorithm [Athman Bouguettaya et al., 2014].

- As in K-nearest neighbours, the distance matrix for every object is calculated, using an arbitrary distance measure.

- Find the smallest distance in the distance matrix.
- Merge the two objects with the smallest distance to each other.
- Update the distance matrix with the new cluster.
- Repeat point 2-4, if there is more than one cluster.

When merging two clusters, there are different ways of calculating the distances between them.

Minimum: The distance between two clusters, is the smallest distance between two objects in two different clusters.

$$Proximity(C_i, C_j) = \min_{x \in C_i, y \in C_j} distance(x, y) \quad (5.4.1)$$

Maximum: The distance between two clusters, is the largest distance between two objects in two different clusters.

$$Proximity(C_i, C_j) = \max_{x \in C_i, y \in C_j} distance(x, y) \quad (5.4.2)$$

Group average: The distance between two clusters, is the average distance between all observations in the two clusters.

$$Proximity(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} proximity(x, y)}{m_i \cdot m_j} \quad (5.4.3)$$

Ward's method: Wards method looks at variance in clusters. It calculates the variance if a new cluster is formed and merges two clusters that has the smallest increase in variance.

C_i : Observations in cluster i

C_j : Observations in cluster j

m_i : Number of observations in cluster i

m_j : Number of observations in cluster j

With the algorithm, a number of clusters can be specified, and when the algorithm hits its desired number of clusters it stops, and returns the clusters and which objects it is associated with.

5.5 Latent Dirichlet Allocation

One way to find topics within a collection of documents is Latent Dirichlet Allocation (LDA), which was developed by David Blei, Andrew Ng and Michael I. Jordan. In general, a topic model is build around the idea that some words occur more often than others, so that a document about dogs will have words such as "dog" and "bone". On the other hand, those words are less likely to appear in documents regarding horses.

This section will give a general introduction to LDA, and the interested reader may refer to [Blei et al., 2003] for a more in-depth explanation by the inventors.

The William Randolph Hearst Foundation will give \$1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be \$200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive \$400,000 each. The Juilliard School, where music and the performing arts are taught, will get \$250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual \$100,000 donation, too.

Figure 5.5.1 Topic models assumes that a document can have several topics. Here the words for every topic is highlighted in the same color. The labels of the topic is not provided by the model, but is judged by humans. The example is taken from Blei et al., 2003

An example of a text that has a mixture of topics, can be seen in figure 5.5.1.

When talking about topic modelling words, documents and corpus will be defined as the following terms:

- **Words:** Words are represented in vectors of size of the vocabulary, so $1, \dots, V$, where a 1 denotes that the word is equal to the first word in the vocabulary. All other fields are 0, making the vector incredibly sparse.
- **Document:** An array of N words expressed as $\mathbf{w} = (w_1, w_2, \dots, w_N)$ where w_n is the n th word in the array.
- **Corpus:** A collection of M documents denoted as $D = \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M$

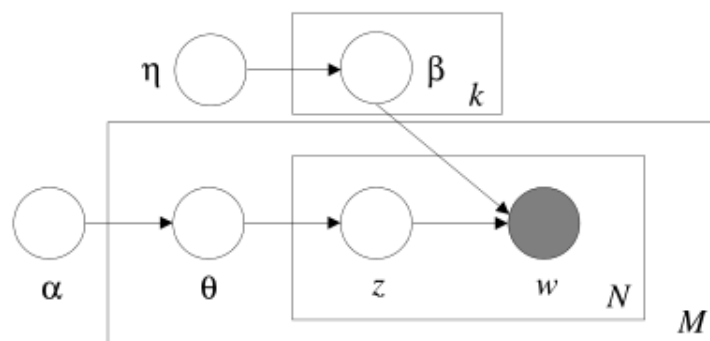


Figure 5.5.2 LDA is a three layered model, where the outer plate is the documents and the inner plate visualises the continued choices of words and topics regarding a document.

Figure 5.5.2 provides a graphical model for LDA, where each node is variable (white is non-observable, grey is observable). The edges denote that which nodes rely on each other. So, for instance, θ is taken from a

distribution of α . Using the same logic, each word w comes from the distribution of z and β - expressed as the multinomial probability $p(w_n|z_n, \beta)$.

α and η are Dirichlet parameters of the distributions θ and β respectively. [Blei et al., 2003] describe those relationships in a more detailed manner, but it is important to mention that changing α and η shifts the Dirichlet probabilities.

The dimensionality of the topics K (the number of topics) is assumed to be known in LDA, and is therefore chosen before applying the model. There is no "ground truth" on how to estimate the topics, and it is quite difficult to grasp around a guess of the number of topics in huge collections of documents. Other topic modelling methods such as Latent Semantic Indexing (LSI) and Non-Negative Matrix Factorization (NNMF) also assumes that the number of topics is known. The next section provides a way to estimate the number of topics by treating LDA as a NNMF model (i.e. not probabilistic).

5.5.1 Natural number of topics in LDA

Despite the fact that LDA is a probabilistic generative model, it can be viewed as an NNMF model, as suggested by [Arun et al., 2010] where a given Document-Word Frequency matrix M is separated into to matrices: A Topic-Word matrix M_1 and a Document-Topic matrix M_2 with sizes $z \cdot w$ (topic/word) and $d \cdot z$ (document/topic) respectively.

The proposed way of finding a right number of topics is done by minimising the following equation:

$$SymmetricKLDivergence(M1, M2) = KL(C_{M1} \parallel C_{M2}) + KL(C_{M2} \parallel C_{M1}) \quad (5.5.1)$$

where C_{M1} is the distribution of singular values in M_1 and C_{M2} is distribution of the normalised vector $L \cdot M_2$ (here L is a vector with lengths of each document in M_2). KL is the Kullback–Leibler divergence [Kullback and Leibler, 1951]

The proposed measure should then be seen to diverge around a certain number or range of topics. One way this could look is illustrated by figure 5.5.3

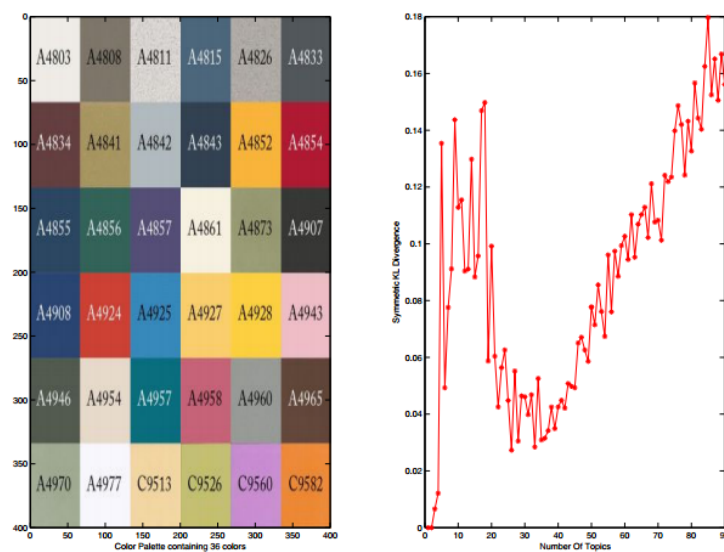


Figure 5.5.3 This figure illustrates a palette of 36 colors - corresponding to 36 topics. The graph on the right hand side, which is the divergence of the palette, seems to diverge around 30-40 topics while increasing heavily afterwards. Such spike reveals the number of topics in a corpus in LDA.

Chapter 6

Analysis

Contents

6.1	Data Architecture	49
6.2	ETL	51
6.3	Introduction to the data	52
6.4	Cleaning data	52
6.5	Data Exploration	54
6.6	Feature Engineering	66
6.7	Clustering	81
6.8	Choosing What to Predict	81
6.9	Model Selection	86

The framework in chapter 4 outlined a process that could be used to implement predictive analytics to an enterprise environment. Chapter 5 described a couple of models for predictive analytics. This chapter puts the framework and the models to the test, and follows the guidelines provided in order to implement the predictive publishing feature for Falcon Social.

6.1 Data Architecture

Data is gathered from three different locations - Falcon Socials database, Falcon socials insight database and the Facebook API. After the initial data mining, the data is processed and calculations are made in order to present the data in the most convenient way (exactly how is explained later).

Figure 6.1.1 visualises the data flow that is used in this project. Each element will be described in the following.

- **Falcon.post:** This table contains the most common historical data about Facebook posts of all kinds (dark posts, paid posts, unpublished posts and more). It is the starting point of the ETL process.

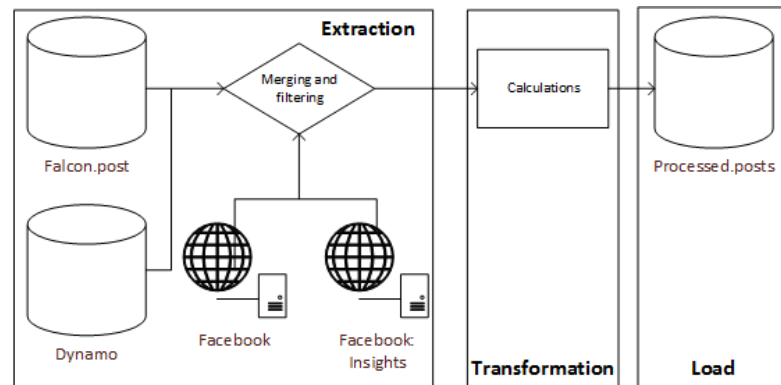


Figure 6.1.1 The data flow for this project.

- **Dynamo:** This database stores data related to Facebook Insights e.g. post impressions, consumptions, reach and a lot more. The data normally requires an API key for the Facebook account it belongs to, which Falcon has obtained from users of their platform.
- **Facebook:** Facebook provides a public API where, given an access token, it is possible to query information about a customers Facebook page and its posts to the extent of what the customer has chosen to make visible. This involves location, brand followers, posts (likes, created time, comments, shares) and other interesting data ¹.
- **Facebook Insights:** This is the live Facebook API. This is used if the Dynamo database have missing values or missing insights.
- **Merging and filtering:** This part merges the Falcon database and the Dynamo database into Facebook post objects. The list of post objects is filtered with the following filters:
 - Paid reach: The post should not have been paid to increase the reach. Paid reach is difficult to predict, as the success of a post when paid for is highly influenced by the price, which can not be extracted and target group.
 - The language has to be English in order for sentiment analysis to work.
 - The page of a post should not be a brand parent page, to avoid getting wrong number of followers, see section 6.8.2.2.
 - The post should not be a dark post. If a post is a dark post, it means that the post is not shown in the page feed, but is used for targeting specific groups.
 - The page a post belongs to should have more than 0 followers. This ensures that test pages are not included.

¹<https://developers.facebook.com/docs/graph-api/reference>

- The post should have more than 0 impressions and 0 consumptions. This ensures a post has in some way interacted with some users.
- **Calculations:** This is where most of the calculations are made, such as averaging, topic modelling, etc.
- **Processed.posts:** Contain the calculated features of the individual posts.

Preprocessing and storing the data on a new location is key to provide a data structure that is compatible with machine learning models and swiftly retrievable, so it can be used for real-time applications.

6.2 ETL

In order to support an ongoing machine learning process (i.e. constant updates of features and visualisations), processed and clean data needs to be available. This would also save time and minimise errors regarding data processing. For this task an ETL tool was developed - from now on mentioned as preprocessor. The preprocessor was developed in Python using SQLAlchemy for database connections, NLTK and Gensim for calculations and topic modelling.

With reference to figure 6.1.1, the preprocessor takes care of the ETL process in the following manner:

- **Extraction:** Content from Falcon.post, Dynamo, Facebook Insights and Facebook sources are queried and put into Python objects. Missing values are either calculated or replaced by default values where necessary to ensure high data quality. This will later on be very convenient when machine learning models are being trained and evaluated.
- **Transformation:** The extracted data is loaded into a SQLAlchemy object (which maps directly to the target database). Each post is mapped to a page/customer in the target database. Meta-data about that customer is updated for every post.
- **Load:** After the processing of data is completed, the posts and customers are loaded into the new database. Below is an example how SQLAlchemy loads posts to the target database using the mapping of Post() to the database using Python.

```
post = Post()
post.set_body_length(post_load.body_text)
post.set_dayofweek(post_load.created_date)
post.id = post_load.id
post.likes = post_load.likes
post.shares = post_load.shares
post.comments = post_load.comments
post.reach = post_load.reach
post.media_type = post_load.mediatype
post.set_hour_of_day(post_load.created_date)
post.set_relative_likes(post_load.likes, post_load.followers)
post.set_relative_shares(post_load.shares, post_load.followers)
post.set_relative_comments(post_load.comments, post_load.followers)
post.set_engagement(post_load.likes, post_load.shares, post_load.comments, post_load.followers
)
```

```

post.set_lexical_diversity(post_load.body_text)
post.days_since_last = post_load.days_since_last
post.set_sentiment(post_load.body_text)
post.page_id = post_load.fbid
post.page_rank_url = post_load.pagerank_url
post.message = post_load.body_text
post.set_status_type(post_load.id, graph)
post.created_date = post_load.created_date
post.set_topic_number(clean(post_load.body_text))
s.add(post)
s.commit()

```

Preprocessor is built-up around delta-loads, where posts can be loaded in as they come and not all at once. That way it can be avoided to truncate the whole database and perform the same calculations every time the target database needs to be updated. There are several advantages to this approach for Falcon Social: The preprocessor can run over-night jobs in order to avoid daily performance instabilities and ensures that the trained machine learning model can be updated to match the latest trends.

6.3 Introduction to the data

There is a huge amount of data linked to a Facebook post and it originates from different places on Facebook.

The Facebook Graph API contains more simple data, such as from which page the post originates, its likes, shares, comments, created date, etc. All of which are open to the public, depending on the customer's page settings. Data that a business wants to keep more private and use for data analysis and strategy can be found at Facebook Insight. Entries such as consumptions and impressions are stored there among other interesting metrics. Insights do however require an API key from the page where the post originated. An API key for the page is private, but can be shared to allow others to data mine the insights of the page.

With help from Falcon Social it is possible to get historical data all the way from October 2007 until today. Since Facebook has changed a lot since 2007, the content has as well. It is therefore necessary to investigate how the data is structured and stored, which will be done in the next section.

6.4 Cleaning data

In an ideal world, all collected data would be stored in the same format, there would be no outliers due to errors, and that data would be complete (no zero or null values). In the real world the data is a lot more flawed. This project faces similar challenges, although working with data from Falcon Social and Facebook whose entire business is founded on big data, the data used in this project is mostly of good quality - there shouldn't be too many missing values, strange labelling or other issues as described in section 4.1.2.

6.4.1 Data issues and missing values

The Post database provided by Falcon Social is in general a healthy database as described in the concept of uniformity defined in section 4.1.2:

- **Accuracy:** The posts stored in the Post database are continuously updated, which means the entries related to a specific post are mostly accurate. Sometimes the number of likes, comments, shares, etc. may deviate a bit from the truth if the posts haven't been crawled for a while.
- **Completeness:** In the table below there is an overview of important features, where it can be seen that some columns have up to 51.24 % missing values. Some columns, on the other hand, do not have None-values.
- **Consistency:** By investigating manually it seems that the database of Facebook posts contains full consistency of values. Since only the Post database and its related customer database is considered, the risk of lacking consistency is low (a higher number of tables will increase this chance).
- **Uniformity:** Uniformity seems to be strictly obeyed. For instance, all dates in the createddate column is in UTC.

Column	% None-values	N	n
name	12.89%	10000	1289
objectid	51.24%	10000	5124
reach	23.48%	10000	2348
caption	14.99%	10000	1499
link	8.83%	10000	883
likes	0.19%	10000	19
message	0.33%	10000	33
sourceid	0%	10000	0
createddate	0%	10000	0
type	0%	10000	0
id	0%	10000	0

Table 6.4.1 An overview of some of the missing values in the Falcon Social Post database.

The biggest problem of the Post database seems to be the missing values. This is addressed by querying the Facebook API in order to get the correct values. This is especially relevant for missing likes and links. Missing captions and messages are mostly due to empty strings, which can be handled by setting an empty string instead of setting it to null. Reach could be data mined from the Dynamo database. The flow in which the querying from Facebook and Dynamo happens, can be seen in the database architecture figure 6.1.1.

For the boundaries of the project, name, objectid and reach was not used. Handling those missing values would require even more data mining of Facebook.

6.5 Data Exploration

Most of the data available to this project is typically data mined from Facebook by Falcon Social. Exploring outliers, correlations and distributions of data on different levels will give a great overview of what the strengths and challenges the data has. Also, this will help unveil some hidden tendencies in the Facebook data that might not be obvious or publicly documented.

6.5.1 Correlation matrix

To explore if there are any relations between the different attributes in the dataset, a correlation matrix has been created using 76010 posts, this can be seen in figure 6.5.1. The strong correlations in the matrix are the different dependent variables correlating with each other. Likes, consumptions and impressions correlate strongly with a positive magnitude. Also the average likes, average consumptions and average impressions correlate strongly. One thing to notice, is that likes correlate more with impressions than consumptions. The reason may be that if a post gets many likes, Facebook boosts its organic reach, thus resulting in more impressions, but will not necessarily result in more clicks. An example could be a funny status update. This could get impressions and likes, but not necessarily consumption, as there is nothing to click on in a funny status update.

Another thing to notice is the weekdays that correlate negatively with each other. This makes of sense, as they are mutually exclusive. The same can be said for some of the media types, as, for example, a status and an image, or an album and an image type.

Regarding the prediction of the dependent variable, the most useful attributes are the average dependent variable for a customer, which are average likes, consumptions and impressions. Table 6.5.1, shows the most correlated features apart from the averages. This shows that the link attribute is negatively correlated with most of all the dependent variables. Hashtags and mentions are positively correlated. Regarding consumptions the most correlated feature is album. This can be because consumptions regarding a post inside an album aggregates to the album post. Impressions are correlated most strongly with videos, suggesting that Facebook boosts organic reach with videos. Facebook definitely have videos as a central part of their ranking ², so this strategy would make a lot of sense.

Performing correlations with this data set also contains some issues. 65% of total likes are distributed within 1% of the Facebook posts. This means that the correlations of the features are mostly influenced by a few number of posts, which does not give the full picture of the correlations of all the posts.

²<http://drivingsalesnews.com/facebook-algorithm-changes-will-improve-relevancy-of-videos-in-news-feed/>

Likes	Bool hashtag: 0.06743	Bool mention: 0.067	Link: -0.0517	Photo: 0.0363
Consumptions	Album: 0.1011	Bool mention: 0.0583	Video: 0.0434	Link: -0.0331
Impressions	Video: 0.1132	Bool mention: 0.0978	Bool hashtag: 0.06743	Link: -0.0391

Table 6.5.1 Strongest correlated attributes in relation to different dependent variables

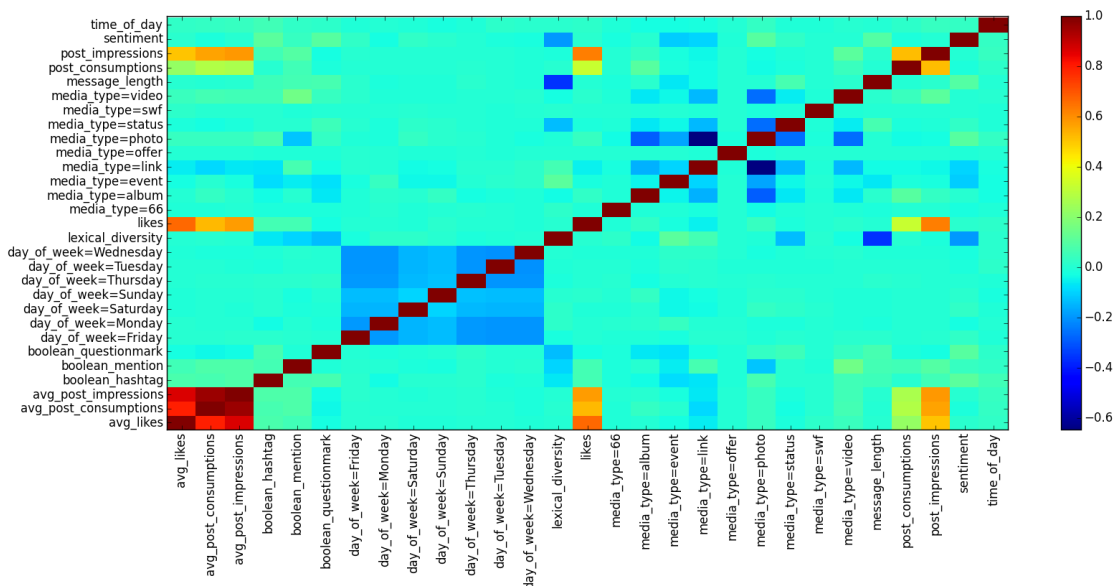


Figure 6.5.1 Correlation matrix for all posts

6.5.2 Distribution of posts

Figure 6.5.2 shows that "photos" account for most of the posts after "links". This was quite surprising, that the type "status" was not the mostly used. This tendency probably comes from Facebook pages creating a lot of "albums" and as a part of their social media strategy, has to posts many "photos". "Event" and "status" are the media types that are used most infrequently. Events is probably hosted on the customers own website and the status update with only text, is not engaging as a customer to submit.

Figure 6.5.3b shows that companies post most of their posts in normal working days.

Figure 6.5.5b shows all posts sorted after likes. It is very clear that the vast majority of the posts have a quite small amount of likes. This is the nature of the Facebook data. The few posts that for some reason become viral, receive a huge amount of likes, but the vast majority of posts never get above 100 likes. 1% of the posts has 65% of the total likes.

The same can be seen with customer like averages seen on figure 6.5.5a. There are few customers with a huge amount of likes, and a lot of customers with a very low average number of likes. 1% of the customers

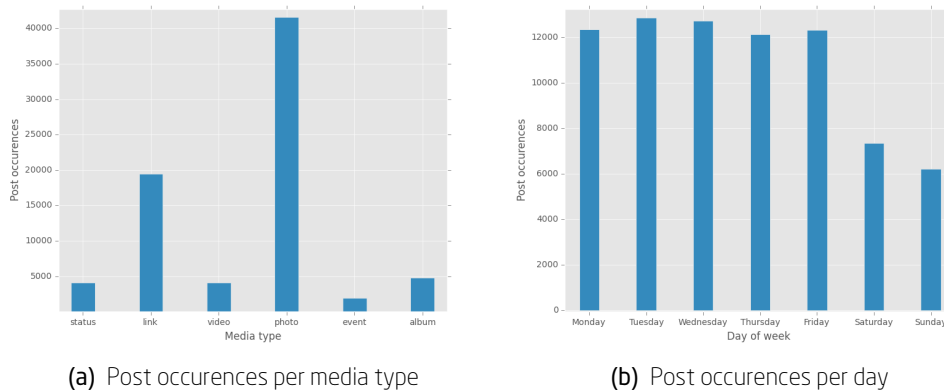


Figure 6.5.2 Barchart of post distribution over media types and day of week

has 62% of the total average likes.

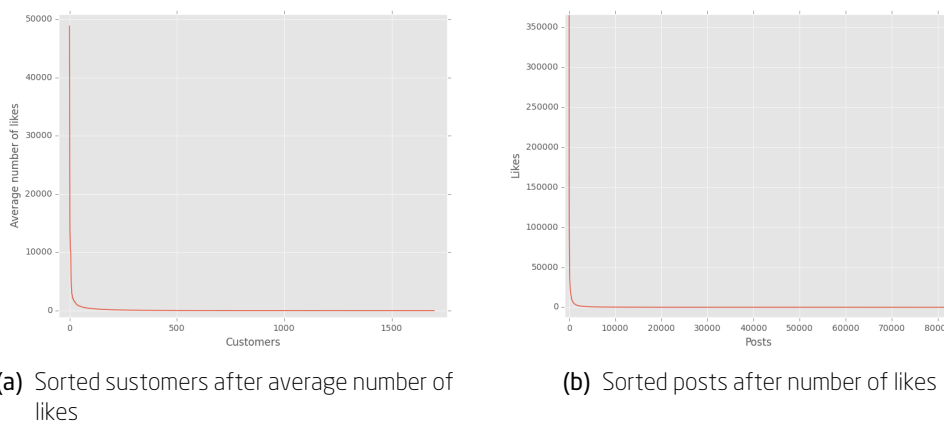


Figure 6.5.4 Sorted posts and customers after number of likes and average likes

6.5.3 Outliers

The previous section hinted that the data set was heavily dominated by a few customers and posts having a high amount of likes. Figure 6.5.5b showed that a lot of likes is distributed only to a few posts. Same happened in figure 6.5.5a where there are a few customers who has the majority of likes. This can be summarised a bit into some key statistics:

- Of all the 1699 customers in the data set, only 101 customers have above 400 likes on average.
- The top 7 customers all have more than 11,000 likes - on average

- 755 customers get below 5 likes on average.
- The average for all posts are 340 likes - the median is 6

The fact that the difference between the average and median is so substantial can cause a lot of trouble when trying to predict likes.

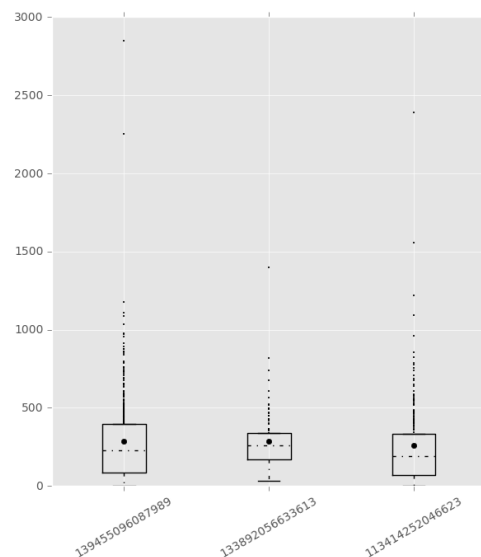


Figure 6.5.6 Boxplots of the likes of three randomly selected Facebook pages.

Also on an individual page level, the posts tend to vary a lot. Figure 6.5.6 shows three randomly selected customers, with 450, 102, and 225 submitted posts (in the same order as on the figure). The boxplots reveal that data set also can contain outliers when looking on a page individually.

Since the influence of outliers is so dominant to the data, extreme outliers are removed. This will ensure that evaluations metrics such as MSE will be more correct, since a prediction of an outlier with 360.000 likes will most likely be wrongly predicted by a lot. 3% of the outliers in each tail - top and bottom - are therefore removed.

6.5.4 The problem of UTC time and an international business

Facebook stores the creation date of a post in UTC time. This means that posts that are created at 15 o'clock UTC can for some customers locally be at 10:00:00 in New York or in Poland 16 o'clock. There are at least two ways of fixing this problem when doing machine learning.

- Get a customer's country/address and from this, extract the timezone. With the the local timezone, create the local time.

- Provide context regarding customers, ensuring some sort of identification of a customer.

Extracting countries and addresses was done, but the information of customers timezone was only available for 51% of the customers. Due to the lack of data, converting UTC timestamps to local time was not done.

Regarding the second point, context is given for a customer, using the features: average likes, impressions, consumptions and the variance of likes, impressions and consumptions. See section 6.8.2.4, for further explanation of the contextual features.

The last note about the timing of a post, is that for some customers, the time of day is not important when doing prediction. If a large customer like e.g. Jaguar that have fans from all over the world, submits a post, this post will be received at different local times all over the world. This could result in time of day not being important.

6.5.5 Customer exploration

To get a grasp of what the data looks like for a specific customer, two customers will be used as an example, and their data will be analysed. The two companies are Roskilde Festival and Franklin & Marshall.

6.5.5.1 Roskilde Festival

Roskilde Festival is the biggest festival in Denmark with up to 90.000 people attending every year. The Facebook page has 283.535 followers. It is used to post events, new music bookings, music from the festival, pictures from the festival and much more.

The media type:

Roskilde Festival posts mostly links (17 %), videos (15 %) and photos (58%). When looking at the boxplots for the different kinds of media types in figure 6.5.10, it can be seen that the amount of outliers is large. This is the nature of a lot of the data on Facebook, but especially Roskilde Festival. An example of huge differences in likes can be seen in figure 6.5.8 and 6.5.7. Regarding performance of the different media types, video and album performs better than photos. Especially albums relating to consumptions perform a lot better than the rest of the media types. This comes from consumptions that are aggregated when clicking through the photos of an album. When looking at impressions for photos the median is 2000 which is very low. This comes from an image being contained in an album with many pictures. An example is <https://www.facebook.com/10151892491956706>. It has 21 impressions, which is unusual for Roskilde, but is in an album of 160 photos.

Post timing

Roskilde Festival submits posts to Facebook all days of the week. The day with the highest median of likes is Tuesday, and the day with the highest mean is Sunday - see figure 6.5.14. The day that performs the worst

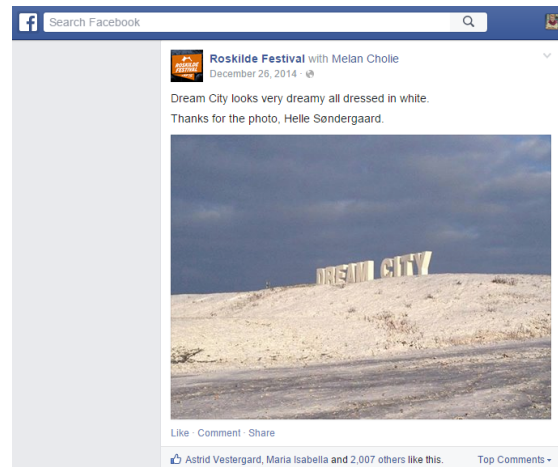


Figure 6.5.7 Post that got 2009 likes



Figure 6.5.8 Post that got 26 likes

is Thursday. There is a strange effect, where the day Roskilde Festival submits a lot of posts, is the day Roskilde Festival performs the worst. This could be because Thursday is 'spammed' with posts and the individual posts therefore are being ignored.

Regarding the time of day, Roskilde Festival is successful at 9-10 o'clock and then again from 19-20 o'clock and at 6 o'clock. Especially at 20 o'clock and 6 o'clock with a large median and mean, but large variance. At 17 o'clock it also seems like Roskilde Festival perform well, with a low variance.

At 8 o'clock in the morning, Roskilde Festival submits lots of posts. This time also performs very badly, as seen with the day Thursday.

Scatter plots

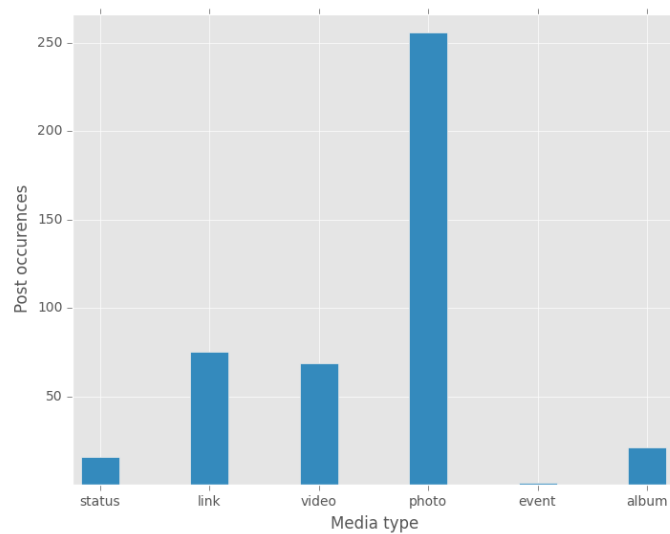


Figure 6.5.9 Number of posts for media types for Roskilde

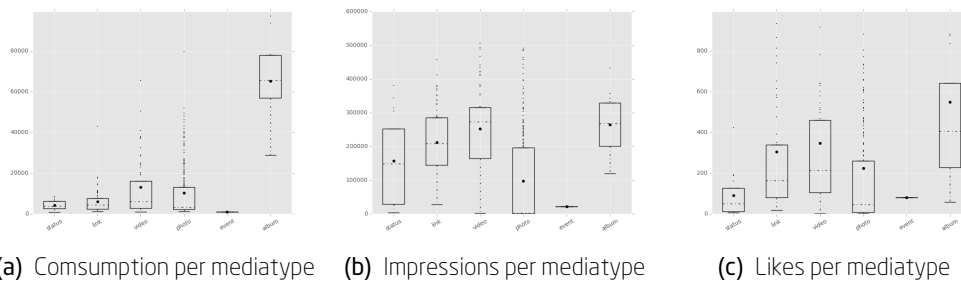


Figure 6.5.10 Boxplots of likes, consumptions and impressions, in regard to the media type for Roskilde

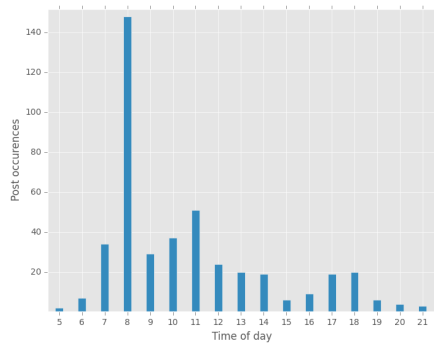
To gain some insights from the continuous variables, scatter plots are used. They are shown in figure 6.5.16

The sentiment scatter plot does not show a lot about the sentiments influence on likes, apart from a small local optimum at 0.1. Message length shows a clear optimum at about 10 words. There is another optimum at about 50 words, this only has 3 posts, and therefore not a lot of data to back up this optimum. Regarding lexical diversity, there is a small optimum at about 0.935, but nothing significant.

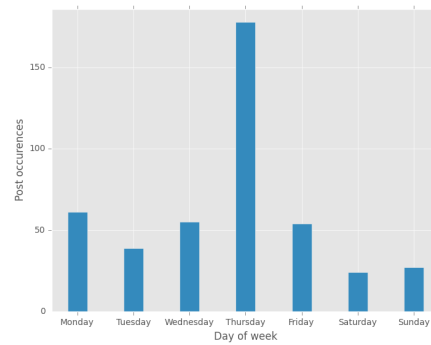
Social media advice

Providing Roskilde Festival with a couple of advice it would be to post more albums, photos and videos. Avoid posting at 7 or 8 in the morning and on thursdays. Submit posts that are slightly positive and keep the message length close to 10.

6.5. Data Exploration

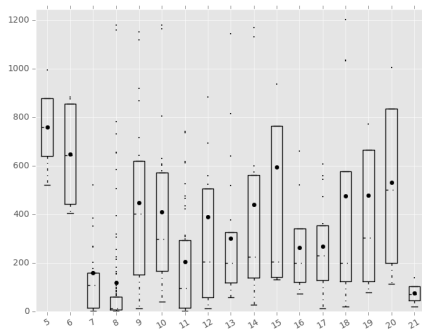


(a) Number of posts per day from Roskilde

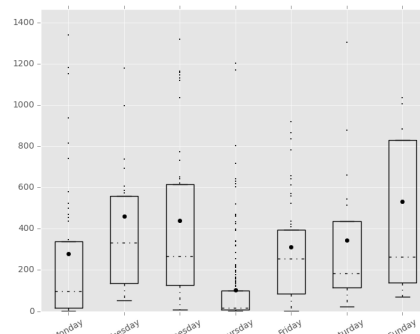


(b) Number of posts per hour of day

Figure 6.5.12 Barchart of the likes and timing



(a) Likes per hour



(b) Likes per day

Figure 6.5.14 Boxplots of likes in regard to the timing of posts

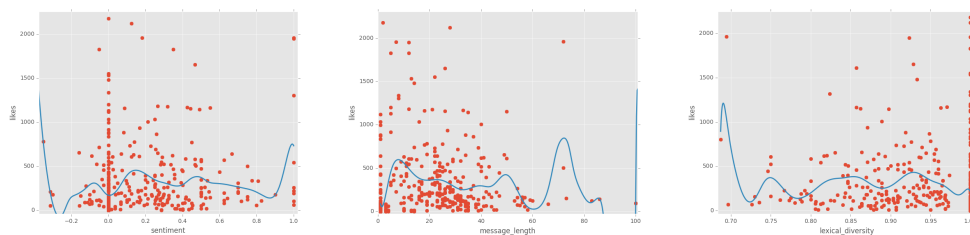


Figure 6.5.16 Scatter plots of sentiment, message length and lexical diversity for Roskilde Festival. A non parameterized regression function has been fitted to the data.

6.5.5.2 Franklin & Marshall

Franklin & Marshall is an Italian fashion brand. They produce clothing and accessories that are sold in both a webshop and in stores. Franklin & Marshall use their Facebook page to post pictures of their clothing,

status updates with sale updates, videos of their clothing and url's to their website.

Media type performance

Franklin & Marshall posts mostly photos (73 %) and videos (14 %). A better visualisation of the distributions of their posts can be seen in figure 6.5.17.

Regardless of what Franklin & Marshall wants to predict (in terms of likes, consumptions, and impressions), the album type is the type of post that performs the best by far, as can be seen on figure 6.5.18.

Posts that perform well on consumptions are posts with type album and photos. For impressions, links, videos, album, and photos all perform more or less equally good. In terms of likes, photos and album perform quite well.

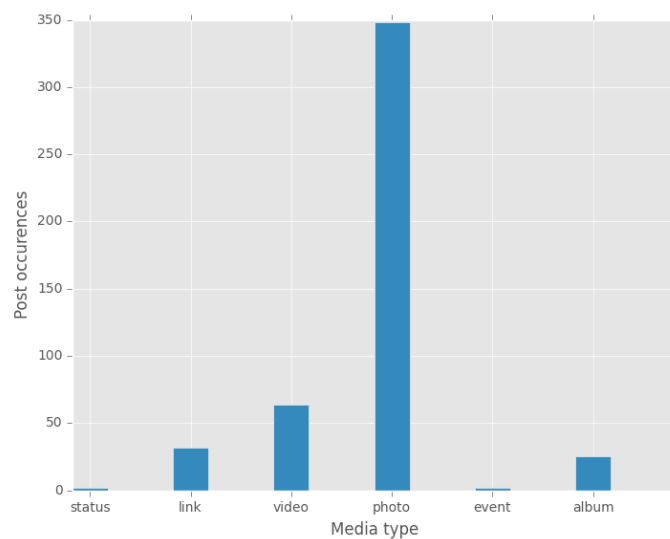


Figure 6.5.17 Number of posts for media types from Franklin & Marshall

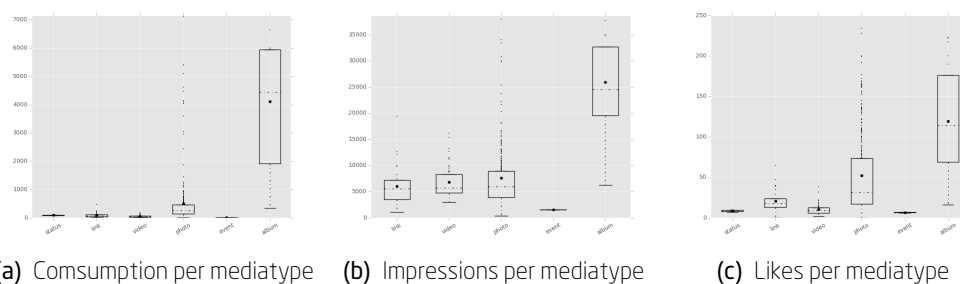


Figure 6.5.18 Boxplots of likes, consumptions and impressions, in terms of the media type from Franklin & Marshall

Post timing

Franklin & Marshall posts at ordinary business hours, with the most posts posted at 15 o'clock, see figure 6.5.23a. The posts published at 16 o'clock and 18 o'clock performs well in terms of achieved likes, with both high median and mean.

Regarding the day to post, Franklin & Marshall post primarily on normal working days, with most posts on Wednesdays, this can be seen in figure 6.5.21b. When looking at the boxplot figure 6.5.23b, the working day that perform the best is Friday, with a higher number of median likes and mean likes. Saturday and especially Sunday performs very well, compared to the rest of the working days. Franklin & Marshall should take advantage of this and post more on Sundays.

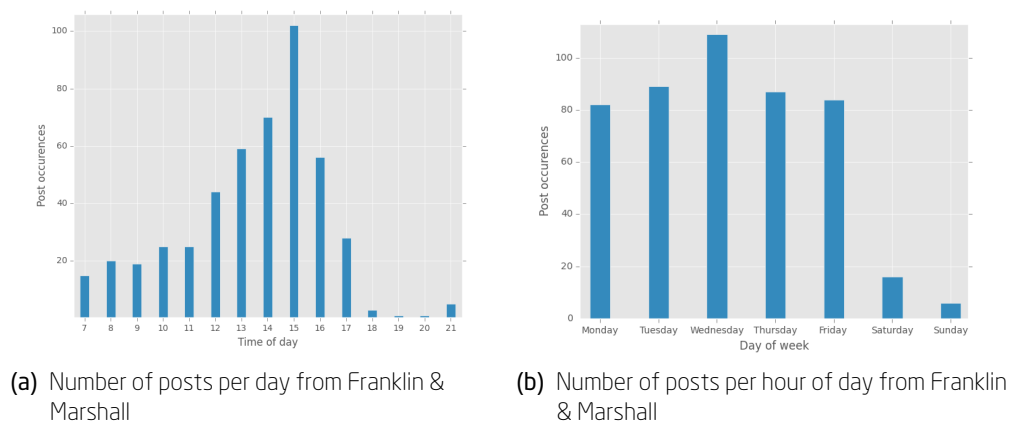


Figure 6.5.20 Barchart of the likes and timing

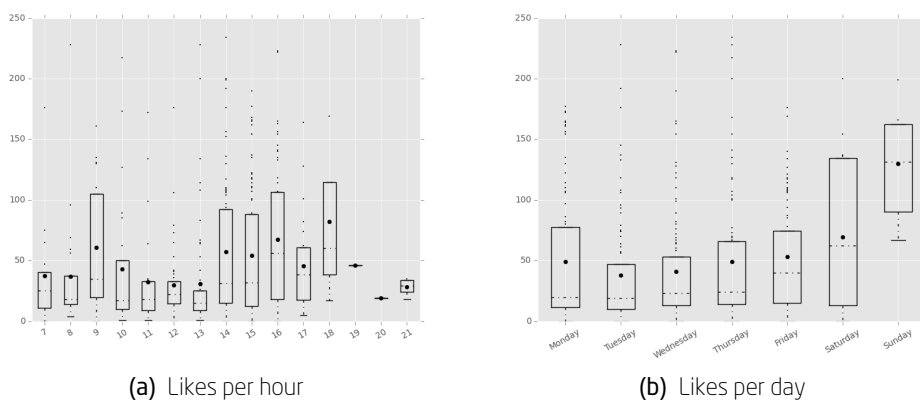


Figure 6.5.22 Boxplots of likes in regard to the timing of posts from Franklin & Marshall

Scatter plots

Figure 6.5.24 provides an overview of the sentiment, message length and lexical diversity plotted against the

number of likes achieved by the post.

The sentiment scatter plot does not show any clear correlation or optimum regarding sentiment. Message length shows an optimum, at 7 words with about 80 likes. Increasing the number of words more than 7, decreases the number of likes. The lexical diversity scatter plot shows a decrease in likes if the lexical diversity is below 0.85.

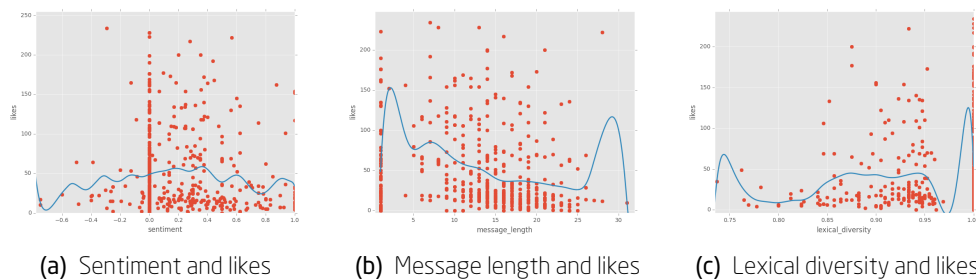


Figure 6.5.24 Scatter plots of sentiment, message length and lexical diversity for Franklin & Marshall

Social media advice

Some advice for Franklin & Marshall, based on the scatter plots, could be: Submit more albums and photos. Post at 9 o'clock in the morning or around 16 - 18 o'clock. Submit more posts in the weekends, especially on Sundays.

6.5.6 Likes over time

When providing feedback with the web tool built in this project, it is important to give the user an indication of when the number of consumptions/likes/impressions (the dependent variables) will be achieved. The dependent variable does not necessarily follow a linear development. For instance many impressions are achieved within a short amount of time in the beginning of a post creation. 7 hours after post submission, the rate of impressions has decreased significantly. An illustration of this effect, can be seen on figure 6.5.26 for consumptions and figure 6.5.28 for impressions. Each figure represents three posts that have small, medium and large numbers of consumptions and impressions, but they still show exactly the same effect.

To answer the question about when a post has received the number of impressions or consumptions, see table 6.5.2. It can be seen how many hours in average it takes for a post to reach a certain percentage of its full number of dependent variables. For instance it takes roughly 9 hours and 20 minutes for a post to get 70% of its consumptions. When indicating when a post has received the number of predicted number of e.g. impressions, it is not crucial that this is an exact number, as long as a user has the idea of how long to wait. Therefore for 95% of the impressions it takes 25 hours, for consumptions this number is around 30 hours.

It has not been possible to extract likes over time, but it is assumed that the time taken to receive impressions and consumptions are the same it would take likes to receive the number of likes.

6.5. Data Exploration

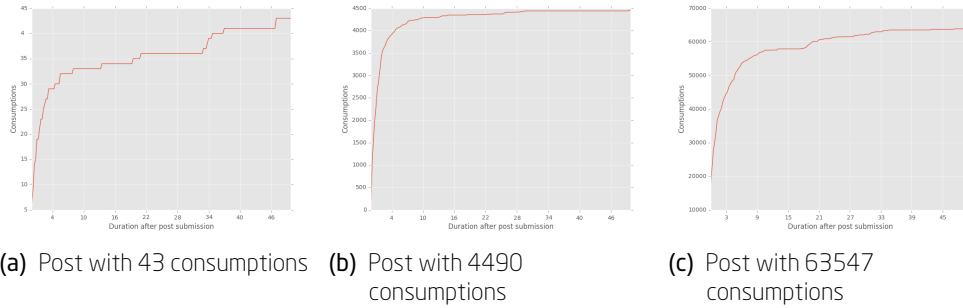


Figure 6.5.26 Consumptions over duration in hours

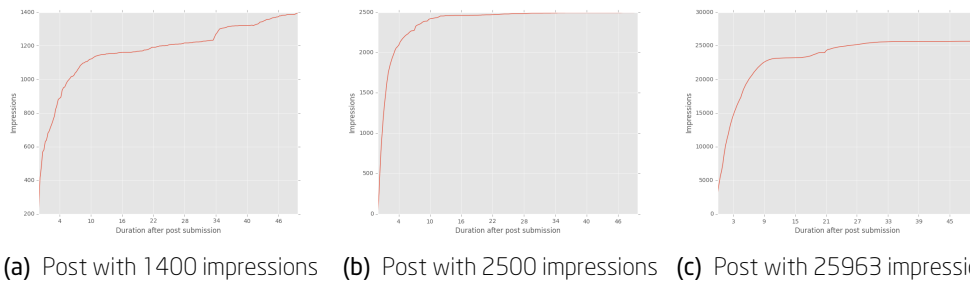


Figure 6.5.28 Impressions over duration in hours

Percentage	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%
Hours for con	4.55	5.30	6.33	7.71	9.33	11.58	14.88	19.2	23.05	30.1
Hours for imp	5.53	6.27	7.08	8.07	9.62	10.71	12.31	15.45	19.62	25.0

Table 6.5.2 Duration in hours before number of impressionsconsumption reaches specific percentage

6.6 Feature Engineering

From a Facebook post and its corresponding page numerous features can be created. The task of identifying the features and, amongst them, selecting the right features will be described in the following section.

6.6.1 User centric feature creation

User centric feature creation as described in section 4.3.1, involves uncovering the underlying truth in the form of features. In this project that means figuring out what makes a Facebook post successful.

First of all, what makes a Facebook post successful is a very subjective and unclear domain, and it is frequently subject to change.

6.6.1.1 Quality content

At Falcon Social, Marketing Operations Manager, May Laursen, and Head of Customer Strategy, Helle Tyllesen who are experts in assisting with social media strategies both agree on the following strategies regarding what defines a successful post:

- Use photo, links or video
- Link to trusted sites (sites that Facebook trusts, such as big news sites)
- Keep content short and precise - less than 70 or 90 characters is optimal
- Limit the amount of information in a post - a post should only contain one message that is to be delivered to its audience
- Avoid like-baiting
- Talk about people
- Use relevant hashtags
- Business verticals matter - post content and structure should be different depending on your area of business
- Timing of the content - when was the post created, how long time since last post was uploaded.
- The sentiment of a post.

Extracting those metrics from the data set will therefore be a good starting point in order to identify which features can be used to predict a successful post.

6.6.1.2 Simple features

The previous section gave some strategies regarding how the success of a Facebook post could be affected. These strategies have been turned into specific features, as listed below:

- Message length
- Time since last post
- Time of day of publication
- Day of publication
- Sentiment
- PageRank of site that is linked to
- Media type - photo, link, status, etc.
- Lexical Diversity

6.6.1.3 Additional features

To improve a learning algorithm, the worst predicted posts were looked through to check if it was possible to create features that would incorporate the reason for this post to be wrongfully predicted.

From this analysis multiple issues were found regarding the prediction of number of likes:

- Some posts were deleted or inaccessible after they had been posted. An example is the post with id 124155458979_10152643477378980. Predicted number of likes is 1850.8 but actual number of likes is 4.0. When trying to get the post from the Facebook API, an unsupported get request error comes up. This can mean many things, from the post being age restricted to the post being deleted. Therefore an additional feature was created - `unsupported_get_request`.
- Some posts that were wrongfully predicted had specific status types, fx if a photo is a `mobile_status_update` or `added_photos`. Knowing what kind of status is created is quite relevant. If a photo is added to an album this post will get the `added_photos` type. In general photos in albums gets fewer likes as there can be many photos in an album.
- Another problem when predicting was the posts that were wrongly predicted nearly all had a specific story type. A story is: "Text from stories not intentionally generated by users, such as those generated when two people become friends, or when someone else posts on the person's wall" ³. With this feature it is possible figure out if a post has been added to an album, set as a cover photo, shared a link. It provides more information than just the status type.

³https://developers.facebook.com/docs/graph-api/reference/v2.2/post?locale=da_DK

- From reading an article about getting more clicks, it was pointed out, that asking questions could influence a posts successfulness ⁴. Therefore a new simple boolean feature was created, that checked if a questionmark was present in a post.
- Another article⁵ suggested that mentioning people was important. Therefore two feature were created: the number of mentions and a boolean attribute that checked if someone was mentioned in the post.
- Including a hashtag in a post, is suggested in an article⁶ to ensure visibility. Two features were created: the number of hashtags in a post and a boolean checking if a hashtag is apparent in a post.
- When looking through a customer's Facebook page albums, it becomes quite clear that some albums perform a lot better than other albums. Looking through Roskilde Festivals timeline photos, shows that the photos have from 200-5000 likes. Comparing this album to another album called Dream City is being built at #rf14, the likes are from 1-19 likes. Therefore getting the album name that a post is posted to, could be an important feature.

6.6.1.4 From Facebook post to processed post

The following gives an example of the process of transforming a Facebook post into its respective features.

The post is a Facebook post from Roskilde Festival, which can be viewed in figure 6.6.1 ⁷. After the data of the Facebook post has been cleaned, the features are created. These feature values can be seen in table 6.6.1.

A simple example of a feature that is created is the questionmark feature: Figure 6.6.1 shows a question mark in the text, this corresponds to the boolean questionmark being true in table 6.6.1. Another simple example is the number of name tags feature. The feature value is 3, because 3 pages/people are mentioned in the post: Förtress, Get Your Gun and Heimatt.

⁴<http://thenextweb.com/socialmedia/2014/11/14/tested-best-advice-get-clicks-facebook-heres-worked>

⁵<http://www.jeffbullas.com/2013/01/21/how-to-get-more-likes-on-your-facebook-page/>

⁶<http://tagcandy.com/blog/3-big-ways-successful-facebook-marketers-use-facebook-hashtags/>

⁷https://www.facebook.com/orangefeeling/photos/a.99196301705.99302.10248811705/10152107205331706/?type=1&relevant_count=1



Figure 6.6.1 Example of a Roskilde Festival post

Feature name	page_id	Likes	Shares	Comments	Sentiment
Value	10248811705	125	4	5	0.75
Feature name	Media type	Days since last post	Message length	Day of week	Time of day
Value	Photo	0	53	Tuesday	18
Feature name	Lexical diversity	Page rank url	Unsupported get request	Status type	Processed story
Feature value	0.898	www.uhoert.dk	False	Added photos	0
Feature name	Topic number label	Likes label	Impressions label	Consumption label	Number of name tags
Value	6	True	False	True	3
Feature name	Boolean mention	Hashtag number	Boolean hashtag	Boolean question-mark	Boolean timeline photo
Value	True	0	False	True	True
Feature name	Boolean coverphoto	Boolean mobile upload	Followers	Avg. impressions	avg. consumptions
Value	False	False	277583	152641	14446
Feature name	Avg. likes	Var. impressions	Var. consumptions	Var. likes	Post consumption
Value	267	22903831933	1068192641	147562	10243
Feature name	Post impressions	page rank			
Value	41894	4			

Table 6.6.1 Post that has been processed into features

6.6.2 Data centric feature creation

This section will explain the methods behind creating the features that are more data driven.

6.6.2.1 Performing PCA

In order to find features using a data driven approach, PCA is used. The dataset is normalized with l_2 norm, before PCA is applied. The analysis returns a number of components, where each feature is represented in the component by a weight. This weight explains how much the individual feature contributes to the component.

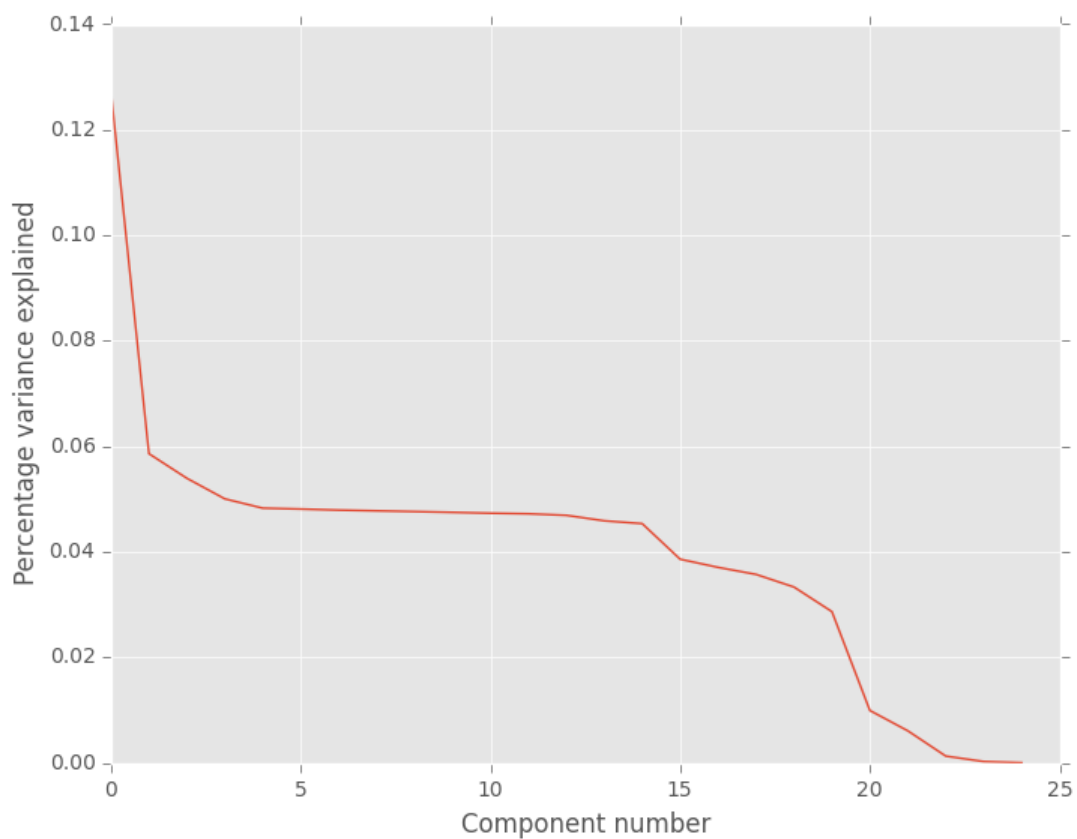


Figure 6.6.2 Variance explained for every component

Looking at the components contribution to the total variance in the dataset, the two first principal components explain 19.6%, see figure 6.6.2. The first component account for 14% of the variance, the second component account for 5.4% of the variance.

The individual features contribution for the two components can be seen in the appendix B.

The first components most important features are avg_post_impressions, avg_post_consumptions, avg_likes. Extracting a post with a low value from the first component and a post with a high value of the first component, the influence of the average features can easily be seen:

```
[avg_likes, avg_post_consumptions, avg_post_impressions, boolean_hashtag, boolean_mention,
boolean_questionmark, day_of_week=Friday, day_of_week=Monday, day_of_week=Saturday,
day_of_week=Sunday, day_of_week=Thursday, day_of_week=Tuesday, day_of_week=Wednesday,
lexical_diversity, media_type=66, media_type=album, media_type=event, media_type=link,
media_type=offer, media_type=photo, media_type=status, media_type=swf, media_type=video,
message_length, sentiment, time_of_day]
High value of first component
[ 0.10657266 0.02504934 0.02430318 0. 0. 0. 0.
 0. 0. 0. 0. 0.00881888 0.
 0.00381232 0. 0. 0. 0. 0.
 0.00491343 0. 0. 0. 0.00014955 0.00275301
 0.00074808]
Low value of first component
[ 4.36156501e-06 5.43041464e-06 1.00753579e-05 0.00000000e+00
 0.00000000e+00 8.66686504e-03 0.00000000e+00 0.00000000e+00
 1.17851130e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 2.99985617e-03 0.00000000e+00 0.00000000e+00
 0.00000000e+00 7.17791680e-03 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.58927068e-03
 2.33171889e-03 1.99488407e-03]
```

The post with a larger value of the first component, has 0.1, 0.025 and 0.024 as avg_likes, avg_post_consumptions, avg_post_impressions, whereas the low value post has 4.36156501e-06, 5.43041464e-06 and 1.00753579e-05.

The second component's most important features are event, questionmark, message length, and hashtag. The feature weights are not as distinct as in the first component, though event has quite a large weight. Two posts one with high value of second component and low value can be seen below:

```
High value of first component
[ 0.00000000e+00 3.87886760e-07 7.93166471e-07 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 9.07367797e-03 0.00000000e+00
 0.00000000e+00 2.08068204e-03 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.56059614e-02 0.00000000e+00 0.00000000e+00 6.28870134e-02
 8.57538633e-04 4.48848916e-03]
Low value of first component
[ 2.18078251e-06 8.35120194e-04 5.05911587e-05 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 8.99114808e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 3.81231721e-03 0.00000000e+00 0.00000000e+00
 2.29961025e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 7.47764725e-05
 0.00000000e+00 2.99232611e-03]
```

The two posts are distinctly different in terms of event, message length, hashtag and questionmark.

Figure 6.6.3 shows the posts projected onto the two first components. The posts that have a like count below the median number of total likes, are blue. These can be clearly seen to the right of the figure. This effect comes from the first component, where posts that have small values of average likes, consumptions and impressions are positioned closer to zero.

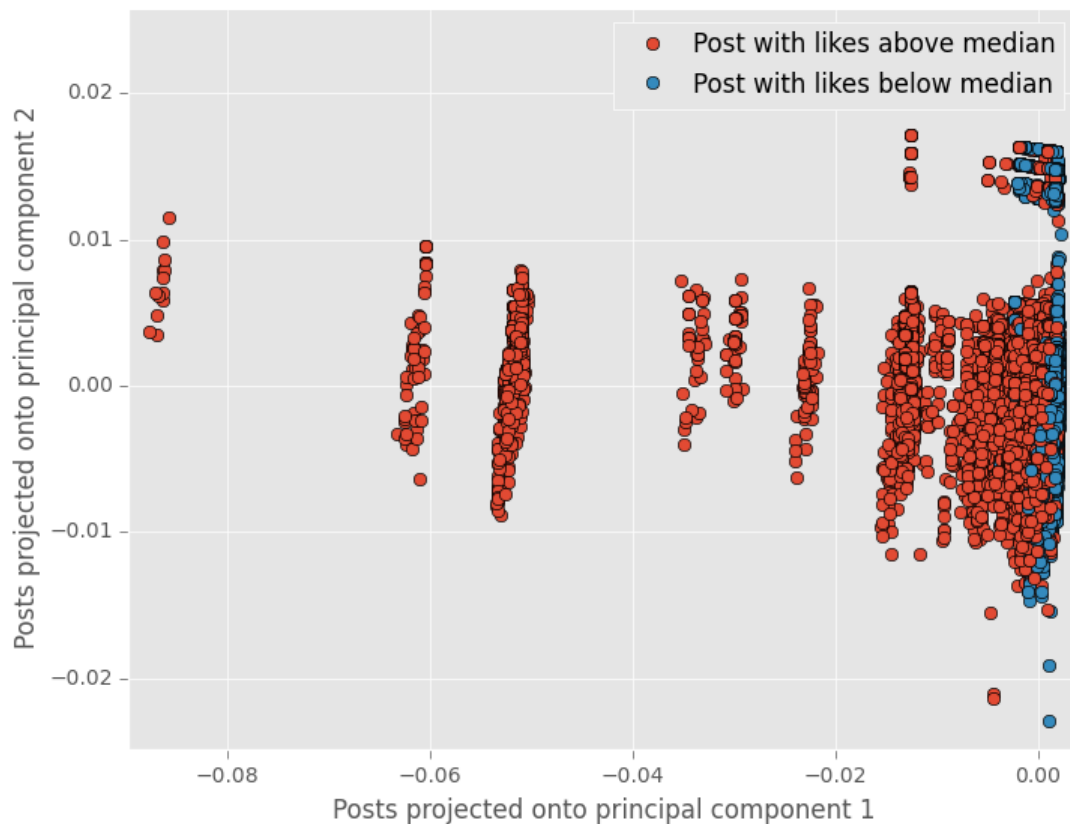


Figure 6.6.3 Data projected onto the two main principal components

From PCA it was found that most of the variance in the dataset comes from the average likes, consumptions and impressions, but that also the features event, questionmark, message length, hashtag and sentiment contain some of the variation. These are some of the features that are worth looking at when doing feature selection.

6.6.2.2 Topic modelling

The aim of topic modelling as a feature is to investigate and capture if certain topics have influence regarding a posts performance. It was previously explained in section 6.6.1 that in order to make a successful post, publishers should avoid making like-baiting posts and perhaps talk about people.

As explained in section 5.5.1, finding the number of optimal topics in a large collection can be a challenging task, and there is no straightforward way to do it. Especially when dealing with Facebook posts, where the

number of words per post is rather limited, compared to an article or blog post. Topic modelling in this case is therefore not expected to capture all the correct topics in the corpus. It is rather used to improve prediction by assigning a topic to each post, and investigate if there is a tendency for the specific topic. If a topic generally performs above/below average, this could possibly help capture the successfulness of a post, and hopefully increase prediction accuracy.

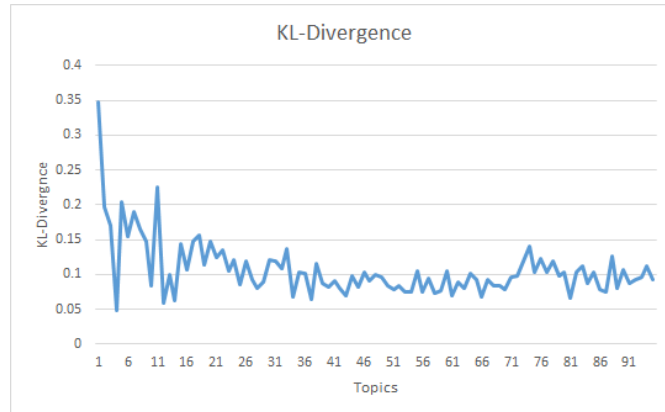


Figure 6.6.4 In LDA it is possible to estimate the number of topics in a corpus by looking at the KL-divergence. Although in this case there doesn't seem to be any clear divergence.

In order to find the topics, an attempt to look at the KL-divergence, to find the optimal number of topics is made. The results can be seen at figure 6.6.4. The figure indicates that there is no clear indication of where optimal number of topics are, and the divergence seems to be more or less the same from around 20 topics. In order to proceed here, 18 topics are chosen to exist in the data. The topics could not directly be chosen from the KL-divergence, so a more un-educated guess had to be made. Again, it is not so much the correctness of the topics in the corpus that is important here.

The results of the topic modelling is listed in appendix C. One of the topics found contains the following words and corresponding likelihoods:

Topic 7: 0.044*win + 0.023*enter + 0.021*share + 0.020*chance + 0.017*like + 0.013*facebook + 0.013*winner + 0.012*christmas + 0.012*give + 0.011*post

A topic such as the above may subjectively indicate some sort of like or share baiting, but looking at table 6.6.2, the topic doesn't seem to have a high fraction of posts below the median.

Topic 9, which has the highest fraction of likes below the median, does not contain any words which would indicate that it encourages like baiting, foul language, or anything else than might encourage the post to be badly received by Facebook or an audience.

Topic 9: 0.016*new + 0.014*water + 0.012*take + 0.011*excite + 0.011*week + 0.009*never + 0.009*fashion + 0.009*get + 0.008*run + 0.008*garden

In fact, figure 6.6.5a is an example of a Facebook post that talks about soccer and there are no factors that

Topic	0	1	2	3	4	5	6	7	8
Fraction(%)	0.54	0.56	0.53	0.56	0.54	0.54	0.58	0.57	0.57
Topic	9	10	11	12	13	14	15	16	17
Fraction (%)	0.59	0.57	0.56	0.56	0.54	0.55	0.52	0.58	0.56

Table 6.6.2 Overview of less than median likes. The posts are here grouped based on their topic, and is evaluated whether or not the post is below the median of the customer it belongs to.

indicate that the post should perform badly. The post gets 206 likes which is fairly good. On the other hand, the post from 6.6.5b tries to encourage redirections to a external and url-shortened link.

Looking at the findings in this section it is not clear that topic modelling, in its current state will make a huge impact when doing predictions. Though even just a slight improvement from topic modelling will be valuable for Falcon Social and their customers, as this will result in better predicted posts.



- (a) This Facebook post is labelled as topic 9, which generally performs poorly. In this case the post performs better than its average likes. It got 206 likes versus the average of 100



- (b) Towels R Us promotes their towels in this post, and its related to topic 9. The post only gets 0 likes, and is below its average of likes. Without making distinct conclusions, avoiding this type of post could help improving the brands social media success.

Figure 6.6.5 Example of posts labelled as topic number 9

6.6.2.3 Natural Language Processing

In order to analyse the text of a Facebook post, the text must first be prepared for analysis, this preparation is called NLP and is mentioned in section 4.3.2.2.

The Facebook posts from the Falcon Social database come in various qualities. Some posts may be without text, some with a lot of spelling errors and some in different languages. Therefore the following is applied to the text of a Facebook post:

1. HTML is removed
2. The words are tokenised
3. The words are POS-tagged
4. Finally, the text is lemmatised

The post seen in figure 6.6.7, will be used as an example in the rest of the calculation.

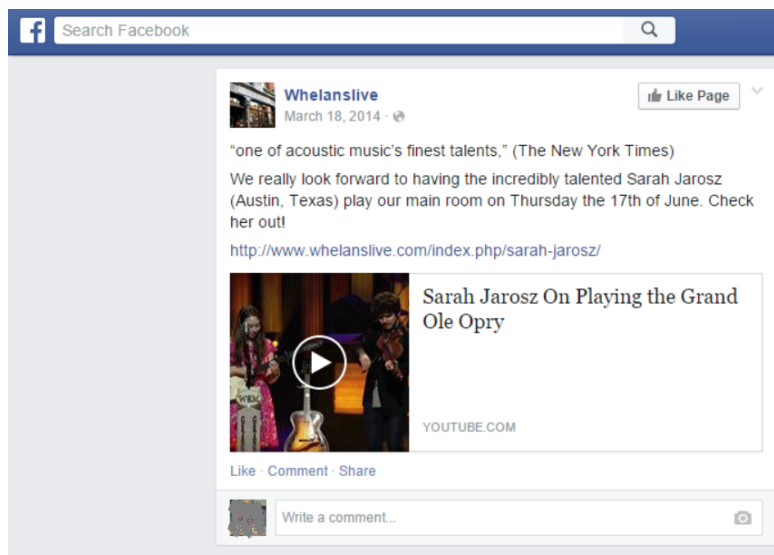


Figure 6.6.7 Example of a Facebook post before it is processed with NLP methods such as tokenisation.

With the usage of stemming, tokenisation, removal of stop words, short words and HTML, the text from the post in figure 6.6.7, can be seen preprocessed in the box below.

Processed Post

one acoustic music fine talent new york times we really look forward incredibly talented sarah jarosz austin texas play main room thursday june check

Now that the post is cleaned and only meaningful words are left, methods such as sentiment analysis and lexical diversity can be calculated.

Sentiment Analysis

Using the TextBlob API⁸ the sentiment of the post can be calculated. The range of the sentiment is from -1 to 1 (-1 being negative and 1 being positive). Applying sentiment analysis to the post in figure 6.6.7 results in a sentiment score of 0.32. Looking at the post it is not surprising that it scores above 0, as words such as *incredibly* are mentioned.

Lexical Diversity

Using the Facebook post above, the lexical diversity is calculated:

$$LD(post) = \frac{\#unique\ words}{\#total\ words} = \frac{24}{2} = 1 \quad (6.6.1)$$

Detecting Language

Since only English posts are considered in the scope of this project, a way of detecting if a post is written in English was necessary. The Python library *langdetect*⁹ provides the functionality to detect the language(s) of a text string. *Langdetect* works by using a Naive Bayesian filter with character n-grams¹⁰. The following text is taken from the Facebook post above.

The code snippet below produces the output `'en'`, which means that the library correctly predicts the language of the text to be English.

```
from langdetect import detect
detect("one acoustic music fine talent new york times we really look forward incredibly
       talented sarah jarosz austin texas play main room thursday june check")
>>> 'en'
```

6.6.3 Performing feature Selection

To decide which specific features are relevant for the different dependent variables and learning methods (classification and regression), feature selection is done.

6.6.3.1 Feature selection using the embedded method

Embedded feature selection, mentioned in section 4.3.3.3 does feature selection as a part of training a model.

The model to choose is the model that performs best on the data. In this project Random Forest model is the best performing model, see section 7.3.2. Feature selection using Random forest is done using the technique described in 5.2.1.

⁸<http://textblob.readthedocs.org/en/dev/>

⁹<https://pypi.python.org/pypi/langdetect/1.0.1>

¹⁰<http://www.slideshare.net/shuyo/language-detection-library-for-java>

This section does not answer the question: how to get more like, but rather what is important when trying to predict likes. In the following only the interesting parts of figure 6.6.8 will be explained.

6.6.3.2 Describing important features using regression

When predicting the dependent variable of posts, it is very clear that the average customer attributes are by far the most important when predicting new posts. All the other attributes only in a smaller way affects the dependent variable of a post.

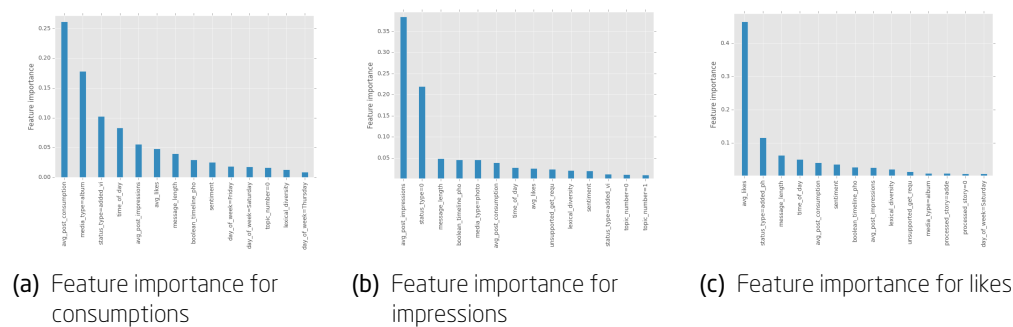


Figure 6.6.8 Feature importance for different dependent variables for regression

Consumptions: Interesting things to see from figure 6.6.9a, are that the album type is quite important regarding the prediction of consumptions. The reason for this could be that when a user looks through the pictures in an album, every click on a photo is added as a consumption to both the picture and the album. Another important thing for consumption is the status type added video. The added video attribute means that the post is a native Facebook video, this makes people click the video, maybe to stop the sound, play the sound, pause a video or play a video.

Impressions: The attribute status type o is very important for impression prediction. The status type o is when a post has not got a status type. These posts are normally cover photo updates, events or profile picture changes. Average number of impressions for posts with status type that is o is 12143, if the status type is not o, the average number of impressions is 38488. Another important factor is boolean timeline photo. This album contains all posts that are added to the timeline. It makes good sense that these photos get more impressions. Average number of impressions for photos in the timeline album is 38666 and average for photos not in this album is 32223.

Likes: Having added photos means that the post is a photo and has been added to an album. These posts have in average of 531 likes, whereas posts without the added photos attribute have 173 likes on average. Message length, time of day, and sentiment has a smaller importance on likes and is what would be expected to be important for likes.

6.7 Clustering

If a customer has few posts, it can be difficult to do a correct prediction of the post's performance. This is mainly due to the fact, that the model has not learned enough about the customer, and the variance of the prediction will likely be high. Therefore an attempt to solve this issue was to cluster customers based on their average number of likes, consumption and impressions. If a customer has few posts, instead of trying to predict the number of likes for that customer, an attempt is made to find another customer that is similar (in the same cluster) and do prediction using this customer instead.

This was at least the goal of clustering. The clustering should be done when considering likes, impressions and consumption. Doing, for example, clustering of the text a customer has written, would lead to clusters that represent pages that write about the same topics. The problem with this is that two Facebook pages that write about the same topics might have hugely different average number of likes, consumptions and impressions. As this thesis is about predicting successfulness of a Facebook post, which is defined as likes, impressions and consumptions, this is what the clustering algorithm should try to contain in the clusters.

Agglomerative hierarchical and k-means clustering is done for customers using average likes, average consumptions and average impressions. Finding the right amount of clusters is a difficult problem, that is solved by simply looking at the clusters created, and see if they make sense. For this problem, 10 clusters were found to make good sense. If the number of clusters is increased, a lot of clusters turns out with only one customer, which did not help to solve the original problem.

The results of agglomerative clustering can be seen in appendix D.1 and the results of k-means can be seen in appendix D.2. Here the number of customers in the different clusters are shown together with 3 sample customers from a cluster. For every customer, the number of average likes, average consumptions and average impressions are shown. There are still some big differences internally in the clusters, and the customers are not quite as similar as was hoped for. In e.g. appendix D.2, there are two companies where one company has average likes 1, consumption 19 and impressions 470, compared with a company in the same cluster with 89 average likes, 1317 consumptions and 4444 average impressions. The difference between companies is one reason why clustering could not be taken into account when training a model. Another reason was that even though two companies have the same kind of average likes, consumptions and impressions, they might behave very differently on social media.

6.8 Choosing What to Predict

Falcon Social wanted to predict the amount of engagement a social media post would receive, as written in their initial project suggestion (see appendix A). This was later scoped to specifically predicting amount of engagement for Facebook posts. Engagement is not a specific metric, and therefore defining it can be difficult.

6.8.1 Metrics for post engagement

There are many different metrics when defining the engagement of a post. Some examples are: number of likes, comments and shares, click through rate, clicks ¹¹, organic reach, reach, impressions, a specific action (go to a website, buy a product), frequency of clicks, and so on. ¹². These metrics are all quite different and can affect the way engagement is calculated significantly. For some companies lots of likes on a post is a success, whereas for other companies reach is the single most important thing. Even for a specific customer, posts would be measured differently. One post could be created to get the brand name out to as many people as possible, here high reach would be a success. Another post for the same company could be created to get feedback on the product, here number of comments would be important in terms of engagement.

6.8.2 Different company sizes

Customers have huge differences in terms of followers on Facebook. For one customer a successful post will get 20 likes, but for another company 1000 likes for a post would be successful. This needs to be taken into account when training a model.

6.8.2.1 Engagement score

To take the size of a company into account, the first approach was to create a dependent variable that was dependent on both followers, likes, comments and shares. This dependent variable that translates into a score of successfulness, will be between 0 and 1. Where 0 is an unsuccessful (low engagement) post, 0.5 is a post that has average engagement and 1 is a post with very high engagement.

A score that is widely used ¹³ is:

$$Engagement(post) = \frac{comments + likes + shares}{followers} \quad (6.8.1)$$

This way of scoring engagement is a simplified metric, that takes a complicated problem and over simplifies it. Here is an example of this oversimplification:

Lets say Carlsberg, a customer of Falcon Social, creates a new post that is marketed on its Danish Facebook page. It has some text and a link to a new beer flavour. The post intrigues lots of people to click the link (1000 clicks), and see what this new flavour is all about. A few people like the post(10), and it gets one comment mentioning that the new flavour looks awesome. Calculating the engagement score: $Engagement(post) = (1 + 10 + 0) / 100767 = 0.000109$. In this example consumption is not taken into account, and therefore the engagement is wrongly represented. A solution to this problem would be to include consumptions, but

¹¹<http://www.socialmediaexaminer.com/facebook-page-metrics/>

¹²<http://www.jonloomer.com/2013/09/10/facebook-ads-metrics/>

¹³<http://simplymeasured.com/blog/2014/02/19/facebook-engagement-rate/>

then what about impressions, reach, viral reach, etc.? Another problem is the importance of a metric in an engagement calculation. It is more casual to like a post than sharing it and posts gets more likes than shares. But how important is a share compared to a like? Maybe this is also dependent of the Facebook page. For every new metric that is added to the calculation of the dependent variable, more variation is introduced to the data and the harder it will be to learn and evaluate the underlying truth. There are many different metrics regarding a post and many different ways of calculating the engagement for a post^{14 15 16}. The notion of one engagement score was dropped, for the reasons mentions above.

6.8.2.2 Relative dependent variable

Instead of trying to solve all the problems in one engagement score, another approach was to create different dependent variables, that took the number of followers of a Facebook page into account. A metric where likes, comments, and shares are taken relative to the followers of the page. This metric was a lot more straightforward:

$$relative_likes(post) = \frac{likes}{followers} \quad (6.8.2)$$

$$relative_comments(post) = \frac{comments}{followers} \quad (6.8.3)$$

$$relative_shares(post) = \frac{shares}{followers} \quad (6.8.4)$$

There is a very important assumption in this simple equation: Followers and number of likes, shares and comments follow a linear trend. The more followers the more likes, shares and comments.

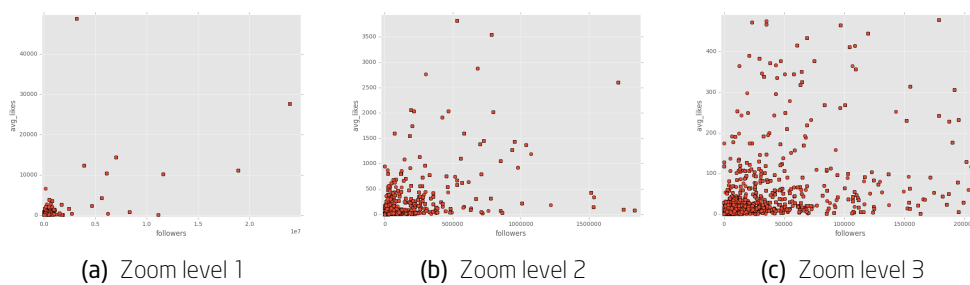


Figure 6.8.1 Posts plottet with average number of likes for a company and the number of followers

When looking closer at figure 6.8.1, it was found that some companies had huge number of followers, but a small number of average likes. This phenomenon needed further investigation.

¹⁴<https://www.linkedin.com/pulse/20140711022232-33996286-here-s-the-correct-formula-to-calculate-your-facebook-post-engagement-rate>

¹⁵<http://simplymeasured.com/blog/2013/08/14/facebook-metrics-defined-engagement-rate/>

¹⁶<http://www.socialbakers.com/blog/467-formulas-revealed-the-facebook-and-twitter-engagement-rate>

An example of this was the Facebook page Mentos Poland. They had 11.617.372 followers, but average number of likes was 28. Going to the global Mentos page, showed that they also had 11.617.372 followers. Brands with more than one Facebook page, can set up multiple pages for individual locations. This is called Facebook page parent-child structure¹⁷. For child pages of a brand, the like count shown is the aggregated number of followers of all attached pages. This is a problem, as child pages will have huge number of followers, not reflecting the number of people actually liking the specific page. Therefore the regional fan count is extracted from the Facebook insights API¹⁸. This can easily be done for child pages, but for parent pages this is not possible. Therefore parent pages are not considered in this project.

6.8.2.3 The problem of relative likes

From the figures at 6.8.1 it can be seen that the relationship between average number of likes and the number of followers is not linear, even though there is a positive correlation at 0.8378. To highlight the problem, a small example of two companies will be explained:

Looking at figure 6.8.1a, it can be seen in the right bottom corner that there are some companies with more than 5.000.000 likes, but a very low number of average likes. An example of a company with many followers but low average of likes is the official Egyptian Facebook page for Samsung: Samsung Egypt. They have 6.736.429 followers on their page and an average number of likes at 320. An example of a company with many followers and many average number of likes, is Jaguar. They have 6.747.584 number of likes, but 10338 number of average likes. Lets compare a post that gets 4000 number of likes with Samsung Egypt and Jaguar:

$$relative_likes_{samsung} = \frac{4000}{6736429} = 0.000593786411168 \quad (6.8.5)$$

$$relative_likes_{jaguar} = \frac{4000}{6747584} = 0.00059280477279 \quad (6.8.6)$$

For Jaguar 400 number of likes is a post that under performs compared to the rest of the posts. For Samsung Egypt a post that gets 4000 likes is a post that is performing very well. Looking at their individual relative likes, these two posts looks like they perform equally well, even though this is not the case. This is one reason relative likes was not chosen as a dependent variable.

Predicting relative likes as opposed to likes, results in a MdAPE value that is a lot worse, compared to predicting likes. The table in appendix E, shows that the best model for predicting relative likes, has a MdAPE value of 62.5 % and likes has a MdAPE value of 50.18 %. Not only does predicting likes with the context of the customer make a lot of sense, it also performs a lot better.

¹⁷<http://startups.co.uk/facebook-for-franchises-what-is-a-parent-child-structure/>

¹⁸https://developers.facebook.com/docs/graph-api/reference/v2.2/insights?locale=da_DK

6.8.2.4 Solution to different sizes of companies

Going back to the original use case of the problem: predicting the amount of engagement, a decision was made to predict engagement on the three most important metrics: Likes, impressions and consumptions. These dependent variables were pointed out by both Helle Tyllesen and May Laursen to be important. For every dependent variable an individual model was created.

As discussed above the performance of a Facebook post is very dependent on the context of the page that it belongs to. This means that most of the features in the dataset cannot predict likes well on their own, but need the context of the customer. As written in section 4.3.3.4, these features are called context features.

To account for the customer context, six features were created: average number of likes, consumptions and impressions and the variance of the likes, consumptions and impressions. When predicting the dependent variable, the average and variation of likes, consumptions and impressions make sure that the algorithm knows in what area of likes, consumptions and impressions it should predict. This mechanism ensures that the company size is taken into account in the context of how well a post normally performs.

Another solution was to include a feature that indicated which customer a post belonged to. This means that the number of features would be very big and the dataset very sparse. This creates some problems for some of the models, especially nearest neighbour has problems in high dimensions [Marimont and Shapiro, 1978]. Another problem is that some customers have many posts and some customers very few. For the customer that has many posts, the feature indicating this customer will be an important feature in the trained tree. For a small customer this feature will not be very important and might not even be considered when training a decision tree. Therefore the context is added in the form of the customer averages and variances.

6.8.3 Classification

When doing classification the dependent variable has to be a set of specified labels. The labels in this project are created by checking if a posts number of likes/consumptions/impressions is larger or smaller compared to the customer median of likes/impressions/consumptions. The reason the median is chosen and not the mean is because the median is robust to outliers [Ripley, 2004], which the dataset is full of. As explained above, a new model is created for every type of dependent variable.

When classifying a new object, the goal is to classify the post as getting above or below the customer median in likes/impressions/consumptions. The reason for this is that social media managers are people who are very busy. They want quick indicators assisting them while creating a post. One of these indicators could inform them, that the post they are creating is below standard for a particular Facebook page. This would make them look further at the post, to see if anything could be tweaked, further improving the post.

6.9 Model Selection

The model selection will be based on a few parameters from each model, and the models will be evaluated on the same validation set. It is a variation of the Grid Search, but is in this case implemented for easy customisation.

6.9.1 Choosing the right model

For this project, a few models have been selected to perform the prediction for both classification and regression:

- Dummy - a model that performs simple guesses
- Random Forest
- Decision Tree
- K-Nearest Neighbours

Other models such as Support Vector Machines, Logistic Regression, and Naive Bayes also exist, and might be applicable as well. The reasoning behind choosing Random Forest and Decision Trees for both classification and regression in predictive analytics in collaboration with a company, first of all lies in their explainability - the features in top of the tree are the most important, and investigating provides insights that are easily explained to executives and other stakeholders.

K-Nearest Neighbours is mainly chosen in order to have a different model that is able to compare with the tree based regressors and classifiers. The dummy model, as explained in section 4.6.1, is used to perform simple guesses that validates the other models perform better than "random" or extremely simplified guesses.

6.9.2 Parameters

In order to choose the right model, the optimal parameters of the model must be found. Since the parameters will vary from each model and data set, an empirical approach seems like the easiest way to choose the parameters. For the models mentioned in the previous section, the following parameters are considered:

- Dummy - Strategy
- Random Forest - Number of trees and max depth (if the depth is not limited, the complexity and memory required by the model will increase heavily as the number of trees increases)
- Decision Tree - Again, max depth is varied and minimum split. Since Random Forest is an ensemble method, finding the right parameters for a tree may increase the performance of the Random Forest method.

- K-Nearest Neighbours - Number of neighbours and the leaf size.

The parameters should be chosen so that validation set errors are minimised while still maintaining a low training error level.

6.9.3 Training, Validation and Test

The target data set for this analysis will be separated into a training and test set, where it is split on a specific date, so that the training data is older than the test data.

When comparing the models, their performance on the validation set is measured to find the best one.

The test set is then applied on the model, which then denotes its final performance. Using this score eliminates any bias previously made when choosing the model.

The distribution of the sets are:

- Training: 51507 posts - 68,6 %
- Test: 12604 - 16,8 %
- Validation: 10918 posts - 14,6 %

The uneven percentages is a result of the way the data is split. Instead of just setting a fixed number, the training set contains values before 14. September 2014 (from January 1st), the test set starts after the training set and ends at November 11th. Finally, the validation set is from November 11th to December 31 st. This way, the test set will be the "future" - entirely separated from the training and validation of the models.

Chapter 7

Results and Evaluation

Contents

7.1	Results	89
7.2	Evaluation of models	94
7.3	Testing with unseen data	96

7.1 Results

All the models previously mentioned in model selection 6.9 have been trained and validated with several parameter ranges - and a combination of those, this method is also known as Grid search explained in section 4.5.1. This procedure was automatised and ended up with 3644 classification models and 3632 regression models. They were scored with relevant metrics defined in 4.6 for three dependent variables: likes, consumptions and impressions.

This section will provide an overview of how the models perform. The models have used the same data set for classification and regression.

7.1.1 Classification

For classification, three variables were chosen for prediction. The measures for likes, impressions and consumptions were divided into two categories: above or below average for the customer who made the post. That means in theory a classifier should be able to correctly guess the label of the categories 50% of the time (assuming 50/50 distribution of the labels in the test set).

For each combination of parameters a model was fitted to the training set, and chosen with the validation set. The parameters were iterated in the following way:

- Dummy: Stratgy - Uniform, constant, most frequent and stratified
- Decision Tree: Max depth from 2-19 and minimum sample split from 1-40
- RandomForest: Number of trees/estimators from 10 to 200 by 10, and max depth from 1-19
- KNeighbors: K-neighbors from 1-49 and minimum leaf size in range 1-3

Model	Error:Validation
Post.likes_label	
DecisionTreeClassifier	0.310569364
DummyClassifier	0.38299055
KNeighborsClassifier	0.386246327
RandomForestClassifier	0.298737394
Post.consumptions_label	
DecisionTreeClassifier	0.341432428
DummyClassifier	0.454025727
KNeighborsClassifier	0.406304589
RandomForestClassifier	0.34230586
Post.impressions_label	
DecisionTreeClassifier	0.408996318
DummyClassifier	0.504962382
KNeighborsClassifier	0.457099408
RandomForestClassifier	0.415559469

Table 7.1.1 The best performing error rate for the validation set for each model. A total of 3642 models were evaluated to predict likes, impressions and consumptions.

The results of the classification can be seen in table 7.1.1, where an overview of the performance of each classifier is given. A sanity check for the predictions can be made in relation to the DummyClassifier. As previously mentioned, guessing randomly should score around 50 %, which more or less holds for consumptions and impressions. For likes, however, the score is 38.3 % which is the score of the DummyClassifier when predicting likes. The best strategy for the DummyClassifier in this case, is guessing on the most frequent class. Using that strategy, the accuracy will match the distribution of the class. As previously seen, there are a lot of posts with 0 likes, and there are some posts that have gone viral. This influences the number of posts below the average.

It can be seen in table 7.1.1 that the KNeighborsClassifier performs worse than the DummyClassifier when predicting likes. The difference between the two models is 0.4%, in favour of the DummyClassifier. For consumption and impressions, the KNeighborsClassifier performs better. If the distributions between the classes had been more even for likes, as it is the case with both consumptions and impressions, the performance of the DummyClassifier most likely would have performed worse.

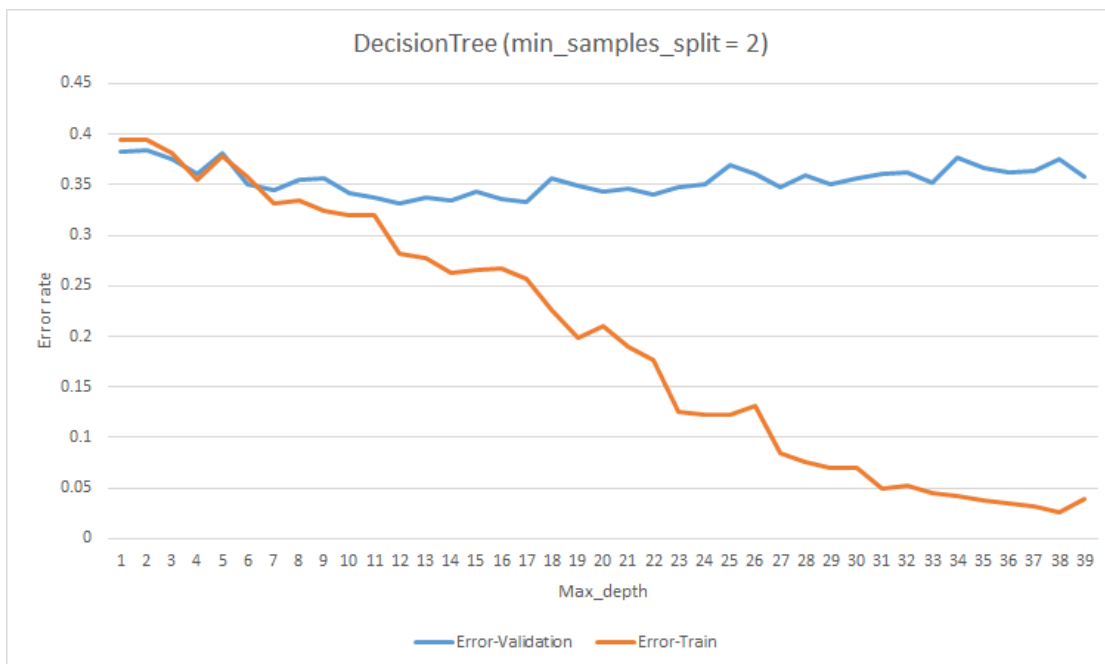


Figure 7.1.1 The learning curve for a Decision Tree with the default setting for minimum samples split in the tree, where it tries to predict the likes label. The optimal max depth seems to be around 17.

The DecisionTreeClassifier outperforms the DummyClassifier on all the dependent variables. It can be seen from its learning curve on figure 7.1.1 that if the max depth gets too high (i.e. the complexity of the model increases), the training error get too low - which means it overfits. The optimal point for max depth is in the range 10-17. For every point after that the training errors are too low and the test errors start to increase. Looking at figure 4.5.4, this follows the theory of overfitting data. The error rate will most likely not reach much higher than random (for likes that is 38.3 %).

Random Forest is influenced by the trees trained in the model, and the performance of Random Forest will be heavily influenced by the trees in the forest. Therefore, the optimal level of complexity in the DecisionTreeClassifier can probably help increasing performance of the RandomForestClassifier.

The learning curves can be inspected for the RandomForestClassifier in figure 7.1.2 and 7.1.3. In the first learning curve, the RandomForestClassifier with 170 estimators seem to decrease in training error when the max depth increases - the overfitting here is a result of the individual decision tree depths.

In figure 7.1.3 the max depth is set to 17, and the number of estimators/trees are varied, and it can easily be seen that adding to the number of trees does not change much in respect to both training and validation error.

To summarise this section, the best models based on the validation set are:

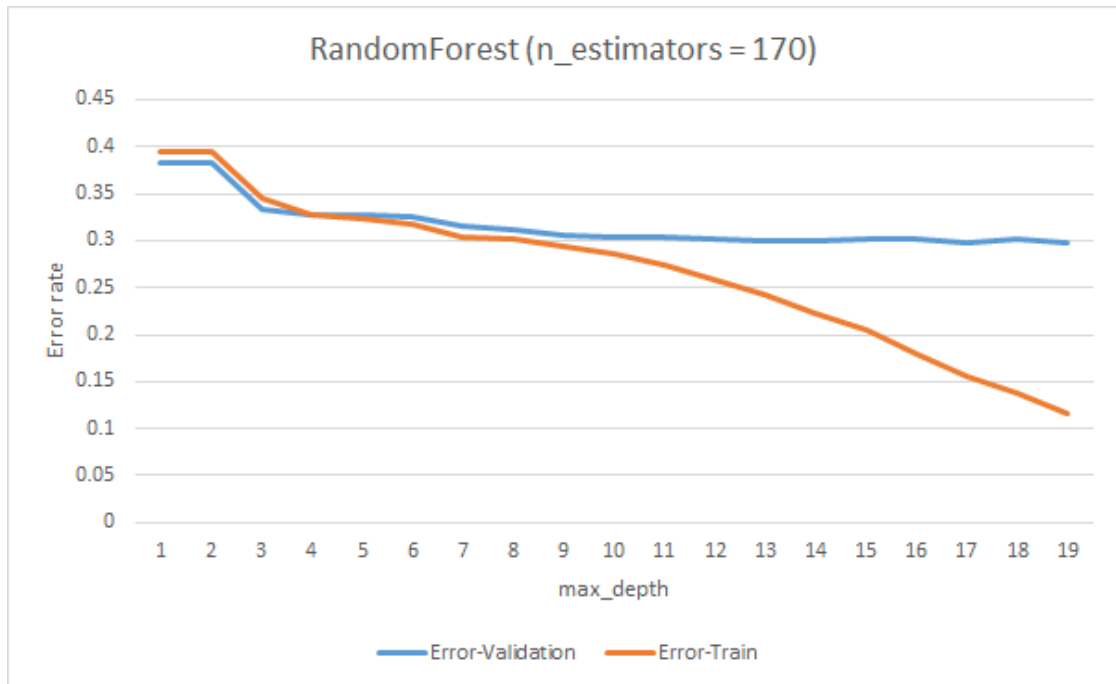


Figure 7.1.2 The learning curve for Random Forest with 180 trees. The curve indicates that the validation error doesn't increase in nearly the same rate as the training error decreases.

- Consumptions: RandomForest (34.2 %) and DecisionTree (34.1 %)
- Impressions: DecisionTree (40.9 %) and RandomForest (41.6 %)
- Likes: RandomForest (29.9 %) and DecisionTree (31.1 %)

7.1.2 Regression

Regression seeks to predict the same dependent variables as classification, although they are here the continuous values instead the "above/below" labels used in classification. The variables are consumptions, impressions and likes.

The models are evaluated by comparing the mean squared error (MSE), median absolute percentage error (MdAPE) and coefficient of determination (R^2) of the validation set. The same training and validation data as in classification is applied.

Choosing the best models should be done carefully. An evaluation metric such as R^2 might not tell the full story, as a bad fit might systematically over- and under-predict data. MdAPE has proven to handle outliers well and it is fairly reliable[Armstrong and Collopy, 1992], and the best models will therefore be

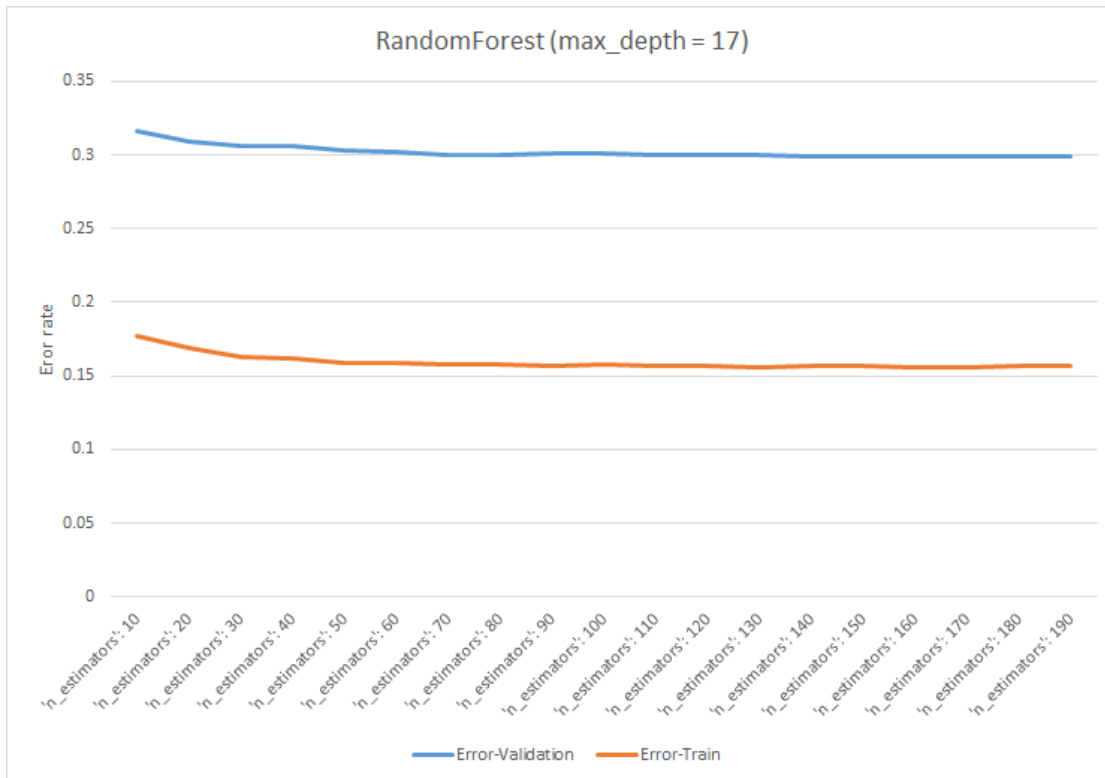


Figure 7.1.3 For a Random Forest with maximum depth of 17, the error rate does seem to decrease with number of estimators in the forest.

chosen using that as performance metric. MdAPE can prove to be useful due to the data set containing some significant outliers (as described in section 6.5.3)

The RandomForestRegressor scores 50.15 % and the DecisionTreeRegressor 50.90 % for the prediction of likes. RandomForestRegressor also scores the highest on the remaining two categories, consumptions and impressions, and DecisionTreeRegressor is scoring close to the same.

In order to decide which model to chose, the evaluation metric proposed in section 4.6.4.2 is used. The models are chosen from table 7.1.2.

Again, DecisionTree and RandomForest perform significantly better than KNeighbours, and the best performing models for each metric, based on the validation set, are:

- Consumptions: RandomForest and DecisionTree
- Impressions: RandomForest and DecisionTree
- Likes: RandomForest and DecisionTree

Prediction	MSE	R2	MdAPE
Insight.post_consumptions			
DecisionTreeRegressor	37758087	0.67	67.58
DummyRegressor	114897789	0.00	93.41
KNeighborsRegressor	65373872	0.43	73.20
RandomForestRegressor	35266584	0.69	67.57
Insight.post_impressions			
DecisionTreeRegressor	14432146087	0.81	51.45
DummyRegressor	76613245871	0.00	92.79
KNeighborsRegressor	25646800292	0.67	54.41
RandomForestRegressor	14103250864	0.82	50.31
Post.likes			
DecisionTreeRegressor	2770125	0.63	50.90
DummyRegressor	7397318	0.00	87.18
KNeighborsRegressor	3034543	0.59	60.00
RandomForestRegressor	1948348	0.74	50.18

Table 7.1.2 Results for the regression models, where the models are compared against each other with Mean Squared Error MSE, the Coefficient of Determination R2 and Median Absolute Percentage Error MdAPE

7.2 Evaluation of models

This section seeks to compare the best performing models for each dependent variable.

7.2.1 Classification

In order to choose the best model for predicting consumptions, impressions and likes the closely performing models must be evaluated. Previously in section 7 a Z-test was suggested for evaluation of a model containing a high number of samples N. In this case, N is 12605, which can be considered a large number.

R is used to calculate the Z-test with a confidence interval of 99 % and the null hypothesis that the two models are equally good.

Consumptions

Random Forest and Decision Tree are evaluated against each other. The output from the statistical software R is shown below:

```
prop.test(x,n = c(N, N),alternative = "greater", conf.level = alpha, correct=FALSE)

2-sample test for equality of proportions without
continuity correction

data: x out of c(N, N)
```

```
X-squared = 0.0908, df = 1, p-value = 0.3816
alternative hypothesis: greater
99 percent confidence interval:
 -0.01209393 1.00000000
sample estimates:
prop 1 prop 2
0.6586 0.6568
```

The test reveals that the p-value is 0.3816, and it is therefore not possible to reject the null hypothesis - the models does not perform significantly different.

Impressions

In the Z-test below, the null hypothesis can't be rejected in the prediction of impressions:

```
prop.test(x,n = c(N, N),alternative = "greater", conf.level = alpha, correct=FALSE)

2-sample test for equality of proportions without
continuity correction

data: x out of c(N, N)
X-squared = 1.1574, df = 1, p-value = 0.141
alternative hypothesis: greater
99 percent confidence interval:
 -0.007787902 1.000000000
sample estimates:
prop 1 prop 2
0.5911 0.5844
```

The p-value is 0.141 as above 0.01.

Likes

The prediction of likes has a bit higher difference, and for the same test the results are:

```
prop.test(x,n = c(N, N),alternative = "greater", conf.level = alpha, correct=FALSE)

2-sample test for equality of proportions without
continuity correction

data: x out of c(N, N)
X-squared = 4.2505, df = 1, p-value = 0.01962
alternative hypothesis: greater
99 percent confidence interval:
 -0.001526516 1.000000000
sample estimates:
prop 1 prop 2
0.7013 0.6894
```

Here the p-value (0.01962) is also higher than 0.01, although it is very close.

To summarise the tests, the models used to predict each dependent variable are:

Likes: Both Random Forest and Decision Tree

Consumption: Both Random Forest and Decision Tree

Impressions: Both Random Forest and Decision Tree

For likes, impressions and consumptions it was not possible to tell if Decision Tree or Random Forest had the best performing model.

The K-Nearest Neighbours algorithm did, however, perform poorly on all the before-mentioned predictions.

7.2.2 Regression

Choosing the best model for regression is done using a t-test with the assumption that the mean of their residuals subtracted from each other is 0 (as explained in 4.6.4.2). The p-value used is 0.01, corresponding to a 99 % confidence interval.

The test concluded the following:

Likes

The t-test scored 0.47, which is much higher than 0.01. The null hypothesis could therefore not be rejected.

Consumptions

The t-test scored the p-value 0.03, which is above the accepted p-value the null hypothesis cannot be rejected.

Impressions

The t-test yielded the p-value 0.82, which is above the threshold of 0.01. The null hypothesis can therefore not be rejected.

In all three categories it could not significantly be decided if a forest performed better than a single tree.

7.3 Testing with unseen data

For testing purposes, new and unseen data set is used. It contains 10914 posts and spans over the course of 2 months - November and December 2014.

7.3.1 Classification

In section 7.1.1, it was seen that Random Forest and Decision Tree performed equally good when predicting likes, consumptions and impressions. When verifying with the test set, Random Forest is chosen with the best validation parameter, rather than using Decision Trees.

Classification yields the following error rates for the test set:

- Likes: Random Forest 28.15 % (29.87 % on the validation set) (170 estimators, 17 depth of the trees and minimum samples split of 2)
- Consumptions: Random Forest 32.63 % (34.23 % on the validation set) (30 estimators, 10 depth of the trees and minimum samples split of 2)

- Impressions: Random Forest 43.16 % (41.56 % on the validation set)(50 estimators, 19 depth of the trees and minimum samples split of 2)

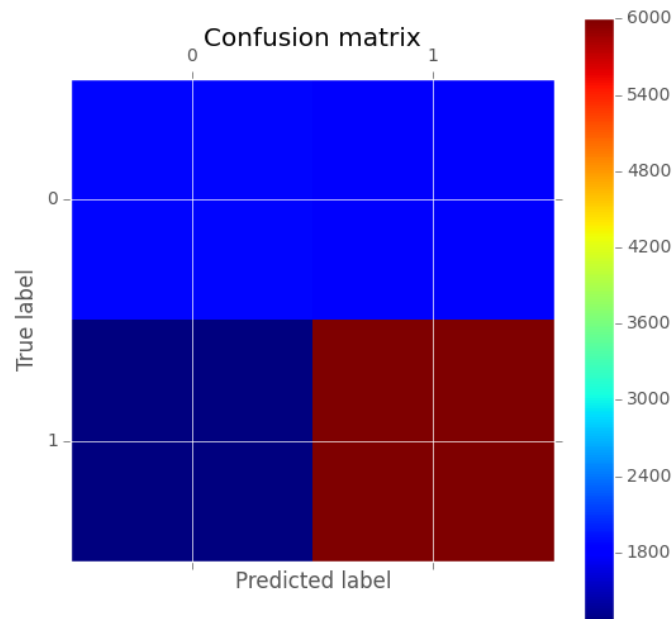


Figure 7.3.1 Confusion matrix for the test set when trying to predict likes above or below average.

The test set consisted of 7256 values above average (label 1) and 3658 below average (label 0). The average is based on the previous posts from the training data. Likes from the validation and test sets are not included when calculating the average. That way, the test set hold no hidden information about the future. Classification of likes generated the confusion matrix in figure 7.3.1. The matrix reveals about 6000 posts are correctly labelled as label 1 (in table 7.3.1 it can be seen that the value is actually 6003). Therefore 1253 are wrongly labelled as label 0, corresponding to only 17 % error rate or 83 % success rate. The label 0 success rate is only 51.1 %. It is an interesting discovery that misclassification of below average labels is so high. One could argue this might be due to the fact that badly performing posts tend to be of a more random nature, rather than above average posts with good content. Guessing on the most frequent value (above average) would have yielded only 66.5 % success rate - compared to the 71.85 % of the Random Forest model.

		Likes	
		0	1
True	0	1877	1781
	1	1253	6003
		Predicted	

Table 7.3.1 Confusion matrix for likes, where the number classifications are plotted against their true values.

7.3.2 Regression

On all categories, Random Forest is better than or significantly equal to Decision Tree in terms of MdAPE. It was not possible, based on the errors of the predictions on the validation set, to decide which model performed best.

Since either of the two models can be picked, Random Forest is chosen to be tested on the unseen data. Random Forest models yielded the following MdAPE scores:

- Likes: 52.80 % (110 estimators, 10 depth of the trees and minimum samples split of 2)
- Consumptions: 74.77 % (100 estimators, 19 depth of the trees and minimum samples split of 2)
- Impressions: 54.07 % (60 estimators, 19 depth of the trees and minimum samples split of 2)

7.3.3 Weighting posts by time

Social media is changing as written in section 2.3. Strategies that worked 6 months ago, might now be outdated. This means that the underlying truth regarding getting a successful post with older posts has changed. Therefore instead of saying that each post is equally important to the prediction, weights are assigned to each post. This ensures that posts that are younger, have a larger weight compared to older posts.

To apply weights, the best performing learning algorithm, has been chosen which is the random forest regressor. The random forest regressor can be supplied with a list of weights which are assigned to each post. A simple linear weight strategy has been applied, after the posts has been sorted regarding date:

$$Weight(post_i) = \frac{i}{N} \quad (7.3.1)$$

Where i is the number of posts in sorted order according to the post date. N is number of posts.

This is not an optimal way of doing post weighting, but gives an indication if adding post weighting improves the model.

7.3. Testing with unseen data

Predicting likes, improved the test error level slightly. Compared to predicting likes without weighting the MdAPE value was 52.80 %, with the weighting the MdAPE value is 50.64 %. This is only a slight improvement, but still shows that adding weights might further improve the model, especially when improving the way the weighting is created.

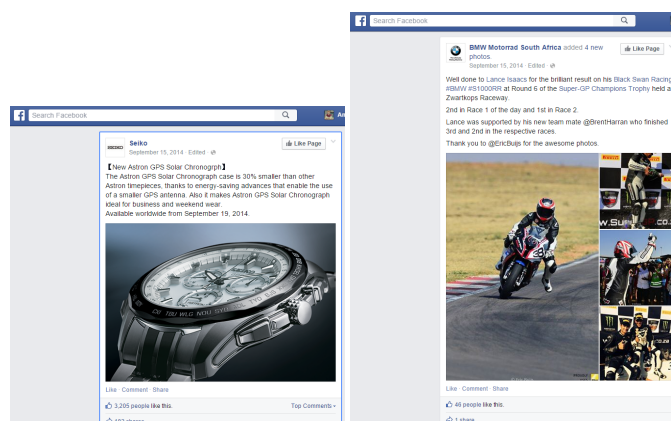
7.3.4 Example Predictions

While evaluating the overall performance of the model, some practical examples is a good way to get an overview of how the model performs. There will always be some posts that are easily predictable - and some that entirely misses the target.

This section takes a closer look at two posts - one where the prediction is correct and another where the prediction is far off.

Prediction of likes	Actual	Customer Average	Prediction	Percentage Error
Bad prediction (a)	3208	775	523	513 %
Good prediction (b)	46	44	46	0 %

Table 7.3.2 Example of a good and a bad prediction done by the model.



- (a) This post got 3208 likes and the brand normally gets 775 likes. This is a really successful post, but the prediction was only 523.
- (b) The post from BMW was correctly predicted to get 46 likes. BMW gets 46 likes on average, suggesting that posts around its average are easier to predict.

Figure 7.3.2 Example of predictions of Facebook posts

The two posts presented in figure 7.3.2 are posts from two different brands. In the post from Seiko, a brand

new watch is marketed and the post receives 3208 (the picture says 3205, but the models were trained on 3208 - an insignificant difference for this example). Seiko normally get 755 likes on average. In the post from BMW, the prediction hits the exact number of likes. The feature importance for likes shown in figure 6.6.9c shows that the number of likes a brand achieves on average is the most important when predicting likes. This could be a good explanation of the low error rate for the BMW post.

Chapter 8

Describing the Product

Falcon Social had the goal of helping their customers create better social media posts. They were interested in getting a tool that both tried to predict how well a social media post performed, but also gave suggestions regarding further improving a social media post. In this thesis a web service has been implemented. To demonstrate the power of our thesis, a website that uses this web service has been created.

8.1 RESTful web service

Falcon Social wanted to keep the integration into their platform minimal. Interviews were created with people working at Falcon Social, who had created their own tools that were integrated up against their social platform. They all recommended keeping our tool loosely integrated with Falcon Socials platform. This means that the web service created would run on its own server using its own database, web server and technology. The people interviewed also recommended implementing the service as a RESTful web service. This ensures that integration is simple and quick, opposed to SOAP based web services, that are both claimed dead, but also more complex to implement and integrate ¹

The RESTful web service can be accessed at url localhost/post_predict. The input to the web service is a JSON object (send by a post request) containing information regarding a post and returns a JSON object that contains data regarding prediction, suggestions and prediction over time. An example of the input and output can be seen in the two boxes below.

The reason the post request was chosen, was that this type of HTTP method asks the server to create some new resource/object on the server. The new resource that is created is a new training object, based on the post that is send to the web service.

```
Object:{  
  message: "test message",  
  mediaType: "link",
```

¹<http://www.javaworld.com/article/2071222/java-app-dev/web-services-are-dead---long-live-rest.html>

```
dateTime: "08/01/2015 16:37:37",
customerName:"Villasport The Woodlands",
dependentVariable:"consumptions",
}
```

Listing 8.1.1 Example of input JSON object

```
Object:{
  dateGraph: [
    [
      "2015-01-08 16:00:00",
      20,
    ],
    [
      "2015-01-08 17:00:00",
      17,
    ],
    ...
  ],
  prediction: {
    coefficient_of_variance: 71,
    lower_bound: 9,
    predicted_dependent_variable: 20,
    upper_bound: 31,
  }
  suggestions: [
    {
      executed: false,
      increase: "22.5",
      suggestion_name: "Try asking a question",
    },
    {
      executed: true,
      increase: "0.0",
      suggestion_name: "Try mentioning a page",
    },
    {
      executed: false,
      increase: "13.7",
      suggestion_name: "Write eleven words",
    }
  ]
}
```

Listing 8.1.2 Example of output JSON object

The web service consists of three components: Current prediction, predictions over time and the improvement suggestions for the post. These components will be explained in the next section.

8.1.1 Current prediction

The current prediction is the prediction of the amount of likes/impressions/consumptions. This amount is given with a lower and an upper bound, defining the 95% confidence interval.

The suggestions are created by initially receiving the kind of variable to predict. The dependent variable decides which model to load. When the correct model is loaded, the post defined by the attributes in 8.1.1, is transformed into an object with the correct features for the model to predict.

The upper and lower bounds are calculated using the different trees/estimators for the loaded model. These estimators are the decision trees that are trained in the data as described in 5.2. For every estimator in the forest, the prediction is done. If the random forest prediction is uncertain, the individual trees's predictions will have a large variance. With the predictions from each estimator, the 95% confidence interval and mean value are calculated. The mean value is the same as the predicted value from the model.

To have a uniform way of indicating how certain the estimators predict the dependent variable, the coefficient of variation is calculated. This coefficient is a standardised way of measuring relative variance ². It ensures that different distributions can be compared, which is needed for the different prediction's of posts.

8.1.2 Predictions over time

To help users decide what time to publish a post, the number of likes a post will get over a week are predicted. To make the predictions, all features of a post is kept the same, except the time of day, the day of the week and the time since last post was created. For every hour with starting point at the time the prediction is done, a new time of day, day of week and time since last is calculated. These feature values are used to create a new training object, that is predicted using the Random Forest classifier. The prediction and the time are stored and appended to a list that contains all times and their prediction over a week. This list can be seen in 8.1.2, at the dateGraph attribute.

8.1.3 Suggestions

To further help users improve a Facebook post suggestions are provided in the interface. These are created using the same method as the predictions over time.

The suggestions come in the following form:

- If the user should ask a question. Using the boolean_questionmark feature.
- If the user should mention a page. Using the boolean_mention feature.
- If the user should use a hashtag. Using the boolean_hashtag feature.

²http://www.ats.ucla.edu/stat/mult_pkg/faq/general/coefficient_of_variation.htm

- The number of words. Using the `message_length` feature.

For each of the first three boolean features, the boolean feature is negated, prediction is done and if the prediction is greater than the current prediction a suggestion is returned to the user.

For the message length suggestion a threshold is set to 10 words. This threshold defines how many words deviated from the actual number of words a user is willing to get suggestions about. If a user has written 23 words, the algorithm will go through the number of words from 13 - 33 and find the most optimal number of words to write. The reason for this is that the first user test showed, that simply suggesting the best number of words to write, did not help the user^{8.2.4.3}.

8.2 Describing the website

The website is built up using the RESTful web service described above in section 8.1. The architecture of the website can be seen in figure 8.2.1. The site is built up, so that the user first selects what is infrequently changed for a post. This is what should be predicted (likes, impressions or consumptions) and the media type to post. With these two attributes selected, a user can start writing a post.

To effectively help the user, suggestions are provided regarding increase in the predicted number of likes/impressions/consumptions as the user is writing a new post. The kind of suggestions that are provided is described in 8.1.3. The suggestions are provided next to the post, so that they can easily be seen.

8.2.1 Describing users and use cases

To get a better understanding of the users and what tasks they are trying to solve, a typical user and use cases are described here.

8.2.1.1 Users

The typical user of our website will be social media managers. They are characterised by being busy and having many different tasks to solve in a day³.

People that do not necessarily work full time with Social media will also be using this tool. All users will be proficient in computers, social media and the web.

³<https://blog.bufferapp.com/social-media-manager-schedule-checklist>

8.2.1.2 Use cases

The goal of this project is to help users improve a Facebook post while it is being written. Therefore the task which a user performs, is to create a Facebook post, select a time to post and revise the text written in the post. To be more precise, the user has to:

- Choose the variable that should be predicted.
- Choose the media type for post.
- Write the text in the post.
- Choose a date for post scheduling.
- Optimise a post given the feedback in the form of predictions over time and suggestions.

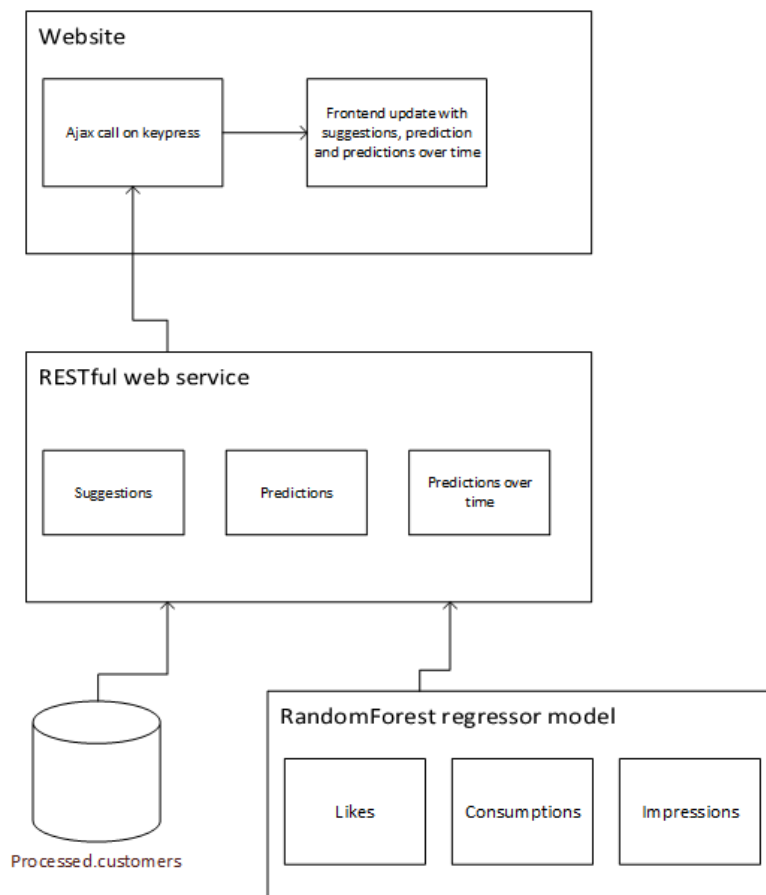


Figure 8.2.1 The architecture of the website

8.2.2 Live prediction

To ensure live prediction, the two following things are necessary:

- Ajax calls to the RESTful web service
- Low response times for the RESTful web service

Ajax

Ajax is a web development technique, that makes it possible for a page to send and retrieve data asynchronously to and from a server. This makes it possible for a user to interact with a page without refreshing a site. Ajax is typically used when interaction with a website is important.

In this website, Ajax is used every time a user interacts with the website. This includes changing the dependent variable, media type, time to post or writing a new character in the post. All these actions calls a Javascript method, that collects the current written post, and sends it to the RESTful web service, using jQuery's Ajax function. If the response from the web service is successful, the new suggestions, prediction and predictions over time are updated in the frontend.

Response time

The response time highly depends on the predictions over time. This is due to the fact, that for every hour over the course of a week, a prediction must be made. This adds up to $24 * 7 = 168$ predictions. In order to also display the upper and lower bound of the uncertainty, a prediction must be made for each estimator in the random forest as described in section 8.1.1. The number of total predictions is then $168 * \text{number of estimators}$. This affects the response time for the webservice. Table 8.2.1 shows response time regarding number of days to predict, for 10 estimators. The number of days to predict was chosen as 4, as users both get reasonable response time but also has the ability to predict a couple of days into the future.

Number of days to predict	3	4	5	6	7
Avg response time in seconds	0.1960	0.2069	0.2370	0.2400	0.2650

Table 8.2.1 Response times for a single Ajax call, given the number of days to predict

According to [Nielsen, 1993], response times between 0.1 and 1 seconds, need no special feedback, and is the limit of a user's flow of thought to stay uninterrupted. This is very important for live prediction. Otherwise the user will think too much about the slow feedback/prediction. Response times below 0.1 seconds make the user feel like the response is reacting instant.

8.2.3 Prototypes

With the users and use cases described in 8.2.1 identified, different sketches were created.

8.2. Describing the website

The first prototypes were low fidelity prototypes. The reason the low-fidelity prototypes were created to begin with, was to create multiple different design alternatives, making sure the functionality of the website was clear and get to think about how to represent and display different aspects of the prediction and suggestion.

Figure 8.2.2 shows some of the very early thoughts about which features to suggest to the user, how to visualize the suggestion of these features, how to visualize the actual prediction and the predicted likes over time. In the top left corner, an early prototype of the interface can be seen, with prediction over time, the actual prediction with a warning icon, some suggestions and a text box.

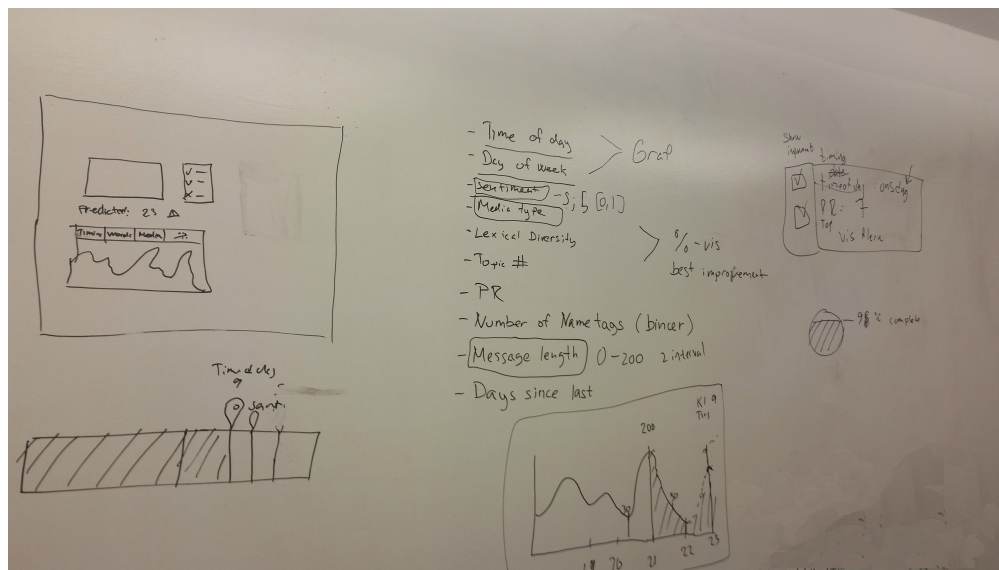


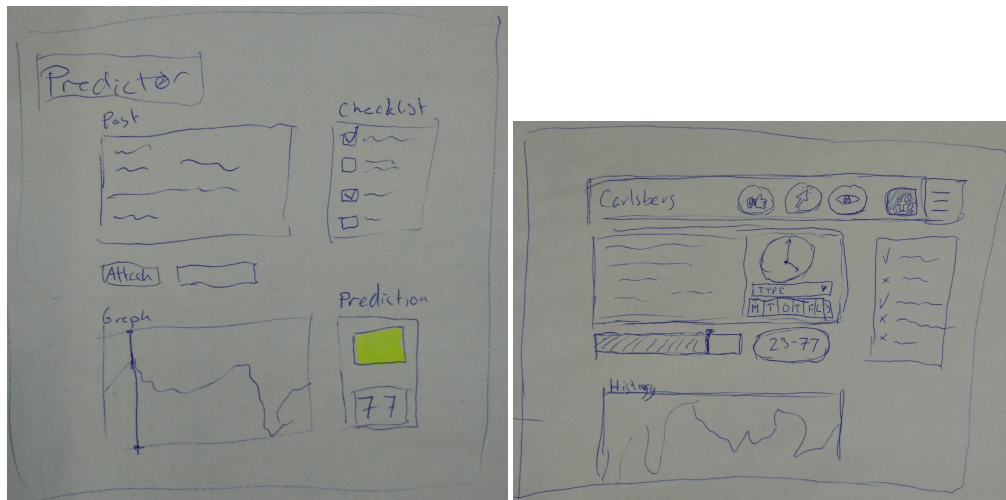
Figure 8.2.2 Early sketching

With a common goal of which machine learning features should be suggested and what the website should be able to do, 6 sketches, were created that can be seen below.

Sketch 1: (figure 8.2.4a) The thought was to show suggestions to improve in a checklist box. When a suggestion was checked, a new prediction would be created and seen in the prediction box. The attach button seen in sketch 2, was included to get the possibility to upload a picture. The graph shows predicted number of likes over time.

Sketch 2: (figure 8.2.4b) In this sketch, the date picker next to the text box, is shown in a more visual way. The suggestion box is to the right, but now the suggestions can not be chosen, they just show which suggestion has been met and which has not. The number of likes is shown in a bar and the uncertainty is shown as an interval next to it. In the bottom the prediction over time can be seen. In the top, the user chooses what will be predicted.

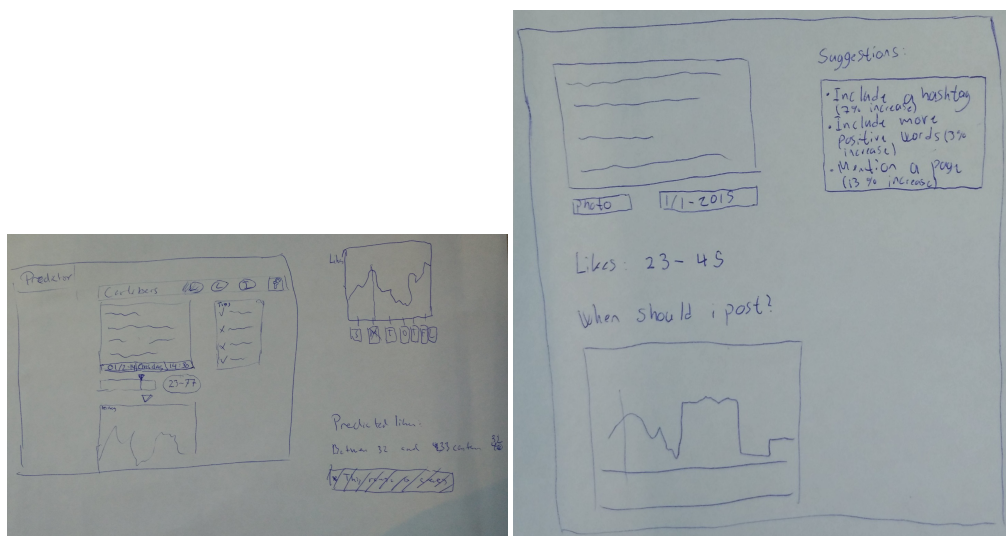
Sketch 3: (figure 8.2.6a) The date picker is slightly different from the last picker. The prediction over time



(a) Sketch 1

(b) Sketch 2

Figure 8.2.3 Prototypes



(a) Sketch 3

(b) Sketch 4

Figure 8.2.5 Prototypes

in the top right corner, can now be clicked to choose another day.

Sketch 4: (figure 8.2.6b) Suggestions are positioned next to the text box. The suggestions now also show increase if a suggestion is followed. Predicted likes is only shown as an interval. Prediction over time is below the predicted number of likes.

Sketch 5: (figure 8.2.8a) In the top the dependent variable is chosen. The suggestions regarding the post

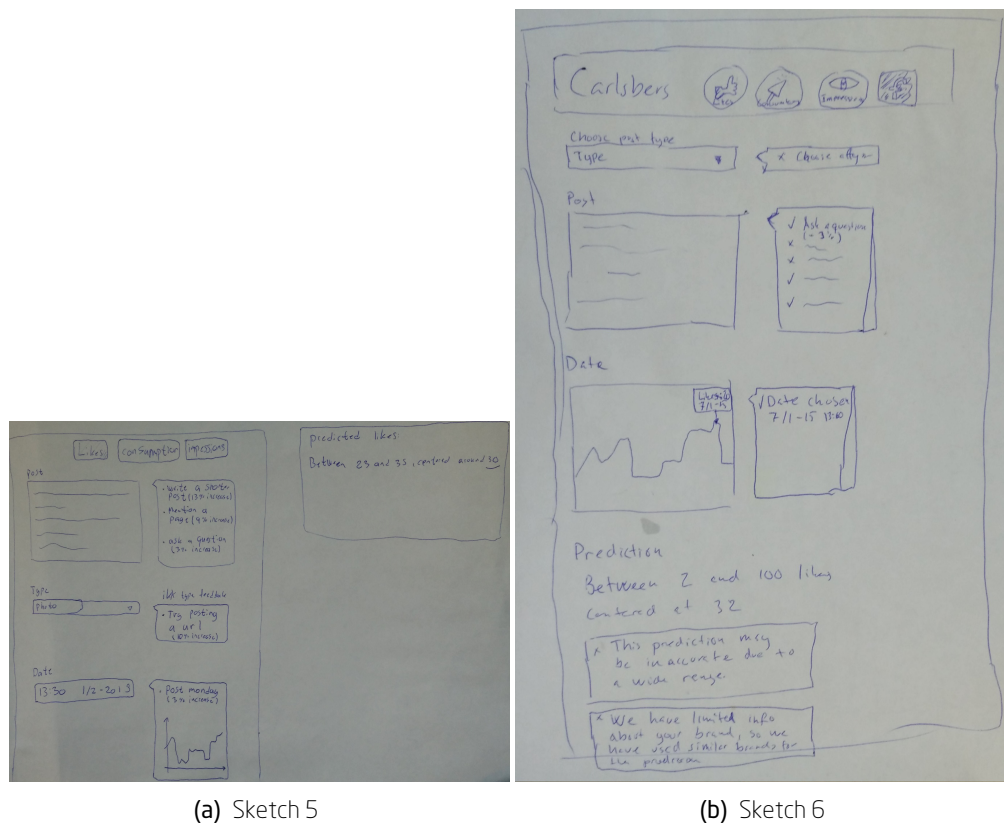


Figure 8.2.7 Prototypes

text is positioned next to the text box. Below the text box a dropdown to select media type can be found. Suggestions regarding the media type is placed next to the media type dropdown. Below the media type dropdown a date picker can be seen. Next to the date picker, suggestions regarding dates are shown. This includes a graph showing predictions over time. The prediction is displayed as an interval, but still showing the exact prediction.

With these different sketches, the best elements were chosen in order to create a new prototype - see figure 8.2.4b for details.

Sketch 6: (figure 8.2.8b) From prototype 3 the suggestion box chosen was positioned to the right of the text box that indicated which features had been met. The inspiration comes from the way forms are validated especially regarding passwords and usernames. An example can be seen at figure 8.2.9. This type of validation is called inline validation, where feedback is provided to a user in order for the user to change a password, username, etc. The inline validation is very successful regarding things like success rate, satisfaction rating, completion time, errors made and number of eye fixations[Treder, 2013]. Also from prototype 3 it was chosen to make the user select the dependent variable in the top of the page, since this is an element that will not often be changed, compared to the text and date chosen. From sketch 4 the

lower and the upper bound of the prediction as well as the exact predicted number of likes was chosen to be displayed. Including the upper and lower bound of the prediction ensures that the user can make an informed decision when deciding when to submit a post and what suggestions to include, based on the uncertainty. From sketch 5 the suggestions include the increase in percentage if a specific suggestion is to be met. In this way social media managers can decide if it is worth including a suggestion. Also from sketch 5 a dropdown was chosen for the media type.

Additional design changes in the final sketch was to put the media type after choosing what to predict, as this metric is not changed very often. Additionally, an inline validation that indicates that a media type should be chosen.

Inline validation is also done for the date picker. When a date is selected, the site displays the selected date with the following text: "Date chosen 15/02/2015 11:20:10".

In the bottom of the page, two text boxes shows up. One of them is indicating that the current prediction is uncertain. The other text box indicates that not many posts are available for a specific customer, which might result in a bad prediction.



The image shows a registration form titled "Join Twitter today." It contains two input fields. The first field is labeled "Full name" and contains the text "Marcin Treder". To its right is a green checkmark and the text "✓ Name looks great." The second field is labeled "Email address" and contains the text "marcin@uxpin.com". To its right is a red 'X' icon and the text "✗ This email is already registered. Want to login or recover your password?"

Figure 8.2.9 Inline form validation

8.2.4 Usability test

The reason a website was developed, was to test if the solution has potential for Falcon Social to integrate it into their solution. Part of building the website was to iteratively get user feedback from usability test from both peers, but also employees from Falcon Social. This would ensure the website that was created would provide the right kind of visual feedback in form of predictions, suggestions and predictions over time.

8.2.4.1 Thinking aloud

To perform the usability tests, the Thinking Aloud method was taken to use. This method is characterized by 3 things [Nielsen, 2012].

1. Choose representative users
2. Make them perform representative tasks on what is being tested.

3. Let the user think aloud and take notes of what happens or record the test.

The reason this method was chosen was that it is cheap, flexible and powerful. It provides some very good usability insights, that can be easily incorporated into developing in an iterative fashion.

8.2.4.2 User test preparation

The tests were performed in a relaxed atmosphere with people that matched the users described in section 8.2.1.1. The users were told to say everything they were thinking of out loud as they tested the website.

The tasks they had to accomplish, were based on the identified use cases, found in section 8.2.1.2. The tasks were the same through all iterations of the usability tests. The tasks they were told to do were:

Imagine you are a social media manager for Roskilde Festival. You have to write and schedule a new Facebook post for Roskilde. You can choose to create any post you want. Roskilde Festivals intentions is to create as good a post as possible.

- Write a draft for a Facebook post.
- Try follow some of the advice, do they make sense?
- Are they executable?
- Do they actually improve likes?
- How many likes do you think this post is going to get?
- What time would you post this post?
- Try selecting a time to post. Is this a good time to post?
- Now try to optimize a post regarding the suggestions.

8.2.4.3 User test 1

4 users tested the first prototype. These were students at DTU in the ages from 24 - 29, proficient with computer, the web and social media. The first test was mostly done to get feedback about the suggestions and prediction.

Insights learned from the user test:

1. Suggestion regarding number of words were not very useful. The reason for this was that the number of words that performed best, could be very far from the actual number of words written. For instance a user writing 10 words, but the suggestion suggests 55 words.
2. Predicted number of likes was not overt for the user.

Roskilde Festival

Choose dependent variable

Likes ▼

Choose media type

album ▼

Write your message

- Try mentioning a page (27.3% increase)
- Write 35 words (120.3% increase)
- Try asking a question (5.7% increase)

Predicted number of likes between 209 and 1,259 centered around 734

This prediction may be inaccurate due to a wide range.

Figure 8.2.10 First iteration of prototype

3. When writing a question, users reported lower likes and not higher likes as suggested, when a question was written.

Actions taken to improve the interface for the next iteration.

1. The suggested number of words to write is only 10 words away from the number of words written.
2. The prediction is put into a box, with the prediction being written in bold text.
3. Questionmark suggestion was removed.

8.2.4.4 User test 2

With the improvements from user test 1 implemented and the site more refined, 6 users were tested. The users were again DTU students, though not the same from user test 1. Again the users were proficient with computer, the web and social media.

Insight found from the usability test:

1. The predicted number of likes were still not overt for most of the users.
2. Difficult to know how to select a date to publish the post at.
3. Some of the advice were contradictory. For instance when a specific message length was suggested, lets say 7 more words. The user would write 4 more words and expect the number of likes to increase.
4. Predictions being slow. When a user wrote some text, the prediction would take about 1 second.

8.2. Describing the website

Roskilde Festival

Select what you would like to predict

likes▼

Media type chosen: likes

Select what mediatype should be used

event▼

Media type chosen: event

Write the text for the Facebook post

Try including a hashtag (397.6% increase)


Try mentioning a page (50.2% increase)

Write 5 more words (14.8% increase)

Click on the graph to select a date to post

Zoom 1m 3m 6m YTD 1y All Jan 13, 2015 To Jan 16, 2015

Please select a date to publish the post



Predicted number of likes

Between **113** and **603** centered around **358**

This prediction is uncertain.

Figure 8.2.11 Second iteration of prototype

5. Unsure how to decrease the uncertainty of the prediction.

Actions taken to improve the website:

1. To ensure the number of likes were more overt, they were placed fixed in the bottom of the screen, taking up more space and being a more central part of the website.
2. The message length suggestion was removed.
3. Instead of providing a full weeks suggestions, predict 4 days into the future. This decreases the number of predictions that has to be calculated and thereby reduces response time as mentioned in section 8.2.2.
4. When writing that a prediction is uncertain, provide suggestions on how to reduce the uncertainty. This is done by writing "This prediction is uncertain. Try select another time or alter the text."

113

8.2.4.5 User test 3

The second test was done primarily with people from Falcon Social. The insight obtained from the second test were more functional problem than usability problems.

Roskilde Festival

Select what you would like to predict

likes▼

Media type chosen: likes

Select what mediatype should be used

link▼

Media type chosen: link

Write the text for the Facebook post

Check out this awesome new band we have booked! @arcticMonkeys

Everything is perfect, well done :)

Click on the graph to select a date to post

Zoom 1m 3m 6m YTD 1y All

1500
1000
500
0

12:00 16. Feb 12:00 17. Feb 12:00 18. Feb

16. Feb 17. Feb 18. Feb

Highcharts.com

Date chosen 15/02/2015 11:21:48

Predicted number of likes between **209** and **1,259** centered around **734**

This prediction is uncertain.
Try select another time or alter the text.

Figure 8.2.12 Third iteration of prototype

Insights found from the third user test:

1. Text spacing problem. Some people had problems when writing a word and then pressing space, this would cause different a prediction, even though no new words were written.
2. Only show suggestions that can improve the post. In user test two, suggestions that had been met was crossed out, they are now removed.
3. Some people asked if a post could be better from excluding a hashtag or mention in a post.

Actions taken to further improve the site:

1. The way message length was being calculated was changed.
2. Only suggestions that improve the post are shown in the suggestion list. If all suggestions has been met, a text saying that the post is perfect is being shown, as can be seen in 8.2.12.
3. Deleting hashtags or mentions in order to improve a post, was added as functionality to the site.

Chapter 9

Conclusion

The task of predicting how a Facebook post would perform even before it was posted is an interesting machine learning problem. Investigating and trying to predict something which might be somewhat impossible to achieve. This encouraged that a high level of domain knowledge of Facebook and social publishing was achieved before the analysis could actually begin.

This thesis outlined a detailed framework on how to make and apply a predictive analytics framework in the enterprise world, while also attempting to do so afterwards in collaboration with Falcon Social.

Some lessons learned:

- Ensemble models such as Random Forest are powerful on noisy data
- Feature engineering is important - and time consuming
- Topic modelling may not be very useful in social media
- Evaluating regression models is difficult, but necessary

The predictions were made both with regression and classification, and both instances were compared with dummy models to perform sanity checks on the predictions. Regression and classification both yielded that Random Forests performed better or equal than single trees or the K-nearest neighbour algorithm with a 99 % confidence interval. The predictions could sometimes be spot on, but also be completely wrong. The latter was especially true for outliers, which proved to be one of the great challenges for the predictions.

One could argue that there is some degree of randomness associated with the performance of a post, especially in terms of likes. Two very similar posts can perform very differently, while bad quality posts will consistently perform bad. When two posts are published, the initial likes on the post may influence how well the post will be exposed. For instance, if a user with a high number of friends likes the post, it may mean more than if a user with a few number of friends likes the post. Also, the Facebook news algorithm contains 100.000 different metrics that influences the exposure of a post for a single user. This is a highly complex system that can be difficult to capture in a model. It could be interesting to make a test, where

two very similar posts (in terms of features and exposed audience) are compared to each other, in order to investigate if they will perform equally.

A tool was developed for Falcon Social to test on, in order to get a feeling of how having predictive analytics in their existing platform might work. It was hosted on their servers in order for them to introduce it as proof of concept for their clients.

To summarise, predicting performance of Facebook posts is a challenging task with many surprises along the road. Especially when Facebook constantly makes changes to their news feed algorithm. The predictive analytics framework made was intended so this solution could be expanded for other social networks such as Twitter and Google+, but also for other predictive analytic problems.

9.1 Future work

There are quite a lot of different things that could be interesting to explore, to further improve this project.

9.1.1 Features, features, features

Exploring additional features would be interesting, in order to improve prediction. Some additional features are:

- Picture analysis features. These could be things like resolution, primary colour, colour intensity, and so on. When most of the posts in the data set are images, this could prove important regarding prediction.
- Trend mentions. Has a specific trend been mentioned in a post. This could be created using trending hashtags in Twitter.
- Importance of mention. Use Klout score to see how important a page is that has been mentioned.
- Audience. What kind of people have this post been targeted.
- Words. Use N-grams to include most used important words as features, using e.g. tf-idf.
- Local time. If a customer does not have a location, prompt the customer for a location.

9.1.2 Business verticals

This project focused quite a lot on the customer context. Another interesting context to add is the business verticals. A business vertical describes groups of customers. Examples are healthcare, media, banking, etc.

An example could be to simply have a boolean feature indicating the vertical. This might have influence on a posts successfulness.

9.1.3 Paid reach

Paid reach would provide customers with a very powerful tool, indicating how much reach/impressions/-consumptions would be achieved by paying for it. This task is quite difficult. When paying for reach, the group the post is targeted at, has a huge influence. When a post performs well it could be because the target group was correctly chosen, or because the post was a good/funny/interesting post.

9.1.4 More social networks

Including social networks such as Twitter, Instagram and Google Plus. Having multiple social media networks, means that Falcon Social can provide the same service no matter what social network their customer is using. Exploring what defines and influences the individual networks could also be quite exciting.

9.1.5 Improved interface

Further thinking about how the various feedback elements should be displayed to the user. Features like message length, sentiment and lexical diversity would be interesting to visualize for the user in a way that helps the user. Maybe visualizing the suggestions like time of day, makes sense.

9.1.6 Improved post weight

Improving the post weight, so that it takes time into account. Asking social media managers would be required in order to figure out the speed of the changes in Social media networks and see if the post weights could be fitted this speed.

Bibliography

Adobe Social Unveils Predictive Publishing for Facebook | *Business Wire* [2013].

URL: <http://www.businesswire.com/news/home/20130423006871/en/Adobe-Social-Unveils-Predictive-Publishing-Facebook>

Adobe Social Unveils Predictive Publishing for Facebook | *EON: Enhanced Online News* [2013].

URL: http://www.enhancedonlinenews.com/portal/site/eon/permalink/?ndmViewId=news_view&newsId=20130423006872&newsLang=en&permalinkExtra=

Anderson, E. [1936], 'The species problem in iris'.

Armstrong, J. S. and Collopy, F. [1992], 'Error measures for generalizing about forecasting methods: Empirical comparisons'.

Arun, R., Suresh, V., Madhavan, C. and Murty, N. M. [2010], 'On finding the natural number of topics with latent dirichlet allocation: Some observations'.

Athman Bouguettaya a, Q. Y., Liu, X., Zhou, X. and Song, A. [2014], 'Efficient agglomerative hierarchical clustering'.

Bergstra, J. and Bengio, Y. [2012], 'Random search for hyper-parameter optimization'.

Blei, D., Ng, A. and Jordan, M. [2003], 'Latent dirichlet allocation'.

Bleiman, L. [2001], 'Random forests'.

Brezillon, P. and Gonzales, A. J. [2014], 'Context in computing'.

Digital marketing | *Adobe Marketing Cloud* [2014].

URL: <http://www.adobe.com/solutions/digital-marketing.html>

Energy Applications | *Kaggle* [2014].

URL: <https://www.kaggle.com/solutions/energy>

Freund, Y. and Schapire, R. E. [1999], 'A short introduction to boosting'.

Guang-yong, H. [n.d.], 'Study and practice of import scrum agile software development'.

- Guyon, I. and Elisseeff, A. [2003], *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research.
- Guyon, I. and Steve Gunn, M. N. a. [2007], *Feature Extraction Foundations and Applications*, Pearson - Prentice Hall.
- Hastie, T., Tibshirani, R. and Friedman, J. [2008], 'The elements of statistical learning'.
- Hawkins, D. M. [2004], 'The problem of overfitting'.
- Helle Tyllesen [2014], 'Interview with helle tyllesen'.
URL: https://drive.google.com/file/d/oB_TZYZTk-VGabXpLaDctXoISU2M/view?usp=sharing
- Jason Brownlee [2014], 'Discover feature engineering, how to engineer features and how to get good at it - machine learning mastery'.
URL: <http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>
- Kullback, S. and Leibler, R. [1951], 'On information and sufficiency'.
- Loshin, D. [2013], *Business Intelligence*, Elsevier Inc.
- Marimont, R. B. and Shapiro, M. [1978], 'Nearest neighbour searches and the curse of dimensionality'.
- Melissa Leiter [2013], 'What makes a well crafted facebook post?'.
URL: www.socialmediatoday.com/content/what-makes-well-crafed-facebook-post
- Nielsen, J. [1993], 'Response times: The 3 important limits'.
URL: <http://www.nngroup.com/articles/response-times-3-important-limits/>
- Nielsen, J. [2012], 'Thinking aloud: The #1 usability tool'.
URL: <http://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/>
- Ripley, B. D. [2004], 'Robust statistics'.
- Ron Kohavi [1998], 'Glossary of terms journal of machine learning'.
URL: <http://robotics.stanford.edu/~ronnyk/glossary.html>
- Rosset, S., Perlich, C. and Zadrozny, B. [2006], 'Ranking-based evaluation of regression models'.
- Sara Piccola [2014], 'Breaking: Changes coming to facebook pages' news feeds'.
URL: <http://www.socialystacked.com/2014/11/breaking-changes-coming-facebook-pages-news-feeds/>
- Siegler, E. [2013], 'Predictive analytics'.
- Sklearn grid search* [2014].
URL: http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html

- Sra, S. and Dhillon, I. S. [2006], Generalized nonnegative matrix approximations with bregman divergences, in Y. Weiss, B. Schölkopf and J. Platt, eds, 'Advances in Neural Information Processing Systems 18', MIT Press, pp. 283–290.
URL: <http://papers.nips.cc/paper/2757-generalized-nonnegative-matrix-approximations-with-bregman-divergences.pdf>
- Tan, S., Cheng, X., Wang, Y. and Xu, H. [2009], 'Adapting naive bayes to domain adaptation for sentiment analysis'.
- Treder, M. [2013], 'The ultimate ux design of form validation'.
URL: <http://designmodo.com/ux-form-validation/>
- Turney, P. D. [n.d.], 'The management of context-sensitive features: A review of strategies'.
- Welch, B. L. [1947], 'The generalization of student's problem when several different population variances are involved'.
- Zhai, M.-Y., Rui-Hua Yu, S.-F. Z. and Zhai, J.-H. [2012], 'Feature selection based on extreme learning machine'.

Appendix A

Falcon's project proposal

Publishing

In the Falcon Social platform, our clients prepare and schedule their social media posts. We would like to utilize the vast amount of social media data we have, to give some intelligent indicators/advice to

The overall task is to predict the amount of engagement a post will get, once posted to a social media channel. We want to be able to give indicators/advice to our clients on what, where, when to post in social channel in order to gain as much attention as possible.

We are allowed to make a classifier based on business verticals ex. travel or media/entertainment which can be a value in the calculation as an industry metrics.

Example: When a client has entered a post into the Falcon Social platform and is about to publish it to a social channel, a classifier/analyser would suggest:

- "We can see you are writing about X. Consider posting this on a Friday as posts with this topic usually get 18% more attention on Fridays."
- "You might want to throw in an image. People love images and usually retweet image posts 25% more than non-image posts"
- "We noticed that your last 12 posts have been around roughly the same topics. Up for a change maybe? People usually engage 31% more with posts that are different from the usual content."

Datasources: News articles/blogs (80M per month) Tweets (3-5M per month) Facebook posts/comments (2-3M per month) Other social posts (0,5-1M per month) + historical data till 2006.

Appendix B

PCA

Component 1 feature weights
(avg_post_impressions, -0.58981361983434333)
(avg_post_consumptions, -0.56982563159355826)
(avg_likes, -0.55462287286736245)
(boolean_mention, -0.071525187583312957)
(boolean_hashtag, -0.068646771036515855)
(media_type=video, -0.063005027429678673)
(media_type=link, 0.059541591787725147)
(media_type=photo, -0.024970832359614798)
(boolean_questionmark, 0.021878017786778049)
(media_type=event, 0.016537195249430672)
(media_type=status, 0.016066828056365882)
(media_type=album, -0.013668145231146338)
(media_type=swf, -0.011138542353296079)
(day_of_week=Wednesday, 0.010680503592917306)
(day_of_week=Friday, -0.010430925236413601)
(time_of_day, -0.0093258056592834069)
(message_length, -0.0092909830814904368)
(sentiment, -0.0083901127649750745)
(day_of_week=Thursday, -0.004881557038303304)
(media_type=66, 0.0041612497683682914)
(day_of_week=Tuesday, 0.0028665619190802699)
(day_of_week=Saturday, 0.0020864144240287266)
(media_type=offer, 0.0018381434037654451)
(day_of_week=Sunday, -0.00067377976128259859)
(day_of_week=Monday, 0.00036937227355338912)
(lexical_diversity, 0.00031522319652960975)

Component 2 feature weights
(media_type=event, 0.51821623672005623)
(boolean_questionmark, -0.32669385505678855)
(message_length, -0.32564094282857797)
(boolean_hashtag, -0.32227958199877227)
(sentiment, -0.29413642812111979)
(media_type=video, -0.26155964067927623)
(boolean_mention, -0.23679034862169768)
(media_type=status, -0.23650642700150451)
(day_of_week=Monday, 0.2014491418344441)
(media_type=link, 0.15790793101976158)
(day_of_week=Saturday, -0.13108903955926698)
(day_of_week=Friday, -0.1105582760534079)
(media_type=photo, -0.096120454676292183)
(media_type=album, 0.090328086823724063)
(day_of_week=Sunday, -0.086684302640279515)
(media_type=66, 0.080595448488097396)
(day_of_week=Thursday, -0.072863999461449974)
(day_of_week=Wednesday, 0.070393530056161724)
(day_of_week=Tuesday, 0.069176708703294815)
(avg_post_consumptions, 0.052697184718133429)
(avg_likes, 0.038876135411755254)

```
(avg_post_impressions, 0.034726654398949341)
(lexical_diversity, 0.020706313969943702)
(media_type=offer, 0.018557467580395622)
(time_of_day, -0.015214072727711057)
(media_type=swf, -0.0011428647463235808)
```

Appendix C

Topic Models

Topic 1: 0.012*year + 0.012*idea + 0.011*celebrate + 0.011*drive + 0.010*photo + 0.010*help + 0.009*know + 0.009*test + 0.008*jaguar + 0.008*news

Topic 2: 0.028*day + 0.024*dont + 0.022*weekend + 0.020*get + 0.018*good + 0.013*come + 0.012*great + 0.012*make + 0.011*everyone + 0.011*see

Topic 3: 0.016*night + 0.010*danish + 0.010*youll + 0.008*customer + 0.008*award + 0.007*discount + 0.007*vogue + 0.007*need + 0.007*school + 0.007*room

Topic 4: 0.020*night + 0.018*free + 0.014*tonight + 0.013*park + 0.012*party + 0.012*join + 0.011*week + 0.011*get + 0.011*saturday + 0.010*book

Topic 5: 0.018*tip + 0.013*car + 0.013*think + 0.012*winter + 0.011*check + 0.011*ajax + 0.010*way + 0.010*make + 0.009*name + 0.008*blog

Topic 6: 0.038*click + 0.027*deal + 0.025*view + 0.024*product + 0.024*dress + 0.020*beautiful + 0.018*look + 0.017*get + 0.016*buy + 0.016*flash

Topic 7: 0.024*watch + 0.020*video + 0.015*woman + 0.012*sign + 0.011*classic + 0.010*chicken + 0.010*blue + 0.008*cover + 0.008*lovely + 0.008*new

Topic 8: 0.044*win + 0.023*enter + 0.021*share + 0.020*chance + 0.017*like + 0.013*facebook + 0.013*winner + 0.012*christmas + 0.012*give + 0.011*post

Topic 9: 0.031*diamond + 0.028*end + 0.027*offer + 0.023*new + 0.018*sunday + 0.014*store + 0.014*ring + 0.013*shop + 0.012*gold + 0.011*great

Topic 10: 0.016*new + 0.014*water + 0.012*take + 0.011*excite + 0.011*week + 0.009*never + 0.009*fashion + 0.009*get + 0.008*run + 0.008*garden

Topic 11: 0.014*one + 0.011*colour + 0.010*like + 0.010*favorite + 0.009*make + 0.008*many + 0.008*wait + 0.007*look + 0.007*love + 0.007*year

Topic 12: 0.032*new + 0.020*home + 0.017*festival + 0.015*look + 0.010*see + 0.010*first + 0.009*open + 0.009>window + 0.009*time + 0.009*early

Topic 13: 0.020*live + 0.016*music + 0.012*band + 0.012*show + 0.010*info + 0.010*september + 0.008*festival + 0.008*set + 0.008*stage + 0.007*get

Topic 14: 0.012*travel + 0.011*member + 0.010*holiday + 0.009*could + 0.009*learn + 0.009*save +

0.008*home + 0.008*spring + 0.008*always + 0.007*club

Topic 15: 0.013*get + 0.013*one + 0.013*love + 0.010*set + 0.010*perfect + 0.009*summer + 0.009*three + 0.009*new + 0.009*day + 0.009*black

Topic 16: 0.023*new + 0.020*check + 0.016*make + 0.014*photo + 0.013*track + 0.013*release + 0.012*claim + 0.010*state + 0.009*feature + 0.009*light

Topic 17: 0.019*follow + 0.015*world + 0.012*social + 0.012*game + 0.012*football + 0.011*today + 0.011*twitter + 0.011*team + 0.011*goal + 0.011*match

Topic 18: 0.017*style + 0.012*collection + 0.011*business + 0.010*sale + 0.009*shop + 0.009*march + 0.009*lady + 0.009*new + 0.008*watch + 0.008*find

Appendix D

Clustering

D.1 Agglomerative Clustering

```
Number of customers: 22
[ 267. 14446. 152641.]
[ 239. 5817. 183500.]
[ 2876. 2898. 150824.]
Number of customers: 1539
[ 10. 437. 3672.]
[ 89. 1317. 4444.]
[ 1. 19. 470.]
Number of customers: 2
[ 48869. 64579. 1133706.]
[ 10225. 36229. 1218980.]
Number of customers: 100
[ 129. 708. 20335.]
[ 38. 2538. 26520.]
[ 111. 939. 22536.]
Number of customers: 5
[ 1371. 18351. 404396.]
[ 10338. 30338. 366520.]
[ 2018. 21915. 326189.]
Number of customers: 2
[ 11088. 86239. 1571344.]
[ 27679. 39612. 1594789.]
Number of customers: 3
[ 4246. 9656. 671985.]
[ 14328. 27450. 620396.]
[ 12330. 49265. 682422.]
Number of customers: 23
[ 661. 15324. 101271.]
[ 7.80000000e+01 1.79100000e+03 9.52790000e+04]
[ 1447. 6762. 68458.]
Number of customers: 2
[ 2264. 10841. 540307.]
[ 729. 3491. 474153.]
Number of customers: 1
[ 174. 158023. 113721.]
```

D.2 K-means Clustering

```
Number of customers: 100
[ 129. 708. 20335.]
[ 38. 2538. 26520.]
[ 111. 939. 22536.]
```

```
Number of customers: 2
[ 11088. 86239. 1571344.]
[ 27679. 39612. 1594789.]
Number of customers: 5
[ 1371. 18351. 404396.]
[ 10338. 30338. 366520.]
[ 2018. 21915. 326189.]
Number of customers: 18
[ 267. 14446. 152641.]
[ 239. 5817. 183500.]
[ 2876. 2898. 150824.]
Number of customers: 1537
[ 10. 437. 3672.]
[ 89. 1317. 4444.]
[ 1. 19. 470.]
Number of customers: 3
[ 4246. 9656. 671985.]
[ 14328. 27450. 620396.]
[ 12330. 49265. 682422.]
Number of customers: 2
[ 48869. 64579. 1133706.]
[ 10225. 36229. 1218980.]
Number of customers: 29
[ 661. 15324. 101271.]
[ 7.80000000e+01 1.79100000e+03 9.52790000e+04]
[ 316. 1302. 58806.]
Number of customers: 2
[ 2264. 10841. 540307.]
[ 729. 3491. 474153.]
Number of customers: 1
[ 174. 158023. 113721.]
```

Appendix E

Extended Results

Prediction	MSE	R ²	MdAPE
Insight.post_consumptions			
DecisionTreeRegressor	37758087	0.67	67.58
DummyRegressor	114897789	0.00	93.41
KNeighborsRegressor	65373872	0.43	73.20
RandomForestRegressor	35266584	0.69	67.57
Insight.post_impressions			
DecisionTreeRegressor	14432146087	0.81	51.45
DummyRegressor	76613245871	0.00	92.79
KNeighborsRegressor	25646800292	0.67	54.41
RandomForestRegressor	14103250864	0.82	50.31
Post.likes			
DecisionTreeRegressor	2770125	0.63	50.90
DummyRegressor	7397318	0.00	87.18
KNeighborsRegressor	3034543	0.59	60.00
RandomForestRegressor	1948348	0.74	50.18
Post.relative_likes			
DecisionTreeRegressor	9	0.48	65.69
DummyRegressor	17	0.00	84.70
KNeighborsRegressor	7	0.62	62.50
RandomForestRegressor	17	0.04	67.36

Table E.0.1 Results for the regression models, where the models are compared against each other with Mean Squared Error MSE, the Coefficient of Determination R² and Median Absolute Percentage Error MdAPE