

lendable

Fast loans at fair rates

Loan Application Classification Problem

OUTLINE

- Introduction
- Exploratory Data Analysis
 - Univariate Analysis
 - Bivariate Analysis
 - Feature Engineering
- Model Building and Evaluation
- Model Comparison
- Insights
- Model Governance
- Model Productionalization
- Conclusion



Introduction



Problem Statement: Develop a classification model to predict the likelihood of a loan application's success or rejection.

Business Impact: Accurate classification supports effective lending decisions, reduces financial risk, and enhances customer satisfaction.

Challenge: Balancing model precision (reducing risky approvals) and recall (minimizing missed opportunities).



Data Provided

- **Application Samples:** Loan applications from 2020 with labels indicating success or rejection.
- **Credit Features:** Attributes describing applicants' credit profiles.
- **Data Dictionary:** Documentation for understanding the dataset.



Proposed Approach

- **Exploratory Data Analysis (EDA):**
 - Understand the dataset and identify trends or issues (e.g., missing values, class imbalance).
 - Visualize distributions and correlations between features.
- **Feature Engineering:** Derive new variables, encode categorical features, and normalize numerical attributes.
- **Model Selection:**
 - Choose an appropriate classification algorithm (e.g., logistic regression, decision trees, or Neural Networks).
 - Address class imbalance using techniques like oversampling or class weights.
- **Model Evaluation:**
 - Use metrics such as precision, recall, F1-score, and ROC-AUC to assess performance.
 - Conduct cross-validation for robust results.
- **Decision Justification:** Explain trade-offs (e.g., prioritizing recall over precision).
- **Model Productionization:** Discuss how the model could be integrated into the lending process, ensuring scalability and compliance.

Univariate Analysis

Distribution of the individual variables :

- Categorical/ Ordinal– Frequency count
 - Employment Type, Application Date
- Numerical – Histogram, Density plot (Spread and modality), Box & Violin plot (Outlier detection)
 - Amount, Term, Age of oldest/youngest, No of active/default/settled accounts etc.,

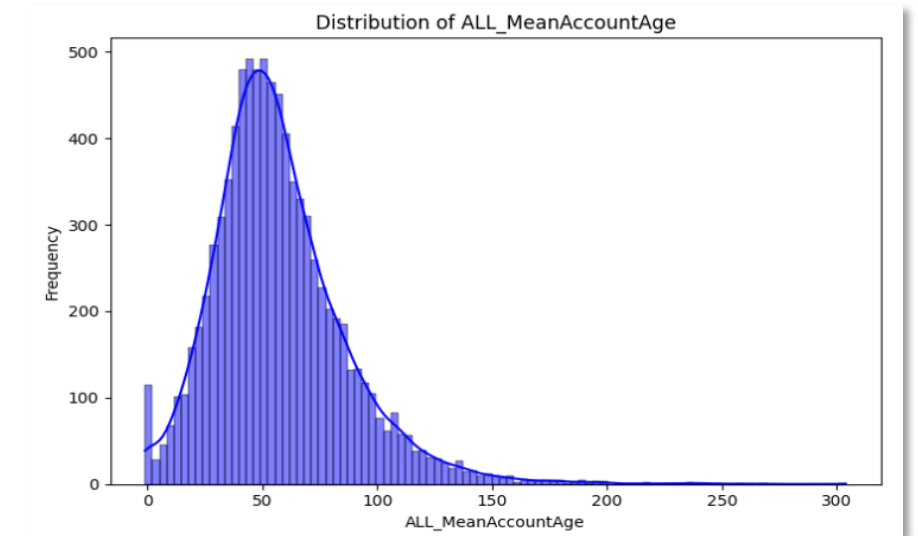
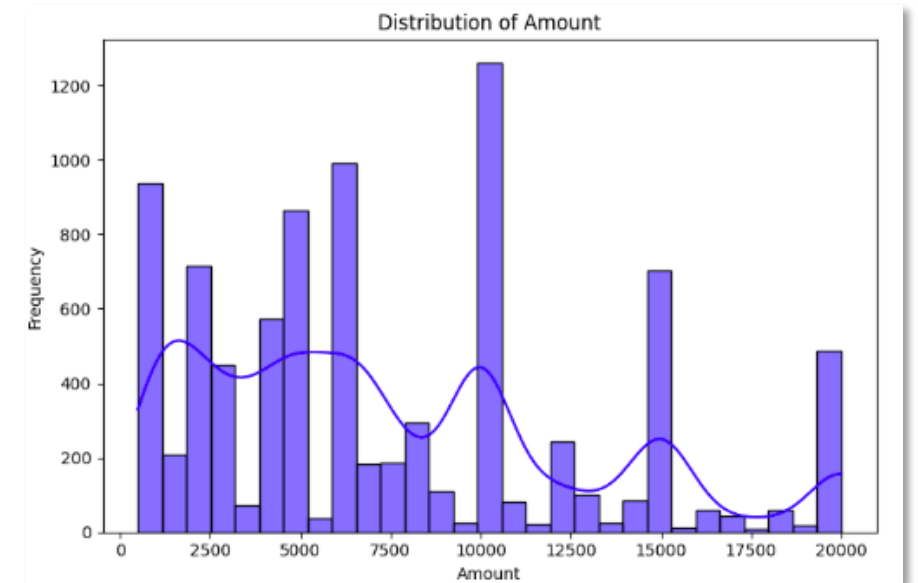
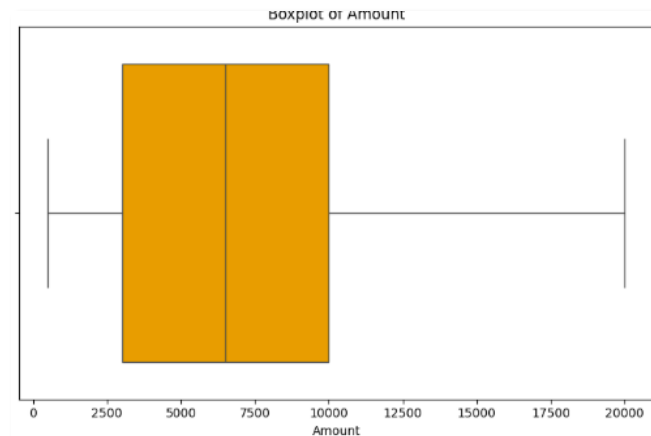
Insights :

- **87%** of people applying for loan are full time employees
- Applications are sought out across the year uniformly: ~25% in every Quarter and are also uniform across all days of Week (~15%)
- **89%** of the loan applications were rejected – **Heavy class imbalance**
- Amount Distribution is multi modal with multiple peaks; meaning the data represents different groups or patterns.
- No visible outliers are indicated in box plot for amount and term variables with **75%** of the applicants requesting below **10k**
- Mean account age is around **50** months

EmploymentType	
Employed - full time	0.871934
Employed - part time	0.064429
Self employed	0.047248
Retired	0.016390

Quarter	
1.0	0.249728
2.0	0.264690
3.0	0.243199
4.0	0.242383

Success	
0	0.895784
1	0.104216

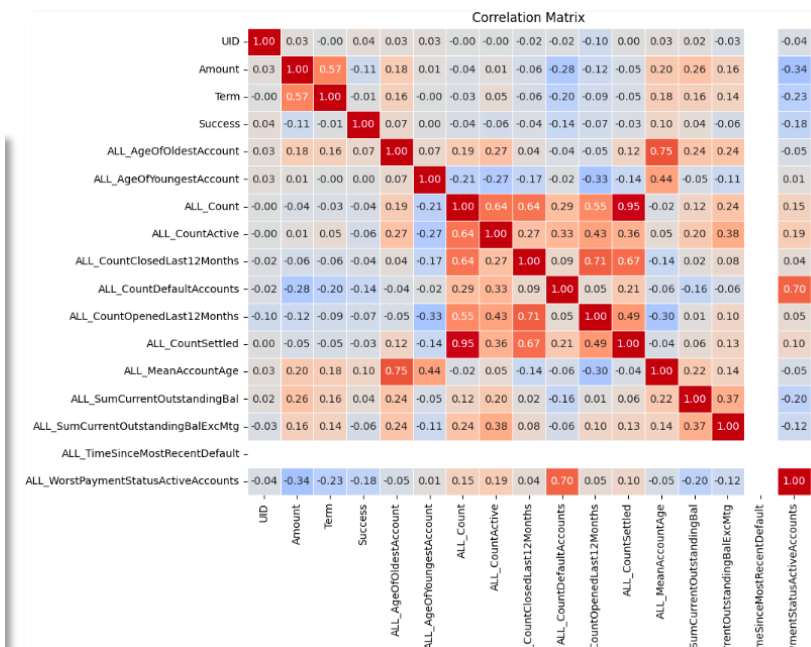
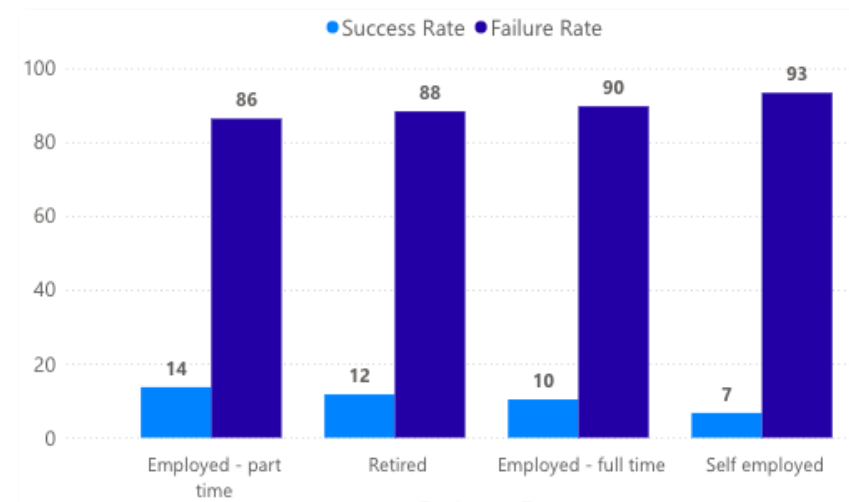
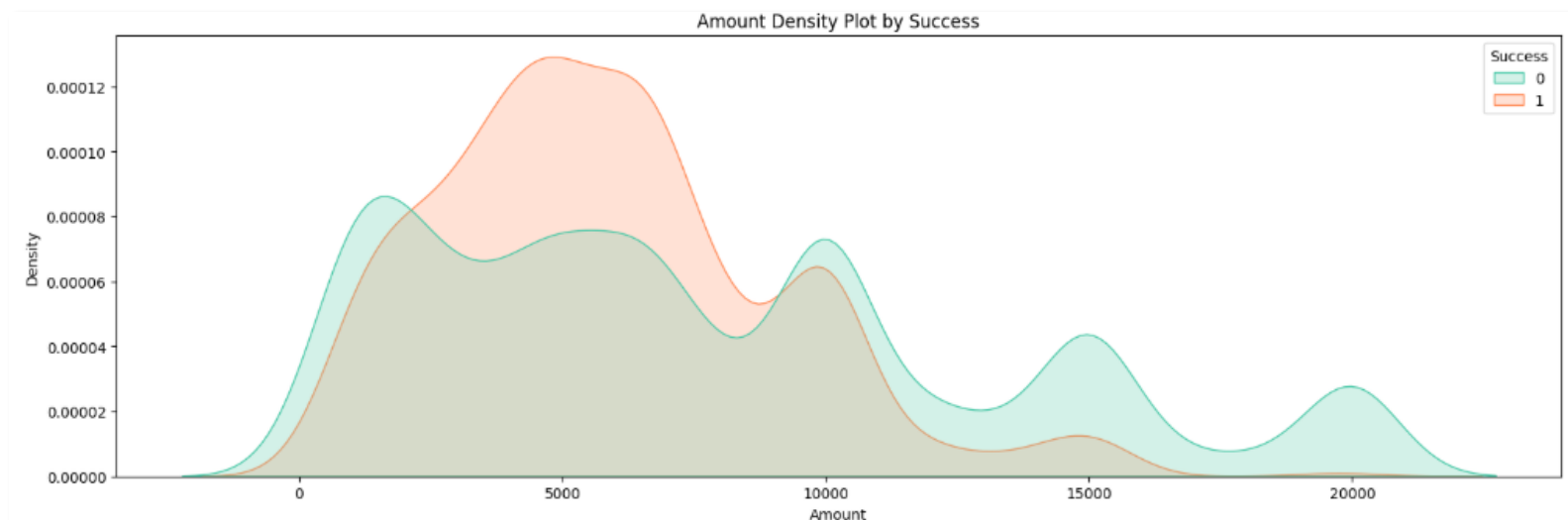


Bivariate Analysis

Distribution of the variables wrt target variable as well as individual correlations

Insights:

- Success (~90%) and failure rate (~10%) are **uniformly distributed across employment type** and follows the population distribution
- Similarly, Success rate distribution across numerical variable seems uniform
 - Amount around 5000 GBP has high probable success rate
 - High rejection rate for requestion term time of 60 months
- Similar correlation of all variables wrt target variable (~0 – 0.1); meaning all variables are required to predict target
- High correlation between **All_accounts** and **All_accounts_settled** (doesn't add extra info in predicting target)



Data Preparation and Feature Engineering

Data Cleaning :

- After merging two Excel sheets, the resulting dataset final_df has 8,847 rows and 20 columns. There are no missing values.
- **Date Column imputation**
 - The Date column was transformed into multiple features such as:
 - **Year**: Extracted the year value.
 - **Month**: Extracted the month value.
 - **Day of Week**: Derived the specific day (e.g., Monday, Tuesday).
 - **Quarter**: Determined the fiscal quarter (e.g., Q1, Q2).
 - **IsWeekend**: Added a binary flag to indicate weekends (1 for Saturday/Sunday, 0 otherwise).
- Removed Year (2020 is only value and uninformative of predicting) and ALL_TimeSinceMostRecentDefault (irrelevant data for prediction).

Categorical Variable Encoding :

- One Hot Encoding – 'EmploymentType', 'Month', 'DayOfWeek', 'Quarter' - this encoding transformed each unique category value into a binary vector.

Unstructured Variable Handling – Loan Purpose (NLP):

- **SBERT Sentence Transformers:**
 - The Loan Purpose column, a text-based unstructured variable, was converted into numerical embeddings using SBERT (Sentence-BERT).
 - SBERT, a pre-trained transformer model, generated a fixed 384-dimensional embedding for each text entry.
 - This representation captured the semantic meaning of each loan purpose, making it suitable for machine learning models.

Data Normalisation:

- All numerical columns were scaled using StandardScaler, which standardizes features by removing the mean and scaling to unit variance.
- **Training Data**: The scaler was fit to the training dataset to calculate the mean and standard deviation.
- **Test Data**: The test dataset was transformed using the previously fitted scaler to maintain consistency. This ensures that no information from the test set leaks into the model during training.

Data Splitting :

- Split into 80% training and 20% testing data: Training Data: 7,078 samples; Testing Data: 1,769 samples.
- **Stratified Split**: Data split ensures class balance using stratify=y. Maintains the proportion of labels across training and testing datasets.

Model Building and Evaluation

Modelling Approach :

- **SMOTE for Class Imbalance:**
 - Synthetic Minority Oversampling Technique (SMOTE) balances the training set by oversampling the minority class (Success=0).
 - Applied on each fold during cross-validation and the full training set for final evaluation.
- **Machine Learning Models:**
 - Logistic Regression, XGBoost, and Random Forest: Evaluated with and without unstructured variable (Loan Purpose) embeddings.
 - Models were trained on SMOTE-balanced data and unbalanced data for comparison.
- **Hyperparameter Tuning with Cross-Validation:**
 - Used GridSearchCV for XGBoost with the following hyperparameters:
 - `n_estimators`: Number of trees in the model (50, 100, 200).
 - `max_depth`: Maximum depth of each tree (3, 6, 10).
 - `learning_rate`: Step size shrinkage to prevent overfitting (0.01, 0.1, 0.2).
 - `subsample` and `colsample_bytree`: Proportions of samples and features used to grow trees.
- Evaluated using 3-fold cross-validation with accuracy as the scoring metric. Best model is chosen based on highest cross-validation accuracy.
- **Neural Network:** A 3-layer sequential model with:
 - **Input Layer:** 128 neurons with ReLU activation.
 - **Hidden Layer:** 64 neurons with ReLU activation.
 - **Dropout Layers:** To reduce overfitting (30% and 20% dropout rates).
 - **Output Layer:** Single neuron with sigmoid activation for binary classification.
 - Compiled using Adam optimizer and binary cross-entropy loss.
 - Trained for 20 epochs with a batch size of 32 and 20% validation split. Final accuracy was evaluated on the test set.

Evaluation Metrics:

- **Accuracy:** Measures overall correctness of predictions. Reported for all models and during cross-validation.
- **Precision:** Relevant when the cost of approving risky loans is high. Focuses on minimizing false positives (non-performing loans).
 - *Useful for maintaining a low-risk loan portfolio and reducing defaults.*
- **Recall:** Critical for maximizing customer inclusivity and fairness. Minimizes false negatives (missed potential customers).
 - *Prioritized when missing potential borrowers is costly (losing customers to competitors or losing customer trust).*
- **F1-Score:** Balances precision and recall for a comprehensive evaluation.

Model Performance Comparison (Class 1)

No.	Features	Sampling/ Tuning	Model	Accuracy	Precision	Recall	F1 - Score
1	Structured	No Over sampling	Logistic	90%	57%	7%	12%
2			Random Forest	90%	63%	10%	18%
3			XgBoost	90%	55%	29%	38%
4	Structured + Embeddings	No Over sampling	Logistic	90%	52%	7%	12%
5			Random Forest	89%	38%	2%	3%
6			XgBoost	90%	53%	28%	37%
7	Structured + Embeddings	SMOTE	Logistic	73%	24%	75%	37%
8			Random Forest	89%	42%	11%	18%
9			XgBoost	90%	51%	34%	41%
10			Gradient Boost	88%	46%	32%	38%
11	Structured + Embeddings	SMOTE + Hyper parameter Tuning	XgBoost	89%	47%	34%	40%
12	Structured + Embeddings	SMOTE + Hyper parameter Tuning + Cross Validation	XgBoost	89%	46%	32%	38%
13	Structured + Embeddings	SMOTE + Hyper parameter Tuning + Cross Validation	Neural Network	90%	51%	34%	41%

Model Insights

1. Accuracy vs. Recall Tradeoff:

- High accuracy models like Logistic Regression and Random Forest have poor recall, indicating difficulty in identifying Successful applications.
- SMOTE improves recall but sometimes sacrifices accuracy, especially for Logistic Regression.

2. Best Overall Model:

- **XGBoost with SMOTE** consistently delivers the best balance between precision, recall, and F1-Score (40%), making it the most effective model for this problem.

3. Impact of Feature Engineering:

- Adding embeddings enhances model performance, particularly for XGBoost, by improving recall without sacrificing much accuracy.

4. Hyperparameter Tuning:

- Hyperparameter tuning does not significantly alter XGBoost's performance but ensures consistency across cross-validation.

Model Governance Considerations

1. Data Governance:

- No sensitive data is used protecting user privacy.

2. Model Explainability:

- Feature importance using decision trees to explain key drivers of model predictions

3. Performance Monitoring:

- Tracking key metrics like accuracy, recall, and F1-score on production data. Data distributions needs to be continuously checked and should retrain models if necessary. Performance tracking can be automated using tools like MLflow or custom dashboards (Power BI).

4. Model Version Control:

- Git or MLflow to manage and store versions systematically.

5. Regulatory Compliance

- Adherence to laws and guidelines specific to finance industry or region (FCA, Basel regulations, etc.,)

6. Ethical and Fairness Considerations

- Model should be fair and does not reinforce biases in predictions. Sp bias audits needs to be conducted by analyzing outcomes across different demographic groups and Including diverse data in training to avoid under-represented biases.



Model Productionilization – Key Steps

1. Saving the Trained Model

- Ensures the model can be reused without retraining. Serialize the trained model into a file using Python's pickle library and save it as a .pkl file (xgboost_model.pkl).

2. Loading and Reusing the Model


- Use the pickle.load() function to deserialize and load the model when needed allowing quick deployment. The model can be used for predictions immediately after loading, ensuring consistency between development and production environments.

3. Integrate into Pipelines for APIs:

- Deploy the saved model in a production pipeline or as part of an API for real-time predictions. This step involves integrating the model into a live system to provide predictions on incoming data (Ex: Flask or FastAPI for REST API).

4. Monitor and Retrain

- Periodically monitoring the model's performance on new production data and retrain if performance degrades. Automate this process if possible. Over time, data distributions may change requiring updates to the model to maintain accuracy.



THANK YOU

