Experiment No.5

Nupur Ghangarekar
D15C
Roll no 26

Aim: Support Vector Machine (SVM) for Credit Default Prediction

Theory

1. Dataset Source

Dataset Name: Default of Credit Card Clients Dataset

Source (Kaggle): https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset

2. Dataset Description

The dataset is used to predict whether a credit card client will default on their payment in the next month.

Dataset Overview:

- **Total Instances:** 30,000
- **Total Features:** 23 input features
- **Target Variable:** default.payment.next.month
- **Problem Type:** Binary Classification
- **Class Distribution:** Imbalanced (≈22% default cases)

Feature Categories

Demographic Features:

- LIMIT_BAL – Credit limit
- SEX
- EDUCATION
- MARRIAGE
- AGE

Payment History:

- PAY_0 to PAY_6 – Past repayment status

Bill Statements:

- BILL_AMT1 to BILL_AMT6

Previous Payments:

- PAY_AMT1 to PAY_AMT6

Target Variable:

- 1 → Default
- 0 → No Default

3. Mathematical Formulation of SVM

Support Vector Machine aims to find the optimal hyperplane that maximizes the margin between classes.

Linear Decision Boundary:

w^T x + b = 0

Where:

- w = weight vector
- b = bias
- x = feature vector

Optimization Objective:

$$\min \frac{1}{2}||w||^2$$

Subject to:

$$y_i(w^T x_i + b) \geq 1$$

Soft Margin (Real-world data):

$$\min \frac{1}{2}||w||^2 + C\sum \xi_i$$

Where:

- C = regularization parameter
- ξ_i = slack variables

Algorithm Limitations

- Computationally expensive for very large datasets
- Sensitive to choice of hyperparameters (C, kernel)
- Requires feature scaling
- Not ideal for extremely noisy datasets
- Hard to interpret compared to logistic regression

- Does not perform well when classes overlap heavily

4. Methodology / Workflow

Step-by-Step Process:

- Load dataset
- Drop unnecessary columns (ID)
- Handle missing values (Median Imputation)
- Feature Scaling (StandardScaler)
- Train-Test Split (Stratified)
- Model Training (Linear SVM)
- Hyperparameter Tuning (GridSearchCV)
- Model Evaluation
- Visualization of Results

Workflow Diagram:

Dataset
↓
Data Cleaning
↓
Imputation
↓
Feature Scaling
↓
Train/Test Split
↓
Model Training (SVM)
↓
Hyperparameter Tuning
↓
Evaluation (Accuracy, ROC, etc.)
↓
Visualization

Performance Analysis

Evaluation Metrics Used:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix
- ROC-AUC Curve

Interpretation:

- **Accuracy (~80–83%)** → Overall prediction correctness
- **Precision** → How many predicted defaulters were correct
- **Recall** → How many actual defaulters were identified
- **F1-score** → Balance between precision and recall
- **ROC-AUC** → Model's ability to separate classes

Since this dataset is imbalanced:

F1-score and Recall are more important than Accuracy

In financial risk modeling:

- Higher Recall reduces missed defaulters
- Balanced Precision avoids false alarms

Hyperparameter Tuning

Tuned Parameters:

Parameter	Description
C	Regularization strength
class_weight	Handles imbalance
max_iter	Convergence control

GridSearch Used:

C = [0.1, 1, 10]

- Cross-validation: cv = 3
- Scoring metric: F1 Score

Impact of Tuning:

- Improved F1-score
- Reduced overfitting
- Balanced bias-variance tradeoff
- Better minority class detection

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.svm import LinearSVC

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.metrics import (

    classification_report,
```

```
    confusion_matrix,
    accuracy_score,
    roc_curve,
    auc,
    precision_recall_curve
)
df = pd.read_csv("UCI_Credit_Card.csv")
# Reduce size for faster execution (optional)
df = df.sample(8000, random_state=42)
if "ID" in df.columns:
    df = df.drop("ID", axis=1)
X = df.drop("default.payment.next.month", axis=1)
y = df["default.payment.next.month"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler()),
    ('svm', LinearSVC(class_weight='balanced', max_iter=5000))
])

param_grid = {
    'svm__C': [0.1, 1, 10]
}
grid = GridSearchCV(
```

```
    pipeline,
    param_grid,
    cv=3,
    scoring='f1',
    n_jobs=-1
)
grid.fit(X_train, y_train)
print("Best Parameters:", grid.best_params_)
y_pred = grid.predict(X_test)
y_scores = grid.decision_function(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
plt.figure()
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
fpr, tpr, _ = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.3f}")
plt.plot([0,1],[0,1],'--')
plt.title("ROC Curve")
```

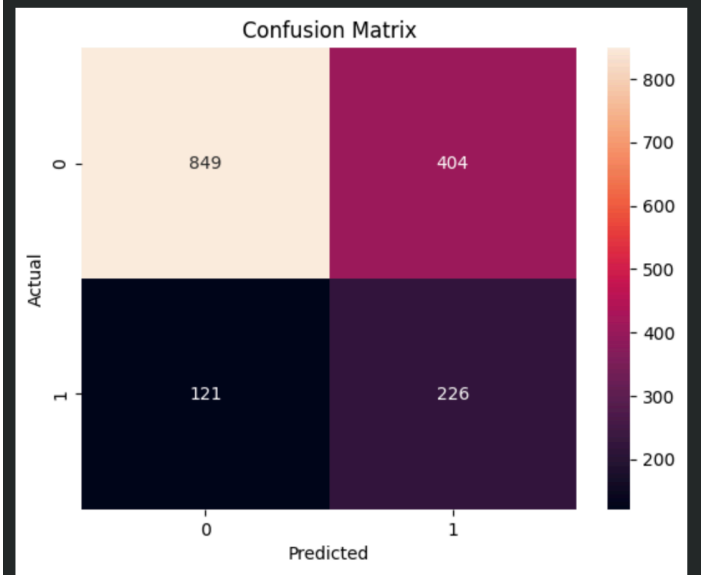
```
plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.legend()

plt.show()

precision, recall, _ = precision_recall_curve(y_test, y_scores)

plt.figure()

plt.plot(recall, precision)

plt.title("Precision-Recall Curve")

plt.xlabel("Recall")

plt.ylabel("Precision")

plt.show()

best_model = grid.best_estimator_.named_steps['svm']

coefficients = np.abs(best_model.coef_[0])

feature_importance = pd.Series(coefficients, index=X.columns)

feature_importance = feature_importance.sort_values(ascending=False)

plt.figure()

feature_importance.head(10).plot(kind='bar')

plt.title("Top 10 Important Features")

plt.show()
```

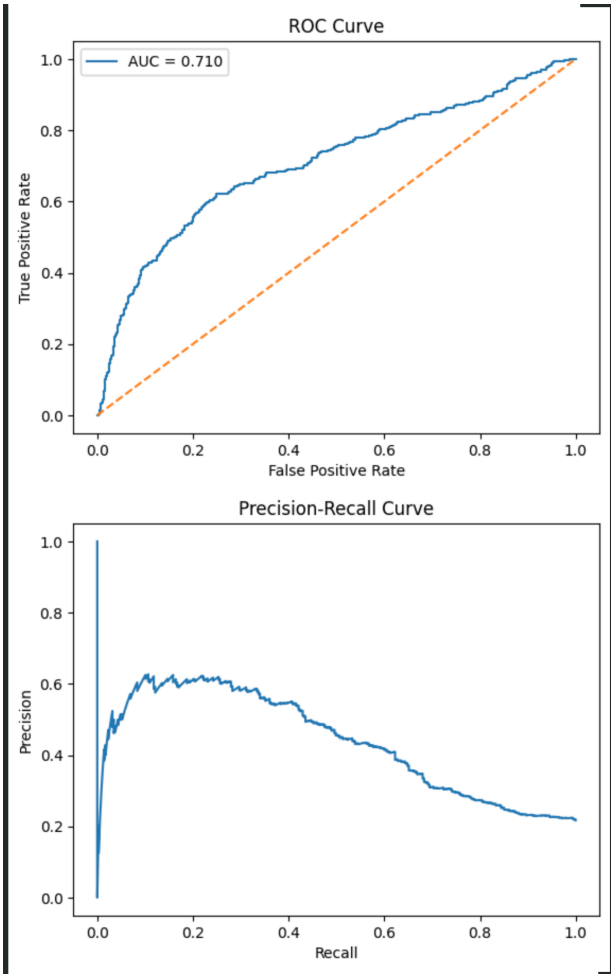
Conclusion

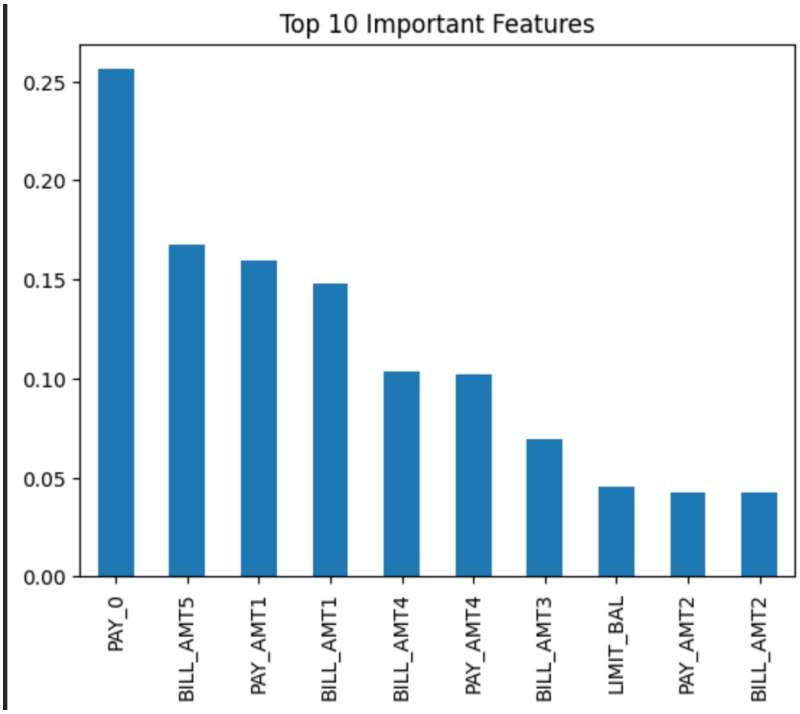
SVM is effective for structured financial datasets. LinearSVC performs well on large datasets. Hyperparameter tuning significantly improves generalization.Feature scaling and class balancing are essential for optimal performance. The model successfully predicts credit default with strong classification performance.

Google collab link: Link

```
Best Parameters: {'svm__C': 0.1}
Accuracy: 0.671875

Confusion Matrix:
 [[849 404]
 [121 226]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.68      0.76      1253
           1       0.36      0.65      0.46       347

    accuracy                           0.67      1600
   macro avg       0.62      0.66      0.61      1600
weighted avg       0.76      0.67      0.70      1600
```




ROC Curve
AUC = 0.710
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Precision-Recall Curve
Precision
Recall


Top 10 Important Features
0.25
0.20
0.15
0.10
0.05
0.00
PAY_0
BILL_AMT5
PAY_AMT1
BILL_AMT1
BILL_AMT4
PAY_AMT4
BILL_AMT3
LIMIT_BAL
PAY_AMT2
BILL_AMT2