```
#EXP1

import pandas as pd
from sklearn.linear_model import LinearRegression

# Load the dataset
data = pd.read_csv("advertising.csv")

# Selecting only the TV and Sales columns
tv_data = data[['TV', 'Sales']]

# Splitting the data into independent (X) and dependent (y) variables
X = tv_data[['TV']]
y = tv_data['Sales']

# Create and fit the model
model = LinearRegression()
model.fit(X, y)

# Get the slope (coefficient) and intercept
slope = model.coef_[0]
intercept = model.intercept_

print("Slope (Coefficient):", slope)
print("Intercept:", intercept)



#EXP2



import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt

x=df.iloc[:,[1,2]].values
y=df.iloc[:,-1].values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
import statsmodels.api as sm
x_train_sm=sm.add_constant(x_train)
model=sm.OLS(y_train,x_train_sm).fit()
print("Model Coefficients=",model.params)
print("F-statistic=",model.fvalue)
print("T-statistic=",model.tvalues)
print("Residual sum of squares=",model.ssr)

#EXP3


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('smarket.csv')
df.info()
df['Direction'] = df['Direction'].replace("Up",0)
df['Direction'] = df['Direction'].replace("Down",1)
X = df.iloc[:,2:8].values
Y = df.iloc[:,-1].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3)
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
```

```
model.fit(x_train, y_train)

KNeighborsClassifier(n_neighbors=3)

y_pred = model.predict(x_test)

from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(y_test, y_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels =
[False, True])

cm_display.plot()

plt.show()


from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))



#EXP5

import pandas as pd

data = pd.read_csv("Smarket data.csv", index_col=0)

from sklearn.neighbors import KNeighborsClassifier


# Define predictors and response

X = data[['Lag1', 'Lag2', 'Lag3', 'Lag4', 'Lag5', 'Volume']]

y = data['Direction']


# Initialize KNN model

knn = KNeighborsClassifier(n_neighbors=3)


# Fit the model

knn.fit(X, y)

from sklearn.preprocessing import LabelEncoder


# Initialize LabelEncoder
```

```
label_encoder = LabelEncoder()

# Encode the categorical labels in y
y_encoded = label_encoder.fit_transform(y)

# Now, y_encoded contains numerical labels instead of 'Up' and 'Down'
# Fit linear regression model
model = sm.OLS(y_encoded, X).fit()

# Compute metrics
cp = model.mse_total / (model.mse_total - 2 * model.mse_resid)
aic = model.aic
adjusted_r_squared = model.rsquared_adj
bic = model.bic
print("Mallow's Cp:", cp)
print("AIC:", aic)
print("Adjusted R-squared:", adjusted_r_squared)
print("BIC:", bic)

#EXP6
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
college_data = pd.read_csv("College_Data.csv")

# Select quantitative variables for plotting histograms
quantitative_vars = ['Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc', 'F.Undergrad', 'P.Undergrad']

# Define the number of bins for histograms
num_bins_list = [10, 20, 30]
```

```
# Plot histograms for selected variables with different numbers of bins
for var in quantitative_vars:

    plt.title(f'Histogram of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')
    for num_bins in num_bins_list:
        plt.hist(college_data[var], bins=num_bins, alpha=0.5, label=f'{num_bins} bins')
    plt.legend()

    plt.show()


#EXP8


import pandas as pd
from scipy.stats import pearsonr, spearmanr, kendalltau
college_data = pd.read_csv('College.csv')
selected_columns = ['Private', 'Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc']
pearson_corr = college_data[selected_columns].corr(method='pearson')

# Calculating Spearman correlation
spearman_corr = college_data[selected_columns].corr(method='spearman')

# Calculating Kendall correlation
kendall_corr = college_data[selected_columns].corr(method='kendall')
print("\nPearson Correlation:")
print(pearson_corr)
```

```
print("\nSpearman Correlation:")
print(spearman_corr)

print("\nKendall Correlation:")
print(kendall_corr)

#EXP9



import numpy as np
import scipy.stats as stats

# Generate random data for demonstration
np.random.seed(42)
data1 = np.random.normal(loc=10, scale=2, size=100)
data2 = np.random.normal(loc=12, scale=2, size=100)

# Simple Hypothesis Testing (one-sample t-test)
t_stat, p_value = stats.ttest_1samp(data1, 10)
print("Simple Hypothesis Testing:")
print("t-statistic:", t_stat)
print("p-value:", p_value)

# Student’s t-test (independent samples t-test)
t_stat, p_value = stats.ttest_ind(data1, data2)
print("\nStudent’s t-test:")
print("t-statistic:", t_stat)
print("p-value:", p_value)

# Paired t-test
t_stat, p_value = stats.ttest_rel(data1, data2)
```

```
print("\nPaired t-test:")
print("t-statistic:", t_stat)
print("p-value:", p_value)

# Correlation
corr_coef, p_value = stats.pearsonr(data1, data2)
print("\nCorrelation:")
print("Correlation coefficient:", corr_coef)
print("p-value:", p_value)

# Tests for Association (Chi-square test of independence)
# Create contingency table for demonstration
observed = np.array([[30, 10], [20, 40]])
chi2_stat, p_value, dof, expected = stats.chi2_contingency(observed)
print("\nTests for Association (Chi-square test of independence):")
print("Chi-square statistic:", chi2_stat)
print("p-value:", p_value)



#EXP10000000000000000000


import numpy as np

# Define a square matrix
A = np.array([[4, 2],
         [1, 3]])

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)
```

```
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:")
print(eigenvectors)
```