MODULE 4: Linear Regression - Assignment

In this assignment, we will show our understanding on how to apply the Linear Regression model to a selected dataset in Python.

Please, find the dataset on Canvas in the FILES section.

Description of the `advertising` data:

Advertising data sales (in thousands of units) for a particular product advertising budget (in thousands of dollars) for TV.

STEP 1: Pre-process the dataset

In this step, you need to do the following:

- Import the necessary libraries
- Load the dataset
- Explore the dataset
- Summarize the dataset
- Analyze the dataset

Import the necessary libraries

```
#Import the libraries
import pandas as pd
import numpy as py
import seaborn as sn
import matplotlib.pyplot as plt

```

Load the dataset

Comments:

```
#Load dataset
df = pd.read_csv("/content/advertising.csv")
```

Explore the dataset

Note:

Here we can also check outliers, missing values, etc. You can apply all the DATA CLEANING skills that you learn in the Advanced Python course.

```
#Show the head of the data
df.head()

```

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	12.0
3	151.5	16.5
4	180.8	17.9

Next steps: New interactive sheet

```
#Show the shape of the data
df.shape

```

```
(200, 2)
```

```
#Get the info of the dataset with "advertising.info()"
df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   TV      200 non-null    float64
 1   Sales   200 non-null    float64
dtypes: float64(2)
memory usage: 3.3 KB
```

In the next step we will find the variable tyoes which we can also find them using `.info()` as we did in the last step.

```
#Show the types of the variables
var_types = df.dtypes
var_types

```

```
             0
TV     float64
Sales  float64

dtype: object
```

```
#Show the total null values in the data frame
df.isnull().sum()

```

```
       0
TV     0
Sales  0

dtype: int64
```

Summarize the data

```
#Use describe() for summarizing the data
#Round it to 2 decimals places
round(df.describe(),2)

```

	TV	Sales
count	200.00	200.00
mean	147.04	15.13
std	85.85	5.28
min	0.70	1.60
25%	74.38	11.00
50%	149.75	16.00
75%	218.82	19.05
max	296.40	27.00

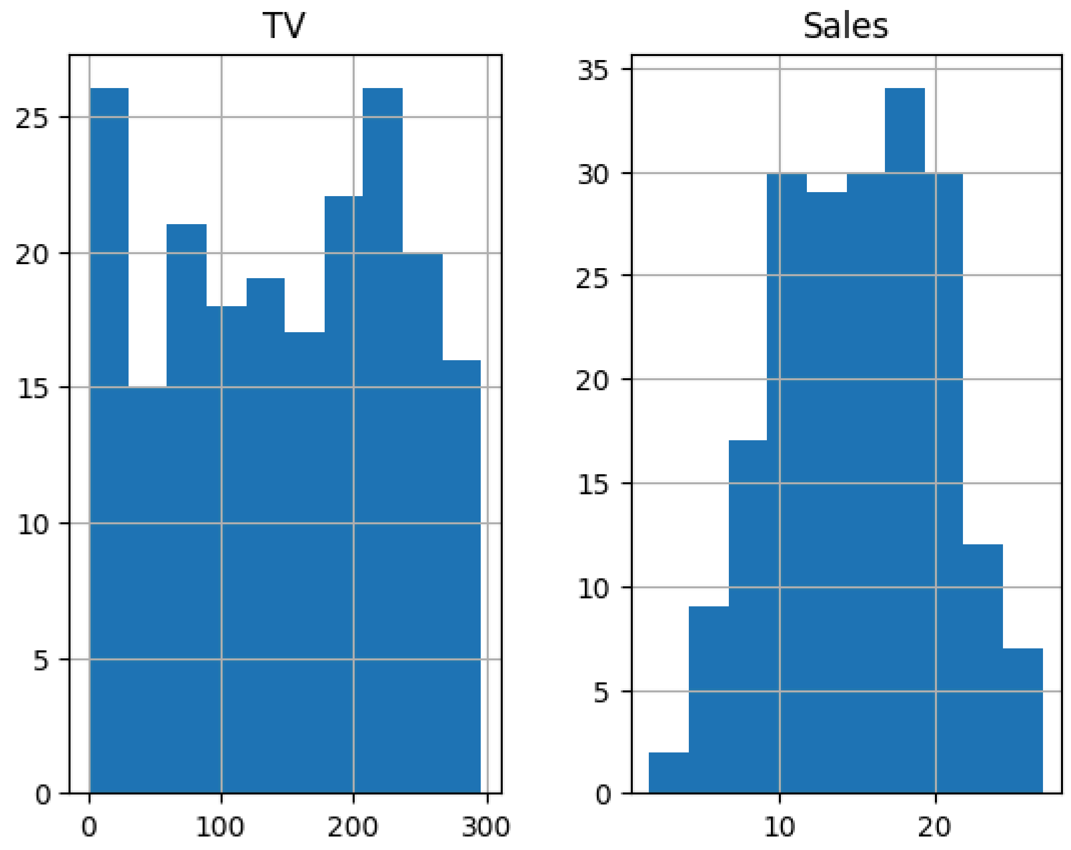
Analyze the data

Visualize the data

```
#Plot Histograms
hist_visual = df.hist(bins=10)
hist_visual

```

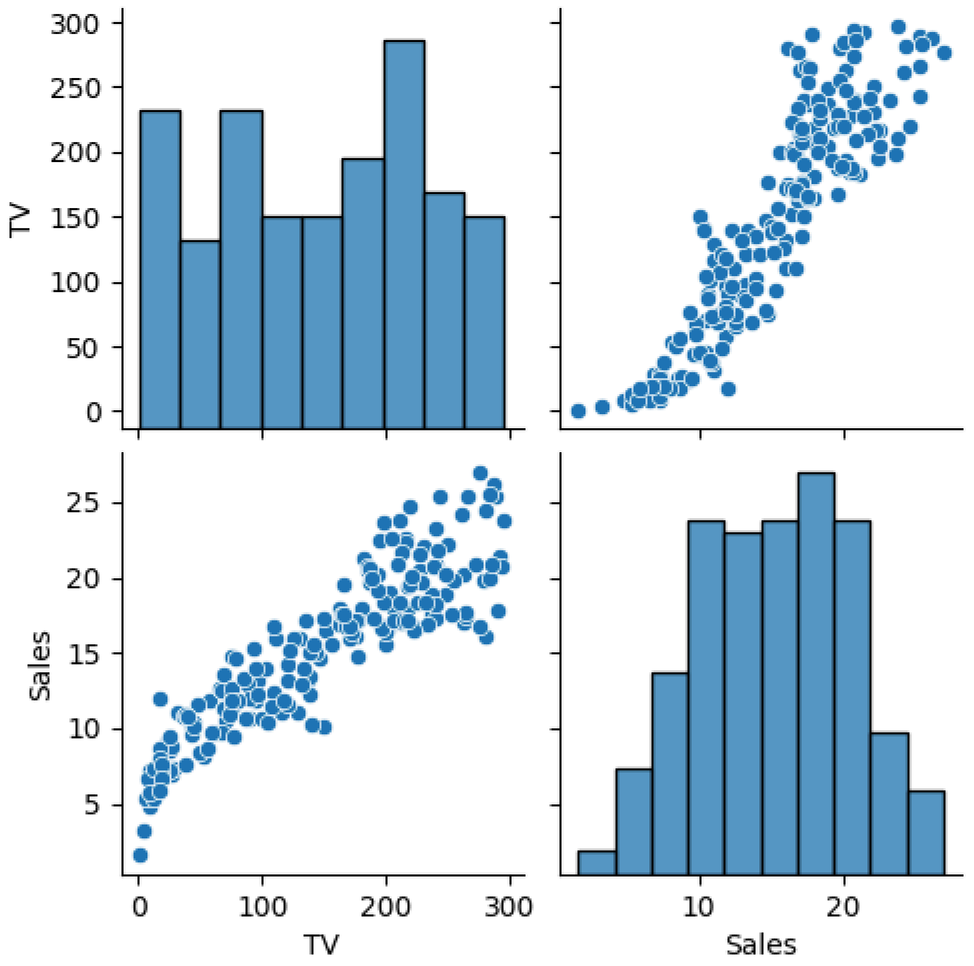
```
array([[<Axes: title={'center': 'TV'}>,
        <Axes: title={'center': 'Sales'}>]], dtype=object)
```



```
#Plot a pair plot to see the relationship between the variables
sn.pairplot(df)

```

```
<seaborn.axisgrid.PairGrid at 0x7bc7b9d62f90>
```

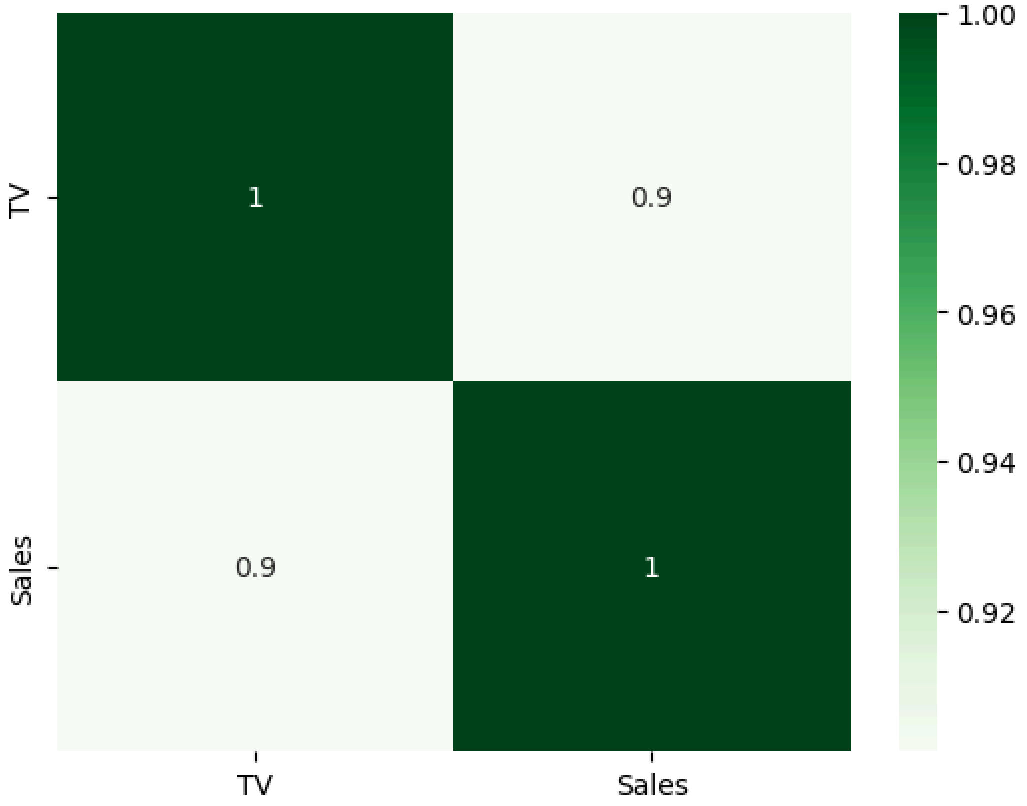


```
#Check correlation between TV and Sales
corr_matrix = df.corr()

#Heatmap
sn.heatmap(corr_matrix, annot=True, cmap= 'Greens')

```

```
<Axes: >
```



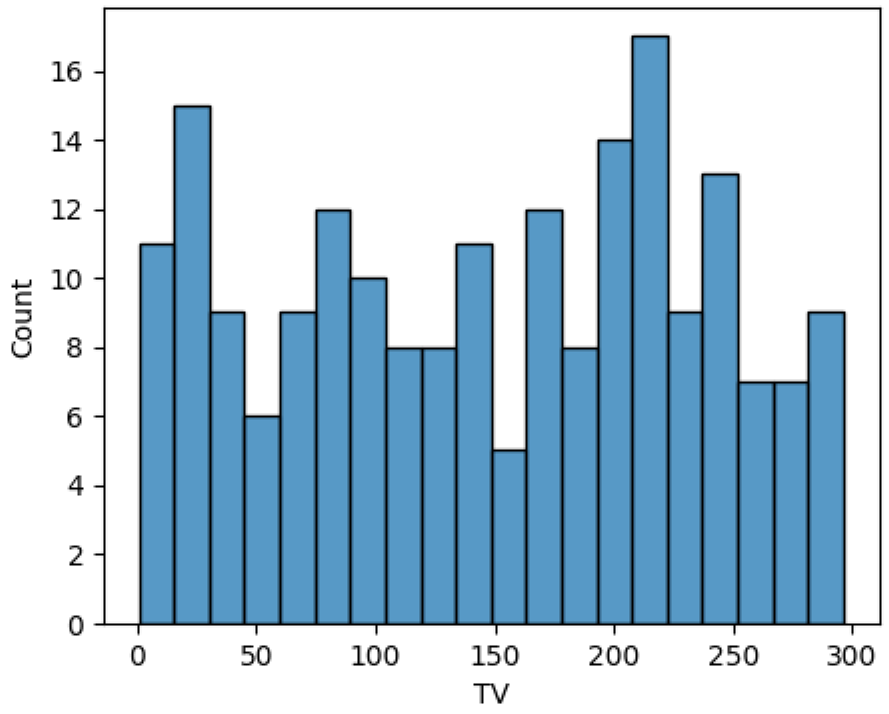
```
#Using the method "kendall" when using "".corr"
df.corr(method= 'kendall')

```

	TV	Sales
TV	1.000000	0.727994
Sales	0.727994	1.000000

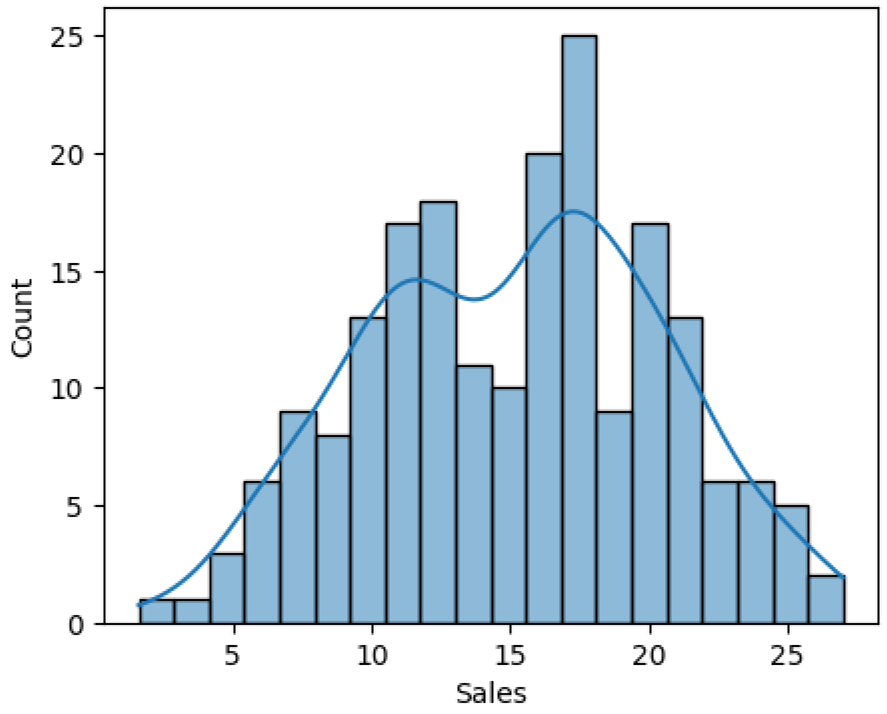
```
#Do an outlier analysis for TV
plt.figure(figsize=(5, 4))
sn.histplot(df['TV'], bins=20)
plt.show()

```



```
#Do an outlier analysis for Sales(the target variable)
plt.figure(figsize=(5, 4))
sn.histplot(df['Sales'], bins=20, kde=True)
plt.show()

```

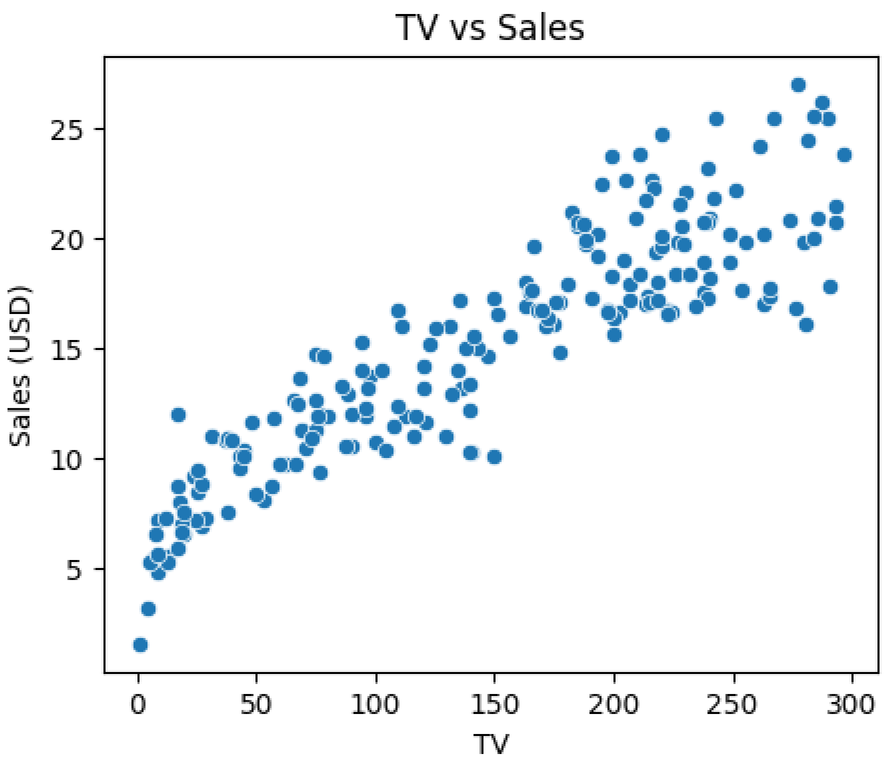


Question: Do you see any considerable outliers?

Answer: Yes there are a few

Plotting Salary vs. YearsExperience

```python
#Plot a scatter plot to see how Sales is related to TV variable
plt.figure(figsize=(5,4))
sn.scatterplot(x='TV', y='Sales', data=df)
plt.title('TV vs Sales')
plt.xlabel('TV')
plt.ylabel('Sales (USD)')
plt.show()
```



Question: What can you see here about the correlation?

More TVs = More Sales

STEP 2: Apply the Machine learning Model:

Here we will apply the ML model:

- Import the necessary libraries
- Build the model
- Display the results

Import the necessary libraries

```
#Import the library for splitting train/test data
from sklearn.model_selection import train_test_split

#Import for using the ML model
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm #Linear regression with statsmodels

```

Buil the model

First we will prepare our dataframes for the x-array and y-array

```
#Create a X and Y data frames
X = df['TV']              #used to make predictions
Y = df['Sales']           #we'll predict this variable
```

Split the dataset into training and test dataset: Training 70% and test 30% sets

```
#Split the data into training and test sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=

```

```
#Print the shape of the training and test data for X and Y
print("The shape of X_train is: ", X_train.shape)
print("The shape of X_test is: ", X_test.shape)
print("The shape of Y_train is: ", Y_train.shape)
print("The shape of Y_test is: ", Y_test.shape)
```

```
The shape of X_train is:  (140,)
The shape of X_test is:  (60,)
The shape of Y_train is:  (140,)
The shape of Y_test is:  (60,)
```

```
#The model requires a 2-D array
#Make the changes necessary for this
x_train = X_train.values.reshape(-1, 1)
x_test = X_test.values.reshape(-1, 1)

print("x_train shape is: ", x_train.shape)
print("x_test shape is: ", x_test.shape)
```

```
x_train shape is:  (140, 1)
x_test shape is:  (60, 1)
```

We build our linear regression model. We already import the necessary library.

```
#Set Linear Regression Model
model = LinearRegression()

```

Train the model with the training data. We use `.fit()` for that.

```
#Fit the training data
response = model.fit(x_train, Y_train)

```

Display Results

```
#Get the intercept and coefficient of the model
intercept = response.intercept_

coeff = response.coef_

```

```
#Print the intercept and coefficient of the model
print("The intercept is: ", intercept)
print("The coefficient is: ", coeff)
```

```
The intercept is:  6.928475121355413
The coefficient is:  [0.056007]
```

```
#Print the standard form using the intercept and coeeficient
print("Then, we have: y= %d + %d * x"  %(intercept, coeff))

```

```
Then, we have: y= 6 + 0 * x
/tmp/ipython-input-3284058802.py:2: DeprecationWarning: Conversion of an
  print("Then, we have: y= %d + %d * x"  %(intercept, coeff))
```