# Credit Risk Modelling

## 1 Introduction

This study will focus on the use of supervised machine learning techniques to predict the likelihood of an individual defaulting on their loan. Data used is from the '*All Lending Club* loan data-set', which contains information about 8929 parties, characteristics about their personal and work life as well as whether their previous loan was 'cleared off' (defaulted). The modelling techniques used will be Logistic Regression, K-Nearest Neighbours (KNN) and Support Vector Machines (both linear and with kernels). This will allow for a wide range of results, both from simpler machine learning models to very sophisticated and computationally expensive ones. We expect the more complex models to perform in a superior way. The report will discuss data preparation, methodologies used and results obtained.

## 2 Data Preparation and Cleaning

### 2.1 Re-sampling

The data-set contains information from 8929 individuals who were issued a loan, which is reduced to 6802 after removing null observations. The dependent variable "*charged off*" shows whether an individual defaulted $(1)$ or did not default $(0)$ on their loan. It is very unbalanced, containing $5759$ individuals not defaulting and $1043$ defaulting on their loans.

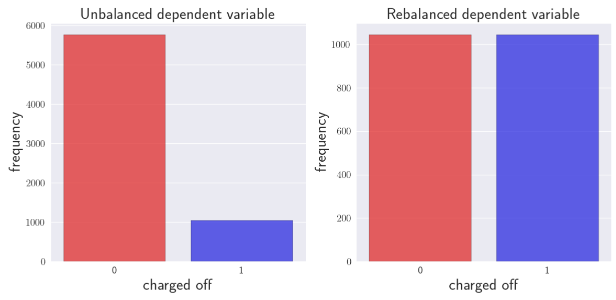

**Figure 1:** Dependent variable before and after under-sampling

The predictive power of models trained on this data may not be strong, which is why it is necessary for us to re-sample it. Techniques of re-sampling include over-sampling, which includes the under-represented class multiple times, and under-sampling, which reduces the over-represented class, removing observations. We chose to under-sample the data to mitigate the risks of over-fitting caused by over-sampling, at the cost of a reduced predictive power. The re-sampled data contains 1043 individuals defaulting and 1043 not defaulting, a data-set which is much smaller than the original. Apart from over-fitting, other benefits from taking this approach are faster computational efficiency of the machine learning models.

### 2.2 Feature Selection

The data-set originally contained 21 features. We removed *earliest_cr_line* and *issue_d* as they are variables referring to the issue date of the loans and when the earliest credit line was, which do not have much effect on the default state of a loan, but would add a lot of features to the model when turned into dummy variables. If left in, they could lead to over-fitting and less predictive power, due to the curse of dimensionality. The features *home_ownership*, *verification_status*, *application_type* and *purpose* are categorical. We transformed them into dummy variables before training the models.

Feature selection was conducted using "Recursive Feature Elimination", a sub-set of the *scikit-learn* library. The base model trained was a Logistic Regression model and the scorings used to determine the best model were: *AUC*, *accuracy* and *f1* (explained in more depth in the methodology section). Recursive Feature Elimination (RFE) works by recursively considering a smaller and smaller amount of features up to a certain threshold specified. Results showed that for all three measures considered, a model with 11 features performed the best. The selected important features were the following: *loan_amnt*, *term*, *installment*, *dti*, *open_acc*, *pub_rec_bankruptcies*, *total_acc*, *log_annual_inc*, *fico_score*, *home_ownership_RENT* and *mort_acc*.

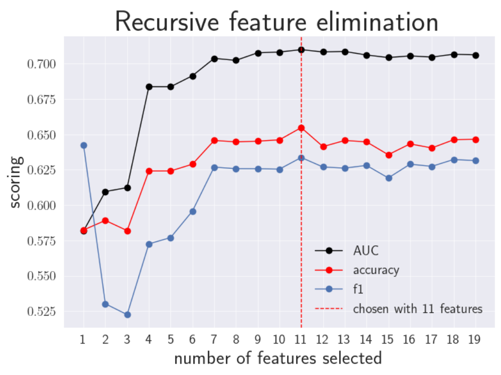

**Figure 2:** Feature Selection conducted based on a Logistic Regression model

Furthermore, we transformed the independent variables by removing the mean and scaling to unit variance using *scikit-learn*'s StandardScaler library, leading to an easier and more computationally inexpensive training of the models.

## 3 Methodology

### 3.1 Models fitted

The data-set was fitted with three different categories of supervised machine learning models: Logistic Regression, K-Nearest Neighbours (KNN) and Support Vector Machines (SVM).

#### 3.1.1 Logistic Regression

Logistic Regression is one of the classic methodologies used to tackle classification problems. It is a binary classifier which aims to define a hyper-plane which best separates the data. We are then able to predict the probability of a point lying on either side of the hyper-plane (the further away a point is from the division, the more confident we are in the prediction). The probability function is defined as the following:

$$\mathbb{P}(y^{(i)} = 1 | x^{(i)}; \beta) = \frac{1}{1 + \exp(-\beta_0 + \sum_{j=1}^{N} \beta_j x_j^{(i)})}$$

Logistic regression was implemented through the *scikit-learn* LogisticRegression class.

#### 3.1.2 K-Nearest Neighbours (KNN)

The KNN algorithm is based on the assumption that an observation will be very similar to the ones next to it. Given a number k (which we established through cross-validation), for each data-point the algorithm will look for the closest k value and classify it based on the prevalent class of its neighbours. We implemented this through the *scikit-learn* KNeighborsClassifier class.

### 3.1.3 Support Vector Machines (SVM)

Our data was then fitted with Support Vector Machines, which is more sophisticated and computationally intensive. The model aims to find a hyper-plane in which the distance between its further points is greatest. The parameter C, tuned through cross-validation measures the tolerance for error. We aim to solve the following optimisation problem:

$$\max_{\beta_0,\beta,M} M \;\; s.t. \;\; y^{(i)}\left(\frac{\beta_0}{||\beta||}+\frac{\beta^T}{||\beta||}x^{(i)}\right) \geq M(1-\epsilon_i), \forall i = 1,...,N$$
$$M \geq 0, \quad \epsilon_i \geq 0 \; \forall i, \quad \sum_i \epsilon_i \leq C$$

The linear version of this model will aim to find a hyper-plane which is linear, which may not be the best predictor, especially if the data-set has a large amount of non-linearity. We extended the basic model through kernels, testing both a polynomial kernel and a Gaussian Radial Basis Function (RBF) kernel. A *polynomial* kernel transforms a linear feature space into a polynomial one and has the following function:

$$\phi(x,z) = (x^T z + c)^d$$

where *d* is the degree of the polynomial (which will be tuned through cross-validation). A *Gaussian Radian Basis Function* kernel is the function defined as following:

$$\phi(x,z) = \exp(-\gamma||x-z||^2) \qquad \gamma = \frac{1}{2\sigma^2}$$

a bell shaped function (taking values in the range $[0,1]$) which aims to find support vector classifiers in infinite dimensions. *x* and *z* are two points and $\gamma$ is a scaling parameter (tuned through cross-validation), adjusting how wide or narrow the bell shape will be. Linear, polynomial and RBF kernels were all implemented through the *scikit-learn* SVC class (Support Vector Classifier).

## 3.2 Assessment measures

All of the models were fitted and tuned in the training set with a 5-fold cross-validation in order to generate an optimal set of hyper-parameters. A confusion matrix was generated with the test set, outlining the amount of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) present. In order to select the best set of parameters, the following three scoring measures (implemented with the *scikit-learn* metrics library) were used: *AUC*, *accuracy* and *f1 score*.

An ROC curve plots the rate of false positives against the rate of true positives for all probability thresholds in a classification problem. AUC is the metric referring to the area under the plotted curve, in the range $[0,1]$ with $1$ being the optimal model.

Accuracy score computes a ratio between the correctly observed results and the total amount of observations tested. Although the measure is simple, it may be misleading, as the incorrectly classified results could be unbalanced (e.g. all of them being false positives or false negatives).

To combat this, we used the *f1 score* measure, computed as following: f1 $= \frac{2TP}{2TP+FP+FN}$, which produces a score balancing between false positives and false negatives. This is important in our study as we are trying to find the optimal balance between miss-classified defaulters and miss-classified non-defaulters. This ensures the user of this model won't incur too much credit risk by giving loans to individuals who are likely to default, but avoids being too conservative, potentially losing out on profitable opportunities.

## 4 Results

All of the errors reported are out-of-sample errors (errors measured on the test set). To compute these errors, the data set was split as following: 80% was used to train and 20% was used to compute test errors.

### 4.1 Logistic Regression

The fitted logistic regression model had no hyper-parameters to tune and produced the following out-of-sample errors: $0.745$ (AUC), $0.682$ (accuracy) and $0.683$ (f1). It was performing relatively well despite its simplicity and computational efficiency, highlighting that it is possible for a classical and less-sophisticated model to achieve the best results.

### 4.2 K-Nearest Neighbours

The number of neighbours in the KNN algorithm was tuned in the training set using 5-fold cross validation. Values for k were selected in the range $[5, 1000]$, where the optimal parameter was found to be $500$. This was the value which maximised both the accuracy and f1 metrics. Slightly higher AUC values were observed with a larger value of k, at the cost of the other two measures. The model produced the following out-of-sample errors: $0.726$ (AUC), $0.653$ (accuracy) and $0.650$ (f1), results which are similar to the previously trained Logistic Regression model.

### 4.3 Support Vector Machines

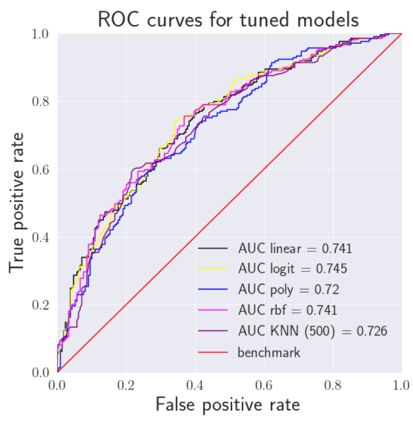

**Figure 3:** Comparison of three out-of-sample ROC curves for SVM models

Support Vector Machines were fitted with a *linear*, *polynomial* and *RBF* kernel. The hyper-parameter C (for error threshold) was tuned through cross-validation in all three models, the $d$ parameter was tuned for the *polynomial* model and the value for $\gamma$ was tuned for the *RBF* model. The linear model performed the best with $C = 1$, producing the following scores: $0.741$ (AUC), $0.667$ (accuracy) and $0.670$ (f1).

The *polynomial* support vector machine was tuned both with different degrees $[2, 3, 4, 5, 10]$ and different values of $C$ $[10, 5, 1, 0.01, 0.001]$, where the optimal parameters were found to be $d = 3$ and $C = 10$. A $C$ value of $0.01$ produced slightly better results with AUC, at the cost of a big drop in performance with the other two measures. The model fitted with optimal parameters obtained the following out-of-sample results: $0.720$ (AUC), $0.656$ (accuracy) and $0.673$ (f1).

The *RBF* Support Vector Machine was tuned with different values for $\gamma$ $[1, 0.1, 0.001, 0.0001, 0.00001]$ as well as different parameters for $C$ $[10, 5, 1, 0.01, 0.001]$, where the optimal results were found to be $\gamma = 0.001$ and $C = 10$ under all three scoring measures. The model fitted with optimal parameters obtained the following out-of-sample results: $0.741$ (AUC), $0.675$ (accuracy) and $0.681$ (f1).

### 4.4 Comparing models

Overall, the best performing model under the AUC metric was Logistic regression (Figure 4). This is surprising as it was the simpler model out of the list and also the most computationally efficient. Reasons for this could be that the model performs well even without a very large number of observations or features (we under-sampled our data therefore had a lot less observations to fit the models with) and the underlying data could be linear. However, overall there was not much difference in performance between models using the AUC metric (Figure 3). Under the *accuracy* and *f1* metrics,

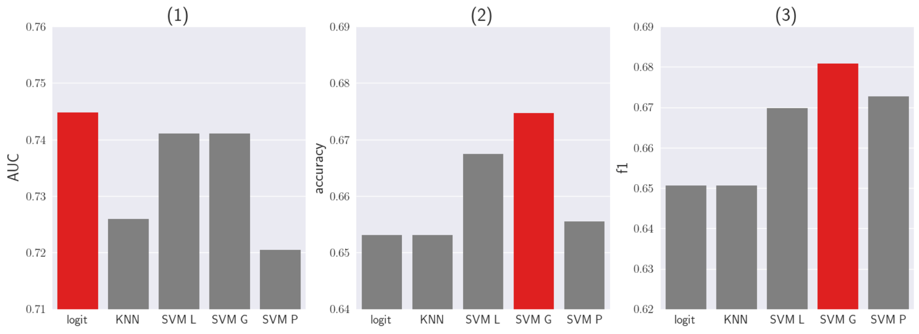

**Figure 4:** Comparison of different machine learning models (logit=Logistic Regression, KNN=K-Nearest Neighbours, SVM L=Linear SVM, SVM G=RBF SVM and SVM P=Polynomial SVM) under AUC (1), accuracy (2) and f1 (3) metrics

the RBF SVM model outperformed all of the others, although the results were not far superior to other fitted models. SVM models could have potentially performed much better with a larger amount of observations, which could have been achieved using over-sampling instead of under-sampling, with the risk of over-fitting.

## 5 Conclusion

In conclusion, the results obtained by our models were very counter-intuitive. While we originally expected for simpler models like Logistic Regression to be drastically outperformed by more complex ones like the Support Vector Machine, this was not the case (as RBF SVM outperformed Logistic Regression only slightly in two out of the tree scoring metrics). A cause of this could have been the choice of balancing the data using an under-sampling technique, which provided much less data to the model. Therefore, in a future iteration of this research, over-sampling could be tested to see if results improve. The high performance of Logistic Regression could have also been due to the feature selection process using it as a base model to select useful variables, therefore more accurate results could obtained using a different base model (e.g. KNN or SVM).