```
import numpy as np

import pandas as pd

df pd.read_csv(r"C:\Users\310623205040\Downloads\diabetes_datasetcsv.csv")

df.head()

df.shape

df.dtypes

df.info()

df.describe()

sr pd.Series(df['class'])

pos sr.value_counts()

print (pos)

df['Age'].mean()

df['Age'].median()

df['Age'].mode()

df['Age'].std()

df['Age'].var()

df['Age'].skew()

df['Age'].kurt()

import seaborn as sns

import matplotlib.pyplot as plt

sns.histplot(df['Age'], kde=True)

plt.show()



import matplotlib.pyplot as plt
import pandas as pd

import numpy

from sklearn import datasets, linear model

from sklearn.metrics import mean squared error, 2 score

from sklearn, linear model import LogisticRegression

#Te calculate accuracy measures and confusion matrix from sklearn import metrics

from sklearn,model_selection impert train_test_split

diabetes X, diabetes y datasets.load diabetes (return X_ysTrue)

diabetes X diabetes X, пр.лемaxis, 2) #Split the data into training/testing sets

diabetes Xtrain diabetes X-20]

diabetes X test diabetes X-20:

#Split the targets into training/testing sets

diabetes_y_train to diabetes_y:-20)

diabetes and diabetes test Y-20

#Create Linear regression object

regr linear model.Linearitegression()

#Train the model using the training arts regr.fit(diabetes X_train, diabetes y_train)

#Nake predictions using the testing set

diabetes and pred regr.predict(diabetes X_test)

#Create Logistic regression abject Logistic model togisticflegression()

Logistic model.fit(diabetes X train, diabetes_y_train)

#The coefficients

print('Coefficients: \n', regr.coef_)

#The mean sqwined error
```

```
print('Mean squared error: %.2f mean squared error(diabetes y test, diabetes_y pred))

The coefficient of determination 1 is perfect prediction

print('Coefficient of determination: 3.21 r2 score(diabetes_y_test, diabetes_y_pred))

y_predict Logistic_model.predict(diabetes_X_train)

#print("y predict/hat", y predict)

y predict
```

seport pandas as pd

Smport numpy pr

import setplotlib.pyplot plt

smpert seaborn 315

diabetes df = pd.read
cav('https://raw.githubusercontent.com/ummishrab/MachineLearning/master/Datasets/diabet
es.tav")

diabetes df.head()

X = diabetes_df.drop(['Outcome'], axis=1)

Y diabetes_df ['Outcome']

from sklearn.preprocessing import MinMaxScaler

scaler MinMaxScaler (feature_range=(0,1))

scaled_data=scaler.fit_transform(X)

scaled_data

from sklearn, model selection import train test_split

X_train, X test, Ytrain, Y test train test_split(scaled data, Y, test size 8.2, random state =0)

#Linear Regression

from sklearn.linear model import Linear Regression

lin reg LinearRegression()

Initialize Linear Regression model (REMOVE normalize)

lin reg Linear Regression(copy X=True, fit intercept=True, n jobs Nane)

Fit the model

lin_reg.fit(X_train, Y_train)

#Get score

```
r2lin_reg.score(X_test, Y_test)

print (f"Linear Regression Score: (r2:.4f)")

from sklearn.metrics import mean_squared_error, r2_score

predictions = lin_reg.predict(X_test)

predictions

mean_squared_error(Y_test, predictions)

r2_score (Y_test, predictions)
```

```
import pandas as pd

import numpy as np

#Read CSV files

data_1= pd.read_csv(r"C:\Users\310623205040\Downloads\carl.csv")

data 2 pd.read_csv(r"C:\Users\310623205040\Downloads\car2.csv")

Convert to Datoframe

df1 pd.DataFrame(data_1)

df2 pd.Dataframe(data_2)

Ensure both dotoframes have the amount and amount columns

if "Amount in dfl.columns and "Amount1 in df2.columns:

Assign amountl column from df2 to df 1

df1["Amount1df2"Amount"]

Check if prices match

df1"prices_match" np.where(df1"Amount"]=df1'Amount1', True, False)

Compute price difference

df1"price_diff np.where(df1'Anount'] == df1 "Amount", e, df1 'Amount" df1 'Amount1"))

Print Dataframe

print(df1)

else:

print("Error: Required columns ("Amount" in df1 and "Amount 1' in df2) are missing.")
```

```
import numpy as np

import matplotlib.pyplot as plt

5/11

from scipy.stats import norm

%matplotlib inline.

#define constants

in 998.8

sigma 73.10

x1 = 900

x2= 1100

#calculate the z-transform

z1(x1mu) / sigma

z2 = (x2mu) / sigma

x= np.arange(z1, z2, 0.001) range of x in spec

x_all= np.arange(-10, 10, 8.001) # entire range of x, both in and out of spec

#mean 8, stddev 1, since Z-transform was calculated

y norm.pdf(x,0,1)

y2 norm.pdf(x_all,0,1)

#build the plot

fig, ax plt.subplots(figsize=(9,6))

plt.style.use('fivethirtyeight')

ax.plot(x_all,y2)

ax.fill_between (x,y,, alpha=0.3, color='b')

ax.fill_between(x_all,y2,0, alpha=0.1)
ax.set_xlim([-4,4])

ax.set_xlabel('# of Standard Deviations Outside the Mean')

ax.set yticklabels([])

ax.set_title('Normal Gaussian Curve')

plt.savefig('normal_curve.png', dpi=72, bbox_inches'tight')

plt.show()
```

```
import pandas as pd

iris pd.read_csv(r"C:\Users\310623205040\Downloads\iris_data.csv")

print(iris.head())

import pandas as pd

import matplotlib.pyplot as plt

from seaborn import load_dataset

iris load_dataset("iris")

iris.plot(kind='density', subplots True, layout=(4,4), sharex=False, figsize=(10, 8))

plt.show()

import matplotlib

import matplotlib.pyplot as plt

import seaborn as sns

data sns.load_dataset("iris")

data.head()

silky data data.species == 'silky'

virginica data data.species as 'virginica"]

plt.title("Flowers (Setosa & Virginica)")

sns.kdeplot(x=setosa.sepal length, yasetosa sepal width, shadesTrue, crape'lleds", shade
lowestaFalse) sns.kduplot(x=virginica.sepal length, yavirginica.sepal width, shadesTrue,
chaps Blues',

shade lowest False);
```

```
import pandas as pd

#Load the dataset

iris pd.read_csv(r"C:\Users\310623205040\Downloads\iris_data.csv")

#Exclude non-numeric columns

iris_numeric iris.select_dtypes (include=['number'])

#Compute correlation

correlation_matrix iris_numeric.corr()

print(correlation_matrix)

import pandas as pd

import seaborn as sb

import matplotlib.pyplot as plt

#Load dataset

iris pd.read_csv(r"C:\Users\310623205040\Downloads\iris_data.csv")

#Drop the categorical column 'species'

iris_numeric iris.select_dtypes(include=['number'])

#Create the heatmap

fig, ax = plt.subplots (figsize=(8,6)) # Set figure size

plt.title("Iris Correlation Plot")

sb.heatmap(iris_numeric.corr(), annot=True, cmap='coolwarm', ax=ax)

plt.show()

import pandas as pd

iris

pd.read_csv(r"C:\Users\310623205040\Downloads\iris_data.c
print(iris.head())

sv")
```

```
import pandas as pd

import matplotlib.pyplot as pit

Load dataset

iris a pd.read_csv("C:\Users\310623205040\Downloads\iris_data.csv")

Print actual column names to verify

print(iris.columns)

#Standardize column names

iris.columns = iris.columns.str.strip().str.lower()

Define color mapping for species

colors('setosa's 'red', 'versicolor': 'blue', 'virginica's 'green')

Create scatter plot

fig, ax = plt.subplots()

for i in range(len(iris)):

ax.scatter(iris "sepal length"][i], iris['sepal width[i], color colors iris 'species'][i]])

plt.xlabel("Sepal Length")

plt.ylabel("Sepal Width")

plt.title("Iris Dataset Scatter Plot")

plt.show()
```

```
import pandas as pd

iris =

pd.read_csv(r"C:\Users\310623205040\Downloads\iris_data.csv

")

print(iris.head())

fig plt.figure(figsize = (15,20))

ax fig.gca()

iris.hist(ax = ax)

plt.show()
```

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt # To visualize

from mpl toolkits.mplot3d import Axes3D

data pd.read_csv(r"C:\Users\310623205040\Downloads\pima_diabetes.csv")

data.head()

fig plt.figure(figsize=(4,4))

ax fig.add_subplot(111, projection='3')

fig plt.figure()

ax fig.add_subplot(111, projection='3d')

x= data['Age'].values

y= data['Glucose'].values

z= data['Outcome'].values

ax.set_xlabel("Age (Year)")

ax.set_ylabel("Glucose (Reading)")

ax.set_zlabel("Outcome (8 or 1)")

ax.scatter(x, y, z, cz'r', marker='o')

plt.show()
```

```
from mpl_toolkits.basemap import Basemap

import matplotlib.pyplot as plt

fig plt.figure(figsize (12,12))

m = Basemap()

m.drawcoastlines (linewidth=1.0, linestyle='dashed', color='red')

plt.title("Coastlines", fontsize=20)

plt.show()

fig plt.figure(figsize (12,12))

m = Basemap()

m.drawcoastlines (linewidth=1.0, linestyle='solid', color='black')

m.drawcountries (linewidth=1.0, linestyle='solid', color='k')

plt.title("Country boundaries", fontsize=20)

plt.show()

fig plt.figure(figsize (12,12))

m = Basemap()

m.drawcoastlines (linewidth=1.0, linestyle' solid', color='black')

m.drawcountries (linewidth=1.0, linestyle'solid', color='k')

m.drawrivers (linewidth=0.5, linestyle'solid', color='#0000ff")

plt.title("Major rivers", fontsize=20)

plt.show()

fig plt.figure(figsize = (12,12))

m = Basemap()
```

```
m.drawcoastlines (linewidth=1.0, linestyle='solid', color='black')

m.drawcountries (linewidth=1.0, linestyle'solid', color='k')

m.fillcontinents (color='coral', lake_color='aqua', alpha=0.9)

plt.title("Color filled continents", fontsize=20)

plt.show()
```