Hybrid Regression-Classification Model for Credit Default Risk

This project aims to build a hybrid machine learning model that combines **classification** (to predict whether a borrower will default) and **regression** (to estimate the potential default amount). The final output is the **Expected Loss**, which provides a more insightful measure of financial risk than classification alone.

Dataset

https://www.kaggle.com/datasets/wordsforthewise/lending-club

Use any of the following public datasets:

- LendingClub Loan Data
    - accepted_2007_to_2018Q4.csv.gz: contains accepted loans with repayment status.
    - rejected_2007_to_2018Q4.csv.gz: contains rejected loan applications (no repayment info)
- FICO Credit Score Dataset
- Kaggle Credit Default Risk Dataset

Only the **accepted loans data** can be used for building default classification and default amount regression models since it includes repayment outcome.

The **rejected loans data** does not contain repayment info and therefore cannot be used directly to predict default or default amount, but **can be used for advanced analysis** (see extension below).

Features typically include:

- Credit Score
- Income
- Loan Amount
- Loan Term
- Employment Status

Objectives

1. Train a classification model to predict **P(Default)** (probability of default).
2. Train a regression model to predict **Default Amount** (loss given default).
3. Calculate **Expected Loss = P(Default) × Default Amount**.

Methodology

Step 1: Preprocessing

- Handle missing **values**
- Encode categorical *variables*
- Normalize numerical features

Step 2: Modeling

Classification:

- Logistic Regression
- Random Forest
- SVM

Regression:

- Linear Regression
- Support Vector Regression
- Ridge / Lasso (optional)

Step 3: Evaluation

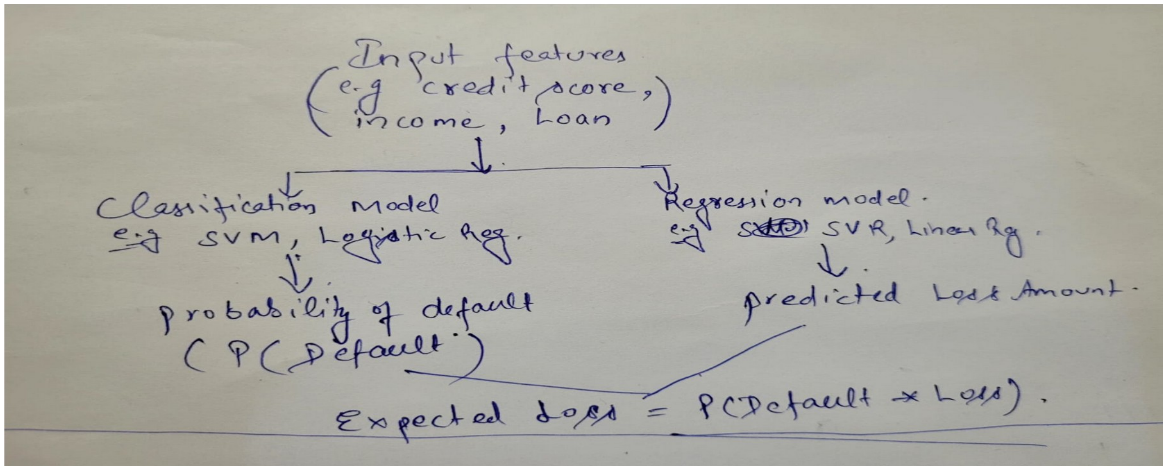
- **Classification Metrics**: Accuracy, F1-score, AUC-ROC
- **Regression Metrics**: RMSE, R^2
- **Combined Metric**: Expected Loss per borrower

Sample code: Python Toy Example

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.model_selection import train_test_split

# Sample Toy Dataset
data = pd.DataFrame({
    'credit_score': [700, 600, 550, 720, 580, 630, 660, 540],
    'income': [5000, 3500, 3000, 6000, 2800, 3200, 4000, 2600],
    'loan_amount': [2000, 2500, 3000, 1800, 3200, 2400, 2100, 3500],
    'default': [0, 1, 1, 0, 1, 1, 0, 1],
    'default_amount': [0, 1500, 2000, 0, 2200, 1200, 0, 3000]
})

X = data[['credit_score', 'income', 'loan_amount']]
y_class = data['default']
y_reg = data['default_amount']

# Train-Test Split
X_train, X_test, y_class_train, y_class_test = train_test_split(X, y_class,
test_size=0.25, random_state=42)

# Classification Model
clf = LogisticRegression()
```

```
clf.fit(X_train, y_class_train)
p_default = clf.predict_proba(X_test)[:, 1]

# Regression Model on Defaulters Only
X_reg_train = X_train[y_class_train == 1]
y_reg_train = y_reg[y_class_train == 1]
reg = LinearRegression()
reg.fit(X_reg_train, y_reg_train)

# Predict Default Amount
loss_amount = reg.predict(X_test)

# Expected Loss
expected_loss = p_default * loss_amount

# Results Table
results = X_test.copy()
results['P(Default)'] = p_default.round(2)
results['Predicted Loss Amount'] = loss_amount.round(2)
results['Expected Loss'] = expected_loss.round(2)
print(results)
```

Flow Diagram


Input features
(e.g credit score, income, Loan)
Classification Model
e.g SVM, Logistic Reg.
probability of default
(P(Default)
Regression model.
e.g SVR, Linear Reg.
predicted Loss Amount.
Expected Loss = P(Default × Loss).