ML & DL : Experiment - 5

Aim:

Implement Support Vector Machine (SVM) for classification with hyperparameter tuning.

Dataset Description:

The Credit Card Default Dataset is a real-world financial dataset that contains detailed information about 30,000 credit card clients of a Taiwanese bank. The dataset is designed to predict whether a customer will default on their credit card payment, making it a widely used benchmark for credit risk modeling and binary classification tasks. It captures customer demographics, credit limits, historical repayment behavior, billing amounts, and past payment records, all of which are critical indicators of financial reliability. Due to its predominantly numerical feature space, structured financial attributes, and clearly defined target variable, the dataset is particularly well-suited for Support Vector Machine (SVM) classification, especially when combined with proper feature scaling and hyperparameter tuning. The dataset reflects realistic banking scenarios and is commonly used in academic research and applied machine learning experiments focused on financial risk assessment.

- File Type: CSV (Comma Separated Values)
- Dataset Size: 30,000 rows × 25 columns (24 features + 1 target)
- Target Variable: dpnm (1: Default, 0: No Default)

Column Name	Data Type	Description
ID	Numerical (Integer)	Unique identifier for each client.
LIMIT_BAL	Numerical (Integer)	Credit limit assigned to the client (in New Taiwan Dollars).
SEX	Categorical (Integer)	Gender (1: Male, 2: Female).
EDUCATION	Categorical (Integer)	Education level (1: Graduate school, 2: University, 3: High school, 4: Others).
MARRIAGE	Categorical (Integer)	Marital status (1: Married, 2: Single, 3: Others).

Dataset Source: https://www.kaggle.com/datasets/mariosfish/default-of-credit-card-clients

---

SVM (Support Vector Machine)

Theory:

A Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression, particularly effective in high-dimensional and non-linearly separable data. SVM is grounded in statistical learning theory and operates on the principle of constructing an optimal decision

boundary—called a maximum-margin hyperplane—that best separates data points of different classes. Unlike traditional classifiers that focus on minimizing classification error alone, SVM seeks to maximize the margin, i.e., the distance between the separating hyperplane and the closest data points from each class. These closest points are known as support vectors, and they entirely determine the position and orientation of the decision boundary. Given a training dataset:

$$\{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathbb{R}^d,\ y_i \in \{-1, +1\}$$

SVM aims to find a hyperplane defined by:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

that separates the two classes while maximizing the margin.

Limitations:

1. **High Computational Cost for Large Datasets:** SVM training involves solving a quadratic optimization problem, whose time complexity typically ranges between $O(n^2)$ and $O(n^3)$ where n is the number of training samples. As a result, SVM becomes computationally expensive for large datasets, especially when using non-linear kernels like the RBF kernel. Memory consumption also increases significantly because kernel-based SVMs require storing and computing the kernel matrix of size n x n.
2. **Sensitivity to Hyperparameter Selection:** SVM performance is highly dependent on the choice of hyperparameters, particularly:
    a. C (regularization parameter)
    b. Kernel type
    c. γ (for RBF kernel)

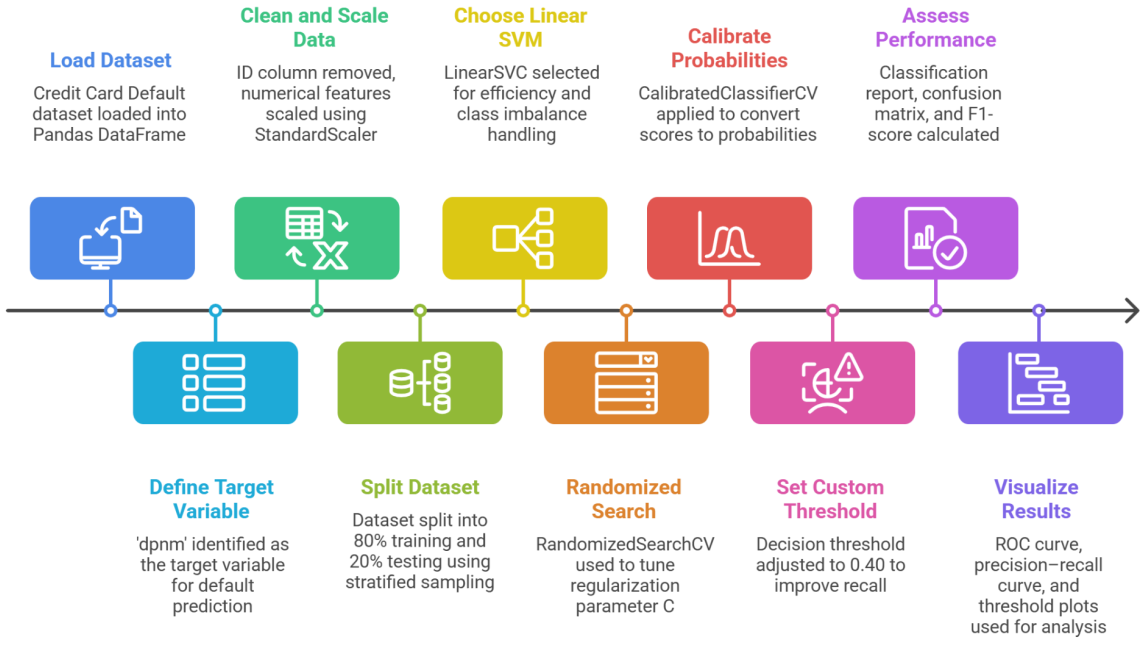
    Poorly chosen hyperparameters can lead to overfitting (large C, large γ) or underfitting (small C, small γ). As a result, SVM almost always requires extensive hyperparameter tuning, increasing both computational cost and model development time.
3. **Lack of Interpretability:** SVM is often regarded as a black-box model, especially when non-linear kernels are used. Unlike decision trees or logistic regression, SVM does not provide easily interpretable rules or feature importance measures. This limitation is particularly significant in financial and banking applications, where model transparency and explainability are often required for regulatory compliance.
4. **Poor Performance with Highly Overlapping Classes:** When class distributions significantly overlap, the margin maximization principle of SVM becomes less effective. In such cases, many data points violate margin constraints, increasing reliance on slack variables and reducing the model's ability to generalize. This issue is common in credit risk datasets, where good and bad credit applicants may share similar financial profiles.

Workflow:

1. **Data Collection:** The Credit Card Default dataset (credit_card_data.csv) is loaded into a Pandas DataFrame. The dataset contains customer demographic information, credit limits, repayment history, bill statements, and past payment amounts, which are used to predict whether a customer will default on credit card payment.
2. **Target Identification:** The target variable is identified as dpnm, representing credit card default status (1 = default, 0 = non-default). All remaining columns are treated as input features.
3. **Data Preprocessing:** The customer identifier column (ID) is removed as it does not contribute to prediction. The dataset contains only numerical features, allowing direct use without categorical

encoding. Feature scaling is performed using StandardScaler to standardize all input variables, which is essential for Support Vector Machines to ensure stable and efficient optimization.

4. **Train–Test Split:** The dataset is split into 80% training data and 20% testing data using stratified sampling to preserve the original class imbalance. A fixed random state is used to ensure reproducibility of results.
5. **Model Selection (Linear SVM):** A Linear Support Vector Machine (LinearSVC) is selected due to its computational efficiency on large datasets and high-dimensional feature space. Class imbalance is handled using class_weight='balanced', ensuring higher importance is given to the minority default class during training.
6. **Hyperparameter Tuning with Randomized Search:** Hyperparameter tuning is performed using RandomizedSearchCV with 3-fold cross-validation. The regularization parameter C is sampled from a log-uniform distribution to efficiently explore both small and large regularization strengths. The F1-score is used as the optimization metric to balance precision and recall for the imbalanced dataset.
7. **Probability Calibration:** Since LinearSVC does not natively provide probability estimates, CalibratedClassifierCV with sigmoid calibration (Platt scaling) is applied to convert decision scores into well-calibrated class probabilities. This step enables threshold tuning and probability-based evaluation.
8. **Decision Threshold Optimization:** Instead of using the default probability threshold (0.50), a custom threshold of 0.40 is applied to improve recall and F1-score for the default class. Threshold tuning allows better control over the trade-off between false positives and false negatives, which is critical in credit risk modeling.
9. **Model Evaluation:** The calibrated and threshold-optimized model is evaluated on the test set using: Classification Report (precision, recall, F1-score), Confusion Matrix, Accuracy Score, F1-score for the default class. These metrics provide a comprehensive assessment of model performance under class imbalance.
10. **Visualization and Performance Analysis:** Graphical evaluation techniques such as confusion matrix visualization, ROC curve, precision–recall curve, and F1-score versus decision threshold plots are used to analyze classification behavior and justify the selected threshold.



Performance Analysis:

The performance of the optimized Linear Support Vector Machine (SVM) model for credit card default prediction is evaluated using Accuracy, Confusion Matrix, and class-specific metrics including Precision, Recall, and F1-score. Given the class imbalance in the dataset, special emphasis is placed on the default (positive) class.

1. **Overall Accuracy:** The calibrated Linear SVM model achieves a test accuracy of 81.13%, indicating that the model correctly predicts the default status for approximately 81 out of every 100 customers. This demonstrates strong overall classification performance, especially considering the inherent imbalance in credit default data.
2. **Class-Specific Performance (Precision, Recall, and F1-Score)**
    a. Class 0 (Non-Default Customers)
        i. Precision: 0.84
        ii. Recall: 0.94
        iii. F1-score: 0.89

    The model performs exceptionally well in identifying non-default customers. A recall of 94% indicates that the vast majority of reliable customers are correctly classified, minimizing unnecessary false alarms and ensuring stability in credit approval decisions.

    b. Class 1 (Default Customers)
        i. Precision: 0.62
        ii. Recall: 0.37
        iii. F1-score: 0.46

    The model shows moderate precision but lower recall for default customers. A precision of 62% means that when the model predicts a default, it is often correct. However, a recall of 37% indicates that some default cases remain undetected, which is a common challenge in highly imbalanced financial datasets. Threshold tuning and class weighting help mitigate this issue, but further improvements may require more complex models.
3. **Confusion Matrix Breakdown:** The confusion matrix provides deeper insight into prediction behavior:
    a. True Negatives (4377): Non-default customers correctly classified
    b. True Positives (491): Default customers correctly identified
    c. False Positives (296): Non-default customers incorrectly predicted as defaulters
    d. False Negatives (836): Default customers missed by the model

    The relatively low number of false positives helps prevent denying credit to trustworthy customers, while the presence of false negatives highlights the difficulty of capturing all risky clients using a linear decision boundary.
4. **F1-Score for Default Class:** The F1-score of 0.4645 for the default class reflects a balanced trade-off between precision and recall. This metric confirms that the combination of class balancing, probability calibration, and threshold optimization improves minority-class performance compared to standard accuracy-focused training.

Hyperparameter Tuning:

Support Vector Machines (SVMs) are sensitive to hyperparameter selection, particularly the regularization parameter C, which controls the trade-off between margin maximization and classification error. Proper tuning is essential to achieve good generalization, especially for imbalanced datasets like credit card default prediction.

The code uses RandomizedSearchCV with 3-fold cross-validation to efficiently explore the hyperparameter space while reducing computational cost. Model performance is optimized using the F1-score, which balances precision and recall for the minority (default) class.
Tuned Hyperparameter:

- **C (Regularization Parameter):** Sampled from a log-uniform distribution (0.01 to 10) to evaluate both weak and strong regularization settings.

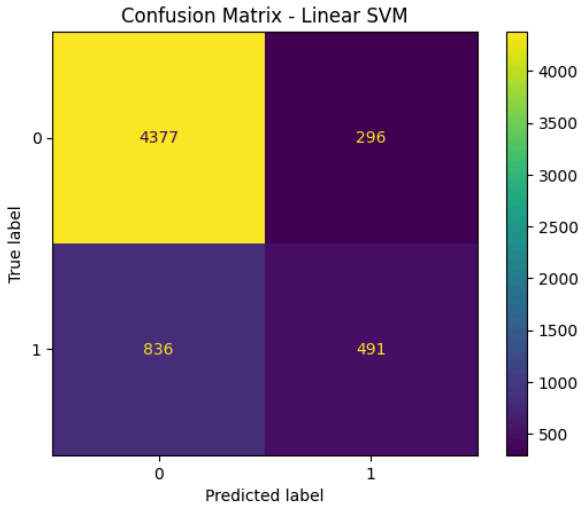
The optimal value of C provides the best balance between underfitting and overfitting. This tuned model is then used for probability calibration and threshold adjustment, leading to improved detection of default cases while maintaining stable overall accuracy.

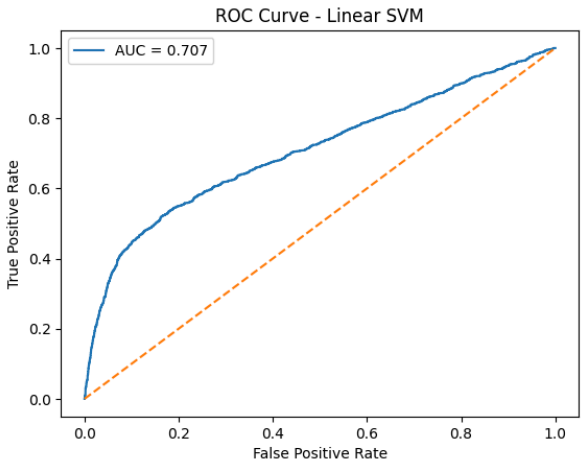
Code & Output:

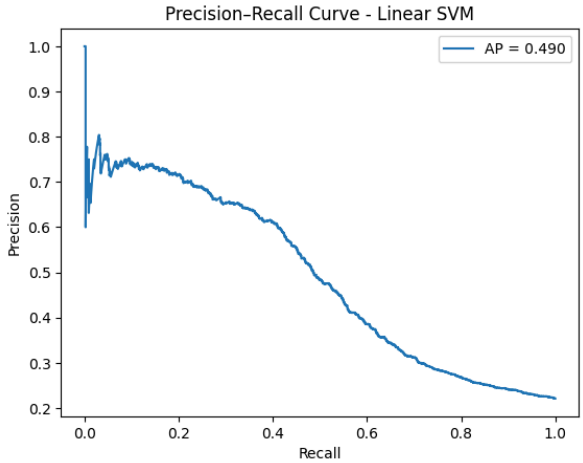
```
import pandas as pd
import numpy as np

from sklearn.model_selection import
train_test_split, RandomizedSearchCV
from sklearn.preprocessing import
StandardScaler
from sklearn.svm import LinearSVC
from sklearn.calibration import
CalibratedClassifierCV
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    accuracy_score,
    f1_score
)
from scipy.stats import loguniform

df =
pd.read_csv('credit_card_data.csv')

df.drop('ID', axis=1, inplace=True)

X = df.drop('dpnm', axis=1)
y = df['dpnm']

X_train, X_test, y_train, y_test =
train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42
)

scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled =
scaler.transform(X_test)

# 4. Base Linear SVM
base_svm = LinearSVC(
    class_weight='balanced',
    dual=False,
    max_iter=8000,
    random_state=42
)

param_dist = {
    'C': loguniform(1e-2, 10)
}

random_search = RandomizedSearchCV(
    estimator=base_svm,
    param_distributions=param_dist,
    n_iter=15,
    cv=3,
    scoring='f1',
    n_jobs=-1,
    verbose=1,
    random_state=42
)

print("Tuning Linear SVM ... ")
random_search.fit(X_train_scaled,
y_train)

print("\nBest Parameters:",
random_search.best_params_)

calibrated_svm =
CalibratedClassifierCV(
    random_search.best_estimator_,
    method='sigmoid',
    cv=3
```

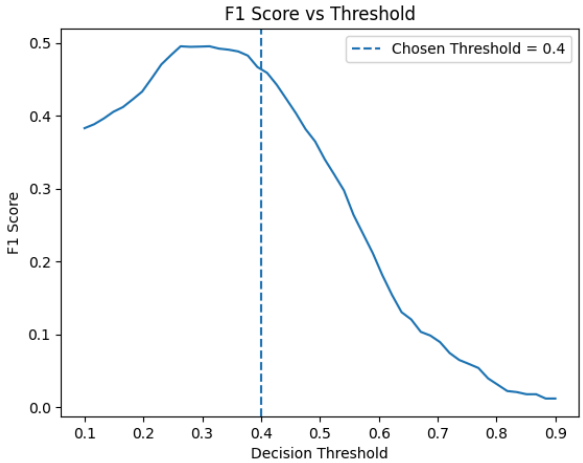
```
)
calibrated_svm.fit(X_train_scaled,
y_train)

y_probs =
calibrated_svm.predict_proba(X_test_
scaled)[:, 1]

threshold = 0.40   # tuned threshold
(default is 0.50)
y_pred = (y_probs ⩾
threshold).astype(int)

# 8. Evaluation
print("\n--- Classification Report
---")
print(classification_report(y_test,
y_pred))

print("\n--- Confusion Matrix ---")
print(confusion_matrix(y_test,
y_pred))

print("\n--- Accuracy ---")
print(f"{accuracy_score(y_test,
y_pred):.4f}")

print("\n--- F1 Score (Default
Class) ---")
print(f"{f1_score(y_test,
y_pred):.4f}")

import matplotlib.pyplot as plt
from sklearn.metrics import
ConfusionMatrixDisplay

cm = confusion_matrix(y_test,
y_pred)

disp =
ConfusionMatrixDisplay(confusion_mat
rix=cm)
disp.plot()
plt.title("Confusion Matrix - Linear
SVM")
plt.show()

from sklearn.metrics import
roc_curve, auc

fpr, tpr, _ = roc_curve(y_test,
y_probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label=f"AUC =
{roc_auc:.3f}")
plt.plot([0, 1], [0, 1],
linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Linear SVM")
plt.legend()
plt.show()

from sklearn.metrics import
precision_recall_curve,
average_precision_score

precision, recall, _ =
precision_recall_curve(y_test,
y_probs)
avg_precision =
average_precision_score(y_test,
y_probs)

plt.figure()
plt.plot(recall, precision,
label=f"AP = {avg_precision:.3f}")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve -
Linear SVM")
plt.legend()
plt.show()

thresholds = np.linspace(0.1, 0.9,
50)
f1_scores = []

for t in thresholds:
    y_temp = (y_probs ⩾
t).astype(int)

f1_scores.append(f1_score(y_test,
y_temp))

plt.figure()
```

```
plt.plot(thresholds, f1_scores)
plt.axvline(x=threshold,
linestyle='--', label=f"Chosen
Threshold = {threshold}")
plt.xlabel("Decision Threshold")
plt.ylabel("F1 Score")
plt.title("F1 Score vs Threshold")
plt.legend()
plt.show()
```









Conclusion:

This experiment demonstrates the effective use of a Linear Support Vector Machine (SVM) for credit card default prediction on a real-world financial dataset. Feature scaling, class balancing, and hyperparameter tuning improved model performance and generalization. Probability calibration and threshold tuning enhanced minority-class detection, making the model suitable as a reliable baseline for credit risk assessment.