BANK CUSTOMER CHURN

Data Mining and Text Analysis – Summative Assessment
Monday 26 June 2023

Executive Summary

Problem statement

Recently, the bank has seen an increase in the rate of customer churn. Churn results in lowered business and competitors are likely to benefit from this loss. Currently, the bank has not found any useful ways to identify those at risk of attrition so that we can encourage them to remain with us. Therefore, it is important that a machine learning framework is created to predict the indicative characteristics of a customer who is at risk of becoming attrited.

Customer churn profiles

The bank has in-depth data on thousands of its customers. This richness of the data has made it possible to utilise machine learning algorithms within the Python framework to build profiles using specific customer attributes to determine which combination was best at predicting whether a customer was likely to churn.

Before any building of profiles could commence, appropriate pre-processing of the data was conducted to allow transform the data into a useable form. An exploratory Correlation map of all the numeric attributes was created, and this allowed for a visual representation of which indicators were directly related. From here, appropriate customer profiles were built; those that would contain likely indicators of customer attrition.

The three profiles were created and input to a Random Forest Classifier. 10,000 customer entries were used to train each model on what patterns to observe and the remaining 2001 customer entries were used as test cases to see if the trained model could correctly predict their attrition status.

Of the three profiles, Profile 1 proved the best: it contained the attributes showing the total amount spent in transactions and the total number of transactions by a customer, their card category, their credit limit, their age and their gender. This produced a prediction that was within the acceptable bounds of error with 99.75% accuracy and 100% precision when compared to the actual status of the customer.

This Profile is now recommended to be used as a framework by the bank to help predict customers that may be at risk of becoming attrited, such that they can be targeted with promotions and discounts. Future work would involve further testing of the model and tuning to ensure that its predictions remain of a high quality.

Fraud prediction

Attempts to predict fraud using geographical location (Country) were unsuccessful. Future work would involve further investigation into a more complex formulation of a profile to predict fraudulent activity.

Table of Contents

Task 1

As requested by the manager of the bank, in the context of the rate of customer churn, analyses have been conducted to produce suggestions to two of the solutions that are of stakeholders' interest. These solutions are Profile (customer profiles that are likely to become attrited customers), and Geographical location (if customer location is an appropriate predictor for the likelihood of fraudulent activity).

The raw customer information BankChurners training and test files were pre-processed, mined and analysed in Python using the NumPy, Pandas, Matplotlib, Seaborn and Scikit-learn packages to produce outputs [1].

General pre-processing

The raw dataset contained thousands of customers with 22 attributes, so the data was read into Pandas DataFrames, to better understand the shape of them and to enable appropriate pre-processing before analysis. Appendix A lists these attributes, whether they are discrete or continuous, if the attribute contains unknown or null values. and if the field needed to become a categorical dummy/indicator value, as was done using the pandas.get_dummies method [2]. This method creates a new attribute for each possible outcome in the DataFrame, and labels it 0 or 1 depending on if the customer satisfies that particular outcome. This is known as one-hot encoding [3]. The code snippet for this is shown in Appendix B.

The first check that was done on the dataset was a check for any missing values; using the .isnull(.sum() methods [4, 5]. This returned that there were no missing values, unlikely for a large dataset. This led to further investigation of the Balances, using the .value_counts() method [6] to get percentages of each value. The code and output for this are shown in Appendix C, which gave a Balance of $0.00 occurring in 36.17% and 36.83% of customer balances in the training set the testing set respectively. The likelihood is the zeroes correspond to a large proportion of missing values. Justification of replacing the values from literature was not found – and Farrukh et al., argue that replacing nulls with 0s is effective [7].

A) Profile

To build profiles that may indicate customer churn, a Correlation map was created using Seaborn and Matplotlib [8, 9]. Since the map cannot take strings, some attributes were ranked numerically; for example, Card_Category had Blue, Silver, Gold and Platinum replaced with 1 to 4. The replacement was done using the pandas .replace() method [10] on the DataFrame. For Income_Category, each was replaced with the midpoint of that range, while $120K + was set at 150. All unknowns in Income_Category and Education_Level were set as the midpoint of possible values. For attributes that could not be ranked numerically, such as Geography, they were omitted from the map. The map and code are shown in Appendix D.

Ambiguity in the data mean it was not immediately obvious whether Attrition_Flag or Exited was the correct indicator on whether a customer had become attrited. For this reason, profiles were tested against both individually.

As the Correlation map only shows bivariate correlations [11], attributes that showed correlations with the relevant indicators were then selected, and attributes that correlated with those were grouped together. The profiles were built with the attributes shown in Table 1. Individual DataFrames were built for each profile [12]. DataFrames for Attrition_Flag and Exited were also created. These are shown in Appendix E.

Since both Attrition_Flag and Exited are discrete, the Scikit-learn Random Forest Classifier [13] was used with n_estimators = 1000. The Random Forest Classifier takes NumPy array as inputs, so the DataFrames were converted. A model was created for each profile and each target, meaning six in total. The code for the model for Profile 1 is shown in Appendix F1.

Table 1: Attributes of the profiles created for Attrition_Flag and Exited prediction.

Profile	**Attributes**
Profile 1	Total_Trans_Amt Total_Trans_Ct Card_Category Credit_Limit Age Gender
Profile 2	Balance NumOfProducts IsActiveMember
Profile 3	Gender Credit_Limit Income_Category Age Card_Category HasCrCard

To evaluate the success of each model, various measures were used. In both datasets there are 12,001 customers, of which 1627 have the Attrition_Flag attribute and 2437 have the Exited attribute. When the normal approximation of the Binomial distribution is applied with a two-tailed 95% confidence interval [14] (Equation 1), it gives a confidence range of $\pm$ 0.61% and $\pm$ 0.72% for Attrition_Flag and Exited respectively. Hence, if the Mean Absolute Error (MAE) of the predicted values is less than these, it suggests a successful prediction.

$$p \pm z_{1-\propto/2}\sqrt{\frac{p(1-p)}{n}}$$

where p is proportion of interest
n is sample size
$\propto$ is confidence level
$z_{1-\propto/2}$ is 1.96 for 95% confidence.

Equation 1: Normal approximation for a binomial proportion [14].

Further indicators of the success of the models are Accuracy, Precision, Recall and F1 Score (equations below). These can be calculated through the Confusion Matrix which gives the proportion of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) predicted by the model. This matrix is output using the Scikit-learn confusion_matrix() method [15] (code in Appendix F1). Accuracy, Precision, Recall and F1 Score are as follows [16]:

- **Accuracy:** proportion of true predictions against all predictions
- **Precision:** proportion of true positives against all positives
- **Recall:** proportion of true positives that are correctly classified
- **F1 Score:** harmonic mean of Precision and Recall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 2: Accuracy of a prediction.

$$Precision = \frac{TP}{TP + FP}$$

Equation 3: Precision of a prediction.

$$Recall = \frac{TP}{TP + FN}$$

Equation 4: Recall of a prediction.

$$F1\ Score = \frac{2TP}{2TP + FP + FN}$$

Equation 5: F1 Score of a prediction.

The outputs are shown in Appendix F2, while full results are shown below. Table 2 shows the MAE, Table 3 shows TP, FP, TN and FN, and Table 4 shows the Accuracy, Precision, Recall and F1 Score for each.

Table 2: MAE for Profiles 1-3 for Attrition_Flag and Exited.

	MAE	
	Attrition_Flag	**Exited**
Profile 1	0.2499%	3.5982%
Profile 2	8.5457%	7.7461%
Profile 3	3.9980%	4.74763%

Table 3: TP, FP, TN and FN for Profiles 1-3 for Attrition_Flag and Exited.

	Attrition_Flag				Exited			
	TP	**FP**	**TN**	**FN**	**TP**	**FP**	**TN**	**FN**
Profile 1	393	0	1603	5	343	15	1586	57
Profile 2	237	10	1593	161	282	37	1564	118
Profile 3	328	10	1593	70	327	22	1579	73

Table 4: Accuracy, Precision, Recall and F1 Scores for Profiles 1-3 for Attrition_Flag and Exited.

	Attrition_Flag				Exited			
	Accuracy	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Profile 1	99.750%	100%	98.744%	0.994	96.402%	95.810%	85.750%	0.905
Profile 2	91.454%	95.951%	59.548%	0.735	92.254%	88.401%	70.500%	0.784
Profile 3	96.002%	97.041%	82.412%	0.891	95.252%	93.696%	81.750%	0.873

By comparing the MAE to the confidence intervals for the Attrition_Flag and Exited, the only model that performed within that range was Profile 1 on the Attrition_Flag value. This may be expected, as the attributes in Profile 1 have the strongest correlation to the Attrition_Flag (see Appendix D1). Conversely, Profile 2 has the largest MAEs at 8.5457% and 7.7461% for Attrition_Flag and Exited respectively. This is expected because Profile 2 had the fewest attributes, which meant its models had fewer features than the other two profiles. The more relevant features are trained into the model, the better quality its predictions [17].

The number of FP and FN are noticeably higher in Profile 2 than in Profile 1 and Profile 3. Profile 2 also performs poorly in terms of Recall, at 59.548% and 70.500% for Attrition_Flag and Exited respectively. Its F1 Scores were consequently low. This further proved that fewer attributes produce unreliable predictions for customer churn.

It is noteworthy that Profile 1 produced 0 False Positive predictions when predicting Attrition_Flag, hence giving it 100% Precision and 99.750% Accuracy. Its Recall was 98.744% with an F1 Score of 0.994, which makes it the best performing profile. It is recommended that the bank use Profile 1 to help predict whether a customer is at risk of becoming an Attrited Customer.

No tuning or other overfitting prevention methods on the model was used in this evaluation. This would likely have improved its predictions, since an n_estimators = 1000 was set as a rule of thumb. With this, there is a risk of overfitting the data, which also was not mitigated against. A future iteration of profile building would involve tuning the model to produce better predictions.

B) Geographical location

The bank hopes to understand if a customer's account location can be used as a good indicator of whether it is likely to be fraudulent. The is_fraud attribute is already a binary numeric indicator, so pre-processing methods were not required on it.

The Geography attribute has three distinct values; France, Germany and Spain, but since these are non-numeric, the Pandas get_dummies() method [2] was used, as previous. To determine whether the Geography attribute is a good indicator to fraudulent activity, three other attributes were selected as comparators: Card_Category, Gender and Tenure. As above, Card_Category and Gender were passed through the Pandas get_dummies() method [2] as they are also non-numeric discrete values. Tenure is continuous, so no pre-processing was needed.

In both datasets, there were 49 out of 12,001 customer accounts that had been flagged as fraudulent. When the normal approximation of the Binomial distribution is applied with a two-tailed 95% confidence interval (Equation 1), this gives a confidence range of $\pm$ 0.12% hence, a MAE of less than this would indicate successful prediction.

As in Part A), the Scikit-learn Random Forest Classifier [13] was used with n_estimators = 1000. The creation of the DataFrame profiles is shown in Appendix G1, training and testing of the model is shown in Appendix G2, and outputs are shown in Appendix G3.

Table 5: MAE for Geography, Card_Category, Gender and Tenure for is_fraud.

	Mean Absolute Error (%)	
Geography	0.0995%	0.0995%
Card_Category	0.0995%	0.0995%
Gender	0.0995%	0.0995%
Tenure	0.0995%	0.0995%

Table 6: TP, FP, TN and FN for Geography, Card_Category, Gender and Tenure for is_fraud.

	TP	FP	TN	FN
Geography	0	0	1999	2
Card_Category	0	0	1999	2
Gender	0	0	1999	2
Tenure	0	0	1999	2

Table 7: Accuracy, Precision, Recall and F1 Scores for Geography, Card_Category, Gender and Tenure for is_fraud.

	Accuracy	Precision	Recall	F1 Score
Geography	99.900%	Indivisible by 0	0.000%	0.00
Card_Category	99.900%	Indivisible by 0	0.000%	0.00
Gender	99.900%	Indivisible by 0	0.000%	0.00
Tenure	99.900%	Indivisible by 0	0.000%	0.00

What is immediately apparent is all attributes garnered the same predicted results; and upon closer inspection, each model predicted that none of the 2001 accounts in the test set were fraudulent accounts. The training set has 47 out of 10,000 accounts listed as fraudulent (0.47%) while the testing set has 2 out of 2001 which is 0.01%. Since there are only two in the test set, the accuracy seems high at 99.900%, but it had not managed to predict a fraudulent account.

In both the training and test sets fraud is so rare that using one attribute to predict appears unviable. Belkin et al., showed that incrementally increasing the number of features will create “near perfect” predictions [18]. Hence, it can be concluded that not only can the Geography attribute on its own not be used to predict the prevalence of fraud, no one attribute can do so. Sailusha et al., compare the Random Forest and the Adaboost algorithms on the same Kaggle BankChurners dataset and conclude that they both produce similar accuracies, but Random Forest produces better precision, recall and F1 Scores [19]. Thus, the Random Forest method is the correct route, but improvements in the feature setting is needed to produce reliable fraud detection.

Task 2

Python

Python has a strong reputation and is the "most preferred language for scientific computing" [20]. It is dynamically typed, which allows flexibility when dealing with datasets where there is no prior knowledge of types [21, 22]. Python's syntax is also simple and intuitive to use, when compared to other statically typed languages like Java [23].

Beyond this, the wealth of free packages available in Python libraries made processing of the dataset and presentation of outputs easy and manageable. A strong presence in online community forums such as Stack Overflow made information and troubleshooting accessible.

At times through the investigation, since the BankChurners csv files had 22 attributes and thousands of entries, understanding the data purely in the Python frameworks was challenging. Microsoft Excel was utilised in parallel to help contextualise the physical shape of the tables.

Trade-offs for Python's flexibility, however, do exist. Python is known as having slower performance than other languages due to machine referencing and querying to translate its simple syntax [24, 25, 26]. A solution is different interpreters like PyPy and the availability of extension libraries of Python to C/C++ to improve performance (hence the speed of NumPy [27]).

R vs Python

The statistical language R could have been used as an alternative to Python. R is a more complex compiler language which is strong in machine learning and statistics, suiting advanced programmers. It also performs faster than Python [28], while continuing to be heavily relied on in the academic/scientific world [29] and has a larger set of packages than Python. Since the BankChurners investigation was on a relatively small dataset not needing exceedingly complex statistical analysis and processing time rarely exceeded one minute, Python was still favoured over R in this case.

WEKA vs Python

WEKA is a powerful data mining/machine learning tool that run on the Java environment and provides a simple GUI front-end for less experienced programmers. It can also be coded through Java to enable the use of SQL databases [30]. WEKA, however, is only able to handle small datasets and has a low range of algorithms offered [31] (not necessarily a problem for this task). Furthermore, Mitrpanont et al., determined that in the context of machine learning algorithms, Python performs better at predicting outcomes along with higher precision and recall than WEKA in dialysis data analysis [32]. Since Python predicts outcomes better and is more flexible, Python is still favoured over WEKA – despite WEKA's Java Machine enabling faster performance with machine learning algorithms.

Task 3

Multicollinearity

A current issue that is being discussed widely in the machine learning and deep learning spaces is the multicollinearity problem and its impacts in various areas [33, 34, 35]. Chan et al., describe multicollinearity as the condition when two or more independent variables share a linear relationship – which can cause outputs from a model to be unreliable [36]. The presence of two (or more) attributes sharing a linear correlation creates difficulty in evaluating the marginal impact of each attribute. The likely outcome of this is overfitting to the training scenario and worsened generalisability.

Some of the discussed solutions include, perhaps intuitively, collecting more data, such that the model can better differentiate between the attributes. Consequently, there is a resource cost in collecting more (if further collection is possible). And the possibility of deteriorating of data quality and processing performance.

Further solutions such as variable selection methods which utilise heuristic algorithms to combine or disregard attributes in the model. This is achieved through incrementally adding attributes and then removing those with the highest residual sum of squares error [37]. This method comes with high computational cost, and the possibility of worsening the model in trying to mitigate multicollinearity.

Neural networks are known to be superior to statistical models [38], including in the context of multicollinearity [39]. Machine learning algorithms are a better solution because they uncover complex patterns and learn relationships dynamically. Artificial neural networks, however, are unable to extract variables [40], which means principal component analysis must be conducted first to select attributes.

These solutions to these problems have been validated by the scientific community, but the biggest conclusion is that as models become more complex and more reliable, the computational cost can become huge. Even with supercomputers available today, there is a large resource cost in creating a high-quality model to combat multicollinearity.

Uncertainty, noise and overfitting

The unavoidable issue of less-than-perfect large datasets is uncertainty in evaluating the quality of outcomes [41]. Sources of this include implicit noise that is likely to be present through variability and incompleteness. There is further uncertainty in the fact that models rarely capture an entire population; rather a sample of it [42]. There is an inherent uncertainty when generalising on larger scales. The quantification of uncertainty that comes from multiple sources is an ongoing issue in the academic community. Psaros et al., present a multitude of complex methods for quantifying the total uncertainty in a neural network [43] which include Bayesian methods, Ensembles, Functional priors and others. Tavazza et al., present three different approaches which include using the quantile loss function, machine learning

methods of the error, and Gaussian methods [44]. These methods come at varying costs in terms of time & monetary resource and computational cost.

Noise in a large dataset is a common in machine learning. Noise can be in the form of class noise (mislabelling) or attribute noise (incorrect values). Johnson & Khoshgoftaar state that class label noise "directly inhibits the predictive performance of machine learning algorithms" [45]. Gupta & Gupta state that the main methods for noise identification are Ensemble techniques, distance-based algorithms and single learning-based techniques. Once noisy instances have been identified, some argue that it is best to leave them and make a model robust enough to deal with the noise in the data [46]. Others say to filter them and replace them with an appropriate value – in this investigation, the midpoint value was taken.

Noise is one of the main causes of overfitting in a predictive model [47]. If a model is designed with too much complexity, it can result in it predicting outcomes on training sets perfectly but performing poorly on testing sets. When noise causes the model to learn its fluctuations, overfitting can be exacerbated. Dealing with overfitting due to noise without removing it can only be achieved through either increasing the quantity of accurate data to reduce the proportion of random error, or by fine tuning the model to account for the noise [48, 49]. These issues with overfitting have applications in this BankChurners project because reducing the loss through customer attrition is an important business objective.

References

[1] I. Stančin and A. Jović, "n overview and comparison of free Python libraries for data mining and big data analysis," in *2019 42nd International convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE*, 2019.

[2] pandas, "pandas.get_dummies," PyData, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html. [Accessed 23 June 2023].

[3] V. K. Singh, N. S. Maurya, A. Mani and R. S. Yadav, "Machine learning method using position-specific mutation based classification outperforms one hot coding for disease severity prediction in haemophilia 'A'," *Genomics,* vol. 112, no. 6, pp. 5122-5128, 2020.

[4] pandas, "pandas.isnull," PyData, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.isnull.html. [Accessed 23 June 2023].

[5] pandas, "pandas.DataFrame.sum," PyData, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sum.html. [Accessed 23 June 2023].

[6] pandas, "pandas.DataFrame.value_counts," PyData, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html. [Accessed 23 June 2023].

[7] Y. A. Farrukh, Z. Ahmad, I. Khan and R. Elavarasan, "A sequential supervised machine learning approach for cyber attack detection in a smart grid system," in *2021 North American Power Symposium (NAPS)*, 2021.

[8] seaborn, "seaborn.heatmap," PyData, 2023. [Online]. Available: https://seaborn.pydata.org/generated/seaborn.heatmap.html. [Accessed 23 June 2023].

[9] matplotlib, "matplotlib.pyplot.plot," matplotlib, 2023. [Online]. Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html. [Accessed 2023 June 2023].

[10] pandas, "pandas.DataFrame.replace," PyData, 2023. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html. [Accessed 23 June 2023].

[11] R. Haining, "Bivariate Correlation with Spatial Data," *Geographical Analysis,* vol. 23, no. 3, pp. 210-227, 1991.

[12] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastaval and T. Baker, "Analysis of dimensionality reduction techniques on big data," *IEEE Access,* vol. 8, pp. 54776-54788, 2020.

[13] scikit-learn, "sklearn.ensemble.RandomForestClassifier," scikit-learn, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed 23 June 2023].

[14] W. Feller, "On the normal approximation to the binomial distribution," *The annals of mathematical statistics,* vol. 16, no. 4, pp. 319-329, 1945.

[15] scikit-learn, “sklearn.metrics.confusion_matrix,” scikit-learn, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. [Accessed 23 June 2023].

[16] H. Dalianis, “Evaluation Metrics and Evaluation,” in *Clinical Text Mining*, Germany, Springer International Publishing, 2018, pp. 45-53.

[17] A. Zheng and A. Casari, Feature engineering for machine learning: principles and techniques for data scientists., O'Reilly Media Inc, 2018.

[18] M. Belkin, D. Hsu, S. Ma and S. Mandal, “Reconciling modern machine-learning practice and the classical bias–variance trade-off.,” in *National Academy of Sciences*, 2019.

[19] R. Sailusha, V. Gnaneswar, R. Ramesh, G. R. Rao and et al., “Credit Card Fraud Detection Using Machine Learning,” in *Credit Card Fraud Detection Using Machine Learning*, 2020.

[20] S. Raschka, J. Patterson and C. Nolet, “Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence,” *Information,* vol. 11, no. 4, p. 193, 2020.

[21] L. Tratt, “Dynamically Typed Languages,” in *Advances in Computers*, Elsevier, 2007, pp. 149-184.

[22] A. Holkner and J. Harland, “Evaluating the dynamic behaviour of Python applications,” in *Thirty-Second Australasian Conference on Computer Science*, 2009.

[23] A. S. Saabith, M. M. M. Fareez and T. Vinothraj, “Python current trend applications-an overview.,” *International Journal of Advance Engineering and Research Development,,* vol. 6, no. 10, 2019.

[24] K. R. Srinath, “Python – The Fastest Growing Programming Language,” *International Research Journal of Engineering and Technology (IRJET),* vol. 4, no. 12, 2017.

[25] R. Power and A. Rubinsteyn, “How fast can we make interpreted Python?,” *arXiv preprint arXiv,* vol. 1306, no. 6047, 2013.

[26] Q. Zhang, L.Xu, X. Zhang and B. Xu, “Quantifying the interpretation overhead of Python.,” *Science of Computer Programming,* vol. 215, p. 102759, 2022.

[27] S. v. d. Walt, S. C. Colbert and G. Varoquaux, “The NumPy array: a structure for efficient numerical computation,” *IEEE Computing in Science and Engineering,* vol. 13, no. 2, pp. 22-30, 2011.

[28] J. Brittain, M. Cendon, J. Nizzi and J. Pleis, “Data scientist’s analysis toolbox: Comparison of Python, R, and SAS Performance.,” *SMU Data Science Review,* vol. 1, no. 2, p. 7, 2018.

[29] D. Donoho, “50 years of data science,” *Journal of Computational and Graphical Statistics,* vol. 26, no. 4, pp. 745-766, 2017.

[30] E. Frank, M. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. H. Witten and L. Trigg, “Weka-a machine learning workbench for data mining,” *Data mining and knowledge discovery handbook,* pp. 1269-1277, 2010.

[31] C. Thornton, F. Huttler, H. H. Hoos and K. Leyton-Brown, “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms,” in *Proceedings*

of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining .

[32] J. Mitrpanont, W. Sawangphol, T. Vithantirawat, S. Paengkaew and P. Suwannasing et al., "A study on using Python vs Weka on dialysis data analysis," in *2017 2nd International Conference on Information Technology (INCIT)*, 2017.

[33] K. I. Sundus, B. H. Hammo, M. B. Al-Zoubi and A. Al-Omari, "Solving the multicollinearity problem to improve the stability of machine learning algorithms applied to a fully annotated breast cancer dataset," *Informatics in Medicine Unlocked,* vol. 33, p. 101088, 2022.

[34] V. J. Raymond, R. Raj and J. Retna, "Investigation of android malware with machine learning classifiers using enhanced pca algorithm," *Comput. Syst. Sci. Eng,* vol. 44, no. 3, pp. 2147-2163, 2023.

[35] S. M. H. L. K. T. B. W. K. C. S. W. P. Z. W. H. a. Y. L. C. J. Y. L. Chan, "A correlation-embedded attention module to mitigate multicollinearity: An algorithmic trading application," *Mathematics,* vol. 10, no. 8, p. 1231, 2022.

[36] J. Y. L. Chan, S. M. H. Leow, K. T. Bea, W. K. Cheng, S. W. Phoong, Z. W. Hong and Y. L. Chen, "Mitigating the multicollinearity problem and its machine learning approach: a review," *Mathematics,* vol. 10, no. 8, p. 1283, 2022.

[37] J. Cheng, J. Sun, K. Yao, M. Xu and Y. Cao, "A variable selection method based on mutual information and variance inflation factor," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy,* vol. 268, no. 120652, 2022.

[38] M. M. Samy, R. E. Almamlook, H. I. Elkhouly and S. Barakat, "Decision-making and optimal design of green energy system based on statistical methods and artificial neural network approaches," *Sustainable Cities and Society,* vol. 84, p. 104015, 2022.

[39] C. P. Obite, N. P. Olewuezi, G. U. Ugwuanyim and D. C. Bartholomew, "Multicollinearity effect in regression analysis: A feed forward artificial neural network approach," *Asian journal of probability and statistics,* vol. 6, no. 1, pp. 22-33, 2022.

[40] C. K. Chandrasekhar, H. Bagyalaskshmi, M. R. Srinivasan and M. Gallo, "Partial ridge regression under multicollinearity," *Journal of Applied Statistics,* vol. 43, no. 13, pp. 2462-2473, 2016.

[41] M. Kläs and A. M. Vollmer, "Uncertainty in machine learning applications: A practice-driven classification of uncertainty," in *Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE*, , Västerås, Sweden, 2018.

[42] P. S. Levy and S. Lemeshow, Sampling of populations: methods and applications, John Wilet & Sons, 2013.

[43] A. F. Psaros, X. Meng and Z. Zou, "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons," *Journal of Computational Physics,* vol. 477, p. 111902, 2023.

[44] F. Tavazza, B. DeCost and K. Choudhary, "Uncertainty Prediction for Machine Learning Models of Material Properties," *ACS Omega 6,* vol. 6, no. 48, pp. 32431-32440, 2021.

[45] J. M. Johnson and T. M. Khoshgoftaar, "A Survey on Classifying Big Data with Label Noise," *Journal of Data and Information Quality,* vol. 14, no. 4, pp. 1-43, 2022.

[46] S. Gupta and A. Gupta, “Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review,” *Procedia Computer Science,* vol. 161, pp. 466-474, 2019.

[47] X. Ying, “An overview of overfitting and its solutions." Journal of physics: Conference series,” *Journal of physics: Conference series,* vol. 1168, p. 022022, 2019.

[48] C. Bu and Z. Zhang, “Research on overfitting problem and correction in machine learning,” *Journal of Physics: Conference Series,* vol. 1693, no. 1, p. 012100, 2020.

[49] S. Shaweke and X. Liu, “Overfitting mechanism and avoidance in deep neural networks,” *arXiv preprint arXiv:1901.06566,* 2019.

[50] G. V. Rossum, “Python Programming Language,” in *USENIX annual technical conference*, 2007.

Appendices

Appendix A: Table of BankChurners attributes

Attribute	Discrete/Continuous	Unknowns	Dummies required
CCNum	Neither	N	N
Trans_date_Time	Neither	N	N
Attrition_Flag	Discrete	N	N
Surname	Neither	N	N
Age	Continuous	N	N
Gender	Discrete	N	Y
CreditScore	Continuous	N	N
Geography	Discrete	N	Y
Tenure	Continuous	N	N
Education_Level	Discrete	Y	Y
Income_Category	Discrete	Y	Y
Card_Category	Discrete	N	Y
Credit_Limit	Continuous	N	N
Total_Trans_Amt	Continuous	N	N
Total_Trans_Ct	Continuous	N	N
Balance	Continuous	N	N
NumOfProducts	Continuous	N	N
HasCrCard	Discrete	N	N
IsActiveMember	Discrete	N	N
EstimatedSalary	Continuous	N	N
is_fraud	Discrete	N	N
Exited	Discrete	N	N

Appendix B: pandas.get_dummies method used on DataFrame

```
89
90  dfTrain5_upsampled =pd.get_dummies(dfTrain5, columns=['Gender', 'Geography', 'Education_Level', 'Income_Category',
91                                                        'Card_Category'])
92
93  dfTest5_upsampled =pd.get_dummies(dfTest5, columns=['Gender', 'Geography', 'Education_Level', 'Income_Category',
94                                                      'Card_Category'])
95
```

Appendix C: Pandas .value_counts() method on Balances in DataFrame

Appendix C1: .value_counts() code

```
89  print('')
90  print('Train Balances')
91  print(100 * dfTrain['Balance'].value_counts(normalize=True).head())
92  print('')
93  print('Test Balances')
94  print(100 * dfTest['Balance'].value_counts(normalize=True).head())
```

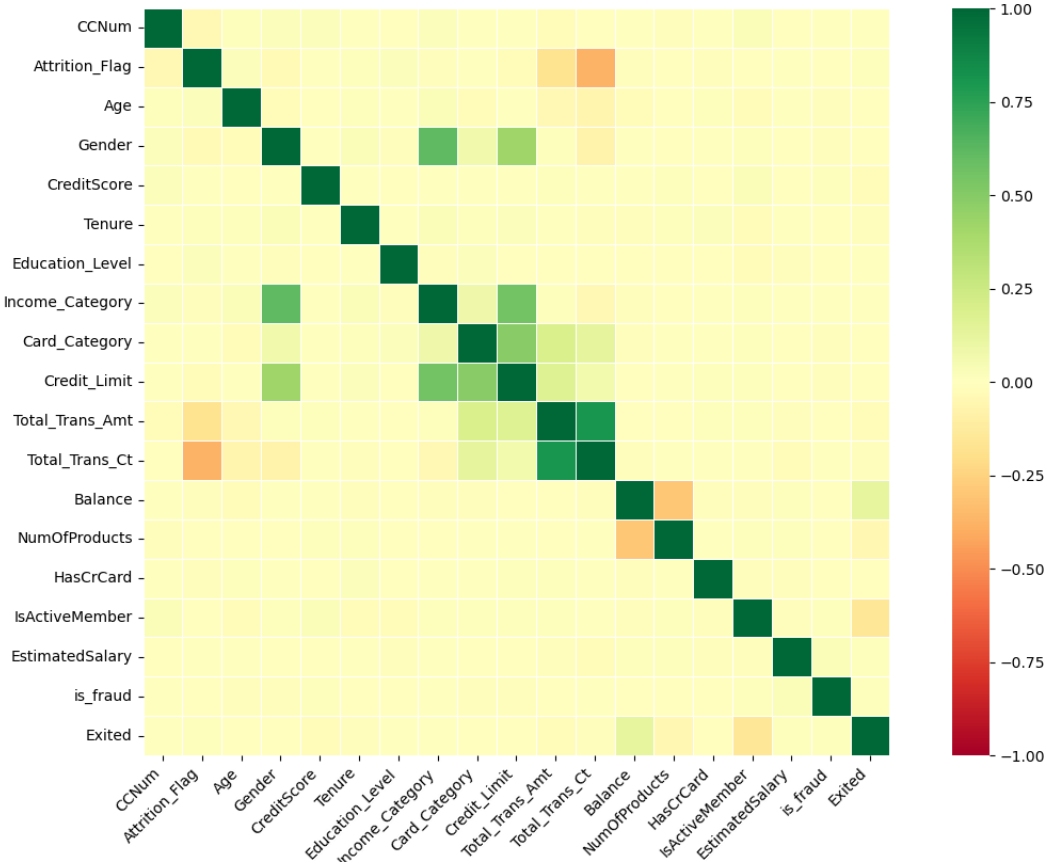
Appendix C2: .value_counts() on Balances console output for the Training and Testing datasets

```
Train Balances
0.00         36.17
130170.82     0.02
105473.74     0.02
85304.27      0.01
159397.75     0.01
Name: Balance, dtype: float64

Test Balances
0.00         36.831584
105473.74     0.099950
142662.68     0.049975
182781.74     0.049975
123204.88     0.049975
Name: Balance, dtype: float64
```

Appendix D: Seaborn Correlation map pre-processing and output

Appendix D1: Seaborn/Matplotlib Correlation map for BankChurners train



Appendix D2: Seaborn/Matplotlib Correlation map plotting method

```
315  cmap = "RdYlGn"
316  plt.figure(figsize=(20, 20))
317  ax = sns.heatmap(
318      dfTrain6.corr(),
319      vmin=-1, vmax=1, center=0,
320      cmap=cmap,
321      square=True,linewidths=.5
322  )
323  ax.set_xticklabels(
324      ax.get_xticklabels(),
325      rotation=45,
326      horizontalalignment='right'
327  )
328  plt.show()
```

Appendix D3: Pre-processing of non-numerical fields for correlation map using the Pandas .replace() method

```
dfTrain2['Attrition_Flag'] = dfTrain2['Attrition_Flag'].replace(['Existing Customer'], 0)
dfTrain2['Attrition_Flag'] = dfTrain2['Attrition_Flag'].replace(['Attrited Customer'], 1)
dfTrain2['Gender'] = dfTrain2['Gender'].replace(['M'], 1)
dfTrain2['Gender'] = dfTrain2['Gender'].replace(['F'], 0)
dfTrain3 = dfTrain2
dfTrain3['Income_Category'] = dfTrain3['Income_Category'] .replace(['Less than $40K'], 20)
dfTrain3['Income_Category'] = dfTrain3['Income_Category'] .replace(['$40K - $60K'], 50)
dfTrain3['Income_Category'] = dfTrain3['Income_Category'] .replace(['$60K - $80K'], 70)
dfTrain3['Income_Category'] = dfTrain3['Income_Category'] .replace(['$80K - $120K'], 100)
dfTrain3['Income_Category'] = dfTrain3['Income_Category'] .replace(['$120K +'], 150)
dfTrain3['Income_Category'] = dfTrain3['Income_Category'] .replace(['Unknown'], 85)

dfTrain4 = dfTrain3.copy()
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['Uneducated'], 1)
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['High School'], 2)
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['College'], 3)
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['Graduate'], 4)
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['Post-Graduate'], 5)
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['Doctorate'], 6)
dfTrain4['Education_Level'] = dfTrain4['Education_Level'] .replace(['Unknown'], 3.5)

dfTrain6 = dfTrain4.copy()
dfTrain6['Card_Category'] = dfTrain6['Card_Category'] .replace(['Blue'], 1)
dfTrain6['Card_Category'] = dfTrain6['Card_Category'] .replace(['Silver'], 2)
dfTrain6['Card_Category'] = dfTrain6['Card_Category'] .replace(['Gold'], 3)
dfTrain6['Card_Category'] = dfTrain6['Card_Category'] .replace(['Platinum'], 4)

```

Appendix E: Creation of profiles and target DataFrames for Profiles 1-3

```
attritionFlagTrain = np.array(dfTrain5_upsampled['Attrition_Flag'])
exitedTrain = np.array(dfTrain5_upsampled['Exited'])

attritionFlagTest = np.array(dfTest5_upsampled['Attrition_Flag'])
exitedTest = np.array(dfTest5_upsampled['Exited'])

#Profile 1
dfTrainAttProfile1 = dfTrain5_upsampled.loc[:, ['Total_Trans_Amt', 'Total_Trans_Ct', 'Card_Category_Blue',
                            'Card_Category_Gold', 'Card_Category_Platinum', 'Card_Category_Silver', 'Credit_Limit',
                            'Age', 'Gender_F', 'Gender_M']]

dfTestAttProfile1 = dfTest5_upsampled.loc[:, ['Total_Trans_Amt', 'Total_Trans_Ct', 'Card_Category_Blue',
                            'Card_Category_Gold', 'Card_Category_Platinum', 'Card_Category_Silver', 'Credit_Limit',
                            'Age', 'Gender_F', 'Gender_M']]

#Profile 2
dfTrainAttProfile2 = dfTrain5_upsampled.loc[:, ['Balance', 'NumOfProducts', 'IsActiveMember']]
dfTestAttProfile2 = dfTest5_upsampled.loc[:, ['Balance', 'NumOfProducts', 'IsActiveMember']]

#Profile 3
dfTrainAttProfile3 = dfTrain5_upsampled.loc[:, ['Gender_F', 'Gender_M', 'Credit_Limit', 'Income_Category_$120K +',
                                                'Income_Category_$40K - $60K', 'Income_Category_$60K - $80K',
                                                'Income_Category_$80K - $120K', 'Income_Category_Less than $40K', 'Age',
                                                'Card_Category_Blue', 'Card_Category_Gold', 'Card_Category_Platinum',
                                                'Card_Category_Silver', 'HasCrCard']]

dfTestAttProfile3 = dfTest5_upsampled.loc[:, ['Gender_F', 'Gender_M', 'Credit_Limit', 'Income_Category_$120K +',
                                                'Income_Category_$40K - $60K', 'Income_Category_$60K - $80K',
                                                'Income_Category_$80K - $120K', 'Income_Category_Less than $40K', 'Age',
                                                'Card_Category_Blue', 'Card_Category_Gold', 'Card_Category_Platinum',
                                                'Card_Category_Silver', 'HasCrCard']]

```

Appendix F: Random Forest Classifier for Profiles

Appendix F1: Random Forest Classifier and Confusion Matrix code for Profile 1

```
#Profile 1 attrition flag model
rf1Attrition = RandomForestClassifier(n_estimators=1000)
rf1Attrition.fit(dfTrainAttProfile1, attritionFlagTrain)
rf1AttritionPredictions = rf1Attrition.predict(dfTestAttProfile1)
rf1AttritionErrors = abs(rf1AttritionPredictions - attritionFlagTest)
rf1AttritionErrors = float(np.mean(rf1AttritionErrors))
rf1AttritionErrors = round(rf1AttritionErrors, 7) * 100
rf1AttritionConfusionMatrix = confusion_matrix(attritionFlagTest, rf1AttritionPredictions)

#Profile 1 exited model
rf1Exited = RandomForestClassifier(n_estimators=1000)
rf1Exited.fit(dfTrainAttProfile1, exitedTrain)
rf1ExitedPredictions = rf1Exited.predict(dfTestAttProfile1)
rf1ExitedErrors = abs(rf1ExitedPredictions - exitedTest)
rf1ExitedErrors = float(np.mean(rf1ExitedErrors))
rf1ExitedErrors = round(rf1ExitedErrors, 7) * 100
rf1ExitedConfusionMatrix = confusion_matrix(exitedTest, rf1ExitedPredictions)
```

Appendix F2: Mean absolute error and Confusion Matrices outputs for Profiles 1-3 for Attrition_Flag and Exited attributes

```
Profile 1 Attrition flag: Mean absolute error:  0.24988 %
Profile 1 Attrition flag: Confusion matrix:
 [[1603    0]
 [   5  393]]
Profile 1 Exited: Mean absolute error:  3.5982 %
Profile 1 Exited: Confusion matrix:
 [[1586   15]
 [  57  343]]
-----------------------------------------
Profile 2 Attrition flag: Mean absolute error:  8.54573 %
Profile 2 Attrition flag: Confusion matrix:
 [[1593   10]
 [ 161  237]]
Profile 2 Exited: Mean absolute error:  7.74613 %
Profile 2 Exited: Confusion matrix:
 [[1564   37]
 [ 118  282]]
-----------------------------------------
Profile 3 Attrition flag: Mean absolute error:  3.998 %
Profile 3 Attrition flag: Confusion matrix:
 [[1593   10]
 [  70  328]]
Profile 3 Exited: Mean absolute error:  4.74763 %
Profile 3 Exited: Confusion matrix:
 [[1579   22]
 [  73  327]]
```

Appendix G: Random Forest Classifier for Fraud indicator

Appendix G1: Creation of profiles and target DataFrames

```
#Geography profile
dfTrainGeography = dfTrain9_upsampled.loc[:, ['Geography_France', 'Geography_Germany', 'Geography_Spain',]]
dfTestGeography = dfTest9_upsampled.loc[:, ['Geography_France', 'Geography_Germany', 'Geography_Spain']]

#Card category profile
dfTrainCardCategory = dfTrain9_upsampled.loc[:, ['Card_Category_Blue', 'Card_Category_Gold', 'Card_Category_Platinum',
                                                 'Card_Category_Silver']]
dfTestCardCategory = dfTest9_upsampled.loc[:, ['Card_Category_Blue', 'Card_Category_Gold', 'Card_Category_Platinum',
                                               'Card_Category_Silver']]

#Gender profile
dfTrainGender = dfTrain9_upsampled.loc[:, ['Gender_F', 'Gender_M']]
dfTestGender = dfTest9_upsampled.loc[:, ['Gender_F', 'Gender_M']]

#Tenure profile
dfTrainTenure = dfTrain9_upsampled.loc[:,['Tenure']]
dfTestTenure = dfTest9_upsampled.loc[:,['Tenure']]
```

Appendix G2: Random Forest Classifier and Confusion Matrix code for Geography, Card_Category, Gender

```
#Geography fraud model
rfGeography = RandomForestClassifier(n_estimators=1000)
rfGeography.fit(dfTrainGeography, fraudTrain)
rfGeographyPredictions = rfGeography.predict(dfTestGeography)
rfGeographyErrors = abs(rfGeographyPredictions - fraudTest)
rfGeographyErrors = float(np.mean(rfGeographyErrors))
rfGeographyErrors = round(rfGeographyErrors, 7) * 100
rfGeographyConfusionMatrix = confusion_matrix(fraudTest, rfGeographyPredictions)

#Card category fraud model
rfCardCategory = RandomForestClassifier(n_estimators=1000)
rfCardCategory.fit(dfTrainCardCategory, fraudTrain)
rfCardCategoryPredictions = rfCardCategory.predict(dfTestCardCategory)
rfCardCategoryErrors = abs(rfCardCategoryPredictions - fraudTest)
rfCardCategoryErrors = float(np.mean(rfCardCategoryErrors))
rfCardCategoryErrors = round(rfCardCategoryErrors, 7) * 100
rfCardCategoryConfusionMatrix = confusion_matrix(fraudTest, rfCardCategoryPredictions)

#Gender fraud model
rfGender = RandomForestClassifier(n_estimators=1000)
rfGender.fit(dfTrainGender, fraudTrain)
rfGenderPredictions = rfGender.predict(dfTestGender)
rfGenderErrors = abs(rfGenderPredictions - fraudTest)
rfGenderErrors = float(np.mean(rfGenderErrors))
rfGenderErrors = round(rfGenderErrors, 7) * 100
rfGenderConfusionMatrix = confusion_matrix(fraudTest, rfGenderPredictions)
```

Appendix G3: Mean absolute error and Confusion Matrices outputs for Geography, Card_Category, Gender and Tenure

```
Geography fraud: Mean absolute error:  0.09995 %
Geography fraud: Confusion matrix:
 [[1999    0]
 [   2    0]]
-----------------------------------------
Card Category fraud: Mean absolute error:  0.09995 %
Card Category fraud: Confusion matrix:
 [[1999    0]
 [   2    0]]
-----------------------------------------
Gender fraud: Mean absolute error:  0.09995 %
Gender fraud: Confusion matrix:
 [[1999    0]
 [   2    0]]
-----------------------------------------
Tenure fraud: Mean absolute error:  0.09995 %
Tenure fraud: Confusion matrix:
 [[1999    0]
 [   2    0]]
```