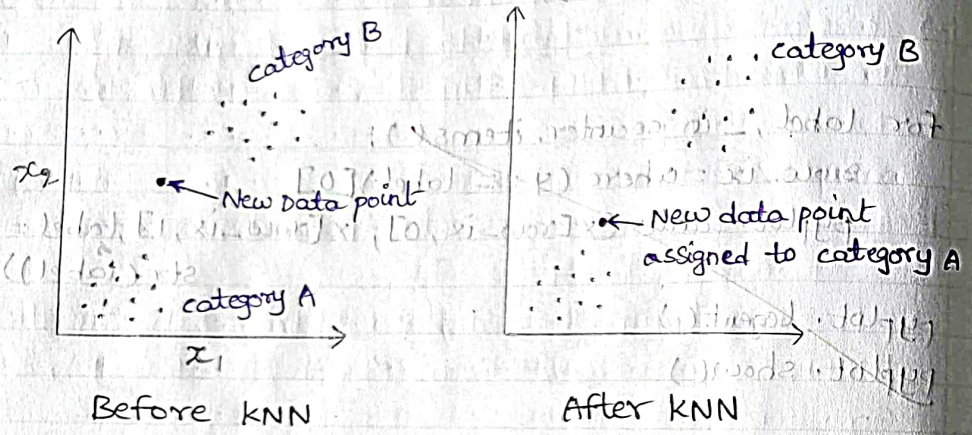
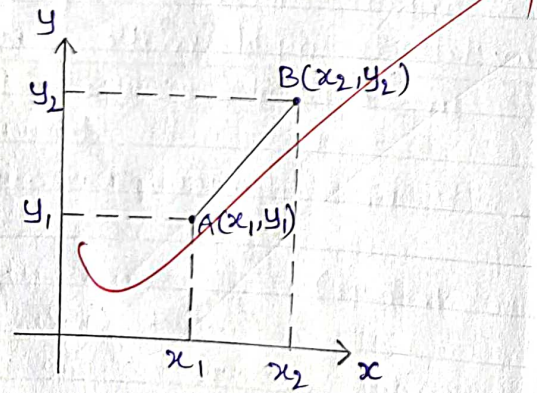
Graphical Representation:



Euclidean Distance:



BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY

Experiment - 4

Aim: Write a program to implement K-Nearest Neighbor algorithm to classify the iris data set: print both correct and wrong predictions.

Description:

K-Nearest Neighbor (KNN): The K-Nearest Neighbor (KNN) algorithm is a simple, non-parametric and lazy learning used for classification and regression tasks in machine learning. In KNN, a data point is classified by a majority vote of its K Nearest Neighbors in the feature space.

Steps:

1) selecting k: choose the number of neighbors, k.
2) Calculating Distance: Compute the distance between the query point and all other points in the dataset: (commonly using Euclidean distance)
3) voting: Identify the k-Nearest points and let them 'vote' for the label or average their values (regression)

Euclidean distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where, d is the euclidean distance
(x_1, y_1) is the coordinate of the first point.
(x_2, y_2) is the coordinate of the second point.

	Id	sepal Lengthcm	sepal widthcm	petal Lengthcm	petal widthcm	species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
...	...	...	...	...	...	...
149	150	5.9	3.0	5.1	1.8	virginica

150 rows × 6 columns

	Id	sepal Lengthcm	sepal widthcm	petal Lengthcm	petal widthcm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
...	...	...	...	...	...
149	150	5.9	3.0	5.1	1.8

150 rows × 5 columns

```
0     Setosa
1     Setosa
...    ...
149   virginica
Name: species, Length: 150, dtype: object
```

	Id	sepal Lengthcm	sepal widthcm	petal Lengthcm	petal widthcm
61	62	5.9	3.0	4.2	1.5
92	93	5.8	2.6	4.0	1.2
...	...	...	...	...	...
47	48	4.6	3.2	1.4	0.2

112 rows × 5 columns

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
data_set = pd.read_csv('iris.csv')
data_set

X = data_set.iloc[:, :-1]
y = data_set.iloc[:, -1]
X

y

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25, random_state = 0)
x_train
```

```
61     versicolor
92     versicolor
...        ...
47     setosa
Name: Species, Length: 112, dtype: object
```

	Id	sepal Length cm	sepal width cm	petal Length cm	petal width cm
114	115	5.8	2.8	5.1	2.4
62	63	6.0	2.2	4.0	1.0
...	...	...	...	...	...
83	84	6.0	2.7	5.1	1.6

```
114    virginica
62     versicolor
...        ...
83     versicolor
Name: species, dtype: object
```

▽ KNeighborsClassifier
KNeighborsClassifier()

```
array(['virginica', 'versicolor', 'setosa', ....,
       'versicolor'], dtype = object)
```

original label	predicted label	correct/wrong
virginica	virginica	correct
versicolor	versicolor	correct
...	...	...
...	...	...
versicolor	versicolor	correct

y_train

X_test

y_test

```
classifier = KNeighborsClassifier(n_neighbors = 5).fit(X_train,
                                                       y_train)
classifier

ypred = classifier.predict(X_test)
ypred

from sklearn import metrics
i = 0
print("\n-------------------------------")
print('%-25s %-25s %-25s' % ('original label', 'predicted
  label', 'correct/wrong'))
print("-------------------------------")
```

Confusion Matrix:

```
[[13  0  0]
 [ 0 16  0]
 [ 0  0  9]]
```

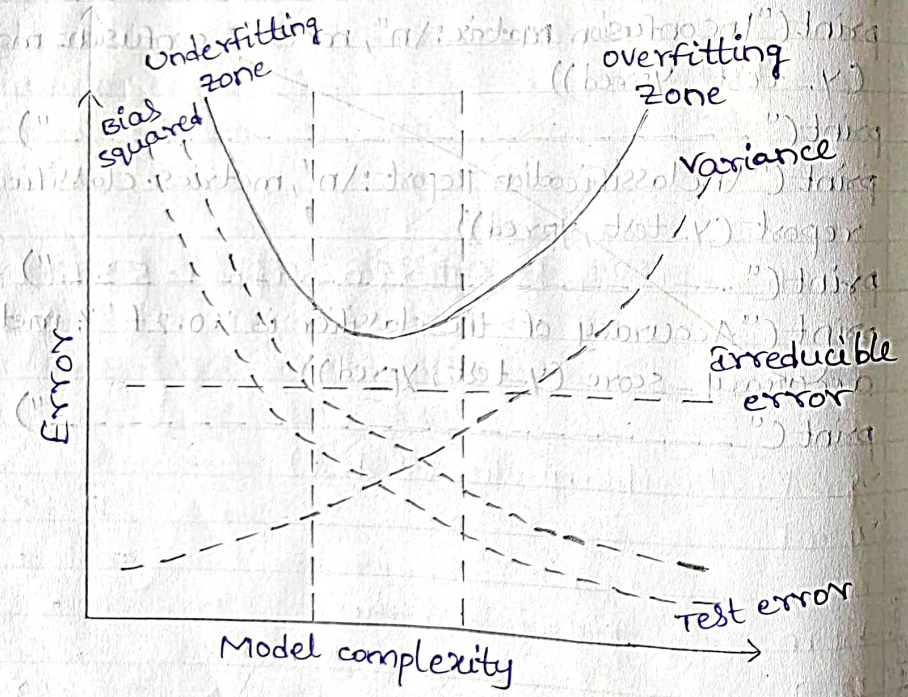
Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	1.00	1.00	16
virginica	1.00	1.00	1.00	9
accuracy			1.00	38
macro avg	1.00	1.00	1.00	38
weighted avg	1.00	1.00	1.00	38

Accuracy of the classifier is 1.00

21/9/24


```
for label in y_test:
    print('%-25s %-25s'%(label,ypred[i]),end=" ")
    if(label==ypred[i]):
        print('%-25s'%('correct'))
    else:
        print('%-25s'%('wrong'))
    i=i+1
print("-------------------------------------------")
print("\nconfusion matrix:\n", metrics.confusion_matrix
(y_test,ypred))
print("-------------------------------------------")
print("\nclassification Report:\n", metrics.classification_
report(y_test,ypred))
print("-------------------------------------------")
print("Accuracy of the classifier is %0.2f'% metrics.
accuracy_score(y_test,ypred))
print("-------------------------------------------")
```



BVC
GROUP OF INSTITUTIONS
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

Experiment -5

Aim: Develop a program for Bias, Variance, Remove duplicates, Cross Validation.

Description:

Bias: While making predictions, a difference occurs between prediction values made by the model and actual values / expected values and this difference is known as bias error or Errors due to bias.

There are 2 types:

1. Low Bias
2. High Bias

1. **Low Bias:** A low bias model will make fewer assumptions about the model becomes able to capture the important features of our dataset.
2. **High Bias:** A model with a high bias makes more assumptions, and the model becomes unable to capture the important features of our dataset.

Variance: Variance tells that how much a random variable is different from the expected value. These are 2 types.

1. Low variance
2. High variance.

1. Low variance: A small variation in the prediction of the target function with changes in the training dataset.
2. High variance: A large variation in the prediction of the target function with changes in the training dataset.

cross validation: cross validation is a technique that used in machine learning to access the performance and generalization of a ability of a model. It involves partitioning the available data into multiple subsets called Folds. And iteratively training and evaluating the model on different combination of these folding techniques.

program:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

np.random.speed(42)
x = np.linspace(0, 10, 100).reshape(-1, 1)
y = 3 * x + np.random.randn(100).reshape(-1, 1)
```

```
x = np.vstack((x, x[:10]))
y = np.vstack((y, y[:10]))
print(x)
print(y)

x_train, x_test, y_train, y_test = train_test_split(x,y,
 test_size=0, random_state=42)

model = LinearRegression()
model.fit(x_train, y_train)

y_train_pred = model.predict(x_train)
train_error = mean_squared_error(y_train, y_train_pred)
y_test_pred = model.predict(x_test)
test_error = mean_squared_error(y_test, y_test_pred)
print("Training error (bias):", train_error)
print("Testing error (variance):", test_error)
Remove duplicate samples
x_unique, indices = np.unique(x, axis=0, return_index=
 True)
y_unique = y[indices]

perform cross-validation:

cross_val_errors = []
for i in range(5):
```

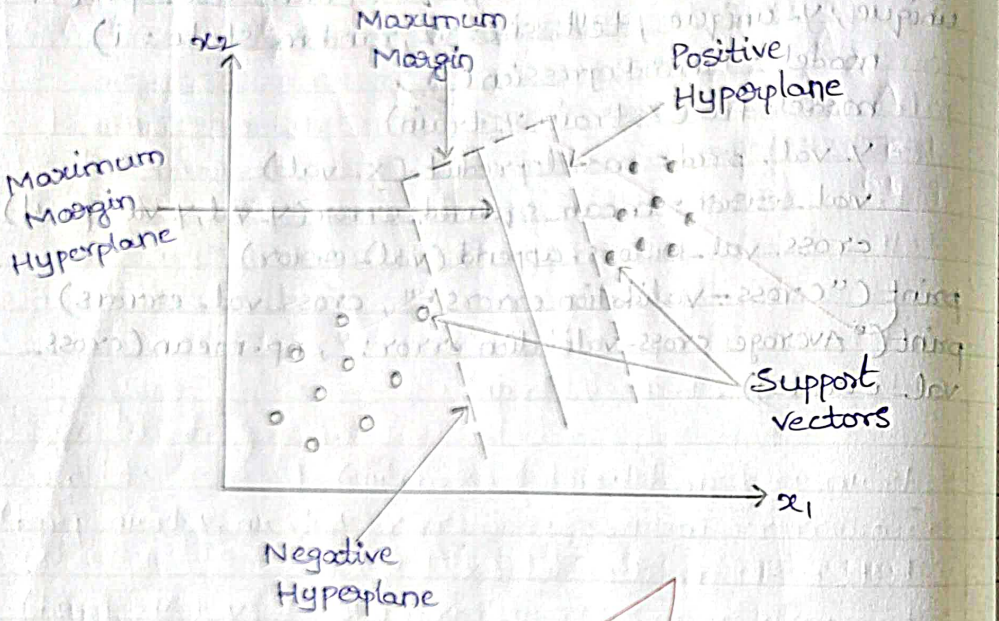
```
[[0.        ]
 [0.1010101]
 ...
 [3.26983277]]
```

```
LinearRegression()
LinearRegression()
```

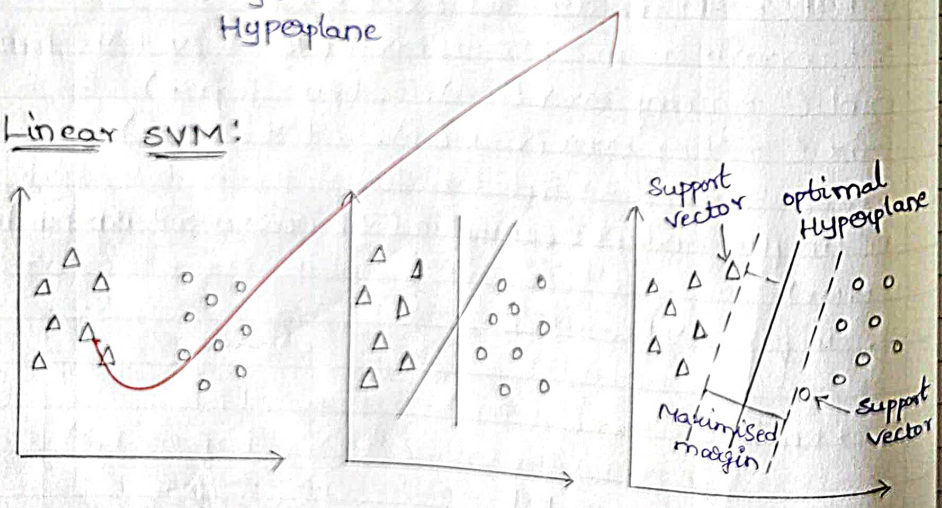
```
Training error (bias): 0.9087396281299075
Testing error (variance): 0.41897811060883222
```

Cross-validation errors: [0.8943963399542353, 0.7118382675593 08, 0.9664698336127481, 0.9156270854451775, 1.1754593805313762]
Average cross-validation error: 0.9327581842056 9


```
X_train, X_val, y_train, y_val = train_test_split(X_
unique, y_unique, test_size=0.2, random_state=i)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_val_pred = model.predict(X_val)
    val_error = mean_squared_error(y_val, y_val_pred)
    cross_val_errors.append(val_error)
print("Cross-validation errors:", cross_val_errors)
print("Average cross-validation error:", np.mean(cross_
val_errors))
```



Linear SVM:



BVC
GROUP OF INSTITUTIONS
GODAVARIVU - AMALAPURAM - RAJAHMUNDRY

Experiment - 6

Aim: Write a program to implement suppor Vector Machines and principle component Analysis.

Description:

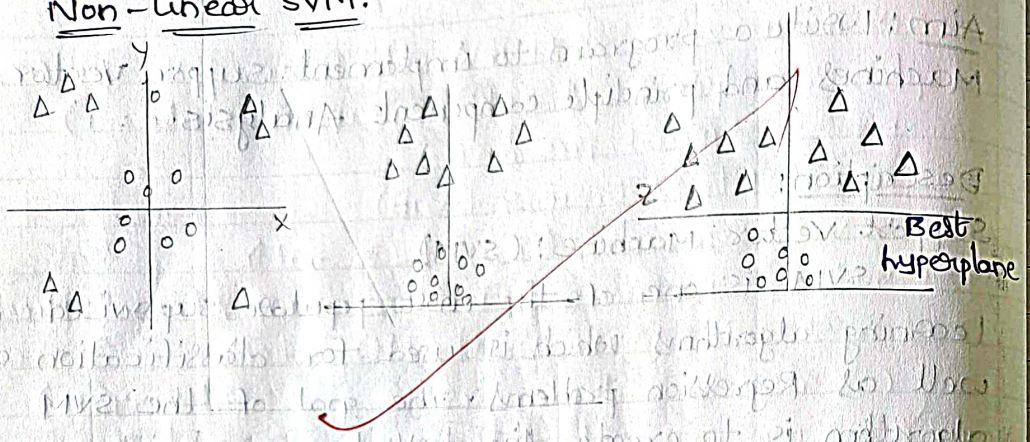
Support Vector Machine: (SVM)

SVM is one of the most popular supervised learning algorithms which is used for classification as well as Regression problems. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes. So that we can easily put the new data point in the correct category in the future. This best decision boundary is called hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors.

Types of SVM:

1. Linear SVM: Linear SVM is used for linearly seperable data, which means if a dataset can be classified into two classes by using a single straight line, then such that data is termed as linearly seperable data and classifier is used called as Linear SVM classifier. Seperate the data is called the Linear SVM.

Non-Linear SVM:



2. Non-Linear SVM: Non-linear SVM is used for non-linearly seperated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier. Not seperate the data is called Non-linear SVM.

Algorithm steps:

Step-1: Input the dataset
Step-2: classify the dataset
Step-3: Apply the SVM machine learning with four kernal functions (Linear polynomials, sigmoid, and Radial Based Function (RBF))
Step-4: Specify the Hyper-plane.
Step-5: If obtained accuracy and validity is NOT acceptable then go to step-4.
Step-6: END.

Program:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import matplotlib.pyplot as plt
```

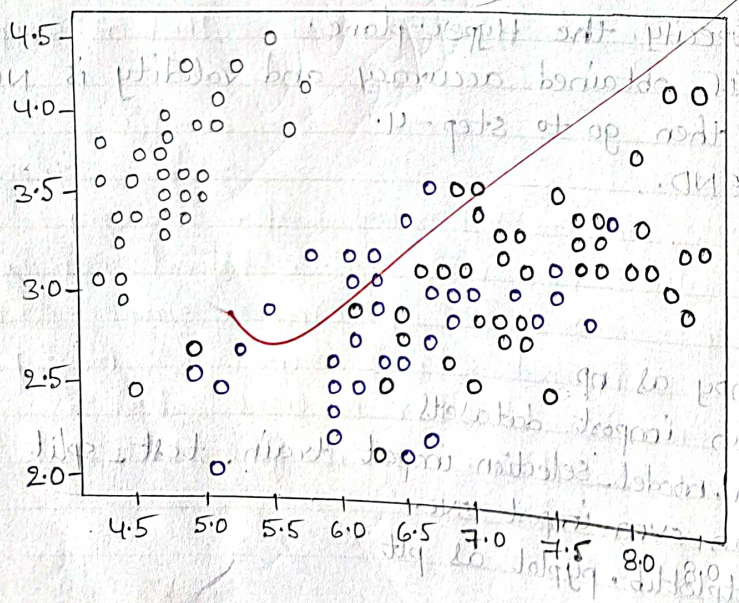
```
iris = datasets.load_iris()
x = iris.data[:, :2]
y = iris.target

x_train, x_test, y_train, y_test = train_test_split
(x, y, test_size = 2.0, random_state = 42)

clf = SVC(kernel='linear', C=1)
clf.fit(x_train, y_train)

accuracy = clf.score(x_test, y_test)
print(f"Accuracy: {accuracy:.2f}")

plt.figure(figsize=(8,6))
plt.scatter(x[:,0], x[:,1], c=y, cmap='viridis')

h=0.02
x_min, x_max = x[:,0].min() -1, x[:,0].max() +1
y_min, y_max = x[:,1].min() -1, x[:,1].max() +1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.
arange(y_min, y_max, h))
```

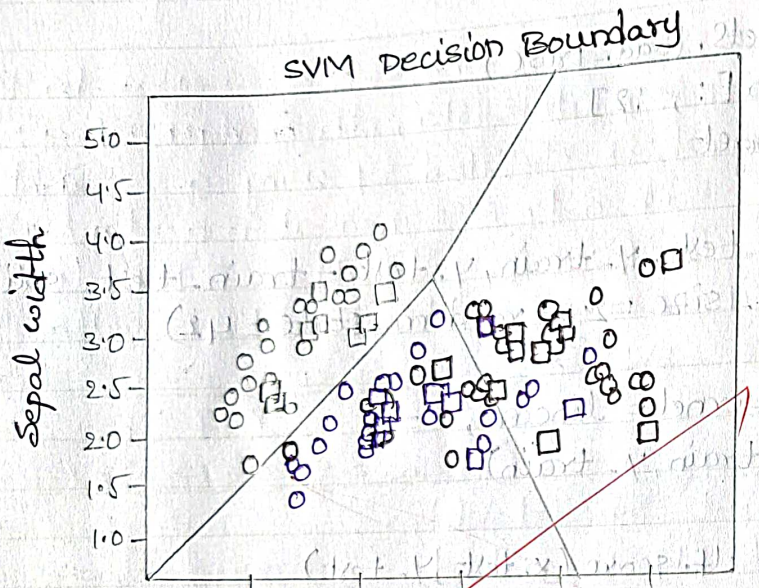
SVC

```
SVC(C=1, kernel='linear')
```

Accuracy : 0.90

<matplotlib.collections.PathCollection at 0x20094b654l0>





25/9/24


```
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap='viridis', alpha=0.5)
plt.scatter(x_train[:, 0], x_train[:, 1], marker='o',
 c=y_train, s=100, edgecolors='k')
plt.scatter(x_test[:, 0], x_test[:, 1], marker='s', c=y_
 test, s=100, edgecolors='k')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('SVM Decision Boundary')
plt.show()
```