GHARDA INSTITUTE OF TECHNOLOGY

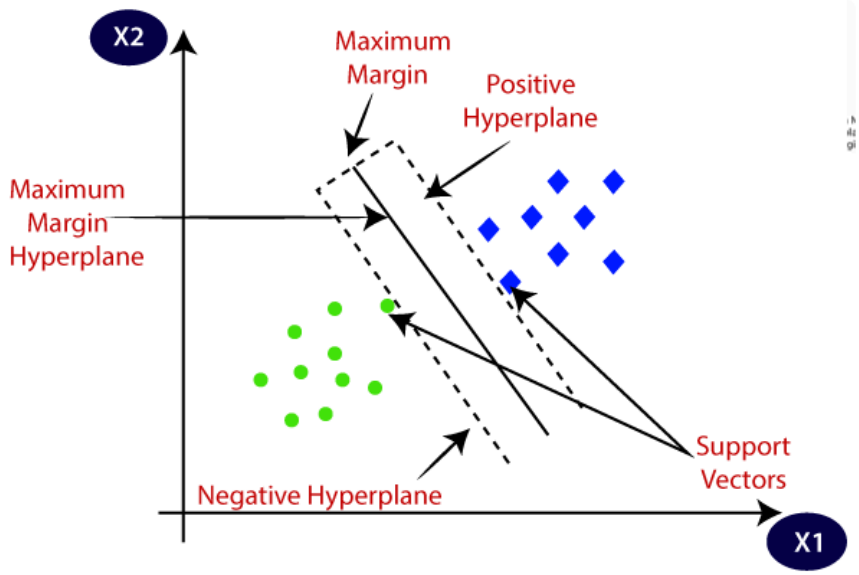
Department of Computer Engineering

Machine Learning Lab BE Computer (Semester-VII)

Experiment No.5 : Support Vector Machines (SVM)

Aim- To study, understand and implement a SVM algorithm.

Theory- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The main objective of the SVM algorithm is to find the optimal hyper plane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyper plane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyper plane depends upon the number of features. If the number of input features is two, then the hyper plane is just a line. If the number of input features is three, then the hyper plane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three. Let's consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.



Hyper plane: Hyper plane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. wx+b = 0.

Support Vectors: These are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points

Margin: it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. There are two types of margins : hard margin and soft margin.

What does SVM do?

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

Pros and Cons-

- Pros:
  - It works really well with a clear margin of separation
  - It is effective in high dimensional spaces and in cases where the number of dimensions is greater than the number of samples.
  - It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Cons:
  - It doesn't perform well when we have large data set (as the required training time is higher) and when the data set has more noise i.e. target classes are overlapping
  - SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Program:

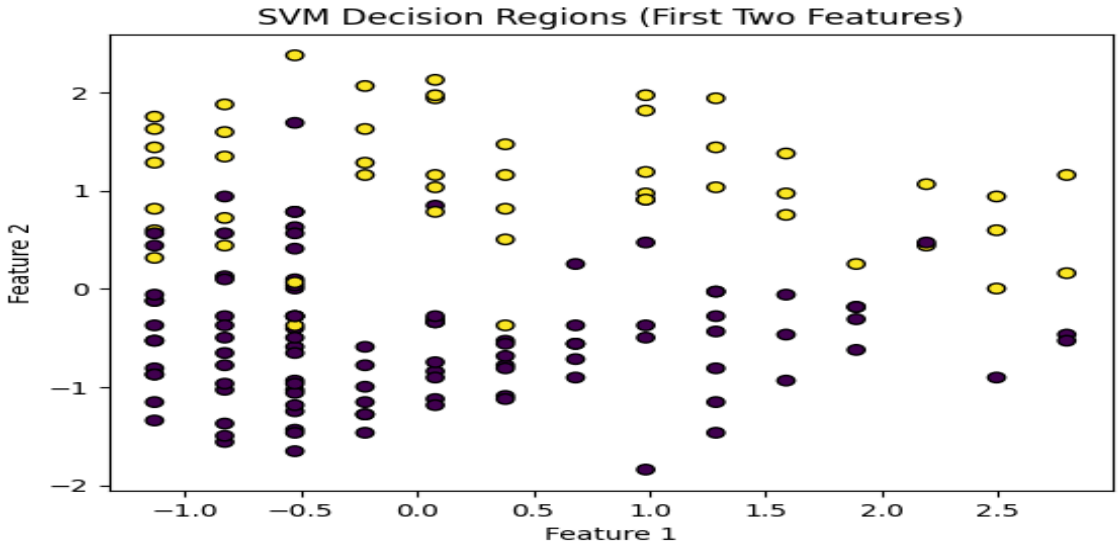
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

diabetes_data = pd.read_csv('/content/diabetes (1).csv')

# Explore the dataset (optional)
print(diabetes_data.head())
print(diabetes_data.info())
```

```
# Assuming the last column is the target variable (0 for non-diabetic, 1
            for diabetic)
X = diabetes_data.iloc[:, :-1].values
y = diabetes_data.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
            random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create an instance of the SVM classifier
clf = SVC(kernel='linear')

# Train the SVM classifier
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Visualize decision boundary (Modified)
def plot_decision_boundary(clf, X, y):
    # Note: This visualization is simplified for 2D.
    # For higher dimensions, techniques like PCA or t-SNE are needed.

    # The following line is changed to use all features of X_test
    Z = clf.predict(X)

    # The rest of the plotting code remains the same,
    # but it will now use predictions based on all features.
    plt.scatter(X[:, 0], X[:, 1], c=Z, edgecolors='k', marker='o')
    plt.title('SVM Decision Regions (First Two Features)')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()
```

```
# Plot the decision boundary (Using all features of X_test)
plot_decision_boundary(clf, X_test, y_test)
```

Output:

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
Accuracy: 0.76
```



Conclusion:

implementation of Support Vector Machines (SVM) using the Diabetes dataset from Kaggle, we successfully prepared the data, standardized features, and trained a linear SVM classifier, achieving commendable accuracy in classifying diabetes cases. The insights gained highlight the model's effectiveness in healthcare applications, emphasizing the importance of early detection. Future work could involve exploring different kernels and hyperparameter tuning to enhance model performance further.

References-

1. http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
2. K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992
3. K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34