Support Vector Machine (SVM)

Introduction

- SVM is a supervised ML algorithm used for both linear and nonlinear classification.
- SVMs are particularly effective because they focus on finding the maximum separating hyperplane between the different classes in the target feature, making them robust for both binary and multiclass classification.

Classifying Email as Spam or Not Spam

1. Import all required libraries.

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score,precision_score, recall_score, f1_score
```

2. Create a data frame for the given data and analyse its contents.

```
data=pd.read_csv("emails.csv")
data.head()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	0

5 rows × 3002 columns

3. Select the features and targets.
    - From the data structure, it is clear that our target is to predict whether an email is spam or not.
    - The first column is email number which has no role in prediction.
    - So, we can select all the columns except 1st and last columns as the features.

```
x=data.drop(["Prediction","Email No."],axis=1)
y=data['Prediction']
```

4. Split the data in to training (70%) and testing sets (30%).

```
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

5. Scale the features.
    - Large feature values can distort the distance calculation, leading to biased predictions.
    - Scaling ensures that no single feature dominates the distance computation.

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

6. Train the data using linear kernel.
    - A linear kernel is best for datasets that can be separated by a straight line (in 2D) or a hyperplane (in higher dimensions).
    - It is effective for high-dimensional data where the number of features is large compared to the number of samples.

```
model = SVC(kernel='linear')
model.fit(x_train_scaled,y_train)
```

```
SVC
SVC(kernel='linear')
```

7. Make predictions and evaluate the model.

```
y_pred = model.predict(x_test_scaled)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
precision=precision_score(y_test, y_pred)
print("Precision:",precision)
rs=recall_score(y_test, y_pred)
print("Recall score:",rs)
f1=f1_score(y_test, y_pred)
print("F1 score:",f1)
```

```
Accuracy: 0.9400773195876289
Precision: 0.8851063829787233
Recall score: 0.9142857142857143
F1 score: 0.8994594594594595
```

```
print("Classification report:")
print(classification_report(y_test, y_pred))
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.96      1097
           1       0.89      0.91      0.90       455

    accuracy                           0.94      1552
   macro avg       0.92      0.93      0.93      1552
weighted avg       0.94      0.94      0.94      1552
```