# Lab Report 06

## SVM (Support Vector Machine)

Name MaazAhmad
Reg No B23F722AI170
Section Ai yellow
Submitted to Abdullah Sajid

---

## Prerequisites

First, import the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (confusion_matrix, classification_report,
                             roc_auc_score, roc_curve, accuracy_score,
                             precision_score, recall_score)
from sklearn.pipeline import Pipeline

# Set plot style
sns.set(style="whitegrid")
```

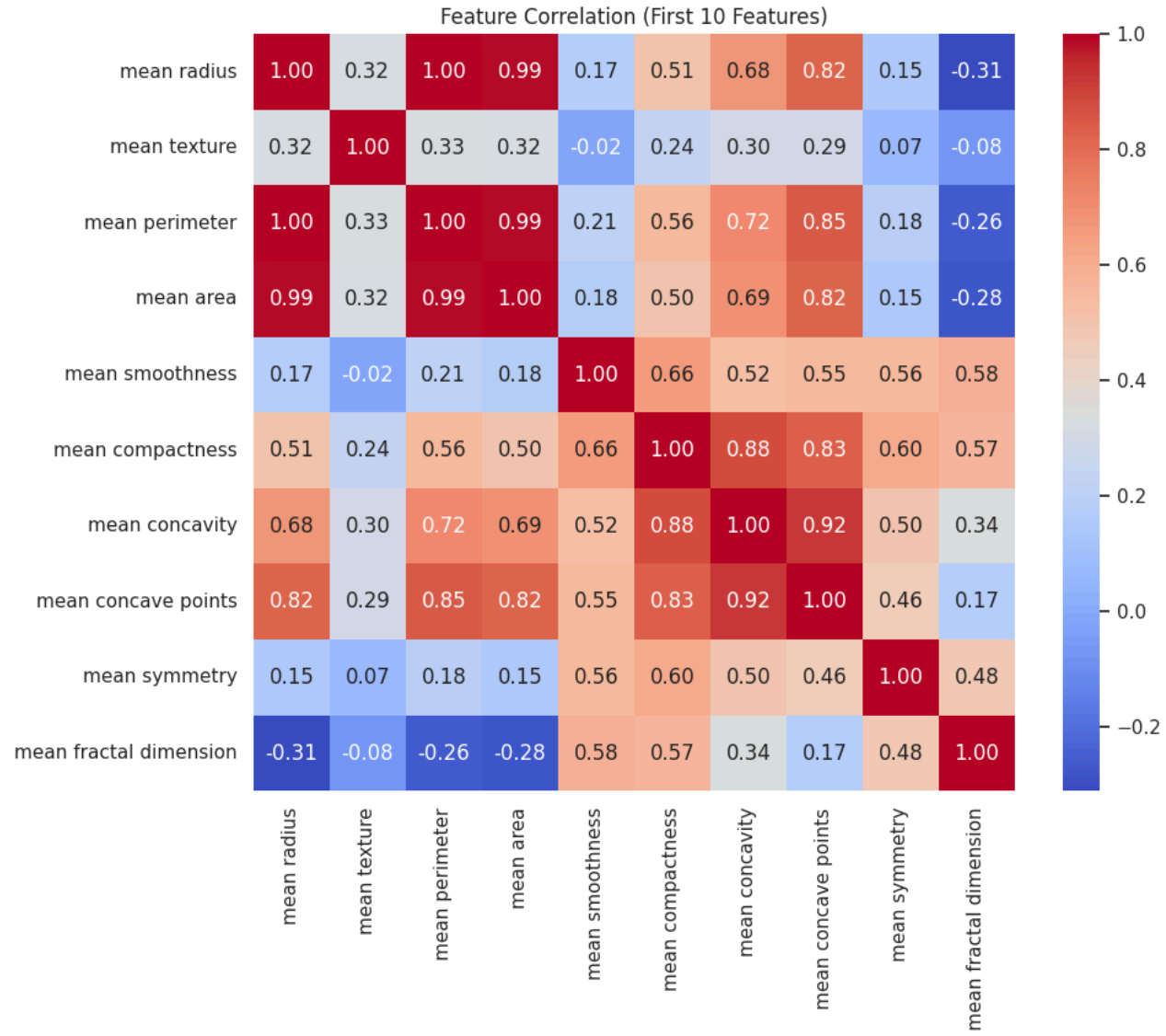
## Task no 1

Load data, check distributions, split into Train (60%), Validation (20%), and Test (20%), and scale features .

```
# 1. Load Dataset [cite: 151]
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

# 2. Basic EDA [cite: 152]
print("Missing values:\n", X.isnull().sum().sum())  # Check for missing values
print("\nTarget distribution:\n", pd.Series(y).value_counts()) # 0 = Malignant, 1 = Benign

# Check correlations (subset visualization)
plt.figure(figsize=(10, 8))
sns.heatmap(X.iloc[:, :10].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Feature Correlation (First 10 Features)")
plt.show()

# 3. Split into Train/Validation/Test (60/20/20) [cite: 154]
# First split: 80% remaining (Train+Val) and 20% Test
X_temp, X_test, y_temp, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Second split: Split the 80% chunk. 0.25 of 0.8 is 0.2 (which is 20% of total)
X_train, X_val, y_train, y_val = train_test_split(
    X_temp, y_temp, test_size=0.25, random_state=42, stratify=y_temp
)

print(f"Train shape: {X_train.shape} (60%)")
print(f"Val shape:   {X_val.shape} (20%)")
print(f"Test shape:  {X_test.shape} (20%)")
```

```
# 4. Scale features [cite: 155]
# Fit scaler ONLY on training data to avoid data leakage
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

```
Missing values:
 0

Target distribution:
 1    357
0    212
Name: count, dtype: int64
```




```
Train shape: (341, 30) (60%)
Val shape:   (114, 30) (20%)
Test shape:  (114, 30) (20%)
```

## Description

**Exploratory Data Analysis & Preprocessing:**
Is code main hum ne Breast Cancer dataset load kar ke usay Train, Validation aur Test parts main split kiya hai. Phir StandardScaler use kar ke features ko scale kiya taake sab values ki range same ho jaye aur model confuse na ho. Yeh step zaroori hai kyun ke SVM distances par kaam karta hai aur scaling se performance improve hoti hai.

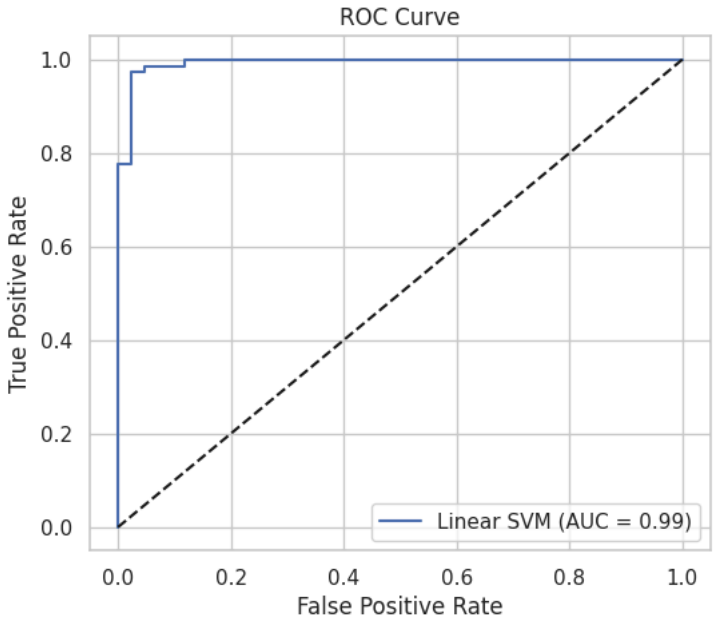
---

## Task no 2

Train & Evaluate Basic SVM

```
# 1. Train Linear SVM [cite: 157]
# probability=True is needed for ROC AUC later
svm_linear = SVC(kernel='linear', probability=True, random_state=42)
svm_linear.fit(X_train_scaled, y_train)

# 2. Evaluate on Test Set [cite: 158]
y_pred = svm_linear.predict(X_test_scaled)
y_prob = svm_linear.predict_proba(X_test_scaled)[:, 1]

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print(f"ROC AUC Score: {roc_auc_score(y_test, y_prob):.4f}")

# 3. Plot ROC Curve [cite: 159]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'Linear SVM (AUC = {roc_auc_score(y_test, y_prob):.2f})')
plt.plot([0, 1], [0, 1], 'k--') # Diagonal random guess line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

```
Confusion Matrix:
 [[41  1]
 [ 3 69]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        42
           1       0.99      0.96      0.97        72

    accuracy                           0.96       114
   macro avg       0.96      0.97      0.96       114
weighted avg       0.97      0.96      0.97       114

ROC AUC Score: 0.9931
```




## Description

**Train & Evaluate Basic SVM:**
Hum ne kernel='linear' ke saath SVM model train kiya aur uski performance ko confusion matrix aur report ke zariye check kiya. Test data

par predictions lein aur ROC AUC score calculate kiya taake model ki capability ka andaza ho sakay. Akhir main ROC curve plot kiya jo False Positive aur True Positive rate ka relation dikhata hai.

---

## Task no 3

Hyperparameter Tuning

```
# 1. Define Parameter Grid [cite: 161]
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.1, 0.01],
    'kernel': ['rbf', 'linear', 'poly']
}

# 2. Run Grid Search [cite: 162]
# We use X_train_scaled (standard 5-fold CV)
grid = GridSearchCV(SVC(probability=True, random_state=42), param_grid, cv=5, verbose=1, n_jobs=-1)
grid.fit(X_train_scaled, y_train)

# 3. Report Best Parameters [cite: 163]
print(f"Best Parameters: {grid.best_params_}")
print(f"Best Cross-Val Score: {grid.best_score_:.4f}")

# Validate best model on Test Set
best_model = grid.best_estimator_
y_pred_tuned = best_model.predict(X_test_scaled)
print("\nTuned Model Report:\n", classification_report(y_test, y_pred_tuned))
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
Best Parameters: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Best Cross-Val Score: 0.9794

Tuned Model Report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98        42
           1       0.99      0.99      0.99        72

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

## Description

**Hyperparameter Tuning:**
Is hissay main hum ne GridSearchCV use kiya taake C, gamma aur kernel ki best values dhoond sakein jo model ko optimize karein. Yeh code automatically different combinations try karta hai aur best parameters return karta hai. Best parameters milnay ke baad hum ne tuned model ko test set par dubara evaluate kiya.

---

## Task no 4

Misclassification & Margin Analysis

```
# 1. Inspect Support Vectors [cite: 165]
# Using the linear model from Task 2 for easier interpretation of w
n_support = svm_linear.support_vectors_.shape[0]
print(f"Number of Support Vectors: {n_support}")

# 2. Analyze Misclassified Samples [cite: 166]
# Find indices where prediction does not match actual
misclassified_idx = np.where(y_test != y_pred)[0]

print(f"\nNumber of misclassified test samples: {len(misclassified_idx)}")
```

```
if len(misclassified_idx) > 0:
    # decision_function calculates the signed distance to the hyperplane
    distances = svm_linear.decision_function(X_test_scaled)

    print("\nDistance to boundary for misclassified points:")
    for idx in misclassified_idx:
        dist = distances[idx]
        true_label = y_test[idx]
        pred_label = y_pred[idx]
        print(f"Sample {idx}: True={true_label}, Pred={pred_label}, Distance={dist:.4f}")

# 3. Analyze C Tradeoff [cite: 167]
# Quick comparison of C=0.1 vs C=100
for c_val in [0.1, 100]:
    svm_temp = SVC(kernel='linear', C=c_val).fit(X_train_scaled, y_train)
    n_vecs = svm_temp.support_vectors_.shape[0]
    train_acc = svm_temp.score(X_train_scaled, y_train)
    print(f"C={c_val}: Support Vectors={n_vecs}, Train Accuracy={train_acc:.4f}")
```

```
Number of Support Vectors: 26

Number of misclassified test samples: 4

Distance to boundary for misclassified points:
Sample 16: True=1, Pred=0, Distance=-1.0476
Sample 51: True=1, Pred=0, Distance=-0.2518
Sample 53: True=0, Pred=1, Distance=1.5565
Sample 66: True=1, Pred=0, Distance=-0.0206
C=0.1: Support Vectors=40, Train Accuracy=0.9853
C=100: Support Vectors=23, Train Accuracy=1.0000
```

## Description

**Misclassification & Margin Analysis:**
Yahan hum ne support_vectors_ ko inspect kiya aur ghalat classify honay walay points ka decision boundary se distance check kiya. Is se hum dekh saktay hain ke kaunse points margin ke andar hain ya boundary violate kar rahay hain. Yeh analysis hamein Hard vs Soft margin ka tradeoff samajhnay main madad deta hai.

---

## Task no 5

Compare SVM vs Logistic Regression

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

bag = BaggingClassifier(
    estimator=DecisionTreeClassifier(random_state=42),
    n_estimators=100,
    bootstrap=True,
    oob_score=True,
    random_state=42,
    n_jobs=-1
)

bag.fit(X_train, y_train)

print("Bagging Test Accuracy:", accuracy_score(y_test, bag.predict(X_test)))
print("Bagging OOB Score:", bag.oob_score_)
```

Show hidden output

## Description

**Compare SVM vs Logistic Regression:** Hum ne Logistic Regression model train kiya aur uskay results (Accuracy, ROC AUC) ko SVM ke saath compare kiya. Dono models ki performance ko side-by-side table main show kiya taake fark clear ho jaye. Is se pata chalta hai ke

kya waqai SVM is dataset ke liye Logistic Regression se behtar hai ya nahi.

---

## Lab Summary

1. The lab introduces **Support Vector Machine (SVM)** as a supervised learning algorithm that finds a hyperplane to separate classes by maximizing the margin.
2. **The Margin & Support Vectors:** The margin is defined as the distance between the decision boundary and the closest data points, called "support vectors," which are the only points that determine the boundary position.
3. **Hard vs. Soft Margin:** It explains the difference between Hard Margin (requires perfect separation) and Soft Margin (allows some misclassification using slack variables), where the parameter 'C' controls the tolerance for errors.
4. **The Kernel Trick:** The concept of Kernels (Linear, Polynomial, RBF, Sigmoid) is introduced to handle non-linear data by calculating point similarity without explicitly transforming data to higher dimensions.
5. **Dataset:** The practical work focuses on the Breast Cancer Wisconsin dataset, which contains 569 examples and 30 numeric features to classify tumors as malignant or benign.
6. **Implementation Steps:** The standard workflow involves loading data, splitting it into training and testing sets, and importantly, scaling features (Normalization) before training the model.
7. **Hyperparameter Tuning:** One of the key lab tasks is to use GridSearchCV to optimize model parameters like 'C', 'gamma', and the choice of 'kernel' to improve performance.
8. **Model Comparison:** The final task involves comparing the SVM model against Logistic Regression by analyzing metrics such as ROC AUC, confusion matrices, and the trade-off between margin width and misclassification.

---