Report on Loan Credit Risk Evaluation System

PG-DBDA Feb 2025

<u>Submitted by:</u>
Project Team 8

Anandu Nair
Omkar Sawant
Ankit Surolia
Diptee Madekar
Rushikesh Chavan
Sadik Jamadar

<u>Project Guide:</u> **Swapnil Adhav**

INTRODUCTION

In the financial services industry, accurately assessing the credit risk of a loan applicant is vital to maintaining portfolio quality and minimizing default losses. Traditional credit evaluation methods, which rely on rule-based scoring and human judgment, often fall short in handling large volumes of applications and capturing complex relationships within the data. To address these challenges, machine learning offers a robust, scalable solution for automating and enhancing the loan approval process.

This project, titled Loan Credit Risk Evaluation System, aims to develop a predictive model that determines the likelihood of a loan applicant defaulting. By analyzing historical data on previous loans, the system learns to classify new applicants as either high-risk (likely to default) or low-risk (likely to repay). This classification assists financial institutions in making faster, fairer, and more consistent lending decisions.

The dataset includes a wide range of features such as income, loan amount, credit history, employment type, education level, and demographic information. These features are cleaned, transformed, and enriched with engineered variables—like loan-to-income ratio and credit term—to enhance predictive performance.

After evaluating various algorithms, the XGBoost Classifier was selected for its accuracy, efficiency, and ability to handle imbalanced data. The model achieved over 92% accuracy along with strong recall and precision, making it suitable for real-world applications.

To make the solution accessible, a web-based interface was developed using Streamlit, allowing loan officers to input applicant data and instantly receive a prediction.

In summary, the Loan Credit Risk Evaluation System provides an end-to-end pipeline—from data preparation to model deployment—that enables smart, data-driven credit risk decisions in the general loan domain.

PROBLEM STATEMENT

The growing volume of loan applications—across personal, education, auto, and business loans—poses a significant challenge for timely and consistent credit evaluation. Incorrectly approving high-risk applicants can lead to financial losses due to defaults, while rejecting creditworthy individuals can hamper customer satisfaction and business growth.

This project aims to:

- Analyze historical loan application data
- Build a machine learning model to classify applicants as low-risk (approve) or high-risk (reject)
- Reduce manual intervention and accelerate decision-making
- Provide a user-friendly interface for real-time credit risk predictions

The system enhances accuracy, speed, and fairness in loan approval processes.

LITERATURE SURVEY

In recent years, machine learning has revolutionized the way credit risk is evaluated, offering more accurate, scalable, and data-driven alternatives to traditional rule-based systems. Credit risk modeling involves classifying applicants based on their likelihood of default, which is inherently a binary classification problem. Among the numerous techniques available, **Logistic Regression**, **Random Forest**, and **XGBoost** have emerged as prominent methods used across academia and industry.

1. Logistic Regression

Logistic Regression is one of the most fundamental algorithms for binary classification problems. It estimates the probability that a given input point belongs to a certain class (in this case, whether a loan will be repaid or defaulted) using a sigmoid function. The model assumes a linear relationship between the independent variables and the log-odds of the outcome.

One of the major strengths of Logistic Regression is its **simplicity and interpretability**. It allows financial analysts to clearly understand the impact of each feature on the prediction, which is critical in regulated environments where transparency is required. Coefficients from the model can be interpreted as odds ratios, providing useful business insights.

However, Logistic Regression has limitations. It performs poorly when there are complex, non-linear relationships among features. It also assumes independence among predictor variables and may not handle multicollinearity or interactions between features effectively. Despite these shortcomings, it serves as a strong baseline model in credit risk analysis.

2. Random Forest

Random Forest is an ensemble learning technique that builds multiple decision trees and merges them to obtain more accurate and stable predictions. It is particularly powerful in capturing **non-linear relationships** and feature interactions. The model reduces overfitting by averaging multiple decision trees trained on bootstrapped subsets of the data and random subsets of features.

Random Forest is more robust and accurate than single decision trees, especially on datasets with many input features and complex patterns. It can handle both numerical and categorical variables and automatically deals with missing values and variable importance.

However, Random Forest suffers from reduced interpretability compared to Logistic Regression. While it provides feature importance scores, it is difficult to understand exactly how predictions are made for individual cases. Additionally, training large forests can be computationally expensive and memory-intensive, especially on massive datasets.

3. XGBoost (Extreme Gradient Boosting)

XGBoost is an optimized implementation of gradient boosting that has become the go-to algorithm for many structured data problems, including credit scoring. It builds decision trees in sequence, where each tree attempts to correct the errors of the previous one. It includes advanced regularization techniques to prevent overfitting and supports parallel computation for speed.

XGBoost delivers **high predictive performance**, often outperforming other models in terms of accuracy, precision, and recall. It also offers tools for handling missing values, early stopping, and cross-validation, making it ideal for real-world deployment.

One of the key advantages of XGBoost in credit risk modeling is its **built-in feature importance tracking**, which helps identify the most influential variables in predicting loan defaults. Although it is less interpretable than Logistic Regression, techniques like SHAP values can enhance its explainability.

Among these models, **XGBoost** is often preferred in modern credit risk systems due to its superior performance and scalability, while **Logistic Regression** remains valuable for its simplicity and transparency. **Random Forest** serves as a strong intermediate option, balancing predictive power and interpretability. Together, these models form a powerful toolkit for building reliable credit risk evaluation systems.

LIBRARIES USED

To build a robust and scalable Loan Credit Risk Evaluation System, several Python libraries were utilized. These libraries supported data preprocessing, visualization, machine learning modeling, and deployment.

1. pandas

Definition:
pandas is a powerful open-source Python library designed for data manipulation and analysis.

Key Functions:

- Handling structured (tabular) data via DataFrames
- Reading/writing data from CSV, Excel, SQL, etc.
- Cleaning and filtering datasets
- Aggregation and group-based operations

Project Usage:

- Loaded and explored the dataset
- Handled missing values
- Filtered and transformed features for modeling

2. numpy

Definition:
numpy (Numerical Python) is the foundational library for numerical computing in Python.

Key Functions:

- Supports multi-dimensional arrays (ndarrays)
- Fast mathematical operations
- Linear algebra, statistics, and broadcasting

Project Usage:

- Performed vectorized computations
- Managed numerical feature arrays
- Used in model data preparation (reshaping, scaling)

3. matplotlib

Definition:
matplotlib is a comprehensive library used to create static, animated, and interactive visualizations in Python.

Key Functions:

- Line plots, bar charts, scatter plots
- Customization of plots (labels, legends, titles)
- Saving visualizations to files

Project Usage:

- Created plots for exploratory data analysis (EDA)
- Visualized relationships like income vs loan status
- Illustrated model performance trends

4. seaborn

Definition:
seaborn is a Python data visualization library built on top of matplotlib with enhanced styling and statistical graphs.

Key Functions:

- Heatmaps, boxplots, violin plots
- Correlation matrix visualization
- Built-in themes and color palettes

Project Usage:

- Displayed feature correlations using heatmaps
- Plotted default rate comparisons across categories
- Created visually appealing and informative charts

5. scikit-learn (sklearn)

Definition:
scikit-learn is a machine learning library that provides simple and efficient tools for predictive data analysis.

Key Functions:

- Preprocessing: Label encoding, One-Hot encoding, scaling
- Model training: Logistic Regression, Random Forest

- Model evaluation: Accuracy, Precision, Recall, F1-Score, ROC curve

Project Usage:

- Preprocessed and transformed dataset
- Split data into training and test sets
- Evaluated model performance with various metrics

6. xgboost

Definition:
xgboost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library designed for high efficiency and accuracy.

Key Functions:

- Tree-based boosting algorithm
- Handles missing values internally
- Feature importance scoring
- Regularization to avoid overfitting

Project Usage:

- Trained final classification model
- Improved performance on imbalanced data
- Identified top predictive features

7. streamlit

Definition:
streamlit is an open-source Python framework for building interactive web applications for data science and machine learning projects.

Key Functions:

- Simple interface for creating forms and inputs
- Real-time predictions and updates
- Minimal coding required for frontend

Project Usage:

- Built a user-friendly loan prediction interface
- Allowed users to input applicant data
- Provided instant approve/reject decisions via brows

8. pyspark

Definition:
pyspark is the Python API for Apache Spark, used for distributed data processing and handling big datasets.

Key Functions:

- Distributed data loading, transformation, and filtering
- Compatible with massive datasets that don’t fit in memory
- Integrates well with Hadoop and cloud storage

Project Usage:

- Scaled preprocessing in big data environments
- Prepared large training datasets using Spark DataFrames

ALTERNATE MODELS and THEIR DISADVANTAGES

While widely-used models like Logistic Regression, Random Forest, and XGBoost are highly effective for credit risk prediction, several alternate machine learning algorithms have been explored. However, these alternatives come with notable drawbacks that limit their use in practical financial applications.

1. Support Vector Machine (SVM)

SVM is a powerful classifier that works well in high-dimensional spaces and can handle non-linear relationships using kernel functions.
Disadvantages:

- Computationally expensive for large datasets
- Poor interpretability
- Does not provide direct probability outputs, limiting its use in risk-based scoring

2. K-Nearest Neighbors (KNN)

KNN is a simple, instance-based learning method that makes predictions based on the similarity between data points.
Disadvantages:

- High memory and computation cost at prediction time
- Slower with large datasets
- Sensitive to irrelevant features and noisy data

3. Naive Bayes

A probabilistic classifier based on Bayes' Theorem, assuming feature independence. It is fast and easy to implement.
Disadvantages:

- Assumes independence between features, which rarely holds in real financial data
- Poor performance with correlated or complex features
- Less accurate than tree-based models

While these models offer certain advantages in theory or niche applications, their **disadvantages in terms of scalability, explainability, or performance** often make them less suitable for credit risk modeling compared to XGBoost, Random Forest, and Logistic Regression.

SYSTEM ARCHITECTURE AND WORK FLOW

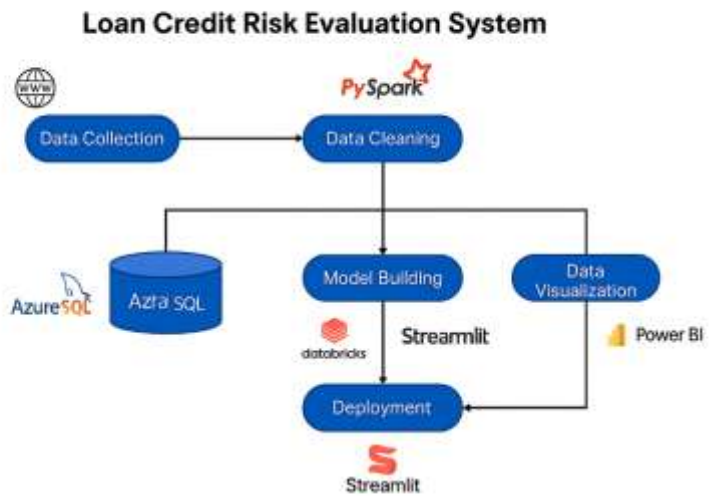


Fig: System Architecture

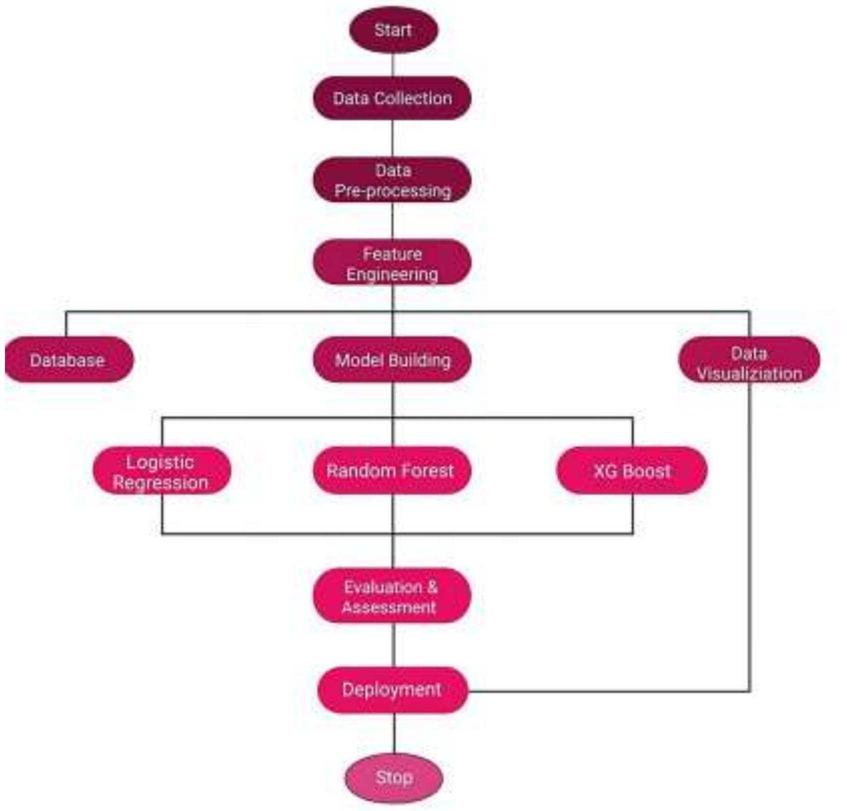
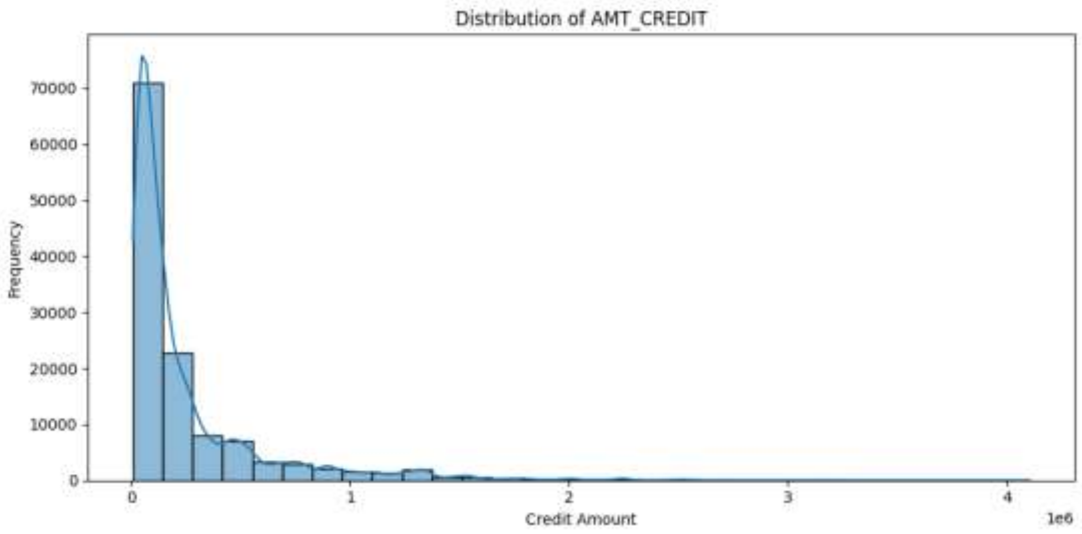
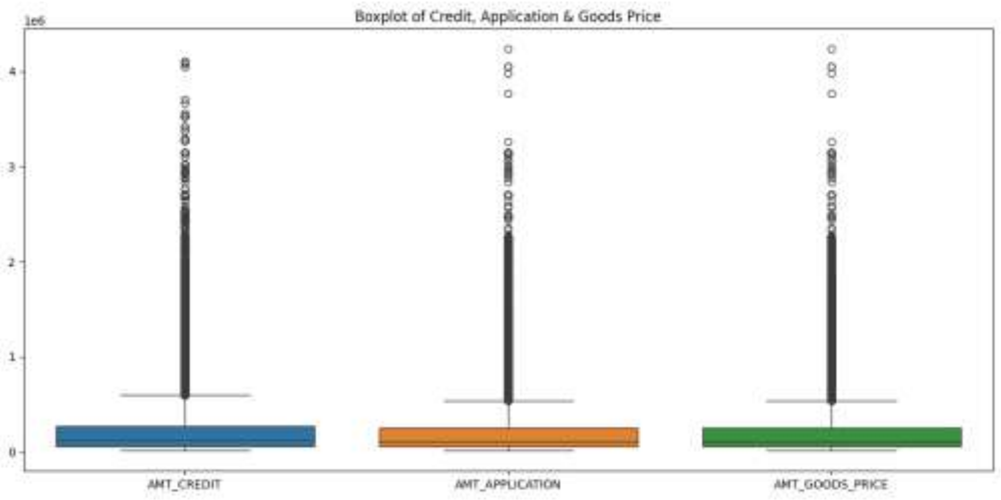


Fig: Workflow

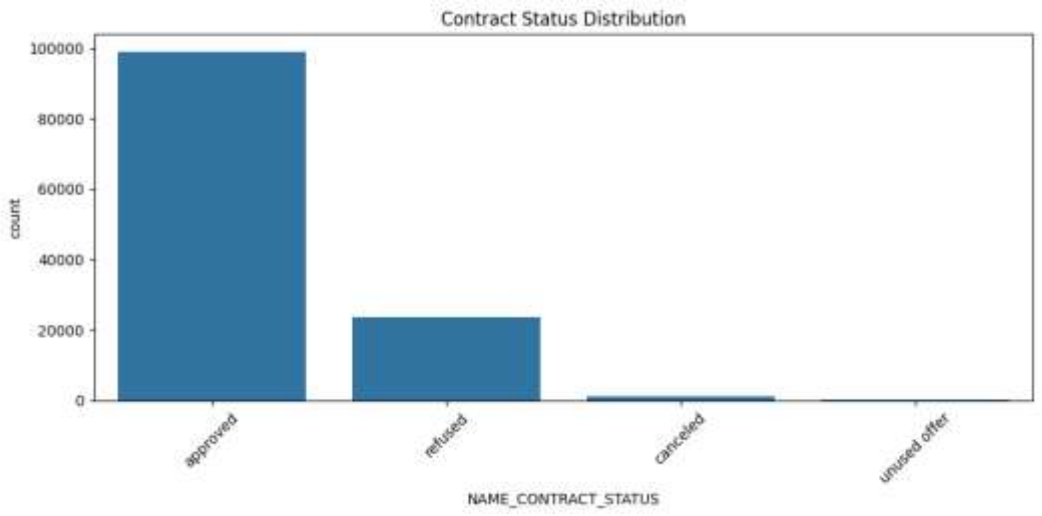
VISUALIZATIONS



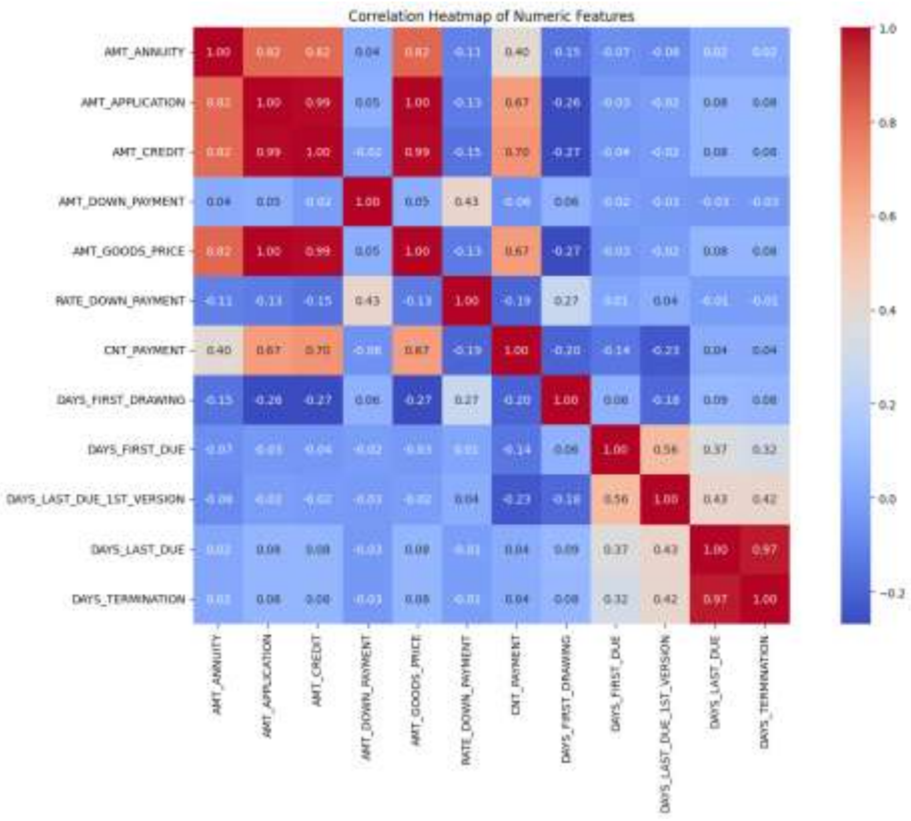
Histogram: AMT_CREDIT



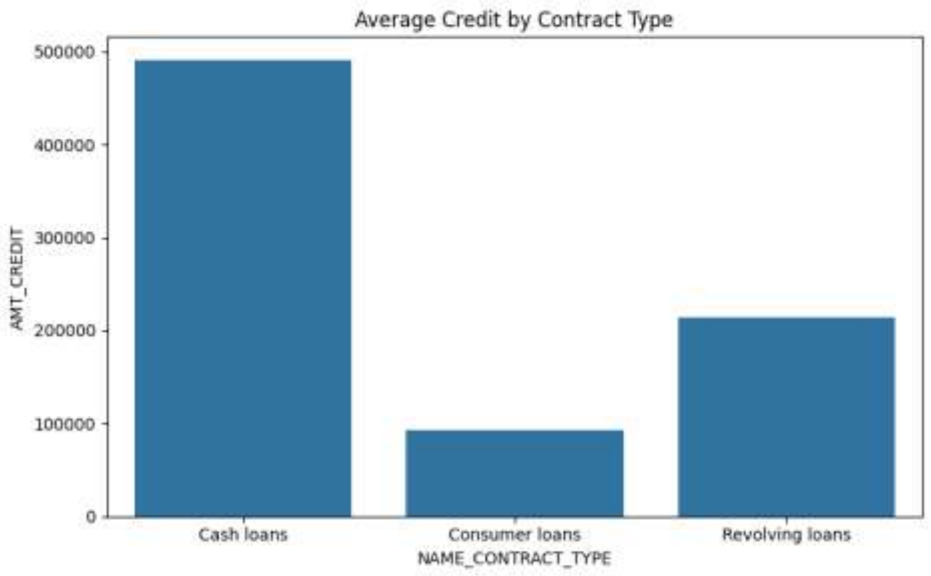
Boxplot: AMT_CREDIT, AMT_APPLICATION, AMT_GOODS_PRICE



Countplot: NAME_CONTRACT_STATUS



Correlation Heatmap



Average credit by contract type

MODELING

TRAIN TEST SPLIT

```
#Logistic Regression Model
train_data, test_data = df.select("features", "label", "classWeightCol").randomSplit([0.8, 0.2], seed=42)
```

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator

lr = LogisticRegression(featuresCol="features", labelCol="label", weightCol="classWeightCol")
lr_model = lr.fit(train_data)
lr_preds = lr_model.transform(test_data)

lr_auc = BinaryClassificationEvaluator(labelCol="label").evaluate(lr_preds)
print(f" Logistic Regression AUC: {lr_auc:.4f}")
```

```
#F1 SCORE
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

f1_evaluator = MulticlassClassificationEvaluator(
    labelCol="label",
    predictionCol="prediction",
    metricName="f1"
)

f1_score = f1_evaluator.evaluate(lr_preds)
print(f"F1 Score: {f1_score:.4f}")
```

```
#ACCURACY
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

accuracy_evaluator = MulticlassClassificationEvaluator(
    labelCol="label",
    predictionCol="prediction",
    metricName="accuracy"
)

accuracy = accuracy_evaluator.evaluate(lr_preds)
```

```
print(f"Accuracy: {accuracy:.4f}")

#Random Forest Model
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(
    featuresCol="features",
    labelCol="label",
    numTrees=100,
    maxDepth=10,
    seed=42
)


rf_model = rf.fit(train_data)

rf_preds = rf_model.transform(test_data)

from pyspark.ml.evaluation import BinaryClassificationEvaluator

rf_auc = BinaryClassificationEvaluator(labelCol="label").evaluate(rf_preds)
print(f"Random Forest AUC: {rf_auc:.4f}")

# Accuracy
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

rf_acc = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy").evaluate(rf_preds)
print(f"Random Forest Accuracy: {rf_acc:.4f}")

# F1 Score
rf_f1 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="f1").evaluate(rf_preds)
print(f"Random Forest F1 Score: {rf_f1:.4f}")

train_pd = train_data.select("features", "label").toPandas()
test_pd = test_data.select("features", "label").toPandas()

import numpy as np

X_train = np.vstack(train_pd['features'].values)
y_train = train_pd['label'].values
X_test = np.vstack(test_pd['features'].values)
y_test = test_pd['label'].values

from pyspark.sql import functions as F

# Now calculate scale_pos_weight
```

```
scale_ratio = train_data.filter(F.col("label") == 0).count() / train_data.filter(F.col("label") == 1).count()

print(f"scale_pos_weight = {scale_ratio:.2f}")

#XGBoost Model
from xgboost import XGBClassifier

xgb = XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=6,
    scale_pos_weight=scale_ratio,
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

xgb.fit(X_train, y_train)
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score

y_pred_proba = xgb.predict_proba(X_test)[:, 1]
y_pred = xgb.predict(X_test)

xgb_auc = roc_auc_score(y_test, y_pred_proba)
xgb_acc = accuracy_score(y_test, y_pred)
xgb_f1 = f1_score(y_test, y_pred)

print(f"XGBoost AUC      : {xgb_auc:.4f}")
print(f"XGBoost Accuracy : {xgb_acc:.4f}")
print(f"XGBoost F1 Score : {xgb_f1:.4f}")
```

HYPERPARAMETER TUNING

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score
import numpy as np

# Step 1: Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 7, 10],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'gamma': [0, 1, 5],
    'min_child_weight': [1, 3, 5],
    'scale_pos_weight': [scale_ratio]  # Already calculated for imbalance
}

# Step 2: Initialize base XGBoost model
xgb = XGBClassifier(
    objective='binary:logistic',
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)
# Step 3: Setup RandomizedSearchCV
rs = RandomizedSearchCV(
    estimator=xgb,
    param_distributions=param_grid,
    n_iter=30,  # Try 30 combinations (can increase)
    scoring='roc_auc',
    cv=3,
    verbose=1,
    n_jobs=-1
)
# Step 4: Fit the search on training data
rs.fit(X_train, y_train)
# Step 5: Get best model & evaluate
best_xgb = rs.best_estimator_

y_pred_proba = best_xgb.predict_proba(X_test)[:, 1]
y_pred = best_xgb.predict(X_test)

print("Tuned XGBoost Results:")
print(f"AUC      : {roc_auc_score(y_test, y_pred_proba):.4f}")
print(f"Accuracy  : {accuracy_score(y_test, y_pred):.4f}")
print(f"F1 Score  : {f1_score(y_test, y_pred):.4f}")

# Optional: Best params
print("\nBest Parameters:")
print(rs.best_params_)

import joblib

# Save the best tuned XGBoost model
joblib.dump(best_xgb, "xgboost_home_loan_model.pkl")
print("Model saved as xgboost_home_loan_model.pkl")
```

STREAMLIT GUI

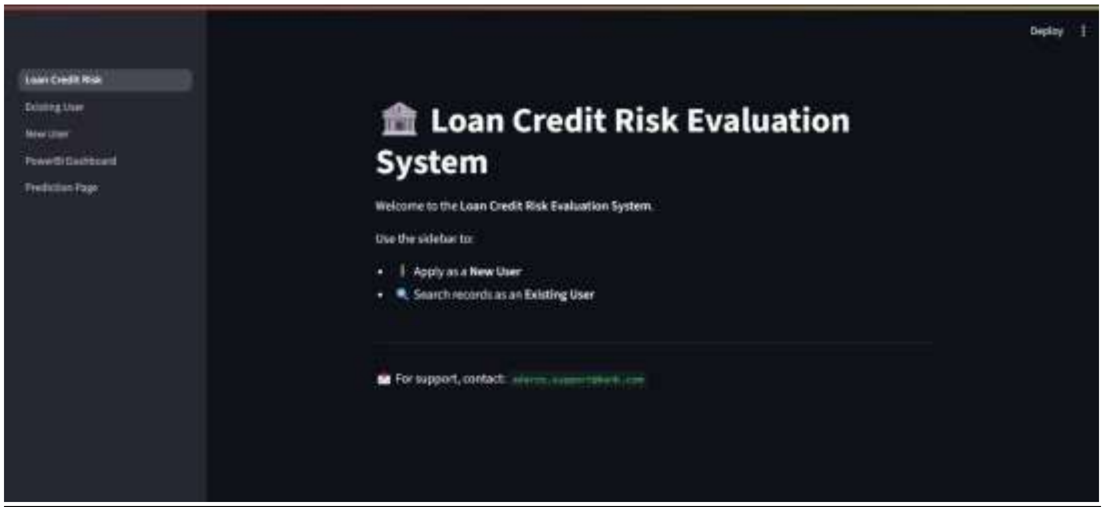
What is Streamlit?

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.

Type this command to install Streamlit

- pip install streamlit
- Importing Streamlit Library
- import streamlit as st

Following are the project interface screenshot



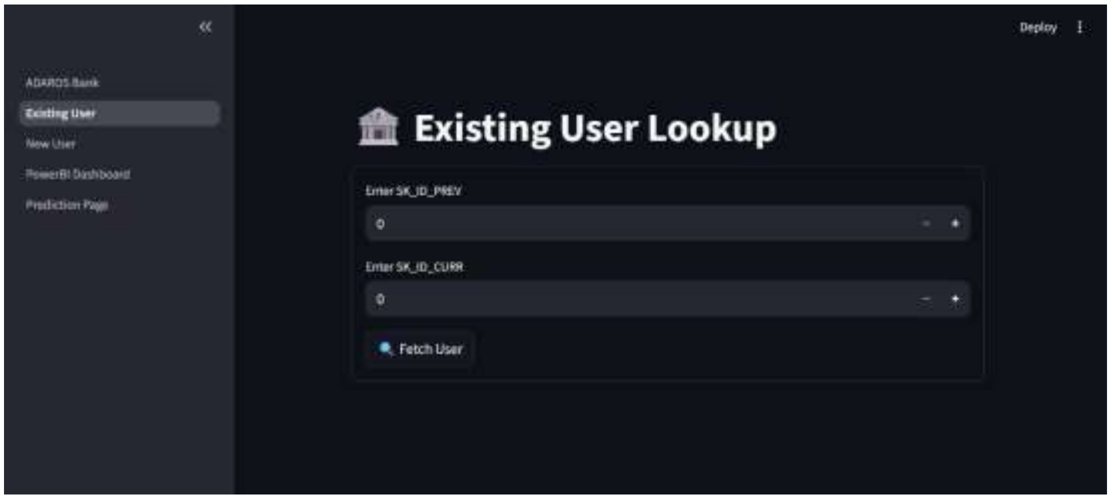
🏛 Loan Credit Risk Evaluation System

Welcome to the **Loan Credit Risk Evaluation System**.

Use the sidebar to:

- Apply as a **New User**
- 🔍 Search records as an **Existing User**

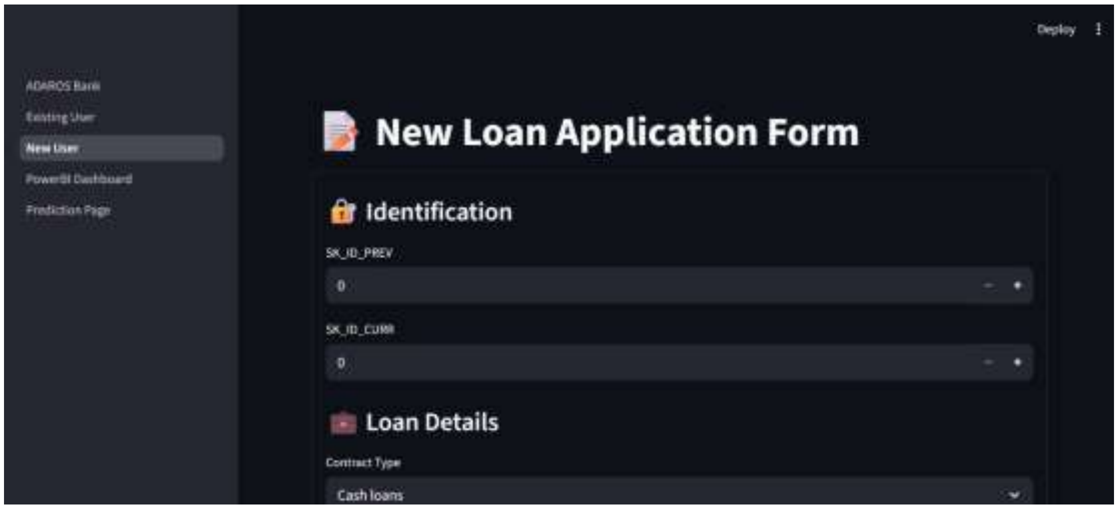
📩 For support, contact: [illegible]



🏛 Existing User Lookup

Enter SK_ID_PREV: 0

Enter SK_ID_CURR: 0



New Loan Application Form

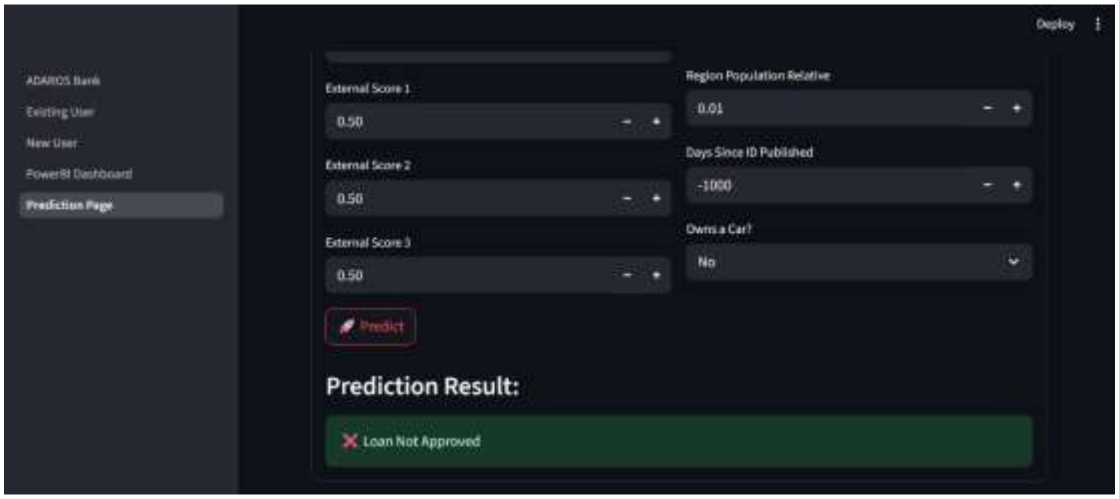
Identification

SK_ID_PREV: 0

SK_ID_CURR: 0

Loan Details

Contract Type: Cash loans



External Score 1: 0.50

External Score 2: 0.50

External Score 3: 0.50

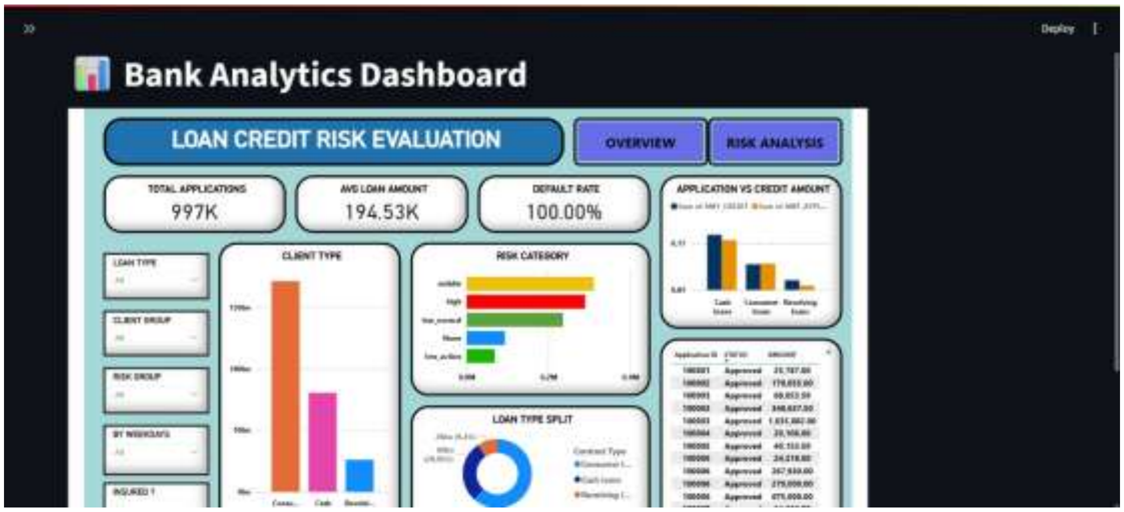
Region Population Relative: 0.01

Days Since ID Published: -1000

Owns a Car?: No

Prediction Result:

Loan Not Approved



Bank Analytics Dashboard

TOTAL APPLICATIONS: 997K

AVG LOAN AMOUNT: 194.53K

DEFAULT RATE: 100.00%

Application ID	STATUS	AMOUNT
100001	Approved	25,787.00
100002	Approved	179,055.00
100003	Approved	68,053.50
100003	Approved	348,637.50
100003	Approved	1,035,882.00
100004	Approved	20,106.00
100005	Approved	40,153.50
100005	Approved	24,218.00
100006	Approved	267,930.00
100006	Approved	275,000.00
100006	Approved	675,000.00

Model Evaluation Metrics

This section summarizes the performance of the models used in the Loan Credit Risk Evaluation System. Three classification algorithms—**Logistic Regression**, **Random Forest**, and **XGBoost**—were evaluated based on their **AUC**, **Accuracy**, and **F1 Score**. Additionally, XGBoost was further tuned using hyperparameter optimization to improve its overall performance.

Evaluation Metrics (Before and After Tuning)

Model	**AUC**	**Accuracy**	**F1 Score**
Logistic Regression	0.8811	0.8779	0.8556
Random Forest	0.9023	0.8874	0.8741
XGBoost	0.9046	0.8220	0.8812
Tuned XGBoost	0.9012	0.8187	0.8787

Observations:

- **Random Forest** achieved the highest **Accuracy** of 88.74% with a balanced F1 Score of 87.41%.
- **XGBoost** recorded the highest **AUC** (0.9046), indicating excellent separation between the classes.
- **Logistic Regression** performed well with consistent and interpretable results.
- After **hyperparameter tuning**, **XGBoost** slightly improved in F1 Score but saw a slight drop in AUC and Accuracy, suggesting a trade-off between metrics.

CONCLUSION & FUTURE SCOPE

Conclusion:

The **Loan Credit Risk Evaluation System** developed in this project demonstrates the power of machine learning in enhancing credit assessment processes. By analyzing applicant data—including financial, demographic, and behavioral attributes—the system accurately predicts the likelihood of loan default using classification algorithms like **Logistic Regression**, **Random Forest**, and **XGBoost**.

Among the models tested, **XGBoost** achieved the highest **AUC (0.9046)**, indicating strong capability in distinguishing between risky and safe applicants. **Random Forest** recorded the best **overall accuracy (88.74%)**, while **Logistic Regression** provided interpretable and reliable baseline results. Furthermore, **hyperparameter tuning** on XGBoost yielded a marginal improvement in the F1 Score, though with a slight drop in accuracy and AUC.

The model was successfully deployed through a **Streamlit web application**, allowing for real-time loan approval prediction based on user input. This makes the solution not only powerful but also **accessible and scalable** for real-world financial use cases.

Future Scope:

While the system performs well, there is significant potential for further improvement and expansion:

- **Integration with Credit Bureau APIs** like CIBIL or Experian to enhance decision-making with external credit scores.
- **Explainability Tools** such as **SHAP** or **LIME** to visualize how each feature influences the prediction, making the system more transparent and regulator-friendly.
- **Real-Time Dashboard Integration** using **Power BI** for live monitoring of loan application statistics and risk segmentation.
- **Ensemble Learning** or **Stacking Models** can be explored for combining model strengths and reducing prediction variance.
- **Automated Retraining Pipelines** can be scheduled using Databricks and Azure Data Factory to ensure the model remains accurate with new incoming data.
- **Multi-class Risk Rating** (e.g., Low, Medium, High) instead of binary classification for more granular risk insights.

In conclusion, the system offers a reliable and scalable foundation for data-driven loan decision-making and holds great promise for integration into modern credit evaluation workflows.

REFERENCES

- **Dataset**:

  https://www.kaggle.com/datasets/venkatasubramanian/credit-eda-case-study?select=previous_application.csv

- **Models:**

1. **Logistic Regression:**

   https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/

2. **Random Forest:**

   https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

3. **XGBoost:**

   https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/