Classification of Buy-Now-Pay-Later default clients through machine learning techniques tested on credit card default clients

Elizabeth Esnard

TMU Student#: 501342464

TMU Supervisor: Ceni Babaoglu

Date: 2025-06-06

Toronto Metropolitan University

Table of Contents

Abstract

Buy Now, Pay Later (BNPL), the financial service of purchasing products across short timelines in multiple installments, has over 389,000 active monthly users in Canada ("Statista Ltd., 2025a). BNPL services are available through third-party FinTech companies such as Klarna, Afterpay, and Affirm, with their apps exceeding 600,000 downloads in Q4 2024 Statista Ltd., 2025b), up 43% from Q4 in 2023.

Consumer Reports (2023) found in the United States, BNPL users "reported poorer financial health than nonusers across a wide variety of factors." The increase in BNPL usage as well as recent cost-of-living crises has led to news reports on some consumers defaulting on their BNPL payments (Andrew Ross Sorkin et al., 2025). As more Canadians turn to BNPL services, it is imperative for major banks to consider their BNPL clients' credibility to mitigate loss due to default.

There are several studies conducted on classification of credit card default; Yeh, I.-C., & Lien, C. (2009) compared six machine learning techniques such as K-nearest neighbour, artificial neural networks, and classification trees. Alam et. al (2020) investigated credit card default in imbalanced datasets and normalized with methods like SMOTE and random oversampling; Lawi, A. & Aziz, F. (2021) expanded on prior work to test support vector machine algorithms for classifying. The research on BNPL, in comparison, is still emerging (Taş, C. 2023).

This research paper aims to replicate the machine learning techniques of classifying credit card default on simulated BNPL clients and then compare the evaluation metrics to determine if the existing techniques can be efficiently translated onto BNPL transactions.

This paper will test the majority-recommended classification algorithms of artificial neural networks, LightGBM, random forest, and support vector machines based off the studies and conclusions of Yeh, I.-C., & Lien, C. (2009), Yang, S., & Zhang, H. (2018), Sayjadah, Y. et al, (2018) and Putri et al. (2021) respectively. The paper will use accuracy and model performance as expressed by the AUC value, based on Yang, S., & Zhang, H. (2018) to evaluation the results of the models and identify the best performing.

Literature Review

Buy now, pay later landscape in 2025

Predicting client default is a common research topic for banks and financial institutions. Debt is available to clients through multiple means like personal loans, credit cards, mortgages, lines of credit for businesses, and so on. Banks and emerging fintech (financial technology) companies are interested in meeting the demands of consumer clients and securing their business through BNPL services offered at the point of purchase by retailers, through some Canadian banking apps, or directly through the fintech apps.

BNPL options are increasing their market share in Canada, as seen in downloads per month. Per Statista, Q4 2024 saw Canadians download 621,993 BNPL apps like Klarna, Affirm, or Afterpay (Statista, 2025b). The financial culture in Canada due to the market share of the five national banks (Royal Bank of Canada, Toronto-Dominion Bank, Bank of Nova Scotia, Bank of Montreal, and Canadian Imperial Bank of Commerce) and their partnerships with card brands (Visa, MasterCard, American Express, and Interac) results in many Canadians still primarily relying on credit cards for their purchases, even after changing their preferred payment method due to cost of living increases (Statista, 2022; Statista, 2025c; Statista, 2025d).

There are reports Americans are starting to default on their BNPL loans; Klarna reported a 17% increase year-over-year on credit losses in May 2025 (Andrew Ross Sorkin et al., 2025). Consumer Reports found, in a survey of 2,017 Americans in 2022, that overall BNPL users felt less economically stable than their non-BNPL peers; 40% of BNPL respondents answered they didn't have enough money to pay their bills on time; 48% said they felt they had more debt than they could handle, and 34% reported they had over-drafted on their bank accounts (Consumer Reports, 2022). Despite this, Consumer Report's BNPL case study (2023) found American consumers had positive feelings about BNPL products, attracted by the ease of use and perceived no-interest cost and supposed no-fees.

Machine learning credit default models

BNPL default prediction is untapped in its research potential. It is still emerging as financial service and therefore has few studies looking into it professionally that are also publicly published. Credit card default prediction using machine learning techniques is well studied and continues to grow as new techniques are developed.

Yeh, I.-C., & Lien, C. (2009) is a well cited study that examined six data mining techniques to forecast the probability of default. The study used K-nearest neighbor classifiers (KNN), logistic regression, discriminant analysis, Naïve Bayesian classifiers, artificial neural networks, and classification trees. They split their data, sourced from a Taiwanese bank, into test and training data and trained their models using error rate and area ratio as their evaluation measure. They concluded artificial neural networks performed the best on their test data, based on the evaluation measures.

Yang, S. & Zhang, H. (2018) compared five data mining techniques to predict credit card default: logistic regression, neural networks, support vector machines, XGBoost (a distributed gradient boosting algorithm), and LightGBM (gradient learning framework based on tree learning). They used the ROC curve, the relationship between the true positive rate and the false positive rate, as well as the area under the curve (AUC) to evaluate the performance of their models, similar to Yeh, I.-C., & Lien, C. (2009). Yang and Zhang concluded LightGBM had the best classification results.

Sayjadah, Y. et al., (2018) tested logistic regression, random forest, and decision trees to "find the correlation and predictive power of factors contributing to credit card default." They found random forest algorithm performed best based on accuracy and AUC out of three machine learning techniques. Yash, H. et al (2023) also compared three models – logistic regression, support vector machines, and artificial neural networks – and concluded support vector machines were the most accurate. The study relied solely on accuracy for its evaluation measurement.

Bhandary, R., & Ghosh, B. K. (2025) compared three statistical methods and three machine learning methods to predict credit card default: linear discriminant analysis, logistic regression, support vector machines, XGBoost, random forest, and deep neural networks. They found the machine learning techniques greatly outperformed the statistical methods, with deep neural networks as the top performing model. This study also evaluated results based on the area under the receiver operating characteristic curve, as well as F1 score, G-mean, accuracy, sensitivity, specificity, and precision.

As technologies continue to advance, researchers will continue to compare statistical methods and machine learning methods to predict credit card default. Considering the work of emerging researchers like Taş, C. (2023), research akin to credit default prediction will be published on BNPL data, just as this paper intends to investigate.

Methodology

Based off the conclusions of the abovementioned papers, this paper selected artificial neural networks, LightGBM, support vector machines, and random forest to be the four experimental machine learning techniques to test on a simulated BNPL dataset.

Artificial Neural networks

Artificial Neural networks (ANN) mimic the connectivity of human brain neurons by forming layers through which information moves from input to output, with one or several hidden layers between. The inputs layer has a set number of features which are passed to all units of the hidden layers until finishing at the output layer. (Moocarme, M., et. al, 2021).

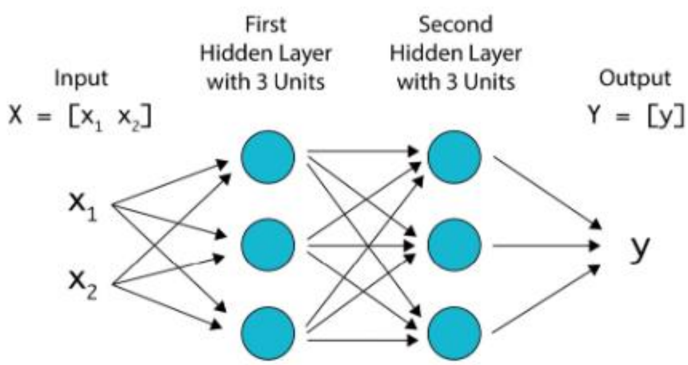


Figure 1. A diagram depicting ANN's layers, with 2 hidden layers. (Moocarme, M., et. al, 2021)

An ANN is beneficial when there is large amount of data, can handle unstructured data, and does not require feature selection for optimal performance. Python's Keras will be used to develop a classification model.

LightGBM
LightGBM is a gradient-boosting ensemble learning technique that sequentially combines models to produce an overall more robust model. In this method of machine learning, multiple models are trained by learning from the mistakes of the model before it in the sequence. What makes LightGBM different from general gradient boosting is its optimizations in theory and technique and therefore improving performance and accuracy. (Wyk, A. van. (2023).

The LightGBM model will be developed within Python using the LightGBM library.

Support Vector Machine Classifers
Support vector machines (SVMs) perform well for classification tasks with small or medium sized datasets. Géron, A. (2022) described SVM classifiers as “fitting the widest possible street between two classes,” known as *large margin classification.* In Figure 2, the classification through SVM is determined by the instances (observations within the dataset) located on the “edge” of the “street,” with the solid line representing the decision boundary of the classifier.

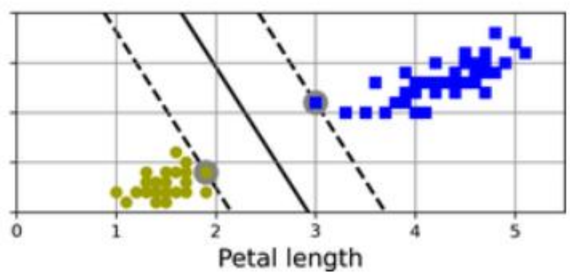


Figure 2. A diagram of SVM's 'streets' as represented by the dashed lines on the 'Iris' dataset. Géron, A. (2022).

SVM is sensitive to feature scaling, however. If the vertical scale and the horizontal scale of a plot are vastly different, it will make the “street” difficult to interpret. It’s therefore important to normalize the dataset prior to the SVM model (Géron, A. (2022)).

Random Forest
Random forest is an extension of decision trees, within the realm of ensemble methods like LightGBM. A random forest model is a collection of several decision tree classifiers which run independently; once a decision is made it enacts “voting” to find the most common / average decision. The classifiers constructed are trained on *random* subsets of attributes of the same dataset, differentiating it from typical decision tree models (Eckroth, J. (2018).

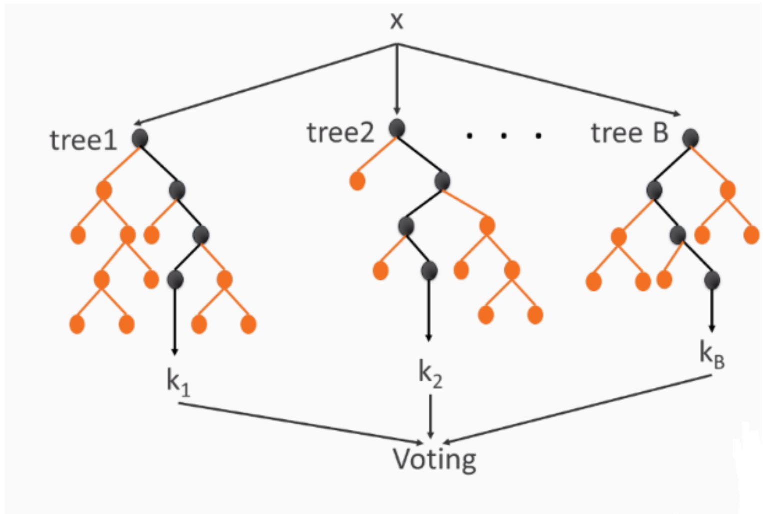


Figure 3. A depiction of 'voting' in a random forest model.
Eckroth, J. (2018).

Random forest models thrive with sufficient attributes and accuracy is key. They can be resource-heavy if there are too many trees in the model. (Eckroth, J. (2018).

Evaluation Measures

The main evaluation metrics for this paper are accuracy (i) and area under the recall-precision curve. These are the metrics that will be reported on and used to compare models, in support with support from (ii) recall, (iii) precision, and (iv) F1 score:

$$\text{(i) } Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

$$(ii)\ Recall = \frac{TP}{(TP+FN)}$$

$$(iii)\ Precision = \frac{TP}{(TN+FN)}$$

$$(iv)\ F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

These five evaluation metrics and common measures for classification models and are utilized in multiple studies such as Yang, S., & Zhang, H. (2018), Talaat, F. M., et al. (2024), and Alam, T. M., et al. (2020). Area under the recall-precision curve is utilized by Sayjadah, Y., et al. (2018), Bhandary, R., & Ghosh, B. K. (2025), and Yash, H., et al. (2023).

Precision, recall, and F1 score will not be reported on or used as a final evaluation metric because they will be used specifically to calculate the mean Average Precision (mAP) which is then used to calculate the area under the recall-precision curve, (a.k.a ‘AUC,’ ‘ROC,’ or ‘AUC curve.') The AUC curve plots precision against recall as a function of the model's confidence thresholds (Klingler, N., 2021). The AUC curve is a useful metric because it balances the trade-off of the between precision and recall; the AUC curve maximizes the effect of both metrics to give a better understanding of a model's accuracy (Shah, D., 2022).

F1 score on its scale of 0 to 1 indicates whether a model is a total failure or perfect. In this paper, however, F1 will act as a contribution to the AUC calculation because its value indicates the most optimal score threshold between precision and recall (Shah, D., 2022).

Evaluation metrics will be computed via the Python library scikit-learn's package 'metrics.'

Methodology Visualization

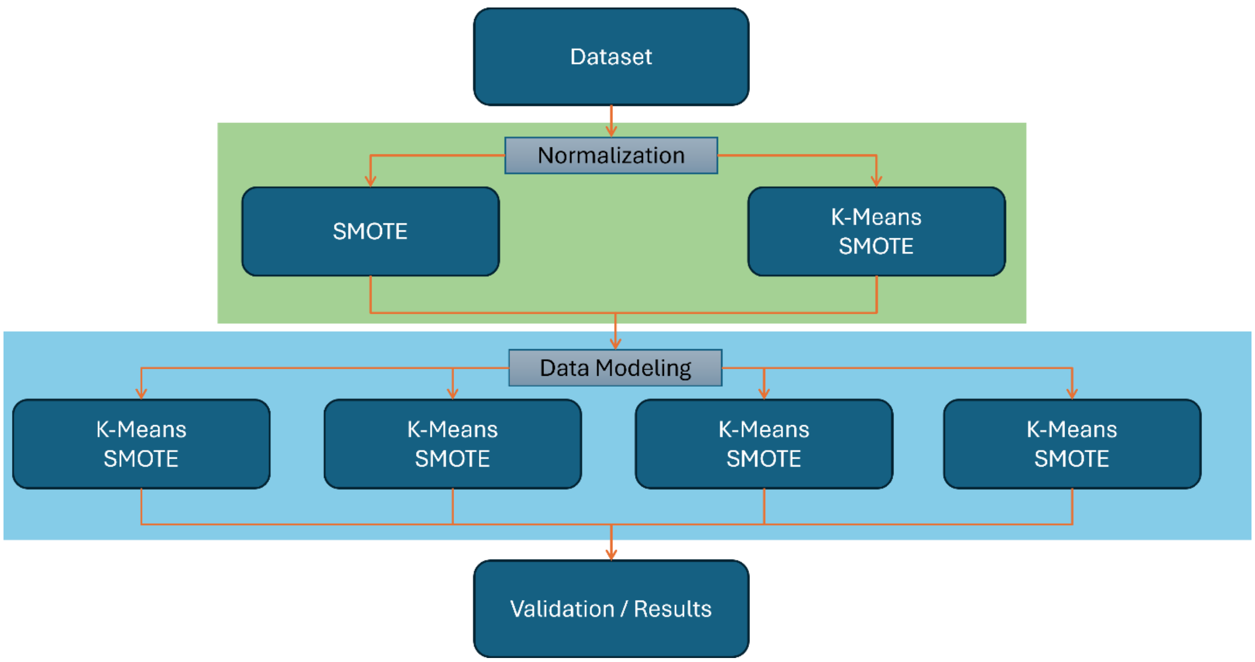


Figure 4. Visualization of research methodology.

Applications of data mining on buy now, pay later

Do the machine learning techniques practiced on credit card data to predict default classification work equally well on buy now, pay later data?

Credit cards and BNPL have multiple similarities; they allow a consumer to purchase something immediately and pay off in installments over a set-period of time. A credit card is often approved after an extensive credit check and is then sent to a client with a pre-determined credit limit. BNPL transactions have credit approvals on the spot; it's possible for BNPL credit approval to vary purchase-to-purchase depending on the user and the transaction. This difference in approval leaves room for further research into BNPL default prediction: for example on investigating how few variables are needed to approve credit at low risk therefore allowing financial institutions to encourage more spending.

There has been extensive research on which machine learning techniques are the best to accurately predict BNPL defaults. This paper has picked four (artificial neural networks, LightGBM, random forest, and support vector machines) to test and investigate if they will result in similar accuracy and AUC values when used on simulated BNPL data. Depending on the results of the study, this paper's findings could assist financial institutions with their BNPL business plans; create a new algorithm from scratch or adjust their credit card models to accommodate BNPL.

GitHub Repository, Link and Walkthrough

This paper is catalogued in the following GitHub repository with invitation: https://github.com/elizabethanneTMU820/CIND820_BNPL

Initial exploratory data analysis was performed in its own file for testing and refinement and is stored in the 'eda' folder of the repository. The output of the EDA transforms the raw dataset, sourced from Kaggle (bdoey1. (n.d.).), into a preprocessed csv file called 'clean_BNPL_python' which then goes on to be used in resampling processes. The EDA findings will be summarized below in 'Data Preprocessing.'

The working dataset 'clean_BNPL_python' (available in csv or xlsx) are also saved in the top level of the repository. The raw data from Kaggle is too large to be stored in the repository. The link to the original dataset is found in the README.txt file.

Initial results and past iterations of models can be found under the 'Initial Results' folder.

Final results are stored at the top of the repository.

Summary of Dataset Features

The dataset of the paper is sourced from Kaggle, user Brandon Doey (bdoey1. (n.d.).). The dataset lacks metadata to give context to its upload date, feature descriptions, or original source. Some features are clear thanks to their column names ('loan_amnt,' 'int_rate,' 'annual_inc') and others are less so ('mort_acc,' 'num_accts_120_pd'). However, it contains relevant features to the paper's intention of studying BNPL classification; there is a clear target feature ('loan_status'), as well as features other studies have defined as key (Talaat, F. M., et al, 2024) like payment details and outstanding bill amounts.

Table 1. The variable type and description of each feature from the dataset.

Feature Name	Variable Type	Description
loan_amnt	integer	Monetary amount of the loan
loan_term	character	Length of the loan in months

int_rate	numeric	Interest rate of the loan
monthly_payment	numeric	The monetary amount to be paid every month
sub_grade*	character	
emp_title	character	Job position title
emp_length	character	Length of time at current position
home_ownership	character	Category of home ownership
annual_inc	numeric	Annual income
verification_status†	character	Category of verification status of income
loan_purpose	character	Category of loan purpose
addr_state	character	US State of address
total_dti	numeric	Total debt to income
delinq_2yrs†	integer	Count of delinquency instances within last two years
open_acc†	integer	Length of loan account since opened in years
application_type	character	Category of application type
cur_acct_delinq	integer	Logical variable of current delinquency
tot_coll_amt	integer	Total collateral amount
tot_cur_bal	integer	Total current balance of the loan
mort_acc*	integer	
num_accts_120_pd†	integer	Number of accounts within a 120-day period
pub_rec_bankruptcies	integer	Number of publicly recorded bankruptcies
tax_liens	integer	Number of publicly recorded tax liens
credit_limit	integer	Credit limit
total_bal_ex_mort*	integer	
hardship_flag†	character	Logical variable to signal client may be at risk of financial hardship
disbursement_method†	character	Method of repayment
debt_settlement_flag†	character	Logical variable loan settlement
loan_status	character	Target variable

* Description unknown

† Educated guess of description, based on feature name

The original dataset holds 1,048,575 observations and 29 features. This paper reduced the observations to only those who borrowed between $1 and $3,500. This resulted in 59,092 observations. Doing so leads to some obvious conclusions, like the max loan_amount we observe is $3,500. Despite the reduction, there are still several outliers, such as the observation with a $7,000,000 annual income.

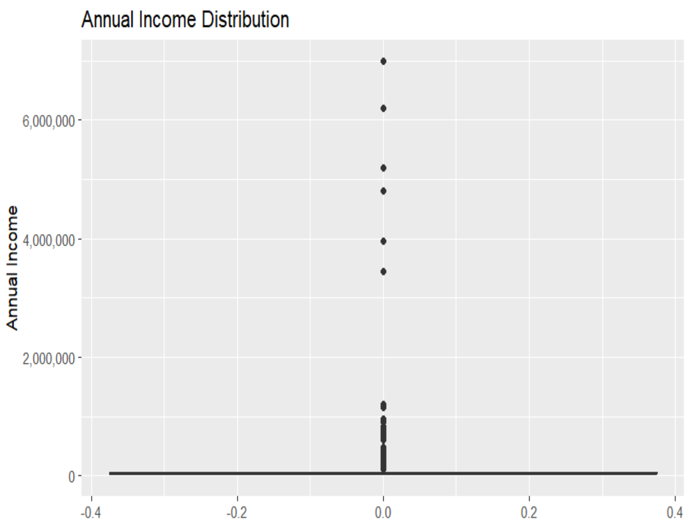
The summary statistics of the cleaned dataset, numeric features only, is as follows:

Table 2. Summary statistics of reduced dataset, numerical non-categorical/logical features

	loan_amnt	int_rate	monthly_payment	annual_inc	total_dti	tot_coll_amt	tot_cur_bal	credit_limit
count	59092	59092	59092	59092	59053	59092	59092	59092

mean	2416.88	13.2	81.55	54438.96	18.15	257.19	90534.45	115381.94
std	759.87	4.79	25.78	67295.12	13.91	1880.44	111906.41	130769.65
min	1000	5.31	29.76	0	0	0	0	0
25%	2000	9.76	63.18	32756	10.69	0	15496.75	30450
50%	2500	12.69	85.44	48000	17.17	0	40149.5	62967
75%	3000	15.99	101.8	67000	24.4	0	135296	167608.25
max	3500	30.99	149.53	7000000	999	208593	1389307	9999999

As seen in the boxplot distribution of Figure 5, there are several high-earning outliers. Whether this data was sourced from actual data or not, an educated guess can assume any individual making seven figure salaries likely would be the target audience of BNPL apps. Considering the third quartile of the reduced dataset's annual income is $67,000, capping the maximum annual income to $100,000 better captures a fair distribution of wealth of BNPL target users. On the opposite end of the spectrum, low income earners like those earning between $0 and $20,000 are the likely targets of BNPL apps and those individuals should remain in the dataset. Capping the annual income to $100,000 results in Figure 6.



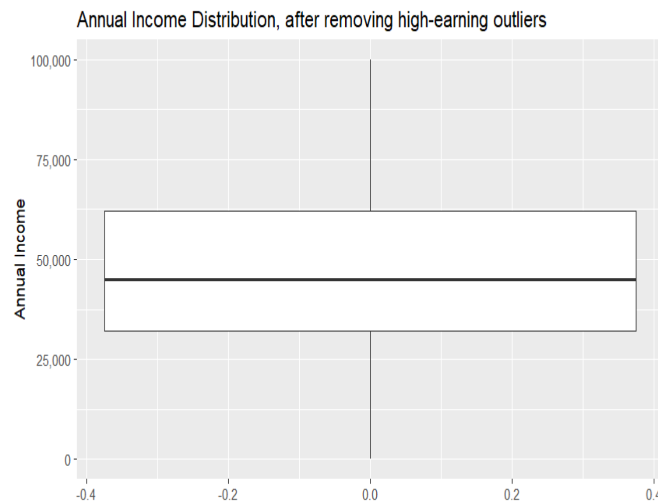


Figure 5 (Left). Annual Income Distribution via Box Plot on reduced dataset.
Figure 6 (Right). Annual Income Distribution with high earning outliers removed and max income capped at $100,000

The target variable for the dataset is 'loan_status.' From the raw data, there are seven categories ranging in solvency, late by X days, and default. This paper aims to predict default of BNPL clients on a binary level: will or will not default. Thus, the seven categories were converted to the target classes resulting in 4,687 as 'default' and 54,366 as 'solvent.' This is a 7.94% to 92.06% ratio.

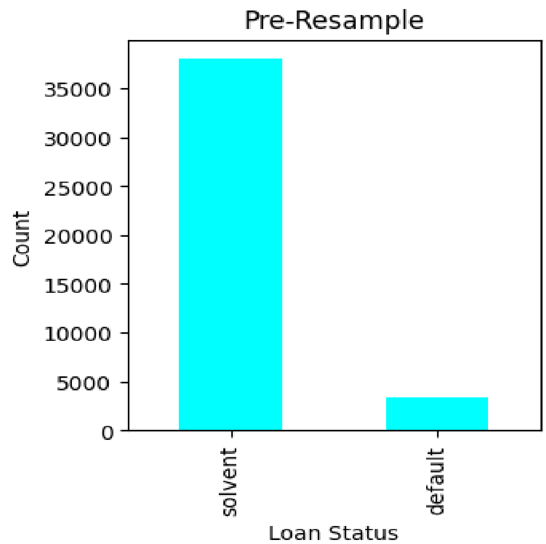


Figure 7. Distribution of target variable in cleaned dataset

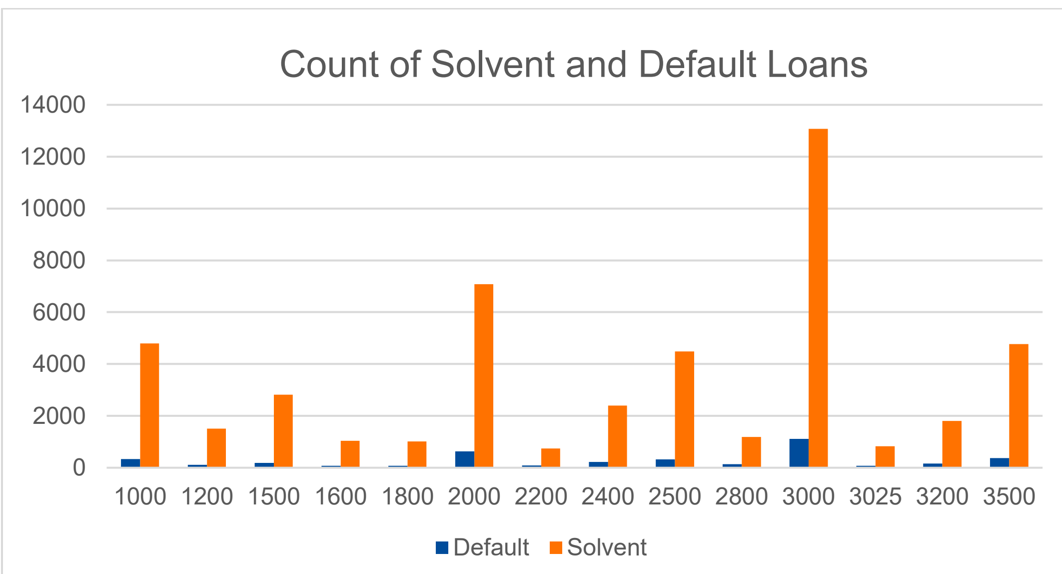


Figure 8. Count of Solvent vs. Default loans summarized to loan amounts where count of defaults was over 49

The loan amounts available in the dataset range (1 – 3,500) increment upward in amounts of 25. There are clusters of loan amounts seen at 'round' values like $1000, 2000, 2500, and $3000, as seen in Figure 8. Although there is a distinct increase in defaults at these values, there are also clear increases in solvent loans.

The most common reason for taking a loan in this dataset is debt consolidation; the financial act of securing a large loan to pay off smaller loans, thus reducing the amount of creditors an individual is facing but at the possible expense of a higher interest rate. The second most common is 'other' which the dataset has no insight to.

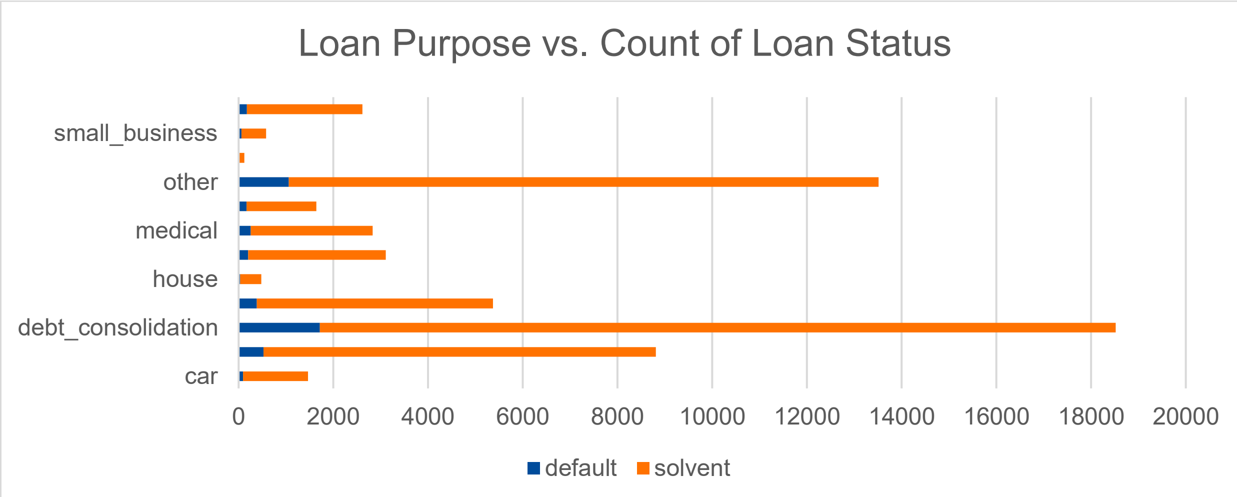


Figure 9. Distribution of loan status by loan purpose

The dataset provides a categorical variable of home ownership status: renting, owning, and mortgage (Table 3). A majority of defaulters rent, but the same can be said for solvent loan borrowers. In the current economic climate, its possible homeownership does not factor into loan default prediction.

Table 3. Summary of homeownership to loan status

	Mortgage	Own	Rent
Default	1407	653	2624
Solvent	20100	7491	26729

Public bankruptcies may, on the surface, also appear to be an indicator of loan default, but the initial analysis of the data is inconclusive. As seen in Table 4, those with 5 and 6 recorded bankruptcies have solvent loans.

Table 4. Summary of public bankruptcies to loan status

Number of Public Bankruptcies	Default	Solvent
0	3956	47584
1	684	6458
2	33	245
3	9	62
4	5	13
5	0	3
6	0	1

Categorical features based on human logic infer certain hypotheses; renters are more likely to default; those with a record of bankruptcy are likely to default; those who borrowed for a house are less likely to default than those who borrowed for a vehicle; those who debt consolidated have a history of debt and may be more likely to default. The possible hypotheses are endless. As seen by Table 3 and 4 however, the data does not completely support some of these hypotheses. Converting Table 3 into percentages, we can see homeowners are equally likely to default or not on loans.

Table 5. Summary of homeownership to loan status, as percentages

	Mortgage	Own	Rent
Default	30%	14%	56%
Solvent	37%	14%	49%

Patterns in the dataset are not implicit based off the exploratory data analysis. Further analysis is required to find insight into predicting defaulting debt users.

Data Preprocessing

The raw dataset from Kaggle was imported into python and standard exploratory data analysis functions were performed to confirm the state of the dataset; confirming dataset shape, summary of the data, column names, and data types. The detailed results are found in the GitHub Repository within the 'eda' folder, file 'BNPL_EDA_Python.

The observations were filtered to strictly loan amounts of less than or equal to $3500. This value was chosen based off a range given by respondents from a Canadian focus group (N'Kaa, 2023). The target variable 'loan_status' is converted to a binary classification of 'default' and 'solvent,' where solvent represents loans that are actively being paid down without late fees or penalties, seen in Figure 7. The imbalance of 7.94% to 92.06%, default to solvent, will be addressed through Synthetic Minority Oversampling Techniques (SMOTE). Details of the SMOTE resampling are found detailed below.

Summary statistics and boxplots reveal (Figure 5) reveal annual income of observations have several outliers, with the maximum value as $7,000,000. The third quartile of $67,000 reduces observations down to 43, 370, thus a max value of $100,000 for annual income was chosen to account for targeted BNPL clients, resulting in 55,168 observations.

Missing values were identified in three attributes: 'emp_title,' (7,812) 'emp_length,' (7,256) and 'total_dti' (36).

'emp_title' was removed entirely as there were no means of consolidation. Inspecting the dataset manually shows this attribute allowed free-type answers without standardization. There are a multitude of spelling errors, extra spaces, and mis-capitalization. Talaat, F. M., et al. (2024) found payment delays and outstanding loan amounts to be the most important features in their study, thus 'emp_title' was dropped.

'emp_length' had its missing values replaced with the feature medium of '5 years.' Although the mode of the variable is '10+ years,' subbing all null values to '10+ years' could skew the dataset due to this valuing being the max of the category. Thus, the medium was chosen as a suitable replacement. Prior to changing the null values to '5 years', the 'emp_length' variable was ordered by the pd.Categorical() function to allow displaying like below, not by count.

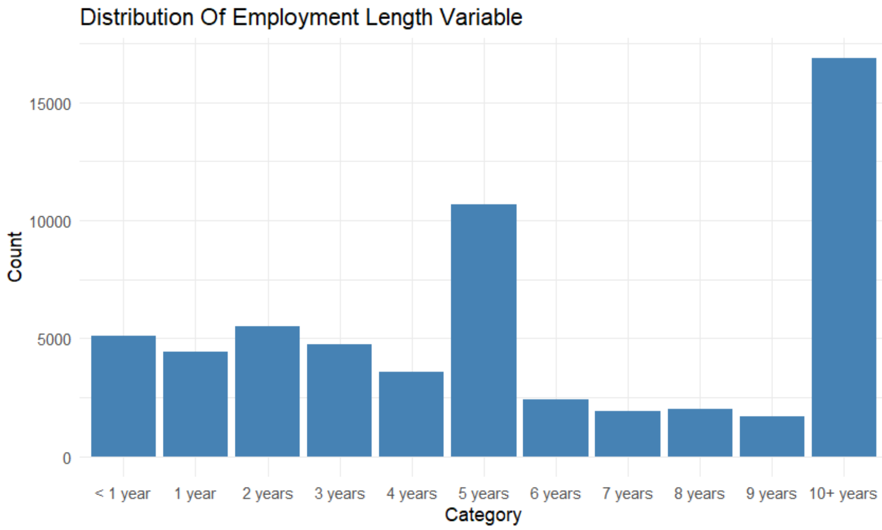


Figure 10. Distribution of Employment Length after Data Manipulation

‘total_dti’ is a variable whose exact meaning is unknown due to the lack of metadata on the dataset from Kaggle. Only 36 observations have missing values for ‘total_dti,’ and these values are all within the ‘solvent’ category of the dataset. Thus these 36 observations were removed, as ‘solvent’ observations are well-represented within the dataset.

The working dataset going forward is shaped 55,129 rows by 28 columns and is exported from the notebook as ‘clean_BNPL_python’ as both an excel file and a comma separated value file.

SMOTE Resampling

The process of converting the working dataset to the datasets to be used in the models follow the process as outlined in Figure 11:

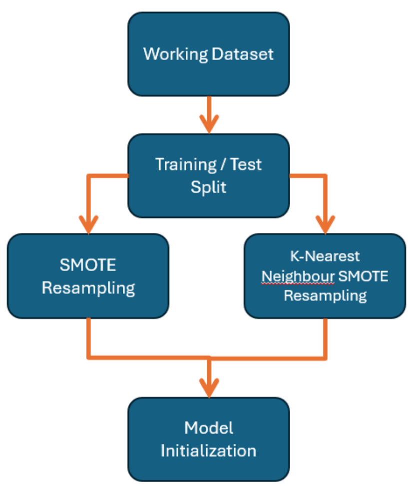


Figure 11: Visualization of Resampling workflow

Alam, T. M., et al., (2020) found SMOTE and K-means SMOTE oversampling methods produced the best results in predicting credit card default on an imbalanced dataset, thus why they were chosen as the oversampling methods of this paper.

The continuous and categorical variables of the dataset are separated into groups (bnpl_continuous, bnpl_categor), indices are identified, and the categorical group undergoes one-hot encoding via pd.get_dummies. This increases the number of columns in the dataset from 28 to 279. The dataset is split into test and training sets with 'train_test_split' from scikit-learn.

Using the SMOTE function from imblearn.over_sampling, a bnpl_smote class instance is created which is then used to transform the x_test and y_test variables bnpl_att_train and bnpl_tar_train, creating resampled variables bnpl_train_att_resample and bnpl_train_tar_resample. Checking the shape of these two new variables, we see the number of observations has increased by nearly 40% for the dependent variable. This is visualized with Figure 12.

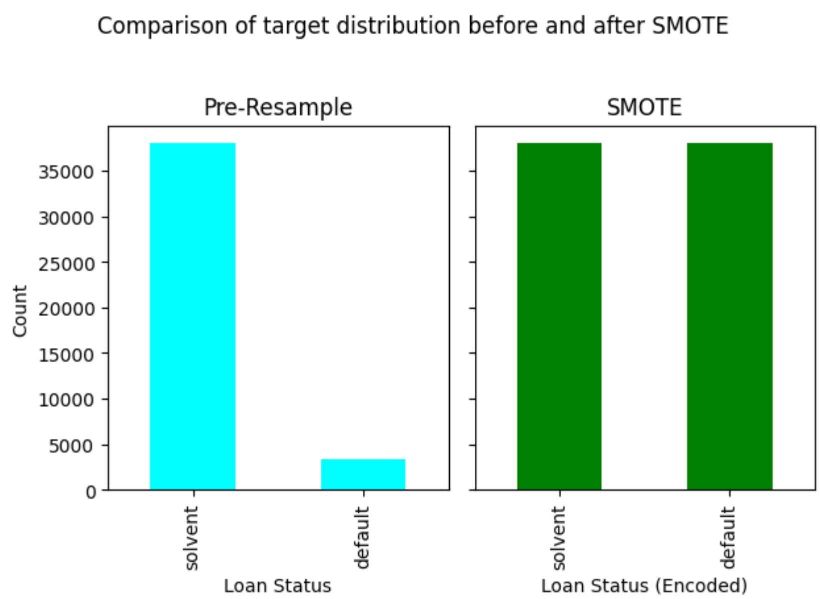


Figure 12. Comparison of number of observations pre and post SMOTE resampling.

The SMOTE process is then repeated using K-means SMOTE resampling, which follows the same process as above except using the function KMeansSMOTE from imblearn.over_sampling, setting the cluster balance threshold parameter to 0.1, and the kmeans estimator parameter to 30.

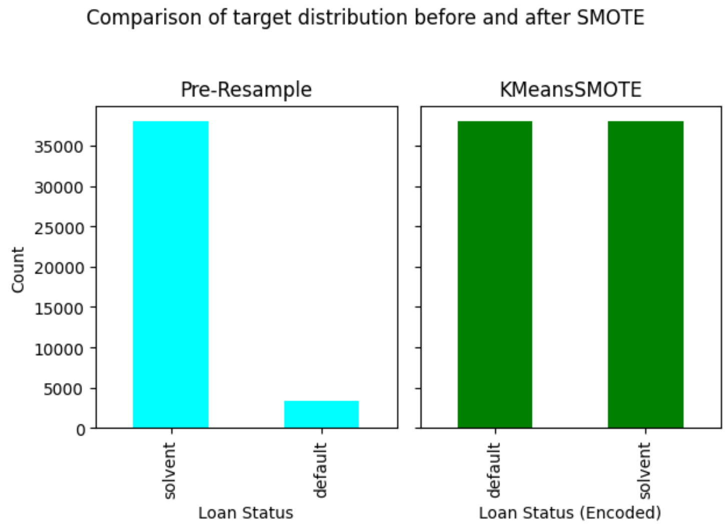


Figure 13. Comparison of number of observations pre and post KMeansSMOTE resampling.

The resampled data is now available for the models. Table 6 summarizes the variables that will be used as the training and test sets:

Table 6. Summary of training and testing datasets for the models

	x_train	**x_test**	**y_train**	**y_test**
SMOTE	bnpl_att_train_resample	bnpl_att_test	bnpl_tar_train_resample	bnpl_tar_test
KMeans SMOTE	bnpl_att_train_km_resample	bnpl_att_test	bnpl_tar_train_km_resample	bnpl_tar_test

Model Design

Models developed are created to compare the majority-recommended classification algorithms artificial neural networks, LightGBM, random forest, and support vector machines based on the studies and conclusions of Yeh, I.-C., & Lien, C. (2009), Yang, S., & Zhang, H. (2018), Sayjadah, Y. et al, (2018) and Putri et al. (2021) respectively.

The methodology of the model development is visualized in Figure 14:

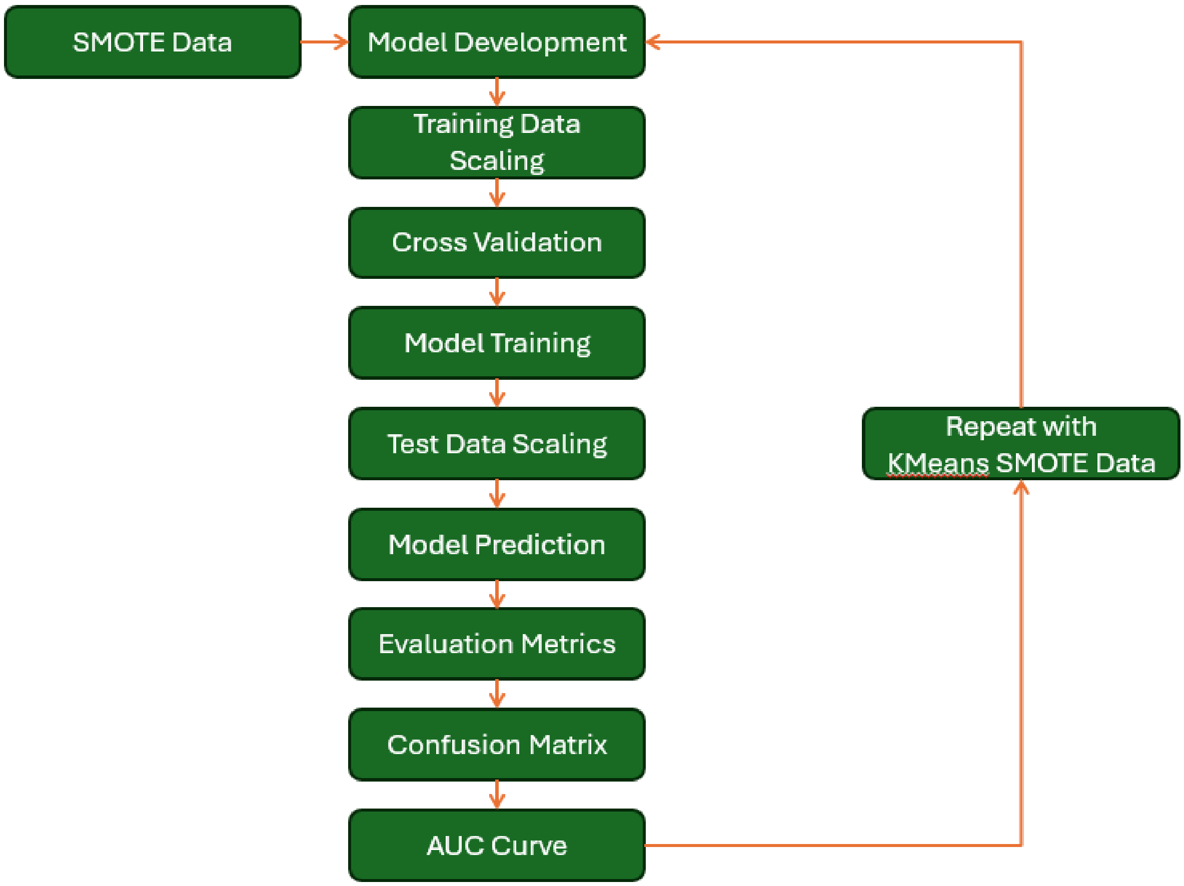


Figure 14: Visualization of model development workflow

Base Model: Logistic Regression

The logistic regression model is initiated and stored. After a change in the imported dataset, the model was upgraded with the parameter max_inter = 200; without this parameter the model does not return cross_val scores.

The SMOTE resampled training data is scaled via MinMaxScaler from sklearn. Initially the model started with StandardScaler, but this resulted in modest to poor accuracy results; testing found that MinMaxScaler was the optimal scaler to use. Cross validation is run to check the robustness of the model and returns a mean of 96% accuracy.

The model is fitted with the scaled attribute bnpl_att_train_resample_scaled and the SMOTE resampled bnpl_train_tar_resample. The y_test variable does not need to be scaled because it is a binary value. Once fitted, the model predicts on x_test (bnpl_att_test_resample_scaled), the predictions of which are held in tar_pred_test_lgrg.

Evaluation metrics for the model are calculated with accuracy_score, recall_score, and f1_score from sklearn. The model returns 93% test accuracy. The computed confusion matrix is displayed via a Seaborn heatmap for visualization purposes. Area under the curve is calculated with sklearn.metric's roc_curve, auc functions. Displayed with RocCurveDisplay, AUC is calculated to be 0.76.

The above is repeated exactly with the Kmeans-SMOTE resampled data and run through the same evaluation metrics. Cross validation returns a 95% mean accuracy; the trained model returns 93% accuracy and 0.77 AUC value.

Random Forest Classifier

The Random Forest Classifier model is initiated with sklearn_ensemble's RandomForestClassifier function. Max features to consider splitting on are set to 50, max depth is set to 5, and number of trees in the forest is set to 100. These values are considered due to examples from Eckroth, J.'s textbook (2018). The created model is saved in the variable bnpl_rf.

The random forest model does not use scaling to increase accuracy. Originally the model was developed without a scaler. After revision, scaling was used on the training data and the accuracy results in cross validation and fitting did not yield higher results, nor lower results. There was no difference between scaled and unscaled data. Within the jupyter notebook's cross validation of random forest, a timer was included to show scaling the data only adds computational time and therefore has no need to be included since it does not increase accuracy. Execution with scaling was 89.33 seconds vs 67.51 seconds, but results may vary.

Both executions return a cross validation score a mean of 85%.

The model is then fitted with the SMOTE resampled x_train and y_train variables, used to predict, and the evaluation metrics are calculated. Accuracy is 82.3% and AUC score is 0.76. A plot using plot_tree is then created and displayed to visualize the first decision tree the random forest generated, as seen by estimators_[0].

The above is repeated with the KMeans SMOTE resampled data. The trained KMeans resampled data model returns an accuracy score of 89% with 0.70 AUC value. The displayed tree with estimators_[0] also returns different split features.

Support Vector Machine

The functionality of an SVM model allows it to operate in high dimensionality, specifically using the svc function from sklearn.svm, but this model development relied on LinearSVC as an original basis. Accuracy results were high, thus it was determined there was no need to use svc and test higher dimensionality.

Within the LinearSVC parameters, the regularization parameter C can be set outside of its default of 1. Following examples in Géron, A's textbook (2023), the model was developed with C = 10, but this did not improve nor decrease the accuracy of the model and so was set to the default value of 1.

The SMOTE resampled data is scaled and used to perform cross validation, returning a mean accuracy of 96%. The model is then trained with the scaled data, used for prediction, and evaluation metrics are calculated. The accuracy returned is 93% and AUC value of 0.77.

The above was repeated with the KMeans SMOTE resampled data; cross validation has a mean accuracy of 95%; trained accuracy is 93%; AUC value is 0.77.

LightGBM

The LightGBM model is initiated with lightgbm's LGBMClassifer function, passing parameters 'gbdt' boosting type, number of estimators at 150, number of leaves at 120, and a learning rate of 0.09. These parameters were used following the example code of Wyk, A. van. (2023).

The SMOTE resampled data is scaled and the model is then run through cross validation and returns a 95% mean accuracy.

The model is fitted on the scaled training data and then used to predict on scaled test variables, and the results are run through evaluation metric calculations. Accuracy is returned at 92% and the confusion matric produced is displayed with seaborn. The AUC curve returns 0.76.

The above is repeated with the KMeans SMOTE Resampled data; cross validation returns a mean of 95.6% accuracy; the trained model has an accuracy of 92% and AUC value is 0.76.

Artificial Neural Network

A function is created to quickly develop hidden layers within a neural network. The function, "create_hidden_layer," takes a dataframe, a neural network model, an integer for the max number of layers wanted, and includes a default divisor of 2 to reduce the amount of neurons per layer as the layers progress to the output layer. This function was developed inspired by the teachings from Barari, S.'s video lectures (2019) and GitHub repository (2019) on neural network theory. The function creates the input layer and hidden layers of a model. The output layer is manually coded by the user.

Thereafter, the model is initiated with Keras Sequential() function, filled with create_hidden_layer function and a max level of 128.

Originally, the neural network model was created with a 'sigmoid' activation and a density of 1 unit, but after multiple runs, the models could not consistently earn high accuracy under these conditions. The output layer's activation was thereafter changed to 'softmax' and density changed to 2 in order to operate the network as a multi-class classifier with two classes. This led to pre-training accuracy returning much higher than with sigmoid activation; 7-8% with sigmoid, 40-50% with softmax.

Encoding the y variable data (y_test, y_train) through sklearn's LabelEncoder function is required else accuracy_score cannot be calculated against the predictions produced by the model. The predictions output as probabilities and must be converted via numpy's argmax back into binary categories.

Sklearn's cross_val_score function does not work with Keras models, so a for loop was created to develop models and run accuracy tests, based off the documentation of Kitchell, L. (n.d). A neural network, unlike a logistic regression model or support vector machine, retains its 'bias' or weights after a run. Per Géron, A., (2023): "For every output neuron that produced a wrong prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction." Thus, a Keras neural network cannot be used repeatedly to cross validate and test robustness as each run will improve its score.

To compensate, the function "create_nn_model" is developed to create a neural network quickly, relying on the "create_hidden_layer" function developed earlier, and it allows customization by the user in case evaluation metric, activation, or loss function needs to be changed.

KFolds from sklearn is initiated to split data into training and test sets and an empty list is created to hold the accuracy calculated from the for loop.

Inside the loop, SMOTE resampled x_train data is split into test and validation sets and y_train data first preprocessed via LabelEncoder and then split into sets. A model is developed, then fitted per the same specifications as the neural network model that will go onto be trained. Predictions are collected and the accuracy score is calculated, then saved into the list outside the loop.

The cross validation of neural networks returns a mean accuracy of 55%.

Moving onto training, the neural network is first compiled, setting accuracy as the evaluation metrics and choosing the relevant parameters. Originally, following demonstrations from Barari, S.'s video lecture series (2019), loss was set to 'binary_crossentropy' with optimizer 'rmsprop.' However, once the model's output layer was changed to 'softmax' these arguments were not training the model correctly and resulted in poor accuracy. Information from Kumar, A. (2020) and Keras documentation (Keras developers, n.d.) confirmed that due to the changes in the output layer to softmax, categorical_crossentropy was the better loss function and optimizer 'adam' is also best suited for said loss function.

The initial weights are saved in a variable in case the model needs to be reset. A callback was also developed to be efficient with resources; if there was no improvement in evaluation metrics (accuracy) after 5 training epochs, the training would cease. The data was scaled, as per other models in the paper, and finally the neural network is fitted.

The training epochs were run 10 and 50 times repeatedly to test and experiment on the training. An increase in epochs could result in better evaluation metrics, however it comes at resources and time expense. Experimentation found results were equally satisfactory at 10 or 50 epochs, thus time efficiency was chosen and the time epoch amount was set to 10. Performing the accuracy post training, the accuracy has increased to 30% and AUC score is 0.74.

The process is repeated with the KMeans SMOTE Resampled data; cross validation returns a mean accuracy of 50%; post training returns 92% accuracy and 0.51 AUC score.

Model Efficiency

Efficiency was monitored throughout the model development via the perf_counter function from python's time and was used to measure execution length of model training. Logistic regression, support vector machine, and LightGBM all executed within 10 seconds, with SVM the fastest at less than 4 seconds. Random forest executed at less than 20 seconds (SMOTE: 15sec; KMeans SMOTE: 14sec), likely due to its parameters. Decreasing some of the parameters (number of trees built from 100 to 50) could increase execution efficiency.

As expected, neural networks took the longest amount of time to execute at over 60 seconds, as they undergo epochs of training. Execution time can likely be decreased by decreasing the number of layers and/or neurons within the network, but that may come at the cost of accuracy. Conversely, one could decrease neurons/layers but increase epoch training to counteract a decrease in accuracy, but then that would increase execution time. Further research would be required to find a balance.

Resource usage was monitored via the python package pympler; to identify any memory leakage SummaryTracker printed a summary of the current state of the code execution compared to any previously printed summary (Pympler developers, n.d.). Following the tracker throughout the notebook, there were no major, unexpected jumps in object memory size.

Findings and Interpretations

The performance evaluation metrics of the models have been collected into Table 7. The table's five numeric columns have also been visualized via Figure 15.

Table 7. Summary of model evaluation metrics

Model	Resampling Method	Accuracy	Recall for Solvent	Recall for Default	F1 Score	AUC value
Logistic Regression	SMOTE	0.930467	0.999409	0.13626	0.96357	0.764413
Logistic Regression	KM SMOTE	0.930286	0.999277	0.135503	0.96348	0.765048
Random Forest	SMOTE	0.823871	0.856486	0.448145	0.89949	0.762312
Random Forest	KM SMOTE	0.894371	0.951833	0.2324	0.94313	0.695284
Support Vector Machine	SMOTE	0.930588	0.999606	0.135503	0.96364	0.766494
Support Vector Machine	KM SMOTE	0.930588	0.999606	0.135503	0.96364	0.76645
LightGBM	SMOTE	0.928351	0.996583	0.142316	0.9624	0.764728
LightGBM	KM SMOTE	0.92684	0.99448	0.147615	0.96156	0.76299
Neural Network	SMOTE	0.305641	0.253384	0.907646	0.40175	0.738676
Neural Network	KM SMOTE	0.920128	0.253384	0.907646	0.40175	0.507232

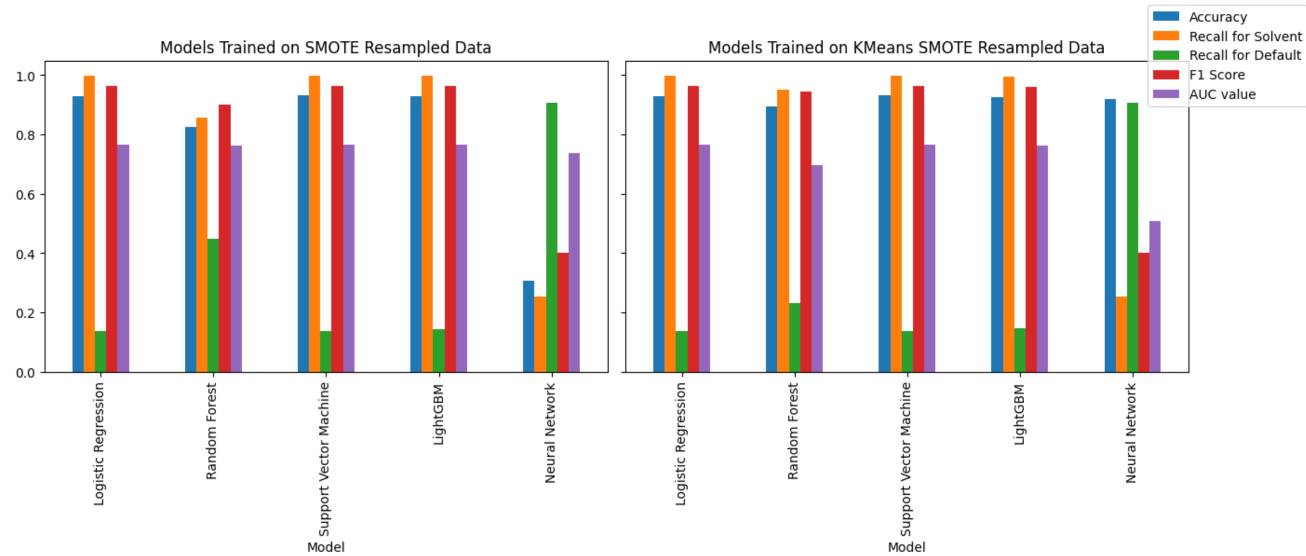


Figure 15. Grouped bar plot of evaluation metrics of models, split along resampling method

An interactive radar chart of the five evaluation metrics, developed with the plotly documentation (Plotly, (n.d.-a), (n.d.-b)) can be found in the GitHub repository under the file 'Eval Metrics Radar Plot.html.'

The two main evaluation metrics of the paper are accuracy and AUC curve value. Perfect accuracy of 1.0 or 100% indicates the model will always be correct in predicting classes, or possibly is overfitted to the dataset. The AUC value represents a model's ability to distinguish between the two classes, solvent and default, ranging from 0.5 (50%) to 1.0 (100%). Per Çorbacıoğlu, Ş. K., & Aksel, G. (2023), a model returning 0.5 – 0.6 is a fail, 0.6 – 0.7 is 'poor', '0.7 – 0.8' is fair and so on. Again, a 1.0 for AUC could indicate overfitting.

The logistic regression model, e.g. the baseline model, of the experiment performed strongly with 93% accuracy and an AUC value of 76% for both resampling methods. In terms of accuracy was slightly outperformed by SVM.

As seen by Table 7, the majority of models, save neural network with KMeans SMOTE resampling data, returned high accuracy and an area under the curve value within the 70% range, meaning the models could be considered 'fair' in their performance.

A baseline with strong results like seen in Table 7 is optimistic for the paper's main question of applying credit card classification models to BNPL classification; a standard model like logistic regression is performing well, so existing models that fintech companies or banks are currently employing can likely account for BNPL transactions.

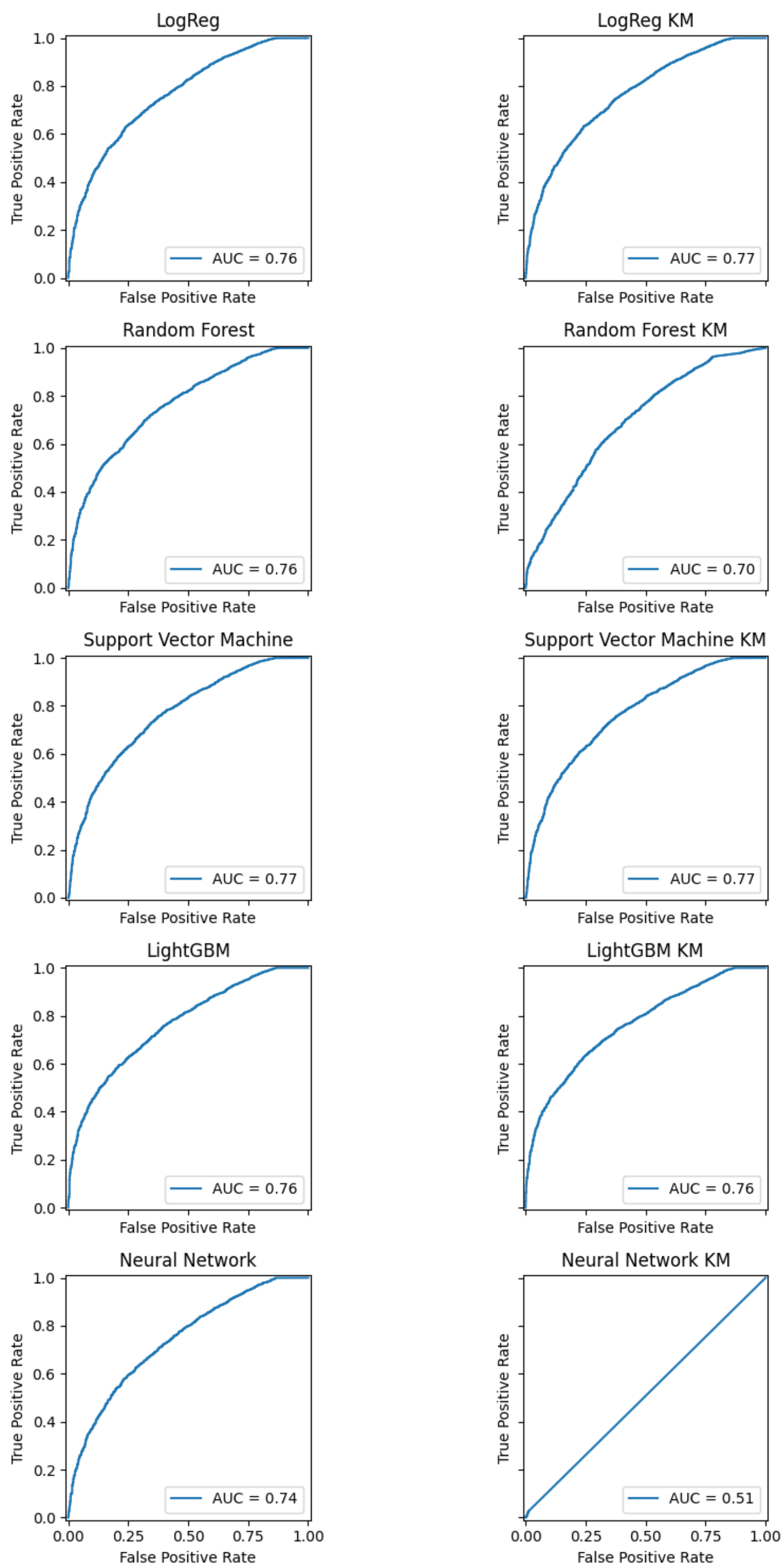


Figure 16. Collection of AUC curve displays of all models

Yang, S., & Zhang, H. (2018) used accuracy and AUC values to compare their models (logistic regression, neural network, support vector machine, XgBoost, and LightGBM) and also returned AUC values within the 70% range (0.72, 0.77, 0.72, 0.77, 0.79 respectively). Considering this comparison, of this experiment's AUC values and the results of Yang, S., & Zhang, H. (2018), there is further evidence that the classification methods of credit card default can be applied to Buy Now, Pay Later transactions with fair accuracy.

Logistic regression, support vector machine, and LightGBM performed the best based on the paper's main evaluation metrics Accuracy and AUC value. These three models also ran with the best efficiency and required minimal tuning to perform strongly. Random forest had solid results, but due to parameter tuning like max depth or number of trees, there is still room for execution improvement.

Neural networks proved to be the least efficient, least accurate model in the experiment; the SMOTE resampled model had classification accuracy only as strong as a coin flip. A neural network can be fine tuned to produce stronger results, but ultimately the experimentation of this paper could not replicate the results of Yeh, I.-C., & Lien, C. (2009), which found neural networks to be to perform classification more accurately than other tested models.

During training and various epoch runs, the neural network models were returning positive results (Figure 17), but when accuracy was manually calculated the results were not as promising (Figure 18).

```
Epoch 2/10
1771/1771 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9496 - loss: 0.1632 - val_accuracy: 0.9989 - val_loss: 0.0026
```

Figure 17. Epoch training output from SMOTE resampled data neural network model.

```
517/517 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.3038 - loss: 100310.6797
Training accuracy: 30.564120411872864%
```

Figure 18. Evaluation metric accuracy calculation from SMOTE resampled data neural network model.

The neural network performance can be further observed in Figure 19. Neural networks, the final row of the figure, experienced much higher values in false negatives. Random forest also experienced high false negatives when handling SMOTE resampled data, but less so with KMeans SMOTE resampled data. Logistic repression, SVM, and LightGM all had approximately 1120 false positives. The confusion matrixes also give insight into a shortfall of SVM, LightGBM and logistic regression that the neural network and random forest excel in: prediction of default.

As seen in Table 7, neural networks were the only model to have strong recall on default classification; 90% for both resampling methods. SVM, LightGBM, and logistic regression had poor results for default recall, all under 20% regardless of resampling method. Random forest had middling results with 44% on SMOTE data and 23% on KMeans SMOTE.

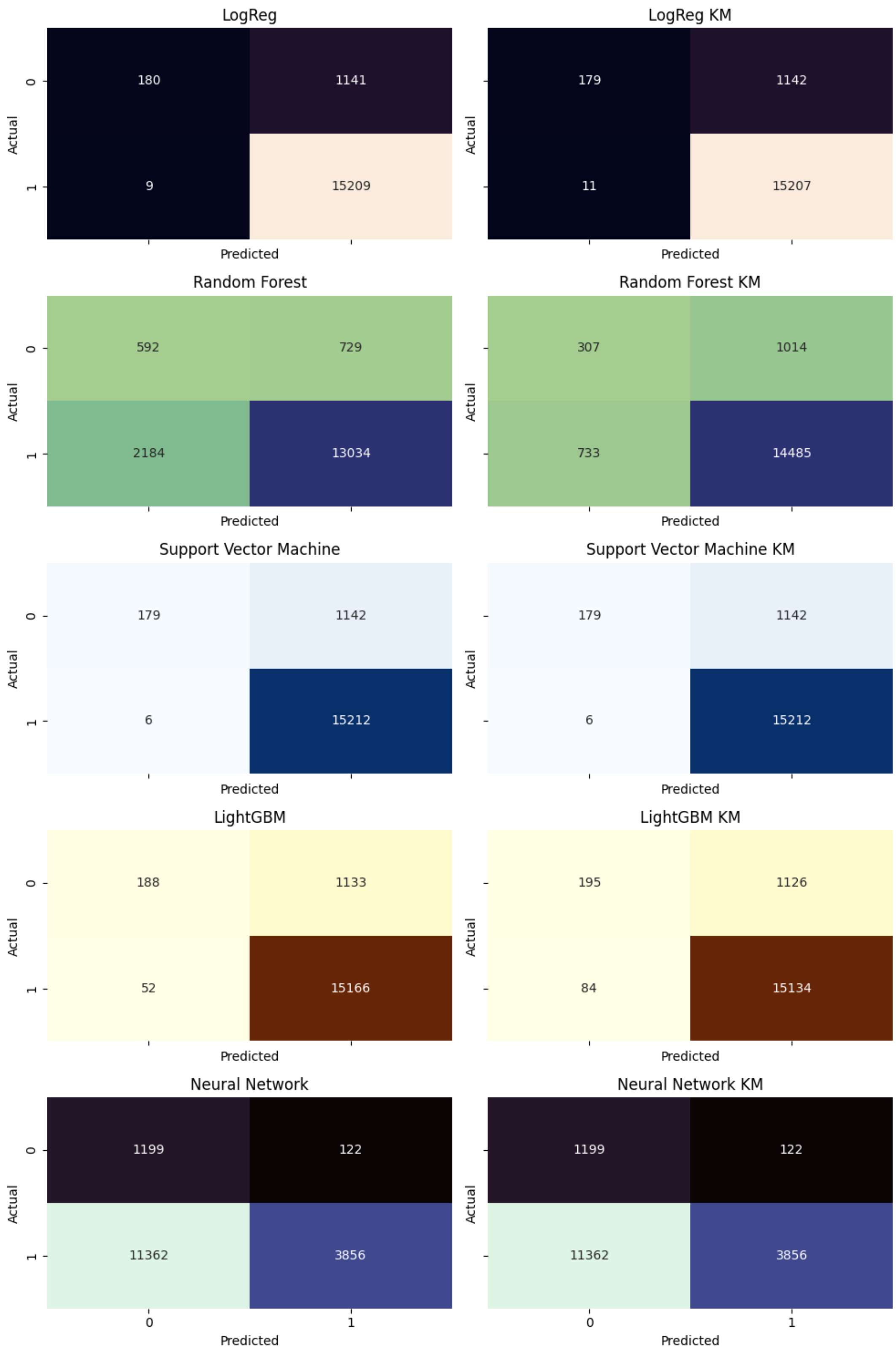


Figure 19: Collection of all confusion matrixes of all models

Although the experiment used SMOTE resampling to oversample the default observations, its clear there was not enough material for the models to accurately identify a default client. Further research is necessary to understand how this discrepancy can be addressed, such as if more attributes are required to identify key factors in classifying a default.

The results of this paper support the original research question: models used for credit card classification can be applied to BNPL datasets and can be accurate in their prediction. There is further research required, however, to counteract the poor accuracy on default, the crux of any loan offering service.

There is interest in the space of BNPL default classification. Taş, C. (2023) partnered with an Istanbul fintech company to experiment with common machine learning models on default prediction and had strong results, like 0.89 AUC value for LightGBM and Random forest, with 88% and 83% recall score on defaults, respectively. Considering Taş, C. was operating with actual BNPL data from a real service provider, the adage "garbage in, garbage out" is relevant. With actual BNPL data, a researcher can make better predictions without having to simulate BNPL transactions.

BNPL models are likely proprietary to their respective fintech or banking company; whether more research will be released publicly is yet to be seen. Likely, however, in those private companies data scientists are regularly tuning models to maximize the AUC curve so the organization can go on to offer more BNPL services to more clients while minimizing risk of loss due to loan delinquency.

As BNPL continues to grow worldwide, researchers who have studied credit card default in the past, like Sayjadah, Y. et al, (2018), Putri et al. (2021), and the others who inspired this paper could repeat their experiments with BNPL data. Again, however, they would need to find true BNPL data to garner strong results as simulated transactions may not prove sufficient.

Limitations and Ethical Considerations

The biggest limitation of this paper is the dataset sourced from Kaggle. Currently there is no BNPL dataset available to the public that accurate represents BNPL transactions as seen in 2025 (e.g. purchases as small as $10 – $50). As seen by Taş, C. (2023), the research that has been done on BNPL has been done in partnership with BNPL fintech companies.

Due to the dataset used for this paper, there was a clear imbalance in the target variables. Although SMOTE resampling methods were used to address this, the recall sources of default observations proved it was not sufficient.

General processing power also proved a limitation, as training the neural network was resource intensive. On a stronger machine with higher operating power, a neural network could get better training through more epoch runs, or by tuning the hidden layer amounts.

The dataset used in this paper had 29 features, only two of which could potentially be considered personal identifying information; job title ('emp_title') and state of residence ('addr_state'). There is no information directly indicating name, government issued identification number, gender, age, marital status, number of dependents, ethnicity, or education level. Some of those items could be inferred based off features in the data (for example, education level could be inferred based off job title), but insofar most sampling biases are minimized when using this dataset.

Buy Now, Pay Later services are shown to be common amongst those who want access to debt quickly and want an easier solution to budgeting (Statista, 2024). As mentioned in the "Literature Review" section, Consumer Reports found that overall BNPL users felt less economically stable than their non-BNPL peers and 40% of BNPL respondents answered they didn't have enough money to pay their bills on time.

Considering the surveyed results that BNPL users, 40% of the time, don't have enough cash on had to pay bills, and BNPL companies make a portion of revenue through late fees and interest on late payments, there may an ethical concern a business could use a default-prediction model to identify those who are likely to default and still offer them service because it would mean revenue. This idea would not be without risk, however; recall Klarna reported a 17% increase year-over-year on credit losses in May 2025 (Andrew Ross Sorkin et al., 2025).

Future Work and Recommendations

A stronger dataset would greatly assist in this experiment's predictions of BNPL default. A stronger dataset could mean a more balance on the target variable or more columns that would assist in predictions. Partnering with a fintech company would be ideal, as seen by Taş, C. (2023).

All models used offer a variety of parameters that can be tuned to enhance predictions. Further experimentation could adjust, for example, the max depth of a random forest or the activator of a neural network. Future papers could also try to improve efficiency in running cross validation or evaluation metrics. For example, the development of logistic regression models, SVM, and LightGBM follow similar methods and don't require additional steps like a neural network compile function. A future paper could try creating a for loop of these three models to more efficiency run the experiment.

As seen by the work of Alam, T et al. (2020), there are several SMOTE resampling methods available for machine learning models, so further papers could explore these as well or other sampling techniques like random oversampling, stratified random sampling, or cluster random sampling.

There continues to be research into credit card default prediction in 2025, as seen by Bhandary, R., & Ghosh, B. K. (2025) who researched six different models with the intention to compare modern machine learning techniques to traditional statistical techniques on predicting default. As new models are developed, new updates released, new datasets available, there will continue to be research into debt default prediction. The work by Taş, C. (2023) proves some focus is shifting towards alternative debt loans, but until more data is available publicly, research may continue to be locked behind private companies.

References

Alam, T. M., Shaukat, K., Hameed, I. A., Luo, S., Sarwar, M. U., Shabbir, S., Li, J., & Khushi, M. (2020). *An investigation of credit card default prediction in the imbalanced datasets.* IEEE Access, 8, 201173–201198. https://doi.org/10.1109/ACCESS.2020.3033784

Andrew Ross Sorkin, Warner, B., Kessler, S., de la Merced, M. J., Kaye, D., & McGregor, G. (2025). *'Bellwether of risks': What 'buy now, pay later' defaults say about the consumer: DealBook Newsletter.* The New York Times.

Barari, S. (2019). *Deep learning in Python : fundamentals of neural network theory*. [Video recording]. SAGE Publications Ltd.

Barari, S. (2019). Neural Network Basics. GitHub repository, https://github.com/soubhikbarari/sage-deep-learning/blob/master/1-NeuralNetworkBasics.ipynb

bdoey1. (n.d.). BNPL Dataset v1 [Data set]. Kaggle. https://www.kaggle.com/datasets/bdoey1/bnpl-data-v1

Bhandary, R., & Ghosh, B. K. (2025). *Credit Card Default Prediction: An Empirical Analysis on Predictive Performance Using Statistical and Machine Learning Methods.* Journal of Risk and Financial Management, 18(1), 23-. https://doi.org/10.3390/jrfm18010023

Consumer Reports. (2022, December). *American experiences survey: Financial attitudes and BNPL use [PDF report].* Consumer Reports. https://article.images.consumerreports.org/image/upload/v1672847923/prod/content/dam/surveys/Consumer_Reports_AES_December_2022.pdf

Consumer Reports. (2023, May). *Buy now, pay later: A case study for a digital finance standard.* https://advocacy.consumerreports.org/wp-content/uploads/2023/05/Buy-Now-Pay-Later-A-Case-Study-for-a-Digital-Standard-1-2.pdf

Çorbacıoğlu, Ş. K., & Aksel, G. (2023). *Receiver operating characteristic curve analysis in diagnostic accuracy studies: A guide to interpreting the area under the curve value.* Turkish Journal of Emergency Medicine, 23(4), 195–198. https://doi.org/10.4103/tjem.tjem_182_23

Eckroth, J. (2018). *Python artificial intelligence projects for beginners : get up and running with artificial intelligence using 8 smart and exciting AI applications (1st edition).* Packt.

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems (Third edition.).* O'Reilly Media, Inc.

imbalanced-learn developers. (n.d.). *Over-sampling methods [Software documentation].* imbalanced-learn.org. https://imbalanced-learn.org/stable/references/over_sampling.html

imbalanced-learn developers. (n.d.).(b) *SMOTE-NC [Software documentation].* imbalanced-learn.org. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTENC.html

Keras developers. (n.d.) *Losses [Software documentation].* Keras. https://keras.io/api/losses/

Keras developers. (n.d.). *Model training APIs [Software documentation].* Keras. https://keras.io/api/models/model_training_apis/

Kitchell, L. M. (n.d.). *Evaluating neural networks. Deep Learning With Keras.* https://kitchell.github.io/DeepLearningTutorial/8evaluatingnns.html

Klingler, N. (2021). *How to analyze the performance of machine learning models: Strategies for confusion matrix, precision, recall, specificity, and F1 score.* Viso.ai. https://viso.ai/deep-learning/analyzing-machine-learning-model-performance-strategies/

Kumar, A. (2020, October 28). *Keras – Categorical cross entropy loss function*. Analytics Yogi. https://vitalflux.com/keras-categorical-cross-entropy-loss-function/

Lawi, A., & Aziz, F. (2018, October). *Classification of credit card default clients using LS-SVM ensemble.* In 2018 Third International Conference on Informatics and Computing (ICIC) (pp. 1–4). IEEE.

Moocarme, M., So, A. V., & Maddalone, A. (2021). *The tensorflow workshop : a hands-on guide to building deep learning models from scratch using real-world datasets (1st edition.).* Packt.

N'kaa, C. (2023). *"Buy Now, Pay Later": Assessment of risks and remedies.* Option Consommateurs. https://option-consommateurs.org/en/research-reports/buy-now-pay-later

Plotly. (n.d.-a). *Subplots: Creating multi-plot figures in Python [Software documentation].* Plotly. https://plotly.com/python/subplots/ GitHub

Plotly. (n.d.-b). *Radar chart (line_polar and scatterpolar) [Software documentation].* Plotly. https://plotly.com/python/radar-chart/

Purkait, N. (2019). *Hands-on neural networks with Keras : design and create neural networks using deep learning and artificial intelligence principles (1st edition).* Packt Publishing.

Putri, N. H., Fatekurohman, M., & Tirta, I. M. (2021, March). *Credit risk analysis using support vector machines algorithm.* In Journal of Physics: Conference Series (Vol. 1836, No. 1, p. 012039). IOP Publishing.

Pympler developers. (n.d.). *Table of contents [Software documentation].* Pympler. https://pympler.readthedocs.io/en/latest/

Sayjadah, Y., Hashem, I. A. T., Alotaibi, F., & Kasmiran, K. A. (2018). *Credit card default prediction using machine learning techniques.* 2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA), 1–4. https://doi.org/10.1109/ICACCAF.2018.8776802

Shah, Deval. (2022). *Mean average precision (mAP) explained: Everything you need to know*. V7 Labs. https://www.v7labs.com/blog/mean-average-precision

Statista (2022). *Inflation impact on online payment use Canada 2022: Share of consumers in Canada who indicated that increasing cost of living changed their preferred online payment methods in 2022. [Dataset].* Statista Ltd.

Statista (2024*). BNPL motivation, by age 2024: Main motivation for consumers worldwide on why they used buy now, pay later (BNPL) more often in 2024, by demographic. (2024). [Dataset].* Statista Ltd.

Statista (2025a). *Buy now, pay later app users in Canada 2025: Biggest buy now, pay later (BNPL) apps in Canada as of March 2025, by number of monthly active users (MAU). [Dataset].* Statista Ltd.

Statista (2025b). *Buy now, pay later downloads Canada 2015–2025: Biggest buy now, pay later (BNPL) apps in Canada from 1st quarter of 2015 to 1st quarter of 2025, by number of downloads. [Dataset].* Statista Ltd.

Statista (2025c). *Visa, Mastercard market share in Canada: Market share of payment card brands - Visa, Mastercard, American Express, or in-market local card schemes - in Canada from 2016 to 2023. [Dataset].* Statista Ltd.

Statista (2025d*). Biggest POS payment methods in Canada 2030: Market share of cash, credit cards, and other payment methods at point of sale (POS) in Canada from 2017 to 2024, with a forecast for 2030. [Dataset].* Statista Ltd.

Talaat, F. M., Aljadani, A., Badawy, M., & Elhosseini, M. (2024). *Toward interpretable credit scoring: integrating explainable artificial intelligence with deep learning for credit card default prediction.* Neural Computing & Applications, 36(9), 4847–4865. https://doi.org/10.1007/s00521-023-09232-2

Taş, C. (2023). *Comparison of machine learning and standard credit risk models' performances in credit risk scoring of Buy Now Pay Later customers* [Master's thesis, ProQuest Dissertations & Theses].

Wyk, A. van. (2023). *Machine Learning with LightGBM and Python : A Practitioner's Guide to Developing Production-Ready Machine Learning Systems (First edition.).* Packt Publishing Ltd.

Yang, S., & Zhang, H. (2018*). Comparison of several data mining methods in credit card default prediction.* Intelligent Information Management, 10(05), 115.

Yash, H., Affan, Saurav, K., & Dhanda, S. S. (2023). *Credit Card Default Prediction Using Machine Learning Models.* 2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT), 1–5. https://doi.org/10.1109/CISCT57197.2023.10351316

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