Predicting Credit Card Approvals Using Machine Learning

Ashara Dangallage Don (210492314)

Department of Computer Science

Queen Mary, University of London

January 27, 2025

1 Project Statement

The Project task is to develop binary regression/classification models to predict the approval status of credit card applications. The goal is to use the credit card dataset, which contains various features describing applicants, to build predictive models. These ML models then should analyse the provided features and determine the likelihood of a credit card application being approved or rejected.

2 Introduction

This project involves the development of machine learning models aimed at predicting credit card approval decisions based on historical application data. By using a combination of data preprocessing, model training and evaluation techniques, this report demonstrates how predictive analytics can improve the accuracy of credit card approval processes.

In the initial analysis, a descriptive statistical analysis was conducted to examine key statistical characteristics and interrelationships among applicant features, including numerical and categorical variables. This analysis involved graphical visualisations, frequency distributions and correlation analyses to gain insight into the dataset's underlying patterns.

Two machine learning models (Logistic Regression and K-Nearest Neighbors) were then used to classify applications as either approved or denied. The models were trained and tested on a preprocessed credit card dataset, which involved steps such as encoding categorical variables, addressing missing data, splitting the data into training and test sets and evaluating the models predictive performance through ROC curves and model accuracy. The results indicate high predictive accuracy for both models, with the KNN classifier displaying slightly better performance. However, the low to medium AUC scores for both models suggests the presence of a class imbalance issue within the dataset.

3 Dataset Analysis

This section explores the analysis of the dataset [1] to identify key patterns and relationships among its features, which include numerical and categorical variables. The main objective is to discern factors that differentiate approved and rejected credit card applicants and evaluate their potential predictive capability. Using descriptive statistics, visualisations and correlation analysis, insights into income, employment history, property ownership and education have been uncovered, along with challenges such as skewed distributions and outliers.

This dataset is related to predicting credit card approval/rejection based on various features, including numerical, categorical and binary. It contains 1548 rows and 14 columns, with 7 numerical features, 8 categorical features and 4 binary features. The target variable ”rejection”, is central to the analysis as it represents the outcome being predicted, with its primary focus being to identify whether specific combinations of demographic or financial related attributes lead to credit card approval or rejection. Additionally, the original dataset was modified by converting the ”date of birth” feature into ”age” to make it more practical for analysis. This new version of the dataset, with the same features (except ”date of birth”), was then saved as a new dataset named ”creditcard.csv” [2].

I performed descriptive statistical analysis [3] of the dataset which revealed differences between approved and rejected users in terms of income, age, number of children and employment history. Approved users, on average, have slightly lower annual incomes ($190309 vs. $200263) but demonstrate far greater variability, with an income range almost three times larger than that of rejected users, although their middle-income ranges (IQRs) are similar. In terms of age, rejected users are marginally older on average (44.37 years vs. 42.62 years), though the overall distribution of ages is comparable across groups, with nearly identical interquartile ranges and ranges. Regarding family size, approved users have a slightly higher average number of children (0.42 vs. 0.37) and display greater variability, as seen by their broader range (up to 14 children, compared to a maximum of 4 for rejected users). Employment history offers an interesting contrast, where rejected users have a higher average number of employed days (-71,484 vs. -57,820). Notably, both groups exhibit negative medians, indicating that the majority of individuals are currently employed. Approved users are more likely to own property, with 900 (”Y”) compared to 110 for rejected users, though ”Y” is common in both groups. For education, ”Secondary / secondary special” is the most frequent level (922 approved, 109 rejected), but ”Higher education” is relatively more prominent among rejected users (55 vs. 371). Based on the results, although most features seem to be minor factors, employment history, property ownership and education stand out as potentially more significant in influencing approval outcomes.

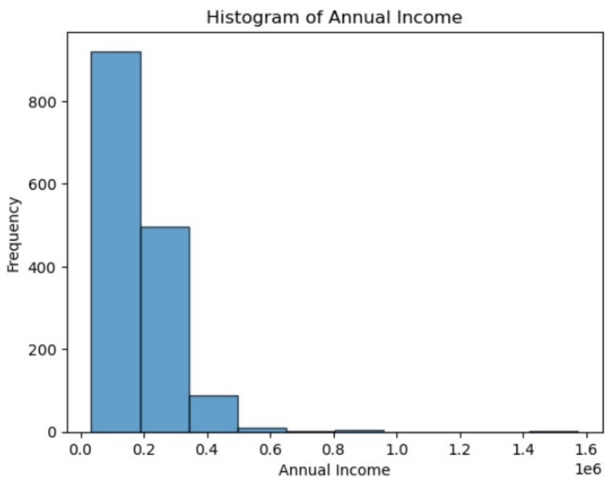


Figure 1: Annual Income demonstrates greater skewness

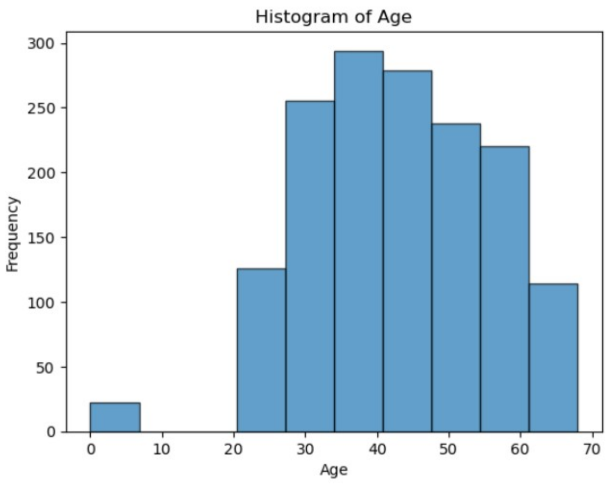


Figure 2: Age demonstrates more uniformity

The numerical features in the dataset include variables such as annual income, age and the number of family members. Key descriptive statistics, such as the mean, median, standard deviation and range, were computed for these features to understand their distributions. For instance, the ”Annual income” exhibited a wide range, indicating significant variability among users, with some outliers in higher income brackets. The ”Age” distribution on the other hand revealed a slightly right-skewed pattern. The standard deviation of numerical features such as ”Employed days” suggests a variety of employment options across the population. For numerical features, missing values (”NaN”) were filtered out before performing calculations like correlation. Additionally, standardization was applied by subtracting the mean and dividing by the standard deviation to normalize the data.

The categorical features in the dataset, including ”Education level”, ”Marital status”, ”Housing type” and ”Type of income”, were analysed using frequency distributions and bar plots [4]. The plots revealed some imbalances within categories, such as a higher proportion of users with ”higher education” compared to other levels or more users classified as being married. To prepare the categorical features for machine learning, they were encoded using one-hot encoding [5] for features with multiple unique categories and binary encoding for features with only two unique values. This ensured that the dataset was ready for numerical computations while retaining the information carried by the categorical variables. Binary features such as ”Car Owner”, ”Property Owner” and the target variable ”Rejection” were also analysed for their proportions. Frequency distributions showed that a significant percentage of users owned property, while fewer users owned cars.

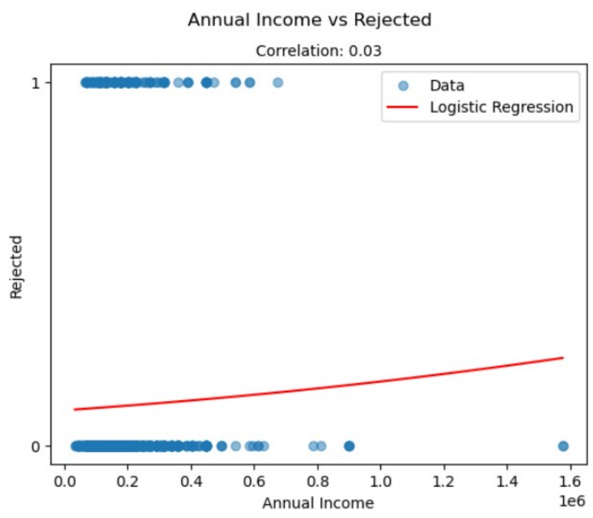
I subsequently conducted a correlation analysis to examine the relationships between the numerical features in the dataset. The results of the analysis yielded values ranging from -1 (strong negative correlation) to +1 (strong positive correlation), with the majority indicating weak correlations. In fact, all correlations between the numerical features and the target variable were found to be between -0.03 and 0.04, which suggests a negligible linear relationship. During this analysis, I also ensured that all NaN values were removed during the data preprocessing stage to achieve clearer results.

Then, I implemented a logistic regression model [6] to evaluate the predictive capability of numerical features in relation to the target variable Rejected. Each feature was analysed individually, which allowed for a more focused examination of its relationship with the target. The logistic regression model uses normalised feature values to maintain consistent scaling and prevent bias from varying feature ranges. It includes an intercept term and starts with weights initialized to zero, which are then iteratively optimised using gradient ascent. To avoid overfitting, the optimization process stops once the weight updates become negligible, ensuring the model has effectively converged.

Feature	**Correlation with Rejection**
Age	0.0439
Annual Income	0.0275
Employment Days	0.0314
Family Size	-0.0307
Children	-0.0216

Table 1: Correlation of Numerical Variables with Rejection

The results of the logistic regression [7] are visualized using logistic regression plots for each feature, with the original feature values on the x-axis and the target outcomes (”Rejected”) on the y-axis. A red logistic regression curve is overlaid on the data points, which represents the model’s probabilistic predictions. These plots clearly show that none of the analysed features exhibit a strong relationship with the target variable, which is highlighted by the flatness of the red curve in each plot.



Logistic Regression Diagram showcasing the relationship between Annual Income and Rejection

From the dataset analysis, skewness is evident in multiple variables, such as income and the number of children. Income shows significant variability, with approved users having a much larger range, leading to positive skewness due to extremely high values. Similarly, the number of children is heavily skewed, as the majority of individuals have none, with only a few cases showing large values (e.g. 14 children). Outliers were a notable issue during the analysis, especially in income and employment history. For example, employment history included extreme values, such as days employed exceeding 365243, which likely represented retirees.

However, some features show strong correlations with the target variable (approval status). Employment history stands out in this regard, as approved users tend to have longer and more variable employment durations, which suggests that stability in employment may positively influence approval. Property ownership also correlates strongly with approval, with approved users far more likely to own property than rejected ones. Education offers a more nuanced relationship, as ”Secondary / secondary special” dominates in both groups, but ”Higher education” appears more frequently among rejected users relative to their size. On the other hand, certain features provide limited utility. For instance, the number of children has low variance, as most individuals have none. Similarly, while there are slight differences in age between approved and rejected users, the overall distributions are too similar to suggest strong differentiation.

Finally, the target variable is also imbalanced, with a significantly larger number of approved users compared to rejected ones. This imbalance poses a challenge for ML models, as they tend to favour the majority class during training, potentially leading to biased predictions. For example, a model might achieve high accuracy by predominantly predicting approvals, but this would come at the cost of poor performance on the minority class (rejected users), resulting in low recall and precision for this group, as I have experienced in my case.

4 Methods

In this section, I describe the models, methods and evaluation approach used to tackle the problem. The methodology includes data preprocessing, the implementation of logistic regression and k-Nearest Neighbors (k-NN) and hyperparameter tuning. Additionally, I highlight notable decisions made during the project and outline key assumptions to ensure clarity.

I created a function called shuffle_and_split to randomly shuffle and split my dataset into training and testing sets. The function starts by setting a random seed with np.random.seed to guarantee consistent results for reproducibility. It then generates indices for the dataset, shuffles them to mix up the data, and calculates a split point based on the specified test_size, which determines the proportion of data allocated to the test set. Using these shuffled indices, the function slices the dataset into training and testing subsets for both features (X) and labels (y). By combining shuffling with controlled splitting, this function ensures the resulting subsets are unbiased.

Then, I performed a series of preprocessing steps to prepare the dataset for training and testing in a machine learning model [8]. First, I separated the dataset into numerical and categorical features, converting them into arrays for easier calculations. For the numerical data, I handled any missing values [9] by calculating the column-wise mean and filling in the gaps, ensuring no null values remain. Then, I encoded the categorical features using one-hot encoding to convert them into numerical representations that ML models can process. Afterwards, I combined the numerical and encoded categorical data into a single feature matrix, X, and performed additional preprocessing. This included normalizing the data [10] by calculating the mean and standard deviation of each feature and scaling them to have a mean of 0 and a standard deviation of 1. To account for bias in the model, I also added a column of ones to the dataset. I then removed any duplicate rows to ensure the dataset contained only unique entries. Finally, I split the processed dataset into training and testing sets using the shuffle_and_split function from the earlier cell. This ensured the data was properly randomized and divided.

I implemented a logistic regression model [11] to train on the dataset using gradient descent and tracked the cost reduction over time. The process started with the definition of a compute_cost function, which calculates the binary cross-entropy cost [12]. The function follows the logistic regression hypothesis

$$h_\theta(X) = \sigma(X \cdot \theta) = \frac{1}{1 + e^{-(X \cdot \theta)}}$$

, which applies a sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

to the dot product of the input features (X) and weights. This generates probabilities, and the cost function

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(X^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(X^{(i)})) \right]$$

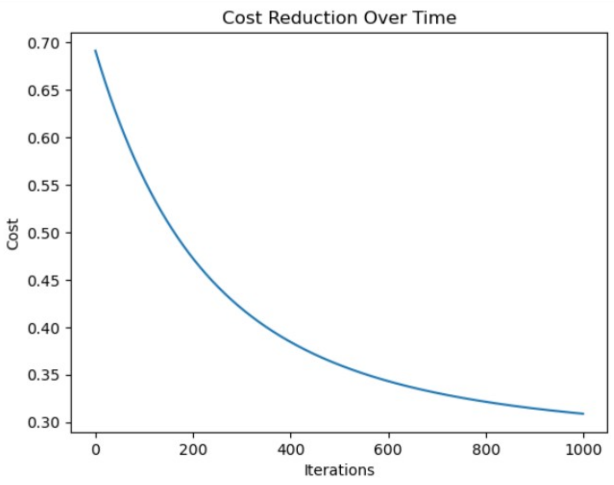
penalizes incorrect predictions by comparing these probabilities with the true labels (y). To ensure numerical stability and avoid issues with logarithms of zero, I added a small constant (1e-15).

The core of the training process was handled by the logistic_regression function, which implements gradient descent [13] to optimize the weights

$$\theta := \theta - \alpha \cdot \frac{1}{m} X^T (h_\theta(X) - y)$$

At each iteration, the model computes predictions $h_\theta(X)$, calculates the gradient of the cost function with respect to the weights, and updates the weights using the specified learning rate. The function also tracks the cost at every step, storing these values in a cost_history list. This iterative process allows the model to minimize the cost function over time, which allows it to make more accurate predictions.

To train the model, I initialised the weights to zeros and set the learning rate to 0.01 with 1,000 iterations. The training was performed on the X_train and y_train datasets, and the cost values were recorded after each weight update. Once training was complete, I visualised the cost reduction by plotting the cost_history against the number of iterations. The resulting graph showed a smooth decline, which indicates that the model was learning effectively.



Cost Reduction over time of the Binary Logistic Regression Model

I then defined a predict function to classify data using the trained logistic regression model and evaluated its accuracy on the test set. The function calculated the probabilities for each input sample using the sigmoid function and classified them as 1 or 0 based on a threshold of 0.5. Using this function, I predicted labels for X_test and compared them to the true labels (y_test) to calculate the model's accuracy.

I then implemented the core functions for the k-Nearest Neighbours (k-NN) [14] algorithm, with the euclidean distance function calculating the straight-line distance between two points in a multi-dimensional space, which serves as the measure of similarity between points. The k_nearest_neighbors function then takes the training dataset, its corresponding labels, a single test point, and the number of neighbours to consider. Then, for each point in the training set, it calculates the distance to the test point, pairs it with the point's class label, and sorts the distances to identify the k closest points. Finally, it predicts the class of the test point by finding the most frequent label among the k nearest neighbors, ensuring the prediction reflects the majority class within the local neighborhood.

I then extended the k-NN implementation to include functions for predictions and hyperparameter optimization. Indeed, the predict function applies the k_nearest_neighbors function to each point in a test dataset, aggregating the predictions into an array, which simplifies the process of making predictions on multiple test points at once. The find_best_k function allows to optimise the k parameter by evaluating the model's performance across a range of potential values. Using a validation dataset, the function predicts the labels for each value of k and calculates the accuracy of the predictions. It keeps track of the value of k that yields the highest accuracy, returning it as the optimal choice. This step ensures that the k-NN model performs the most efficient by tuning the most critical hyperparameter.

I split the training data into a smaller training set and a validation set using the shuffle_and_split function, with 70% for training and 30% for validation. I then evaluated different values of k (from 1 to 20) using the find_best_k function to determine the optimal k that achieved the highest accuracy on the validation set. The selected k value was then printed as the best choice for the k-NN model (8). Thereafter, I used the best k value (8) to predict labels for the test dataset with the predict function. After obtaining the predictions, I calculated the model's accuracy by comparing the predicted labels to the true test labels. Then the final test accuracy was displayed as the output (90.88%).

4.1 Notable Decisions

One notable decision I made in the project was the use of probabilities for classification. Instead of directly assigning class labels, I first calculated probabilities for each class, which allowed for more nuanced predictions. For instance, in logistic regression, probabilities were derived using the sigmoid function and a threshold (0.5) was applied to determine the final classification. Similarly, for k-NN, I calculated the proportion of positive labels among the k nearest neighbours to assign probabilities. This approach allowed me to use metrics like ROC-AUC later on.

The selected models, Logistic Regression and k-Nearest Neighbors, are appropriate for this dataset because in the case of logistic regression, it is particularly suited for binary classification tasks, making it well-suited for analysing the influence of applicant features on credit card approval. Conversely, k-Nearest Neighbors is a non-parametric model that does not assume a specific data distribution, which enables it to identify complex patterns within the dataset, which is advantageous given the combination of numerical and categorical features.

4.2 Key Assumptions

In this project, a few key assumptions were made to simplify the analysis. For logistic regression, I assumed the data was approximately linearly separable, as the model relies on this assumption for optimal performance. Additionally, I assumed that the features were appropriately pre-processed (e.g. normalized) to avoid scaling issues, particularly for algorithms like k-NN that are sensitive to distance metrics. For k-NN, I assumed there were no significant outliers in the dataset, as they could disproportionately affect the distance calculations and the predictions.

5 Results

In this section, the results of the two implemented models, logistic regression and k-Nearest Neighbors (k-NN), are presented and compared across key metrics. The analysis considers test accuracy, ROC-AUC [15] and confusion matrices [16] to assess how effectively each model performed on the dataset.

From the descriptive statistical analysis, several key insights emerged such as approved and rejected users showing notable differences in income, employment history, property ownership and education levels. Approved users exhibited greater variability in income and had longer, more stable employment histories, suggesting these factors may play a role in determining approval. Property ownership also correlated strongly with approval, as approved users were more likely to own property. However, certain features like age and number of children displayed minimal differences between the groups, which offered limited predictive value. Moreover, skewness and outliers were evident in variables such as income and employment history, which could have impacted model performance.

The test accuracy achieved by the logistic regression model [17] is 90.51%, which indicates a strong ability to classify the majority of samples correctly. However, accuracy alone can be misleading in imbalanced datasets. The ROC curve, with an AUC score of 0.52, reveals that the model struggles to distinguish between the two classes. An AUC of 0.5 represents random guessing, so the model's performance is only slightly better than this baseline. This suggests that while the accuracy appears high, the model's probabilistic predictions lack reliability for meaningful differentiation, likely due to the imbalanced target variable or perhaps the weak correlations between the features and the target. The confusion matrix provides a clearer picture of the model's performance. The top-left quadrant shows a high true negative rate (91%), indicating that the model is highly effective at identifying the majority class (approved users). However, the absence of true positives (bottom-right quadrant) and a 9% false negative rate suggest that the model completely fails to correctly classify rejected users. Overall, the logistic regression model's strength lies in its ability to classify the majority class accurately, but its limitations in handling the minority class are noteworthy.

The k-Nearest Neighbors (k-NN) classifier achieved a test accuracy of 90.88%, slightly outperforming the logistic regression model. This indicates the model effectively classifies most samples correctly. Moreover, the ROC curve for k-NN shows an AUC score of 0.63, which is better than random guessing and an improvement over the logistic regression's AUC of 0.52. This suggests that k-NN is better at distinguishing between the two classes and provides more reliable probability estimates. The confusion matrix shows a high true negative rate (91%), meaning the model accurately classifies most approved users. However, the true positive rate is zero, as the model fails to classify any rejected users correctly, just like the logistic regression model. This highlights the same issue with the dataset's class imbalance, where the minority class (rejected users) is underrepresented in predictions. While k-NN marginally improves probability-based metrics like AUC, it still struggles with recall for the minority class. Overall, the k-NN classifier performs marginally better than logistic regression in terms of AUC, which suggests it may offer slight improvements in distinguishing classes. In the end,

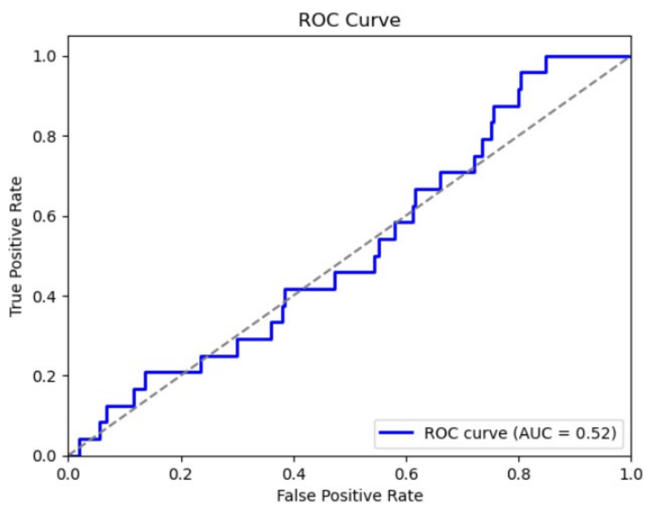


Figure 3: ROC Curve of logistic regression model

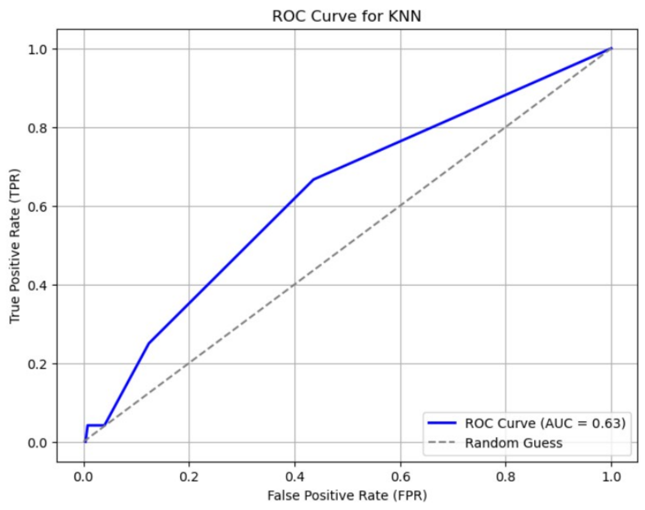


Figure 4: ROC Curve of KNN Classifier model

both models achieved high test accuracy, with k-NN slightly outperforming logistic regression (90.88% vs. 90.51%). This marginal difference suggests that both models are similarly capable of classifying the majority class (approved users) accurately. However, when considering the ROC-AUC score, k-NN demonstrates a clear advantage, achieving an AUC of 0.63 compared to logistic regression's 0.52. This indicates that k-NN is better at distinguishing between the two classes based on probability estimates, though both models remain far from ideal. The confusion matrix for both models also reveals a significant limitation: neither model successfully classified rejected users. Both achieve a high true negative rate of 91%, effectively identifying the majority class (approved users). However, neither model identifies any true positives, resulting in a true positive rate of zero and a failure to classify the minority class (rejected users). Hence, from a practical perspective, if the focus is on overall accuracy or probability-based metrics like ROC-AUC, k-NN may be the slightly better choice as a ML model.

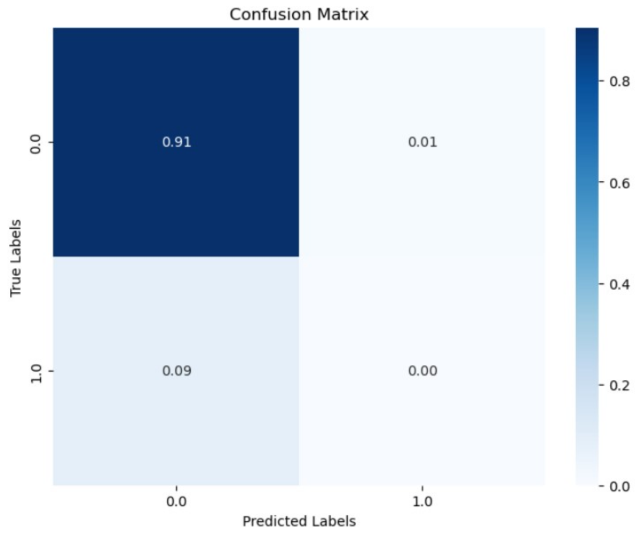


Figure 5: Confusion Matrix of logistic regression model

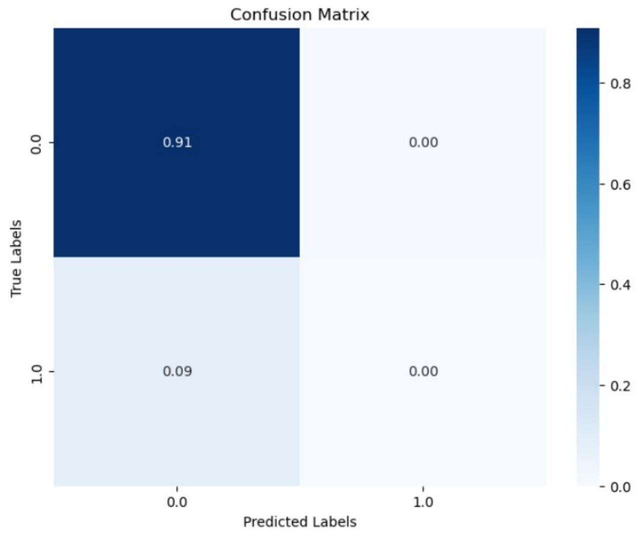


Figure 6: Confusion Matrix of KNN Classifier model

6 Conclusions

The project aimed to predict credit card approvals using a dataset of numerical and categorical features, such as income, employment history, property ownership and education. By implementing and comparing logistic regression and k-Nearest Neighbors (k-NN), my goal was to identify the most effective model for classifying applications as approved or rejected. Moreover, my main motivation for the project stemmed from my strong interest in applying machine learning to finance. I was particularly excited by the opportunity to analyse patterns in credit approval decisions and gain insights into the influence of certain features in rejection.

From the dataset analysis, several key insights emerged such as the fact that the data revealed notable skewness in features like income and the number of children, with income showing a significant range and outliers, particularly among approved users. Employment history also exhibited extreme values, with some representing retirees who were employed for extremely long durations. Correlation analysis showed weak relationships between most features and the target variable, with correlations ranging from -0.03 to 0.04, which indicated minimal linear relationship. Additionally, the target variable (approval vs. rejection) was heavily imbalanced [18], with far more approved cases than rejected ones. This imbalance posed challenges for the models, as it lead to biased predictions favoring the majority class.

In terms of methods and results, logistic regression and k-Nearest Neighbors (k-NN) were implemented and evaluated to predict credit card approvals. Logistic regression achieved a test accuracy of 90.51% but struggled with distinguishing between classes, as evidenced by an AUC of 0.52, which is only slightly better than random guessing. The confusion matrix for logistic regression highlighted its bias toward the majority class, correctly classifying most approved users but failing entirely to identify rejected users. The k-NN classifier, on the other hand, achieved a slightly higher accuracy of 90.88% and demonstrated better performance in terms of probability-based metrics, with an AUC of 0.63. While this showed an improvement in distinguishing between classes compared to logistic regression, k-NN still faced challenges in identifying rejected users, as indicated by the confusion matrix, which revealed a high true negative rate but no true positives.

Overall, K-NN performed slightlt better than logistic regression in terms of AUC, which suggests it was better at handling class distinctions. However, both models were limited by the dataset's unbalanced nature.

In regards to difficulties, the project faced several limitations that impacted the reliability of the models. A significant challenge was the imbalance in the dataset, with far more approved users than rejected ones. This imbalance led to biased predictions, where both logistic regression and k-NN struggled to identify rejected users effectively. A possible fix for this issue would have been to apply techniques like oversampling the minority class or generating synthetic samples using methods like SMOTE. These approaches could have balanced the dataset and improved the models ability to correctly identify both classes.

Another limitation was the lack of strong correlations between most features and the target variable. The correlation analysis revealed weak relationships, which likely hindered the models' predictive capabilities. With more time, feature engineering could have been used to perhaps transform existing features to better capture relationships with the target variable. For example, combining features like income and employment history to create a new metric or applying non-linear transformations could have improved the models performance.

Overall, while the project achieved meaningful insights and reasonable performance, addressing these limitations through dataset balancing and feature engineering could have significantly improved the models effectiveness.

7 Bibliography

References

[1] Queen Mary University of London, "Credit card approval dataset," 2025, provided as part of the final project of the MTH786P module.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[3] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56.

[4] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[5] GeekforGeeks, "One hot encoding in machine learning," 2024, (understanding the process and implementation of one-hot encoding for categorical data). [Online]. Available: https://www.geeksforgeeks.org/ml-one-hot-encoding/

[6] N. Appaji, "Simple logistic regression model in python, step by step," 2023, (a step-by-step explanation of logistic regression implementation in Python). [Online]. Available: https://niranjanappaji.medium.com/simple-logistic-regression-model-in-python-step-by-step-cd20fcbb5be5

[7] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression.* John Wiley & Sons, 2013.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning.* New York: Springer, 2006.

[9] GeekforGeeks, "Ml — handling missing values," 2024, (to explore methods for handling missing values in numerical and categorical features). [Online]. Available: https://www.geeksforgeeks.org/ml-handling-missing-values/

[10] ——, "Data normalization machine learning," 2024, (Explains how normalization scales data for better analysis and model performance). [Online]. Available: https://www.geeksforgeeks.org/what-is-data-normalization/

[11] D. J. . J. H. Martin, "Logistic regression," 2025, (foundational concepts related to logistic regression and its mathematical framework). [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/5.pdf

[12] D. Godoy, "Understanding binary cross-entropy / log loss: a visual explanation," 2018, (Describes log loss for binary classification with visual and mathematical explanations). [Online]. Available: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

[13] P. Meesala, "Step-by-step: Implementing gradient descent variants in python for beginners," (understanding gradient descent optimization and its implementation in logistic regression). [Online]. Available: https://prasad07143.medium.com/variants-of-gradient-descent-and-their-implementation-in-python-from-scratch-2b3cceb7a1a0

[14] MMdata, "k-nearest neighbors," 2019, (Implementation and tuning of the KNN Classifier Algorithm). [Online]. Available: https://www.kaggle.com/code/mmdatainfo/k-nearest-neighbors

[15] GeekforGeeks, "How to plot roc curve in python," 2024, (understanding how to plot the ROC curve and evaluate model performance using the AUC metric). [Online]. Available: https://www.geeksforgeeks.org/how-to-plot-roc-curve-in-python/

[16] "Understanding the confusion matrix in machine learning," 2025, (learn about the confusion matrix and its role in evaluating classification models). [Online]. Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

[17] M. P. LaValley, "Logistic regression," *Circulation*, vol. 117, no. 18, pp. 2395–2399, 2008.

[18] Jon, *Getting a low ROC AUC score but a high accuracy*, 2017, (understanding the discrepancies between accuracy and AUC metrics in classification tasks). [Online]. Available: https://stackoverflow.com/questions/47104129/getting-a-low-roc-auc-score-but-a-high-accuracy