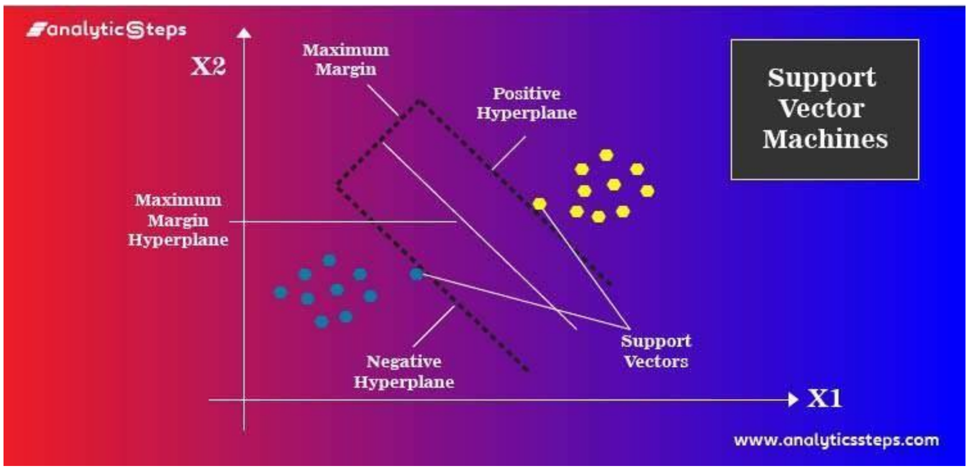
EXPERIMENT NO. 6

Aim: Implement Support Vector Machines.

Theory:

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



Support Vector Machine Terminology:

★ *Hyperplane*: Hyperplane 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:* Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.

- ★ *Margin*: Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin. The wider margin indicates better classification performance.

- ★ *Kernel*: Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space. Some of the common kernel functions are linear, polynomial, radial basis function(RBF), and sigmoid.

- ★ *Hard Margin*: The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.

- ★ *Soft Margin*: When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique. Each data point has a slack variable introduced by the soft-margin SVM formulation, which softens the strict margin requirement and permits certain misclassifications or violations. It discovers a compromise between increasing the margin and reducing violations.

- ★ *C*: Margin maximisation and misclassification fines are balanced by the regularisation parameter C in SVM. The penalty for going over the margin or misclassifying data items is decided by it. A stricter penalty is imposed with a greater value of C, which results in a smaller margin and perhaps fewer misclassifications.

- ★ *Hinge Loss*: A typical loss function in SVMs is hinge loss. It punishes incorrect classifications or margin violations. The objective function in SVM is frequently formed by combining it with the regularisation term.

- ★ *Dual Problem*: A dual Problem of the optimisation problem that requires locating the Lagrange multipliers related to the support vectors can be used to solve SVM. The dual formulation enables the use of kernel tricks and more effective computing.

Types of Support Vector Machine:

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

- ➢ *Linear SVM*: Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.

- ➢ *Non-Linear SVM*: Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

Applications

- *Classification*: SVM is widely used for binary and multi-class classification tasks, such as image classification, text categorization, and spam detection.
- *Regression*: SVM can be applied to regression problems, known as Support Vector Regression (SVR), where it predicts continuous output variables.

- *Anomaly Detection*: SVM is effective in identifying outliers or anomalies in datasets, making it useful for fraud detection, network security, and medical diagnosis.

- *Dimensionality Reduction*: SVM can be used for feature selection or dimensionality reduction by finding a subset of the most informative features that maximize the margin between classes.

Advantages

SVM offers several advantages:

- *Effective in High-Dimensional Spaces*: SVM performs well even in high-dimensional spaces, making it suitable for datasets with many features.

- *Robust to Overfitting*: By maximizing the margin, SVM tends to have good generalization performance and is less prone to overfitting.

- *Versatile*: SVM can handle both linear and non-linear data through the use of different kernel functions.

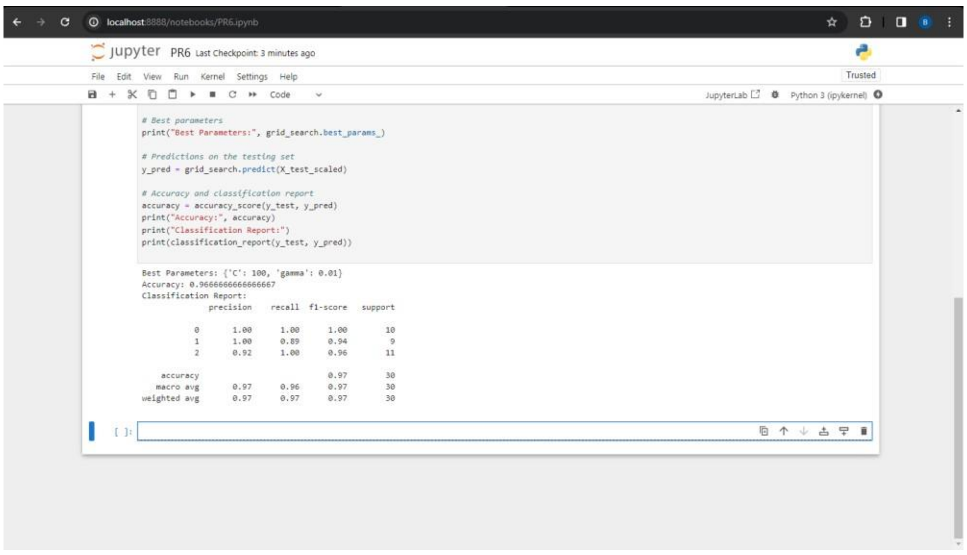
- *Global Optimality*: SVM finds the globally optimal solution for the decision boundary, unlike some other algorithms that may converge to local optima.

Program Code:

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import GridSearchCV

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report


# Load the iris dataset (example dataset from scikit-learn)

iris = load_iris()

X = iris.data

y = iris.target


# Split the data 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)


# Feature scaling

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)


# Parameter tuning

param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [0.01, 0.1, 1, 10]}

svm_classifier = SVC(kernel='rbf')

grid_search = GridSearchCV(svm_classifier, param_grid, cv=5)

grid_search.fit(X_train_scaled, y_train)
```

```
# Best parameters

print("Best Parameters:", grid_search.best_params_)

y_pred = grid_search.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

print("Classification Report:")

print(classification_report(y_test, y_pred))
```

<u>Output</u>:



localhost:8888/notebooks/PR6.ipynb
jupyter PR6 Last Checkpoint: 3 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[ ]:

```
# Best parameters
print("Best Parameters:", grid_search.best_params_)

# Predictions on the testing set
y_pred = grid_search.predict(X_test_scaled)

# Accuracy and classification report
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Best Parameters: {'C': 100, 'gamma': 0.01}
Accuracy: 0.9666666666666667
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.89      0.94         9
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```

<u>Conclusion</u>:

By performing this practical we had learnt to implement Support Vector Machines in python.