import warnings

warnings.filterwarnings('ignore')

# Import the numpy and pandas package

import numpy as np

import pandas as pd

# Data Visualisation

import matplotlib.pyplot as plt

import seaborn as sns

advertising = pd.DataFrame(pd.read\_csv("C:\\Users\\unic\\Downloads\\advertising (1).csv"))

advertising.head()

#advertising.shape()

advertising.info()

advertising.describe()

# Checking Null values

advertising.isnull().sum()\*100/advertising.shape[0]

# There are no NULL values in the dataset, hence it is clean.

# Outlier Analysis

fig, axs = plt.subplots(3, figsize = (5,5))

plt1 = sns.boxplot(advertising['TV'], ax = axs[0])

plt2 = sns.boxplot(advertising['Newspaper'], ax = axs[1])

plt3 = sns.boxplot(advertising['Radio'], ax = axs[2])

plt.tight\_layout()

sns.boxplot(advertising['Sales'])

plt.show()

# Let's see how Sales are related with other variables using scatter plot.

sns.pairplot(advertising, x\_vars=['TV', 'Newspaper', 'Radio'], y\_vars='Sales', height=4, aspect=1, kind='scatter')

plt.show()

# Let's see the correlation between different variables.

sns.heatmap(advertising.corr(), cmap="YlGnBu", annot = True)

plt.show()

X = advertising['TV']

y = advertising['Sales']

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size = 0.7, test\_size = 0.3, random\_state = 100)

# Let's now take a look at the train dataset

X\_train.head()

y\_train.head()

import statsmodels.api as sm

# Add a constant to get an intercept

X\_train\_sm = sm.add\_constant(X\_train)

# Fit the resgression line using 'OLS'

lr = sm.OLS(y\_train, X\_train\_sm).fit()

# Print the parameters, i.e. the intercept and the slope of the regression line fitted

lr.params

# Performing a summary operation lists out all the different parameters of the regression line fitted

print(lr.summary())

plt.scatter(X\_train, y\_train)

plt.plot(X\_train, 6.948 + 0.054\*X\_train, 'r')

plt.show()

y\_train\_pred = lr.predict(X\_train\_sm)

res = (y\_train - y\_train\_pred)

fig = plt.figure()

sns.distplot(res, bins = 15)

fig.suptitle('Error Terms', fontsize = 15) # Plot heading

plt.xlabel('y\_train - y\_train\_pred', fontsize = 15) # X-label

plt.show()

plt.scatter(X\_train,res)

plt.show()

# Add a constant to X\_test

X\_test\_sm = sm.add\_constant(X\_test)

# Predict the y values corresponding to X\_test\_sm

y\_pred = lr.predict(X\_test\_sm)

y\_pred.head()

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

#Looking at the RMSE

#Returns the mean squared error; we'll take a square root

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

#2.019296008966232

#Checking the R-squared on the test set

r\_squared = r2\_score(y\_test, y\_pred)

r\_squared

plt.scatter(X\_test, y\_test)

plt.plot(X\_test, 6.948 + 0.054 \* X\_test, 'r')

plt.show()