Udacity Machine Learning Nanodegree

Capstone Project Report: Lending Club Dataset

Kenneth Gjaeringen
5th December 2017

Table of Content

I. Definition 3

I.I. Problem Statement

I.II. Datasets and Inputs

I.III. Solution Statement

I.IV. Benchmark Model

I.V. Evaluation Metrics

I.VI. Project Design

II. Analysis 9

II.I. Data Exploration

II.II. Exploratory Visualisation

II.III. Algorithms and Techniques

Ensemble methods

Decision Trees methods

Nearest Neighbour methods

II.IV. Benchmark

III. Methodology 15

III.I. Data Preprocessing

III.II. Implementation

III.III. Refinement

IV. Results 21

IV.I. Model Evaluation and Validation

IV.II. Justification

V. Conclusion 28

V.I. Free-Form Visualisation

V.II. Reflection

V.III. Improvement

VI. References 30

Appendix A: Initial Dataset 31

Appendix B: Dataset Transformation 32

Appendix C: Dataset Transformation Removing NAN 33

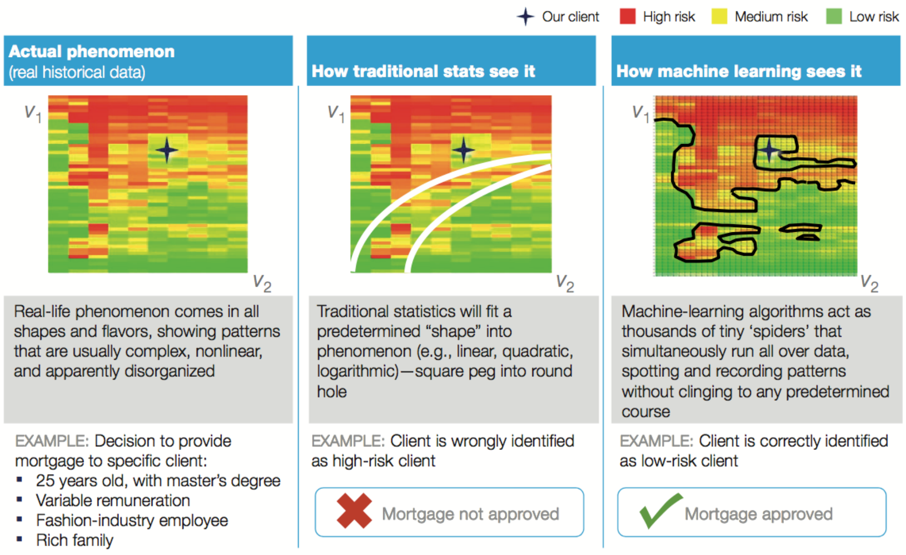
Appendix D: Updated Dataset Excluding the 2015 Cohort 34

Appendix E: List of Algorithms Enabling predict_proba 35

I. Definition

In the consumer financing area being able to identify high risk customers during the origination process would benefit the originator by reducing foreclosure costs and loss severity. A consumer lending business face a large number of decisions when lending money to a consumer. Making sense of all these consumer variables makes it necessary to rely on models and algorithms rather than human discretion. Traditionally a lender base their lending business on standard underwriting policies (see middle chart below showing the traditional risk curve). These policies are quite rigid in that they classify the risk based on various scales, e.g. debt to income no more than 50%, and borrower characteristics such as being employed. In-house risk or rating agency models are then applied to access credit worthiness on consumers, but on a weighted average basis for the entire loan portfolio. None of these standard industry models are able to identify specific consumers who are at risk of default. This Capstone project aim to apply machine learning algorithms to better define consumers who might default. Being able to predict at the underwriting process whether a borrower is likely to be at high risk of default would give the lender further options whether to carry on with the loan application or increase the interest rate to reflect the additional risk. Furthermore, these borrowers could be tracked a lot more closely during their term to ensure a successful redemption of the loan.

In a paper published by McKinsey & Company[1] they compared the traditional approach, see middle chart below, to what a machine learning algorithm could do, right hand chart below. Machine learning offer the opportunity to give a lender a deeper insight into their data. The comparison below demonstrates the opportunity applying a machine learning model to borrower data. Using the available data (present and past) a machine learning algorithm should be able to look at a borrower and determine whether they are a creditworthy borrower or not. Whilst in a traditional approach an applicant might not be considered an eligible applicant due to being wrongly identified as high risk, but using a machine learning approach pockets of creditworthy borrowers (outside the lender credit policy middle chart) can be identified, as can be seen from the third chart below.



Our client
High risk
Medium risk
Low risk
Actual phenomenon (real historical data)
How traditional stats see it
How machine learning sees it
V_1
V_2
Real-life phenomenon comes in all shapes and flavors, showing patterns that are usually complex, nonlinear, and apparently disorganized
Traditional statistics will fit a predetermined "shape" into phenomenon (e.g., linear, quadratic, logarithmic)—square peg into round hole
Machine-learning algorithms act as thousands of tiny 'spiders' that simultaneously run all over data, spotting and recording patterns without clinging to any predetermined course
EXAMPLE: Decision to provide mortgage to specific client:
25 years old, with master's degree
Variable remuneration
Fashion-industry employee
Rich family
EXAMPLE: Client is wrongly identified as high-risk client
Mortgage not approved
EXAMPLE: Client is correctly identified as low-risk client
Mortgage approved

Another area where machine learning can help is the categorisation of borrowers beyond the normal industry classification (stratification). Through feature engineering the machine learning model can find unique borrower clusters that can aid the future product development or targeted for cross selling purposes.

Obviously the above approach assumes there is a historic database to feed the machine learning models. However, for newer entrants into the financial market lack of data shouldn't be an issue as similar datasets can be acquired to test the machine learning models and verify whether the business model is viable or not.

Loan default classification would also be useful for accounting purposes, such as the accounting rule IFRS9, being able to predict based on past loan performance what is the likely outcome and loan provision that would be required for the

[1] McKinsey & Company The Future of Bank Risk Management working paper (http://www.mckinsey.com/business-functions/risk/our-insights/the-future-of-bank-risk-management)

accounts. However, in order to make this work one would also need a full transaction history along with the static borrower data. As detailed transaction history isn't available this wouldn't be possible to carry out.

For a business to be successful, credit decisions must be based on a model that is able to capture the customers risk profile in a timely and accurate manner.

My background is structured finance and data analytics within in the mortgage industry and this dataset is an opportunity to apply machine learning models to improve risk analytics, customer segmentation and loan pricing. Although this project will be focused on classifying a borrower during the application stage whether or not the applicant has a high likelihood of going into default. As the dataset already has the outcome for most of the Lending Club loans I will apply supervised learning models.

I.I. Problem Statement

This Capstone project aim to predicting whether a borrower is likely to default or not using loan data available at origination (although restricted to data available in the Kaggle dataset). From the various inputs a borrower can either be classified as in default or not and is a binary problem or classification problem. In normal circumstances borrowers who tend to default usually have similar characteristics and hence the reason for going down the classification route. However, there will be borrowers with good credit features that will experience life changing events such as illness, divorce, death, unemployment, etc. which will have an impact on the loan performance. However, these factors are outside any model capability due to their random occurrence of these events (considered random due to lack of detailed personal data). I would expect cases like these would normally be outliers.

Using the approach defined by Mitchell (1997) the machine learning problem can be defined as follows:

Task: Classify applicants likely to default based on the training experience;

Performance measure: percentage classification accuracy using a number of measures detailed in the Evaluation Metrics section;

Training experience: Lending Club dataset with final loan status with which the algorithm will train itself;

Feature engineering: reducing the attributes to key variables that help the algorithm prediction will improve model performance and make it less complex[2];

Target function: using the various variables a target function will be generated based on the aim of optimising the loss function, i.e. adjusting the various input variable factors to minimise the incorrect classification.

Target function representation: Supervised learning classification algorithms, using a handful of algorithms to baseline the output to further tune the best performing algorithms;

I.II. Datasets and Inputs

The Lending Club dataset was obtained from kaggle.com and at the time of download contains 74 fields and 887,379 loans. The dataset contain completed loan data for all loans issued through the 2007-2015, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. The file containing loan data through the "present" contains complete loan data for all loans issued through the previous completed calendar quarter. Additional features include number of finance inquiries, address including zip codes, and state, and collections among others.
In this dataset made available on kaggle.com consisting of loan level borrower details, summary transaction history and account-balance data for individual Lending Club consumers. The raw data was available as a CSV flat-file.

There is a data dictionary on the Kaggle website but due to version control issues this document does not reflect the current dataset and is only partially relevant. Only data features available at the time of origination of the loan will be

[2] Guyon and Elisseeff "An Introduction to Variable and Feature Selection" http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

considered during this project plus the final loan status. Any other features related to loan performance will be excluded and deemed irrelevant as it won't represent the data at the point of origination. The field called "loan_status"

To mention a few features from the dataset that would be available at the time of origination:
- Loan grades;
- Term;
- Interest rate;
- Debt to income (DTI);
- Employment Length;
- Home ownership;
- Income verification;

After importing the entire dataset features not required will be removed from the dataset, any remaining features with string categories (such as Home ownership which contains "RENT", "MORTGAGE" and "OWN") will have its own column for each category with a boolean value, i.e. where there field value says "RENT" the "Home_Ownership_RENT" feature will have a "1" otherwise "0". Any value fields will be normalised. Features like Zip code which can be converted into a Region will be kept as it is as Region by itself shouldn't have an extensive impact whether the borrower will default or not. Normally the Region is of concern if you have a high concentration / exposure. Furthermore, borrower defaults in certain towns and regions may be related to a company closing five years from now, it's unlikely that a specific forecast or even a specific distribution of probabilities can be constructed that factor or for any individual factor (Chapter 12 in Davidson & Levin 2014).

Once the dataset has gone through the "ETL process" feature engineering can take place to ensure noise is reduced, improve predictability and reduce training time with less data. Techniques such as Principal Component Analysis and oversampling methods will be applied to the dataset to further understand the loan population and clustering, reduce any imbalance and reduce the dimensionality of the dataset.

I.III. Solution Statement

To tackle the above problem statement described above we will apply supervised learning algorithms to classify borrowers who are likely to default at the application stage. The challenge of supervised learning is to find a function that generalises beyond the training set, so that the resulting function also accurately maps out-of-sample inputs to out-of-sample outcomes.

However, before any machine learning algorithm can be applied a certain amount of data preprocessing needs to be applied to ensure better predictability. The Lending Club dataset contains more data points than is required for this project. In order to solve the default question at the origination stage and minimise look-ahead bias, I will exclude any loan performance data should be excluded as this won't be available during the loan origination process. For the remaining data features the next stage would be to normalise value fields and decompose categorical attributes, i.e. if there are 2 categories making up a Loan Type feature (containing repayment or interest only) an extra field will be generated for each category with "1" where the feature is true and "0" when it's not true.

The population of defaulted borrowers is a smaller segment of the total pool and therefore creates an imbalance of the classes in the dataset. Oversampling techniques such as Adaptive Synthetic (ADASYN) and Synthetic Minority Over Sampling Technique (SMOTE) will be applied to rebalance the dataset. Oversampling techniques uses key data points and creates synthetic instances of data points to rebase the imbalanced data class, i.e. defaulted borrowers. Oversampling randomly replicates minority instances to increase their population. Undersampling randomly downsamples the majority class[3]. Although oversampling techniques generates extra data undersampling can make the independent variable look like they have a higher variance than they do[3].

I.IV. Benchmark Model

After the feature engineering process has been satisfactorily completed a set of baseline models run against the dataset to establish which model is likely to perform best. As it's difficult to determine which supervised learning model will perform best a set of models will be used:
- Ensemble methods
- Decision tree methods

[3] Learning from Imbalanced Classes, Tom Fawcett. https://svds.com/learning-imbalanced-classes/

- Nearest Neighbour methods
- Stochastic Gradient Descent method
- Support Vector Machine method
- Gaussian processes
- Naive Bayes
- Neural Network model

No tuning will take place during this model selection and will form part of the benchmarking of key models.

From this we will reduce the models to a handful of models to further tune and improve with the aim to end up with one machine learning algorithm that can be applied to the test set to predict borrowers who are likely to default. This output can be used in two ways: which borrowers is likely to default and what is the default rate of a portfolio/vintage/cohort.

I.V. Evaluation Metrics

In order to establish the classification performance of a machine learning model key performance metrics needs to be extracted for each benchmark and solution model. One might think accuracy is the obvious option in terms of measuring generic predictive ability, but accuracy can be very misleading and in particular for imbalanced datasets like this project[4]. With the aim of predicting applicants likely to default…..

For this project the following performance metrics will be applied for all models:

- F1-Score
- Area Under the ROC Curves (AUC)

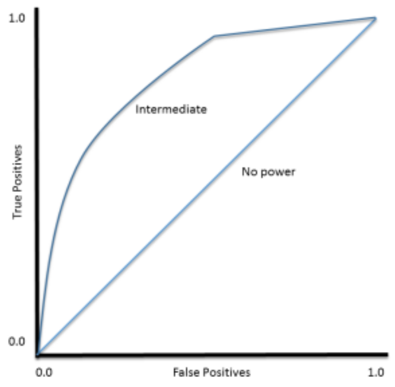
F1 SCORE

F1 Score is the weighted average of precision and recall. Expressed as:

$$F1 - Score = 2 * ((Precision * Recall)/(Precision + Recall))$$

RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

ROC curves is a standard technique for summarising classifier performance over a range of tradeoffs between true positive and false positive error rates[5]. For this project we will be



For a perfect classifier the ROC curve will go straight up the True Positive axis and then along the False Positive axis. A classifier with no power will sit on the diagonal. Furthermore, by adding several machine learning algorithms to this ROC curve will show the optimal algorithm to use going forward.

[4] https://www.svds.com/the-basics-of-classifier-evaluation-part-1/

[5] Data Mining For Imbalanced Datasets: An Overview, Nitesh V. Chawla; http://www3.nd.edu/~dial/publications/chawla2005data.pdf

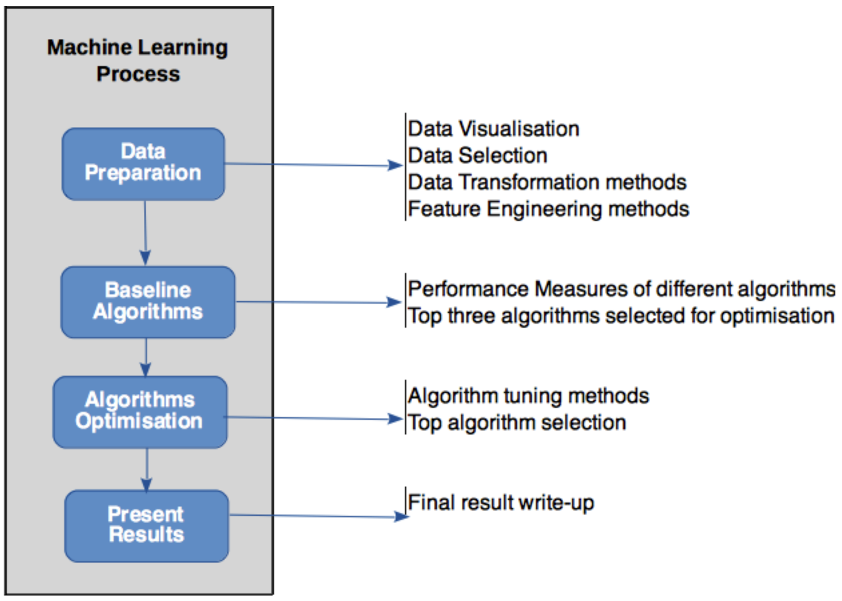
THE AREA UNDER THE ROC CURVE (AUC)

The AUC It is equal to the probability that a random positive example will be ranked above a random negative example.

Although none of the above would give the correct answer. According to Tom Fawcett[6] a better measure would be to calculate the expected costs which is a mixture of confusion matrix inputs and real-world consequences of the classifier decision. Dealing with defaulted borrowers this can be calculated by working out, for each loan, the Probability of Default (PD)(borrower specific) multiplied by Loss Given Default (LGD) (if secured then asset specific plus foreclosure costs) which will give you the expected loss. The expected loss value would be the real-world consequence, however, these values or variables to calculate such ratios are not available in the Lending Club dataset, and we are then left with applying the above measures.

I.VI. Project Design

In order to ensure the best result from a dataset answering a specific problem statement using machine learning (ML) algorithms a rigid workflow needs to be followed otherwise the machine learning algorithms won't be fully optimised and might even cause algorithms to be discarded due to poor workflow process. Below is a summarised view of the proposed workflow which can be applied to any ML problem. The workflow below assumes the problem statement that will be answered by the ML algorithm has been defined in advance.



1. Data Preparation

Most of the project time will be spent in this stage to better understand the data, exclude any data that is not relevant and make the dataset ready for a machine learning algorithm.

The following data preparation steps will be applied:
- Data preprocessing, i.e. clean data, data formatting, data rebalancing, outlier analysis and removal, dimensionality reduction;
- Data transformation, i.e. data normalisation and discretise data to handle nominal values;
- Data summarisation, i.e. create various plots to unravel any obvious relationships in the data.

[6] https://www.svds.com/the-basics-of-classifier-evaluation-part-1/

2. Baseline Algorithms

At the outset of a ML project it might not be obvious which ML algorithm that might perform best. As described in the Benchmark Model section I will apply a number of supervised learning algorithms without any tuning to analyse which model performs the best.

The ML models will be evaluated against a standard set of metrics to further shortlist ML algorithms to optimise (see Evaluation Metrics section for further details).

3. Algorithms Optimisation

From the baseline algorithm process one ML algorithms will be taken forward to be optimised by exploring each algorithm variables. As in the baseline algorithm stage the same evaluation metrics will be applied. The aim of this workflow process is to arrive at a model that generalises enough to be applied in a production environment and used as part of the loan underwriting process.

4. Present Results

A detailed report outlining the workflow process and document steps taken to achieve the output after algorithm optimisation.

Furthermore, the final document will detail operationalisation of the predictive model created, elaborating the process of data gathering and data quality, and how to use the predictive model.

II. Analysis

This section seek to explore the Lending Club dataset (the dataset) in more detail before any machine learning algorithm can be applied. The dataset contains a mixture different types of data; application stage data and up-to-date performance data. This project will not consider any performance data as it's concerned with the data as at the application stage. The remaining dataset will be explored in detail.

II.I. Data Exploration

The dataset downloaded from the Kaggle.com website contains 887,379 rows and 74 fields. Out of these fields 49 of them are related to current performance and other non-relevant data due to its lack of data, i.e. too many blank fields. The only forward looking field that will be retained is the loan_status field as this field will act as the main field the various algorithms will train itself against. The loan_status field contains the current status for each loan. Appendix A displayed the remaining 25 fields and using a Python library called brewery (please note it takes a while to run brewery on this dataset due to its size) they key attributes (null records, strings and unique fields) can be summarised in a tabular format.

Using the output from this Python brewery library enables one to view data fields that can be normalised or discretised. These fields can be identified by their distinct count values, e.g. the field term has two distinct values; '36 months' and '60 months'. However, before any normalisation or discretisation can take place one must first extract the distinct values. The next face would be to extract these distinct values. The following fields had their distinct values extracted:

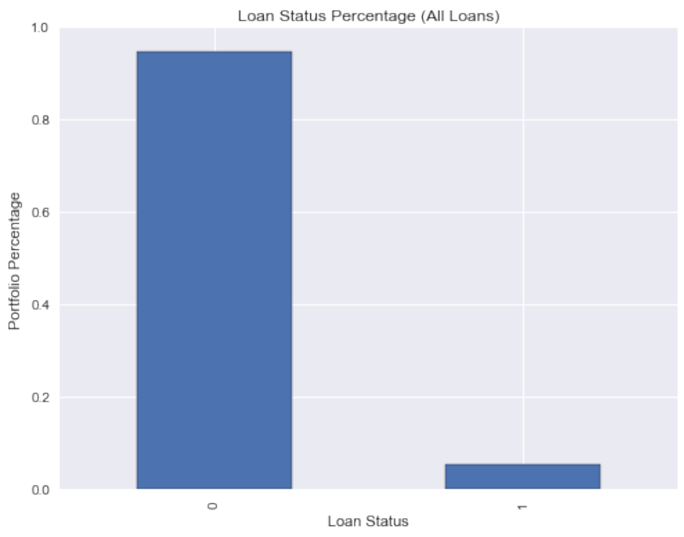
- term - The number of payments on the loan. Values are in months and can be either 36 or 60;
- grade - Lending Club assigned loan grade;
- sub_grade - Lending Club assigned loan subgrade;
- emp_length - Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.;
- home_ownership - The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.;
- verification_status - Indicates if the borrower income was verified by Lending Club, not verified, or if the income source was verified;
- loan_status - Current status of the loan (at the point of dataset extraction);
- purpose - A category provided by the borrower for the loan request;
- initial_list_status - The initial listing status of the loan. Possible values are – W, F;
- application_type - Indicates whether the loan is an individual application or a joint application with two co-borrowers;
- issue_d - The month which the loan was funded, i.e. completion month; and
- addr_state - The state provided by the borrower in the loan application.

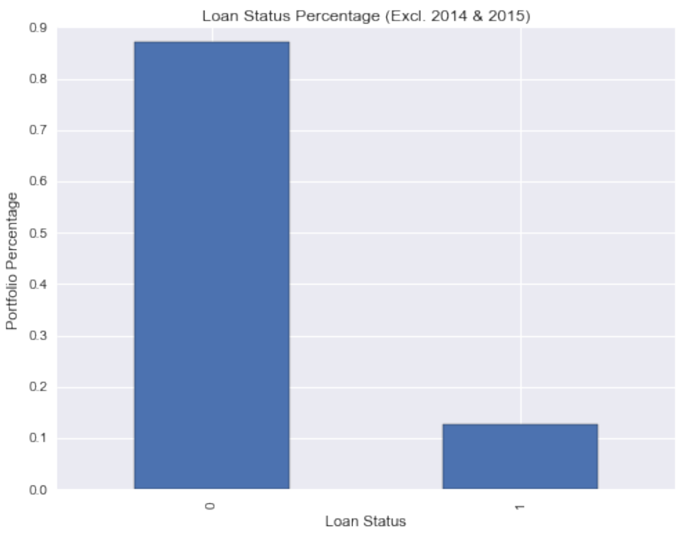
Each of the above fields goes through a normalisation process or discretisation. In the case of the loan_status field values will be converted too boolean values being either 0 for not in default or 1 for default. The fields where there are multiple categories combined into one field are defined as a boolean value in its own field. Once this exercise is completed the original field will be dropped where it has been superseded by multiple fields. The result of this cane be viewed in Appendix B which shows the brewery library output increasing the field conversion to 71 then remove the extra fields resulting in 54 fields remaining. However, there is a number of NAN values in three of the fields and although the fields 'dti' and 'annual-inc' could be supplement with joint account data the number of NAN fields in the 'annual_inc' does not correspond to the NAN values in 'dti'. As the number of of rows impacted only amounted to 29 and is an immaterial number in relation to the number of loans the decision was taken to remove those rows with NAN reducing the dataset total from 887,379 down to 887,350, see Appendix C.

In the next section it is proposed that the 2014 and 2015 completion cohort is excluded from the dataset as these loans are yet to experience any material default activity, which results in a reduction of 74% loans down to 230,628 loans. This dataset containing 2014 and 2015 completion cohorts can be used as a final and 3rd dataset to predict the potential default levels for those two cohorts once the final model has been optimised and finalised.

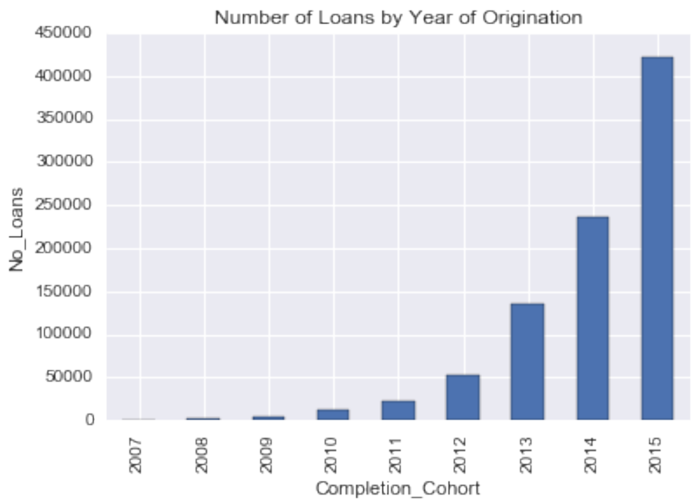
II.II. Exploratory Visualisation

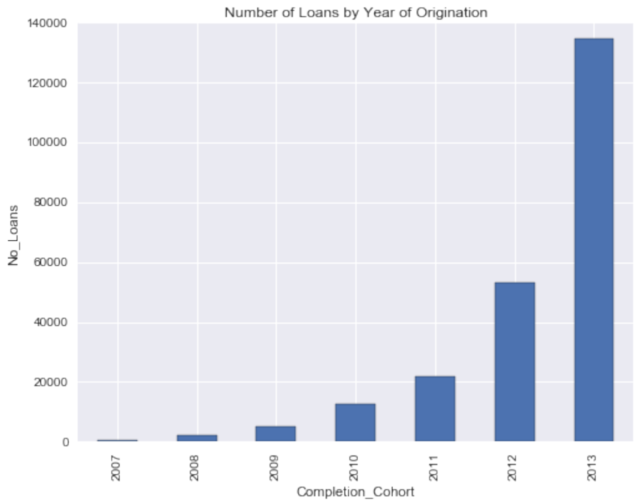
After the above data exploration exercise the current dataset that will be used going forward consist of 887,350 loan records. Out of these there are 94.7% of loan records that are not considered as in default, i.e. classified as neither default or charge off, leaving 5.3% in default. As the target field to train against it can be considered an imbalanced dataset (explored further in feature engineering section). Removing the 2014 and 2015 completion cohort increases the default percentage to 12.8% as can be seen in the right hand chart. The two charts below shows the unbalanced classes and running this dataset through an algorithm the more common class will often yield very high accuracy, which will be dealt with during the evaluation stage.





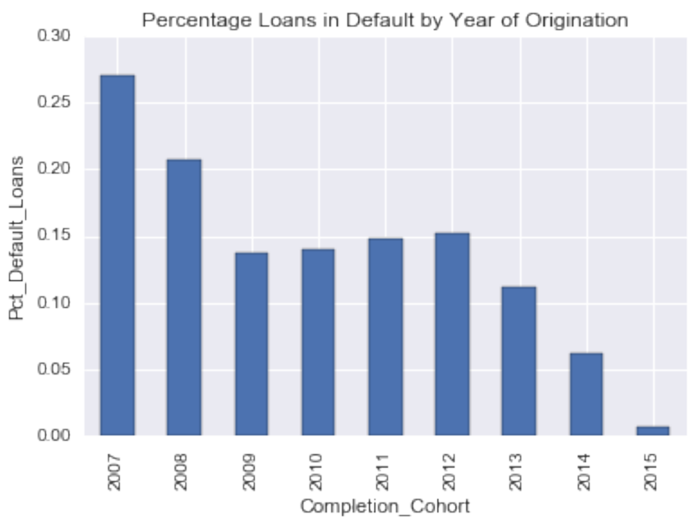
The Lending Club was founded back in 2006 and the below bar chart indicates the number of loans originated since 2007. The upward trajectory of the Lending Club origination is a mixture of the peer-to-peer lending platform having grown in its acceptance and investors looking for higher returns on their investment.

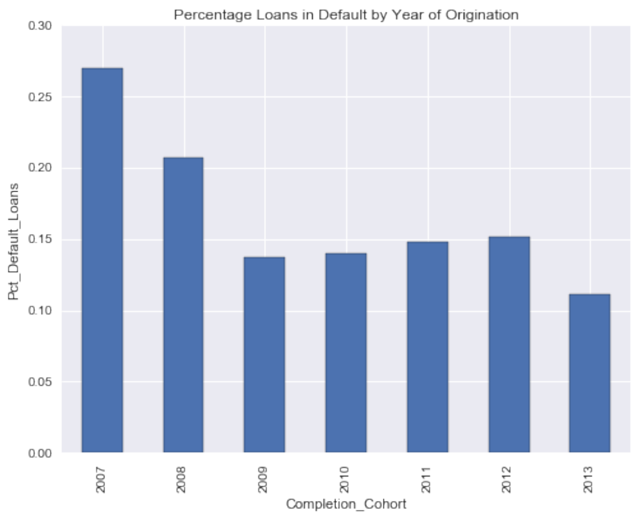




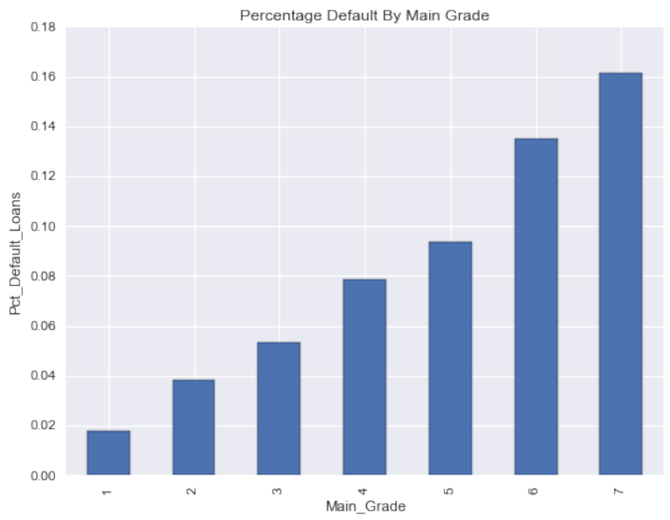
With any new lender/lending platform lending criteria will mature over time as borrowers are progressing through the loan term. The bar chart below shows this trend clearly, albeit the the credit crunch would have impacted the overall default levels during this period between 2008 and 2015.

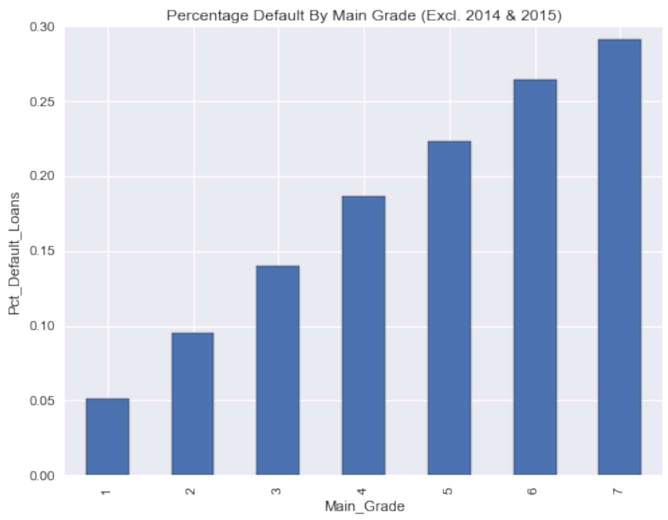
The trend from 2009 through to 2012 might persist unless Lending Club has tightened up their underwriting or arrears recovery processes. However, these numbers might serve as a guide as to default levels one would expect for these type of loans.

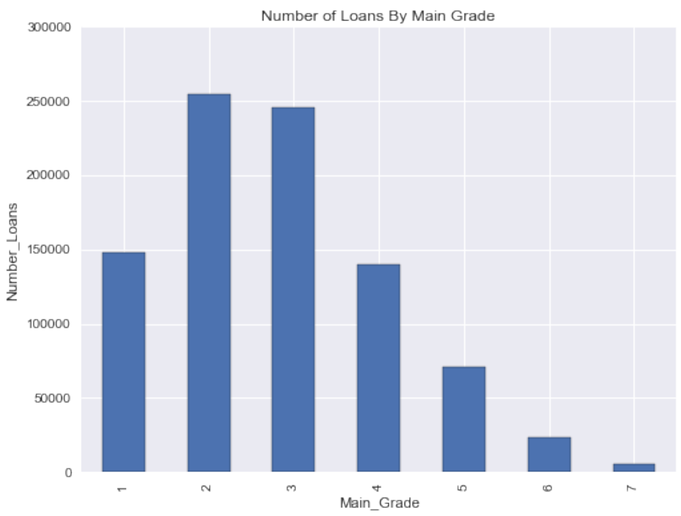


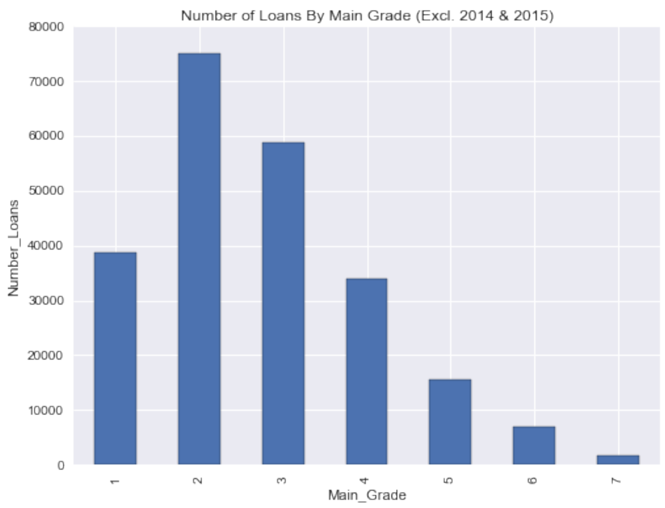


By comparing the Loan_Status with the Main_Grade (1 = lowest risk and 7 = highest grade) you end up with what you would expect of such a ranking system, i.e. Main_Grade 7 has the highest percent of defaulted loans for its category. It is assumed that the Main_Grade and Sub_Grade is a derivative of the US FICO score (consumer credit scoring). However, majority of the loans originated are within grade 1 and 3, see Number of Loans By Main Grade chart.

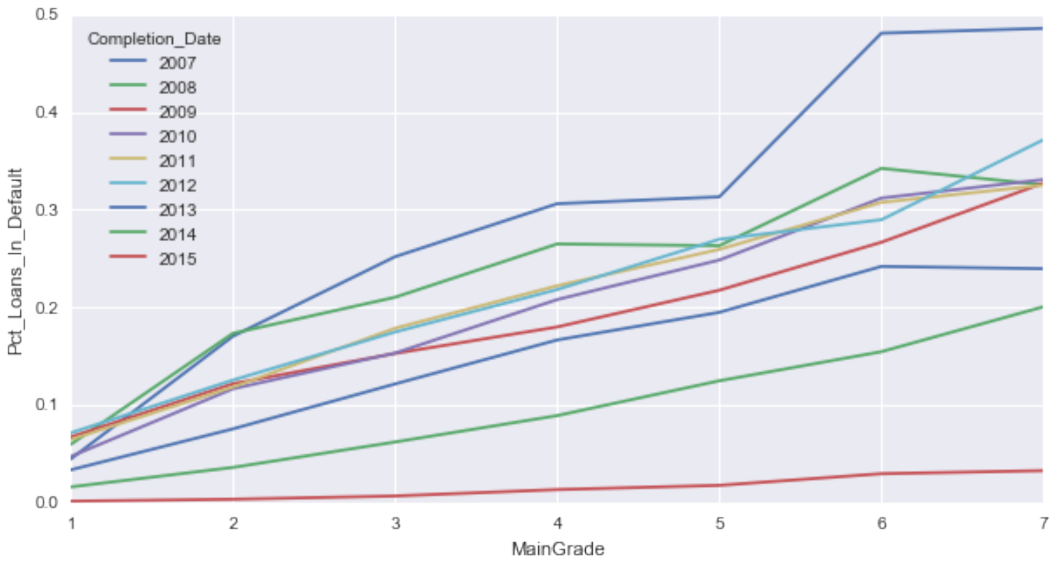




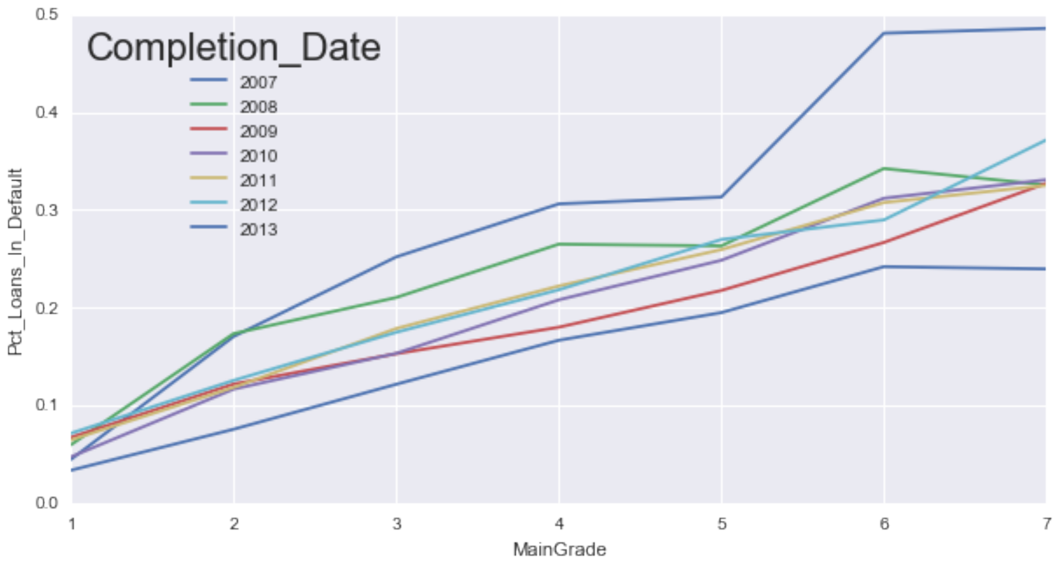




If one delves deeper into the Loan_Status category and group it by the Completion Year and MainGrade the following chart can be extracted. Again the default occurs where it is expected. Apart from 2014 and 2015 completions the remaining completion cohorts a fairly similar. The 2014 completion cohort is different to the previous completion years. 2015 appears to be completely different. Although it might be still too early to determine if 2014 and 2015 is different to the other completion years. It might even be worthwhile to exclude the 2014 and 2015 completion cohort entirely as it further imbalances the dataset.



Removing the 2014 and 2015 completion cohorts results in the following chart.



II.III. Algorithms and Techniques

This Capstone project is concerned with identifying potential borrowers that might be in danger of going into default and hence this is a classification problem applying supervised learning algorithms. Although there are general guidelines[7] as to which algorithms that might best work. However, as with any machine learning problem the best algorithm that will work best with the dataset at hand might not at all be obvious at the start. This project will have no pre-conception as to which algorithm to run with and although it widens the scope a machine learning project needs to be open as to which algorithm to use. However, in the interest of keeping a narrow scope not all supervised learning algorithms will be used.

Being as loan defaults or charge offs does not happen very often (assuming prudent loan underwriting etc.) this dataset will be imbalanced as the percentage of defaults and charge offs will in the minority. The type of classifiers that can hand imbalanced datasets tend to be classifiers that can assign a class weight such as Decision Trees. Decision tree algorithms forces both classes to be addressed through the splitting rules that look at the class variables used in the creation of the trees[8].

---

[7] http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

[8] https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

As detailed in section I.IV Benchmark Model the following supervised algorithms will be applied to the final dataset:

- Ensembles methods
  - Averaging
  - Boosting
- Decision Trees method
- Nearest Neighbour methods

The various algorithms have been selected as they all provide probability estimates called 'predict_proba'. In one of Tom Fawcett's blogs[9] he recommended not to use hard classification measures such as score or predict, hence the choice of algorithms below.

Ensemble methods

Ensemble methods combine predictions from several base estimators built with a given learning algorithm in order to improve generalisation / robustness oner a single estimator[10]. These methods can be broken down into two categories; averaging and boosting methods.

Averaging methods builds on several estimators and extracts estimator's prediction, to then average all predictions into one overall prediction. The models that will be applied are the scikit BaggingClassifier, ExtraTreesClassifier and RandomForestClassifier.

Boosting methods works differently by creating base estimators sequentially and each step one tries to reduce the bias of the combined estimator. The models that will be applied are the scikit AdaBoostClassifier and GradientBoostingClassifier.

Decision Trees methods

Decision trees seek to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Whilst decision trees are simple one must take care not to extend the tree too far (or too deep) as this would cause it to overfit and unable to generalise on new data. The current dataset has 54 fields and there might be an issue with this amount of features causing the decision tree model to overfit. The feature engineering process might help this depending on the results from the feature engineering process.

The model that will be applied is the scikit DecisionTreeClassifier

Nearest Neighbour methods

Unlike the other algorithms nearest neighbour methods locate a predefined number of training samples closets in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbour learning) or vary based on the local density of points (radius-based neighbour learning). Whilst the distance can by any metric in this project the Euclidean distance will be used.

The model that will be applied is the scikit KNeighborsClassifier.

II.IV. Benchmark

The process of benchmarking the various algorithm outputs will be based upon what was details in section I.V. Evaluation Metrics. Each algorithm listed in the above section will be evaluated according to their initial output and assessed accordingly. However, as the classes are unbalanced different metrics that are less sensitive to imbalance when evaluating the predictive performance of classifiers will be used.

The goal is to identify the borrowers of the positive class (the minority class) successfully. For this purpose precision and recall is best applied as evaluation tools as they both provide information about the quality of the classifier. Furthermore, these two measures can be combined into one by using the F1 score, being the harmonic mean of recall and precision

[9] https://www.svds.com/tbt-learning-imbalanced-classes/

[10] http://scikit-learn.org/stable/modules/ensemble.html

(F1 = 2* ((recall * precision) / (recall + precision)). The other benefit of the F1 score is that it favour classifiers that are strong in both precision and recall.

In addition to the F1 score the AUC values for each algorithm will be evaluated as this measure better reflects the predictive capabilities of an algorithm as this problem is dealing with an imbalanced class.

In addition to the two metrics above the model calibration will also be displayed against each other and against the perfect calibrated line. Any divergence from this line indicates biased probabilities.

The initial run will form the basis for which algorithm selection and future performance tuning will be based upon. Only one algorithm will be selected for performance tuning.

Benchmark process:
1. Base algorithm run => Evaluation Metrics
2. Algorithm Ranking and Selection
3. Top algorithm tuning => Evaluation Metrics

III. Methodology

In this section I will document the results from the data processing.

III.I. Data Preprocessing

The data exploration section details the main data preprocessing taking the original dataset from 74 fields down to 25 fields then up to 54 fields which is used for the model runs.

It was originally proposed to apply oversampling techniques such as SMOTE algorithm to rebalance the minority dataset by creating new minority examples. However, this technique has its limitations as it operates by interpolating between rare examples and it can only generate examples within the body of available examples - never outside[15].

There are other suggested techniques for dealing with imbalanced datasets by applying a class weight to the minority cohort or features within each algorithm. Supervised learning classification algorithms containing class_weight as one of their inputs would be selected.

III.II. Implementation

The original proposal contained a number of metrics that were intended to be applied to each algorithm in order to measure its success. However, with further research into imbalanced datasets and how to deal with these datasets made a few of them redundant, i.e. accuracy. For example, the metric Accuracy is not a very useful measure for heavily imbalanced datasets. For a dataset with a default population of 5.3% would result in an incorrect prediction of default/non-default cases if one applied accuracy as a measure[11]. Instead it is recommended to use the Areas Under The ROC Curve[15] (AUC). The F1 score will also be recorded during the optimisation stage for the chosen algorithm. This made the model evaluation approach less onerous than originally planned. The choice of algorithms was guided by the "predict_proba" and the following Python code was run to extract this list (a complete list can be viewed in Appendix E):

```
from sklearn.utils.testing import all_estimators

estimators = all_estimators()

for name, class_ in estimators:
    if hasattr(class_, 'predict_proba'):
        print(name)
```

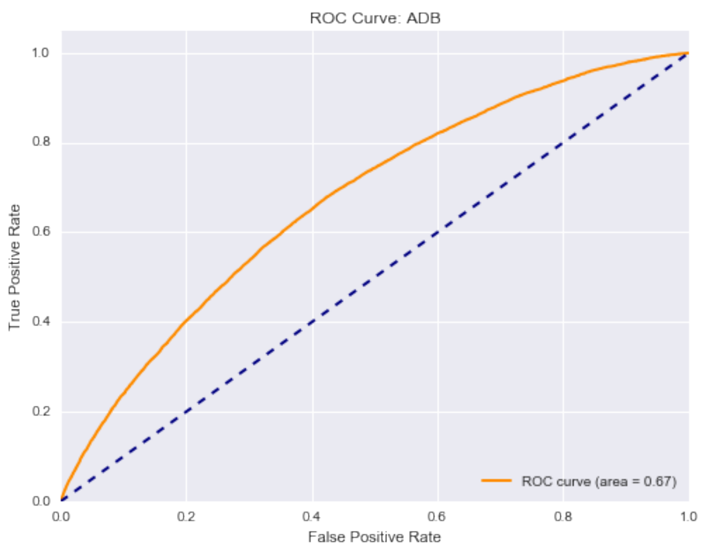
The following classifiers where applied to the dataset:

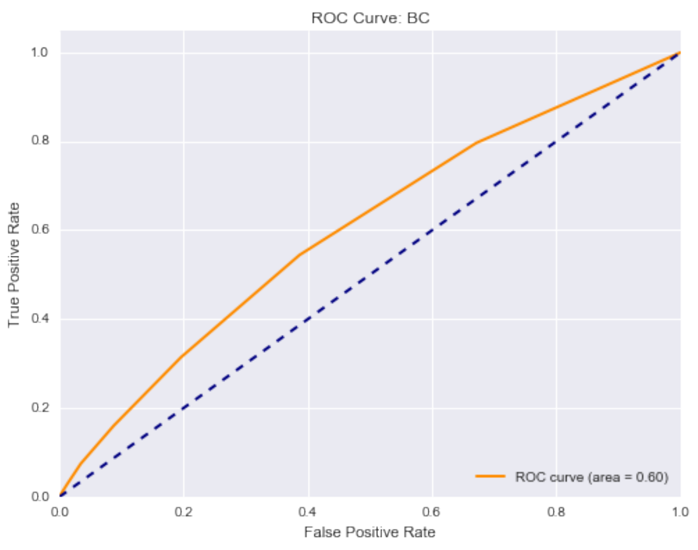
- AdaBoostClassifier
- BaggingClassifier
- DecisionTreeClassifier
- GradientBoostingClassifier
- RandomForestClassifier
- ExtraTreesClassifier
- KNeighborsClassifier

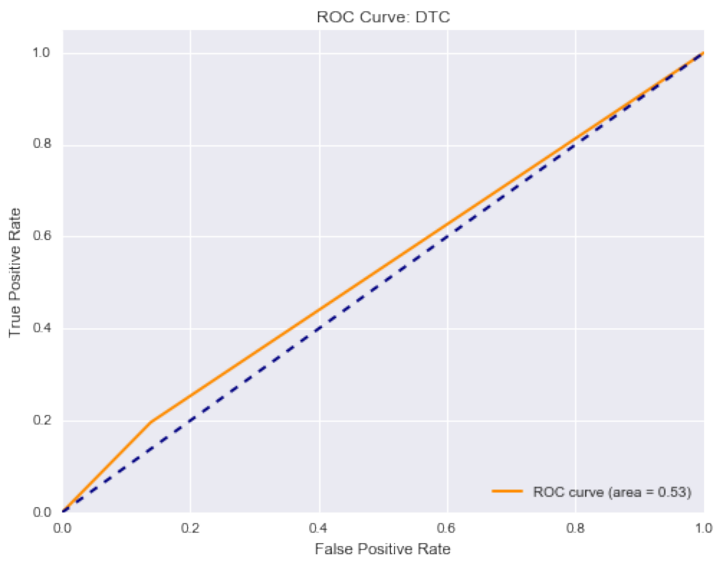
The above classifiers were added to a model list of classifiers. The list of models were then run through a loop to generate ROC charts for each algorithm and produced the following summary and charts:

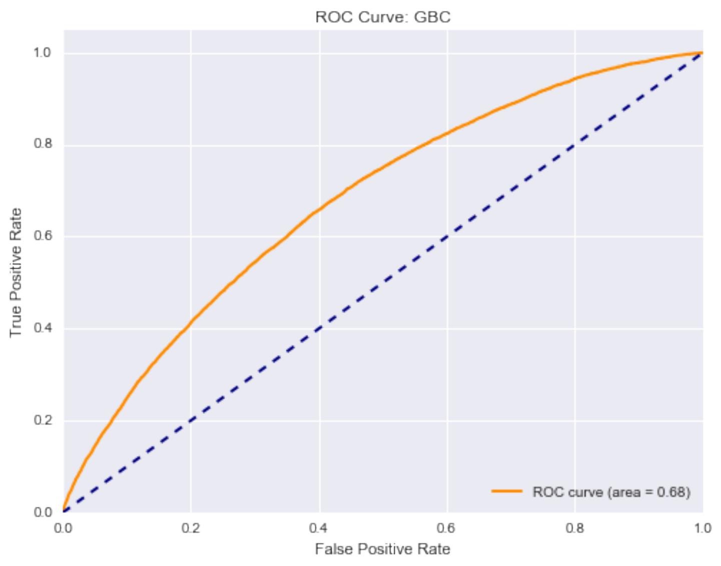
[11] https://www.svds.com/learning-imbalanced-classes/

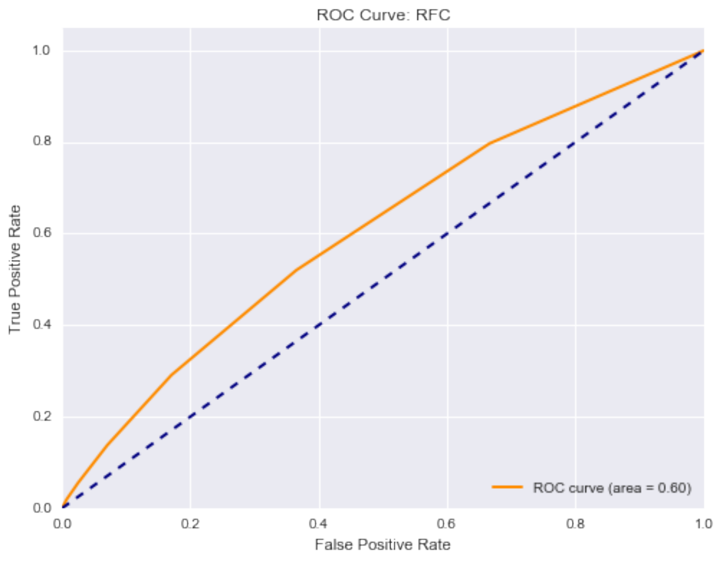
Classifiers	AUC %
AdaBoostClassifier	67%
BaggingClassifier	60%
DecisionTreeClassifier	53%
GradientBoostingClassifier	68%
RandomForestClassifier	60%
ExtraTreesClassifier	59%
KNeighborsClassifier	54%

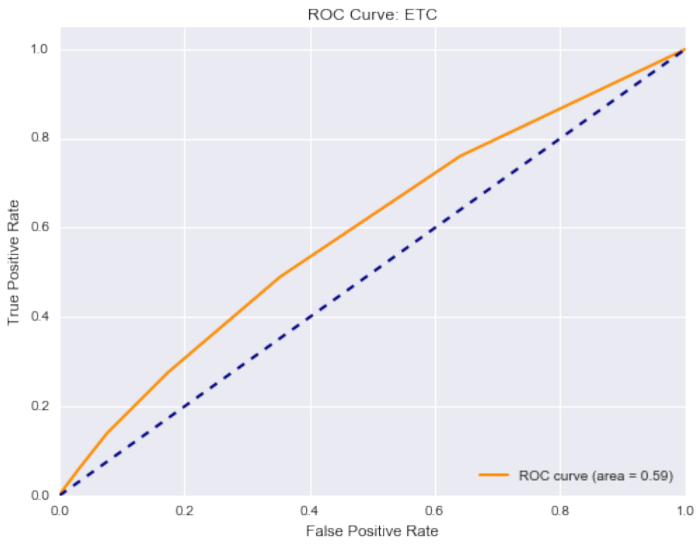


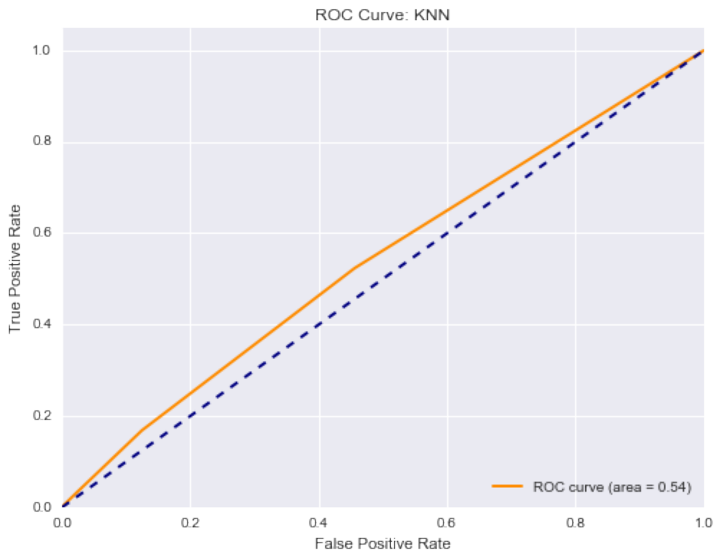












None of the above algorithms were tuned in any way and run with their standard inputs. From the above results the GradientBoostingClassifier (GBC) came out at the top.

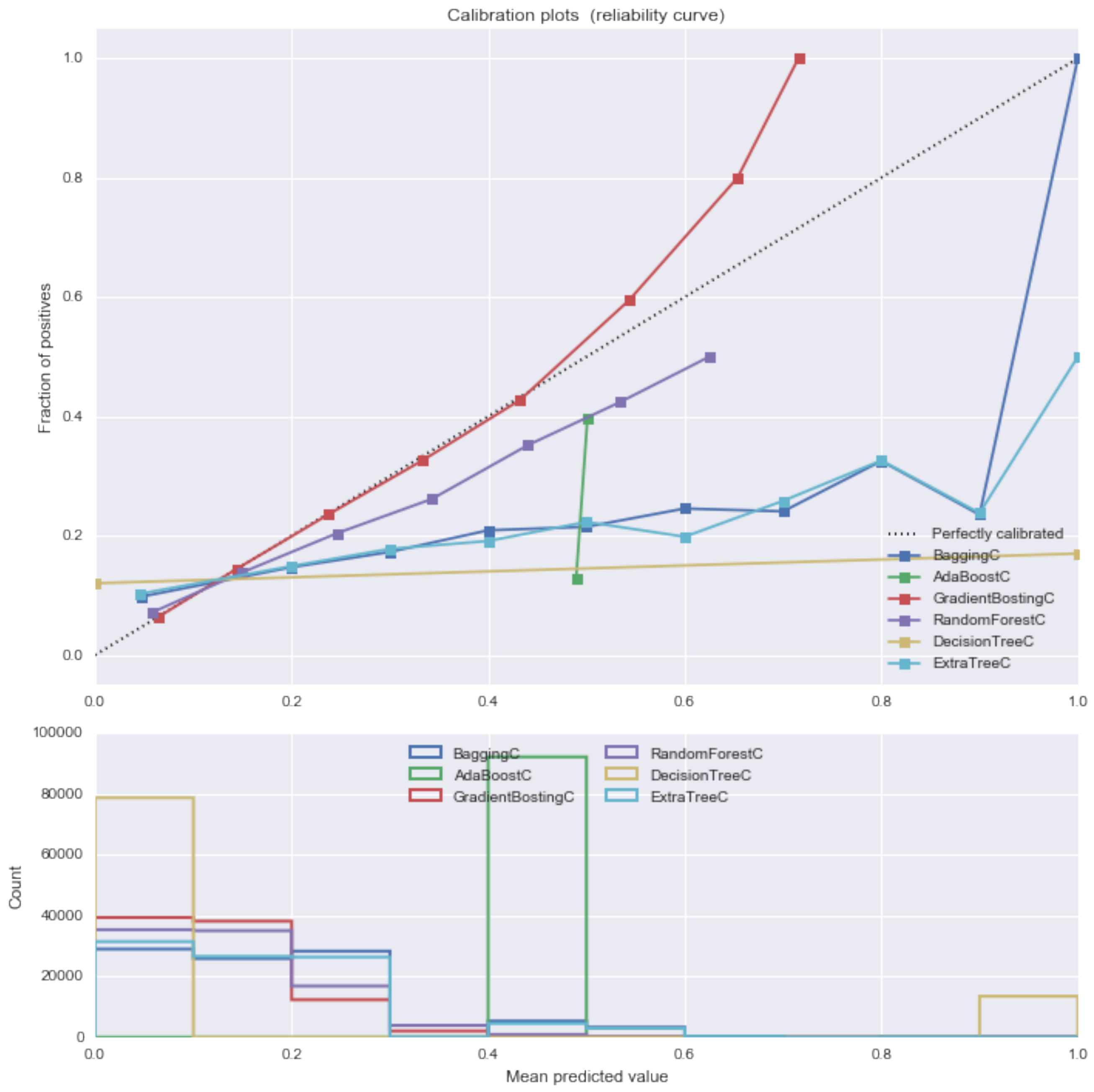
In addition to the AUC measure I have also a chart comparing the calibration of classifiers listed above[12]. Again the algorithms have not been tuned in any way and are the same as for the AUC charts above. The comparison of calibration of classifiers shows how well a classifier is calibrated. A well calibrated classifier are probabilistic classifier for which the output of the predict_proba method can be directly interpreted as a confidence level. The chart below summarises the current baseline models and any model that closely follows the perfectly calibrated line (dotted line) is a well calibrated model and one can have a high confidence in its predictive powers.

From the chart below it can be seen that the GBC initially closely follows the dotted line up a to about 45% mean predictive value were the line starts to move of upwards becoming slightly biased.

The other algorithms starts out close to the perfect calibrated line but then stays almost horizontal for the remaining values. RandomForestClassifier is the only other one that remotely stays close to the dotted line. these algorithm methods all return biased probabilities.

Through further tuning I hope to improve the classification capabilities of the GBC.

[12] http://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html#sphx-glr-auto-examples-calibration-plot-compare-calibration-py



I must add that the use of this method was not initially intended as I became aware of this later one after the proposal submission.

The GBC produces a prediction model in the form of an ensemble of weak prediction models, normally decision trees. It builds the model in a stage-wise fashion like other boosting methods, e.g. AdaBoost, and it generalises them by allowing optimisation of an arbitrary differentiable loss function[13]. The loss function optimisation seek to minimise the “cost” associated with the event.

The main difficulties (if one could classify it as so) was the time it took to run some of these models which limited the scope somewhat and reduced the amount of time available to fully explore all angles. However, in a commercial environment time is a luxury.

The implementation of the models were straight forward based on previous experience from Udacity and other available online resources (such as machinelearningmastery.com). However, one need to be aware of ones computer environment and dataset format. Whilst there were useful code snippets available online a majority had to be abandoned due to the

[13] https://en.wikipedia.org/wiki/Gradient_boosting

time it would have taken to get it to work and one had to weight up the aim of the project and focus on what was required. So a fair amount of time was also spent on trying different visualisation techniques of which a large portion did not work as intended.

III.III. Refinement

With the GBC algorithm chosen for tuning there are a number of parameters that can be tuned. The type of parameters can be split into two sections[14]:

1. Tree specific parameters
2. Boosting specific parameters

Within the tree specific parameters the following will be optimised:

- **min_samples_split** - defines the minimum number of samples required in a node to be considered for splitting. Also used to control over-fitting. Ideally tuned using cross-validation to ensure the appropriate value is identified.
- **min_samples_leaf** - defines the minimum samples required in a leaf and would generally be lower values for imbalanced datasets as the region of the minority class is very small.
- **max_depth** - controls the maximum depth of a tree and helps to control over-fitting as deeper trees will learn particular
- **max_features** - at the number of features to consider while searching for a best split.

For Boosting specific parameters the following will be optimised:

- **learning_rate** - this determines the impact of each tree on the final outcome. The GBC works by starting with an initial estimate which is updated using the output of each tree. the learning rate controls the scale of this change in the estimates. Lower rates do make a difference making it more robust to the specific characteristics of the tree and thus allowing it to generalise better. However, over values also comes at a higher computational cost.
- **n_estimators** - the number of sequential trees to be modelled. Using cross validation during the tuning process should avoid any overfitting.
- **subsample** - number of fractions of observations to be selected for each tree.

There are other parameters which affect the overall functionality of the GBC algorithm, such as:

1. loss
2. init
3. random_state
4. verbose
5. warm_start
6. presort

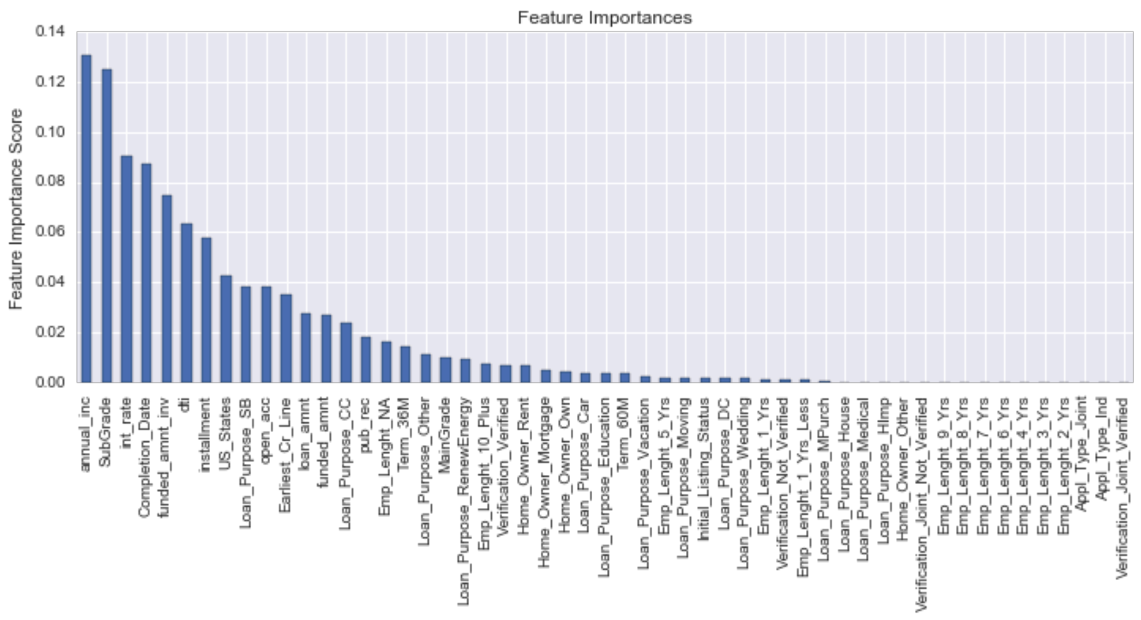
From the above list only random_state is used to ensure results can be reproduced.

For each of the above parameters cross-validation will be used to source the best parameters.

Before I start the parameters tuning I run the GBC through pre-defined function (called modelfit) which generates predictions based on the training data, cross-validate and return the F1 score and AUC value. The baseline return the following values and features:

Model Report:
F1 Score : 0.8135
AUC Score (Train): 0.696209
CV Score : Mean - 0.686244 | Std - 0.001711229 | Min - 0.6838335 | Max - 0.6885803

[14] https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/



The baseline model will go through a series of optimisation steps detailed as follows:

1. Optimise number of estimators for a relative high learning rate (0.1)
2. Having updated the number of estimators with the optimised value from above the max_depth and min_samples_split are jointly optimised again with the learning rate unchanged.
3. The min_samples_split might need further optimisation work depending on the second step and in this optimisation step both min_samples_split and min_samples_leaf are optimised together.
4. Of the final tree parameters the max_features will be optimised after updating the parameters in the third step.
5. At this point we can try to optimise the boosting parameters and before we delve into the learning rate optimisation the subsample will be optimised.
6. Finally the learning rate will be optimised through a series of steps reducing it from 0.1 down to 0.005 but also increasing the n_estimator towards the end to see whether this has in impact on the overall result.

IV. Results

In this section the various algorithm optimisation steps will be outlined in detailed and results documented. From the previous section the model that was chosen in the end was the GradientBoostingClassifier which produced the highest AUC using just the base algorithm without any algorithm tuning.

IV.I. Model Evaluation and Validation

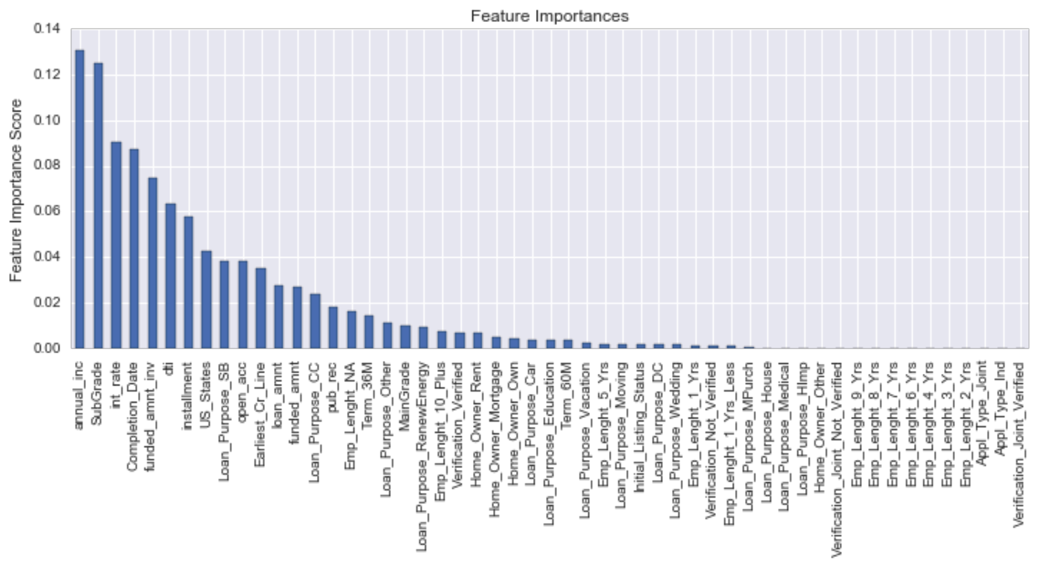
This section will document 10 model optimisations with the aim of improving the baseline model results. The table below summarises the various experiments carried out using cross-validation to achieve most of initial results.

Change Details	Data	Tree Parameters				Boosting Parameters			Other	Result		
		min_samples_split	min_samples_leaf	max_depth	max_features	learning_rate	n_estimators	subsample	random_seed	F1 Score	AUC	Mean CV Score
Baseline	Train	2	1	3	None	0.1	100	1.0	7	81.35%	69.62%	68.86%
n_estimators (20 to 80)	Train	2000	50	8	sqrt'	0.1	80	0.8	7			68.83%
n_estimators (80 to 120)	Train	2000	50	8	sqrt'	0.1	110	0.8	7			68.85%
max_depth (5 to 15) & min_sample_split (1800 to 2600)	Train	2600	50	9	sqrt'	0.1	110	0.8	7			68.86%
min_samples_split (2600 to 3200) & min_samples_leaf (30 to 70)	Train	3200	50	9	sqrt'	0.1	110	0.8	7	81.35%	72.29%	68.91%
max_features (7 to 19)	Train	3200	50	9	7	0.1	110	0.8	7			68.91%
subsample (0.6 to 0.9)	Train	3200	50	9	7	0.1	110	0.8	7			68.91%
learning_rate (0.05)	Train	3200	50	9	7	0.05	110	0.8	7	81.27%	71.34%	68.82%
learning_rate (0.01)	Train	3200	50	9	7	0.01	110	0.8	7	81.29%	71.57%	68.98%
learning_rate (0.005)	Train	3200	50	9	7	0.005	1200	0.8	7	81.31%	71.86%	
learning_rate (0.005)	Test	3200	50	9	7	0.005	1500	0.8	7	81.36%	71.50%	67.89%

Red cells denotes a parameter optimised value.

The baseline model was arrived at using the evaluation metric AUC. Running the GBC model without any tuning gives a good starting point for any future optimisation. The following resulted:

Model Report
F1 Score : 0.8135
AUC Score (Train): 0.696209
CV Score : Mean - 0.686244 | Std - 0.001711229 | Min - 0.6838335 | Max - 0.6885803



By further tuning the GBC one should be able to improve the above numbers. Furthermore, GBC has a few parameters that can be optimised.

Before any parameter tuning can take place one need to start with a set of prefixed parameters. The parameters that will initially be fixed are:

- **min_samples_split = 2000**: the default value is 2, but a value between 0.5% and 1.0% of the total number of loans is selected instead. As this is an imbalanced dataset an amount in-between will be chosen.
- **min_samples_leaf = 50**: default is 1 but a higher number is probably more relevant.

- **max_depth = 8**: the default is 3, however, as there are 54 fields a higher value is needed.
- **max_features = 'sqrt'**: the default is None, however, not all features play an equal part and as this parameter will be tuned 'sqrt' was randomly chosen.
- **subsample = 0.8**: appear to be commonly used and this value will be tuned.

TEST 1

With the above parameters set we will explore what the optimal n-estimators value would be for this dataset. By applying a test values from 20 to 80 in 10 increments the following results is returned:

```
([mean: 0.68371, std: 0.00265, params: {'n_estimators': 20},
  mean: 0.68565, std: 0.00284, params: {'n_estimators': 30},
  mean: 0.68681, std: 0.00282, params: {'n_estimators': 40},
  mean: 0.68731, std: 0.00281, params: {'n_estimators': 50},
  mean: 0.68781, std: 0.00270, params: {'n_estimators': 60},
  mean: 0.68812, std: 0.00265, params: {'n_estimators': 70},
  mean: 0.68835, std: 0.00275, params: {'n_estimators': 80}],
 {'n_estimators': 80},
 0.68834932924859404)
```

TEST 2

The optimal n_estimators with the current learning rate is 80. However, as 80 is the max value in the range give are there any higher values that also work? extending the parameter range from 60 to 120 in 10 increments the following results returns:

```
[mean: 0.68781, std: 0.00270, params: {'n_estimators': 60},
 mean: 0.68812, std: 0.00265, params: {'n_estimators': 70},
 mean: 0.68835, std: 0.00275, params: {'n_estimators': 80},
 mean: 0.68834, std: 0.00284, params: {'n_estimators': 90},
 mean: 0.68842, std: 0.00279, params: {'n_estimators': 100},
 mean: 0.68848, std: 0.00291, params: {'n_estimators': 110},
 mean: 0.68843, std: 0.00272, params: {'n_estimators': 120}],
{'n_estimators': 110},
0.68848445287839277)
```

This time the optimal value caps out at 110 n_estimators. The mean CV score has improved slightly from 68.83% to 68.84%.

TEST 3

Before running the next set of parameter tuning the n_estimator is updated with the new value of 110. The next set of parameters to be tested are max_depth (5 to 15 in 2 increments) and min_samples_split (1800 to 2600 in 200 increments).

```
 mean: 0.68551, std: 0.00346, params: {'min_samples_split': 1800, 'max_depth': 15},
 mean: 0.68534, std: 0.00278, params: {'min_samples_split': 2000, 'max_depth': 15},
 mean: 0.68610, std: 0.00281, params: {'min_samples_split': 2200, 'max_depth': 15},
 mean: 0.68621, std: 0.00371, params: {'min_samples_split': 2400, 'max_depth': 15},
 mean: 0.68643, std: 0.00334, params: {'min_samples_split': 2600, 'max_depth': 15}],
{'max_depth': 9, 'min_samples_split': 2600},
0.68861028864728824)
```

Running the above tuning values results in a max_depth of 9 and min_samples_split of 2600. As with the n_estimators the min_samples_split ended up at the max value in the range and there possible room for an upward move. The mean CV value improved slightly again to 68.86%.

TEST 4

Updating the max_depth with the new value there is still room for change for the min_sample_split (2600 to 3200 in 200 increments). min_samples_leaf will also be optimised (30 to 70 in increments of 10).

```
Model Report:
F1 Score : 0.8135
AUC Score (Train): 0.722885
CV Score : Mean - 0.6891309 | Std - 0.003110668 | Min - 0.6852115 | Max - 0.6927422
```

```
 mean: 0.68816, std: 0.00253, params: {'min_samples_split': 3200, 'min_samples_leaf': 60},
 mean: 0.68787, std: 0.00200, params: {'min_samples_split': 2600, 'min_samples_leaf': 70},
 mean: 0.68768, std: 0.00334, params: {'min_samples_split': 2800, 'min_samples_leaf': 70},
 mean: 0.68803, std: 0.00318, params: {'min_samples_split': 3000, 'min_samples_leaf': 70},
 mean: 0.68844, std: 0.00228, params: {'min_samples_split': 3200, 'min_samples_leaf': 70}],
{'min_samples_leaf': 50, 'min_samples_split': 3200},
0.68913086655154743)
```

min_sample_leaf remains at 50 and min_samples_split increase to 3200. The mean CV value increase to 68.91%.

TEST 5
After updating the min_sample_split value (min_sample_leaf remains the same) the next parameter is the max_features (7 to 19 in increments of 2)

```
[mean: 0.68913, std: 0.00311, params: {'max_features': 7},
 mean: 0.68856, std: 0.00324, params: {'max_features': 9},
 mean: 0.68856, std: 0.00266, params: {'max_features': 11},
 mean: 0.68860, std: 0.00288, params: {'max_features': 13},
 mean: 0.68844, std: 0.00312, params: {'max_features': 15},
 mean: 0.68810, std: 0.00248, params: {'max_features': 17},
 mean: 0.68808, std: 0.00268, params: {'max_features': 19}],
{'max_features': 7},
0.68913086655154743)
```

The max_features value comes out at 7 with the mean Cv value unchanged at 68.91%.

TEST 6
The second last parameter to tune is the subsample with a value range from 0.6 to 0.9. With the other parameters updated the following resulted:

```
([mean: 0.68783, std: 0.00321, params: {'subsample': 0.6},
  mean: 0.68791, std: 0.00263, params: {'subsample': 0.7},
  mean: 0.68831, std: 0.00289, params: {'subsample': 0.75},
  mean: 0.68913, std: 0.00311, params: {'subsample': 0.8},
  mean: 0.68825, std: 0.00261, params: {'subsample': 0.85},
  mean: 0.68818, std: 0.00292, params: {'subsample': 0.9}],
 {'subsample': 0.8},
 0.68913086655154743)
```

The subsample is unchanged at 0.8 and hence the unchanged mean CV.

TEST 7
The final parameter learning_rate and at first the learning_rate will be reduced to 0.05. No grid search is performed during this process as it will start to take longer to run as we start changing the tree sizes as well. However, running the updated learning_rate results in the following:

Model Report:
F1 Score : 0.8127
AUC Score (Train): 0.713407
CV Score : Mean - 0.6882106 I Std - 0.002401509 I Min - 0.6853011 I Max - 0.6908346

This resulted in a slight step backwards in mean CV score as did the AUC percentage.

TEST 8
To verify carry on testing whether lower learning_rates makes it worse a value of 0.01 is added and the following results are extracted:

Model Report:
F1 Score : 0.8129
AUC Score (Train): 0.715691
CV Score : Mean - 0.6897596 I Std - 0.002840271 I Min - 0.6860276 I Max - 0.6929949

The mean CV score improves as does the F1 and AUC values.

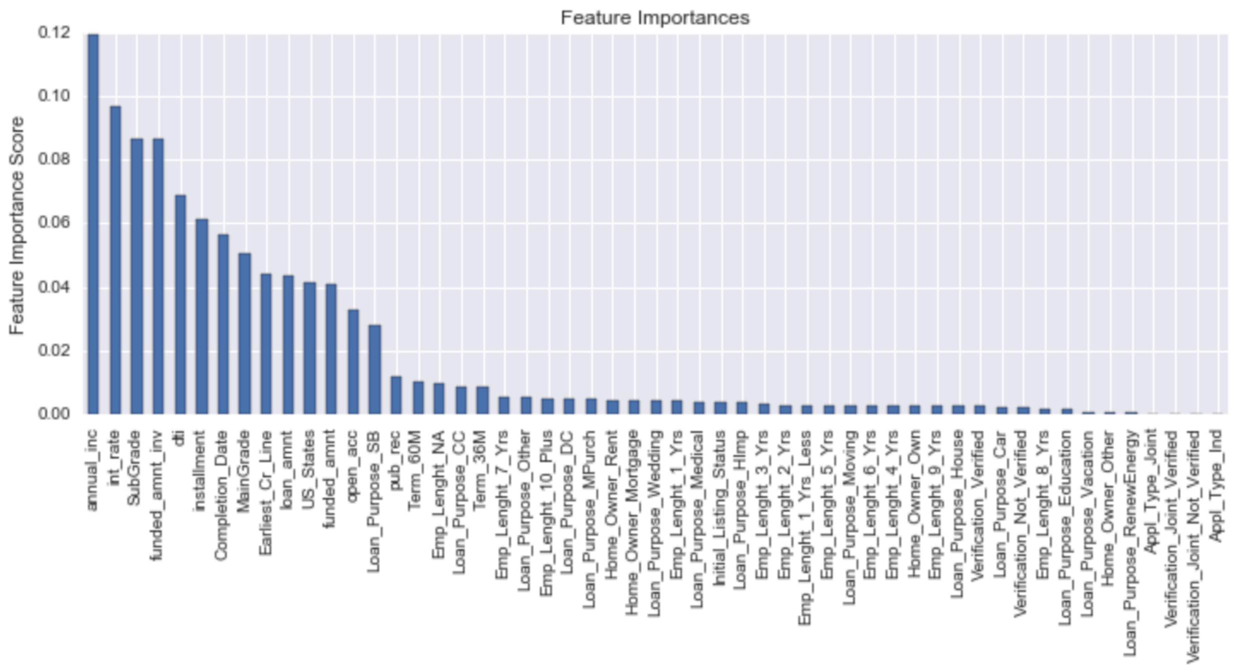
TEST 9
By halving the Test 8 learning_rate to 0.005 and significantly increasing the n_estimators to 1200 the following values are extracted:

Model Report
F1 Score : 0.8131
AUC Score (Train): 0.718640

The above results drops slightly compared to the Test 8 results.

TEST 10
The final test is to run the updated parameters against the test dataset to see if there is change in the metrics.

```
Model Report
F1 Score : 0.8136
AUC Score (Train): 0.715030
CV Score : Mean - 0.6788974 | Std - 0.002184295 | Min - 0.676464 | Max - 0.6823052
```



Testing against the test dataset results in a slight drop in AUC and mean CV score but an increase in the F1 score.

With regards to smaller changes in the dataset and its potential impact on the result I expect the impact to be fairly low due to the number of data points already trained on.

From the above test results I decided in the end to give SMOTE over-sampling a test run to see if it made any difference as I wasn't too confident that the model was predicting properly, i.e. predicting just non-defaulted borrowers. My suspicion was correct as shown in the below classification report:

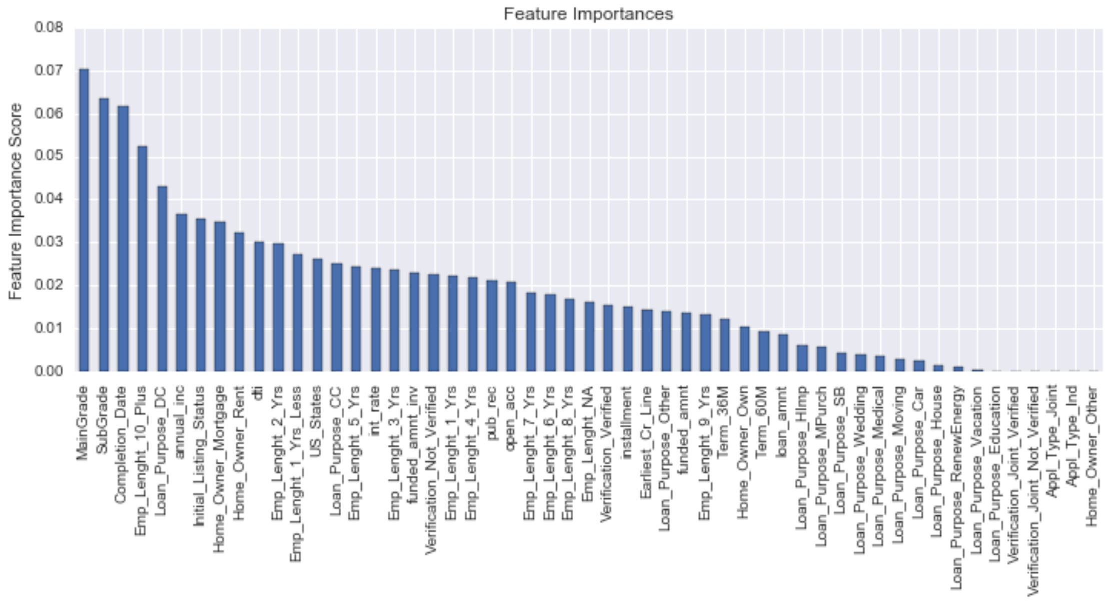
```
             precision    recall  f1-score   support

          0       0.87      1.00      0.93     80478
          1       0.38      0.00      0.00     11774

avg / total       0.81      0.87      0.81     92252
```

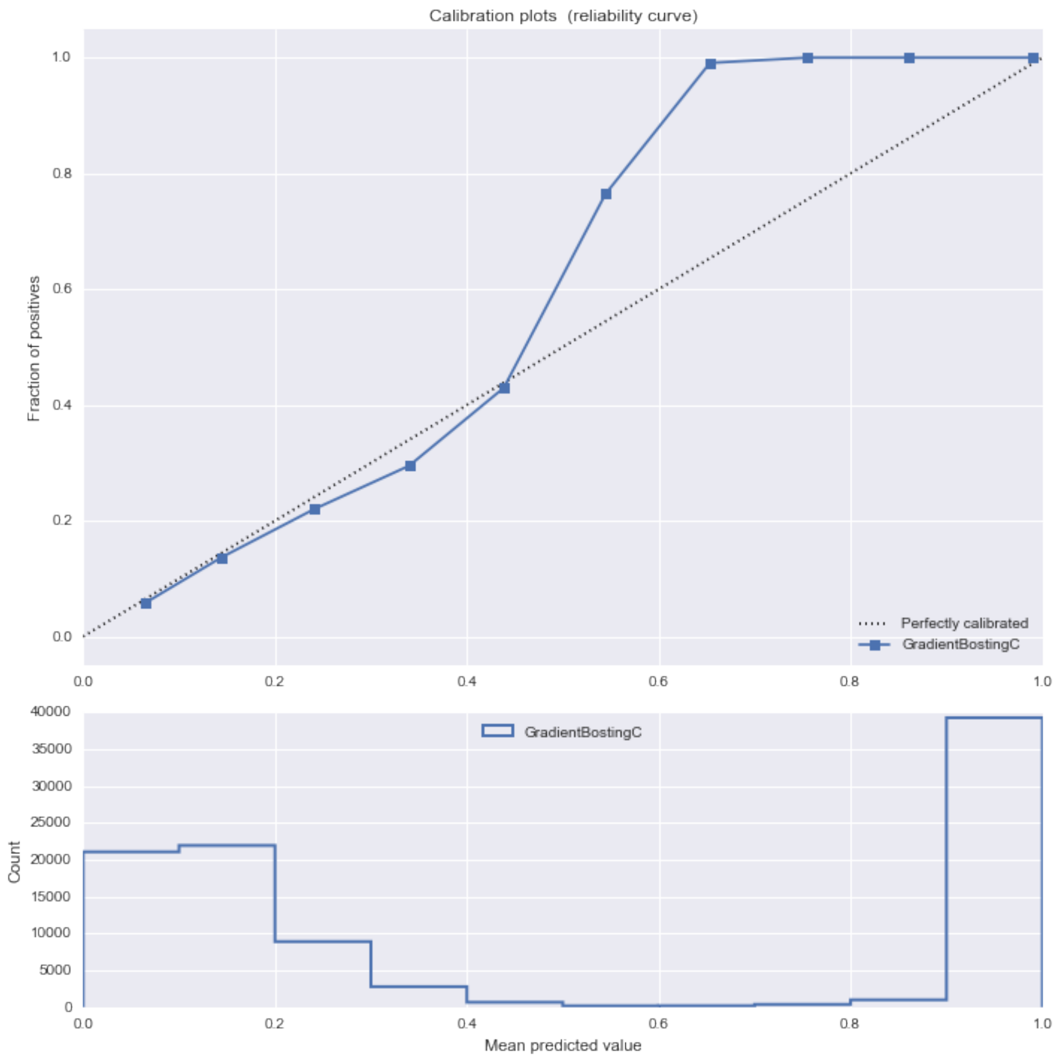
After applying the SMOTE method the following results are extracted:

```
Model Report
F1 Score : 0.9234
AUC Score (Train): 0.958210
CV Score : Mean - 0.9521322 | Std - 0.001145379 | Min - 0.9505584 | Max - 0.9535969
```



After applying the SMOTE over-sampling method the model shows an greatly improved ability to predict. This indicates the robustness of the model as the calibration plot shows below there are further room for improvement to deal with the over calibration past 50% of mean predictive value. However, the level achieved is much better than previously modelled with an AUC score of 95% which would be considered very good[15].

[15] https://datamize.wordpress.com/2015/01/24/how-to-plot-a-roc-curve-in-scikit-learn/



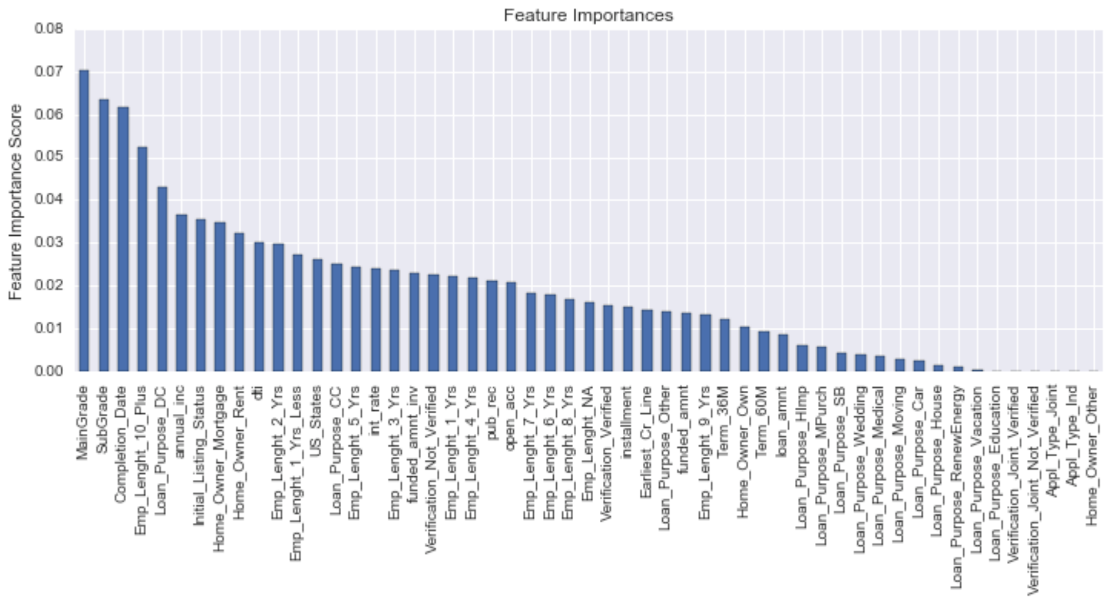
IV.II. Justification

The parameter values that produced the highest values combined where the following:

- min_samples_split = 3200
- min_samples_leaf = 50
- max_depth = 9
- max_features = 7
- learning_rate = 0.1
- n_estimators = 110
- subsample = 0.8

By including the SMOTE over-sampling method and the above parameters the following outputs are extracted:

```
Model Report
F1 Score : 0.9234
AUC Score (Train): 0.958210
CV Score : Mean - 0.9521322 | Std - 0.001145379 | Min - 0.9505584 | Max - 0.9535969
```



Classification Report:

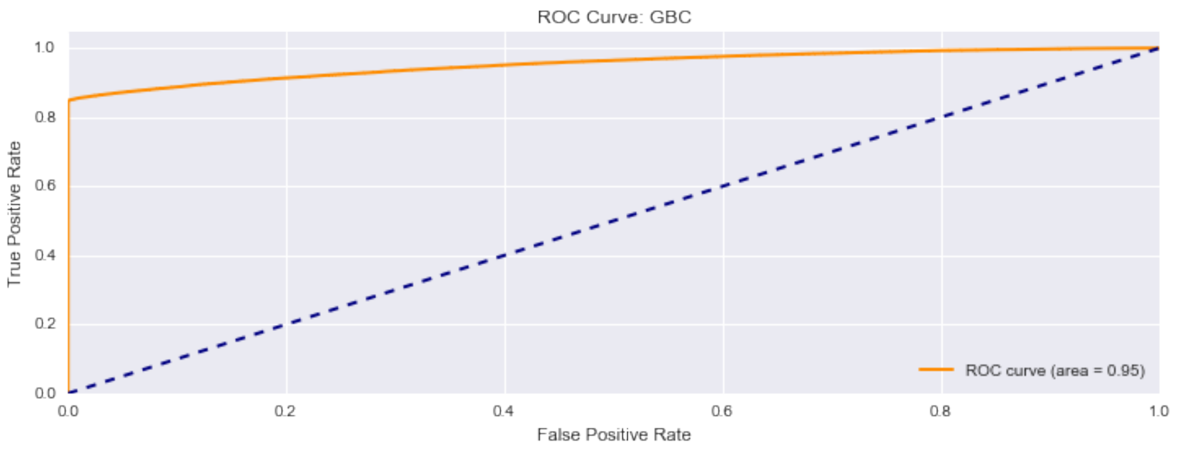
```
             precision    recall  f1-score   support

          0       0.87      1.00      0.93     48164
          1       1.00      0.85      0.92     48380

avg / total       0.93      0.92      0.92     96544
```

Compared to the benchmark output it’s certainly better in terms of AUC and mean CV score. The F1 score considerably as well. However, at the end of the day does the above result make a difference compared to other traditional risk methods? With an AUC value of above 95% it’s much better than luck, however, including such a model within a pricing framework might require further data and testing. Using the above model to manage customers throughout their loan term might be a better platform to fully test this model and is predictability allowing a business to predict customers likely to default and monitor those customers a bit closer.

ROC Curve:

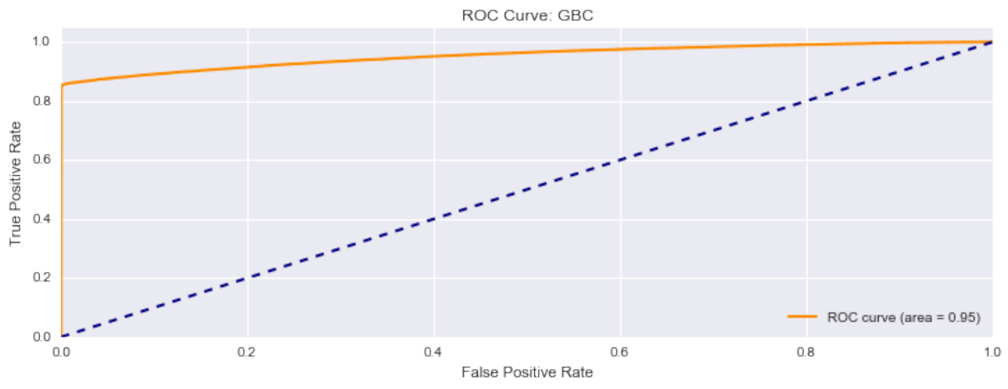


V. Conclusion

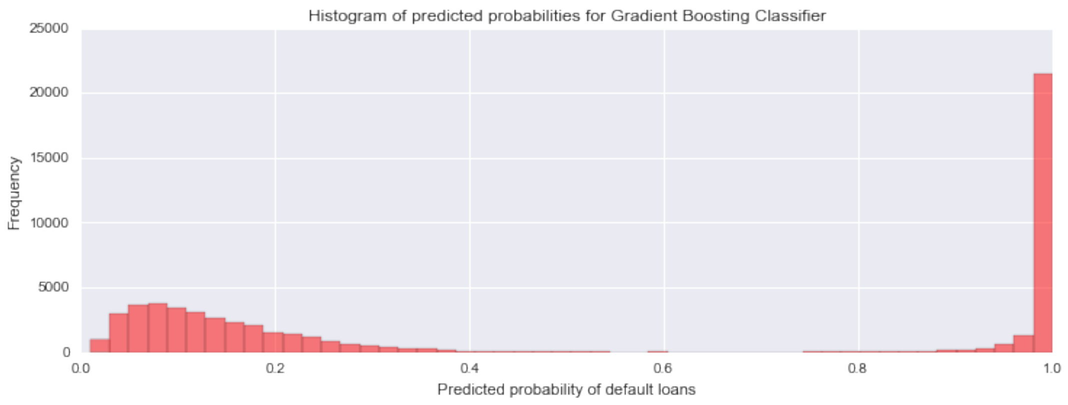
This was a challenging project with regards to dataset size, i.e. number of rows and columns, and trying not to be too over ambitious in terms of algorithms to apply. Although I expected to complete this by end of September 2017 due to personal circumstances I did not have sufficient time to focus on the completion of this project. However, the time in-between submitting the proposal and re-starting this project has not been waisted. Spending time to consider the data exploration and pre-processing is vital in any machine learning project, as is the choice of metrics applied for algorithm evaluation.

V.I. Free-Form Visualisation

Below shows the ROC curve for the Gradient Boosting Classifier (GBC). This plot gives a reasonable insight into the performance of a GBC. Beyond the overall performance measured by AUC, it shows the tradeoff between False Positive and True Positive rate when moving the threshold of probability obtained from the classifier. Since the objective requires a greater focus on borrower likely to default, we can decide to select a higher threshold and improve the True Positive Rate at the expense of higher False Positive, the curve makes this decision easier to assess.



By combining the ROC curve analysis with a histogram of predicted probabilities to understand where threshold will have most variations. For example, we can see that there are a number of borrowers labelled with around 90% probability of default. Moving the threshold from 40% to 90% is unlikely to have any significant impact, however, pushing it further between 90% to 100% will ensure a higher True Positive Rate and lower risk significantly.



Looking at the calibration chart below indicates a well calibrated model up to 50%. The GBC then starts to over calibrate slightly.

V.II. Reflection

A considerable time has been spent thinking about how to deal with the Lending Club dataset and carrying out those steps. From having done this Capstone project and with imbalanced datasets being the norm I feel that this is an area where the Machine Learning Nanodegree could expand upon. I could have chosen a simpler dataset but with my background in financial services I felt I needed to apply my domain knowledge and work on a dataset that hasn't been augmented or balanced in any way.

Certain algorithms like Support Vector Machine (SCV algorithm) and Gaussian Process Classification caused the Jupyter Notebook to fall over and as a result I had to abandon those algorithms. Whether this was due to the computer hardware or trying to run too many algorithms is uncertain at this point in time. Similarly when trying to find an optimal random_state the computer hardware took a considerable while to compute and was abandoned in the end.

The main aspects that I have take away from this Capstone project is the importance of data exploration, preprocessing and dealing with imbalanced datasets. Ignoring these steps in any machine learning project could have serious consequences. For example, pre-screening for certain cancers in the population and not having a proper calibrated machine learning model could end up cause considerable misdiagnosis (False Positives) and mental stress for the relevant patient.

V.III. Improvement

Further knowledge and experience in dealing with imbalanced dataset would have helped considerably. Whilst I did not initially take advantage of the SMOTE over-sampling method it became apparent not doing so would have left a model not able to identify defaulted borrowers and hence adding it in towards the end of the tuning process. Furthermore the ROC curves are producing better results as can be shown in the calibration chart above.

A variety of algorithms are available for use within the supervised learning field. However, the scope of this project had to be kept manageable and hence the low number of algorithms applied. Further improvements could be achieved by applying a few more algorithms.

Whilst the dataset features were fixed, had one had access to the Lending Club database there would have been a potential gain in algorithm performance by adding further personalised data not available in the current dataset. Data such as credit history received from the credit search agencies and other personalised data (mandatory data) could improve the predictive power of an algorithm.

The GBC algorithm chosen indicates there are possible improvements that can be made beyond what was executed in this project. The over calibration in the calibration chart suggests there are further changes to be made either through better data or parameter changes.

VI. References

A. Davidson and A. Levin, Mortgage Valuation Models Embedded Options, Risk and Uncertainty. Oxford University Press, 2014.

T. M. Mitchell, Machine Learning. McGraw Hill, International Edition 1997.

Appendix A: Initial Dataset

Dataset Features:

field_name	record_count	null_count	null_record_ratio	empty_string_count	distinct_count
id	887379	0	0.00%	0	887379
member_id	887379	0	0.00%	0	887379
loan_amnt	887379	0	0.00%	0	1372
funded_amnt	887379	0	0.00%	0	1372
funded_amnt_inv	887379	0	0.00%	0	9856
term	887379	0	0.00%	0	2
int_rate	887379	0	0.00%	0	542
installment	887379	0	0.00%	0	68711
grade	887379	0	0.00%	0	7
sub_grade	887379	0	0.00%	0	35
emp_length	887379	0	0.00%	0	12
home_ownership	887379	0	0.00%	0	6
annual_inc	887379	4	0.00%	0	49385
verification_status	887379	0	0.00%	0	3
issue_d	887379	0	0.00%	0	103
loan_status	887379	0	0.00%	0	10
purpose	887379	0	0.00%	0	14
addr_state	887379	0	0.00%	0	51
dti	887379	0	0.00%	0	4086
earliest_cr_line	887379	29	0.00%	0	698
open_acc	887379	29	0.00%	0	78
pub_rec	887379	29	0.00%	0	33
initial_list_status	887379	0	0.00%	0	2
application_type	887379	0	0.00%	0	2
verification_status_joint	887379	886868	99.94%	0	4

Appendix B: Dataset Transformation

Dataset features:

field_name	record_count	null_count	null_record_ratio	empty_string_count	distinct_count
loan_amnt	887,379	-	0.00%	-	1,372
funded_amnt	887,379	-	0.00%	-	1,372
funded_amnt_inv	887,379	-	0.00%	-	9,856
int_rate	887,379	-	0.00%	-	542
installment	887,379	-	0.00%	-	68,711
annual_inc	887,379	4	0.00%	-	49,385
dti	887,379	-	0.00%	-	4,086
open_acc	887,379	29	0.00%	-	78
pub_rec	887,379	29	0.00%	-	33
Term_36M	887,379	-	0.00%	-	2
Term_60M	887,379	-	0.00%	-	2
MainGrade	887,379	-	0.00%	-	7
SubGrade	887,379	-	0.00%	-	35
Emp_Lenght_10_Plus	887,379	-	0.00%	-	2
Emp_Lenght_9_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_8_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_7_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_6_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_5_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_4_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_3_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_2_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_1_Yrs	887,379	-	0.00%	-	2
Emp_Lenght_1_Yrs_Less	887,379	-	0.00%	-	2
Emp_Lenght_NA	887,379	-	0.00%	-	2
Home_Owner_Rent	887,379	-	0.00%	-	2
Home_Owner_Own	887,379	-	0.00%	-	2
Home_Owner_Mortgage	887,379	-	0.00%	-	2
Home_Owner_Other	887,379	-	0.00%	-	2
Verification_Verified	887,379	-	0.00%	-	2
Verification_Not_Verified	887,379	-	0.00%	-	2
Appl_Type_Ind	887,379	-	0.00%	-	2
Appl_Type_Joint	887,379	-	0.00%	-	2
Verification_Joint_Verifiec	887,379	-	0.00%	-	2
Verification_Joint_Not_Ve	887,379	-	0.00%	-	2
Loan_Purpose_CC	887,379	-	0.00%	-	2
Loan_Purpose_Car	887,379	-	0.00%	-	2
Loan_Purpose_SB	887,379	-	0.00%	-	2
Loan_Purpose_Other	887,379	-	0.00%	-	2
Loan_Purpose_Wedding	887,379	-	0.00%	-	2
Loan_Purpose_DC	887,379	-	0.00%	-	2
Loan_Purpose_HImp	887,379	-	0.00%	-	2
Loan_Purpose_MPurch	887,379	-	0.00%	-	2
Loan_Purpose_Medical	887,379	-	0.00%	-	2
Loan_Purpose_Moving	887,379	-	0.00%	-	2
Loan_Purpose_Vacation	887,379	-	0.00%	-	2
Loan_Purpose_House	887,379	-	0.00%	-	2
Loan_Purpose_RenewEne	887,379	-	0.00%	-	2
Loan_Purpose_Education	887,379	-	0.00%	-	2
Completion_Date	887,379	-	0.00%	-	9
Earliest_Cr_Line	887,379	-	0.00%	-	67
US_States	887,379	-	0.00%	-	51
Initial_Listing_Status	887,379	-	0.00%	-	2
Loan_Status	887,379	-	0.00%	-	2

```
('Subset_2 data shape: ', (887379, 54))
```

Appendix C: Dataset Transformation Removing NAN

Dataset features:

field_name	record_count	null_count	null_record_ratio	empty_string_count	distinct_count
loan_amnt	887,350	-	0.0%	-	1,372
funded_amnt	887,350	-	0.0%	-	1,372
funded_amnt_inv	887,350	-	0.0%	-	9,856
int_rate	887,350	-	0.0%	-	542
installment	887,350	-	0.0%	-	68,704
annual_inc	887,350	-	0.0%	-	49,384
dti	887,350	-	0.0%	-	4,086
open_acc	887,350	-	0.0%	-	77
pub_rec	887,350	-	0.0%	-	32
Term_36M	887,350	-	0.0%	-	2
Term_60M	887,350	-	0.0%	-	2
MainGrade	887,350	-	0.0%	-	7
SubGrade	887,350	-	0.0%	-	35
Emp_Lenght_10_Plus	887,350	-	0.0%	-	2
Emp_Lenght_9_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_8_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_7_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_6_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_5_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_4_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_3_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_2_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_1_Yrs	887,350	-	0.0%	-	2
Emp_Lenght_1_Yrs_Less	887,350	-	0.0%	-	2
Emp_Lenght_NA	887,350	-	0.0%	-	2
Home_Owner_Rent	887,350	-	0.0%	-	2
Home_Owner_Own	887,350	-	0.0%	-	2
Home_Owner_Mortgage	887,350	-	0.0%	-	2
Home_Owner_Other	887,350	-	0.0%	-	2
Verification_Verified	887,350	-	0.0%	-	2
Verification_Not_Verified	887,350	-	0.0%	-	2
Appl_Type_Ind	887,350	-	0.0%	-	2
Appl_Type_Joint	887,350	-	0.0%	-	2
Verification_Joint_Verified	887,350	-	0.0%	-	2
Verification_Joint_Not_Ver	887,350	-	0.0%	-	2
Loan_Purpose_CC	887,350	-	0.0%	-	2
Loan_Purpose_Car	887,350	-	0.0%	-	2
Loan_Purpose_SB	887,350	-	0.0%	-	2
Loan_Purpose_Other	887,350	-	0.0%	-	2
Loan_Purpose_Wedding	887,350	-	0.0%	-	2
Loan_Purpose_DC	887,350	-	0.0%	-	2
Loan_Purpose_HImp	887,350	-	0.0%	-	2
Loan_Purpose_MPurch	887,350	-	0.0%	-	2
Loan_Purpose_Medical	887,350	-	0.0%	-	2
Loan_Purpose_Moving	887,350	-	0.0%	-	2
Loan_Purpose_Vacation	887,350	-	0.0%	-	2
Loan_Purpose_House	887,350	-	0.0%	-	2
Loan_Purpose_RenewEner	887,350	-	0.0%	-	2
Loan_Purpose_Education	887,350	-	0.0%	-	2
Completion_Date	887,350	-	0.0%	-	9
Earliest_Cr_Line	887,350	-	0.0%	-	67
US_States	887,350	-	0.0%	-	51
Initial_Listing_Status	887,350	-	0.0%	-	2
Loan_Status	887,350	-	0.0%	-	2

('Subset_3 data shape: ', (887350, 54))

Appendix D: Updated Dataset Excluding the 2015 Cohort

Dataset features:

field_name	record_count	null_count	null_record_ratio	empty_string_count	distinct_count
loan_amnt	230,628	-	0%	-	1,278
funded_amnt	230,628	-	0%	-	1,285
funded_amnt_inv	230,628	-	0%	-	9,829
int_rate	230,628	-	0%	-	477
installment	230,628	-	0%	-	39,211
annual_inc	230,628	-	0%	-	18,858
dti	230,628	-	0%	-	3,499
open_acc	230,628	-	0%	-	55
pub_rec	230,628	-	0%	-	14
Term_36M	230,628	-	0%	-	2
Term_60M	230,628	-	0%	-	2
MainGrade	230,628	-	0%	-	7
SubGrade	230,628	-	0%	-	35
Emp_Lenght_10_Plus	230,628	-	0%	-	2
Emp_Lenght_9_Yrs	230,628	-	0%	-	2
Emp_Lenght_8_Yrs	230,628	-	0%	-	2
Emp_Lenght_7_Yrs	230,628	-	0%	-	2
Emp_Lenght_6_Yrs	230,628	-	0%	-	2
Emp_Lenght_5_Yrs	230,628	-	0%	-	2
Emp_Lenght_4_Yrs	230,628	-	0%	-	2
Emp_Lenght_3_Yrs	230,628	-	0%	-	2
Emp_Lenght_2_Yrs	230,628	-	0%	-	2
Emp_Lenght_1_Yrs	230,628	-	0%	-	2
Emp_Lenght_1_Yrs_Less	230,628	-	0%	-	2
Emp_Lenght_NA	230,628	-	0%	-	2
Home_Owner_Rent	230,628	-	0%	-	2
Home_Owner_Own	230,628	-	0%	-	2
Home_Owner_Mortgage	230,628	-	0%	-	2
Home_Owner_Other	230,628	-	0%	-	2
Verification_Verified	230,628	-	0%	-	2
Verification_Not_Verified	230,628	-	0%	-	2
Appl_Type_Ind	230,628	-	0%	-	1
Appl_Type_Joint	230,628	-	0%	-	1
Verification_Joint_Verified	230,628	-	0%	-	1
Verification_Joint_Not_Ve	230,628	-	0%	-	1
Loan_Purpose_CC	230,628	-	0%	-	2
Loan_Purpose_Car	230,628	-	0%	-	2
Loan_Purpose_SB	230,628	-	0%	-	2
Loan_Purpose_Other	230,628	-	0%	-	2
Loan_Purpose_Wedding	230,628	-	0%	-	2
Loan_Purpose_DC	230,628	-	0%	-	2
Loan_Purpose_HImp	230,628	-	0%	-	2
Loan_Purpose_MPurch	230,628	-	0%	-	2
Loan_Purpose_Medical	230,628	-	0%	-	2
Loan_Purpose_Moving	230,628	-	0%	-	2
Loan_Purpose_Vacation	230,628	-	0%	-	2
Loan_Purpose_House	230,628	-	0%	-	2
Loan_Purpose_RenewEner	230,628	-	0%	-	2
Loan_Purpose_Education	230,628	-	0%	-	2
Completion_Date	230,628	-	0%	-	7
Earliest_Cr_Line	230,628	-	0%	-	61
US_States	230,628	-	0%	-	50
Initial_Listing_Status	230,628	-	0%	-	2
Loan_Status	230,628	-	0%	-	2

```
('Subset_4 data shape: ', (230628, 54))
```

Appendix E: List of Algorithms Enabling predict_proba

- AdaBoostClassifier
- BaggingClassifier
- BayesianGaussianMixture
- BernoulliNB
- CalibratedClassifierCV
- DPGMM
- DecisionTreeClassifier
- ExtraTreeClassifier
- ExtraTreesClassifier
- GMM
- GaussianMixture
- GaussianNB
- GaussianProcessClassifier
- GradientBoostingClassifier
- KNeighborsClassifier
- LDA
- LabelPropagation
- LabelSpreading
- LinearDiscriminantAnalysis
- LogisticRegression
- LogisticRegressionCV
- MLPClassifier
- MultinomialNB
- NuSVC
- QDA
- QuadraticDiscriminantAnalysis
- RandomForestClassifier
- SGDClassifier
- SVC
- VBGMM
- _BinaryGaussianProcessClassifierLaplace
- _ConstantPredictor
- _DPGMMBase
- _GMMBase
- _LDA
- _QDA