```
import pandas as pd

import numpy as np

add_df = pd.read_csv("https://github.com/YBIFoundation/Dataset/raw/main/Bike%20Prices.csv")

print("First few rows of the bike price dataset:")
print(add_df.head())

add_df.isnull()

add_df.notnull().sum()

print("\nInformation about the dataset:")
print(add_df.info())

print("\nDescriptive statistics of the dataset:")
print(add_df.describe())

print("\nDimensions of the dataset (rows, columns):", add_df.shape)

print("\nData Types of Variables:")
print(add_df.dtypes)

df = add_df.dropna()

df

df[’col namw with float value , ].astype(int)

df["col name" ].replace({0:"xyz" , 1 : "abc"} , inplace = True)

df
```

```
import pandas as pd

import numpy as np

add_df = pd.read_csv("https://github.com/YBIFoundation/Dataset/raw/main/Bike%20Prices.csv")

print("First few rows of the bike price dataset:")
print(add_df.head())

add_df.isnull()

add_df.notnull().sum()

print("\nInformation about the dataset:")
print(add_df.info())

print("\nDescriptive statistics of the dataset:")
print(add_df.describe())

print("\nDimensions of the dataset (rows, columns):", add_df.shape)

print("\nData Types of Variables:")
print(add_df.dtypes)

df = add_df.dropna()

df

df[’col namw with float value , ].astype(int)

df["col name" ].replace({0:"xyz" , 1 : "abc"} , inplace = True)

df
```

```
import pandas as pd

import numpy as np

add_df = pd.read_csv("https://github.com/YBIFoundation/Dataset/raw/main/Bike%20Prices.csv")

print("First few rows of the bike price dataset:")
print(add_df.head())

add_df.isnull()

add_df.notnull().sum()

print("\nInformation about the dataset:")
print(add_df.info())

print("\nDescriptive statistics of the dataset:")
print(add_df.describe())

print("\nDimensions of the dataset (rows, columns):", add_df.shape)

print("\nData Types of Variables:")
print(add_df.dtypes)

df = add_df.dropna()

df

df[’col namw with float value , ].astype(int)

df["col name" ].replace({0:"xyz" , 1 : "abc"} , inplace = True)

df
```

```
import pandas as pd

bike_df = pd.read_csv("SampleSuperstore.csv")

#Display first few rows of the dataset
print("First few rows of the bike price dataset:")
print(bike_df.head())

grouped_stats = bike_df.groupby(’Selling_Price’)[’Year’].describe()

print("\nSummary statistics of selling price grouped by year:")
print(grouped_stats)

mean_price_by_year = bike_df.groupby(’Selling_Price’)[’Year’].mean().tolist()
# Display the list of mean ages for each passenger class
print("\nMean Price for each year:")
print(mean_price_by_year)

import seaborn as sns
import pandas as pd

iris = pd.read_csv(’https://github.com/YBIFoundation/Dataset/raw/main/IRIS.csv’)

print("First few rows of the Iris dataset:")
print(iris.head())

# 1. Display basic statistical details for ’Iris-setosa’
setosa_stats = iris[iris[’species’] == ’setosa’].describe()

# Display the statistical details for ’Iris-setosa’
print("\nStatistical details for ’Iris-setosa’:")
print(setosa_stats)

# 2. Display basic statistical details for ’Iris-versicolor’
versicolor_stats = iris[iris[’species’] == ’versicolor’].describe()


# Display the statistical details for ’Iris-versicolor’
print("\nStatistical details for ’Iris-versicolor’:")
print(versicolor_stats)

# 3. Display basic statistical details for ’Iris-virginica’
virginica_stats = iris[iris[’species’] == ’virginica’].describe()

# Display the statistical details for ’Iris-virginica’
print("\nStatistical details for ’Iris-virginica’:")
print(virginica_stats)
```

```
# imports
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#from sklearn.datasets import load_boston
import warnings
warnings.filterwarnings(’ignore’)
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error


boston_data = pd.read_csv("HousingData.csv")

boston_data.info()

boston_data.isnull().sum()

boston_data.isna().sum().sum()

data = boston_data.dropna()
data

plt.figure(figsize=(12,8))
sns.heatmap(data.corr().abs(), annot= True, cmap= ’coolwarm’);


train_df= data[[’LSTAT’, ’PTRATIO’,’RM’, ’TAX’,’INDUS’,’MEDV’]]
train_df


x = train_df[[’LSTAT’, ’RM’, ’TAX’, ’INDUS’]]
y = train_df[’MEDV’]

scaler = StandardScaler()
x = scaler.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, shuffle=True)

model = LinearRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

y_pred[:5]

y_test[:5]

mse = mean_squared_error(y_test, y_pred)
mse

sns.regplot(x = y_test, y = y_pred, ci= 95)
```

```
#imports
import numpy as np
import pandas as pd
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report

data = pd.read_csv("Social_Network_Ads.csv")

data.sample(5)

data.info()

data

data.isna().sum()

# Target label : ’Purchased’
sns.countplot(data = data, x = ’Purchased’);

# Finding useful features
sns.heatmap(data[[’Age’,’EstimatedSalary’,’Purchased’]].corr(), annot = True, cmap= ’coolwarm’ );

features = data[[’Age’, ’EstimatedSalary’]]
label = data[’Purchased’]

scaler = StandardScaler()
features = scaler.fit_transform(features)

x = features
y = label

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

y_pred

sns.heatmap(confusion_matrix(y_test, y_pred), annot= True)

print(classification_report(y_test, y_pred))
```

```
#imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import warnings
warnings.filterwarnings(’ignore’)

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report

iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data[’target’] = iris.target
data.head()


data.sample(5)

set(iris.target), iris.target_names

X_train, X_test, y_train, y_test = train_test_split(data.drop(’target’, axis=1), data[’target’], test_size=0.2,
random_state=42)

model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
y_pred

sns.heatmap(confusion_matrix(y_test, y_pred), annot = True);

print(classification_report(y_test, y_pred))
```

```
import nltk
from nltk import word_tokenize, sent_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

nltk.download(’stopwords’)
nltk.download(’wordnet’)
nltk.download(’punkt’)
nltk.download(’averaged_perceptron_tagger’)
nltk.download(’omw-1.4’)

text = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce commodo mauris id justo condimentum dignissim. Nullam placerat semper dapibus. Pellentesque ac risus nulla. Phasellus ut dapibus nunc, id aliquam dolor."

print(word_tokenize(text))

print(sent_tokenize(text))

to_tag = word_tokenize(text)

print(pos_tag(to_tag))

stop_words = set(stopwords.words("english"))
print(stop_words)

to_clean = word_tokenize(text)
to_clean


no_stopwords_text = []
for token in to_clean:
    if(token not in stop_words):
        no_stopwords_text.append(token)

print(no_stopwords_text)

stemmer = PorterStemmer()

stemmed_words = []
for token in no_stopwords_text:
    stemmed_word = stemmer.stem(token)
    stemmed_words.append(stemmed_word)

print(stemmed_words)

lemmatizer = WordNetLemmatizer()
lemmatized_words = []
for token in no_stopwords_text:
    lemmatized = lemmatizer.lemmatize(token)  # Assuming you want to lemmatize verbs (you can change the ’pos’ argument as needed)
    lemmatized_words.append(lemmatized)
```

```
print(lemmatized_words)

vectorizer = TfidfVectorizer()

corpus = [
    "I love to eat pizza",
    "Pizza is my favorite food",
    "I enjoy eating pizza with friends",
    "I like to have pizza for dinner",
    "Pizza toppings include cheese, pepperoni, and mushrooms"
]

vectorizer = TfidfVectorizer()
vectorizer

tfidf_matrix = vectorizer.fit_transform(corpus)

feature_names = vectorizer.get_feature_names_out()


print(tfidf_matrix.toarray())

print(feature_names)
```

```
#imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv(’train.csv’)
data

data.head(5)

data.info()

data.isna().sum().sum()

data.isnull().sum()

sns.countplot(x =’Survived’, data = data)

sns.countplot(data=data, x=’Sex’, hue= ’Survived’)

sns.heatmap(data.corr(), annot= True, cmap= ’coolwarm’, linewidths = 1, linecolor = ’black’);

sns.regplot(data=data,x=’Pclass’,y=’Fare’)

sns.histplot(data,x="Fare",bins=15,binwidth=20)

sns.histplot(data = data, x = ’Fare’, hue = ’Survived’,kde = True);
```

```
#imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv(’train.csv’)
data.sample(5)

data.isna().sum()

#Age has a lot of null values and is one of the attributes we need to use.
sns.heatmap(data.corr(), annot = True);


age_null_mask = data[’Age’].isnull()

age_mean = data[’Age’].mean()
age_std = data[’Age’].std()

# generate random ages based on the age distribution of the dataset
age_random = np.random.normal(loc=age_mean, scale=age_std, size=age_null_mask.sum())

# fill in missing age values with random ages
data.loc[age_null_mask, ’Age’] = age_random

# 177 normal random values generated for 177 missing data points
age_random.size

data.isna().sum()

data.sample(7)

sns.boxplot(x=’Sex’, y=’Age’, hue=’Survived’, data=data);
```

```
#imports
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
iris = load_iris()

data = pd.DataFrame(iris.data, columns = iris.feature_names)
data[’label’] = iris.target
data.sample(5)

data.info()

sns.histplot(data = data, x = ’sepal length (cm)’, kde= True);

sns.histplot(data = data, x = ’sepal width (cm)’, kde= True, color = "orange");

sns.histplot(data = data, x = ’petal length (cm)’, kde= True, color = "green");

sns.histplot(data = data, x = ’petal width (cm)’, kde= True, color = "red");

figure = plt.figure(figsize = (12,8))
sns.boxplot(data= data)
plt.show()

from matplotlib.cbook import boxplot_stats
stats = boxplot_stats(data[’sepal width (cm)’])
stats

outliers = stats[0].get("fliers")

outliers
```