Predicting Default of Credit Card Clients Using Three Supervised Machine Learning Algorithms

Thomson Ly, Rene Schuller, Katarina Simanic, and Sukhpreet Singh

School of Computer Science
University of Windsor
Windsor, Canada
{lyt, schuller, simanic, singh2q2}@uwindsor.ca

Abstract

Credit risk management and prediction is crucial for lending institutions (i.e., banks) as substantial financial losses can result from the borrower's default. Machine learning methods can be used to measure and analyze credit risk in an objective manner. This study analyzes default payment data from a Default of Credit Card Clients Dataset provided by the UCI Machine Learning Repository containing 30,000 banking clients from Taiwan with 24 attributes. Prediction of default on the next credit card payment is compared using the following machine learning approaches; SVM (support vector machine), KNN (K-Nearest Neighbor), and Decision Tree. The experimental results indicate that SVM had the highest prediction accuracy for credit card default. Best hyperparameters for the three learning approaches were found. For Decision Trees, an entropy criterion, no max depth, nine max features and a best splitter were the most useful hyperparameters. With KNN the most significant hyperparameters were distance-based weights and using one neighbor. Finally SVC performed best when using an RBF (Radial Basis Function) kernel and a C value of 10 as the hyperparameters. Overall, the results do not provide high confidence in the efficacy of these trained models in predicting defaulting of credit cards due to the low F1-score obtained for default classification. Future work on other machine learning algorithms and hyperparameters is considered.

I. Introduction and Background

In Taiwan, in order to increase market share, card issuing banks over-issued cash and credit cards to applicants who were not qualified [4] [15]. Simultaneously, irrespective of their repayment ability, most cardholders overused credit cards for consumption and amassed hefty debts (credit and cash) [4] [15]. Consequently, credit card issuers in Taiwan faced a cash and credit card debt crisis which caused a blow to consumer finance confidence [15]. This crisis was a major challenge for both cardholders and banks [15].

Normally, in a well-engineered financial system, risk prediction is on the upstream while crisis management is on the downstream [15]. Risk prediction is important to predict business performance or individual customer credit risk, ultimately reducing uncertainty [15]. Risk prediction uses financial information, such as customer transactions and repayment records [15]. Here is where artificial intelligence and machine learning enters the picture. With the evolution of artificial intelligence and machine learning, various algorithms can be employed to predict risk for default of credit card clients. These methods can be used to predict the probability of a delay in the repayment of granted credit [15].

So what does it mean when your credit card is defaulted? Well, credit card default occurs when you have been extremely delinquent on your credit card payments [4]. Default is defined as when a client is unable to repay a loan [6]. The delinquency rate is indicative of the percentage of the past-due loans in the client's overall loan portfolio [5]. The increase of delinquencies results in a noteworthy amount of money loss for banks [5]. Thus, it is vital for banks to have a risk prediction model in place to be able to classify the characteristics that indicate clients who will have a higher probability of default on their credit card loans [5]. This model can be useful to both banks and clients, as it will allow banks to make informed decisions on credit card/loan applications and help bring more awareness to clients on behaviours that can damage their credit score [5].

The aim of the following study is to compare different supervised machine learning approaches and how they perform on the given Default of Credit Card Clients Dataset to accurately predict the target label (default) of an unseen dataset. This target label is used to declare if a customer will default or not default in the next month. The following approaches will be analyzed in relation to the research questions outlined below: Decision Tree, KNN (K-Nearest Neighbor) and SVM (support vector machine). The best hyperparameters for each approach being used will be determined. Finally a presentation of the results will be outlined, including a discussion, and recommendations for future work.

II. Reasearch Questions

- Which hyper-parameters are the most impactful for each of the different algorithms that are tested?
- Which machine learning algorithm is most effective in determining default of credit card clients?

III. Dataset

The dataset being used in the following paper was accessed from the UCI Machine Learning Repository and is titled: Default of Credit Card Clients Dataset [3]. The study completed by Yeh & Lien (2009) provided the data included in this dataset. Yeh & Lien (2009) took payment data in October 2005 from a major bank (a cash and credit card issuer) in Taiwan. The data is comprised of 30,000 customers from the bank, in which 22.12% have default payment on a credit card [15]. This is a multivariate dataset comprised of 30,000 instances (representing distinct credit card clients) and

24 attributes [3]. The attribute characteristics are integer variables. Each observation contains information on default payments, credit data, demographic factors, history of payment, and bill statements of credit card clients from April 2005 to September 2005 [4]. The following attributes are included in the dataset [15]:

- ID: an identifier for each client in the dataset
- LIMIT_BAL: the amount of credit given in NT dollars (including individual and family supplementary credit)
- SEX: the gender of the client (1=male; 2=female)
- EDUCATION: the education level of the client (1=graduate school; 2=university;3=high school; 4=others)
- MARRIAGE: the client's marital status (1= married; 2=single; 3=others)
- AGE: age in years
- PAY_0 to PAY_6: the history of past payments made on credit card in NT dollars from April to September 2005 (each pay period represents a month), -1= pay duly, 1=payment delay for one month; 2 – payment delay for two months; … ; 8=payment delay for eight months; 9=payment delay for nine months and above
- BILL_AMT1 to BILL_AMT6: the amount of the bill payment due for each month from April to September 2005 in NT dollars
- PAY_AMT1 to PAY_AMT6: the amount of previous payment for the month from April to September 2005 in NT dollars
- Target Variable: default payment next month (1 = Yes, 0 = No) – the last variable to be predicted

The main aim of the data presented is to determine clients that are predicted to have credit card default in the next month according to the target label, which is set to 1=Yes for defaulters and 0=No for non-defaulters [4]. Therefore it is a binary classification problem on a relatively unbalanced dataset, as outlined in Figure 1 below. Here we can see that 22.12% of clients will default in the following month. The goal is to train a machine learning model to predict if a customer will make a default on payment.

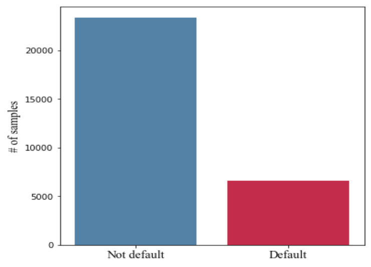


Figure 1: Representation of the Unbalanced Dataset [4].

IV. Literature review

The following studies discussed in this literature review all use the same dataset outlined above, Default of Credit Card Clients Dataset [3].

Yeh & Lien (2009) provided the dataset that is being used in the following paper and were the first to publish a study of the data mining techniques for the predictive accuracy of the probability of default of credit card clients. They identified that there was a real problem in questioning if estimated probability of default produced from data mining methods could represent the "real" probability of default [15]. Since the real probability of default was unknown, the study they completed proposed a novel "Sorting Smoothing Method" to deduce the real default probability [15]. They looked at differences of classification accuracy among six data mining techniques (discriminant analysis, logistic regression, Bayes classifier, KNN, artificial neural networks, and classification trees) and assessed if estimated probability of default produced from data mining methods represented the real probability of default [15]. The authors randomly divided the data into two groups, one for training and one to validate the model [15]. The results found that artificial neural networks performed classification more accurately than the other five methods studied. They found that the predictive probability used by the artificial neural networks was the only one that could be used to represent the real probability of default, and that estimating probability of default was more impactful than classifying clients into binary results (risky and non-risky) [15].

Franceschi (2022) completed a study that aimed to exploit some supervised machine learning algorithms to identify key drivers that determine the likelihood of credit card default. The following algorithms were used; Logistic Regression, Random Forest, and Decision Trees, and SVM (both linear and RBF) [4]. It was found that data preprocessing makes algorithms perform better than when they are trained with the original data, specifically principal component analysis results are approximately the same but computational cost is lowered [4]. The author discovered that oversampling performed slightly better than under sampling since the model was trained on a large amount of data [4]. Furthermore, all models that were implemented in this study were reported to have comparable results in terms of accuracy [4].

Choubey (2018) completed research based on non-parametric algorithms and their ability to predict the default risk for credit card users, developing a logistic regression-based model and machine learning approach. The author analyzed four models; Logistic Regression, Decision Tree, Random Forest, and ANN [2]. Then the results from the models were reviewed with the use of performance metrics such as; accuracy, confusion matrix, Receiver Operating Curve Area Under the Curve score (ROC curve/AUC-Score), and F1 score. The results revealed that the Random Forest model provided the best predictions that were the most stable.

Choi & Huang (2021) also completed a study to predict defaults of credit card clients with machine learning approaches. In their research, they created a framework that makes use of a simple feedforward neural network (or ANN) with a binary feature selection algorithm. Results were benchmarked with other machine learning algorithms as well; SVM, Logistic Regression, and Random Forest [1]. The results of their study indicated that the ANN outperformed other algorithms.

Mesri et al. (2021) presented research that experimented with three machine learning algorithms, Decision Tree, ensemble techniques, and adaptive boosting. All the models performed binary classification to classify which customers would pay their debt and which would not

[8]. The results showed that all the models gave virtually the same results based on the following performance metrics; precision, recall, and F1-score [8].

Jæger-Pedersen (2020) completed a study wherein machine learning techniques are used on the previously outlined dataset. The machine learning methods included in this study are; Logistic Regression, Decision Tree, Random Forest, XGBoost, KNN, and ANN. ROC AUC and confusion matrix were used to assess all the models' performances [6]. It was found that Random Forest was the best performing algorithm, after that it was XGBoost and ANN [6]. Otherwise the author found the worst performing algorithms were KNN and Decision Tree [6].

Overall, the results of the literature review showed that some machine learning algorithms can be used to accurately predict the probability of credit card default. In the studies presented, various algorithms were used, such as SVM and Decision Trees. Each study yielded slightly different results when it came to determining the best performing algorithm, however, there was a general consensus that data preprocessing and feature selection could improve the accuracy of the models. In addition, some studies suggested that estimating the probability of default is more impactful over classifying clients into binary results.

V. Methodology

Python language was used to conduct the analysis in the following study. Several machine learning libraries, and their available statistical frameworks were used. Pandas and numpy were used for mathematics and data cleaning, filtering, and transformation. Seaborn and Matplotlib.pyplot were both used for visualization, such as creating correlation matrices. For the machine learning components, SciKit Learn was the primary library that was used, it includes all the major machine learning algorithms (i.e. KNN, SVC and Decision Trees) as well as tools for the preprocessing of data. In addition IMBlearn was used for oversampling in the data preparation stage.

A. Decision Tree

A Decision Tree is a nonparametric model [2] meaning that no assumptions are made for the data distribution, and it has lazy learning, which means that it doesn't need any training data points, as all training data is utilized in the testing phase [14]. It is a tree-like structure that lets the system weigh possible actions to make decisions [5]. It has a relatively simple layout, and can be binary or non-binary, it starts with a root node, which is the first and main node representing the entire population and branches out to all possible outcomes [6]. Each feature from the dataset is set as an internal node leasing to additional nodes until the leaf node is reached through testing the characteristics of every instance [5]. Based on the testing result, the resultant branch is chosen [5]. At the leaf, the end of the branch, the node is no longer splitting, and a decision is made for that instance [5]. Thus, by following the feature values on the Decision Tree we will be able to obtain the predictive value [1]. There are two types of Decision Trees, regression tree and classification tree, in the following study, the classification tree is used [5]. The classification tree classifies a target into a category [5]. In the following paper, the leaf represents if the credit card holder is a non-defaulter or defaulter [5].

B. KNN - K-Nearest Neighbor

KNN is another nonparametric technique. KNN uses K to classify data to its group, with K representing the number of its nearest neighbors, which is the main deciding factor [12]. With this measure, distance between all points is calculated and then based on previously calculated distances, the closest K points are found [12]. Distance between objects can be determined using distance metrics such as Euclidean, Manhattan, or Minkowski [1]. Then, based on the bulk of the surrounding points, the class is chosen [12]. The K is a positive integer, and when it takes the value of 1, for example, this means that the model is classified to the class of the singular nearest neighbor [12]. Otherwise, when this model is used for classification problems, the output can be computed from the K-most similar instances as the class with the highest frequency [12]. Essentially, the KNN model makes a prediction by referring to the K nearest neighbors of the new object [7]. The model determines the sample's label based on known labels from nearby samples, and it does not "get trained", rather it memorizes. To train a boundary that is used for future predictions and separates two categories, SVM is a more favorable model [7].

C. SVM - Support Vector Machine

SVM is a machine learning algorithm that is used for classification and regression analysis. It is parametric, meaning that it "*summarizes data with a set of parameters of fixed size*" [11]. It is a linear classification algorithm whose goal in the data dimension is to separate two classes through a hyperplane [4]. SVM is good for a small dataset that has few outliers [1]. The idea behind SVM is to find a hyperplane to separate data points, and this hyperplane will divide the space into different domains (each having a type of data) [1]. There are many hyperplanes that can be chosen to separate the two classes of data [1]. Using the kernel trick, the model will increase the dimension of the dataset if it cannot be perfectly separated [7]. Essentially, the goal is to find a plane that has the maximum margin, which is the distance between the hyperplane and the two closest data points that correspond to 2 subclasses [1]. SVM attempts to optimize the algorithm by trying to maximize the margin value to find the best super plane to divide the data into two layers [1]. Support vectors are the data points that are closest to the hyperplane, and a hyperplane is a linear surface that splits the space into two sections [1]. When the hyperplane has been set, the prediction rule is relatively straightforward, it is based on if the test point is on one side or the other of the hyperplane [4]. If more than one hyperplane fits, then the higher margin (distance) between the support vector is chosen [4].

D. Hyperparameter Optimization/Tuning

The machine learning models that are presented and used in this study require hyperparameter optimization in order to get the best predictions. Hyperparameters are parameters that are defined before the training process starts and govern how the model is trained [6]. Hyperparameters can be changed manually when building machine learning models and are not simply learned from the training data [6].

Hyperparameters can have an impact on the performance of the model and influence various factors, such as the complexity of the model and speed of training. Hyperparameter optimization or tuning is the process of locating the most optimal values of hyperparameters for a given machine learning model. This process can be lengthy depending on the complexity of the model and size of the data, and even though it increases the computation time, it also increases the prediction accuracy [6]. Unfortunately, there is generally no singular solution and brute forcing an optimal solution is often needed [6]. This can be completed in several ways, but for the purpose of this study, GridSearchCV will be used. GridSearchCV is a method for hyperparameter optimization/tuning that requires exhaustively searching over a specified parameter grid using cross-validation to evaluate the performance of every combination of hyperparameters [6]. In this study, the "GridSearchCV" package from the "scikit-learn" library was used to perform hyperparameter optimization. Here the process of performing a grid search with cross validation is automated and the optimal hyperparameters that maximize the chosen performance metric are returned. In this approach, an exhaustive search over specified hyperparameter values are performed and the model is evaluated with a k-fold cross-validation [2]. With the k-fold cross validation approach, the data is divided into k-subsets or "folds" of relatively equal size, and model tests are performed k times; for each iteration the model is trained on k-1 folds and tested on the remaining folds [2]. Then the average of k performances of the model is obtained as an estimate of the model's performance [2].

The hyperparameters that will be tuned for the Decision Tree model are criterion, maximum number of features, maximum depth, and splitter. Then the tree is tuned by completing cross-validation and taking the tree with the smallest error rate [6]. Criterion is the hyperparameter that measures the quality of a split. It utilizes two different probability measures of Entropy or Gini Index [13]. Entropy measures the amount of uncertainty in a probability distribution of the dataset. The entropy measure indicates how much of the information is gained. Information Gain or reduction in entropy is calculated by taking the difference between uncertainty before the split and the uncertainty after the split. A lower entropy score indicates a greater Information Gain whereas a higher entropy score indicates a lower Information Gain. The second measure is the Gini index which measures the probability of a random instance being incorrectly classified when selected randomly [6]. The lower the Gini index, the lesser the misclassifications whereas, the higher the Gini index, the greater the misclassifications. The maximum number of features considers which number of features are optimal for a node split. Having a large number of features can lead to the Decision Tree overfitting the data. Whereas a lower number of features can lead to underfitting [13]. In this experiment a range from 1 to 23 features are considered to predict if it will lead to a client defaulting their credit card. Maximum depth controls how deep the tree can grow during the training process. If the tree depth is too great this can lead a Decision Tree model to overfit the data, in which the model becomes overly complex and learns the noise within the training data [5]. This leads to weak generalization when new data is introduced. On the other hand a shallow Decision Tree can lead to underfitting. This results in the model being overtly simple which leads to low accuracy on both the training and testing data. In this experiment the maximum depth is constrained to a range of 1 to 20 or none. If maximum depth is set to none, this means that the tree will keep growing until it reaches a pure node (node that will either classify that a client will default or not default). This allows GridsearchCV to explore all the appropriate depths to strike a balance between overfitting and underfitting. The Splitter hyperparameter chooses which strategy will result in an optimal split (partitioning of data) at each node. The Decision Tree model will select between two strategies, either "best" or "random". The "best" splitter iterates through all the potential splits and selects the "best" one. Whereas, the "random" splitter, selects a feature at random then splits it, at random and then calculates either the Gini or entropy score [13].

Hyperparameter optimization for KNN involves setting the correct number of neighbors, K, which is important to get a good result with this method [6]. It is important to note that low values of K results in high variance, but low bias which leads to overfitting due to sensitivity to outliers present in the dataset. Comparatively, higher values of K results in low variance, but high bias which leads to underfitting [11]. Thus, finding the optimal K value means finding the value that best balances the underfitting and overfitting of the model. In addition, another hyperparameter being optimized is the distance metric. Specifically, the following three distance metrics, Euclidian, Manhattan, and Minkowski will be explored in the experiments. Euclidean distance is used if the dimensions are measuring similar properties such as width, height, and depth of parts [11]. Manhattan distance is used if the dimensions are measuring dissimilar properties such as age, weight, and gender of a patient [11]. Finally, Minkowski distance is considered a generalization of both the Euclidean distance and Manhattan distance, and thus typically used as the distance metric hyperparameter [11].

Another hyperparameter that is worth considering is weights. Sometimes when evaluating a query point, better results are found if neighbours that are closer are assigned higher significance when computing the fit, which is known as a distance-based weight. Uniform based weight does not assign higher significance to any points regardless of distance [13].

For SVM, the parameters being optimized are the C value and the kernel function. Kernel specifies the choice of which kernel type is to be used in the algorithm [9]. For the C value, having a low C value (e.g. 0.1) creates a larger hyperplane margin which can result in a higher amount of misclassification leading to model underfitting. Comparatively, having a high C value (e.g. 100) creates a narrower hyperplane margin which forces more training points to be correctly classified, leading to model overfitting [13]. For spaces where target labels are not easily linearly divisible (i.e. a two-dimensional space), it is possible to use the kernel trick to help consider the problem in a different dimension. A kernel function, such as RBF, is applied to feature variables in order to transform the data so that the SVM algorithm can function in a higher dimension space.

E. Performance Metrics

Performance metrics are measurements used to evaluate the performance of machine learning models. For the purpose of the following study, the following metrics will be used to measure the model's performance; accuracy, precision, recall, and F1 score. For the formulas presented, the following acronyms are used. TP for true positives is for samples whose prediction is non-defaulted and the true class is non-defaulted. TN for true negatives is for samples whose prediction is non-defaulted and the true class is non-defaulted. FP for false positives is for samples whose prediction is non-defaulted, but the true class is defaulted. FN for false negatives is for samples whose prediction is defaulted but the true class is non-defaulted [4].

Accuracy measures the proportion of correct predictions made by a model over the total number of predictions made. The formula is shown below [10].

$$accuracy = \frac{tp+tn}{tp+fp+fn+tn}$$

Precision measures how many true positives there are in relation to the total number of predicted positives, or how many of these predicted positives are correctly identified as positive [10]. In the case of predicting default of credit card clients this would refer to proportion of correctly predicted defaults out of all predicted defaults. The formula is shown below [10].

$$Precision = \frac{tp}{tp+fp}$$

Recall, also known as sensitivity or true positive rate, is a measure that calculates the proportion of true positives over the total amount of actual positives, or rather how many positive instances are correctly predicted as positive [10]. The formula is shown below [10]. This would be the proportion of correctly predicted defaults out of all actual defaults.

$$recall = \frac{tp}{tp+fn}$$

Finally, the F1-score (or F-score) is the harmonic mean of precision and recall (or sensitivity) and is a metric of accuracy [10]. The formula is shown below [10].

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

The most important performance metric is dependent on the specific problem being addressed and goals of the model. Due to the unbalanced distribution of the data, the four-performance metrics outlined above are deemed to be important [10]. For the purpose of this paper, F1-score is considered to be the most important of these performance metrics for predicting default of credit card clients since it implies that there is a need to balance the trade-off between precision and recall in the model predictions. If the F1 score is high that means that the model has a good balance between precision and recall, and therefore is able to correctly identify both positive (defaults) and negative cases (non-defaults) with minimal errors. This is important for the following study since incorrectly classifying a non-default as a default or vice versa could lead to significant financial implications for both client and bank.

VI. Experimental setup

The experiment setup was primarily conducted using Jupyter Notebooks, with the main programming language used being Python version 3. Before conducting any machine learning, some basic exploration and data processing is required. Immediately after reading in the dataset, the first basic cleaning performed was dropping the 'ID' column as this is just an arbitrary primary key and thus will have no direct impact on the prediction of the target label, default. Following this, the other columns were explored to ensure that there were no null/missing values, and it was found that the dataset was complete.

Next categorical variables were identified and explored to make sure there was no unusual or unknown data. When observing the values present for Marriage and Education, it was noted that there were values not documented within the described category, e.g. Marriage (Marital Status) describes 1, 2, 3 as married, single or other respectively, yet there is an additional 0 category containing 54 values. Similarly, for the Education values, the range 1-4 are documented as graduate, university, high school, or others respectively but we see additional values for the undocumented categories of 0, 5, 6. Two methods were considered for dealing with the undocumented data in these columns; either delete the rows associated with them or assign those rows with the mode for that feature (e.g. for marriage it would be 2). Based on the low number of rows with undocumented values relative to the total row count, it was determined to use the first method of deletion. As a result only 399 values were removed of the total 30000 values, which is minimal loss.

To get a sense of how the different variables relate to one another and especially how they are related to the target label, a correlation matrix was built, as seen in Appendix B. It was found that whether a customer defaulted on their credit cards had little to no correlation with any of the other variables. Thus, when the machine learning training and testing is implemented in the experimentation, results are expected to have potentially low F1 score (more false positives and negatives) as it will be difficult for the machine learning algorithms to find strong patterns in the data.

Following exploration, preprocessing of the data was undertaken before the machine learning training could commence. First, the dataset was split into 80% training set, and a 20% testing set. Subsequently, feature scaling is done because machine learning algorithms tend to perform better when features are similarly scaled. There are two possible approaches for feature scaling: min-max scaling (normalization) and standard scaling (standardization). In min-max scaling, values are rescaled to fall between a given range (e.g. 0 and 1). While standard scaling rescales values but does not restrict them to a specific range. This makes standard scaling less sensitive to outliers and based on this fact, standardization was chosen for this experiment. Thus, standardization was done on the dataset by fitting the standard scaler to the training set and transforming both the training and test set off of that fit. By fitting only on the training data, we avoid any information leak from the test set into the training set. The final step in the data preprocessing was oversampling of the training set due to the imbalance in total values between the two categories in the target label (18405 for not defaulted vs 5275 for defaulted). The reason oversampling was chosen instead of under sampling for these experiments is due to Franceschi's (2022) previous work where oversampling was found to perform better.

The data is now prepared for training on each of the three machine learning algorithms (i.e. SVM, KNN and Decision Tree). For each of these algorithms a similar training and evaluation process was followed. To do so, GridSearchCV as mentioned previously, was used in order to determine the most optimal hyperparameters for each machine learning algorithm. For Decision tree, the criterion (Gini or entropy), splitter (best or random), maximum number of features (1 to 23) and max depth of the tree (1 to 20 or none) were tested. For KNN the number of neighbors (from 1 to 19), the weight given to a neighbor (uniform or distance) and different metrics for distance computation (Euclidian, Manhattan, Minkowski) were tried. Finally for SVM, the hyperparameters of kernel type (RBF kernel vs Linear kernel) as well as different values for the regularization parameter C (0.1, 1, 10) were considered. A thing to note for GridSearchCV, 3-fold cross validation instead of the standard/default 5-fold cross validation was used to keep computation time low and reasonable.

These models were then formally trained using the best hyperparameters discovered during the search. The trained models were deployed on the test set to gauge their capabilities. A confusion matrix as well as accuracy, precision, recall and F1 scores were printed.

VII. Experimental results

A. Decision Tree Results

Decision Tree is the first machine learning method that was used. Decision Tree was implemented with GridSearchCV to find optimum hyperparameters. Doing 3-fold cross validations for a total of 5796 fits, the optimum score found was to be 0.880. The score corresponded with the following hyperparameters: criterion was entropy, maximum depth was none, maximum number of features was 9, and the splitter was best. Examining the hyperparameter criterion, entropy was selected over Gini. Entropy as mentioned earlier measures the uncertainty of the information gain from the features. The outputted hyperparameter presumably depends on which value is less, the Gini index or entropy. In this case since entropy has a lower value than the Gini index, the ideal hyperparameter is entropy. Presumably, the optimal maximum depth is none since the algorithm had to grow until it reached a pure node to classify if the client will default or not. However, the maximum number of features of 9 produced the best score. Presumably this value strikes a balance which counters the overfitting and underfitting problem that tends to arise if there are too many or too few maximum number of features.

Following this, the Decision Tree model is then trained with the optimal parameters of criterion set to entropy, the maximum number of features set to 9, maximum depth set to none, and the splitter is set to "best". Applying this model to the test dataset yielded the following confusion matrix which had 3835 true positives, 545 true negatives, 756 false positives and 785 false negatives. The results of this matrix were used to compute the precision, recall, F1 and accuracy scores (see Figure 2). For the F1 score, it was observed that the Decision Tree model has a high F1-score of 0.83 for predicting non-defaulting of credit cards, but an average to poor score of 0.41 for predicting defaulting of credit cards.

	Precision	**Recall**	**f1-Score**
0	0.83	0.84	0.83
1	0.42	0.41	0.41
Accuracy			0.74

Figure 2: Decision Tree Score Table.

B. KNN - K-Nearest Neighbor Results

Continuing, the next machine learning algorithm experimented was KNN. Again GridSearchCV was executed to find the best hyperparameters. Following 3-fold cross validations for a total of 342 fits, the best score was found to be 0.877. This score corresponded to the following hyperparameters: metric was Euclidean, the number of neighbours was 1, and the weights were uniform.

When considering why Euclidean might be the best choice for the metric hyperparameter, it should also be noted how this score compares to other scores if we change the metric to Manhattan or Minkowski while holding the other hyperparameters static (e.g. number of neighbours to 1 and weights to uniform). Doing so, the best score seen when the metric is set to Manhattan was 0.874, while the best score seen when the metric is set to Minkoski was 0.877. Based on these results, it seems that Euclidean and Minkowski are almost identical in score, while Manhattan performs only slightly worse by a mere difference of 0.003. Thus, it can be inferred that the metric hyperparameter does not seem to significantly impact the success of the KNN model.

When observing the number of neighbours hyperparameter, it was seen that as the number of neighbours increases, the best scores tend to decrease, reaching as low as 0.711. This may be because the relevant data clusters for decision making are small, and that it may be best to only consider the singular closest data point when making a decision. However, this result is unexpected as usually lower neighbour counts are susceptible to overfitting on the training data, as this complex model would be the most impacted by outliers [11]. Generally, 5 nearest neighbours tends to strike a better balance between overfitting and underfitting and it is the default value used by the KNN in SciKit Learn.

The last hyperparameter explored was weights and it was seen as the most significant factor in determining the best score. It was found that weighted distance performed better than uniform distance calculations in almost all fits. This result is expected, as it is presumed that points that are closer to the target datapoint would have more commonalities and significance than those that are further away. Note, this is irrelevant when a neighbour is equal to one because there is only one neighbour to consider. Therefore when the final model is trained, with the number of neighbours equal to one, it does not matter if weight is specified as uniform or distance.

The KNN model was trained with the best hyperparameters (i.e. Euclidian, 1 neighbour and uniform weight) and then applied to the test set. This yielded the following confusion matrix which had 3827 true positives, 503 true negatives, 764 false positives and 827 true negatives. The results of this matrix were used to compute the precision, recall, F1 and accuracy scores (see Figure 3). Focusing on the F1 score, it is observed that the KNN model has a high F1-score of 0.83 for predicting non-defaulting of credit cards, but a relatively poor F1-score of 0.39 for predicting defaulting of credit cards.

	Precision	**Recall**	**f1-Score**
0	0.82	0.83	0.83
1	0.40	0.38	0.39
Accuracy			0.73

Figure 3: KNN Score Table

C. SVM - Support Vector Machine Results

The final machine learning algorithm experimented was SVMs using the C-Support Vector classification implementation of SciKit Learn. Following similar methods as the previous two experimentations, for a total of 18 fits, the best score was found to be 0.737. This score corresponded to the following hyperparameters: 'C' was 10, and the kernel function was RBF.

A possible reason that 10 may have been the best value found for C is that it provides the greatest balance between risk of overfitting and underfitting. Based on the experimentation it is likely that values lower than 10 resulted in underfitting the data by being too lenient with the margin leading to a lower score, and thus GridSearchCV determined having a C value of 10 to be the most optimal hyperparameter.

Finding that this model performed best with a kernel function of RBF is understandable when considering the complexity and multidimensionality of the dataset. Given the large number of features within this dataset, finding a linearly separable solution would not be possible. This is evident in the experimentation by how any combination of other hyperparameters with RBF always outperformed other combinations of hyperparameters with the linear kernel function. By choosing RBF as the kernel function, this allows the SVM model to capture complex and non-linear decision boundaries to separate the two classes (defaulted and non-defaulted) of this classification.

The SVM model was trained with the best hyperparameters (C = 10, kernel function as RBF) and then applied to the test set. This yielded the following confusion matrix which had 3827 true positives, 775 true negatives, 852 false positives and 555 false negatives. The results of this matrix were used to compute the precision, recall, F1 and accuracy scores (see Figure 4). For the F1 score, it was observed that the SVM model has a high F1-score of 0.81 for predicting non-defaulting of credit cards, but an average F1-score of 0.52 for predicting defaulting of credit cards.

	Precision	**Recall**	**f1-Score**
0	0.87	0.81	0.84
1	0.48	0.58	0.52
Accuracy			0.76

Figure 4: SVM Score Table

VIII. Discussion of findings

The tuning of hyperparameters and subsequent choosing of the best hyperparameters for the three supervised learning methods, Decision Trees, KNN and SVM were completed. Consecutive training and testing of these new models were recorded and analyzed. The comparison of the confusion matrix values are shown in Figure 5. As well, the F1 scores for both defaulted and non-defaulted classification, along with accuracy of the models are visualized in Figure 6. It can be observed that the highest F1 scores (0.84, 0.52) and accuracy (0.76) corresponds to SVM, indicating that it performed the best on the test sets out of all the supervised learning methods. This is evident through the confusion matrix, as it can be seen that the SVM model is able to predict much more true negatives (and consequently lesser false negatives), than either Decision Tree or KNN models.

In the Decision Tree algorithm, crucial hyperparameters included the criterion, maximum number of features, and splitter. Analysis of the results revealed that employing the entropy criterion led to superior performance. By setting the maximum number of features to 9, the dataset's balance was enhanced, mitigating overfitting and underfitting concerns. Furthermore, employing the 'best' splitter strategy contributed to an improved model score. Notably, the hyperparameter pertaining to maximum depth did not exert a substantial influence on the final results. The most important hyperparameters found for a KNN model was weight and number of neighbors. A distance-based weighing generally leads to greater scoring for models during cross validation. In terms of neighbors, lower amounts of neighbors tended to perform better. This would not likely be the case for all datasets, but rather reflects the high dimensionality and distance between neighbors in this dataset. The metric for distance measurement did not have much impact on the success of the algorithm. In SVM both the C value and use of the kernel trick had great influence on how the model performed. For this dataset using RBF kernel to translate the problem to a higher dimension was invaluable. As well, using a stricter, more exclusive margin (C value of 10) resulted in better outcomes as opposed to larger margins (C value of 0.1, 1).

Other notable observations are that generally F1 scores for defaulted credit card predictions were not very high, as seen by the F1 scores being 0.41, 0.39, and 0.52 for Decision Tree, KNN and SVM respectively. This suggests that the models are much better at predicting when a customer will not default on their credit card, then when they will default. A few potential explanations for this phenomenon are considered. It is possible that due to the generally low correlation between features of the dataset and the target label as indicated by the correlation matrix seen in Appendix B (highest correlation being 0.33), it was difficult to build models on top of these weak relationships. Although oversampling was used to balance the dataset during the training phase of the model, the imbalanced dataset of the actual test set could have been responsible for the low F1 scores seen for the defaulted classification. Due to this low F1 score for default classification, there is not much reason to have high confidence in the efficacy of these trained models in predicting defaulting of credit cards. Despite the fact that accuracy scores observed were relatively high, being 0.74, 0.73 and 0.76 for Decision Tree, KNN and SVM respectively, these results are potentially misleading. This is because if the model were to simply predict all samples to be the major class (non-defaulted), it could obtain fairly high accuracy due to the imbalanced values for the label.

	True Positives	True Negatives	False Positives	False Negatives
Decision Tree	3835	545	756	785
KNN	3827	503	764	827
SVM	3739	775	852	555

Figure 5: Confusion Matrix Table

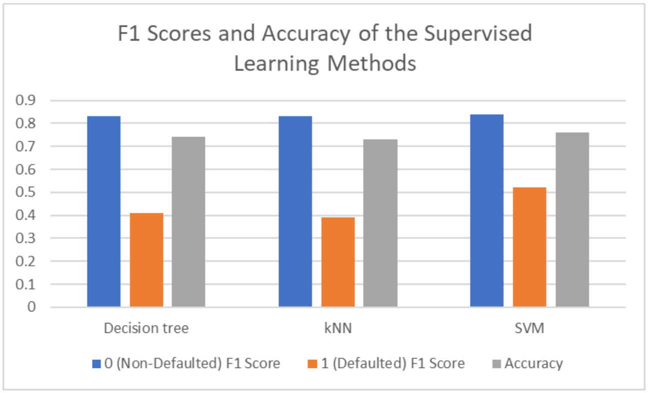


Figure 6: F1 Scores and Accuracy of the Supervised Learning Methods

IX. Conclusion and Future Work

The aim of this paper was to discover which common supervised machine learning algorithm, Decision Trees, KNN or SVM would perform best at predicting if credit card holders would default. A secondary goal was to determine how best to tune these learning models using hyperparameters. It was discovered that none of the machine learning models performed well enough to accurately predict if credit card holders would default. Out of those models that were tested, SVM had the best performance when predicting on the test set. The hyperparameters that emerged as most essential were using a criterion of entropy, 9 features and a best splitter algorithm in Decision Trees, a low number of neighbours and distanced based weight for KNN and using an RBF kernel and a higher C value of 10 for SVC.

Many directions for future research have emerged from this study. Different data scaling strategies could be employed to see if this leads to better training of models, for instance using the min-max scaler instead of the standard scaler. It would also be interesting to do higher fold cross validation when using GridSearchCV, perhaps 5 instead of 3. In terms of preprocessing, using one hot encoding on our categorical data (i.e. sex, marriage, and education) may lead to different results due to easier machine-readable data. Another direction would be to consider different variations in hyperparameters (e.g. trying C values up to 100 in SVM) or looking at hyperparameters that were outside the scope in this project (e.g. the gamma value in SVM, minimum samples split in Decision Trees). Building on our Decision Tree model and using ensemble learning to create a random forest is worthy of examination. Finally exploring how other supervised learning methods, such as neural networks, performed on this dataset compared to those we trained would be worthy of exploration. Especially considering that neural networks were found to be more effective when used on this dataset [15].

References

[1] Choi, P. M. S., & Huang, S. H. (2021). Using Machine Learning to Predict the Defaults of Credit Card Clients. In Fintech with Artificial Intelligence, Big Data, and Blockchain (pp. 133–152). Springer Singapore Pte. Limited. https://doi.org/10.1007/978-981-33-6137-9_4

[2] Choubey, A. M. (2018). Predicting Credit Default Risk via Statistical Model and Machine Learning Algorithms. ProQuest Dissertations Publishing.

[3] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. *Default of Credit Card Clients Data Set.* Irvine, CA: University of California, School of Information and Computer Science.

[4] Franceschi, R. (2022). Machine learning algorithms for predicting default of Credit Card clients. GitHub. Retrieved from https://github.com/robertofranceschi/Default-Credit-Card-Prediction

[5] Gui, L. (2019). Application of Machine Learning Algorithms in Predicting Credit Card Default Payment. ProQuest Dissertations Publishing.

[6] Jæger-Pedersen, R. (2020). Modelling probability of default with machine learning : how well does machine learning perform and can it replace the standard methods?

[7] Liu, R.L. (2018). Machine Learning Approaches to Predict Default of Credit Card Clients. Modern Economy, 9, 1828-1838. https://doi.org/10.4236/me.2018.911115

[8] Mesri, K., Tahseen, I., & Ogla, R. (2021). Default on a credit prediction using decision tree and ensemble learning techniques. Journal of Physics. Conference

Series, 1999(1), 12121–. https://doi.org/10.1088/1742-6596/1999/1/012121

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[10] Rtayli, N., & Enneya, N. (2020). Enhanced credit card fraud detection based on SVM-recursive feature elimination and hyper-parameters optimization. Journal of Information Security and Applications, 55, 102596–. https://doi.org/10.1016/j.jisa.2020.102596

[11] Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). United Kingdom: Pearson.

[12] Sariannidis, N., Papadakis, S., Garefalakis, A., Lemonakis, C., & Kyriaki-Argyro, T. (2020). Default avoidance on credit card portfolios using accounting, demographical and exploratory factors: decision making based on machine learning (ML) techniques. Annals of Operations Research, 294(1-2), 715–739. https://doi.org/10.1007/s10479-019-03188-0

[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[14] Vishwakarma, S. K., Rasool, A., & Hajela, G. (2021). Machine learning algorithms for prediction of credit card defaulters—A comparative study. In Proceedings of International Conference on Sustainable Expert Systems: ICSES 2020 (pp. 141-149). Springer Singapore.

[15] Yeh, I.-C., & Lien, C. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, *36*(2), 2473–2480. https://doi.org/10.1016/j.eswa.2007.12.020

Appendix A

Libraries Used in the Experimentation

ImbalancedLearn

- https://imbalanced-learn.org/stable/

Matplotlib.pyplot

- https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html

NumPy

- https://numpy.org/

Pandas

- https://pandas.pydata.org/

Scikit-Learn

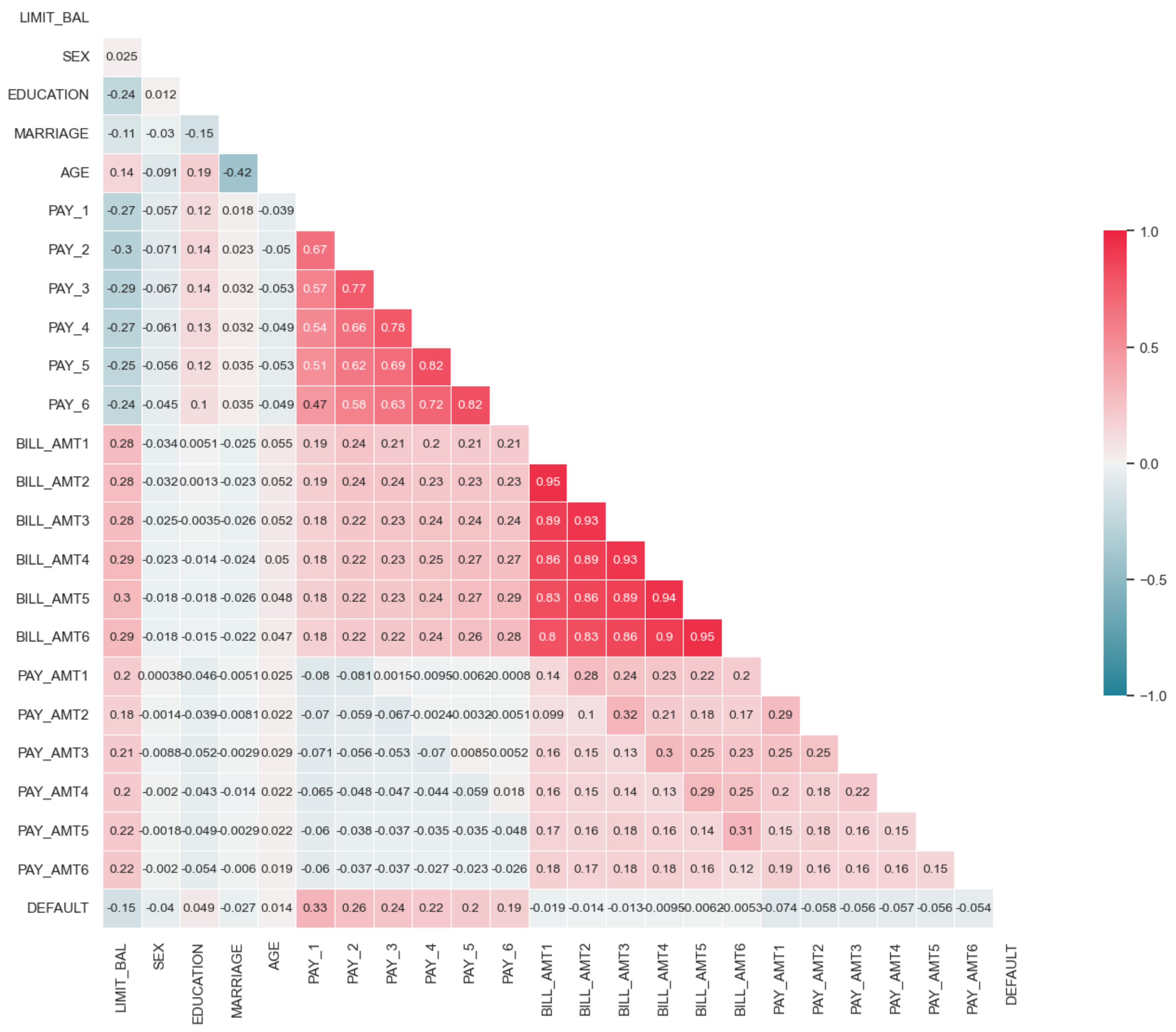
- https://scikit-learn.org/stable/

Seaborn

- https://seaborn.pydata.org/

Appendix B

Correlation Matrix of All Attributes in the Dataset



	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULT
LIMIT_BAL																								
SEX	0.025																							
EDUCATION	-0.24	0.012																						
MARRIAGE	-0.11	-0.03	-0.15																					
AGE	0.14	-0.091	0.19	-0.42																				
PAY_1	-0.27	-0.057	0.12	0.018	-0.039																			
PAY_2	-0.3	-0.071	0.14	0.023	-0.05	0.67																		
PAY_3	-0.29	-0.067	0.14	0.032	-0.053	0.57	0.77																	
PAY_4	-0.27	-0.061	0.13	0.032	-0.049	0.54	0.66	0.78																
PAY_5	-0.25	-0.056	0.12	0.035	-0.053	0.51	0.62	0.69	0.82															
PAY_6	-0.24	-0.045	0.1	0.035	-0.049	0.47	0.58	0.63	0.72	0.82														
BILL_AMT1	0.28	-0.034	0.0051	-0.025	0.055	0.19	0.24	0.21	0.2	0.21	0.21													
BILL_AMT2	0.28	-0.032	0.0013	-0.023	0.052	0.19	0.24	0.24	0.23	0.23	0.23	0.95												
BILL_AMT3	0.28	-0.025	-0.0035	-0.026	0.052	0.18	0.22	0.23	0.24	0.24	0.24	0.89	0.93											
BILL_AMT4	0.29	-0.023	-0.014	-0.024	0.05	0.18	0.22	0.23	0.25	0.27	0.27	0.86	0.89	0.93										
BILL_AMT5	0.3	-0.018	-0.018	-0.026	0.048	0.18	0.22	0.23	0.24	0.27	0.29	0.83	0.86	0.89	0.94									
BILL_AMT6	0.29	-0.018	-0.015	-0.022	0.047	0.18	0.22	0.22	0.24	0.26	0.28	0.8	0.83	0.86	0.9	0.95								
PAY_AMT1	0.2	0.00038	-0.046	-0.0051	0.025	-0.08	-0.081	0.0015	-0.0095	0.0062	0.0008	0.14	0.28	0.24	0.23	0.22	0.2							
PAY_AMT2	0.18	-0.0014	-0.039	-0.0081	0.022	-0.07	-0.059	-0.067	-0.0024	0.0032	0.0051	0.099	0.1	0.32	0.21	0.18	0.17	0.29						
PAY_AMT3	0.21	-0.0088	-0.052	-0.0029	0.029	-0.071	-0.056	-0.053	-0.07	0.0085	0.0052	0.16	0.15	0.13	0.3	0.25	0.23	0.25	0.25					
PAY_AMT4	0.2	-0.002	-0.043	-0.014	0.022	-0.065	-0.048	-0.047	-0.044	-0.059	0.018	0.16	0.15	0.14	0.13	0.29	0.25	0.2	0.18	0.22				
PAY_AMT5	0.22	-0.0018	-0.049	-0.0029	0.022	-0.06	-0.038	-0.037	-0.035	-0.035	-0.048	0.17	0.16	0.18	0.16	0.14	0.31	0.15	0.18	0.16	0.15			
PAY_AMT6	0.22	-0.002	-0.054	-0.006	0.019	-0.06	-0.037	-0.037	-0.027	-0.023	-0.026	0.18	0.17	0.18	0.18	0.16	0.12	0.19	0.16	0.16	0.16	0.15		
DEFAULT	-0.15	-0.04	0.049	-0.027	0.014	0.33	0.26	0.24	0.22	0.2	0.19	-0.019	-0.014	-0.013	-0.0095	0.0062	0.0053	-0.074	-0.058	-0.056	-0.057	-0.056	-0.054	