Importing the required libraries needed for the development of the model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Loading the dataset

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Amazon Sales data.csv")
```

Displaying the first 10 rows of the dataset

```
df.head(10)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 100,\n  \"fields\": [\
n    {\n      \"column\": \"Region\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"Australia and Oceania\",\n
\"Central America and the Caribbean\",\n          \"Middle East and
North Africa\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Country\",\n      \"properties\": {\n        \"dtype\": \"string\",\
n        \"num_unique_values\": 76,\n        \"samples\": [\n
\"Rwanda\",\n          \"Brunei\",\n          \"Kyrgyzstan\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Item Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 12,\n
\"samples\": [\n          \"Meat\",\n          \"Beverages\",\n
\"Baby Food\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sales Channel\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Online\",\n          \"Offline\"\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Order Priority\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"C\",\n
\"M\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Order Date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 100,\n        \"samples\":
[\n          \"1/4/2011\",\n          \"11/26/2011\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Order ID\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 260615257,\n
\"min\": 114606559,\n        \"max\": 994022214,\n
\"num_unique_values\": 100,\n        \"samples\": [\n
122583663,\n          441888415\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ship Date\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 99,\n        \"samples\": [\n
\"11/15/2011\",\n          \"3/28/2017\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Units Sold\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2794,\n        \"min\": 124,\n        \"max\": 9925,\n
\"num_unique_values\": 99,\n        \"samples\": [\n          5518,\n
3015\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Unit Price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 235.59224058433134,\n        \"min\":
9.33,\n        \"max\": 668.27,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          421.89,\n          47.45\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Unit Cost\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
188.2081812485549,\n        \"min\": 6.92,\n        \"max\": 524.96,\n
\"num_unique_values\": 12,\n        \"samples\": [\n          364.69,\
n          31.79\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Revenue\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1460028.7068235008,\n        \"min\":
4870.26,\n        \"max\": 5997054.98,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          623289.3,\n
2251232.97\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Cost\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1083938.2521883622,\n        \"min\":
3612.24,\n        \"max\": 4509793.96,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          398042.4,\n
1814786.72\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Total Profit\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 438537.90705963754,\n   \"min\": 1258.02,\n   \"max\": 1719922.04,\n   \"num_unique_values\": 100,\n   \"samples\": [\n   225246.9,\n   436446.25\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

Info of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Region          100 non-null    object 
 1   Country         100 non-null    object 
 2   Item Type       100 non-null    object 
 3   Sales Channel   100 non-null    object 
 4   Order Priority  100 non-null    object 
 5   Order Date      100 non-null    object 
 6   Order ID        100 non-null    int64  
 7   Ship Date       100 non-null    object 
 8   Units Sold      100 non-null    int64  
 9   Unit Price      100 non-null    float64
 10  Unit Cost       100 non-null    float64
 11  Total Revenue   100 non-null    float64
 12  Total Cost      100 non-null    float64
 13  Total Profit    100 non-null    float64
dtypes: float64(5), int64(2), object(7)
memory usage: 11.1+ KB
```

Description of the dataset

```
df.describe()
```

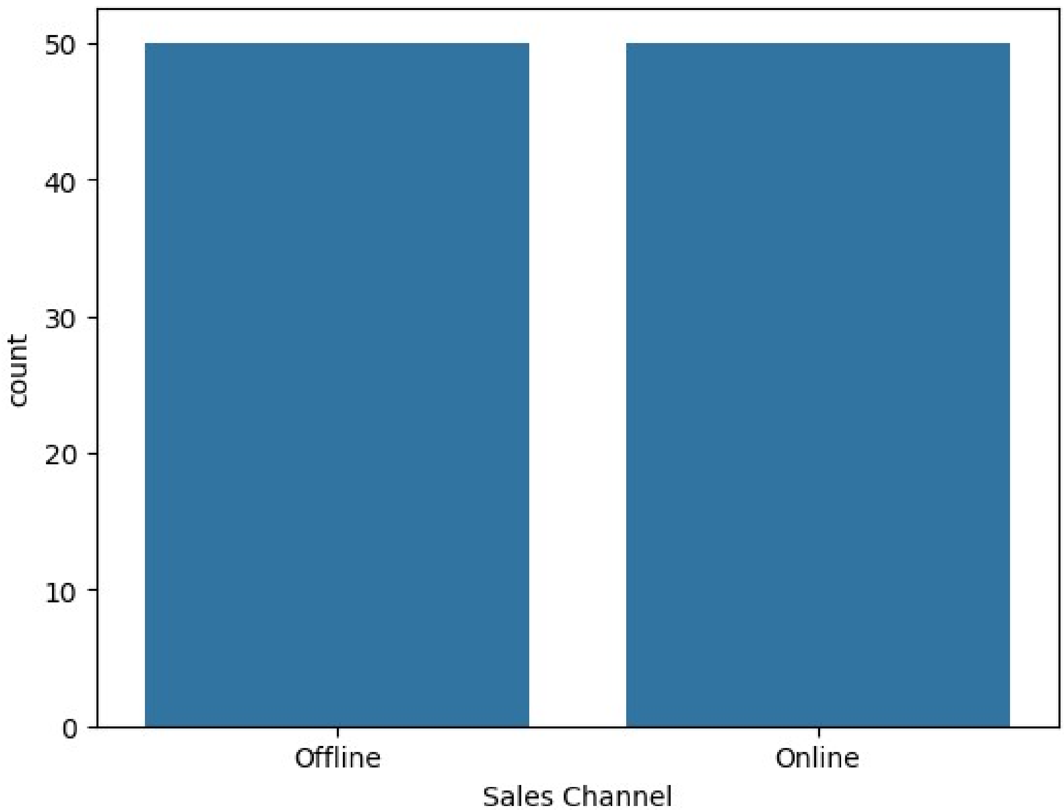
{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"Order ID\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 337613708.4653814,\n   \"min\": 100.0,\n   \"max\": 994022214.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   555020412.36,\n   557708561.0,\n   100.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\

```
n    },\n    {\n      \"column\": \"Units Sold\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3429.652399194458,\n        \"min\": 100.0,\n        \"max\": 9925.0,\
n        \"num_unique_values\": 8,\n        \"samples\": [\n
5128.71,\n          5382.5,\n          100.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Unit Price\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
215.3254206864394,\n        \"min\": 9.33,\n        \"max\": 668.27,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
276.7613,\n          179.88,\n          100.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Unit Cost\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
164.09736449486886,\n        \"min\": 6.92,\n        \"max\": 524.96,\
n        \"num_unique_values\": 8,\n        \"samples\": [\n
191.048,\n          107.275,\n          100.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Total Revenue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1975120.9652460269,\n        \"min\": 100.0,\n        \"max\":
5997054.98,\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          1373487.6831,\n          752314.36,\n          100.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Total Cost\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1499454.9365158535,\n        \"min\": 100.0,\n        \"max\":
4509793.96,\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          931805.6991000001,\n          363566.385,\n
100.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Profit\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 558596.6654892619,\n        \"min\":
100.0,\n        \"max\": 1719922.04,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          441681.98399999994,\n
290767.995,\n          100.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


Countplot of the Sales Channel

```
sns.countplot(x = 'Sales Channel', data = df)
```

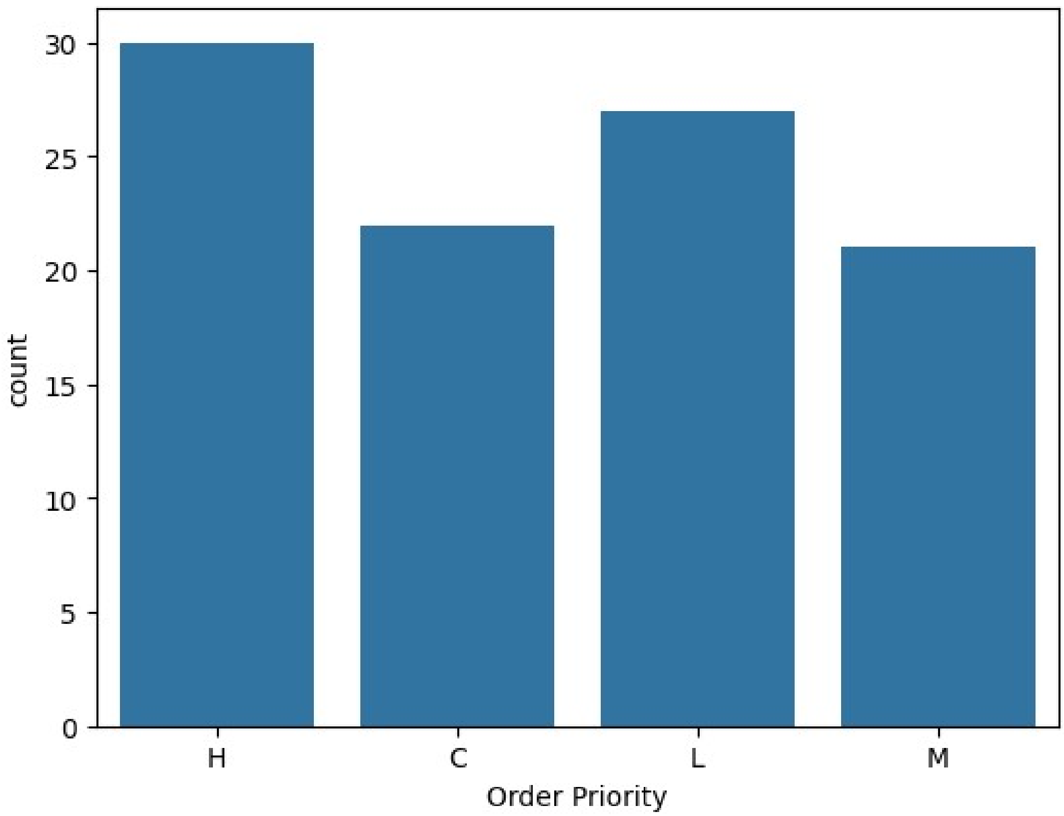
```
<Axes: xlabel='Sales Channel', ylabel='count'>
```



Total count of Order Priority

```
sns.countplot(x = 'Order Priority', data = df)
```

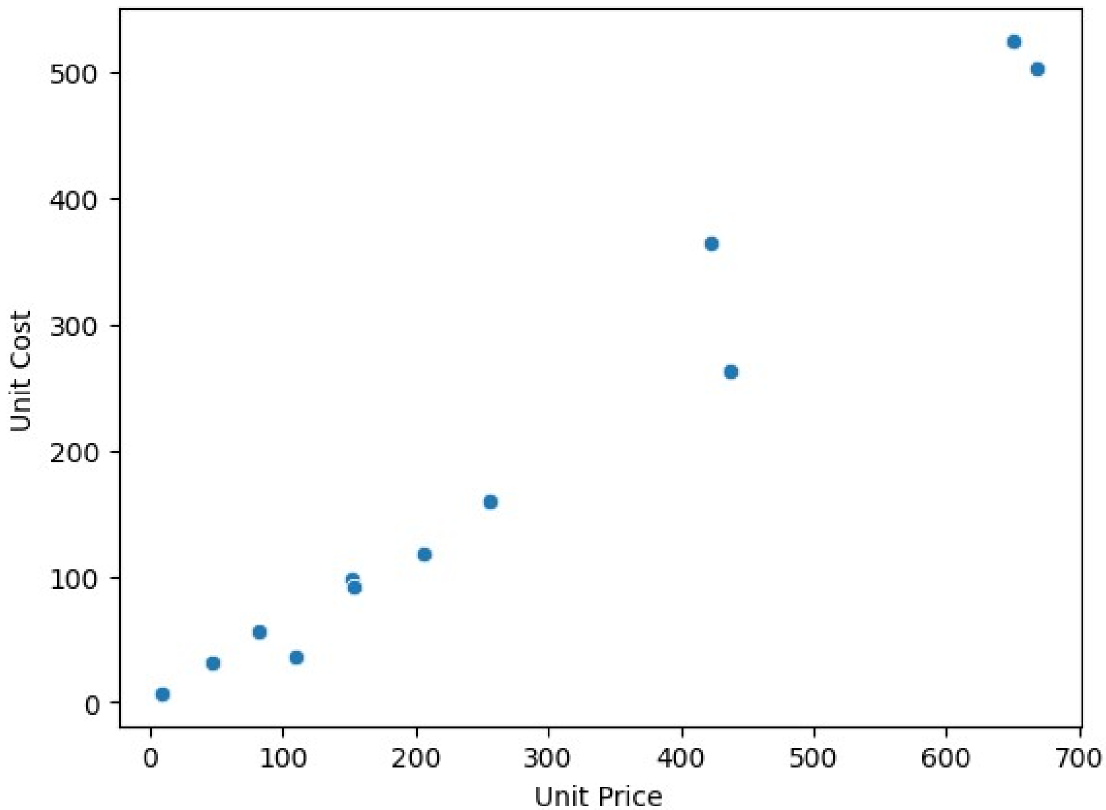
```
<Axes: xlabel='Order Priority', ylabel='count'>
```



Scatterplot of Unit Price and Unit Cost

```
sns.scatterplot(x = 'Unit Price', y = 'Unit Cost', data = df)
```

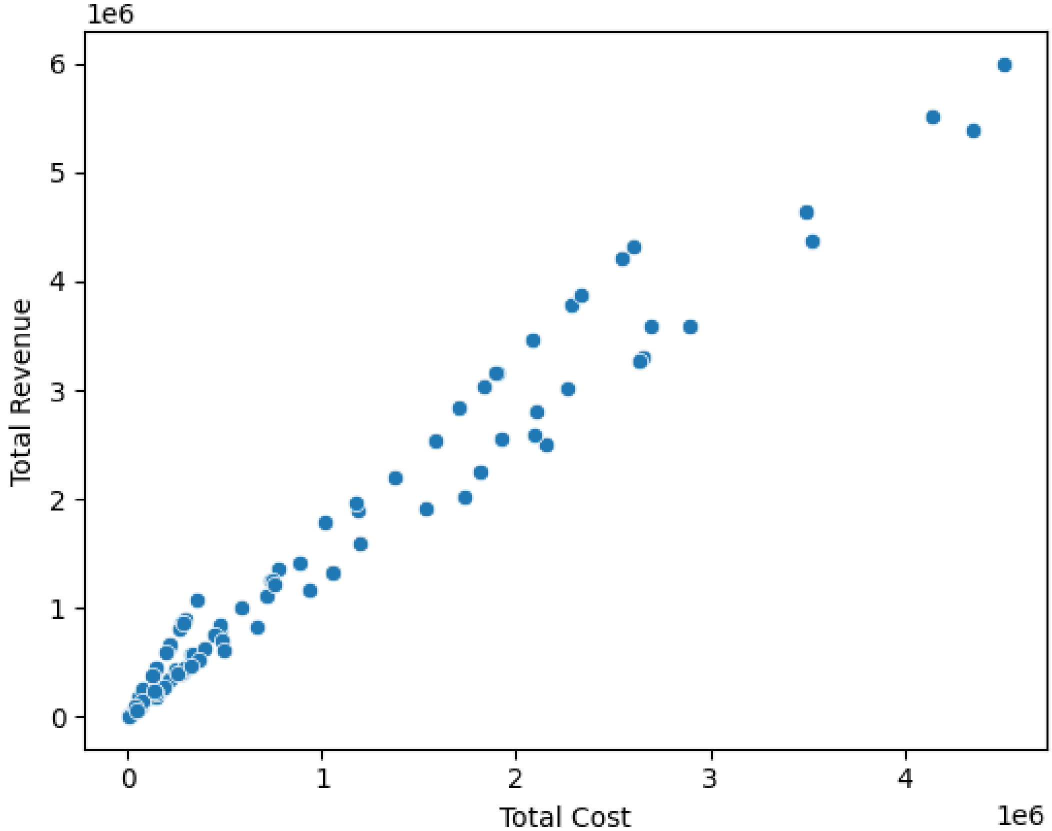
```
<Axes: xlabel='Unit Price', ylabel='Unit Cost'>
```



Scatterplot of total cost and total revenue

```
sns.scatterplot(x = 'Total Cost', y = 'Total Revenue', data = df)
```

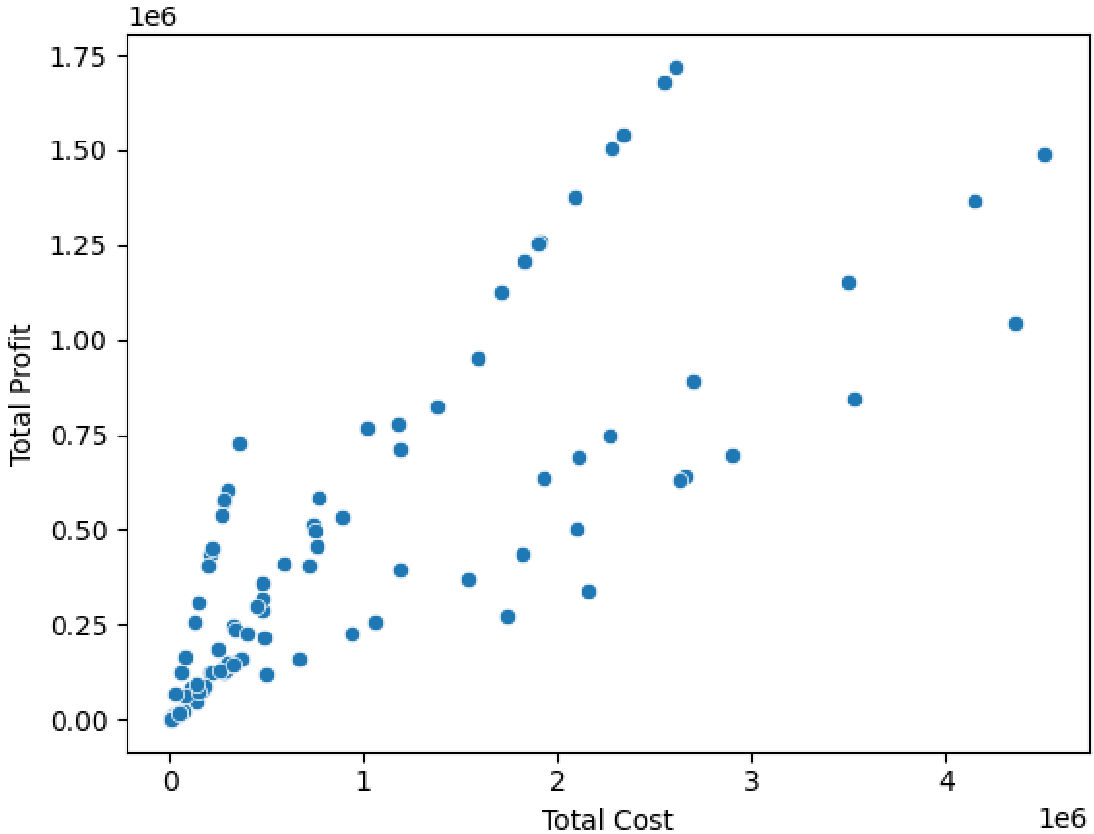
```
<Axes: xlabel='Total Cost', ylabel='Total Revenue'>
```



Scatterplot of total cost and total cost

```
sns.scatterplot(x = 'Total Cost', y = 'Total Profit', data = df)
```

```
<Axes: xlabel='Total Cost', ylabel='Total Profit'>
```



Total revenue generated

```
print("Total revenue generated: ", df['Total Revenue'].sum())
```

```
Total revenue generated:  137348768.31
```

Total Cost

```
print("Total Cost: ", df['Total Cost'].sum())
```

```
Total Cost:  93180569.91000001
```

Total Profit generated

```
print("Total profit generated: ", df['Total Profit'].sum())
```

```
Total profit generated:  44168198.39999999
```

Maximum profit

```
print("Max profit:", df['Total Profit'].max())
```

```
Max profit: 1719922.04
```

Minimum Profit

```
print("Minimum profit:", df['Total Profit'].min())
```

```
Minimum profit: 1258.02
```

Profit Margin

```
df['Profit Margin'] = (df['Total Profit']/df['Total Revenue'])*100
```

Profit margin added to the dataset

```
df.head(10)
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 100,\n \"fields\": [\n   {\n   \"column\": \"Region\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 7,\n   \"samples\": [\n   \"Australia and Oceania\",\n   \"Central America and the Caribbean\",\n   \"Middle East and North Africa\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Country\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 76,\n   \"samples\": [\n   \"Rwanda\",\n   \"Brunei\",\n   \"Kyrgyzstan\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Item Type\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 12,\n   \"samples\": [\n   \"Meat\",\n   \"Beverages\",\n   \"Baby Food\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sales Channel\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Online\",\n   \"Offline\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Order Priority\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"C\",\n   \"M\"\n   ],\n   \"semantic_type\": \"\",\n

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Order Date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 100,\n        \"samples\":
[\n          \"1/4/2011\",\n          \"11/26/2011\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Order ID\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 260615257,\n
\"min\": 114606559,\n        \"max\": 994022214,\n
\"num_unique_values\": 100,\n        \"samples\": [\n
122583663,\n          441888415\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ship Date\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 99,\n        \"samples\": [\n
\"11/15/2011\",\n          \"3/28/2017\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Units Sold\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2794,\n        \"min\": 124,\n        \"max\": 9925,\n
\"num_unique_values\": 99,\n        \"samples\": [\n          5518,\n
3015\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Unit Price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 235.59224058433134,\n        \"min\":
9.33,\n        \"max\": 668.27,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          421.89,\n          47.45\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Unit Cost\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
188.2081812485549,\n        \"min\": 6.92,\n        \"max\": 524.96,\n
\"num_unique_values\": 12,\n        \"samples\": [\n          364.69,\
n          31.79\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Revenue\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1460028.7068235008,\n        \"min\":
4870.26,\n        \"max\": 5997054.98,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          623289.3,\n
2251232.97\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Cost\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1083938.2521883622,\n        \"min\":
3612.24,\n        \"max\": 4509793.96,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          398042.4,\n
1814786.72\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Profit\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 438537.90705963754,\n        \"min\":
1258.02,\n        \"max\": 1719922.04,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          225246.9,\n
```

```
436446.25\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Profit Margin\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.281476858612251,\n        \"min\":
13.558036455000117,\n        \"max\": 67.20351390922403,\n
\"num_unique_values\": 27,\n        \"samples\": [\n
40.97754121770739,\n          13.558036455000117\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


Averge Revenue per unit

```
df['Average Revenue per Unit'] = df['Total Revenue']/df['Units Sold']
```

Averge Revenue per unit column added to the dataset

```
df.head(10)
```

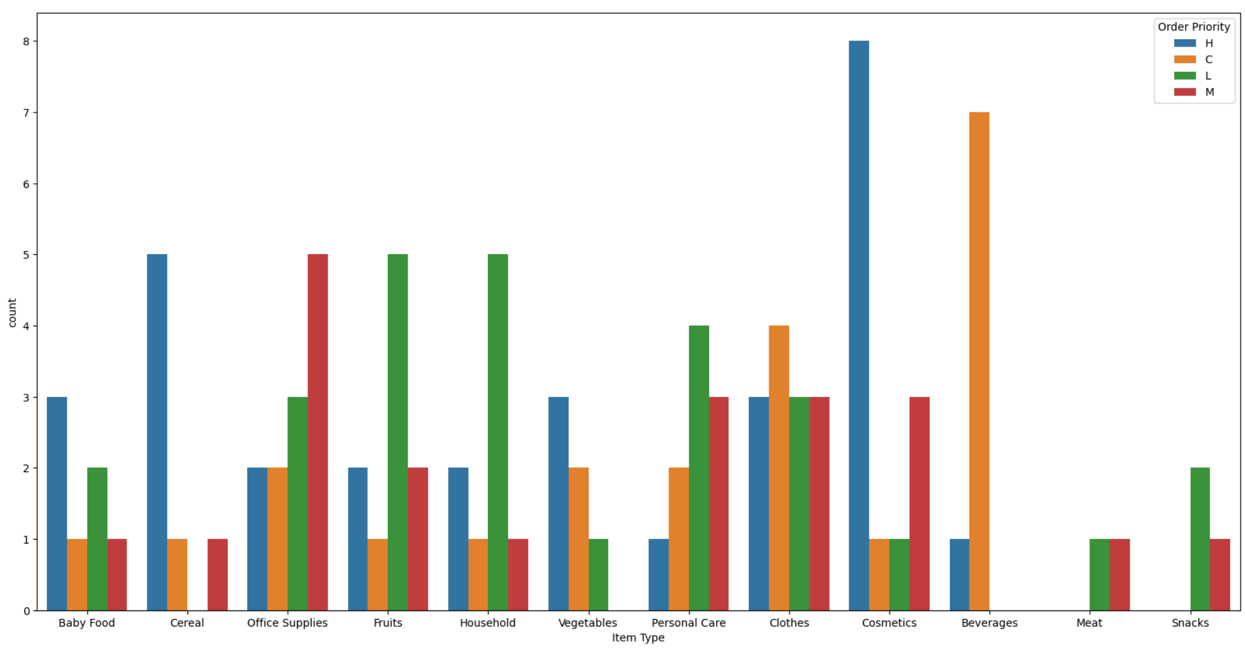

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 100,\n  \"fields\": [\
n    {\n      \"column\": \"Region\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"Australia and Oceania\",\n
\"Central America and the Caribbean\",\n          \"Middle East and
North Africa\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Country\",\n      \"properties\": {\n        \"dtype\": \"string\",\
n        \"num_unique_values\": 76,\n        \"samples\": [\n
\"Rwanda\",\n          \"Brunei\",\n          \"Kyrgyzstan\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Item Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 12,\n
\"samples\": [\n          \"Meat\",\n          \"Beverages\",\n
\"Baby Food\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sales Channel\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Online\",\n          \"Offline\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Order Priority\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"C\",\n
\"M\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Order Date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 100,\n        \"samples\":
[\n          \"1/4/2011\",\n          \"11/26/2011\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Order ID\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 260615257,\n
\"min\": 114606559,\n        \"max\": 994022214,\n
\"num_unique_values\": 100,\n        \"samples\": [\n
122583663,\n          441888415\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ship Date\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 99,\n        \"samples\": [\n
\"11/15/2011\",\n          \"3/28/2017\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Units Sold\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2794,\n        \"min\": 124,\n        \"max\": 9925,\n
\"num_unique_values\": 99,\n        \"samples\": [\n          5518,\n
3015\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Unit Price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 235.59224058433134,\n        \"min\":
9.33,\n        \"max\": 668.27,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          421.89,\n          47.45\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Unit Cost\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
188.2081812485549,\n        \"min\": 6.92,\n        \"max\": 524.96,\n
\"num_unique_values\": 12,\n        \"samples\": [\n          364.69,\
n          31.79\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Revenue\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1460028.7068235008,\n        \"min\":
4870.26,\n        \"max\": 5997054.98,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          623289.3,\n
2251232.97\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Cost\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1083938.2521883622,\n        \"min\":
3612.24,\n        \"max\": 4509793.96,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          398042.4,\n
1814786.72\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Total Profit\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 438537.90705963754,\n        \"min\":
1258.02,\n        \"max\": 1719922.04,\n        \"num_unique_values\":
100,\n        \"samples\": [\n          225246.9,\n
436446.25\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Profit Margin\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.281476858612251,\n        \"min\":
13.558036455000117,\n        \"max\": 67.20351390922403,\n
\"num_unique_values\": 27,\n        \"samples\": [\n
40.97754121770739,\n          13.558036455000117\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Average Revenue per Unit\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
235.59224058433134,\n        \"min\": 9.33,\n        \"max\":
668.2700000000001,\n        \"num_unique_values\": 21,\n
\"samples\": [\n          255.28,\n          421.89\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


Countplot of Item type with respect to Order Priority

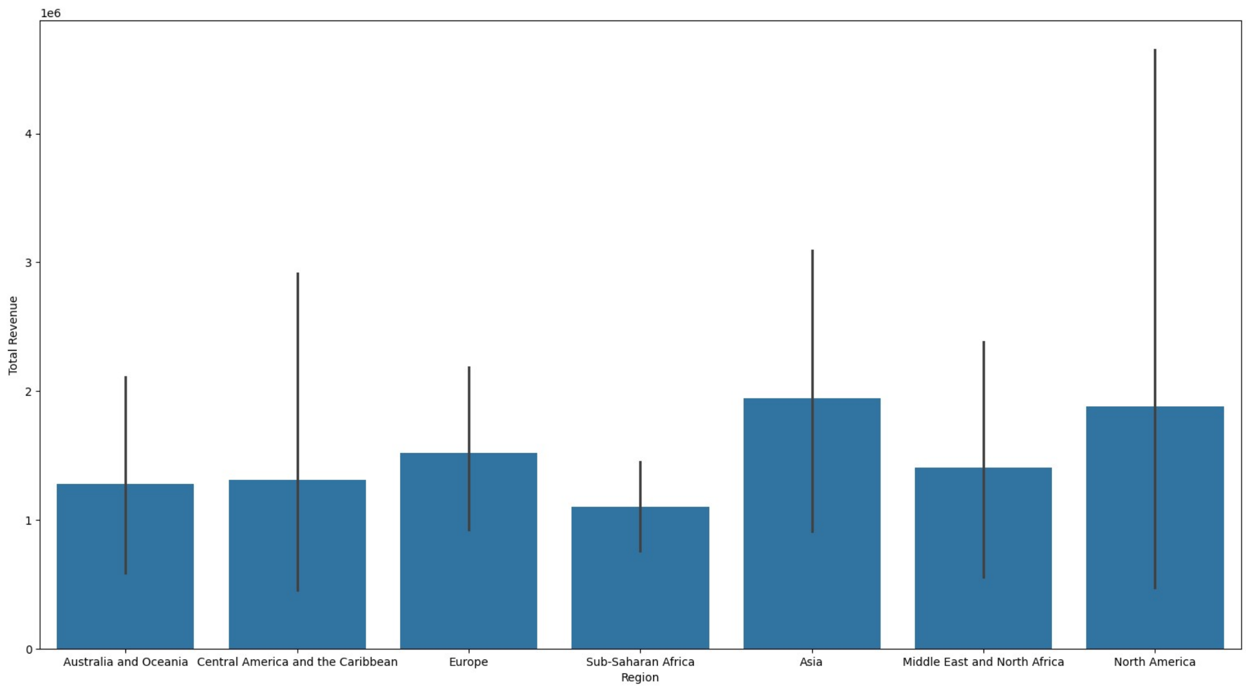
```
plt.figure(figsize = (20,10))
sns.countplot(x = 'Item Type', data = df, hue = 'Order Priority')

<Axes: xlabel='Item Type', ylabel='count'>
```



Barplot between Region and Total Revenue

```
plt.figure(figsize = (19,10))
sns.barplot(x = 'Region', y = 'Total Revenue', data = df)
```

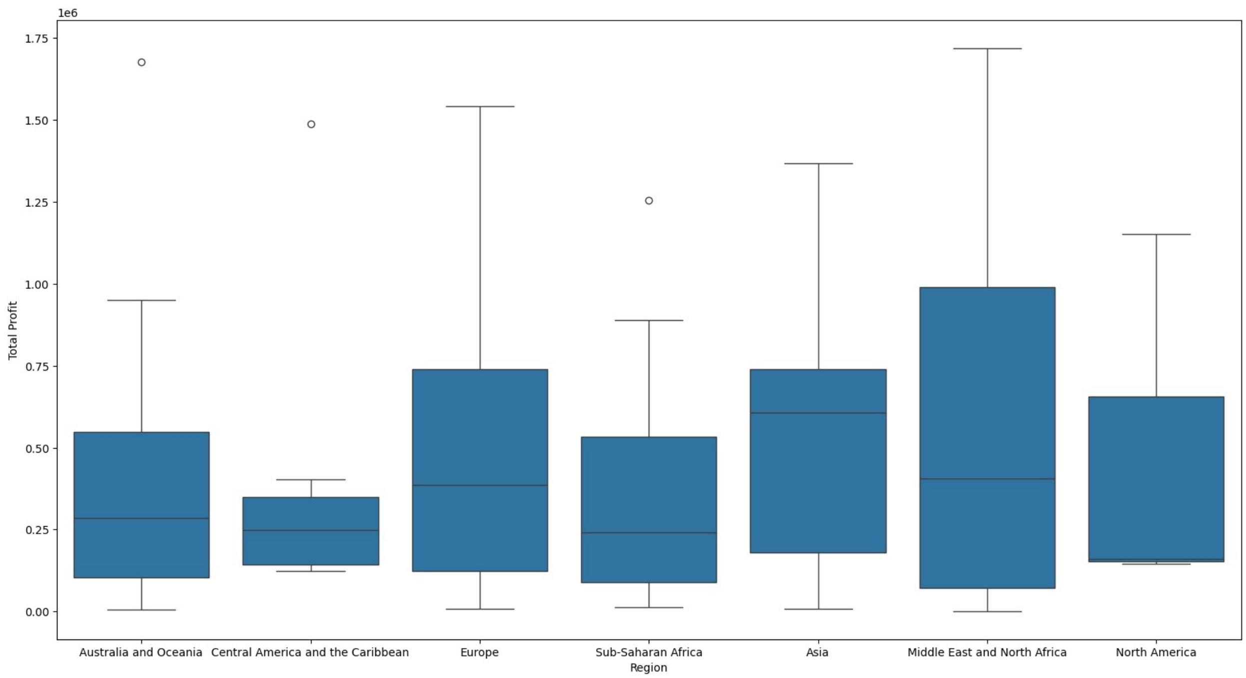
```
<Axes: xlabel='Region', ylabel='Total Revenue'>
```



Barplot between Region and Total Profit

```
plt.figure(figsize = (19,10))
sns.boxplot(x = 'Region', y = 'Total Profit', data = df)
```

```
<Axes: xlabel='Region', ylabel='Total Profit'>
```



Value count in the sales channel

```
df['Sales Channel'].value_counts()

Sales Channel
Offline    50
Online     50
Name: count, dtype: int64
```

Value count of the region

```
df['Region'].value_counts()

Region
Sub-Saharan Africa                   36
Europe                               22
Australia and Oceania                11
Asia                                 11
Middle East and North Africa         10
Central America and the Caribbean     7
North America                         3
Name: count, dtype: int64
```

Value count in Country

```
df['Country'].value_counts()
```

```
Country
The Gambia                4
Sierra Leone              3
Sao Tome and Principe     3
Mexico                    3
Australia                 3
                         ..
Comoros                   1
Iceland                   1
Macedonia                 1
Mauritania                1
Mozambique                1
Name: count, Length: 76, dtype: int64
```

```
df.columns
```

```
Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order
Priority',
       'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit
Price',
       'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit',
       'Profit Margin', 'Average Revenue per Unit'],
      dtype='object')
```

Calculating the dummy values of the columns with String dataset

```
new_r = pd.get_dummies(df['Region'], dtype = int)
region = pd.DataFrame(new_r)
new_c = pd.get_dummies(df['Country'], dtype = int)
country = pd.DataFrame(new_c)

region = region.replace({'True':1, 'False':0})

region
```

{"summary":"{\n \"name\": \"region\",\n \"rows\": 100,\n
\"fields\": [\n   {\n   \"column\": \"Asia\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Australia and Oceania\",\n   \"properties\": {\n

\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   0,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Central America and the Caribbean\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Europe\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Middle East and North Africa\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   1,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"North America\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   1,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Sub-Saharan Africa\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   1,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"region"}

country

{"type":"dataframe","variable_name":"country"}

Finding the dummy values for the sales channel

```
new_Channel = pd.get_dummies(df['Sales Channel'], dtype = int)
Channel = pd.DataFrame(new_Channel)

Channel.head(5)
```


{"summary":"{\n \"name\": \"Channel\",\n \"rows\": 100,\n
\"fields\": [\n   {\n   \"column\": \"Offline\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":

```
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Online\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"Channel"}
```


Finding the dummy values for the Item Type

```
new_item = pd.get_dummies(df['Item Type'], dtype = int)
Item = pd.DataFrame(new_item)

Item
```


```
{"summary":"{\n  \"name\": \"Item\",\n  \"rows\": 100,\n  \"fields\":
[\n    {\n      \"column\": \"Baby Food\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Beverages\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Cereal\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Clothes\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Cosmetics\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Fruits\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
```

```
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Household\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Meat\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Office Supplies\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Personal Care\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Snacks\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Vegetables\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"Item"}
```


Finding the dummy values for the Order Priority

```
new_Priority = pd.get_dummies(df['Order Priority'], dtype = int)
Priority = pd.DataFrame(new_Priority)

Priority
```



```
{"summary":"{\n  \"name\": \"Priority\",\n  \"rows\": 100,\n
\"fields\": [\n    {\n      \"column\": \"C\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

n   },\n   {\n   \"column\": \"H\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"L\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"M\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"Priority"}

```
new_Date = pd.get_dummies(df['Ship Date'], dtype = int)
Date = pd.DataFrame(new_Date)
new_Date1 = pd.get_dummies(df['Order Date'], dtype = int)
Date1 = pd.DataFrame(new_Date1)

Date
```

{"type":"dataframe","variable_name":"Date"}

```
Date1
```

{"type":"dataframe","variable_name":"Date1"}

Concatenating the new calculated values with the original dataframe

```
new_df = pd.concat([region, country, Channel, Item, Priority, Date1, Date, df], axis = 1)

new_df
```

{"type":"dataframe","variable_name":"new_df"}

Dropping columns from the dataframe

```
new_df = new_df.drop({'Region', 'Country', 'Sales Channel', 'Order Date', 'Item Type', 'Ship Date', 'Order Priority'}, axis = 1)

new_df.head(5)
```

{"type":"dataframe","variable_name":"new_df"}

Selecting the predictor the target variable

```
X = new_df.drop('Total Profit', axis = 1)
y = new_df[['Total Profit']]
```

Printing the shape of the predictor and target variable

```
print(X.shape)
print(y.shape)
```

```
(100, 308)
(100, 1)
```

Scaling the dataset

```
Scaler = MinMaxScaler()
X_scaled = Scaler.fit_transform(X)
y_scaled = Scaler.fit_transform(y)
```

Splitting the dataset into train and test

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y_scaled, test_size = 0.2)
```

Printing the shape of the splitted dataset

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(80, 308)
(20, 308)
(80, 1)
(20, 1)
```

Summary of the model

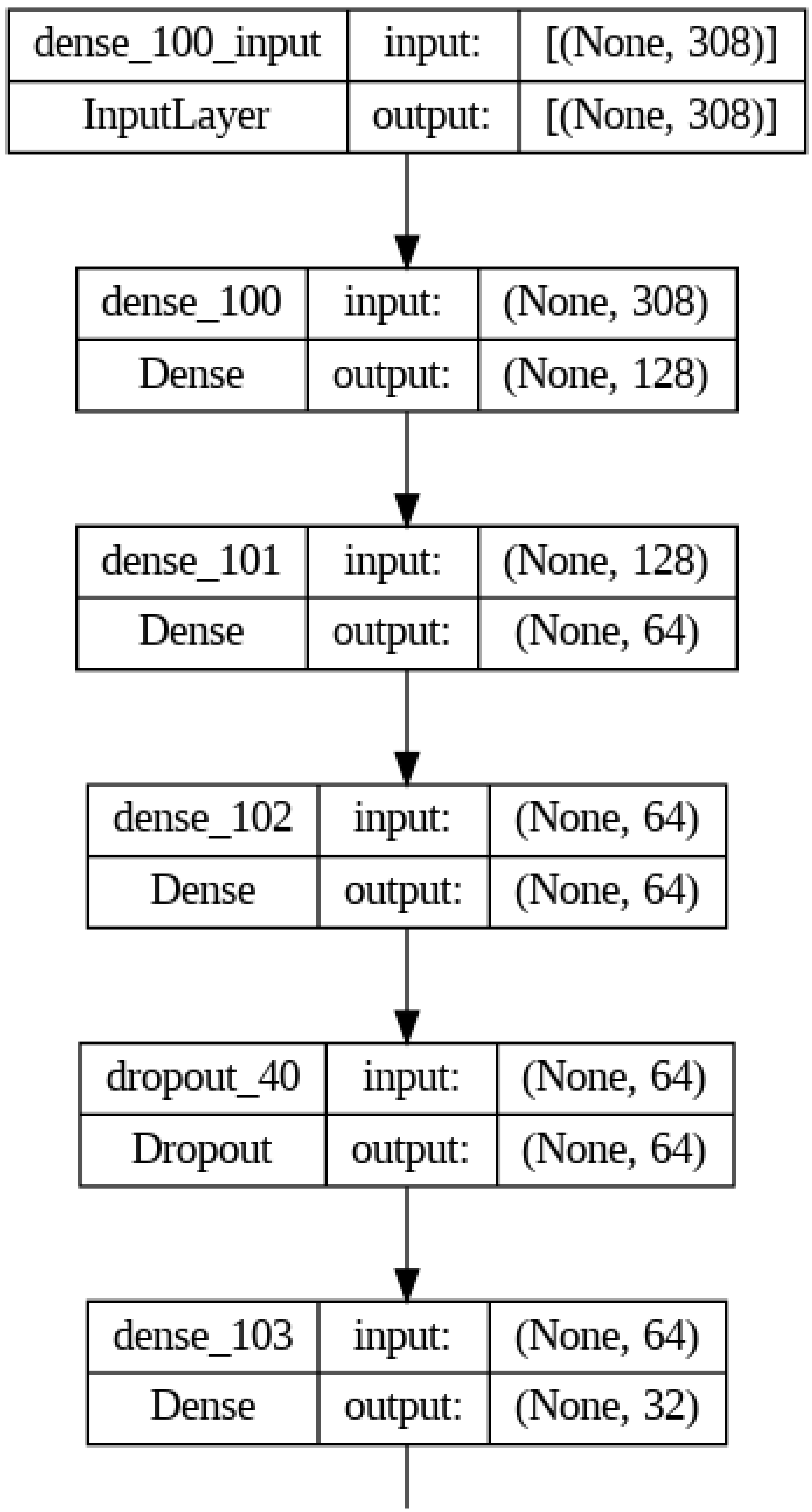
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units = 128, activation = 'linear',
input_shape = (308,)))
model.add(tf.keras.layers.Dense(units = 64, activation = 'relu'))
model.add(tf.keras.layers.Dense(units = 64, activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(units = 32, activation = 'relu'))
model.add(tf.keras.layers.Dense(units = 32, activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(units = 16, activation = 'relu'))
model.add(tf.keras.layers.Dense(units = 16, activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(units = 8, activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(units = 4, activation = 'relu'))
model.add(tf.keras.layers.Dense(units = 1, activation  = 'linear'))
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_100 (Dense)           (None, 128)               39552

 dense_101 (Dense)           (None, 64)                8256

 dense_102 (Dense)           (None, 64)                4160

 dropout_40 (Dropout)        (None, 64)                0

 dense_103 (Dense)           (None, 32)                2080

 dense_104 (Dense)           (None, 32)                1056

 dropout_41 (Dropout)        (None, 32)                0

 dense_105 (Dense)           (None, 16)                528

 dense_106 (Dense)           (None, 16)                272

 dropout_42 (Dropout)        (None, 16)                0

 dense_107 (Dense)           (None, 8)                 136

 dropout_43 (Dropout)        (None, 8)                 0

 dense_108 (Dense)           (None, 4)                 36
```

```
 dense_109 (Dense)           (None, 1)                 5

=================================================================
Total params: 56081 (219.07 KB)
Trainable params: 56081 (219.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Plotting the model

```
keras.utils.plot_model(model, to_file='png', show_shapes=True)
```



dense_100_input
input:
[(None, 308)]
InputLayer
output:
[(None, 308)]
dense_100
input:
(None, 308)
Dense
output:
(None, 128)
dense_101
input:
(None, 128)
Dense
output:
(None, 64)
dense_102
input:
(None, 64)
Dense
output:
(None, 64)
dropout_40
input:
(None, 64)
Dropout
output:
(None, 64)
dense_103
input:
(None, 64)
Dense
output:
(None, 32)

Compiling the model

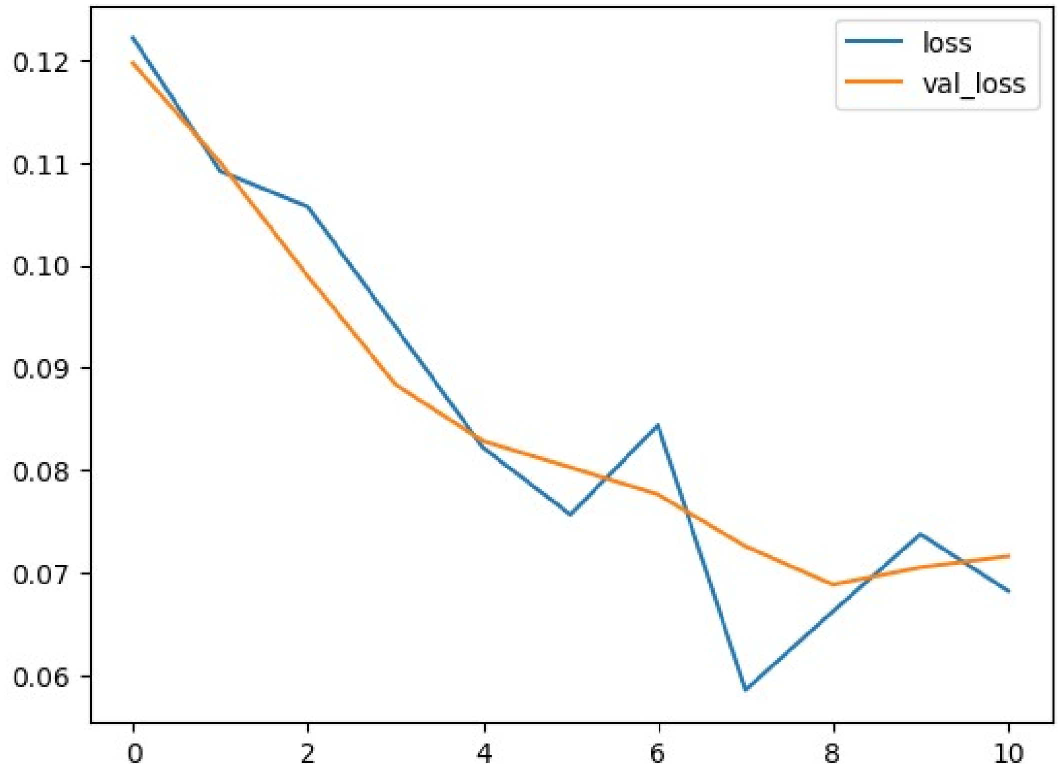
```
model.compile(optimizer = 'Adam', loss = 'mean_squared_error')
from keras.callbacks import EarlyStopping
es = EarlyStopping(patience = 2, monitor = 'val_loss')
model.fit(X_train, y_train, epochs = 25, batch_size = 10,
validation_data = (X_test, y_test), callbacks = [es])
```

```
Epoch 1/25
8/8 [==============================] - 4s 141ms/step - loss: 0.1222 -
val_loss: 0.1197
Epoch 2/25
8/8 [==============================] - 0s 8ms/step - loss: 0.1092 -
val_loss: 0.1100
Epoch 3/25
8/8 [==============================] - 0s 11ms/step - loss: 0.1057 -
val_loss: 0.0989
Epoch 4/25
8/8 [==============================] - 0s 11ms/step - loss: 0.0940 -
val_loss: 0.0884
Epoch 5/25
8/8 [==============================] - 0s 11ms/step - loss: 0.0822 -
val_loss: 0.0828
Epoch 6/25
8/8 [==============================] - 0s 11ms/step - loss: 0.0757 -
val_loss: 0.0803
Epoch 7/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0844 -
val_loss: 0.0776
Epoch 8/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0585 -
val_loss: 0.0726
Epoch 9/25
8/8 [==============================] - 0s 9ms/step - loss: 0.0662 -
val_loss: 0.0688
Epoch 10/25
8/8 [==============================] - 0s 8ms/step - loss: 0.0737 -
val_loss: 0.0705
Epoch 11/25
8/8 [==============================] - 0s 11ms/step - loss: 0.0682 -
val_loss: 0.0716

<keras.src.callbacks.History at 0x7dc69e77c460>
```

Plotting the training history of the model

```
hist = model.history.history
h = pd.DataFrame(hist)
h.plot()
```

```
&lt;Axes: &gt;
```



Prediction by the model

```
  y_predict = model.predict(X_test)
  y_predict
```

```
1/1 [==============================] - 0s 110ms/step
```

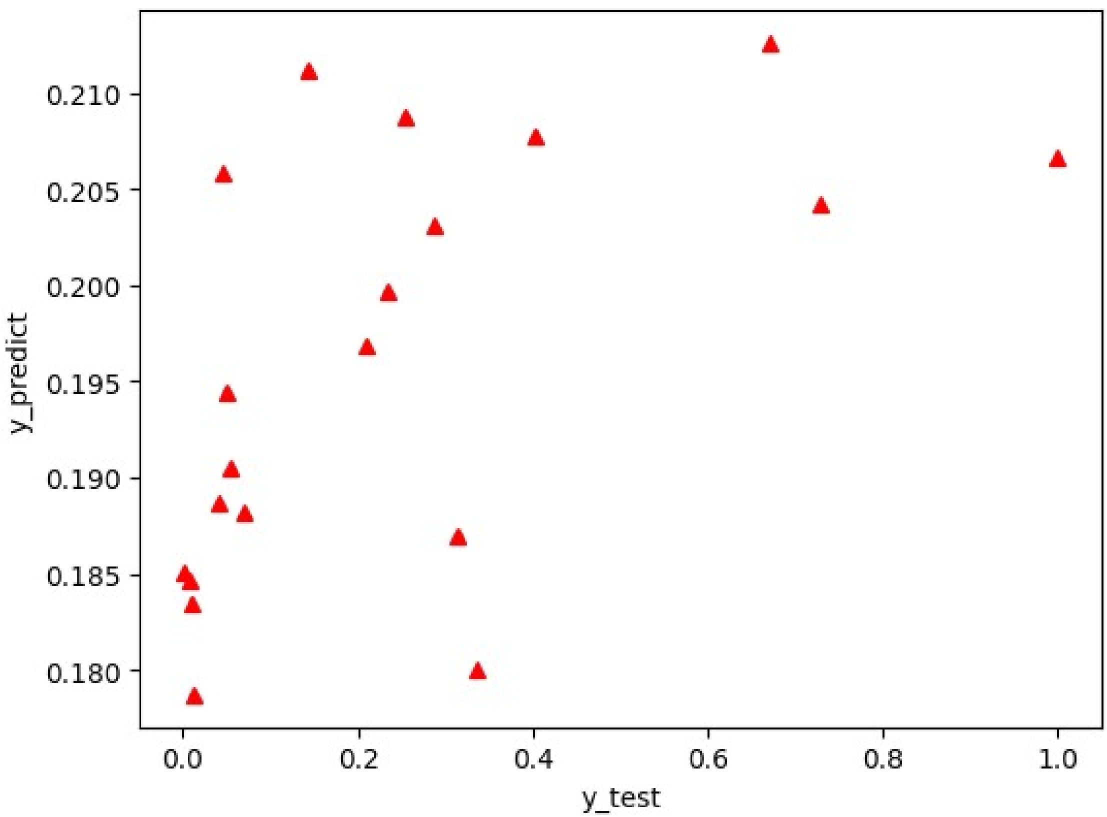
```
array([[0.18816054],
       [0.19966517],
       [0.21119696],
       [0.19055754],
       [0.1869608 ],
       [0.19445625],
       [0.20315206],
       [0.1850196 ],
       [0.20660865],
```

```
        [0.20420228],
        [0.20769827],
        [0.19681492],
        [0.18348233],
        [0.21258318],
        [0.20879586],
        [0.20581448],
        [0.18466815],
        [0.18002333],
        [0.17872919],
        [0.18865734]], dtype=float32)
```

Scatterplot between y_test and y_predict variable

```
plt.plot(y_test,y_predict, '^', color = 'r')
plt.xlabel('y_test')
plt.ylabel('y_predict')
```

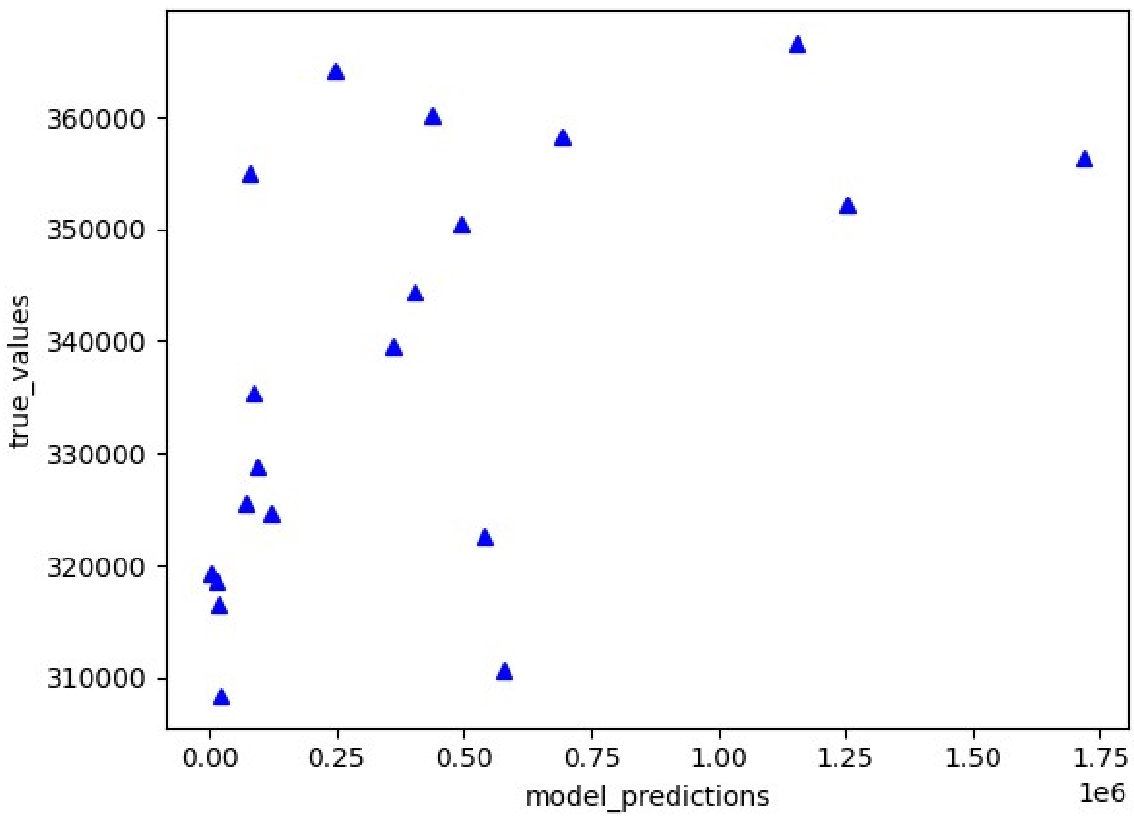
```
Text(0, 0.5, 'y_predict')
```



Scatterplot of true values and model predictions

```
y_predict_original = Scaler.inverse_transform(y_predict)
y_test_original = Scaler.inverse_transform(y_test)
plt.plot(y_test_original,y_predict_original,'^',color = 'b')
plt.xlabel('model_predictions')
plt.ylabel('true_values')
```

```
Text(0, 0.5, 'true_values')
```



Calculation of n

```
k = X_test.shape
k
n = len(X_test)
n
```

```
20
```

Root Mean Squared Error Calculation

```
from sklearn.metrics import
r2_score,mean_squared_error,mean_absolute_error
from math import sqrt
RMSE =
float(format(np.sqrt(mean_squared_error(y_test_original,y_predict_orig
inal)), '0.3f'))
print(RMSE)

459844.656
```

Mean Squared Error calculation

```
MSE = mean_squared_error(y_test_original,y_predict_original)
print(MSE)

211457107519.14764
```

Mean Absolute Error calculation

```
MAE = mean_absolute_error(y_test_original,y_predict_original)
print(MAE)

335498.68456250004
```

Calculation of R2 Score

```
r2 = r2_score(y_test_original,y_predict_original)
print(r2)

0.011547039298282491
```

Saving the model

```
model.save("Predictor.h5")

/usr/local/lib/python3.10/dist-packages/keras/src/engine/
training.py:3103: UserWarning: You are saving your model as an HDF5
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

The score of the ANN model is not upto the mark hence I have included the Linear Regression model also

```
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(X_train, y_train)
model1.score(X_test, y_test)
```

```
0.8984638274200195
```

Prediction of values

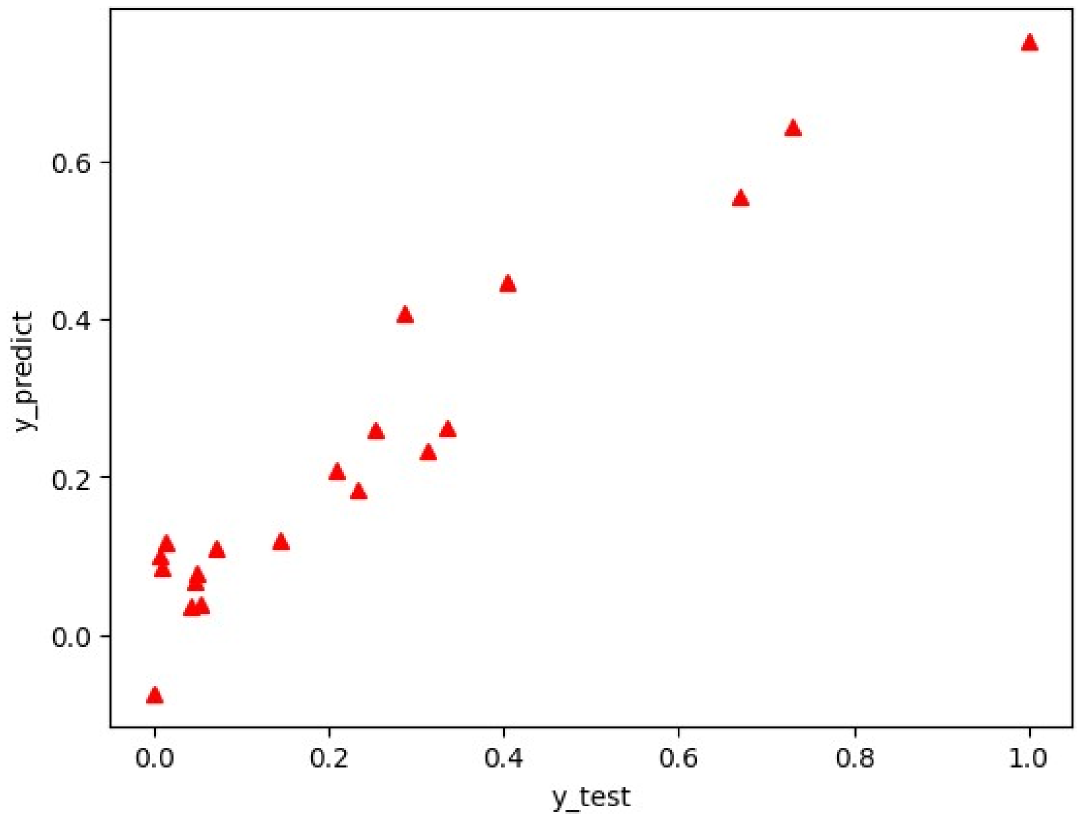
```
y_predict = model1.predict(X_test)
y_predict
```

```
array([[ 0.10921206],
       [ 0.18295459],
       [ 0.11863686],
       [ 0.03824205],
       [ 0.23277901],
       [ 0.07832323],
       [ 0.40672669],
       [-0.0747907 ],
       [ 0.75184372],
       [ 0.64406438],
       [ 0.4465804 ],
       [ 0.2072212 ],
       [ 0.08439128],
       [ 0.55597884],
       [ 0.25980533],
       [ 0.06833848],
       [ 0.09911556],
       [ 0.26105908],
       [ 0.11733028],
       [ 0.0363049 ]])
```

Plotting the predicted values

```
plt.plot(y_test,y_predict, '^', color = 'r')
plt.xlabel('y_test')
plt.ylabel('y_predict')
```

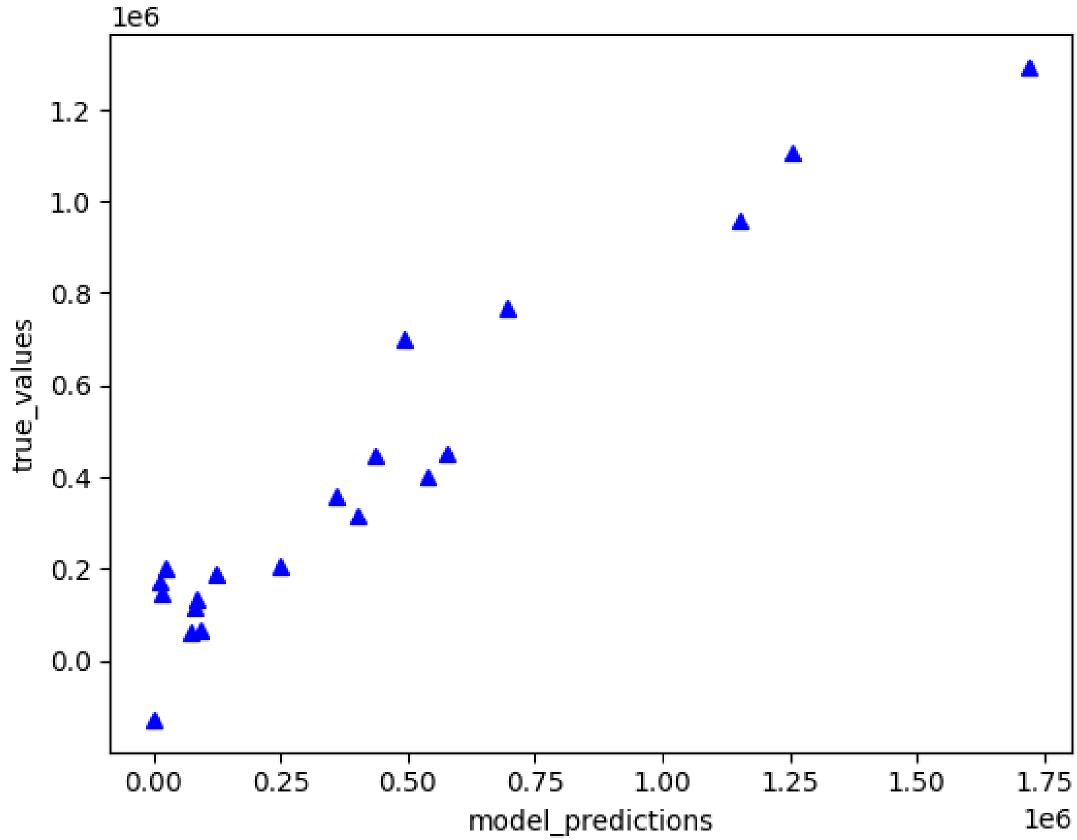
```
Text(0, 0.5, 'y_predict')
```



Plotting the true predictions

```
y_predict_original = Scaler.inverse_transform(y_predict)
y_test_original = Scaler.inverse_transform(y_test)
plt.plot(y_test_original,y_predict_original,'^',color = 'b')
plt.xlabel('model_predictions')
plt.ylabel('true_values')
```

```
Text(0, 0.5, 'true_values')
```



Mean Squared Error calculation

```
MSE = mean_squared_error(y_test_original,y_predict_original)
print(MSE)

21721362792.100307
```

Mean Absolute Error calculation

```
MAE = mean_absolute_error(y_test_original,y_predict_original)
print(MAE)

112361.98772843098
```

Calculation of R2 Score

```
r2 = r2_score(y_test_original,y_predict_original)
print(r2 * 100)
```

```
89.84638274200194
```