**PR1 DW1 Perform the following operations using Python on any opensource dataset**

import pandas as pd

import numpy as np

~data={'NAME':['jay','prince','nisha','neha','raj','riya'],

      'Age':[17,20,21,18,19,25],

      'Gender':['M','M','F','F','M','M'],

      'Marks':[90,75,80,84,71,86] }

~df = pd.DataFrame(data)

~print("Original Data:")

print(df)

~data1 = {

    'NAME': ['jay', 'prince', 'nisha', 'neha', 'raj', 'riya'],

    'Age': [17, 20, 21, 18, 19, 25],

    'Gender': ['M', 'M', 'F', 'F', 'M', 'M'],

    'Marks': [90, 75, np.nan, 84, np.nan, 86]  # Proper missing values }

~df2 = pd.DataFrame(data1)

print("\nData with Missing Marks:")

print(df2)

~df['Gender'] = df['Gender'].map({'M': 0, 'F': 1}).astype(float)

df = df[df['Marks'] > 75]

df = df.drop(['Age'], axis=1)

print("\nCleaned Data (Marks > 75, Gender M/F mapped to 0/1):")

print(df)

or

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

print(df.head())

print("\nMissing Values in Each Column:")

print(df.isnull().sum())

print("\nData Description:")

print(df.describe())

df['species'] = df['species'].map({'setosa': 0, 'versicolor': 1, 'virginica': 2})

df = df[df['petal\_length'] > 1.5]

df = df.drop(['sepal\_width'], axis=1)

print("\nCleaned Dataset:")

print(df.head())

**PR2 DW2 Create an “Academic performance” dataset of students**

~import pandas as pd

import numpy as np

~data\_frame=pd.read\_csv("Academic\_performace.csv")

print(data\_frame)

~print(data\_frame.head())

~print(data\_frame.tail())

~print(data\_frame.describe())

~print(data\_frame.info())

~print(data\_frame.shape)

~print(data\_frame.isnull().any().any())

~print(data\_frame.isnull().sum())

~import seaborn as sns

import matplotlib.pyplot as plt

from scipy import stats

x=data\_frame['Sno']

y=data\_frame['AnnouncementsView']

sns.regplot(x='Sno', y='AnnouncementsView',data=data\_frame)

sns.boxplot(x=data\_frame['AnnouncementsView'])

z=np.abs(stats.zscore(data\_frame['AnnouncementsView']))

print(z)

threshold=3

print(np.where(z>3))

~df=pd.DataFrame({

    'Income':[15000,1000,120000,10000],

    'Age':[25,18,42,51],

    'Department':['HR','Legal','Marketing','Management']})

print(df)

df\_scaled=df.copy()

col\_names=['Income','Age']

features=df\_scaled[col\_names]

**PR3 Descriptive Statistics - Measures of Central Tendency and variability**

~import pandas as pd

import numpy as np

import statistics as st

~data\_frame = pd.read\_csv("/home/ubuntu/DSBDA/test\_AV3.csv")

~print(data\_frame)

~print(data\_frame.info())

~print(data\_frame.mean)

~print(data\_frame['ApplicantIncome'].mean())

~print(data\_frame['ApplicantIncome'].median())

~print(data\_frame.mode())

~print(data\_frame['ApplicantIncome'].mode())

~print(data\_frame.median)

~print(data\_frame['ApplicantIncome'].std())

~print(data\_frame['LoanAmount'].std())

~print(data\_frame['LoanAmount'].var())

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

~print(data\_frame)

~print(data\_frame.info())

~setosa = data\_frame['species'] == 'setosa'

print(data\_frame[setosa].describe())

~virginica = data\_frame['species'] == 'virginica'

print(data\_frame[virginica].describe())

~versicolor = data\_frame['species'] == 'versicolor'

print(data\_frame[versicolor].describe())

~print('\nIris-versicolor')

ver = data\_frame['species'] == 'versicolor'

print(data\_frame[ver].describe())

~print('\nIris-virginica')

virg = data\_frame['species'] == 'virginica'

print(data\_frame[virg].describe())

grouped = data\_frame.groupby('species')

for name, group in grouped:

print(group.select\_dtypes(include=np.number).values.tolist())

**PR4 Linear RegressionModel**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import mean\_squared\_error

df = pd.read\_csv("Boston.csv")

print(df.head())

print(df.describe())

X = df.drop("MEDV", axis=1)

y = df["MEDV"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

print("Model Coefficients:", model.coef\_)

print("\nFirst 10 Predicted vs Actual Prices:")

for i in range(10):

    print(f"Predicted: {y\_pred[i]}, Actual: {y\_test.iloc[i]}")

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

print(f"\nMean Squared Error (MSE): {mse}")

print(f"Root Mean Squared Error (RMSE): {rmse}")

**PR5 logistic regression**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import (

confusion\_matrix,

accuracy\_score,

precision\_score,

recall\_score,

f1\_score

)

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

print(dataset.head())

print(dataset.tail())

X = dataset[['Age', 'EstimatedSalary']]

Y = dataset['Purchased']

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

X, Y, test\_size=0.25, random\_state=0

)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

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

classifier = LogisticRegression(random\_state=0)

classifier.fit(X\_train, Y\_train)

Y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(Y\_test, Y\_pred)

tp = cm[1, 1]

fp = cm[0, 1]

tn = cm[0, 0]

fn = cm[1, 0]

accuracy = accuracy\_score(Y\_test, Y\_pred)

error\_rate = 1 - accuracy

precision = precision\_score(Y\_test, Y\_pred)

recall = recall\_score(Y\_test, Y\_pred)

f1 = f1\_score(Y\_test, Y\_pred)

print("Confusion Matrix:")

print(cm)

print(f"True Positive: {tp}")

print(f"False Positive: {fp}")

print(f"True Negative: {tn}")

print(f"False Negative: {fn}")

print(f"Accuracy: {accuracy:.4f}")

print(f"Error Rate: {error\_rate:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1 Score: {f1:.4f}")

plt.figure(figsize=(6, 6))

sns.heatmap(

cm,

annot=True,

fmt="d",

cmap="Blues",

xticklabels=["Predict No", "Predict Yes"],

yticklabels=["Actual No", "Actual Yes"]

)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

**PR6 Naïve Bayes classificationalgorithm**

import pandas as pd

import numpy as np

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score,f1\_score

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

X = data.drop('species', axis=1)

y = data['species']

print("Value of X=",X)

print( "Value of Y=",y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print("X\_train shape:", X\_train.shape)

print("y\_train shape:", y\_train.shape)

model= GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

TP = cm[0, 0]  # True positives (class 0)

FP = cm[0, 1]  # False positives (class 1)

TN = cm[1, 1]  # True negatives (class 1)

FN = cm[1, 0]

accuracy = accuracy\_score(y\_test, y\_pred)

error\_rate = 1 - accuracy

precision = precision\_score(y\_test, y\_pred, average='macro')

recall = recall\_score(y\_test, y\_pred, average='macro')

f1 = f1\_score(y\_test, y\_pred, average='macro')

print(f"Accuracy: {accuracy:.4f}")

print(f"Error Rate: {error\_rate:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1 Score: {f1:.4f}")

**PR7 Tokenization ,POSTagging, stopwords removal**

import nltk

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer, WordNetLemmatizer

from nltk import pos\_tag

from sklearn.feature\_extraction.text import TfidfVectorizer

import string

~nltk.download('punkt\_tab')

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('averaged\_perceptron\_tagger\_eng')

document =""" Natural Langauage processing is a field of AI It aims to enable to understand ,interpret and generate human langaueg """

tokens= word\_tokenize(document)

pos\_tags = pos\_tag(tokens)

stop\_words = set(stopwords.words('english'))

filtered\_tokens=[word for word in tokens if word.lower() not in stop\_words and word not in

string.punctuation]

stemmer =PorterStemmer()

stemmed\_tokens=[stemmer.stem(word) for word in filtered\_tokens]

lemmatizer = WordNetLemmatizer()

lemmatized\_tokens =[lemmatizer.lemmatize(word)for word in filtered\_tokens]

print("\n original doc\n",document)

print("\n tokens\n", tokens)

print("\n pos tag\n",pos\_tag)

print("\n filtered token after removed stop word removal\n", filtered\_tokens)

print("\n stemmed tokens\n",stemmed\_tokens)

print("\n Lemmetaized Tokens\n",lemmatized\_tokens)

~corpus = [document]

vectorizer = TfidfVectorizer()

x = vectorizer.fit\_transform(corpus)

tfidf\_matrix = x.toarray()

terms = vectorizer.get\_feature\_names\_out()

~print("\nTerm Frequency and Inverse Document Frequency (TF-IDF):")

for i, term in enumerate(terms):

    print(f"Term: {term}, TF-IDF: {tfidf\_matrix[0][i]:.4f}")

**PR8 DV1 titanic 891rows**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

titanic = sns.load\_dataset('titanic')

print(titanic.head())

plt.title('Distribution of Fare Prices of Titanic Passengers')

plt.xlabel('Fare')

plt.ylabel('Density')

plt.show()

plt.figure(figsize=(10, 6))

sns.histplot(titanic['fare'], bins=30, kde=True, color='blue', stat='density')

plt.title('Distribution of Fare Prices of Titanic Passengers')

plt.xlabel('Fare')

plt.ylabel('Density')

plt.grid(True)

plt.show()

**PR9 DV2 titanic Plotaboxplot**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

titanic = sns.load\_dataset('titanic')

titanic.head(10)

titanic.info()

titanic.describe()

titanic.loc[:,["survived","alive"]]

sns.boxplot(x='sex', y='age', data=titanic)

plt.show()

sns.boxplot(x='sex', y='age', hue='survived', data=titanic, palette="Set2")

**PR10 Data Visualization III:- Download the Iris flower dataset**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read\_csv('iris.csv')

print("Features And Types in the iris dataset")

print (df.dtypes)

df.drop(columns='Species').hist(bins=20, figsize=(10,8), color='skyblue',edgecolor='black')

plt.suptitle('Histogram of Each Feature',fontsize=16)

plt.tight\_layout()

plt.show()

plt.figure(figsize=(10,8))

sns.boxplot(x='Species',y='Spetal\_length',data=df)

sns.boxplot(x='Species',y='Spetal\_length',data=df)

sns.boxplot(x='Species',y='Spetal\_length',data=df)

sns.boxplot(x='Species',y='Spetal\_length',data=df)

plt.tight\_layout()

plt.show()

sns.pairplot(df,hue='Species')

plt.suptitle('pairplot of iris dataset',fontsize=16)

plt.tight\_layout()

plt.show()

**PR13 Scala**

***//checks if a number is positive, negative, or zero***

object Number{

def main (args: Array[String]) {

var number = (-100);

if(number==0) {

println("number is zero"); }

else if(number>0) {

println("number is positive"); }

else { println("number is negative"); } } }

**//Print name**

object Name {

def main (args: Array[String]) {

println("my name is name");

} }

**// concanate string**

object String {

def main (args: Array[String]) {

var text: String = "Scala in programming langauge";

println("string is "+text);

} }

**//** **compares two numbers**

object Find\_Number{

def main (args: Array[String]) {

var number1 = 20;

var number2 = 40;

if(number1>number2) {

println("large number is "+number1); }

else

{ println("large number is "+number2); } } }

**// addition of no**

object Addition {

def main(args: Array[String]) {

val num1 = 10

val num2 = 20

val sum = num1 + num2

println("The sum is: " + sum) } }