**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Ahmad Salman **Dated:** 22/11/2023

**Semester:** 7th **Section:** BEE 12C

**CS-471 Machine Learning**

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

**Group Members**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4 - CLO4** | | **PLO5 -CLO5** | **PLO8 -CLO6** | **PLO9 -CLO7** |
| **Name** | **Reg. No** | **Viva / Quiz / Lab Performance** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Afif Arif Siddiqi | 344504 |  |  |  |  |  |
| Muhammad Ali Farooq | 331879 |  |  |  |  |  |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |  |
| Danial Ahmad | 331388 |  |  |  |  |  |
| Ehtishaam Tanveer | 333074 |  |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |  |

**Table of Contents**

[2 Decision Trees, K-NNs and Support Vector Machines 3](#_Toc151773431)

[2.1 Introduction 3](#_Toc151773432)

[2.2 Objectives 3](#_Toc151773433)

[2.3 Theory 3](#_Toc151773434)

[3 Lab Tasks 4](#_Toc151773435)

[3.1 Task 1 – Decision Trees 4](#_Toc151773436)

[3.2 Task 2 – K-Nearest Neighbors 6](#_Toc151773437)

[3.3 Task 3 – K-Nearest Neighbors Part 2 9](#_Toc151773438)

[3.4 Task 4 – Support Vector Machines 11](#_Toc151773439)

[4 Conclusion 13](#_Toc151773440)

# Decision Trees, K-NNs and Support Vector Machines

## 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 using Scikit learn
* Implement K-Nearest Neighbors from scratch
* Implement Support Vector Machines using Scikit learn

## 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.

# Lab Tasks

**Importing necessary libraries**

import os

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import plot\_tree, DecisionTreeClassifier

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from collections import Counter

plt.rcParams["figure.figsize"] = (6, 4)

plt.rcParams["font.family"] = "STIXGeneral"

import warnings

warnings.filterwarnings("ignore")

## Task 1 – Decision Trees

Download a dataset containing at least 4 feature columns and a label column containing discrete data. Use functions from Sci-kit learn to train a model using decision trees. Try the following feature combinations:

* 2 features combination
* 3 features combination
* 4 features combination

Display the trained trees and use the trees to make the predictions for all the 3 combinations. Provide the code and all the relevant screenshots of your work.

### TASK 1 CODE STARTS HERE ###

*# Load the dataset*

data = pd.read\_csv("iris.csv")

features = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]

label = "species"

*# Create subsets of the data with different numbers of features*

X\_f2 = data[features[:2]]

X\_f3 = data[features[:3]]

X\_f4 = data[features[:4]]

y = data[label]

X = [X\_f2, X\_f3, X\_f4]

*# Loop over the subsets*

for i, x in enumerate(X):

*# 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)

*# Scale the features*

    scaler = StandardScaler()

    scaler.fit(X\_train)

    X\_train = scaler.transform(X\_train)

    X\_test = scaler.transform(X\_test)

*# Train a decision tree classifier*

    model = DecisionTreeClassifier(*max\_depth*=2)

    model.fit(X\_train, y\_train)

*# Plot the decision tree*

    plot\_tree(model, *feature\_names*=*list*(x.columns))

    plt.title(

*f*"Decision Tree with {i+2} features\n\

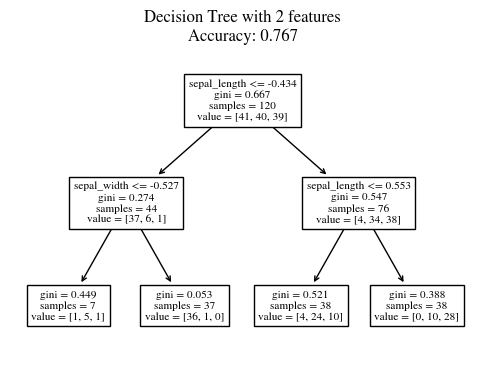
            Accuracy: {model.score(X\_test, y\_test).round(3)}"

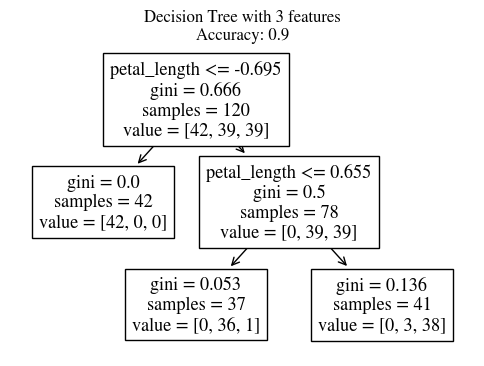
    )

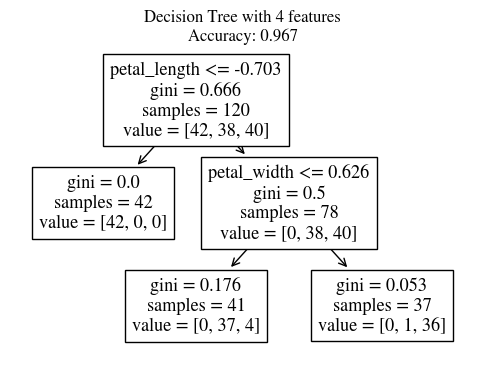
    plt.show()

### TASK 1 CODE ENDS HERE ###

### TASK 1 SCREENSHOT STARTS HERE ###







### TASK 1 SCREENSHOT ENDS HERE ###

## Task 2 – K-Nearest Neighbors

Download a dataset containing at least 3 feature columns and a label column containing discrete data. Write a python program from scratch that uses KNNs to predict the class of an example for the following cases:

* 2 features combination
* 3 features combination

Make a scatter plot showing the predictions in both cases.

### TASK 2 CODE STARTS HERE ###

*# Load the dataset*

data = pd.read\_csv("iris.csv")

features = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]

label = "species"

*# Create subsets of the data with different numbers of features*

X\_f2 = data[features[:2]]

X\_f3 = data[features[:3]]

y = data[label]

X = [X\_f2, X\_f3]

*def* custom\_knn(*X\_train*, *y\_train*, *X\_test*, *k*=5):

    predictions = []

    for test\_point in X\_test:

*# Calculate the euclidean distance*

        distances = np.linalg.norm(X\_train - test\_point, *axis*=1)

        sorted\_indices = np.argsort(distances)

*# Get the labels and counts of the k nearest neighbors*

        k\_nearest\_labels = y\_train.iloc[sorted\_indices[:k]]

        unique\_labels, counts = np.unique(k\_nearest\_labels, *return\_counts*=True)

*# Get the most common label*

        predicted\_label = unique\_labels[np.argmax(counts)]

        predictions.append(predicted\_label)

    return np.array(predictions)

*# Labels for the plot*

leglabels = ["setosa", "versicolor", "virginica"]

*# Loop over the subsets*

for i, x in enumerate(X):

*# Split the data into training and test sets*

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, *test\_size*=0.2)

*# Standardize the features*

    scaler = StandardScaler()

    scaler.fit(X\_train)

    X\_train = scaler.transform(X\_train)

    X\_test = scaler.transform(X\_test)

*# Train a K-nearest neighbors model*

    y\_pred = custom\_knn(X\_train, y\_train, X\_test, *k*=5)

*# Plot the test set and the predictions*

    for k in range(3):

        plt.scatter(

            X\_test[y\_pred == k, 0],

            X\_test[y\_pred == k, 1],

*label*=leglabels[k],

*alpha*=0.5,

        )

    plt.xlabel(x.columns[0])

    plt.ylabel(x.columns[1])

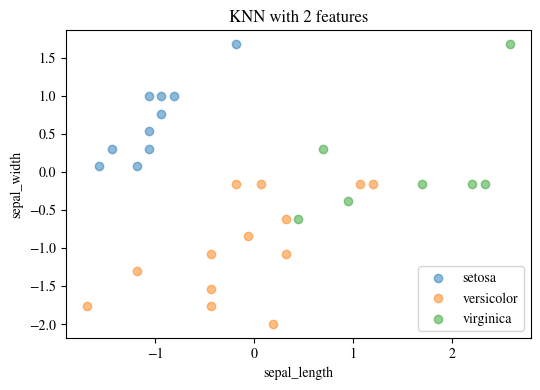
    plt.title(*f*"KNN with {i+2} features")

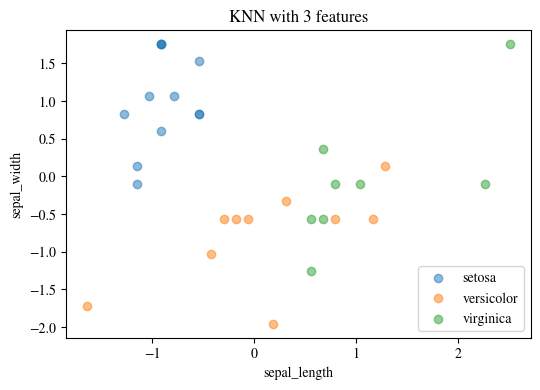
    plt.legend()

    plt.show()

### TASK 2 CODE ENDS HERE ###

### TASK 2 SCREENSHOT STARTS HERE ###





### TASK 2 SCREENSHOT ENDS HERE ###

## Task 3 – K-Nearest Neighbors Part 2

Download a dataset containing at least 4 feature columns and a label column containing discrete data. Use functions from Sci-kit learn to predict the class of an example using KNNs for the following cases:

* 2 features combination
* 3 features combination
* 4 features combination

Make a scatter plot showing the predictions in all cases.

### TASK 3 CODE STARTS HERE ###

*# Load the dataset*

data = pd.read\_csv("iris.csv")

features = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]

label = "species"

*# Create subsets of the data with different numbers of features*

X\_f2 = data[features[:2]]

X\_f3 = data[features[:3]]

X\_f4 = data[features[:4]]

y = data[label]

X = [X\_f2, X\_f3, X\_f4]

*# Labels for the plot*

leglabels = ["setosa", "versicolor", "virginica"]

*# Loop over the subsets*

for i, x in enumerate(X):

*# Split the data into training and test sets*

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, *test\_size*=0.2)

*# Standardize the features*

    scaler = StandardScaler()

    scaler.fit(X\_train)

    X\_train = scaler.transform(X\_train)

    X\_test = scaler.transform(X\_test)

*# Train a K-nearest neighbors model*

    model = KNeighborsClassifier(*n\_neighbors*=5)

    model.fit(X\_train, y\_train)

*# Predict the labels of the test set*

    y\_pred = model.predict(X\_test)

*# Plot the test set and the predictions*

    for k in range(3):

        plt.scatter(

            X\_test[y\_pred == k, 0],

            X\_test[y\_pred == k, 1],

*label*=leglabels[k],

*alpha*=0.5,

        )

        plt.legend()

    plt.xlabel(x.columns[0])

    plt.ylabel(x.columns[1])

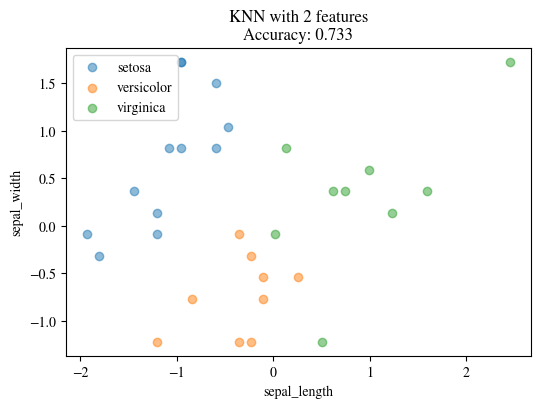
    plt.title(*f*"KNN with {i+2} features\n\

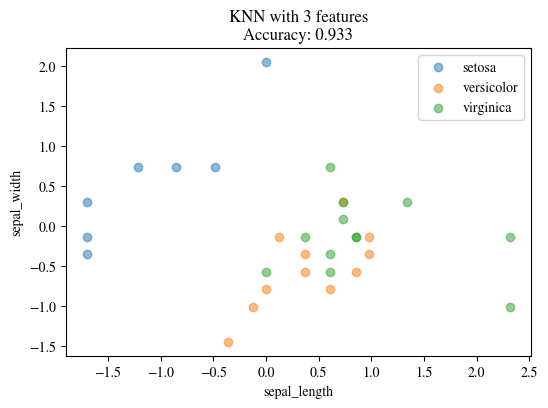
Accuracy: {model.score(X\_test, y\_test).round(3)}")

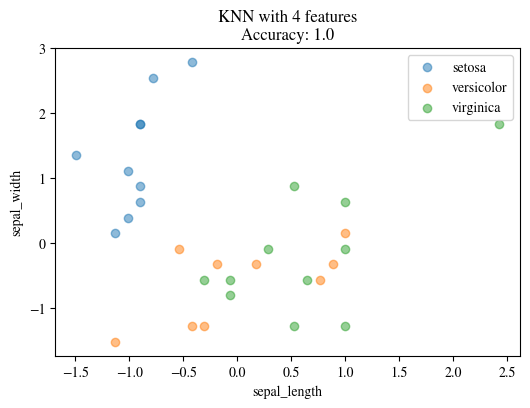
    plt.show()

### TASK 3 CODE ENDS HERE ###

### TASK 3 SCREENSHOT STARTS HERE ###







### TASK 3 SCREENSHOT ENDS HERE ###

## Task 4 – Support Vector Machines

Download a dataset containing at least 4 feature columns and a label column containing discrete data. Use functions from Sci-kit learn to predict the class of an example using Support Vector Machines for the following cases:

* 2 features combination
* 3 features combination

Make a scatter plot showing the predictions in both cases.

### TASK 4 CODE STARTS HERE ###

*# Load the dataset*

data = pd.read\_csv("iris.csv")

features = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]

label = "species"

*# Create subsets of the data with different numbers of features*

X\_f2 = data[features[:2]]

X\_f3 = data[features[:3]]

y = data[label]

X = [X\_f2, X\_f3]

*# Labels for the plot*

leglabels = ["setosa", "versicolor", "virginica"]

*# Loop over the subsets*

for i, x in enumerate(X):

*# Split the data into training and test sets*

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, *test\_size*=0.2)

*# Standardize the features*

    scaler = MinMaxScaler()

    scaler.fit(X\_train)

    X\_train = scaler.transform(X\_train)

    X\_test = scaler.transform(X\_test)

    model = SVC(*kernel*="linear")

    model.fit(X\_train, y\_train)

*# Predict the labels of the test set*

    y\_pred = model.predict(X\_test)

*# Visualize the svm*

    for k in range(3):

        plt.scatter(

            X\_test[y\_pred == k, 0],

            X\_test[y\_pred == k, 1],

*label*=leglabels[k],

*alpha*=0.5,

        )

    plt.xlabel(x.columns[0])

    plt.ylabel(x.columns[1])

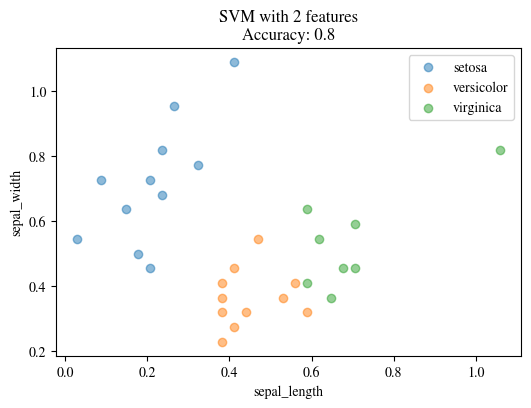
    plt.legend()

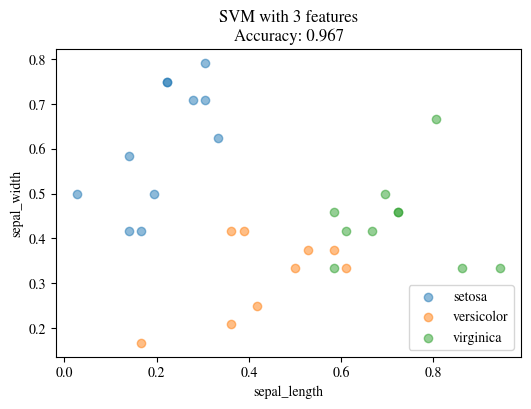
    plt.title(*f*"SVM with {i+2} features\nAccuracy: {model.score(X\_test, y\_test).round(3)}")

    plt.show()

### TASK 4 CODE ENDS HERE ###

### TASK 4 SCREENSHOT STARTS HERE ###





### TASK 4 SCREENSHOT ENDS HERE ###

# Conclusion

In this lab, we have learned how to use Scikit Learn, a powerful python library for machine learning, to implement various algorithms such as decision trees, K-nearest neighbors, and support vector machines. We have also learned how to extract and prepare the training and test datasets, and how to evaluate the performance of the models using different metrics. We have gained hands-on experience in applying machine learning techniques to solve real-world problems using Scikit Learn.