National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Ma'am Neelma Naz **Date:** November 27, 2025

Semester: 7th **Group:** 02

CS471 Machine Learning

Lab 10: Decision Trees, K-NNs and Support Vector Machines

		PLO4	PLO5	PLO5	PLO8	PLO9
		CLO4	CLO5	CLO5	CLO6	CLO7
Student Name	**Reg. No**	**Viva / Quiz / Demo** 5 Marks	**Analysis of Data in Report** 5 Marks	**Modern Tool Usage** 5 Marks	**Ethics** 5 Marks	**Individual and Teamwork** 5 Marks
Hanzla Sajjad	403214					
Irfa Farooq	412564					

Introduction

This laboratory exercise will focus on the Scikit Learn (or SKLearn) library for machine learning implementations in python. Scikit Learn contains many useful functions for fitting models using various machine learning techniques such as linear regression, logistic regression, decision trees, support vector machines, k-means clustering, anomaly detection and more.

Objectives

The following are the main objectives of this lab:

- Implement Decision Trees using Scikit learn
- Implement K-Nearest Neighbors from scratch
- Implement K-Nearest Neighbors using Scikit learn
- Implement Support Vector Machines using Scikit learn

Lab Conduct

- Respect faculty and peers through speech and actions
- The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
- In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code/screenshot/plot between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
- Use the tab key to provide the indentation in python.
- When you provide the code in the report, keep the font size at 12

Theory

Scikit Learn is a python library that contains a wide arsenal of functions pertaining to machine learning. It also contains its own datasets for trying out the machine learning algorithms. Scikit learns API interface can be divided into three types: estimator, predictor and transformer. The estimators are used to fit the model in accordance with some algorithm. The predictors use the fitted model to make prediction on test features. The transformers are used for the conversion of data.

Decision trees are a class of machine learning techniques in which a tree is constructed. The nodes of the tree correspond to the features and the branches from those nodes correspond to the values of those features. The leaf nodes contain the final values of the output labels.

Support Vector Machines (SVMs) are another machine learning technique in which a decision boundary between classes is obtained. The decision boundary is defined by a margin which is defined by a subset of the dataset points. The features that define the margin are referred to as support vectors ("vector" is another term for feature). To train an SVM means to determine the support vectors to acquire the margin. Once the margin is obtained, the label of a test example is predicted depending on where it is situated with respect to the margin.

K-nearest neighbors is another technique in which the distances between the dataset points and a test example are computed. The labels from the K shortest distances are used to get the scores for the labels. The label with the highest score becomes the prediction for the test example. Note that in K-NNs, no training (of weights, tree, decision boundary etc.) is performed and the test example is directly predicted just using the distribution of the dataset.

In this lab, you will use the Sci-kit Learn module (except task 2 which you will need to from scratch) to attempt a variety of machine learning techniques. You will need to make extensive use of the Sci-kit Learn documentation for this lab. Download a dataset containing at least 4 feature columns and a label column containing discrete data. You will need to provide the dataset file as part of the submission.

Lab Task 1 – Decision Trees _____

Write a python program that uses functions from the Sci-kit Learn module to train models using decision trees. Try the following feature combinations:

- 2 features combination (any 2 features of your choice)
- 3 features combination (any 3 features of your choice)
- 4 features combination (any 4 features of your choice)

For *each* of the above combination, display the trained trees and also use the trees to make the predictions. Provide the code and all the relevant screenshots of your work.

Code

```
# Importing Important Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier # For Decision Tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn import tree

# Importing dataset
df = pd.read_csv('heart_disease_uci.csv')
print(df.head())
```

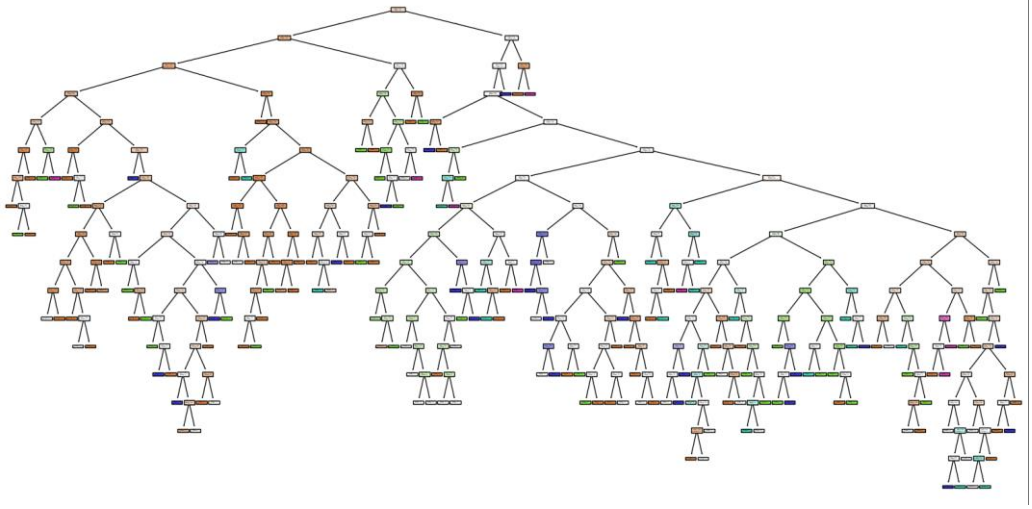
```
# Separating targets and features
Y = df['num']
Y = np.array(Y)
X = df.drop('num', axis=1)
X1 = df[['age', 'trestbps']]
X1 = np.array(X1)
X2 = df[['age', 'chol', 'thalch']]
X2 = np.array(X2)
X3 = df[['chol', 'thalch', 'oldpeak', 'ca']]
X3 = np.array(X3)

# Feature Scaling
sc = StandardScaler()
X1 = sc.fit_transform(X1)
X2 = sc.fit_transform(X2)
X3 = sc.fit_transform(X3)

# Train Test Split
X1_train, X1_test, Y_train, Y_test = train_test_split(
                                                      X1, Y,
                                                      test_size=0.2,
                                                      random_state=0
                                                      )
X2_train, X2_test, Y_train, Y_test = train_test_split(
                                                      X2, Y,
                                                      test_size=0.2,
                                                      random_state=0
                                                      )
X3_train, X3_test, Y_train, Y_test = train_test_split(
                                                      X3, Y,
                                                      test_size=0.2,
                                                      random_state=0
                                                      )
# Trining the tree
dt = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=None,
    random_state=42
)
```

```
# Define labels for confusion matrix based on the unique classes in Y
num_classes = len(np.unique(Y))
cm_column_labels = [f'Predicted:{i}' for i in range(num_classes)]
cm_index_labels = [f'Actual:{i}' for i in range(num_classes)]

# With 2 features
dt.fit(X1_train, Y_train)
Y_pred = dt.predict(X1_test)
plt.figure(figsize=(20,10))
tree.plot_tree(dt, filled=True)
plt.title('Decision Tree with 2 Features')
plt.show()
print('Accuracy score: ', end='')
print(accuracy_score(Y_test, Y_pred))
cm = confusion_matrix(Y_test, Y_pred)
cm_table = pd.DataFrame(data=cm, columns=cm_column_labels,
index=cm_index_labels)
print(cm_table)

# With 3 features
dt.fit(X2_train, Y_train)
Y_pred = dt.predict(X2_test)
plt.figure(figsize=(20,10))
tree.plot_tree(dt, filled=True)
plt.title('Decision Tree with 3 Features')
plt.show()
print('Accuracy score: ', end='')
print(accuracy_score(Y_test, Y_pred))
cm = confusion_matrix(Y_test, Y_pred) # Recalculate cm for this model
cm_table = pd.DataFrame(data=cm, columns=cm_column_labels,
index=cm_index_labels)
print(cm_table)

# With 4 features
dt.fit(X3_train, Y_train)
Y_pred = dt.predict(X3_test)
plt.figure(figsize=(20,10))
tree.plot_tree(dt, filled=True)
plt.title('Decision Tree with 4 Features')
```

```
plt.show()
print('Accuracy score: ', end='')
print(accuracy_score(Y_test, Y_pred))
cm = confusion_matrix(Y_test, Y_pred) # Recalculate cm for this model
cm_table = pd.DataFrame(data=cm, columns=cm_column_labels,
index=cm_index_labels)
print(cm_table)
```

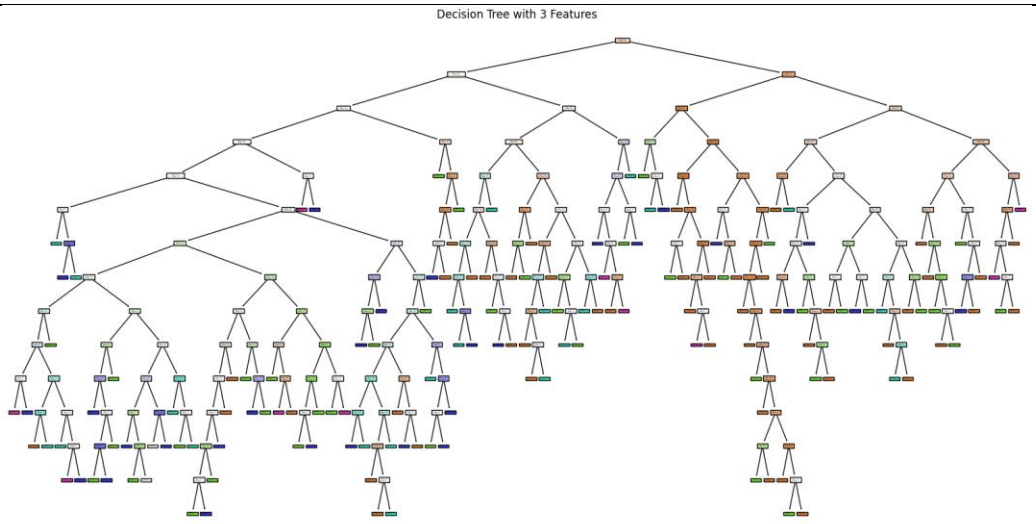
Output Console

```
   id  age     sex    dataset               cp  trestbps   chol    fbs  \
0   1   63    Male  Cleveland   typical angina     145.0  233.0   True
1   2   67    Male  Cleveland     asymptomatic     160.0  286.0  False
2   3   67    Male  Cleveland     asymptomatic     120.0  229.0  False
3   4   37    Male  Cleveland      non-anginal     130.0  250.0  False
4   5   41  Female  Cleveland  atypical angina     130.0  204.0  False

          restecg  thalch  exang  oldpeak        slope   ca  \
0  lv hypertrophy   150.0  False      2.3  downsloping  0.0
1  lv hypertrophy   108.0   True      1.5         flat  3.0
2  lv hypertrophy   129.0   True      2.6         flat  2.0
3          normal   187.0  False      3.5  downsloping  0.0
4  lv hypertrophy   172.0  False      1.4    upsloping  0.0

                thal  num
0      fixed defect    0
1             normal    2
2  reversable defect    1
3             normal    0
4             normal    0
```



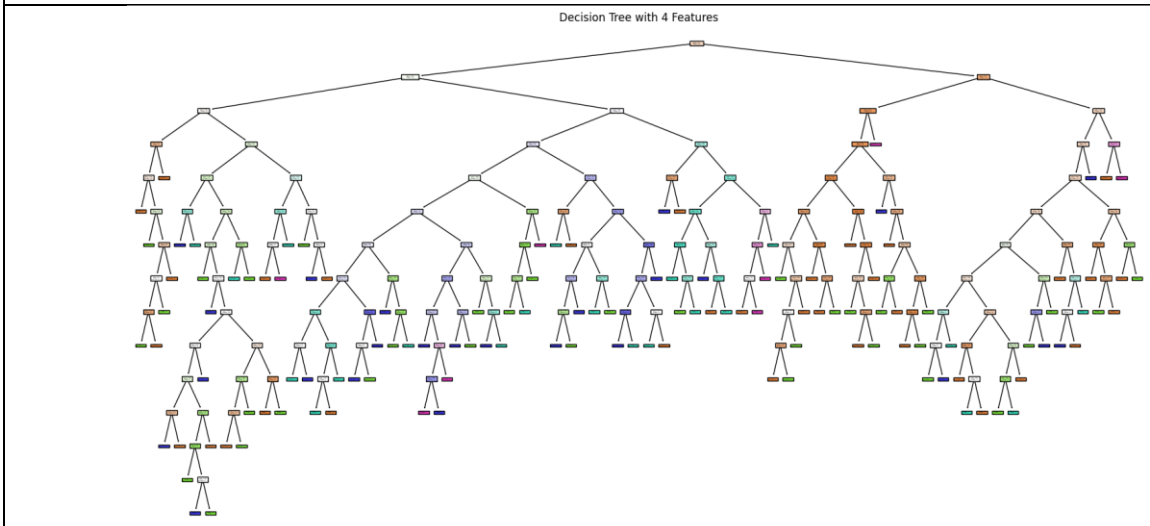
Accuracy score: 0.3333333333333333

	Predicted:0	Predicted:1	Predicted:2	Predicted:3	Predicted:4
Actual:0	20	8	2	3	0
Actual:1	11	2	1	1	1
Actual:2	6	4	1	1	0
Actual:3	3	3	1	2	0
Actual:4	2	0	2	1	0



Accuracy score: 0.30666666666666664

	Predicted:0	Predicted:1	Predicted:2	Predicted:3	Predicted:4
Actual:0	19	4	1	7	2
Actual:1	5	3	4	3	1
Actual:2	3	5	0	2	2
Actual:3	1	5	2	1	0
Actual:4	1	2	2	0	0



Accuracy score: 0.44

	Predicted:0	Predicted:1	Predicted:2	Predicted:3	Predicted:4
Actual:0	21	9	3	0	0
Actual:1	4	7	3	2	0
Actual:2	2	7	1	1	1
Actual:3	1	1	4	3	0
Actual:4	0	0	2	2	1

Lab Task 2 – K-Nearest Neighbors _____

Write a python program from scratch that uses K-NNs to predict the class of a test example for the following cases:

- 2 features combination (any 2 features of your choice)
- 3 features combination (any 3 features of your choice)

For *each* of the above combination, make a scatter plot showing the prediction of your test example. Provide the code and all relevant screenshots.

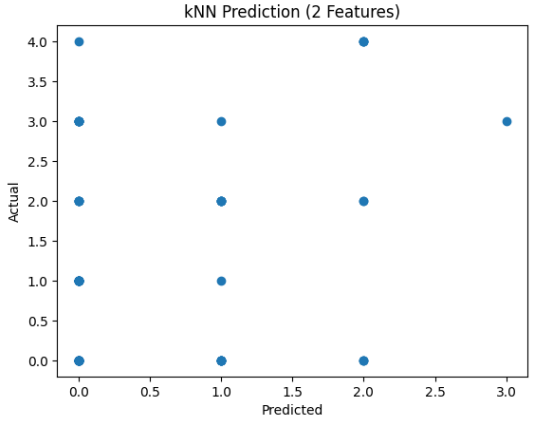
Code

```
# Task 2: Implement kNNs from Scratch
def euclidean_distance(a, b):
    return np.sqrt(np.sum((a - b)**2))

def knn_predict(X_train, Y_train, X_test, k=3):
    Y_pred = []
    for x in X_test:
        # Compute distances to all training points
        distances = [euclidean_distance(x, x_train) for x_train in X_train]
        # Get indices of k nearest neighbors
        k_indices = np.argsort(distances)[:k]
        # Majority vote
        k_labels = [Y_train[i] for i in k_indices]
        pred = max(set(k_labels), key=k_labels.count)
        Y_pred.append(pred)
    return np.array(Y_pred)

# k = 3
Y1_pred = knn_predict(X1_train, Y_train, X1_test, k=3)
Y2_pred = knn_predict(X2_train, Y_train, X2_test, k=3)

def accuracy(Y_true, Y_pred):
    return np.sum(Y_true == Y_pred) / len(Y_true)
```

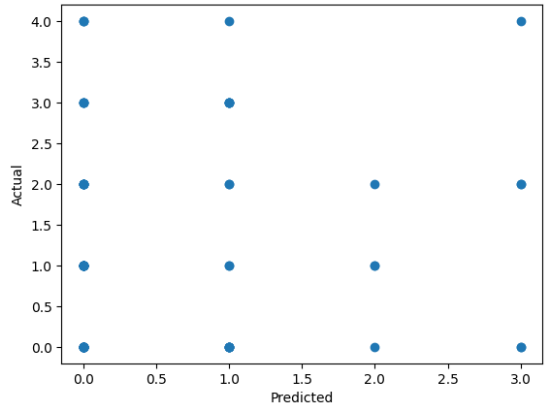
```
# 2-feature
print("Accuracy for 2-feature combination:", accuracy(Y_test, Y1_pred))
plt.scatter(Y1_pred, Y_test)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("kNN Prediction (2 Features)")
plt.show()

# 3-feature
print("Accuracy for 3-feature combination:", accuracy(Y_test, Y2_pred))
plt.scatter(Y2_pred, Y_test)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("kNN Prediction (3 Features)")
plt.show()
```

Output Console

```
Accuracy for 2-feature combination: 0.36
```




```
Accuracy for 3-feature combination: 0.36
```



Lab Task 3 – K-Nearest Neighbors Part 2 _____

Write a python program that uses functions from the Sci-kit Learn module to predict the class of an example using KNNs for the following cases:

- 2 features combination (any 2 features of your choice)
- 3 features combination (any 3 features of your choice)
- 4 features combination (any 4 features of your choice)

For *each* of the above combination, make a scatter plot showing the predictions. Provide the code and all relevant screenshots.

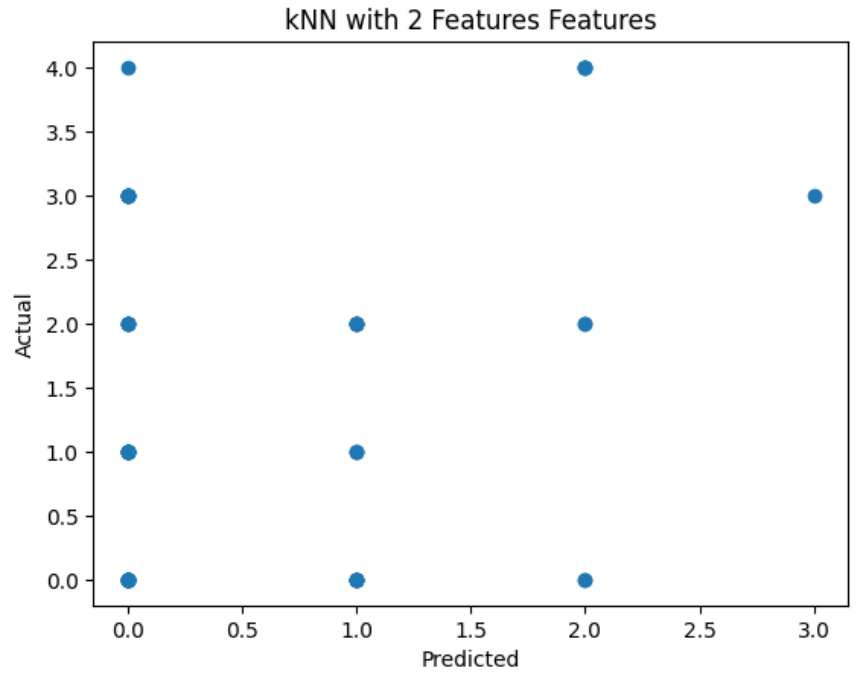
Code

```
# Task 3: Training kNNs
from sklearn.neighbors import KNeighborsClassifier

# using the model for training
knn = KNeighborsClassifier(n_neighbors=3)

# Combining feature arrays
X_train = [X1_train, X2_train, X3_train]
X_test = [X1_test, X2_test, X3_test]
feature_labels = ['2 Features', '3 Features', '4 Features']

# Looping over features = 2, 3, and 4
for i, (X_train, X_test) in enumerate(zip(X_train, X_test)):
    # Train
    knn.fit(X_train, Y_train)

    # Predict
    Y_pred = knn.predict(X_test)

    # Accuracy
    print(f'Accuracy score for {feature_labels[i]}: {accuracy_score(Y_test,
Y_pred)}')
```

```
    # Confusion Matrix
    cm = confusion_matrix(Y_test, Y_pred)
    cm_table = pd.DataFrame(data=cm, columns=cm_column_labels,
index=cm_index_labels)
    print(cm_table)

    # Scatter plot
    plt.scatter(Y_pred, Y_test)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'kNN with {feature_labels[i]} Features')
    plt.show()
```

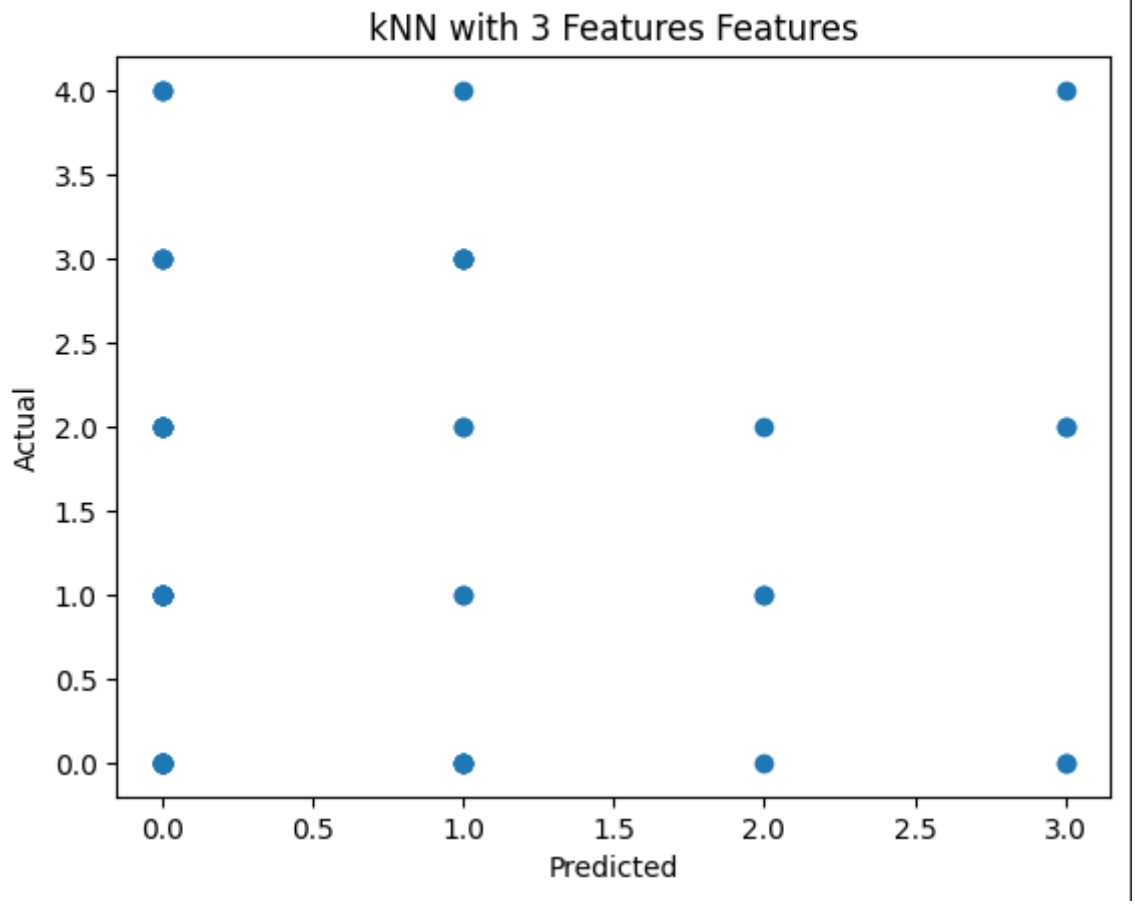
Output Console

```
Accuracy score for 2 Features: 0.41333333333333333
          Predicted:0  Predicted:1  Predicted:2  Predicted:3  Predicted:4
Actual:0           25            6            2            0            0
Actual:1           13            3            0            0            0
Actual:2            5            5            2            0            0
Actual:3            8            0            0            1            0
Actual:4            1            0            4            0            0
```

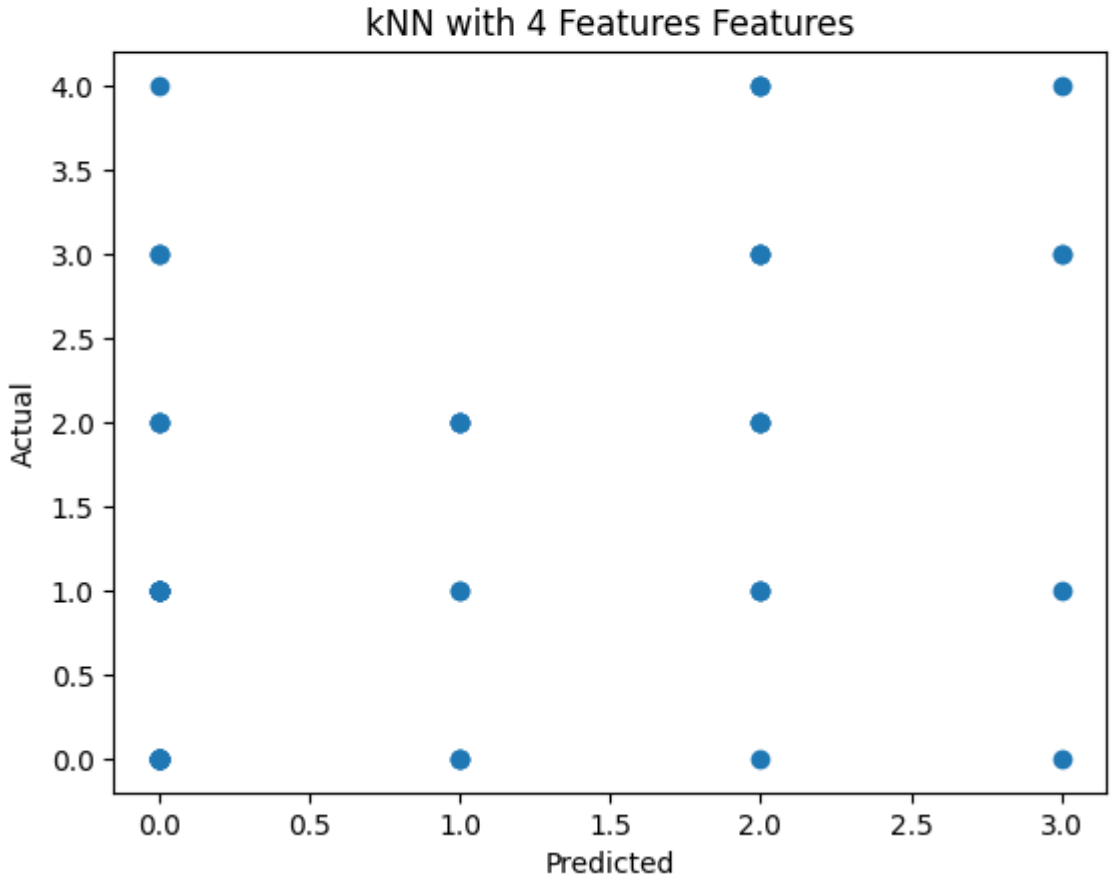


Accuracy score for 3 Features: 0.36

	Predicted:0	Predicted:1	Predicted:2	Predicted:3	Predicted:4
Actual:0	24	6	1	2	0
Actual:1	12	2	2	0	0
Actual:2	7	2	1	2	0
Actual:3	4	5	0	0	0
Actual:4	3	1	0	1	0



```
Accuracy score for 4 Features: 0.48
          Predicted:0  Predicted:1  Predicted:2  Predicted:3  Predicted:4
Actual:0           28            3            1            1            0
Actual:1           10            2            3            1            0
Actual:2            4            4            4            0            0
Actual:3            3            0            4            2            0
Actual:4            1            0            3            1            0
```



Lab Task 4 – Support Vector Machines _____

Write a python program that uses functions from the Sci-kit Learn module to predict the class of an example using Support Vector Machines for the following cases:

- 2 features combination (any 2 features of your choice)
- 3 features combination (any 3 features of your choice)

For *each* of the above combination, make a scatter plot showing the predictions. Provide the code and all relevant screenshots.

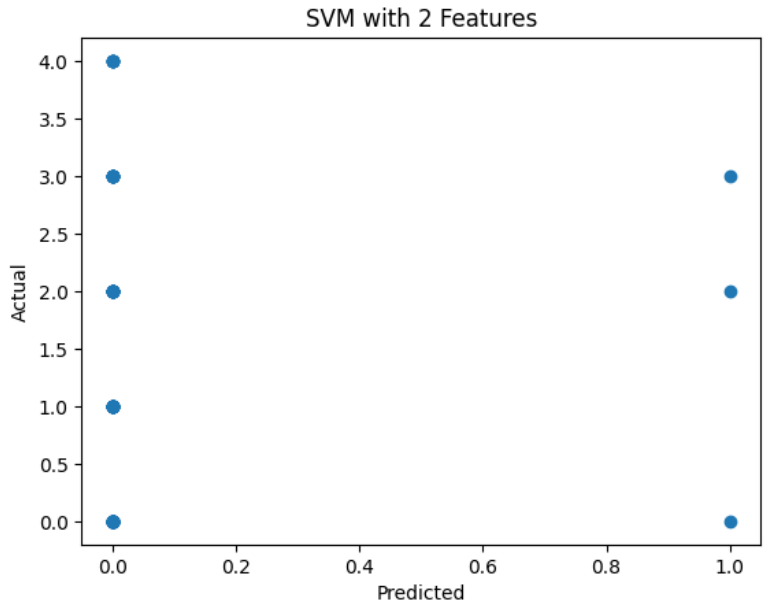
Code

```
# Task 4: Training SVM for training
from sklearn.svm import SVC

# Classification SVM with default RBF kernel
svm_model = SVC(kernel='rbf', C=1, gamma='scale', random_state=42)

# Using your training data
svm_model.fit(X_train, Y_train)

# Combining feature arrays
X_train = [X1_train, X2_train]
X_test = [X1_test, X2_test]
feature_labels = ['2 Features', '3 Features']

# Looping over desired arrays
for i, (X_train, X_test) in enumerate(zip(X_train, X_test)):
    # Train
    svm_model.fit(X_train, Y_train)

    # Predict
    Y_pred = svm_model.predict(X_test)

    # Accuracy
```

```
    print(f'Accuracy score for {feature_labels[i]}: {accuracy_score(Y_test, Y_pred)}')

    # Confusion Matrix
    cm = confusion_matrix(Y_test, Y_pred)
    cm_table = pd.DataFrame(data=cm, columns=cm_column_labels, index=cm_index_labels)
    print(cm_table)

    # Scatter plot
    plt.scatter(Y_pred, Y_test)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'SVM with {feature_labels[i]}')
    plt.show()
```

Output Console

```
Accuracy score for 2 Features: 0.4266666666666667
          Predicted:0  Predicted:1  Predicted:2  Predicted:3  Predicted:4
Actual:0           32            1            0            0            0
Actual:1           16            0            0            0            0
Actual:2           11            1            0            0            0
Actual:3            8            1            0            0            0
Actual:4            5            0            0            0            0
```



Accuracy score for 3 Features: 0.4533333333333333

	Predicted:0	Predicted:1	Predicted:2	Predicted:3	Predicted:4
Actual:0	30	3	0	0	0
Actual:1	12	4	0	0	0
Actual:2	8	4	0	0	0
Actual:3	4	4	1	0	0
Actual:4	3	2	0	0	0