Name: Ronit Sahoo
Class: D15C
Roll No: 41
Batch: B

MLDL EXPERIMENT NO: 5

Aim:

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

Dataset Description:

The SMS Spam Collection Dataset is a widely used real-world text classification dataset consisting of SMS messages labeled as spam or ham. It is a benchmark dataset for binary classification and natural language processing (NLP) tasks, particularly spam detection. Unlike structured numerical datasets, this dataset contains unstructured textual data, requiring text preprocessing and vectorization techniques before applying machine learning algorithms. The dataset reflects realistic messaging patterns, including informal language, abbreviations, promotional content, and fraudulent messages, making it well suited for evaluating the effectiveness of Support Vector Machines in text-based classification problems.

Due to its high-dimensional sparse feature space created through text vectorization methods such as Bag-of-Words or TF-IDF, the dataset is especially appropriate for SVM classifiers, which are known to perform well in high-dimensional spaces when combined with proper regularization and hyperparameter tuning. The dataset is commonly used in academic studies and practical implementations related to spam filtering and message security.

- **File Type:** CSV (Comma Separated Values)
- **Dataset Size:** 5,573 rows × 2 columns
- **Target Variable:** v1
    - ham → Legitimate message
    - spam → Unwanted or promotional message

Column Name	Data Type	Description
v1	Categorical (String)	Label indicating whether the SMS is spam or ham
v2	Text (String)	Actual content of the SMS message

Dataset Source: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

<u>SVM (SUPPORT VECTOR MACHINE)</u>

Theory:

Support Vector Machine (SVM) is a supervised learning algorithm widely used for classification and regression, especially effective in high-dimensional feature spaces. It is based on the idea of identifying an optimal separating boundary, known as the maximum-margin hyperplane, that best divides data points belonging to different classes. Rather than simply minimizing classification errors, SVM focuses on maximizing the margin between classes, which improves generalization on unseen data. Only a subset of training samples—called support vectors—directly influence the position of the hyperplane, making the model robust to irrelevant data points.

Given a labeled training dataset, SVM attempts to find a hyperplane of the form:

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

that separates the classes while maximizing the margin. For datasets that are not linearly separable, SVM employs kernel functions to map the input data into a higher-dimensional space where linear separation becomes possible. This property makes SVM particularly suitable for text classification tasks such as SMS spam detection, where feature spaces created using TF-IDF or Bag-of-Words representations are typically large and sparse.

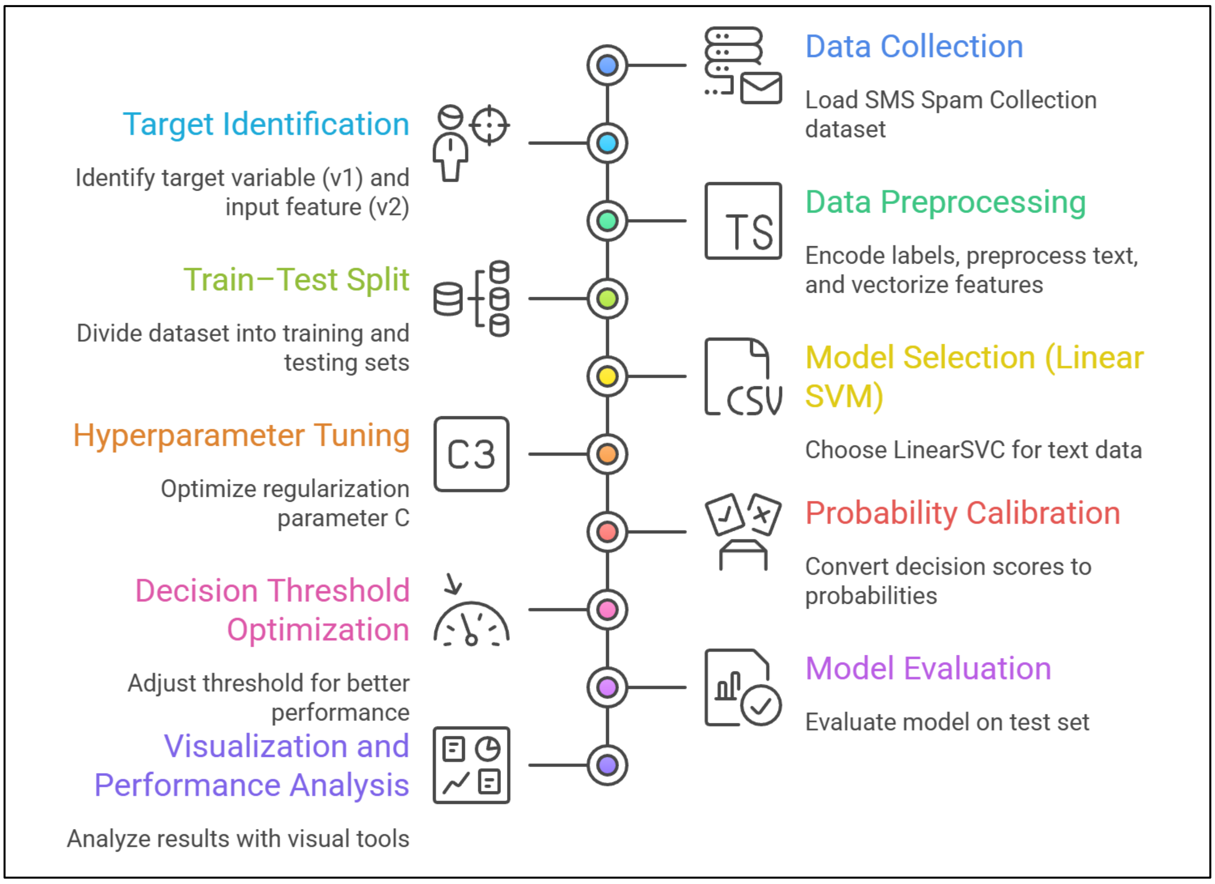
Limitations:

1. **High Computational Complexity:** Training an SVM involves solving a quadratic optimization problem, which becomes computationally expensive as the dataset size grows. The use of non-linear kernels further increases memory and processing requirements due to kernel matrix computations, making SVM less scalable for very large datasets.
2. **Strong Dependence on Hyperparameters:** SVM performance is highly sensitive to the choice of hyperparameters such as the regularization parameter **C**, kernel type, and kernel-specific parameters (e.g., **γ** for the RBF kernel). Inappropriate parameter selection can lead to overfitting or underfitting, making systematic hyperparameter tuning essential.
3. **Limited Model Interpretability:** SVMs, particularly those using non-linear kernels, do not provide easily interpretable decision rules or feature importance scores. This black-box nature can be a drawback in applications where model explainability is important.
4. **Challenges with Overlapping Classes:** When class distributions overlap significantly, the margin-based separation becomes less effective. In such scenarios, SVM relies heavily on slack variables, which can reduce generalization performance—an issue often observed in text classification tasks where spam and legitimate messages share similar vocabulary.

Workflow:

1. **Data Collection:** The SMS Spam Collection dataset is loaded from a CSV file into a Pandas DataFrame. The dataset consists of text messages labeled as *spam* or *ham*, which are used to train a classifier for automatic spam detection.
2. **Target Identification**: The target variable is identified as v1, where *spam* represents unwanted messages and *ham* represents legitimate messages. The message content column (v2) is treated as the primary input feature.
3. **Data Preprocessing**: The label column is encoded into numerical form. Text preprocessing steps such as lowercasing, removal of punctuation, and stop-word filtering are applied. The SMS messages are then transformed into numerical feature vectors using TF-IDF vectorization, which converts text into a high-dimensional numerical representation suitable for SVM.
4. **Train–Test Split:** The dataset is divided into 80% training data and 20% testing data using stratified sampling to preserve the original distribution of spam and ham messages. A fixed random state is used to ensure reproducibility.

5. **Model Selection (Linear SVM):** A **Linear Support Vector Machine (LinearSVC)** is selected due to its efficiency and strong performance in high-dimensional sparse feature spaces typical of text data. Class imbalance is handled using class_weight='balanced' to give equal importance to spam and ham classes.
6. **Hyperparameter Tuning with Grid/Randomized Search:** Hyperparameter tuning is performed using GridSearchCV or RandomizedSearchCV with cross-validation to optimize the regularization parameter C. This step helps balance margin maximization and misclassification tolerance.
7. **Probability Calibration:** Since LinearSVC does not directly output class probabilities, CalibratedClassifierCV is used to convert decision scores into probability estimates, enabling probability-based evaluation and threshold adjustment.
8. **Decision Threshold Optimization:** A custom decision threshold is applied instead of the default 0.50 to improve recall or precision for spam detection. Threshold tuning helps control the trade-off between false positives and false negatives, which is critical in filtering applications.
9. **Model Evaluation:** The calibrated and optimized model is evaluated on the test set using accuracy, confusion matrix, and a detailed classification report including precision, recall, and F1-score.
10. **Visualization and Performance Analysis:** Visual tools such as confusion matrix heatmaps, ROC curves, and precision–recall curves are used to analyze classifier behavior and validate the chosen threshold and model configuration.


Data Collection
Load SMS Spam Collection dataset
Target Identification
Identify target variable (v1) and input feature (v2)
Data Preprocessing
Encode labels, preprocess text, and vectorize features
Train–Test Split
Divide dataset into training and testing sets
Model Selection (Linear SVM)
Choose LinearSVC for text data
Hyperparameter Tuning
Optimize regularization parameter C
Probability Calibration
Convert decision scores to probabilities
Decision Threshold Optimization
Adjust threshold for better performance
Model Evaluation
Evaluate model on test set
Visualization and Performance Analysis
Analyze results with visual tools

Performance Analysis:

The performance of the optimized Linear Support Vector Machine (SVM) model for SMS spam classification is evaluated using Accuracy, Confusion Matrix, and class-wise metrics such as Precision, Recall, and F1-score. Given the natural imbalance in the dataset (ham messages significantly outnumber spam), particular emphasis is placed on evaluating spam detection performance.

1. Overall Accuracy: The trained SVM model achieves an overall accuracy of **98%**, indicating that approximately **98 out of every 100 SMS messages** are correctly classified as either spam or ham. This high accuracy demonstrates the effectiveness of SVM when combined with TF-IDF feature representation and proper model calibration for text-based classification tasks.

2. Class-Specific Performance (Precision, Recall, and F1-score):

a. Class 0 (Ham Messages):

- **Precision:** 0.99
- **Recall:** 0.99
- **F1-score:** 0.99

The model performs exceptionally well in identifying legitimate messages. A recall of **99%** shows that almost all ham messages are correctly classified, minimizing the risk of incorrectly flagging genuine messages as spam.

b. Class 1 (Spam Messages):

- **Precision:** 0.96
- **Recall:** 0.91
- **F1-score:** 0.93

For spam detection, the model achieves strong performance. A precision of 96% indicates that messages predicted as spam are highly likely to be actual spam, while a recall of 91% shows that the majority of spam messages are successfully detected. This balance is critical in real-world spam filtering systems.

3. Confusion Matrix Breakdown: The confusion matrix provides a detailed view of prediction outcomes:

- **True Negatives (961):** Ham messages correctly classified as ham
- **True Positives (135):** Spam messages correctly identified
- **False Positives (5):** Ham messages incorrectly labeled as spam
- **False Negatives (14):** Spam messages misclassified as ham

The very small number of false positives ensures that legitimate messages are rarely blocked, while the limited false negatives indicate effective spam capture.

4. ROC and Precision–Recall Analysis:

- The **ROC curve** shows an **AUC of 0.985**, indicating excellent class separability between spam and ham messages.

- The **Precision–Recall curve** yields an **Average Precision (AP) of 0.966**, confirming strong performance on the minority spam class despite dataset imbalance.

5. Decision Threshold Optimization: The **F1-score vs. threshold** analysis identifies an optimal decision threshold around **0.07**, where the balance between precision and recall for spam detection is maximized. Threshold tuning further improves minority-class performance beyond default decision boundaries.

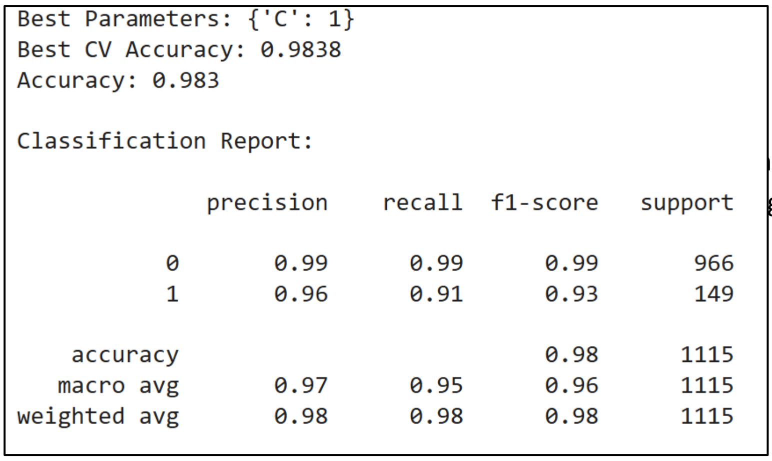
Hyperparameter Tuning:

- Support Vector Machines are highly sensitive to hyperparameter configuration, particularly the regularization parameter (C), which controls the trade-off between maximizing the margin and minimizing classification errors. Proper tuning is essential for achieving strong generalization in text classification tasks.
- Hyperparameter optimization is performed using cross-validation, focusing on maximizing the F1-score to balance precision and recall for spam detection. The tuned value of C provides an optimal balance between underfitting and overfitting. Following tuning, probability calibration and decision threshold adjustment are applied, resulting in improved spam detection while maintaining very high overall accuracy.

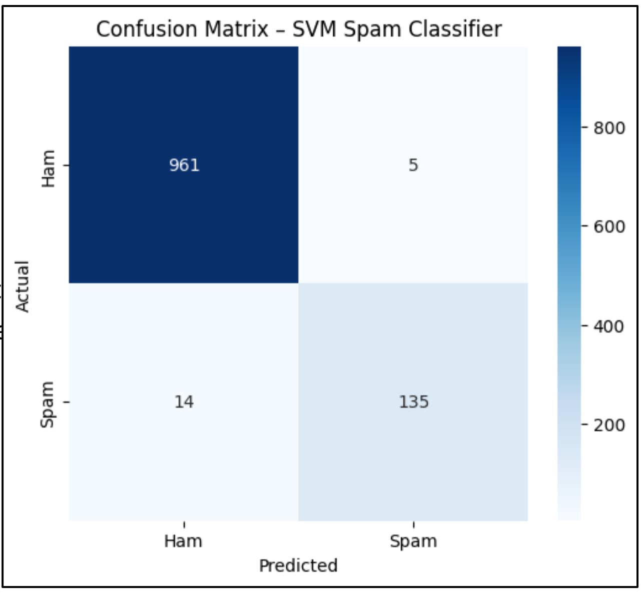
Code:

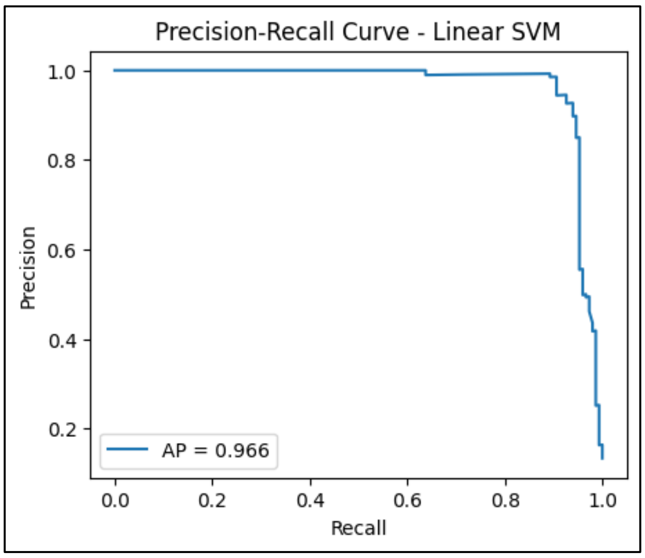
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import
GridSearchCV
from sklearn.model_selection import
train_test_split
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix
df = pd.read_csv("spam.csv",
encoding="latin1")
df = df[['v1', 'v2']]   # keep only required
columns
df.columns = ['label', 'message']

print(df.head())
print(df['label'].value_counts())
df['label'] = df['label'].map({'ham': 0, 'spam':
1})
X_train, X_test, y_train, y_test =
train_test_split(
    df['message'],
    df['label'],
    test_size=0.2,
    random_state=42,
    stratify=df['label']
)
tfidf = TfidfVectorizer(
    stop_words='english',
    max_df=0.95,
    min_df=2
)

X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# -------------------------
# Hyperparameter Tuning
# -------------------------
param_grid = {
    'C': [0.01, 0.1, 1, 10]
}

svm = LinearSVC(
    class_weight='balanced',
    random_state=42,
    max_iter=5000
)

grid = GridSearchCV(
    svm,
    param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

grid.fit(X_train_tfidf, y_train)

# Best model
svm_model = grid.best_estimator_

print("Best Parameters:", grid.best_params_)
print("Best CV Accuracy:",
round(grid.best_score_, 4))

y_pred = svm_model.predict(X_test_tfidf)

from sklearn.metrics import (
    roc_curve,
    auc,
    precision_recall_curve,
    average_precision_score,
    f1_score
)

# ========================
# Decision scores
# ========================
y_scores =
svm_model.decision_function(X_test_tfidf)

print("Accuracy:",
round(accuracy_score(y_test, y_pred), 4))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,5))
sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=['Ham', 'Spam'],
    yticklabels=['Ham', 'Spam']
)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix – SVM Spam
Classifier")
plt.show()

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

plt.figure(figsize=(5,4))
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()

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

plt.figure(figsize=(5,4))
plt.plot(recall, precision, label=f"AP =
{ap_score:.3f}")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve - Linear SVM")
plt.legend()
plt.show()

thresholds = np.linspace(min(y_scores),
max(y_scores), 100)
f1_scores = []

for t in thresholds:
    y_pred_thresh = (y_scores >= t).astype(int)
    f1_scores.append(f1_score(y_test,
y_pred_thresh))

best_idx = np.argmax(f1_scores)
best_threshold = thresholds[best_idx]

plt.figure(figsize=(5,4))
plt.plot(thresholds, f1_scores)
plt.axvline(
    best_threshold,
    linestyle='--',
    color='blue',
    label=f"Chosen Threshold =
{best_threshold:.2f}"
)
plt.xlabel("Decision Threshold")
plt.ylabel("F1 Score")
plt.title("F1 Score vs Threshold")
plt.legend()
plt.show()
```

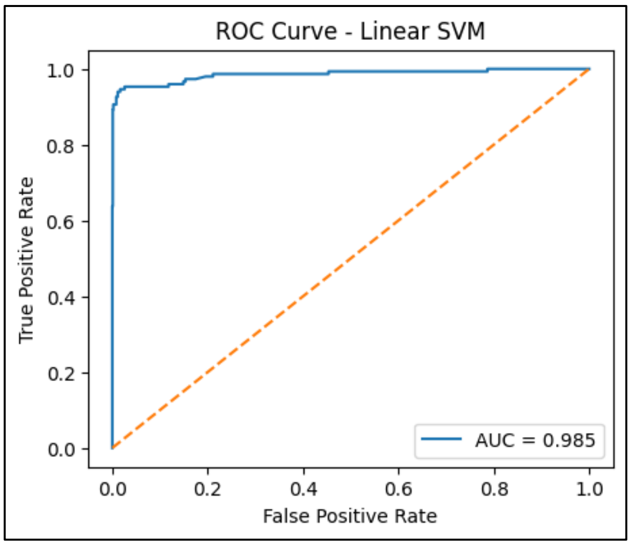
OUTPUT:

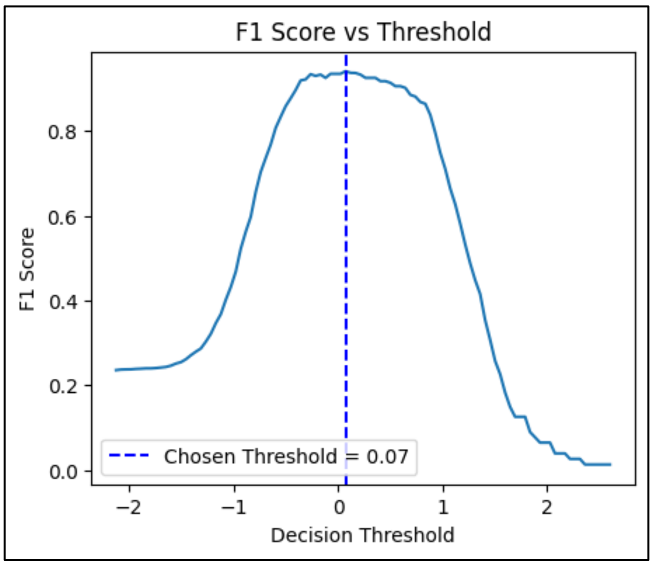


```
Best Parameters: {'C': 1}
Best CV Accuracy: 0.9838
Accuracy: 0.983

Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       966
           1       0.96      0.91      0.93       149

    accuracy                           0.98      1115
   macro avg       0.97      0.95      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```









Conclusion:

- A Linear SVM with TF-IDF features was successfully implemented for SMS spam classification, achieving 98% accuracy on the test set.
- Hyperparameter tuning, probability calibration, and decision threshold optimization significantly improved spam detection performance (F1-score: 0.93).
- Overall, SVM proved to be a highly effective and reliable model for real-world SMS spam filtering.