# Credit Card Approval Prediction – Logistic Regression

## 1. Project Overview

This project builds a **Credit Card Approval Prediction** system using **Machine Learning**. The system automates the approval process, replacing manual reviews with a predictive model trained on historical application data.

## 2. Data Overview & Preprocessing

### 2.1 Dataset Features

The dataset contains 16 anonymized features, which likely correspond to:

- **Demographic Attributes** (Age, Gender, Citizenship, etc.)
- **Financial History** (Debt, Credit Score, Income, Years Employed, etc.)
- **Application Information** (Prior Defaults, Number of Bank Accounts, etc.)
- **Approval Status** (Target Variable: Approved or Denied)

### 2.2 Data Issues Identified

- **Mixed Data Types** (Categorical & Numeric Features)
- **Missing Values** (Denoted by `?` in some categorical fields)
- **Unscaled Features** (Varying numerical ranges impacting model performance)

### 2.3 Data Cleaning Steps

- Replaced `?` with **NaN** values for processing.
- **Mean Imputation** for missing numerical data.
- **Mode Imputation** for missing categorical values.
- **Label Encoding** to convert categorical values into numeric form.
- Dropped unimportant features such as **Zip Code & Driver's License**.
- **Feature Scaling** using `MinMaxScaler` to normalize numeric values.
-

## 3. Exploratory Data Analysis (EDA)

- **Summary Statistics** for understanding data distributions.
- **Correlation Analysis** to determine feature importance.
- **Class Imbalance Check**: Slight imbalance in approval vs. denial rates.

# 4. Model Development

## 4.1 Train-Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
```

## 4.2 Model Selection - Logistic Regression

- **Logistic Regression** was chosen due to its efficiency with correlated features.
- The model was trained on the **scaled training dataset**.

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(rescaledX_train, y_train)
```

# 5. Model Evaluation

## 5.1 Predictions

```
y_pred = logreg.predict(rescaledX_test)
```

## 5.2 Performance Metrics

```
from sklearn.metrics import confusion_matrix, accuracy_score
print("Accuracy:", logreg.score(rescaledX_test, y_test))
print(confusion_matrix(y_test, y_pred))
```

| Metric | Value |
|---|---|
| Accuracy | 83.77% |
| True Negatives | 92 |
| False Positives | 11 |
| False Negatives | 26 |
| True Positives | 99 |

# 6. Model Optimization - Hyperparameter Tuning

**GridSearchCV** was used to fine-tune **tol** and **max_iter** hyperparameters.

```
from sklearn.model_selection import GridSearchCV
param_grid = { 'tol': [0.01, 0.001, 0.0001], 'max_iter': [100, 150, 200] }
grid_model = GridSearchCV(estimator=logreg, param_grid=param_grid, cv=5)
grid_model_result = grid_model.fit(rescaledX, y)
print("Best Parameters:", grid_model_result.best_params_)
```

## Best Results:

| Hyperparameters | Best Value |
|---|---|
| max_iter | 100 |
| tol | 0.01 |
| Best Accuracy | 85.36% |

## 7. Conclusion & Business Impact

- The model **automates** credit card approvals with an **85.36% accuracy rate**.
- **Data preprocessing** significantly improved prediction performance.
- **Feature scaling & categorical encoding** were essential for model training.
- **Hyperparameter tuning** further optimized the logistic regression model.
- The system can help banks **speed up approvals**, reduce manual errors, and **enhance customer experience**.

## 8. Future Improvements

- Explore **ensemble models** (Random Forest, Gradient Boosting) for higher accuracy.
- Address **class imbalance** by oversampling minority classes.
- Deploy the model via **API or Web Application** for real-time predictions.

---