**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Ahmad Salman **Dated:** 18/11/2023

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

**CS-471 Machine Learning**

Lab 9: Introduction to Sci-kit Learn

**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 Introduction to Sci-kit Learn 3](#_Toc151505152)

[2.1 Introduction 3](#_Toc151505153)

[2.2 Objectives 3](#_Toc151505154)

[2.3 Theory 3](#_Toc151505155)

[3 Lab Tasks 4](#_Toc151505156)

[3.1 Task 1 – Linear Regression 4](#_Toc151505157)

[3.2 Task 2 – Logistic Regression 7](#_Toc151505158)

[3.3 Task 3 – K-means Clustering 12](#_Toc151505159)

[4 Conclusion 16](#_Toc151505160)

# Introduction to Sci-kit Learn

## 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:

* Extract and prepare the training and test datasets.
* Implement linear regression using Scikit learn.
* Implement logistic regression using Scikit learn.
* Implement k-means clustering using Scikit learn.
* Implement decision trees 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 libraries and functions:

import os

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

# Import scikit-learn functions

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

from sklearn.linear\_model import LinearRegression, LogisticRegression, Ridge

from sklearn.metrics import mean\_squared\_error, PrecisionRecallDisplay

from sklearn.model\_selection import LearningCurveDisplay, ShuffleSplit

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.cluster import KMeans

import builtins

def print(\*args, \*\*kwargs):

    kwargs["sep"] = ""

    return builtins.print(\*args, \*\*kwargs)

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

# Change to stix

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

import warnings

warnings.filterwarnings("ignore")

## Task 1 – Linear Regression

Download a dataset containing at least 5 feature columns and a label column containing continuous data. Use functions from Sci-kit learn to train a model using linear regression. You will need to split your dataset into training and test portions. Vary the step size and regularization parameters to get at least 6 plots of the training loss and test loss. Lastly, save the weights of the best trained model and use them to make at least five predictions.

Provide the codes and all of the relevant screenshots of your work. Also, give brief explanation of the functions you are using in your codes.

# Load the dataset

data = pd.read\_csv("concrete\_data.csv")

features = ["cement", "slag", "water", "fineagg", "coarseagg"]

label = "strength"

X = data[features]

y = data[label]

print(X.head())

print("\n", y.head())

# Feature scaling

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

y = scaler.fit\_transform(y.values.reshape(-1, 1))

# Split the dataset into training and test portions

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

alpha = [10, 5, 2, 1, 0.5, 0.1]

alpha.sort()

common\_params = {

    "X": X,

    "y": y,

    "train\_sizes": np.linspace(0.1, 1.0, 5),

    "cv": ShuffleSplit(n\_splits=50, test\_size=0.2, random\_state=0),

    "score\_type": "both",

    "n\_jobs": 4,

    "line\_kw": {"marker": "o", "linewidth": 2},

    "std\_display\_style": "fill\_between",

    "scoring": "neg\_mean\_squared\_error",

}

for a in alpha:

    # Train the model using linear regression

    model = Ridge(alpha=a)

    model.fit(X\_train, y\_train)

    # Plot the training and test loss

    LearningCurveDisplay.from\_estimator(

        model,

        \*\*common\_params,

    )

A screenshot of a computer

Description automatically generated

A graph of a number of samples in the training set with Crust in the background

Description automatically generated A graph of a number of samples in the training set with Crust in the background

Description automatically generated

A graph of a number of samples in the training set with Crust in the background

Description automatically generated A graph of a number of samples in the training set

Description automatically generated

A graph of a number of samples in the training set

Description automatically generated A graph of a number of samples in the training set

Description automatically generated

## Task 2 – Logistic Regression

Download a dataset containing at least 5 feature columns and a label column containing discrete data. Use functions from Sci-kit learn to train a model using logistic regression. You will need to split your dataset into training and test portions. Vary the step size and regularization parameters to get at least 6 models of the training. For each model, plot the training loss (vs. epochs), test loss (vs. epochs), precision (vs. epochs) and recall (vs. epochs). Additionally, plot the precision-recall plots for each trained model.

Lastly, save the weights of the best trained model and use them to make at least five predictions. Make a scatter plot for each of your prediction. For this, you will need to show the all of the dataset examples with their labeled classes. Your prediction must be shown as a distinct point in the scatter plots.

Provide the code and all of the relevant screenshots of your work. Also, give brief explanation of the functions you are using in your codes.

# Load the dataset

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

features = [

    "Glucose",

    "BloodPressure",

    "SkinThickness",

    "BMI",

    "DiabetesPedigreeFunction",

]

label = "Outcome"

X = data[features]

y = data[label]

print(X.head())

print("\n", y.head())

# Feature scaling

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

# y = scaler.fit\_transform(y.values.reshape(-1, 1))

# Split the dataset into training and test portions

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

common\_params = {

    "X": X,

    "y": y,

    "train\_sizes": np.linspace(0.1, 1.0, 5),

    "cv": ShuffleSplit(n\_splits=50, test\_size=0.2, random\_state=0),

    "score\_type": "both",

    "n\_jobs": 4,

    "line\_kw": {"marker": "o"},

    "std\_display\_style": "fill\_between",

}

alpha = [10, 5, 2, 1, 0.5, 0.1]

alpha.sort()

for a in alpha:

    model = LogisticRegression(C=a)

    model.fit(X\_train, y\_train)

    LearningCurveDisplay.from\_estimator(

        model,

        \*\*common\_params,

        scoring="neg\_brier\_score",

    )

    LearningCurveDisplay.from\_estimator(

        model,

        \*\*common\_params,

        scoring="precision",

    )

    LearningCurveDisplay.from\_estimator(

        model,

        \*\*common\_params,

        scoring="recall",

    )

    PrecisionRecallDisplay.from\_estimator(

        model,

        X\_test,

        y\_test

    )

# Save the weights of the best trained model

weights = model.coef\_

# print("\nWeights: ", weights)

np.save("weights.npy", weights)

# Load the weights to sklearn model

weights = np.load("weights.npy")

# model = Ridge(alpha=0.1)

# model.coef\_ = weights

# Make predictions

predictions = model.predict(X\_test)

print("\nMean squared error: ", mean\_squared\_error(y\_test, predictions))

A graph of a training set

Description automatically generated with medium confidence A graph of a number of samples in the training set

Description automatically generated

A graph of a number of samples in the training set

Description automatically generated A graph of a logistic regression

Description automatically generated

A graph showing the number of samples in the training set

Description automatically generated A graph of a number of samples in the training set

Description automatically generated

A graph showing a number of samples in the training set

Description automatically generated A graph of a logistic regression

Description automatically generated

A graph of a training set

Description automatically generated with medium confidence A graph of a test results

Description automatically generated with medium confidence

A graph showing a number of samples in the training set

Description automatically generated A graph of a logistic regression

Description automatically generated

A graph of a number of samples in the training set

Description automatically generated A graph of a number of samples in the training set

Description automatically generated

A graph showing a number of samples in the training set

Description automatically generated A graph of a logistic regression

Description automatically generated

A graph of a training set

Description automatically generated with medium confidence A graph of a number of samples in the training set

Description automatically generated

A graph of a training set

Description automatically generated with medium confidence A graph of a logistic regression

Description automatically generated

A graph of a training set

Description automatically generated with medium confidence A graph of testing results

Description automatically generated with medium confidence

A graph of a number of samples in the training set

Description automatically generated A graph of a logistic regression

Description automatically generated

## Task 3 – K-means Clustering

Download a dataset containing at least 4 feature columns. Use functions from Sci-kit learn to perform K-means clustering on the following cases:

•   2 features combination

•   3 features combination

•   4 features combination

For each of the above, perform clustering from k = 2 to K clusters (K is up to your choice). For each combination case, make at least 3 cluster plots. Also, make a graph of cost vs. K for all of the 3 combination cases.  Use the elbow method to determine the best number of clusters in each case.

Provide the code and all of the relevant screenshots of your work. Also, give brief explanation of the functions you are using in your codes.

# Load the dataset

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

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

X = data[features]

y = data["species"]

k\_range = range(2, 8)

# 2 features combination; graph of cost vs. K; elbow method; 3 cluster plots

X2 = X[["sepal\_length", "sepal\_width"]]

distortions = []

for k in k\_range:

    model = KMeans(n\_clusters=k)

    model.fit(X2)

    distortions.append(model.inertia\_)

    y\_pred = model.predict(X2)

    if k <= 4:

        plt.scatter(X2["sepal\_length"], X2["sepal\_width"], c=y\_pred)

        plt.xlabel("Sepal length")

        plt.ylabel("Sepal width")

        plt.title(f"2 features combination; K = {k}")

        plt.grid(alpha=0.3)

        plt.show()

plt.plot(k\_range, distortions)

plt.xlabel("Number of clusters (K)")

# change x axis to integer

plt.xticks(k\_range)

plt.ylabel("Error (Distortion)")

plt.title("Elbow method - 2 features combination")

plt.grid(alpha=0.3)

plt.show()

A graph with yellow and purple dots

Description automatically generated A graph with colored dots

Description automatically generated

A graph of colored dots

Description automatically generated A graph with a blue line

Description automatically generated

Best number of clusters, K, is 5.

# 3 features combination; graph of cost vs. K; elbow method; 3 cluster plots

X3 = X[["sepal\_length", "sepal\_width", "petal\_length"]]

distortions = []

for k in k\_range:

    model = KMeans(n\_clusters=k)

    model.fit(X2)

    distortions.append(model.inertia\_)

    y\_pred = model.predict(X2)

    if k <= 4:

        plt.scatter(X2["sepal\_length"], X2["sepal\_width"], c=y\_pred)

        plt.xlabel("Sepal length")

        plt.ylabel("Sepal width")

        plt.title(f"3 features combination; K = {k}")

        plt.grid(alpha=0.3)

        plt.show()

plt.plot(k\_range, distortions)

plt.xlabel("Number of clusters (K)")

# change x axis to integer

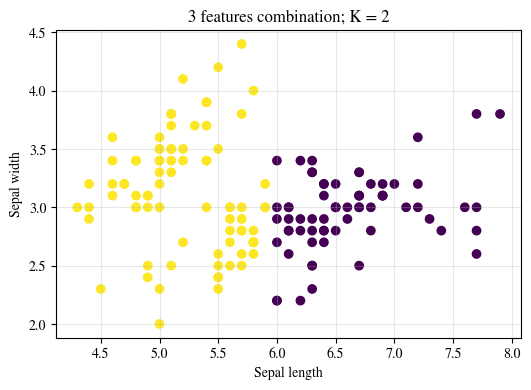
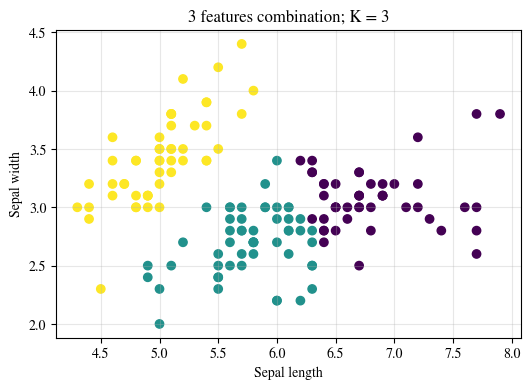
plt.xticks(k\_range)

plt.ylabel("Error (Distortion)")

plt.title("Elbow method - 3 features combination")

plt.grid(alpha=0.3)

plt.show()

A graph of colored dots

Description automatically generated A graph with a blue line

Description automatically generated

Best number of clusters, K, is 5.

# 4 features combination; graph of cost vs. K; elbow method; 3 cluster plots

X4 = X[["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"]]

distortions = []

for k in k\_range:

    model = KMeans(n\_clusters=k)

    model.fit(X2)

    distortions.append(model.inertia\_)

    y\_pred = model.predict(X2)

    if k <= 4:

        plt.scatter(X2["sepal\_length"], X2["sepal\_width"], c=y\_pred)

        plt.xlabel("Sepal length")

        plt.ylabel("Sepal width")

        plt.title(f"4 features combination; K = {k}")

        plt.grid(alpha=0.3)

        plt.show()

plt.plot(k\_range, distortions)

plt.xlabel("Number of clusters (K)")

# change x axis to integer

plt.xticks(k\_range)

plt.ylabel("Error (Distortion)")

plt.title("Elbow method - 4 features combination")

plt.grid(alpha=0.3)

plt.show()

A graph with yellow and purple dots

Description automatically generated A graph with colored dots

Description automatically generated

A graph of different colored dots

Description automatically generated A graph with a blue line

Description automatically generated

Best number of clusters, K, is 5.

# 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 linear regression, logistic regression, and k-means clustering. 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.