Using Machine Learning and Credit Card Data to Predict Customer Attrition

I. Introduction

The credit card market contains a very large number of service providers around the globe and is therefore a highly competitive market. The increasing popularity of the internet and online banking have increased customer credit card use by a great deal [1]. This brings more opportunities for service providers, but also more problems. One of the main challenges is customer attrition (CA, aka customer churn), which in simple terms means the loss of customers over a certain period of time [2]. CA is continuously explored by organisations however, investigating existing customer behaviour can be a very costly and time-consuming process for organisations [2]–[5]. It can also reduce profitability as retaining customers is cheaper than acquiring new ones [6]. It is therefore considered very important to be able to distinguish customers who are likely to leave their current bank and move to another [7], [8].

For this reason, this investigation aims to use machine learning (ML) methods to predict CA. It also aims to pinpoint the reasons behind CA. The solution presented provides a feasible and efficient way to identify customers who are likely to cancel their accounts. The ML methods used for feature selection (FS) were Logistic Regression (LR) and Decision Tree (DT). The ML methods used for classification were k-Nearest Neighbour (kNN) and Support Vector Machine (SVM). The details of these methods are explained in the methods section of the report. The results showed that 10 features contributed the most to CA (most notably Total Transaction Count and Total Transaction Amount). The best trained ML model was a tuned SVM model, with parameters set to: 10 (C value), 0.1 (gamma value), rbf (kernel) and 2 (degree). The best model gave an accuracy of 94%.

II. Preliminary Analysis and Preprocessing

A. Data Set

The data set used was obtained from Kaggle. It is a data set that is used to analyse credit card CA and to model credit card churning. The data set contains 21 columns and 10127 samples (1 sample equals 1 customer). The columns include the ID column (CLIENTNUM), the class column (Attrition_Flag) and 19 features (Table 1.). In the class column 8500 samples were classed as Existing_Customer (binary flag 0) and 1627 samples were classed as Attrited_Customer (binary flag 1). This tells us that the data set is unbalanced. Therefore, it is crucial that the data is stratified when splitting into training and testing sets and when doing cross validation. The features are made up of demographic and credit card utilisation data.

B. Data Preprocessing

In ML, preprocessing the data is essential for building successful models. The following steps were performed to clean and transform the data for efficient and effective ML classification:

1) Removing Irrelevant Data: The data was checked for nulls and duplicates, and the CLIENTNUM column was removed. The categories in each categorical feature were inspected and as a result, all samples containing an 'Unknown' category were removed.

2) Tranforming Data: Ordinal encoding was performed on the features: Education_Level (uneducated = 0, college = 1, high school = 2, graduate = 3, post-graduate = 4, doctorate = 5), Income_Category (<$40K = 0, $40K - 60K = 1, $60K - $80K = 2, $80K - $120K = 3, >$120K = 4) and Card_Category (Blue = 0, Silver = 1, Gold = 2, Platinum = 3). One hot encoding was performed on the feature: Marital_Status. Binary encoding was performed on the feature Gender (Male = 1 and Female = 0) and the class Attrition_Flag (as stated earlier).

C. Data Visualisation

The data was then visualised in different ways to gain a better understanding of the data and to help select appropriate ML algorithms. Figure 1 shows each feature plotted as a histogram. The distribution of the data for each feature, in Figure 1, informed us of features that displayed a Gaussian (normal) distribution. This was important when selecting the standardising technique. This is because some standardising techniques are more effective when the data are normally distributed, and others are more effective when the data are not normally distributed and contain more outliers. From Figure 1, it was observed that half of the numerical features display a normal distribution whilst the other half did not. This presented a dilemma in choosing a standardisation technique. The standardisation technique chosen for the classification algorithms scaled the features to have a mean of zero and a unit variance. This was chosen as the SVM algorithm holds this assumption (that the features are centred around zero and have a unit variance), whilst the kNN algorithm does not and is effective with any standardisation technique.

Figure 2. shows a plot of the correlations of each feature, where the colour matches the strength of the correlation. This plot helped to inform feature selection and the ML methods chosen. This is because some ML algorithms have the assumption that all features are independent of each other. For feature selection, this can mean that when two features are highly correlated one of them is likely to be redundant and should be removed as it will affect the accuracy of these algorithms. However, there are some ML algorithms that do not have the independency assumption and for these it is ok to use features that are correlated. Overall, the correlations plot suggests that most of the features are not highly correlated. Some features showed high correlations and were noted down:

- Total_Trans_Amt and Total_Trans_Ct
- Avg_Open_To_Buy and Credit_Limit
- Months_on_book and Customer_Age
- Income_Category and Gender
- Single and Married
- Avg_Utilization_Ratio and Avg_Open_To_Buy
- Avg_Utilization_Ratio and Credit_Limit

TABLE I.

LIST OF FEATURES IN THE ORIGINAL DATA SET

Feature Name	Feature Description (categories in brackets)
Client Number	Unique identifier for the customer holding the account
Customer Age	Demographic variable, in Years
Gender	Demographic variable (M = male and F = female)
Number of Dependents	Demographic variable (0, 1, 2, 3, 4 and 5)
Educational Level	Demographic variable (uneducated, college, high school, graduate, post-graduate, doctorate and unknown)
Marital Status	Demographic variable (single, married, divorced and unknown)
Annual Income Category	Demographic variable (<$40K, $40K - 60K, $60K - $80K, $80K - $120K, >$120K and unknown)
Card Category	Type of card (Blue, Silver, Gold and Platinum)
Months on book	Period of relationship with bank
Total Relationship Count	Total number of cards held by the customer (1, 2, 3, 4, 5 and 6)
Months Inactive	Number of months inactive in the last 12 months (0, 1, 2, 3, 4, 5 and 6)
Contacts Count	Number of contacts in the last 12 months (0, 1, 2, 3, 4, 5 and 6)
Credit Limit	On the credit card
Total Revolving Balance	On the credit card
Open to Buy Credit Line	Average of last 12 months
Change in Transaction Amount	Q4 over Q1
Total Transaction Amount	Last 12 months
Total Transaction Count	Last 12 months
Change in Transaction Count	Q4 over Q1
Average Card Utilization Ratio	

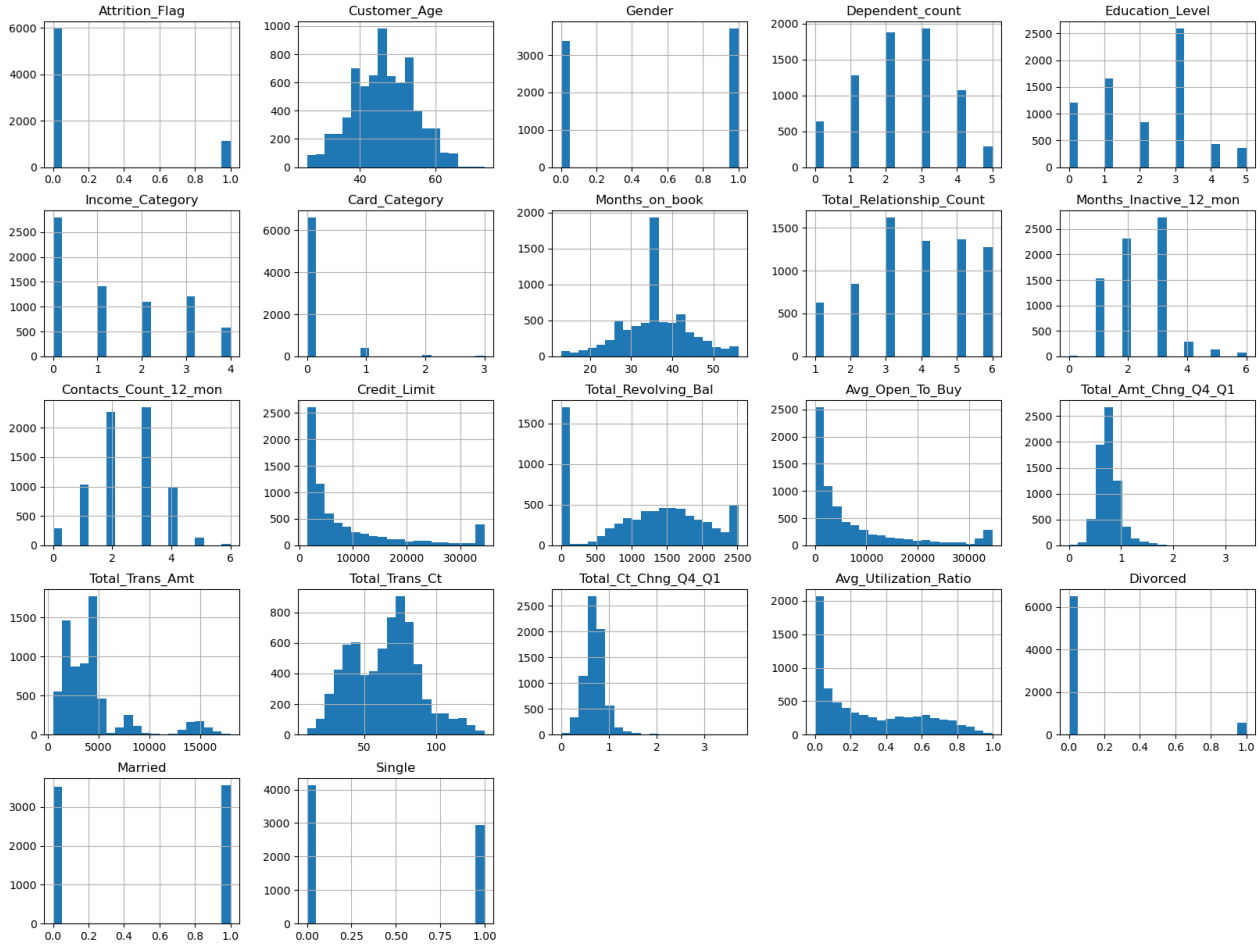

Fig. 1. Plot of histograms for each feature and the class

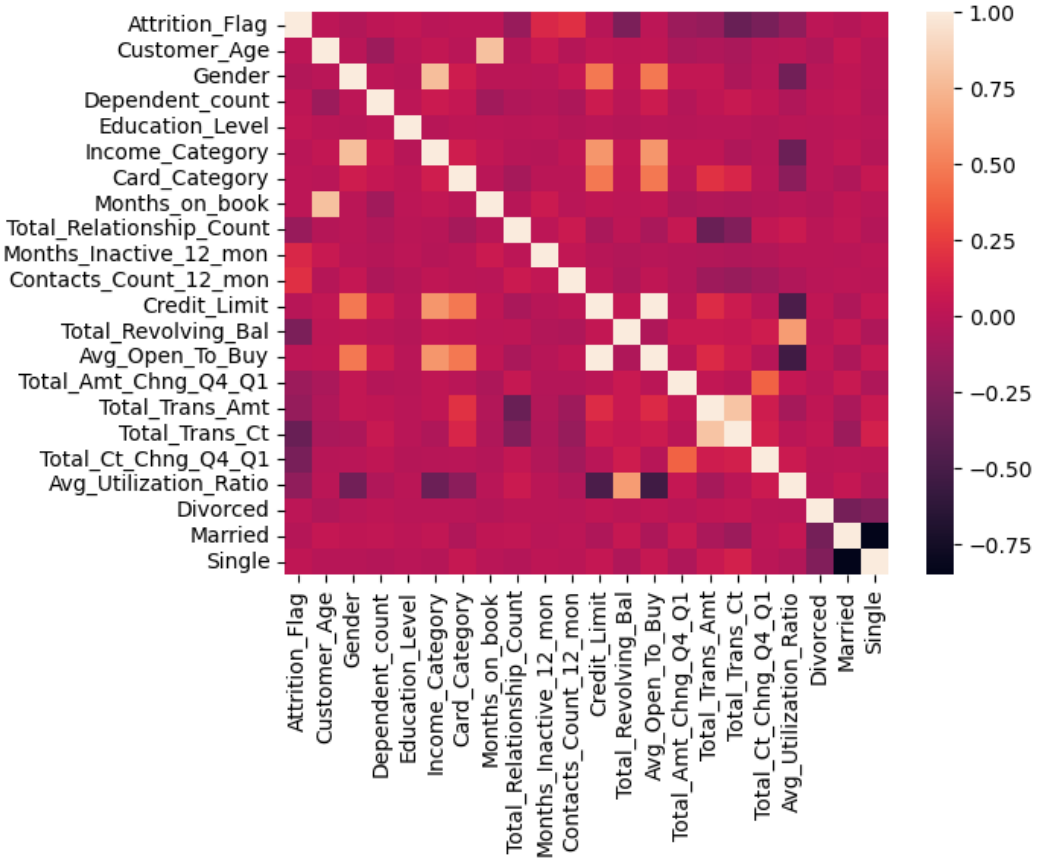

Fig. 2. Plot of correlations for each feature and the class

III. Methods

As described in the introduction, the aim of this work is to find out the reasons behind CA (this can be taken from the main features that contribute to the model) and to train a ML model that can predict CA (i.e., with new data can we predict who is likely to close their credit card account?). A diagrammatic representation of the proposed ML pipeline is given in Figure 3. The following explains each ML algorithm used and why.

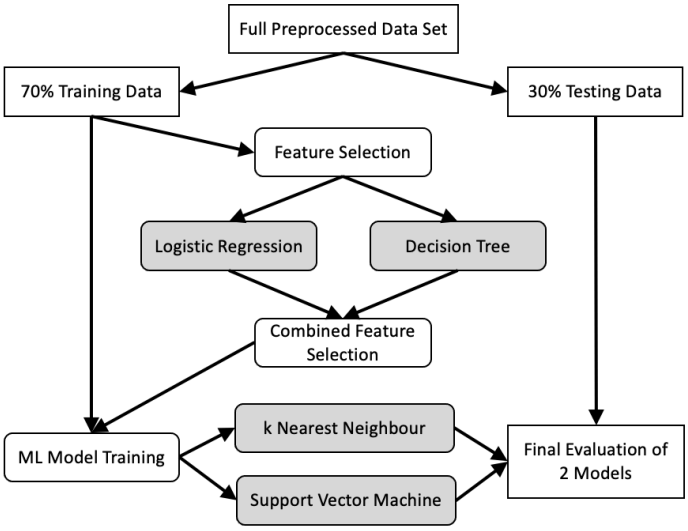

Fig. 3. Diagram of proposed machine learning pipeline. The specific algorithms used are highlighted in grey.

A. Feature Selection Algorithms

Identifying the features that contribute most to model prediction is fundamental to ML. FS not only simplifies the model, but it also helps to reduce processing requirements (which helps to shorten the training time). FS is also key in avoiding the curse of high dimensionality. The following two ML algorithms were chosen because of their capabilities to distinguish features by their level of importance in predicting.

1) Logistc Regression: LR is a type of linear model that is used to analyse the relationship between one or more predictor variables (features) and a binary outcome (class). The assumption of the model is that the predictor variables can be modelled using a logistic function. This is an S-shaped curve that ranges from 0 to 1. The threshold value is usually set to 0.5. Therefore it can be used as a classifier, as if the probability is greater than 0.5 the outcome is classed as 1 and if the probability is less than 0.5 the outcome is classed as 0. For feature selection the coefficients are the most important aspect of this model. To get relevant coefficients min max scaling is necessary. Min max scaling is a standaridising technique that transforms the data of each feature to a given range (in this case it was between 0 and 1). This enables the coefficient of each feature to be compared against each other and ranked from best to worst. The features with the best coefficients are the ones that are the most important when predicting the class and the features with the worst coefficients should be removed.

2) Decision Tree: A DT is a non-linear algortithm that can also be used for feature selection. It works by using a

Greedy approach to select the most important features and then partitions the data set into branch-like segments that are optimised for the separation of the classes. Each branch represents a decision and each leaf represents a predicted class. The tree grows until specifc critera are met, e.g., maximum depth or a threshold level of accuracy. It is a popular ML algorithm because it is easy to visualise and interpret. However, for data sets with lots of features the tree can grow exponentially. This results in a lot of variance and often the Greedy DT will not be optimised. For FS the most important aspect of the DT algorithm is the calculated importance score of each feature. This is usually done by using entropy theory. Entropy is essentially a measure of impurity of that feature. A high entropy means that the feature is impure and is distributed evenly between classes. A low entropy means the the feature is pure and that most of the samples belong to a particular class. The aim is to find the feature that has the lowest entropy. Specifically the DT measures the information gain of a feature (the entropy difference between the original and weighted average of subsets). The higher the information gain the more important the feature.

B. Classification Algorithms

ML classification algorithms are supervised learning algorithms that are deployed to learn patterns in a data set according to the class labels given. Then using unseen data, they have the ability to predict what class the new samples are from. The following two ML classification algorithms were chosen due to their popularity and wide-ranging applications.

1) k-Nearest Neighbour: kNN is non-linear algorithm with the idea that similar samples tend to be grouped together. It is an instance based (lazy) classifier, so works only when samples are added or queried against the rest of the data. This means that when an unknown sample is queried the distance between the sample and the k nearest samples is calculated. The predicted classification is then determined according k-nearest neighbours of the sample. When training a kNN there are two key parameter that influence the quality of the model: The distance metric used and the value of k. There are a few distance metrics that can be chosen, the most popular being the Euclidean distance. The value of k is difficult to select as if it is too small, the model is sensitive to noise points and is at risk of being overfitted. Whilst if k is too large, the neighborhood may include points from other classes and it is at risk of being underfitted. In order to build a robust kNN model different parameter values must be tried and tuned.

2) Support Vector Machine: SVM is both a linear and non-linear classification algorithm. The aim of SVM is to find a hyperplane that maximises the margin (the distance between the closest samples and the hyperplane) between classes. SVM represents the data points as vectors in a higher-dimensional space and it represents the hyperplane as a combination of these vectors. The algorithm works by solving an optimisation problem that seeks to minimise the classification error whilst maximising the margin. Different kernel functions are used to map the data points into higher-dimensional spaces. SVM is generally one of the more popular ML algorithms as it can produce very robust and reliable results. It also gives you a great deal of flexibility in controlling the model complexity and error. However, similarly to kNN, the parameters that give you control over the algorithm must be trialled and tuned.

C. Evaluation Measures

The following metrics were chosen to evaluate the models during training and in the final evaluation:

1) Confusion Matrix: This displays the True Positives (TP, where the model correctly predicts the positive class), True Negatives (TN, where the model correctly predicts the negative class), False Positives (FP, where the model incorrectly predicts the positive class) and False Negatives (FN, where the model incorrectly predicts the negative class).

2) Accuracy: this is the proportion of correctly classified instances out of all the instances in the data set.

3) Precision: This is the proportion of TPs out of all the predicted positives.

4) Recall: This is the proportion of TPs out of all the actual positive instances in the data set.

5) F1 score: The is the harmonic mean of precision and recall.

IV. Experiments and Results

The following describes the experiments and procedures that were carried out in the ML pipeline (Figure 3). To avoid complexity some experiments are not detailed in the diagram but are detailed below. For example, a comparison between training data and feature selected training data for both kNN and SVM model training was performed. In addition, specific and multi-hyperparameter tuning for kNN and SVM model training were performed.

A. Data Splitting

After preprocessing, the data was split into 70% training data (now referred to as just training data) and 30% testing data (now referred to as just testing data). As the data was unbalanced the split was stratified. The training data was used in FS and classification model training. The testing data was left out until the end where it was used to test the best classification models and evaluate their effectiveness at predicting the class. Data splitting is best practice in ML as having unseen data to evaluate the final model is essential. For FS, classification model training and final model testing 10-fold stratified cross validations were performed. Cross validation is when the data is split into k partitions (in this case 10) and training is done on k-1 partitions whilst the last partition is left to test the trained model. This is repeated k times, and the results are averaged. Cross validation is also best practice in ML as it helps to reduce bias, increase model robustness, and increase model accuracy.

B. Feature Selection

1) Individual Feature Selection Algorithms: Before performing the LR, the data was min max scaled. This enabled the coefficient of each feature to be compared against each other for selection purposes. The parameters selected for the LR were all left as default except for the maximum number of itereations. This was set to 8000 as it needed to be high enough for the model to run completely. The parameters selected for the DT were also mostly left as default. This was

done to ensure each feature had a calculated importance score that could be used for evaluating whether to keep it or not. The 'criterion' parameter was set to 'entropy' to ensure the features were being compared against their information gain scores.

2) Combining Feature Selection Algorithms: The results of the two FS algorithms were combined to produce a final set of most important features. They were combined by min max scaling the individual feature score lists (to put them on the same scale) and then calculating the averages. A new data set was then created that contained only features with a score above the median of the scores (Table 2). This new data set was then used to train two classification models and these were compared against the data set with the full set of features.

TABLE II.
FEATURES OF HIGHEST PREDICTIVE IMPORTANCE

Feature Name	Score
Total Transaction Count	1.00
Total Transaction Amount	0.47
Change in Transaction Count	0.38
Total Revolving Balance	0.36
Total Relationship Count	0.22
Change in Transaction Amount	0.16
Months inactive	0.15
Contacts Count	0.14
Customer Age	0.08
Gender	0.04

C. ML Model Training

Before training the classification models the training data and feature selected training data were standardised to have a mean of zero and a unit variance.

1) Initial k-Nearest Neighbour: Firslty, an initial kNN model was trained on the data set with the full features and on the data set with the selected features. The results of these two models were then compared. Most of the parameters for this initial kNN were left as default. the value of k was set to 3 and the 'weights' parameter was set to 'distance' instead of 'uniform' (this ensured that the neighbour points closer to the query point had a greater influence than the neighbours that were further away). The results of these two models are displayed in Table 3. The kNN with the feature selected training data set performed better (or the same) in every metric compared to the the kNN with the full training data set. This highlighted the curse of high dimensionality and demonstrated the power that reducing the number of features can have on model performance. The metrics that showed the largest improvements were class 1 Recall (66% vs 50%) and class 1 F1 (73% vs 60%). This means the model increased the number of positive instances it predicted whilst also reducing the number of false negatives.

1) k-Nearest Neighbour Tuning: The data set selected for tuning was the feature selected training data set. Firstly, the k parameter was tuned individually to find a range of k values to further investigate with other parameters. Figure 4 displays a graph that shows the accuracy score against the k value (number of neighbours). From this graph the k value range of 9 to 15 was chosen to investigate further. For the multi-hyperparameter tuning the following parameter grid was constructed:

- k values: 9 - 15
- Weights: uniform and distance
- Metric: Minkowski, Euclidean and Manhattan

This resulted in a total of 420 fits with the best parameters being:

- k value: 10
- Weights: distance
- Metric: Manhattan

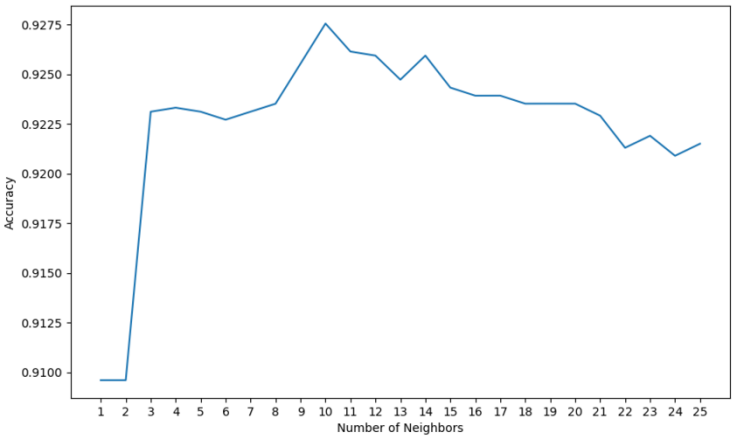

Fig. 4. Graph of accuracy score against the k value (number of neighbours)

2) Initial Support Vector Machine: Similarly to the kNN model, the initial SVM model was trained on the data set with the full features and on the data set with the selected features. The results of these two models were then compared. The only parameter that was set was the 'kernel' was set to 'linear'. The results of these two models are displayed in Table 3. The SVM with the feature selected training data set

TABLE III.
INITIAL MACHINE LEARNING MODEL TRAINING RESULTS

Classifier (Data Set Used)	Accuracy (%)	Precision (%)		Recall (%)		F1 (%)	
		0	1	0	1	0	1
kNN (Full Training Data Set)	90	91	75	97	50	94	60
kNN (Feature Selected Training Data Set)	92	94	81	97	66	96	73
SVM (Full Training Data Set)	90	92	77	97	54	94	64
SVM (Feature Selected Training Data Set)	90	92	76	97	53	94	63

performed almost exactly same in every metric compared to the SVM with the full training data. This demonstrated how FS can help to reduce the processing requirement whilst maintaining the same model performance. Although the two SVM models performed very similarly, the model using the feature selected training data is clearly a much better model as it only uses 10 features compared to the full training data set which uses 19 features.

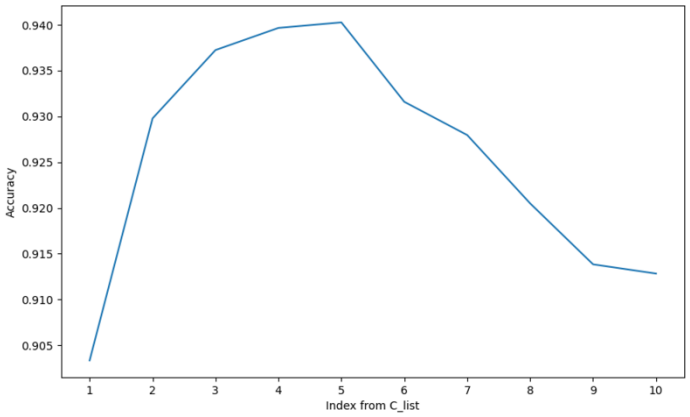

Fig. 5. Graph of accuracy score against the index in the list of C values

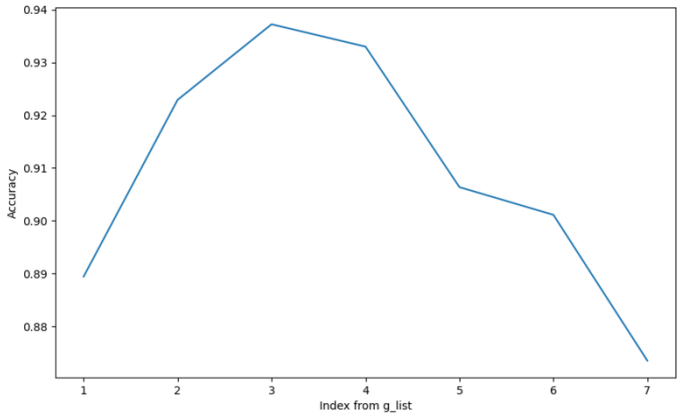

Fig. 6. Graph of accuracy score against the index in the list of gamma values

3) Support Vector Machine Tuning: The data set selected for tuning was the feature selected training data set. Firstly, the C and gamma parameters were tuned individually in the same way the k parameter was tuned (to find a range of more focussed values to further investigate with other parameters). The individual C parameters tuned were (0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, and 5000), whilst the individual gamma parameters tune were (1, 0.5, 0.1, 0.05, 0.01, 0.005 and 0.001). Figure 5 and 6 display the graphs of the accuracy score against the C and gamma parameters respectively. From Figure 5 the C values 1, 5 and 10 were chosen to investigate further. From Figure 6 the gamma values 0.1 and 0.05 were chosen to investigate further. For the multi-hyperparameter tuning the following parameter grid was constructed:

- C values: 1, 5 and 10
- Gamma: 0.1 and 0.05
- Kernel: linear, rbf and poly
- Degree: 2, 3, 4 and 5

This resulted in a total of 720 fits with the best parameters being:

- C values: 10
- Gamma value: 0.1
- Kernel: rbf
- Degree: 2

D. Final Model Evaluation

Before performing the final model evaluation, the testing data was standardised and updated to have just the 10 selected features (so that it matched the training data set).

1) k-Nearest Neighbour: Figure 7 shows the results of the tuned kNN model on the test data set. All the evaluation metrics were very similar to the initial kNN using the feature selected training data set. A reason that it is not noticeably better than the initial kNN, is that it is being tested on unseen data. When this is taken into account the model performs very well. The model is also visibly better at predicting class 0 than class 1. This is likely due to the unbalanced nature of the data and more samples of class 1 may be required to improve the models performance.

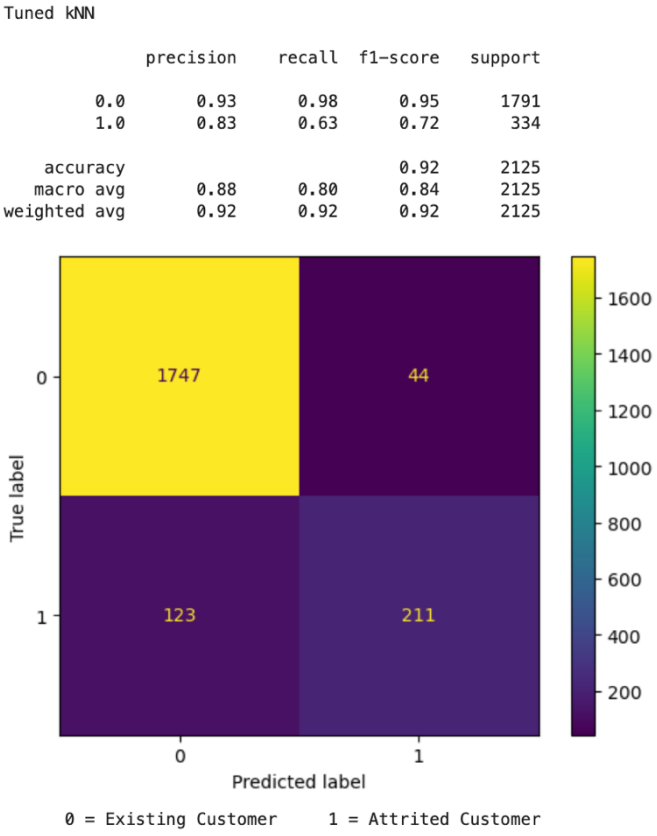

Fig. 7. Results from tuned kNN model

1) Support Vector Machine: Figure 8 shows the results of the tuned SVM model on the test data set. This model performed much better than the initial SVM using the feature selected training data set as all the evaluation metrics are higher (or the same). It also performed better than the tuned kNN model. It was the model with the highest accuracy (94%) and consistently high metrics for class 1 (Precision: 85%, Recall: 78% and F1: 81%). Although these were still

not as high as the metrics for class 0 (Precision: 96%, Recall: 97% and F1: 97%), likely due to the data being unbalanced. The confusion matrix plot shows that the model had a high TP and TN.

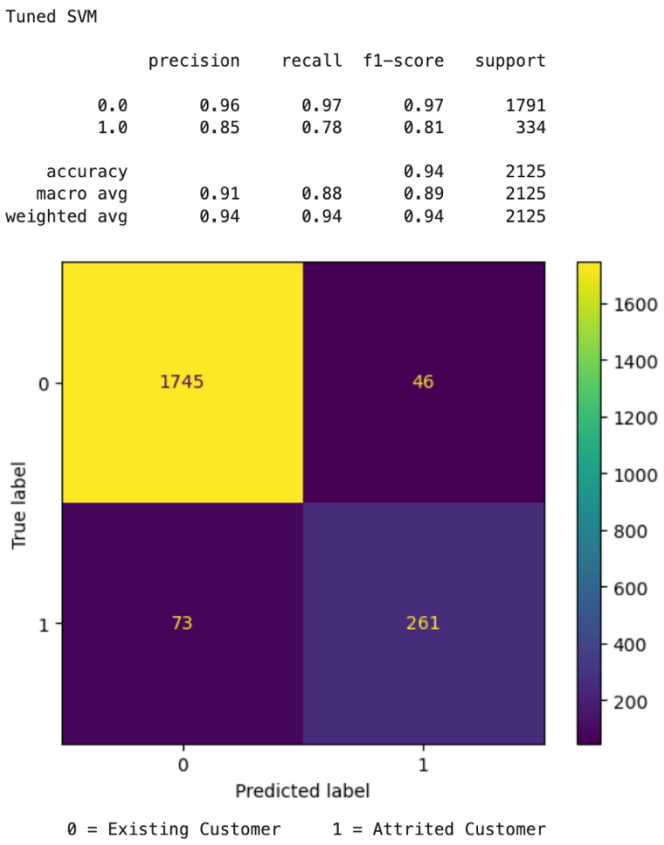

Fig. 8. Results from tuned SVM model

V. Conclusion and Reflection

The objective of this investigation was to identify why customers were cancelling their accounts and to build a ML model to predict which customers were most likely to cancel their accounts. Firstly, the feature selection stage of the ML pipeline highlighted the reasons behind CA (Table 2). This is because it calculated the features that are most important for predicting the class (CA or no CA). Therefore, changes in these features (especially the ones with the highest score value) can help indicate, to any organisation, whether a customer is likely to cancel their account or not. Secondly, the best ML model at predicting the class was the tuned SVM model. This is because it had the highest accuracy (94%, Figure 8) and was the best at correctly predicting attrited customers. This model presents an effective resource for organisations to use to predict CA.

Nonetheless, a few key points are worth raising and a few ML processes were not included in this model and may be beneficial to use in further iterations. Firstly, a key point that has been raised multiple times is that the data was unbalanced. The final model had a high accuracy but because the data was unbalanced there is a risk that it may have been overfitted. Collecting more data for class 1 (Attrited_Customer) may be necessary for future iterations as it can reduce the bias of the model and the risk of it being overfitted. Secondly, although the FS algorithms produced great results (highlighting the most important features) their reliability is uncertain. This is because the FS algorithms were not tuned. In further investigations tuning the FS algorithms may help to provide more accurate and robust results. Thirdly, although the final model had the highest objective accuracy (and other metrics) there was no confidence associated with it. In future investigations of a similar nature, performing t-tests to compare results may help to provide statistical confidence in the results.

In terms of ML processes, an alternative method of FS is Dimension Reduction (DR). Essentially it is a way of transforming data from a high-dimensional space into a lower-dimensional space whilst retaining the meaningful characteristics of the data. A popular DR algorithm is Principal Component Analysis (PCA). Using this algorithm and comparing its results to those of the FS algorithms may have helped to provide alternative insightful results. However, PCA holds the assumption that there are correlations between features. From the correlations plot (Figure 2), it was seen that most of the features were not correlated and therefore PCA was not chosen. Furthermore, a typical technique used to enhance ML models is Ensemble learning. This is a technique that often obtains a better predictive performance (compared to using ML models in isolation) by combining multiple different ML models. However, because the final model performed very well, Ensemble learning was not deemed necessary as it may have increased the risk of overfitting.

In summary, the final solution (tuned SVM) presents an easy mechanism for predicting CA that is not labour intensive. The most important features found also provide a direction and focus for organisations when trying to pinpoint the reasons behind CA. The results also align with the critical need to determine which customers are likely to churn [7], [8]. Thus enabling organisations to save time, cut costs and increase profits [2]–[6], [8].

References

[1] N. X. Hong and L. Yi, “Standing at the crossroads-credit card,” *Reporters’ Notes*, vol. 5, pp. 41–43, 2020.

[2] K. D. ÜNLÜ, “Predicting credit card customer churn using support vector machine based on Bayesian optimization,” *Communications Faculty Of Science University of Ankara Series A1Mathematics and Statistics*, vol. 70, no. 2, pp. 827–836, Dec. 2021, doi: 10.31801/cfsuasmas.899206.

[3] D. AL-Najjar, N. Al-Rousan, and H. AL-Najjar, “Machine Learning to Develop Credit Card Customer Churn Prediction,” *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 17, no. 4, pp. 1529–1542, Dec. 2022, doi: 10.3390/jtaer17040077.

[4] X. Miao and H. Wang, “Customer Churn Prediction on Credit Card Services using Random Forest Method,” in *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development*, 2022, pp. 649–656.

[5] M. Rahman and V. Kumar, “Machine Learning Based Customer Churn Prediction in Banking,” in *Proceedings of the 4th International Conference on Electronics, Communication and Aerospace*

Technology, ICECA 2020, Nov. 2020, pp. 1196–1201. doi: 10.1109/ICECA49313.2020.9297529.

[6] C. A. Rico-Poveda and I. Galpin, "Forecasting Credit Card Attrition using Machine Learning Models," in *ICAIW 2020: Workshops at the Third International Conference on Applied Informatics*, 2020, pp. 120–134. [Online]. Available: http://ceur-ws.org

[7] E. Domingos, B. Ojeme, and O. Daramola, "Experimental analysis of hyperparameters for deep learning-based churn prediction in the banking sector," *Computation*, vol. 9, no. 34, Mar. 2021, doi: 10.3390/computation9030034.

[8] J. Hadden, A. Tiwari, R. Roy, and D. Ruta, "Computer assisted customer churn management: State-of-the-art and future trends," *Comput Oper Res*, vol. 34, no. 10, pp. 2902–2917, Oct. 2007, doi: 10.1016/j.cor.2005.11.007.