CHAPTER 1

INTRODUCTION

1.1 Introduction to the project background

1.1.1 Danish craft breweries

This section will give a brief introduction to the background of Danish craft beer brewing, which will support the following section Target scenario for a better understanding Historically in Denmark, there were ups and downs in the brewing industry. During the 19th century, there were almost one thousand breweries, then the number reduced through closures, mergers and acquisitions, and at the dawn of the new millennium, there were only 18 breweries left in Denmark. However, from 2005 there was an explosion of new breweries opening throughout the country (see figure 1.1), soon there was hardly a spot on the map of Denmark without a local brewery. Despite from brewery that had mass production on standard taste beers, craft beers are mostly considered to have more taste and better quality The definition for craft brewery is very blurry, and the concept of a micro brewery was once virtually unknown in Denmark. In America, an American craft brewery is Small. Annual production of 6 million barrels of beer or less.

Independent. Less than 25 percent of the craft brewery is owned or controlled (or equivalent economic interest) by a beverage alcohol industry member that is not a craft brewer itself.

Traditional. A brewer that has a majority of its total beverage volume in beers, whose flavour derives from conventional or innovative brewing ingredients and their fermenta-tion [8].

Though data shows that the beer consumption had decreased from 5.9 in 1980 to 4.0 in 2006, the business for Danish craft breweries has had a trend of growth the recent years. Figure 1.2 shows from 2003 until 2015, the annual Danish beer sales remain-

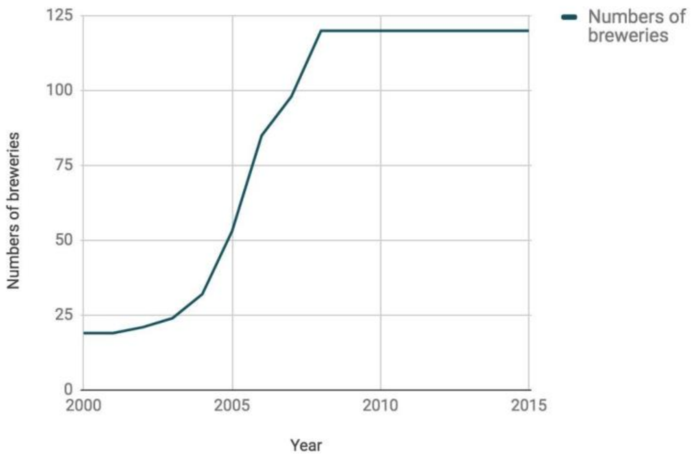


Figure 1.1: Number of Danish breweries from 2000 to 2015, sourced from Danske Ølentusiaster and Bryggeriforeningen (Bryggeriforeningen, known as the Danish Brew-ers Association, is the trade organisation and common voice for breweries and producers of soft drink in Denmark. It represents the beer and soft drink industries towards the parliament, government, media and the public regarding political issues like taxation, environment, food and health)

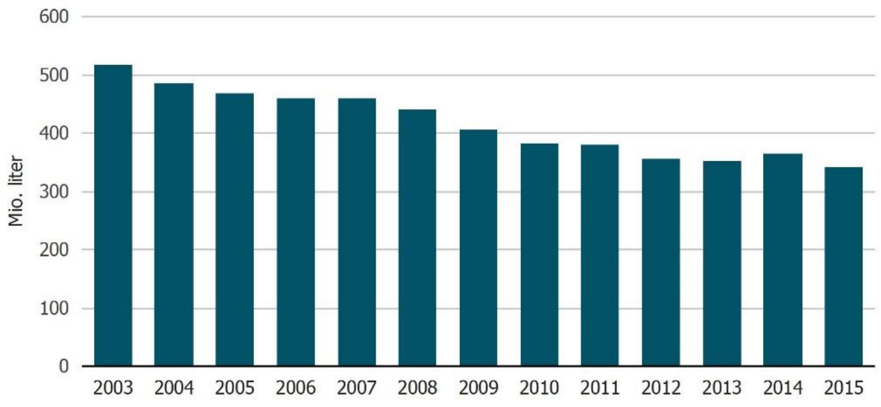


Figure 1.2: Danish beer market sales, from 2003 until 2015, the annual Danish beer sales remaining decreasing (chart adapted form [1])ing decreasing. However, regardless of the decreasing sale and consumption of Danish beers, a growing interest in beer has been noticed in Denmark late years. Result from increasing demand; more craft beers appear on the market. Figure 1.3 shows some new beers appear in the Danish market from 2010-2015, by now, the number is still growing

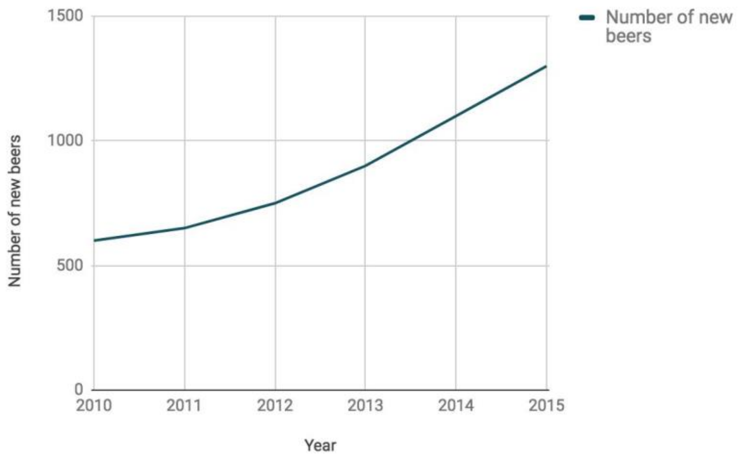


Figure 1.3: The growth on new beers in Danish market from 2010-2015

although the leading roles of the beer market are still big brewing companies like Carls-berg, small breweries' got a positive potential.

The Danish craft beer breweries are small scaled, but their consumers are not only in Denmark. Due to market research from Markerline.com (Figure 1.4), 64.4% of the cus-tomers locate in the rest of Europe(2013), which means many breweries need to handle business and supply chain issues cross the border.

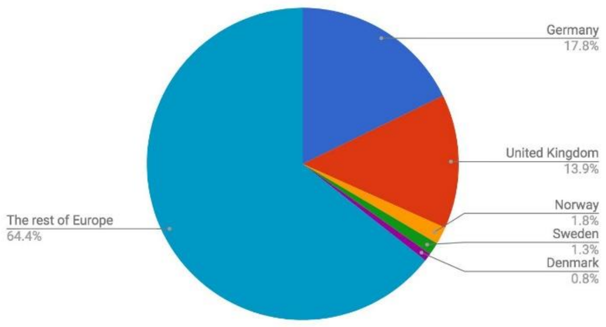


Figure 1.4: Danish beer market geography segmentation(Adapted from market-line.com)

1.1.2 Backorder and Bullwhip effect

The bullwhip effect causes members of the supply chain to overreact to changes in de-mand at the retail level. Minor demand changes at the consumer level may result in large ones at the supplier level. Bullwhip fluctuations create unstable production schedules, resulting in an expensive capacity change adjustments such as overtime, subcontract-ing, extra inventory, backorders, hiring and laying off of workers, equipment additions, under-utilization, longer lead times, or obsolescence of over-produced items [9].

Figure 1.5 shows an example of the interaction of stakeholders in the supply chain. An

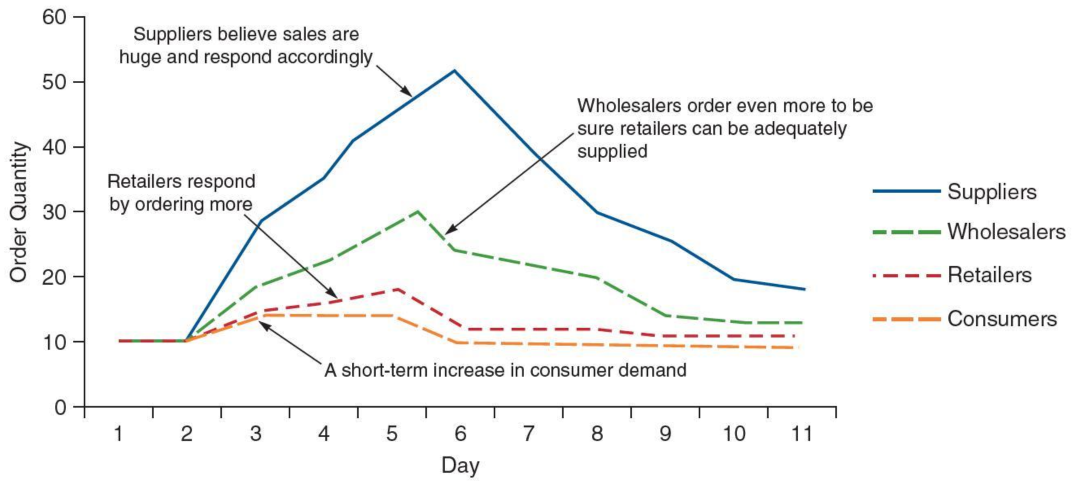


Figure 1.5: An example of bullwhip effect

organisation can shift demand fulfilment to other periods by allowing back orders. That is, orders are taken in one period, and deliveries promised for a later time. The success of this approach depends on how willing customers are to wait for delivery. Moreover, the costs associated with backorders can be difficult to pin down since they would include lost sales, annoyed or disappointed customers, and perhaps additional paperwork.

So solving backorder problems and giving more satisfaction to customers becomes critical in the supply chain and the solution to the problem could be more precise forecasting. When a business can foresee the incoming orders, have more time to purchase and produce, the backorder will no longer be a very thorny issue.

1.1.3 Target scenario

Nowadays, craft beers produced by microbreweries have contributed more in Danish beers global turnover, which brings up the question: For small scaled breweries whose international market scale is larger than the local market, whether can they react fast enough to the market, to ensure a robust supply chain when their demand rises.

As mentioned from the Danish beer brewing industry background, the demand for craft beers is rising by years. For producers, one of the primary purposes of supply chain collaboration is to improve the accuracy of forecasts,especially international busi-ness. In this case, when the brewery needs to collaborate with foreign distributors and retailers, many issues might appear due to the lack of accurate predictions.

In this case, considering customer satisfaction and demand, the producers need to be well prepared to ensure a robust and fast supply chain. With the help of advanced predictions, the business will have time to react before any unexpected event occurs and therefore help the producers be more intelligent on stocking and become more proactive.

CHAPTER 2

OBJECTIVE

By realising the potential issue, this project is tempting to find out the best classifier for backorder predictions. Results will be compared with other classifiers in the evaluation. The research question is:

1. How can Danish brewers use machine learning techniques, train on historical data, predict backorders in their early stage of the supply chain?
2. Are machine learning predictions more efficient than the traditional forecasting?

Details of data requirements and machine learning methods will be explained, and basic terminology will be provided as well.

The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

70% collected data- Training data

30% collected data- Test data

The models are made and for each model accuracy has to be checked. Model with highest accuracy will be the final result.

CHAPTER 3

THEORITICAL BACKGROUND

This chapter will briefly introduce the tools and terminologies that were used or men-tioned in the project.

2.1 Data issues

While converting a dataset to a certain format, there are some practical points to be aware. Real data are often low in quality, and careful checking a process that has become known as data cleaning, which pays off many times over.

2.1.1 Inaccurate values

Typographic errors in a dataset will obviously lead to incorrect values. Typographical or measurement errors in numeric values cause outliers that can be detected by graphing one variable at a time. Erroneous values often deviate significantly from the pattern that is apparent in the remaining values. Sometimes, however, inaccurate values are hard to find, particularly without specialist domain knowledge.

2.1.2 Missing value

Missing values are frequently indicated by out-of-range entries; perhaps a negative num-ber (e.g., -1) in a numeric field that is normally only positive, or a 0 in a numeric field that can never normally be 0. For nominal attributes, missing values may be indi-cated by blanks or dashes. Sometimes different kinds of missing values are distinguished (e.g., unknown versus unrecorded versus irrelevant values) and perhaps represented by different negative integers.

2.1.3 Outlier

An outlier is defined as a dissimilar data point that is far from the rest of the data. It normally explains the abnormal behaviour of a feature. Outliers, being the most extreme observations, may include the sample maximum or sample minimum, or both, depending on whether they are extremely high or low. However, the sample maximum and minimum are not always outliers because they may not be unusually far from other observations/

2.2 Imbalanced dataset

The class imbalance problem is one of the (relatively) new problems that emerged when machine learning matured from an embryonic science to an applied technology, amply used in the worlds of business, industry and scientific research. The class imbalance problem typically occurs when, in a classification problem, there are many more instances of some classes than others. In such cases, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones. In practical applications, the ratio of the small to the large classes can be drastic such as 1 to 100, 1 to 1,000, or 1 to 10,000 (and sometimes even more).

2.2.1 Data sampling

Sampling methods seem to be the dominant type of approach in the community as they straightforwardly tackle imbalanced learning. In general, the use of sampling methods in imbalanced learning consists of the modification of an imbalanced dataset by some mechanism to provide a balanced distribution. The key aspect of sampling methods is the mechanism used to sample the original dataset. Under different assumptions and with various objective considerations, various approaches have been proposed.

2.4 Machine learning

How to build an intelligent machine?

The science of learning plays a key role in the fields of statistics, data mining and artificial intelligence, intersecting with areas of engineering and other disciplines. Generally, there are two types of machine learning, supervised machine learning and unsupervised machine learning.

2.4.1 Supervised machine learning

In supervised machine learning, the task is to predict a quantity based on other quantities. It is useful to distinguish between classification and regression that fall into the category of supervised machine learning: In classification, with given observed values x and have to predict a discrete response y.

Figure 2.1 shows an expample of classification.

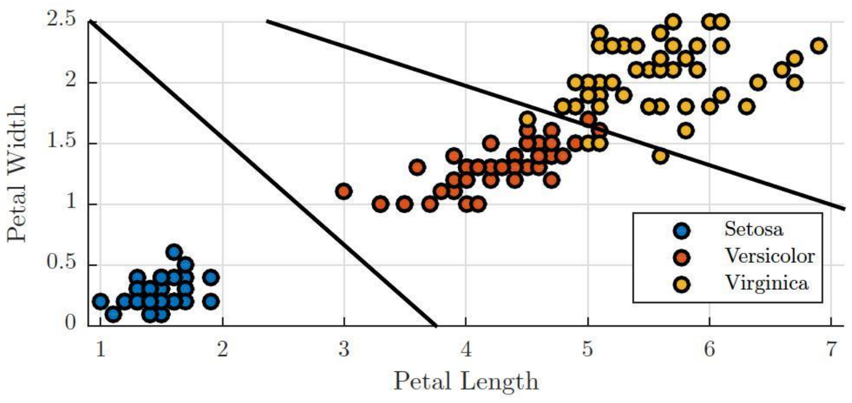


Figure 2.1: A classification problem where observations (the points) and class labels (the colours) are given and the goal is to come up with a rule for determining which class a point belongs to (one such rule is indicated by the lines). The rule can then be applied to new points

In regression problems, we are given observed values x and have to predict a continuous response y. Figure 2.2 shows an example of regression. The distinction in output type has

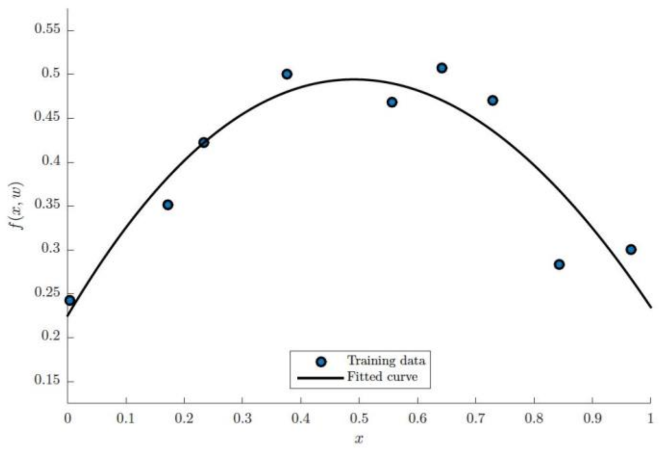


Figure 2.2: A one-dimensional regression problem which predicts the y-values based on the x-values. The titted regression model is indicated by the black line [2].

led to a naming convention for the prediction tasks: regression for predicting quantitative outputs, and classification for predicting qualitative outputs.

2.4.2 Unsupervised machine learning

Apart from supervised learning is unsupervised learning. Unsupervised learning tries to solve this and similar problems where we do not have access to any ”ground-truth” label information (such as the identity of the animal in the image) but we try to discover this labelling from the data alone.

Figure 2.3 shows an example of clustered data points.

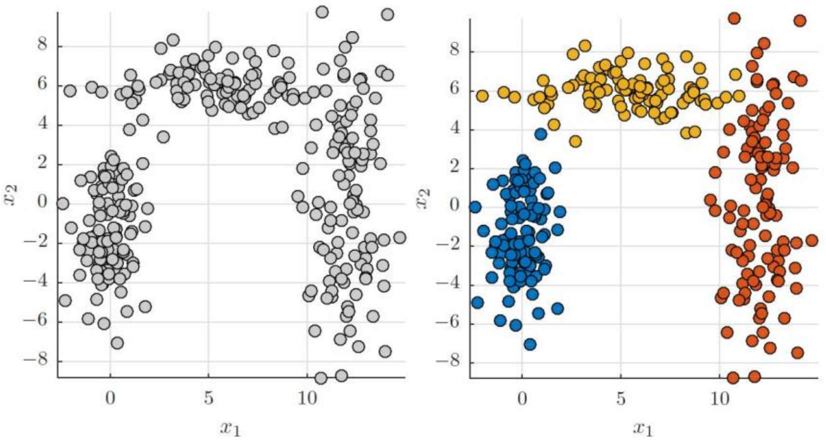


Figure 2.3: A 2D clustering example. A clustering is given a dataset (here the 2D dataset shown in the left-hand pane) and has to estimate plausible divisions of the observations into clusters as indicated in the right-hand pane [2]

In this project, classification method would be tries out to predict a binary target value, the backorders. Common method to deal with classification problems are: Tree based methods, nearest neighbour methods, Bayesian method and neural network.

CHAPTER 4

Techniques Involved

4.1 End to End Pipeline Process

Generically speaking a pipeline has inputs go through a number of processing steps chained together in some way to produce some sort of output.A data analysis pipeline is a pipeline for data analysis.

Usually they're done in some graphical environment such as Alteryx or KNIME (with scripting steps in R or Python, say), each step logically following each step. There is often preprocessing, data checking, analysis, analysis checks, visualization checks, etc., etc., before the final result, which is usually either a data product or set of decisions and their supports.

They're done to make data analysis easier.

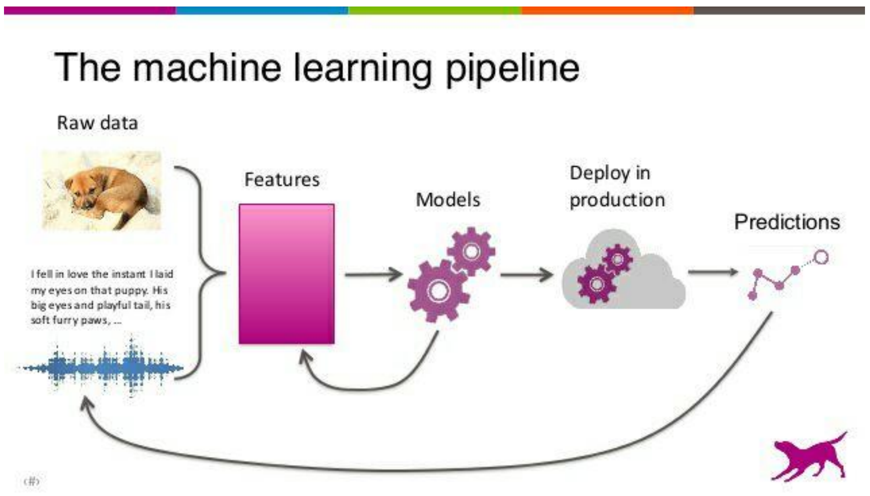

The machine learning pipeline
Raw data
I fell in love the instant I laid my eyes on that puppy. His big eyes and playful tail, his soft furry paws, ...
Features
Models
Deploy in production
Predictions

Figure No. 4.1 Pipelining in machine learning

4.2 Data Normalization

Database normalization, or simply normalization, is the process of organizing the columns (attributes) and tables (relations) of a relational database to reduce data redundancy and improve data integrity. Normalization is also the process of simplifying the design of a database so that it achieves the optimal structure composed of atomic elements.

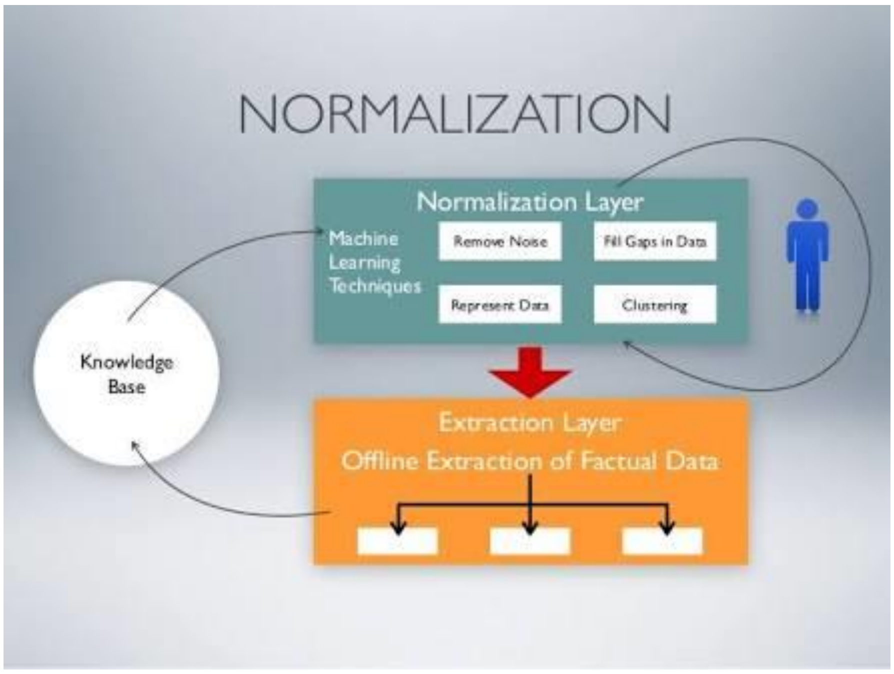


Figure 4.2 Normalization

4.3 Model Evaluation

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over-fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.

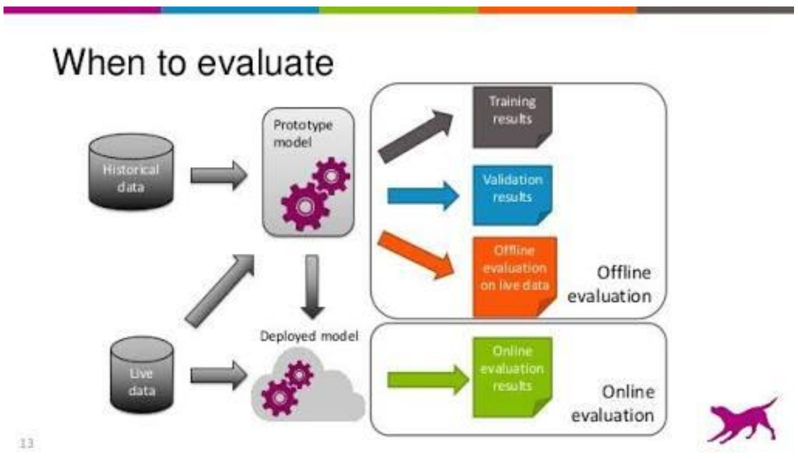


Figure 4.3 Model Evaluation

4.4 Improving Model Accuracy

Improving Model Accuracy-Obtaining a ML model that matches your needs usually involves iterating through this ML process and trying out a few variations. You might not get a very predictive model in the first iteration, or you might want to improve your model to get even better predictions. To improve performance, you could iterate through these steps:

Collect data: Increase the number of training examples

Feature processing: Add more variables and better feature processing

Model parameter tuning: Consider alternate values for the training parameters used by your learning algorithm

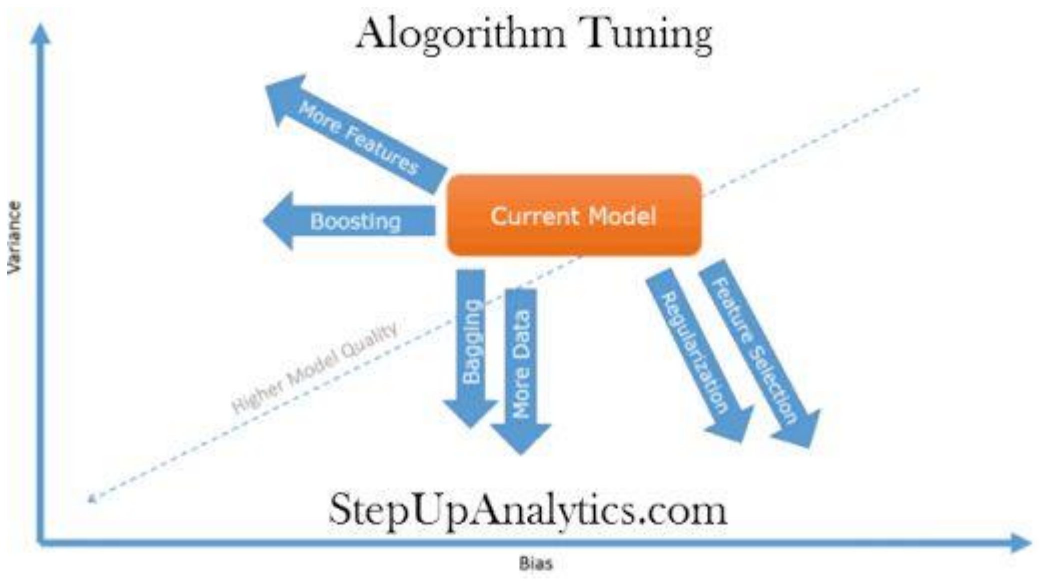


Figure 4.4 Improving Model Accuracy

CHAPTER 5

Software Used

Anaconda is open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system *conda*.

Anaconda Cloud is where data scientists share their work. You can search and download popular Python and R packages and notebooks to jumpstart your data science work.You can also store your packages, notebooks and environments in Anaconda Cloud and share them with your team.If you're looking to share a limited amount of your work publicly then our free service is perfect for you. Sign up today!

If you're looking to share larger collections and collaborate privately then our subscription plans are your answer.

Anaconda Cloud is brought to you by Anaconda Inc.

Language Used:
PYTHON

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementationof Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

CHAPTER 6

Proposed solution

4.1 Introduction to the solution

As discussed in the target scenario, to avoid bullwhip effect, small brewers need to react fast and meantime avoid overreacting when the market changes. By having a more advanced forecasting too, business can identify the market more precisely where the predictions providing more evidence about what's going to happen. A relevant dataset was found to support the prediction process, which contains features from the early stage of the supply chain and has a binary target value.

Many machine learning tools and methods can perform predictions with good perfor-mance, in this project, many classification methods will be tried and evaluated, in order to get the best performance classifier and predictions. Figure 4.1 shows a general ma-chine learning process, and it is also the work flow that this project is implicated. By understanding, discovering data issues and as well visualizing historical data, the scale and definition of the training set will become clearer.

A few standard classifiers were chosen to train the prediction models, and they were tested and evaluated. When the results are under-promising, the training set may be redesigned, some analysis may be conducting in another way, until the best performance shows up. Each classifier method was tested in many ways, and the final evaluation shows a comparisons of performance indicators, which will show the overview of each classifiers and as well the best one that can be trained under certain circumstances.

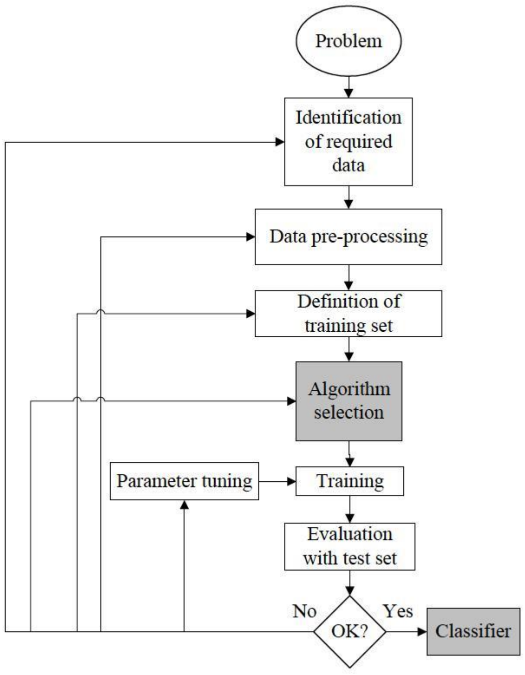


Figure 4.1: Machine learning work flow [5]

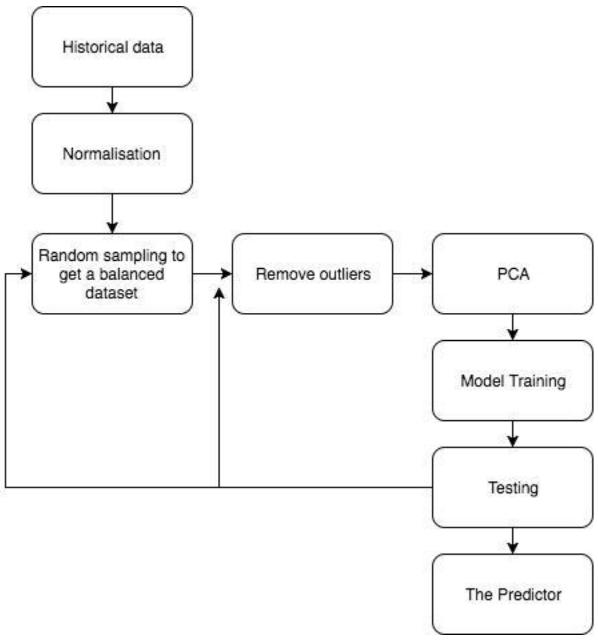


Figure 4.2: Project work flow

4.2 Data pre-processing

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. The goal of this stage is to transform raw data into an under-standable format. In this project, it is required that all features in the training set are numeric. So this section will introduce steps for pre-processing the dataset and also give a understanding about the data contend and scale.

4.2.1 Introduction to the dataset

The dataset used in this project comes from website Kaggle, provided by an anonymous company. Details of the data can be found in appendix A1. The dataset is historical data, which contains information of their products for the 8 weeks prior to the week that is going to be predicted. The data was taken as weekly snapshots at the start of each week. In total, it consists 23 attributes with 1,915,211 entries and, 6 of which are binary, 1 notation and the other 16 are numeric. Table 4.1 shows brief explanation and data types of the 23 attributes: To note, SKU represents random products. The number of actual products is far smaller than millions, but the actual unique SKU for each product is not included as they can change. So a sequential identifier was created where each record identifies a unique product/week combination.

Among the attributes, some represents the status of transiting, like: lead time of the product (lead-time, as weeks) and amount of product in transit from source (in transit qty, units).

Some represents sales and prediction: attribute 6-10 some represents part risk and source

For source performance, it represents product supplier's historical performance. Some suppliers have not been scored and have a dummy value of -99 loaded.

The part risk flag means the shortage of a part: in this project, it could be the shortage of ingredients needed for the production. And others represent inventory levels and stocking: attribute 20-23. Generally, part risk and source issue will create overdue of products, it might lead to a stock shortage. And a stock shortage may lead to the product backorder. This dataset was created on the understanding of the business and also the data availability. All the binary attributes except the target value were created as an risk indicator, based on other calculations that was not stated in this introduction. Reasons for product went on backorders could be multiple, especially when the target value is among a high dimensions space, that is where machine learning can bring values in.

	Attribute	Description	Data type
1	sku	Random ID for the product	Norminal/Discrete
2	lead time	Transit time for product (if available)	Ratio/Continuous
3	in transit qty	Amount of product in transit from source	Ratio/ Discrete
4	forecast 3 month	Forecast sales for the next 3 months	Ratio/ Discrete
5	forecast 6 month	Forecast sales for the next 6 months	Ratio/ Discrete
6	forecast 9 month	Forecast sales for the next 9 months	Ratio/ Discrete
7	sales 1 month	Sales quantity for the prior 1 month	Ratio/ Discrete
8	sales 3 month	Sales quantity for the prior 3 month	Ratio/ Discrete
9	sales 6 month	Sales quantity for the prior 6 month	Ratio/ Discrete
10	sales 9 month	Sales quantity for the prior 9 month	Ratio/ Discrete
11	perf 6 month avg	Source performance for prior 6 month	Ratio/ Discrete
12	perf 12 month avg	Source performance for prior 12 month	Ratio/ Discrete
13	pieces past due	Parts overdue from source	Ratio/ Discrete
14	deck risk	Part risk flag	Binary
15	oe constraint	Part risk flag	Binary

16	ppap risk	Part risk flag	Binary
17	stop auto buy	Part risk flag	Binary
18	rev stop	Part risk flag	Binary
19	potential issue	Source issue for part identified	Binary
20	national inv	Current inventory level for the part	Ratio/ Discrete
21	min bank	Minimum recommend amount to stock	Ratio/ Discrete
22	local bo qty	Amount of stock orders overdue	Ratio/Discrete
23	went on backorder	Product actually went on backorder	Binary

Table 4.1: Attributes from the historical data, a short explanation to them and their data types

4.2.2 Binarization

In statistics, the response to a yes-no question (”yes” or ”no”) is considered to be binary data. But in order to make the whole dataset processable in Python, it is necessary to have all data as numeric data type, so that all features can be involved in one matrix, which is the foundation of the data analyzing. In the dataset, there are binary attributes containing yes/no value, in order to analyse them in a matrix, it is necessary to transfer those values into numeric values. Codes are provided in appendix A2.

4.4 Data issues

In this section all related data processing processes are addressed. For details of the coding please go to appendix A2

4.4.1 Data imbalance

Knowing from the data, 0.7% of the products actually went on backorder, which creates a big imbalance on the classes. It is a common problem for a real dataset. Figure 4.6 shows the proportion of the two predicting classes.

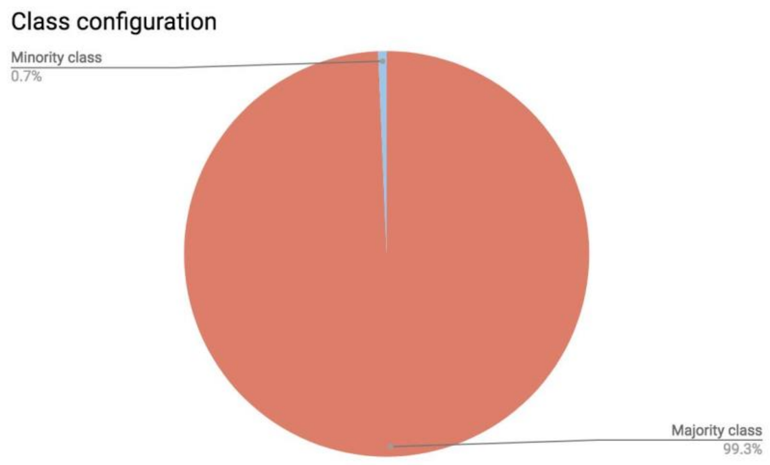


Figure 4.6: A pie chart of proportion of the two predicting classes

Preferably, it is required that a classifier provides a balanced degree of predictive ac-curacy for both the minority and majority classes on the dataset. However, in many standard learning algorithms, it is found that classifiers tend to provide a severely im-balanced degree of accuracy, with the majority class having close to 100% accuracy and the minority class having accuracy of 0 - 10%;

This low accuracy on minority class often suggests that the conventional evaluation practice of using singular assessment criteria, such as the overall accuracy or error rate, does not provide adequate information in the case of imbalanced learning.

4.5 Data Modeling

4.5.1 Design of experiments

When data pre-processing and data cleaning give a better explanation of how the data looks like, the issues from the data for machine learning are also revealed. Outliers detection in high dimensional dataset and dataset extremely imbalanced are the core issues in this project, both of which can confuse classifiers during data training and give inaccurate results.

The variability of data issues were highly considered, so experiments were designed ought to reduce most of the negative influence that these issues can give on the classifier accuracy, so that the best classifier for backorder prediction can be found out.

For the issue of data imbalance, if use full historical data to train classifier, it may give a confusing result where the accuracy is very high but the true class precision is low. A sampled dataset with balanced class values may give a more promising result. For the issue of outliers: Since outliers can be a fraction to the accuracy of the classifiers, for comparison, the data can be trained in two manners: with outliers and without outliers. The result can show if removing outliers can lead to better accuracy.

4.5.2 Definition of training set

A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value" (i.e. the class labels) and several attributes" (i.e. the features or observed variables). A training set is a set of data used to discover potentially predictive relationships. A test set is a set of data used to assess the strength and utility of a predictive relationship.

To reduces data redundancy and data dimensionality, enable machine learning algo -rithms to operate faster and more effectively, a data normalization and Principle com-ponent analysis were conducted. These methods can remain the data quality and main features while reducing data size. After the analysis, the newly generated features may lead to the creation of more concise and accurate classifiers. In addition, the discovery of meaningful features contributes to better comprehensibility of the produced classifier, and a better understanding of the learned concept.

Due to the imbalanced issue, use the full historical data as a training set may not give the best classification results. However, there are many tools and methods to deal with data imbalance, some of which are simple and some others are more intelligent. The sampling methods described in the Theoretical Background are designed to reduce between-class imbalance. Although research indicates that reducing between-class imbalance will also tend to reduce within-class imbalances, it is worth considering whether sampling methods can be used in a more direct manner to reduce within-class imbalances and whether this is beneficial. Based on this, by extracting all data entries with minority class and random sampling data entries from majority class, a balanced dataset is created to reduce that problems that might occur during training on imbalanced data. For comparison to see if the sampling method is beneficial, two training sets were created: Training set one: a training set which includes all entries from cleaned historical data

Training set two: a training set which includes all minority class and same amount of majority class

Algorithms Used

The choice of which specific learning algorithm should be used is a critical step. Once preliminary testing is judged to be satisfactory, the classifier (mapping from unlabeled instances to classes) is available for routine use. As mentioned in literature review, SVM and Neural networks are the best performers on predictions, but in this project it is restrict to try out different classifiers to test for the best model. Neural networks would not be considered for training. Based on the balanced training set, multiple classifiers were trained on cross validation with five folds.

After the algorithm selection, the selected classifiers were trained on different kernel scales, number of neighbours and number of learners to get the best performance for that specific method, so in the chapter of results all the forecasting results are from the best performance that one model can get.

6.1 Logistic Regression

In statistics, **logistic regression**, or **logit regression**, or **logit model** is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression.[2] In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

Logistic regression was developed by statistician David Cox in 1958. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor. The model is a direct probability model and not a classifier.

Logistic regression can be binomial, ordinal or multinomial. Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types, "0" and "1" (which may represent, for example, "dead" vs. "alive" or "win" vs. "loss"). Multinomial logistic regression deals with situations where the outcome can have three or more possible types (e.g., "disease A" vs. "disease B" vs. "disease C") that are not ordered. Ordinal logistic regression deals with dependent variables that are ordered.

In binary logistic regression, the outcome is usually coded as "0" or "1", as this leads to the most straightforward interpretation. If a particular observed outcome for the dependent variable is the noteworthy possible outcome (referred to as a "success" or a "case") it is usually coded as "1" and the contrary outcome (referred to as a "failure" or a "noncase") as "0". Binary logistic regression is used to predict the odds of being a case based on the values of the independent variables (predictors). The odds are defined as the probability that a particular outcome is a case divided by the probability that it is a noncase.

6.2 K- Nearest Neighbour

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

Let us take a few examples to place KNN in the scale :

KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

How does the KNN algorithm work?

Let’s take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS) :

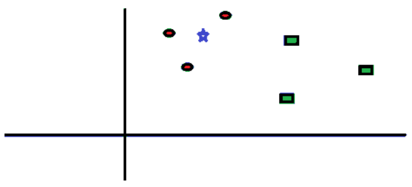


Figure No. 6.10 Red Circles and green squares

You intend to find out the class of the blue star (BS) . BS can either be RC or GS and nothing else. The "K" is KNN algorithm is the nearest neighbors we wish to take vote from. Let's say K = 3. Hence, we will now make a circle with BS as center just as big as to enclose only three datapoints on the plane. Refer to following diagram for more details:

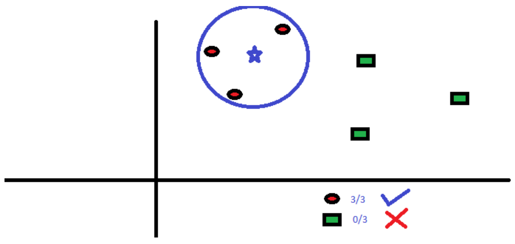


Figure No. 6.11 Finding nearest neighbour

The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm. Next we will understand what are the factors to be considered to conclude the best K.

How do we choose the factor K?

First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. The same way, let's try to see the effect of value "K" on the class boundaries. Following are the different boundaries separating the two classes with different values of K.

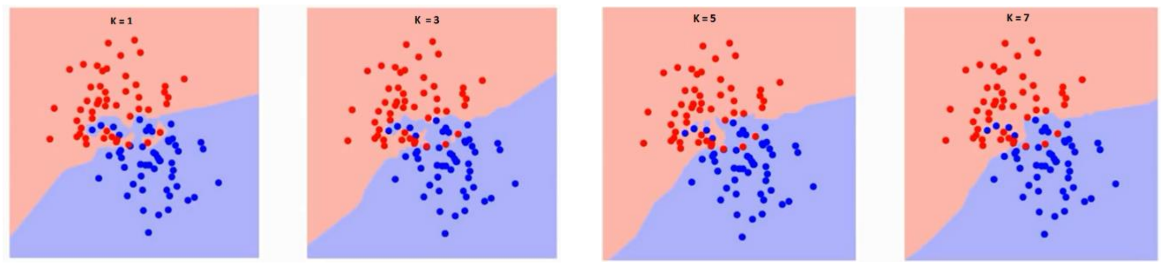


Figure 6.12 Boundaries separating two classes for different values of K

If you watch carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access on different K-value. Following is the curve for the training error rate with varying value of K :

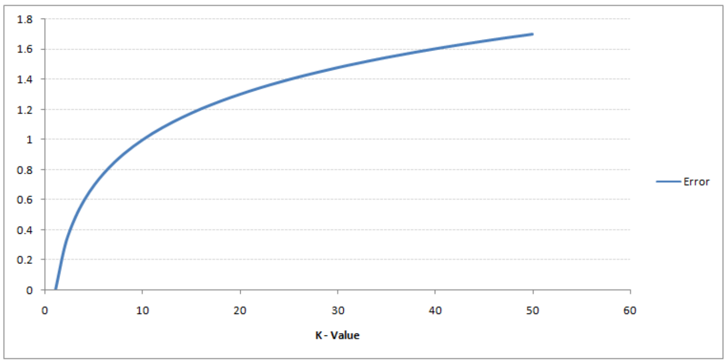


Figure No.6.13 Curve for the training error rate with varying value of K

As you can see, the error rate at K=1 is always zero for the training sample. This is because the closest point to any training data point is itself.Hence the prediction is always accurate with K=1. If validation error curve would have been similar, our choice of K would have been 1. Following is the validation error curve with varying value of K:

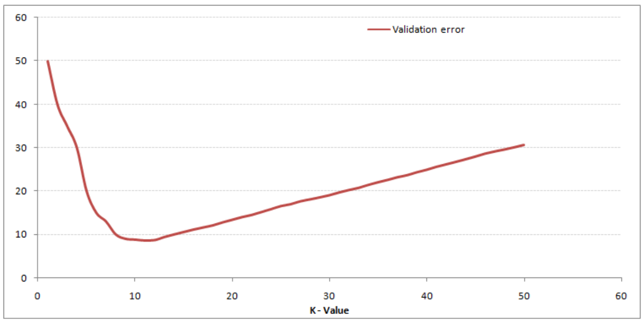


Figure No. 6.14 Validation error curve with varying value of K

This makes the story more clear. At K=1, we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K. To get the optimal value of K, you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K. This value of K should be used for all predictions.

6.3 Ensemble Random Forest

Random Forest is a versatile machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and does a fairly good job. It is a type of ensemble learning method, where a group of weak models combine to form a powerful model.

It works in the following manner. Each tree is planted & grown as follows:

1. Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<M is specified such that at each node, m variables are selected at random out of the M. The best split on these m is used to split the node. The value of m is held constant while we grow the forest.
3. Each tree is grown to the largest extent possible and there is no pruning.
4. Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression).

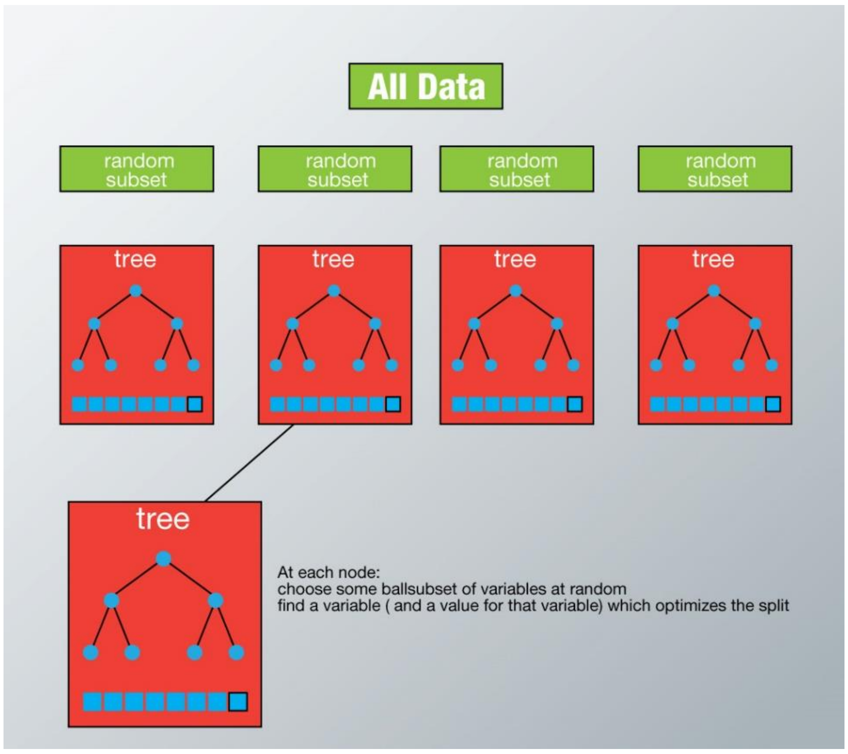

All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some ballsubset of variables at random
find a variable (and a value for that variable) which optimizes the split

Figure No. 6.15 Random Forest

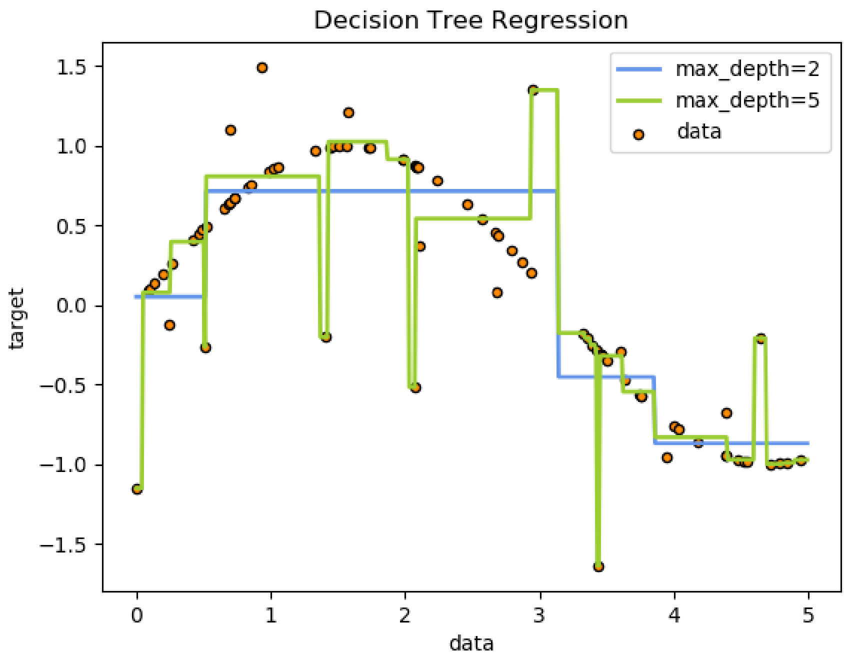
```
# applying Random forest
```

```
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier()

rf_model = rf_clf.fit(trainData, trainLabelE)

print("Training set score for SVC: %f" % rf_model.score(trainData, trainLabelE))

print("Test set score for SVC: %f" % rf_model.score(testData, testLabelE))
```

6.4 Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.



Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

CHAPTER 7

Results

The results for each of the models for test and train data are listed below:

1. Logistic Regression

```
Training score:  0.922302124946
Test score:  0.903371253507
```

2. Random Forest

Training set score: `0.952938296527`
Test set score: `0.928783927244`

3. K- Nearest Neighbour

Training set score for KNN: `0.945755566322`
Test set score for KNN: `0.882435242088`

4. Decision Tree

Training set score: `0.946260516081`
Test set score: `0.912461476039`

Clearly, **Ensemble Random forest** is the best model with highest accurate results among all other models.

CHAPTER 8

Summary

8.1 CONCLUSION

The objective of this project is to find out whether advanced machine learning techniques give better effectiveness for backorders forecasting in the early stage of the supply chain. The results are relevant for circumstances when producers in the supply chain cannot collaborate with other stakeholders, where they have to predict backorders using their existing information. In such cases, the ability to increase forecasting accuracy will result in lower costs and higher customer satisfaction because of more on-time deliveries.

Various analyses were conducted to achieve such a goal. Data pre-processing including binarisation, summary statistics, visualisation, inaccurate values detection, data nor-malisation and data dimensionality reduction, provides a more understandable dataset format with data lacks fulfilled, features scaled and dataset reduced for the preference of future computation. Moreover, during data modelling, different training sets were defined to test model performance on whether imbalanced datasets, a use of cross-validation to avoid over-fitting, as well as model scale adjustments and kernel function selection have all valued the final choice of classifiers.

The answer to the research question ”How can Danish brewers use machine learning techniques, train on historical data, predict backorders in their early stage of the supply chain.” is stated in the report in the chapter proposed solution. From the chapter evaluation, the support vector machine with cubic kernel gives the best results among other classifiers and as well the traditional method, which gives a positive answer to research question two:” Are machine learning predictions more efficient than the traditional fore-casting”.

However, more works could be done to discover rules that can result in kernels’ best performance. The results from this project show, under certain circumstances, some machine learning techniques can give better prediction than that from the traditional ways.

While defining a training set, it is essential to detect the outliers; one needs always to be careful to reduce the dataset since sometimes outliers can also be useful for model training. It can be tricky sometimes to define the best amount of noise in a dataset that can help the classifier perform in the best way. During the definition of a training set, especially in real cases while using real data, data preprocessing is necessary: it helps clean the data also define if the data is imbalanced. An imbalanced dataset can confuse one person with a high overall accuracy rate, where the true positive rate could be much smaller.

In this project, many variables need to be adjusted and controlled. While dealing with missing value, outliers, to dimensionality reduction and model training, many assumptions needed to be made. Generally, among the best performers, the model with the fewest assumptions should be selected. A model that is with the least assumptions is the most robust and can be used in many scenarios. As being small and traditional is a core value of craft brewers, it is important that the prediction method be easy to use. That being said, an approach of using Ensemble predictions might be the best fit for this case.

8.2 FUTURE SCOPE

Various future works can be extended from this project if given more time. First of all, many heavy computations caused from kernel adjustment can be tried out on a better computer, for a more precise algorithm screening. Second of all, more advanced pre-dictions can be achieved. As the models discussed in this project are about predictions from the early stage in the supply chain, which is relying on the producer's existing data. When more collaboration and sharing information can appear in the business in the future, more advanced predictions can be achieved, which also requires more knowledge from not only the business but also how to collect data and choose features. Last but not least, a meeting with the brewer can be arranged to discuss more issues in the predictions from their experience, which can help with feature selection and data col-lection. In conclusion, it is stated that many improvements could be made based on the trained model. One-class training could also be used for the extremely imbalanced training set.

A successful backorder prediction can reduce the bullwhip effect, raise customer and partners satisfaction, and as well help brewers be more proactive in the ever changing market. By being organised and knowing the possible future demand, both operation management and supply chain management can be more efficient by the time saving and fast react

Deep Learning methods can also be incorporated for the same assessment.

CHAPTER 9

References

- https://www.kaggle.com/uciml
- https://www.analyticsvidhya.com/blog/2017/07/debugging-neural-network-with-tensorboard/
- https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/
- https://www.analyticsvidhya.com/blog/2015/11/free-resources-beginners-deep-learning-neural-network/
- Link to the data: https://www.kaggle.com/tiredgeek/predict−bo−trial
- Link to the licience: https://creativecommons.org/licenses/by−nc−sa/4.0/
- Relevant course

Machine Learning A-Z™: Hands-On Python & R In Data Science by UDEMY

APPENDIX

A1. Description of the dataset

The dataset used in this project comes from website Kaggle.

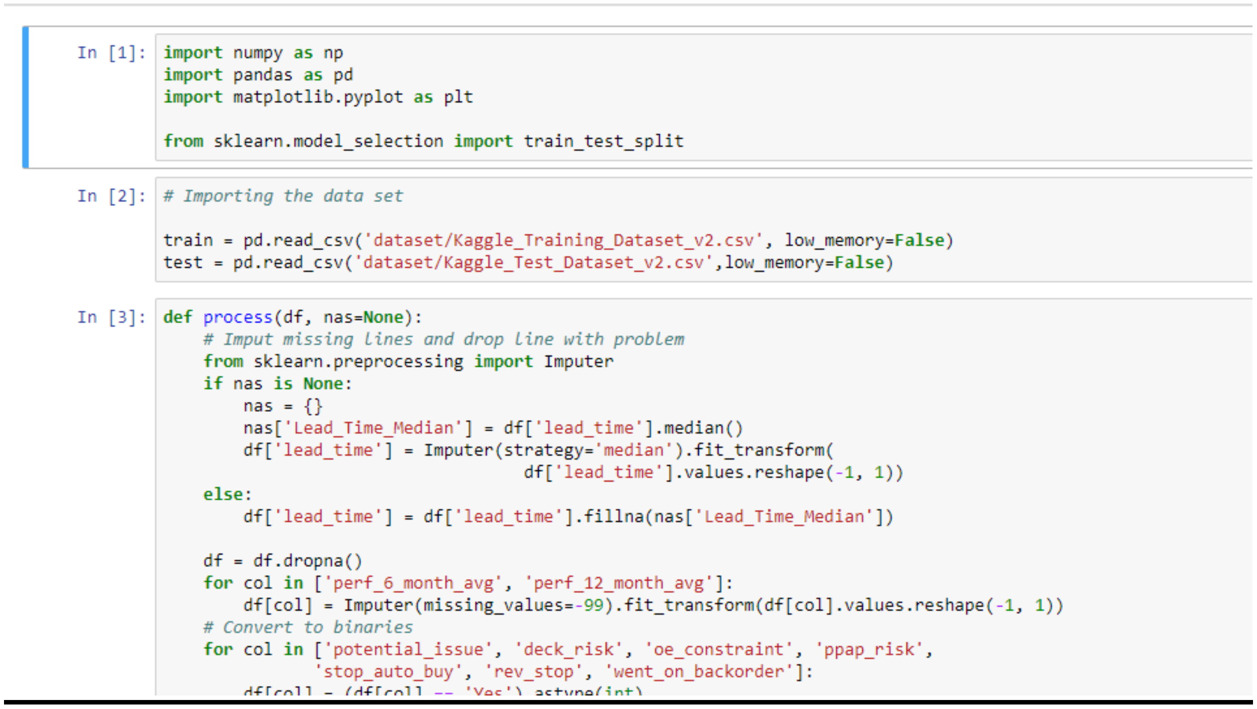
Kaggle is a platform for predictive modelling and analytics competitions in which com-panies and researchers post data and statisticians and data miners compete to produce the best models for predicting and describing the data. This crowdsourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know at the outset which technique or analyst will be most effective.

- Link to the data:
  https://www.kaggle.com/tiredgeek/predict−bo−trial
- Link to the licience:

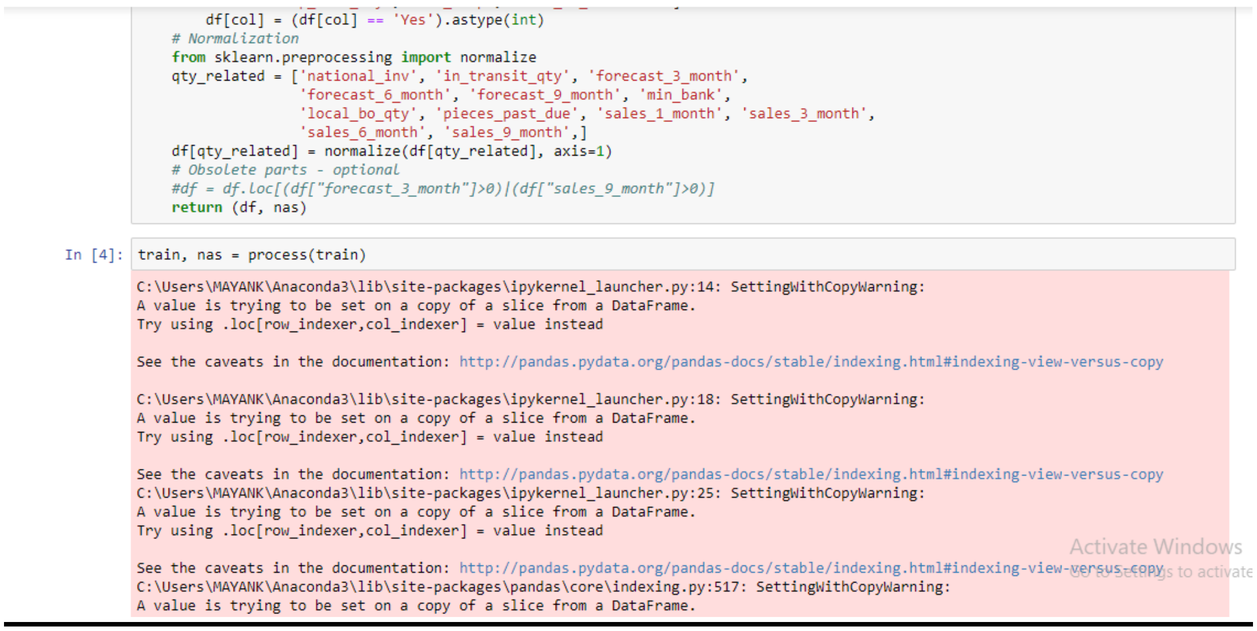
  https://creativecommons.org/licenses/by−nc−sa/4.0/

When import this data to Python, one need to make sure the data output type is column vectors, which is an essential way for the attributes get distinguished into numeric and categorical data. It will do a big favor of convenience for future cleaning and analysis.

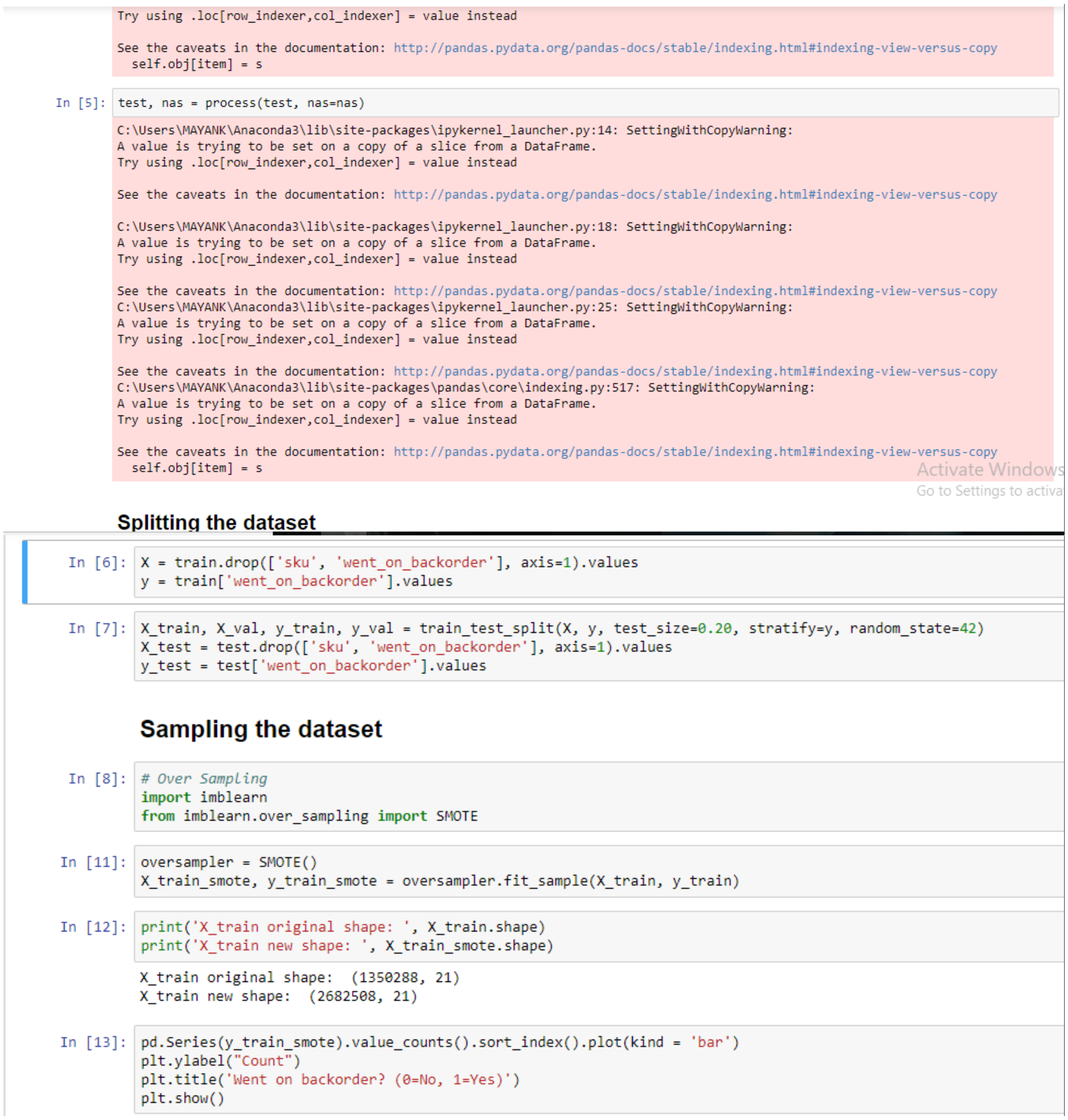
Code Snippets



```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        from sklearn.model_selection import train_test_split

In [2]: # Importing the data set

        train = pd.read_csv('dataset/Kaggle_Training_Dataset_v2.csv', low_memory=False)
        test = pd.read_csv('dataset/Kaggle_Test_Dataset_v2.csv',low_memory=False)

In [3]: def process(df, nas=None):
            # Imput missing lines and drop line with problem
            from sklearn.preprocessing import Imputer
            if nas is None:
                nas = {}
                nas['Lead_Time_Median'] = df['lead_time'].median()
                df['lead_time'] = Imputer(strategy='median').fit_transform(
                                                df['lead_time'].values.reshape(-1, 1))
            else:
                df['lead_time'] = df['lead_time'].fillna(nas['Lead_Time_Median'])

            df = df.dropna()
            for col in ['perf_6_month_avg', 'perf_12_month_avg']:
                df[col] = Imputer(missing_values=-99).fit_transform(df[col].values.reshape(-1, 1))
            # Convert to binaries
            for col in ['potential_issue', 'deck_risk', 'oe_constraint', 'ppap_risk',
                        'stop_auto_buy', 'rev_stop', 'went_on_backorder']:
```



```
                df[col] = (df[col] == 'Yes').astype(int)
            # Normalization
            from sklearn.preprocessing import normalize
            qty_related = ['national_inv', 'in_transit_qty', 'forecast_3_month',
                           'forecast_6_month', 'forecast_9_month', 'min_bank',
                           'local_bo_qty', 'pieces_past_due', 'sales_1_month', 'sales_3_month',
                           'sales_6_month', 'sales_9_month',]
            df[qty_related] = normalize(df[qty_related], axis=1)
            # Obsolete parts - optional
            #df = df.loc[(df["forecast_3_month"]>0)|(df["sales_9_month"]>0)]
            return (df, nas)

In [4]: train, nas = process(train)

        C:\Users\MAYANK\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: SettingWithCopyWarning:
        A value is trying to be set on a copy of a slice from a DataFrame.
        Try using .loc[row_indexer,col_indexer] = value instead

        See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

        C:\Users\MAYANK\Anaconda3\lib\site-packages\ipykernel_launcher.py:18: SettingWithCopyWarning:
        A value is trying to be set on a copy of a slice from a DataFrame.
        Try using .loc[row_indexer,col_indexer] = value instead

        See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
        C:\Users\MAYANK\Anaconda3\lib\site-packages\ipykernel_launcher.py:25: SettingWithCopyWarning:
        A value is trying to be set on a copy of a slice from a DataFrame.
        Try using .loc[row_indexer,col_indexer] = value instead

        See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
        C:\Users\MAYANK\Anaconda3\lib\site-packages\pandas\core\indexing.py:517: SettingWithCopyWarning:
        A value is trying to be set on a copy of a slice from a DataFrame.
```



```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

In [5]:
```
test, nas = process(test, nas=nas)
```

```
C:\Users\MAYANK\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

C:\Users\MAYANK\Anaconda3\lib\site-packages\ipykernel_launcher.py:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\Users\MAYANK\Anaconda3\lib\site-packages\ipykernel_launcher.py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\Users\MAYANK\Anaconda3\lib\site-packages\pandas\core\indexing.py:517: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

Splitting the dataset

In [6]:
```
X = train.drop(['sku', 'went_on_backorder'], axis=1).values
y = train['went_on_backorder'].values
```

In [7]:
```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, stratify=y, random_state=42)
X_test = test.drop(['sku', 'went_on_backorder'], axis=1).values
y_test = test['went_on_backorder'].values
```

Sampling the dataset

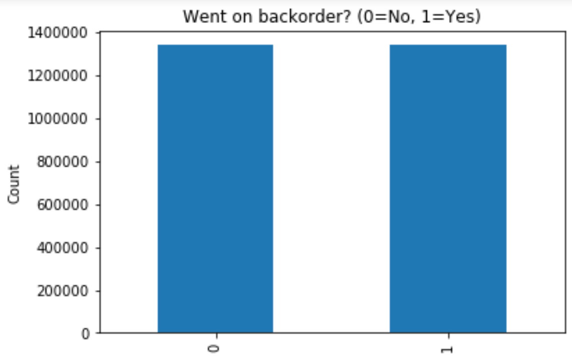
In [8]:
```
# Over Sampling
import imblearn
from imblearn.over_sampling import SMOTE
```

In [11]:
```
oversampler = SMOTE()
X_train_smote, y_train_smote = oversampler.fit_sample(X_train, y_train)
```

In [12]:
```
print('X_train original shape: ', X_train.shape)
print('X_train new shape: ', X_train_smote.shape)
```

```
X_train original shape:  (1350288, 21)
X_train new shape:  (2682508, 21)
```

In [13]:
```
pd.Series(y_train_smote).value_counts().sort_index().plot(kind = 'bar')
plt.ylabel("Count")
plt.title('Went on backorder? (0=No, 1=Yes)')
plt.show()
```



Logistic Regression

```
In [14]: from sklearn.linear_model import LogisticRegression
```

```
In [15]: classifier_linear = LogisticRegression(penalty='l1')
         classifier_linear.fit(X_train_smote, y_train_smote)

Out[15]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

```
In [16]: from sklearn.metrics import roc_auc_score
```

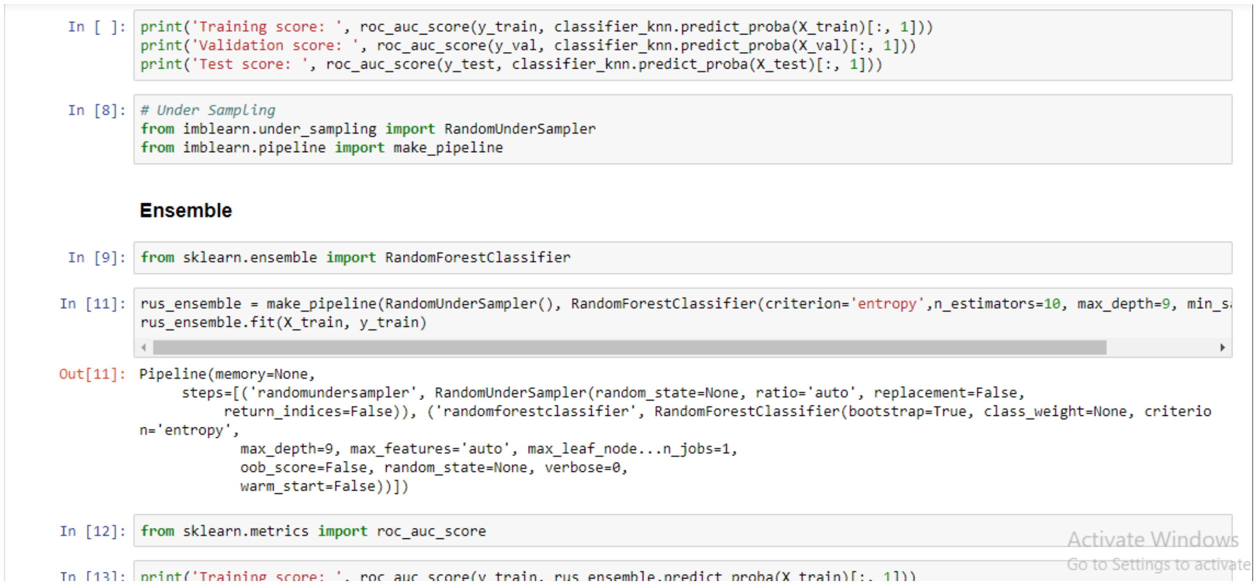
```
In [19]: print('Training score: ', roc_auc_score(y_train_smote, classifier_linear.predict_proba(X_train_smote)[:, 1]))
         print('Validation score: ', roc_auc_score(y_val, classifier_linear.predict_proba(X_val)[:, 1]))
         print('Test score: ', roc_auc_score(y_test, classifier_linear.predict_proba(X_test)[:, 1]))

         Training score:  0.922302124946
         Validation score:  0.917250492486
         Test score:  0.903371253507
```

Decision Tree

```
In [20]: from sklearn.tree import DecisionTreeClassifier
```

```
In [21]: classifier_dt = DecisionTreeClassifier(max_depth=8, min_samples_leaf=5)
         classifier_dt.fit(X_train_smote, y_train_smote)

Out[21]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=5, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')
```

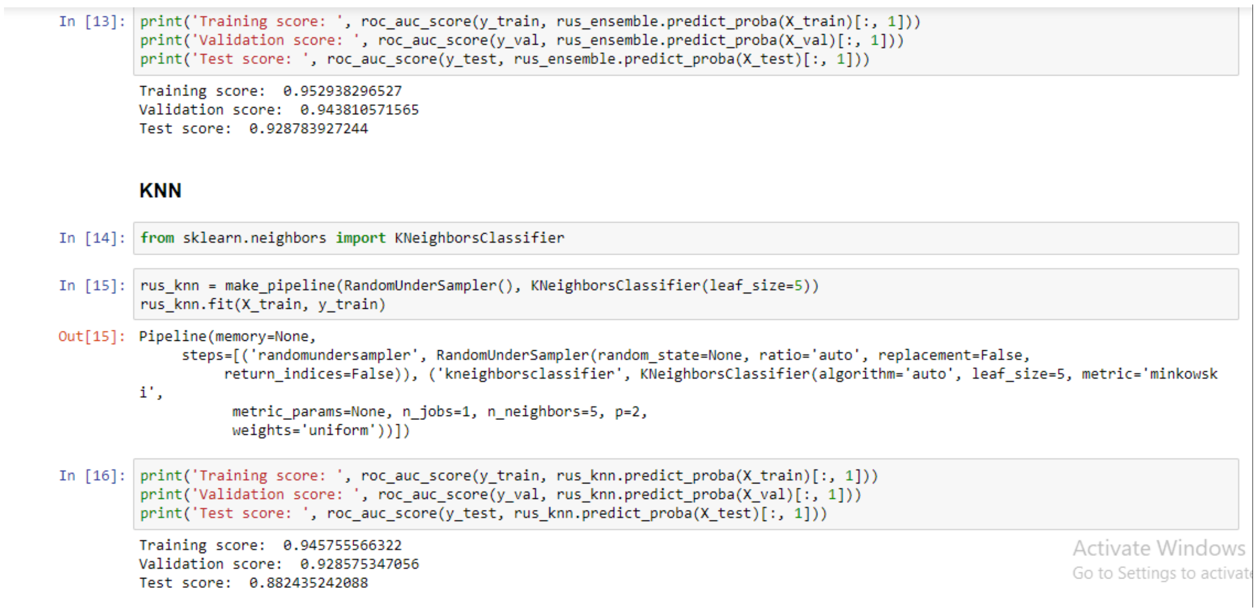
```
In [22]: print('Training score: ', roc_auc_score(y_train, classifier_dt.predict_proba(X_train)[:, 1]))
         print('Validation score: ', roc_auc_score(y_val, classifier_dt.predict_proba(X_val)[:, 1]))
         print('Test score: ', roc_auc_score(y_test, classifier_dt.predict_proba(X_test)[:, 1]))

         Training score:  0.946260516081
         Validation score:  0.931832492907
         Test score:  0.912461476039
```



```
In [ ]: print('Training score: ', roc_auc_score(y_train, classifier_knn.predict_proba(X_train)[:, 1]))
        print('Validation score: ', roc_auc_score(y_val, classifier_knn.predict_proba(X_val)[:, 1]))
        print('Test score: ', roc_auc_score(y_test, classifier_knn.predict_proba(X_test)[:, 1]))

In [8]: # Under Sampling
        from imblearn.under_sampling import RandomUnderSampler
        from imblearn.pipeline import make_pipeline
```

Ensemble

```
In [9]: from sklearn.ensemble import RandomForestClassifier

In [11]: rus_ensemble = make_pipeline(RandomUnderSampler(), RandomForestClassifier(criterion='entropy',n_estimators=10, max_depth=9, min_s
         rus_ensemble.fit(X_train, y_train)

Out[11]: Pipeline(memory=None,
              steps=[('randomundersampler', RandomUnderSampler(random_state=None, ratio='auto', replacement=False,
                   return_indices=False)), ('randomforestclassifier', RandomForestClassifier(bootstrap=True, class_weight=None, criterio
         n='entropy',
                     max_depth=9, max_features='auto', max_leaf_node...n_jobs=1,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False))])

In [12]: from sklearn.metrics import roc_auc_score

In [13]: print('Training score: ', roc_auc_score(y_train, rus_ensemble.predict_proba(X_train)[:, 1]))
```



```
In [13]: print('Training score: ', roc_auc_score(y_train, rus_ensemble.predict_proba(X_train)[:, 1]))
         print('Validation score: ', roc_auc_score(y_val, rus_ensemble.predict_proba(X_val)[:, 1]))
         print('Test score: ', roc_auc_score(y_test, rus_ensemble.predict_proba(X_test)[:, 1]))

         Training score:  0.952938296527
         Validation score:  0.943810571565
         Test score:  0.928783927244
```

KNN

```
In [14]: from sklearn.neighbors import KNeighborsClassifier

In [15]: rus_knn = make_pipeline(RandomUnderSampler(), KNeighborsClassifier(leaf_size=5))
         rus_knn.fit(X_train, y_train)

Out[15]: Pipeline(memory=None,
              steps=[('randomundersampler', RandomUnderSampler(random_state=None, ratio='auto', replacement=False,
                   return_indices=False)), ('kneighborsclassifier', KNeighborsClassifier(algorithm='auto', leaf_size=5, metric='minkowsk
         i',
                    metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                    weights='uniform'))])

In [16]: print('Training score: ', roc_auc_score(y_train, rus_knn.predict_proba(X_train)[:, 1]))
         print('Validation score: ', roc_auc_score(y_val, rus_knn.predict_proba(X_val)[:, 1]))
         print('Test score: ', roc_auc_score(y_test, rus_knn.predict_proba(X_test)[:, 1]))

         Training score:  0.945755566322
         Validation score:  0.928575347056
         Test score:  0.882435242088
```