```
!pip install mplcyberpunk
```

```
Requirement already satisfied: mplcyberpunk in
/usr/local/lib/python3.10/dist-packages (0.7.1)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (from mplcyberpunk) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (1.4.5)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (1.25.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (24.0)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib-
>mplcyberpunk) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib->mplcyberpunk) (1.16.0)
```

```python
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
import mplcyberpunk
```

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

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"}
```

```
df.info()
```

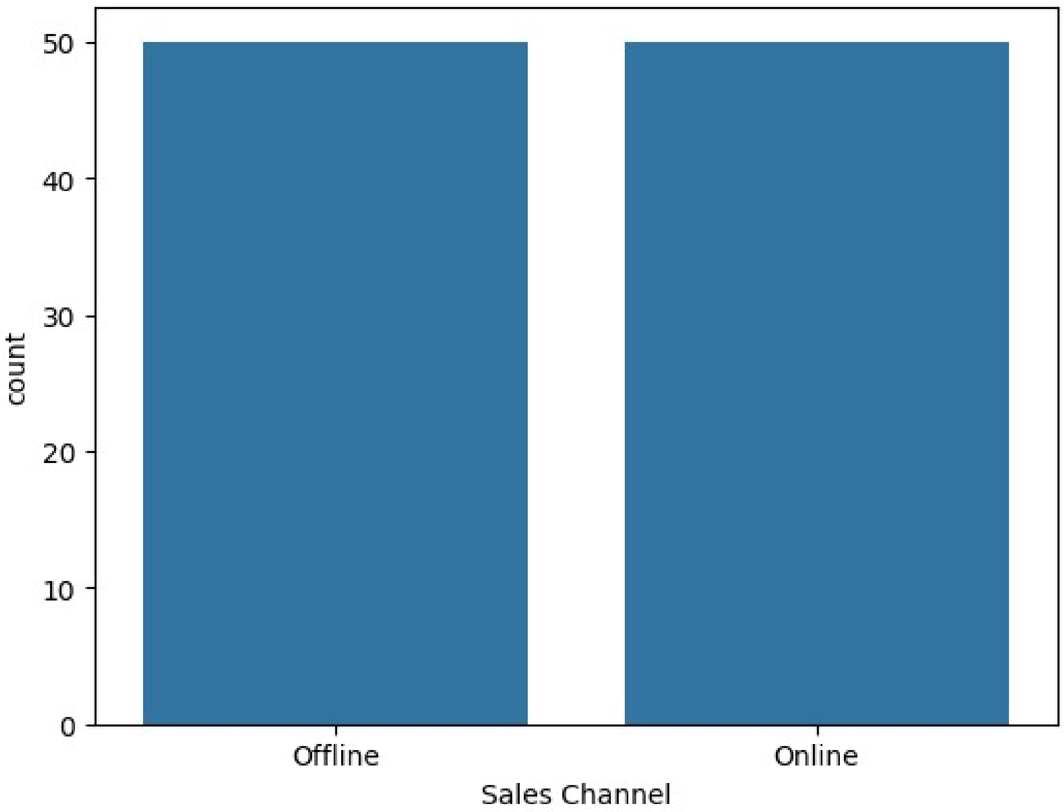
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Region          100 non-null    object
 1   Country         100 non-null    object
 2   Item Type       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
```

```
df.describe()
```

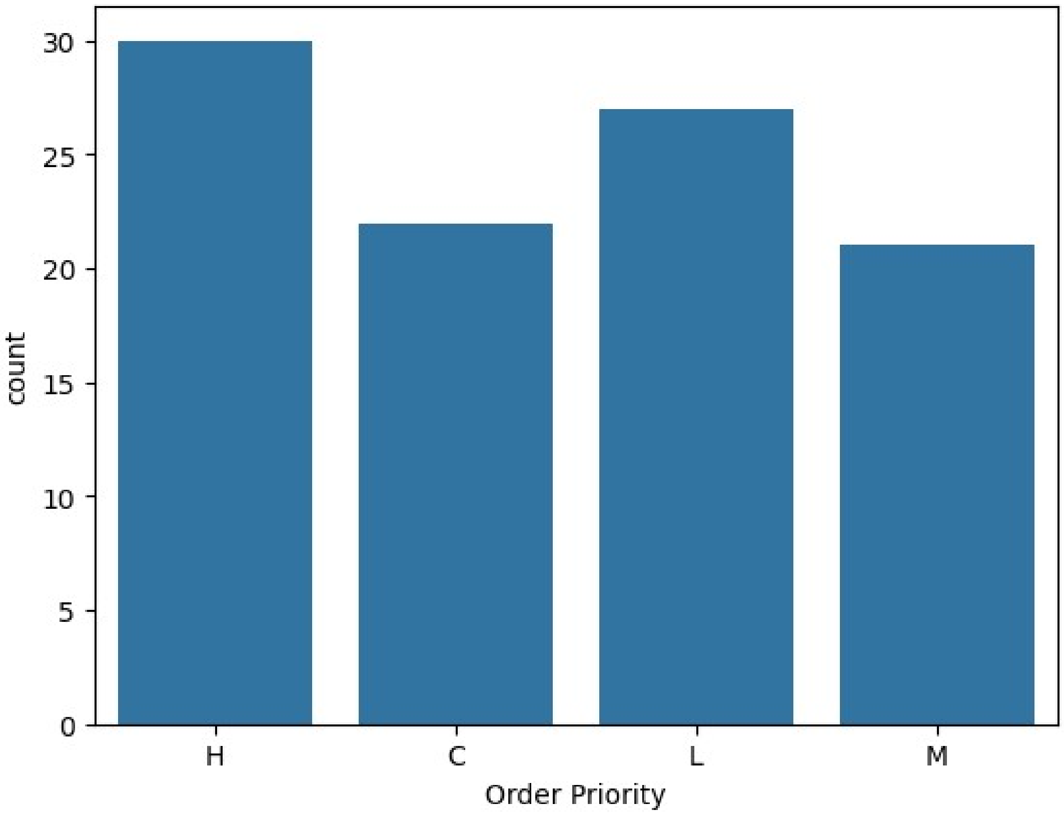
```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"Order ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 337613708.4653814,\n
\"min\": 100.0,\n        \"max\": 994022214.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
555020412.36,\n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

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

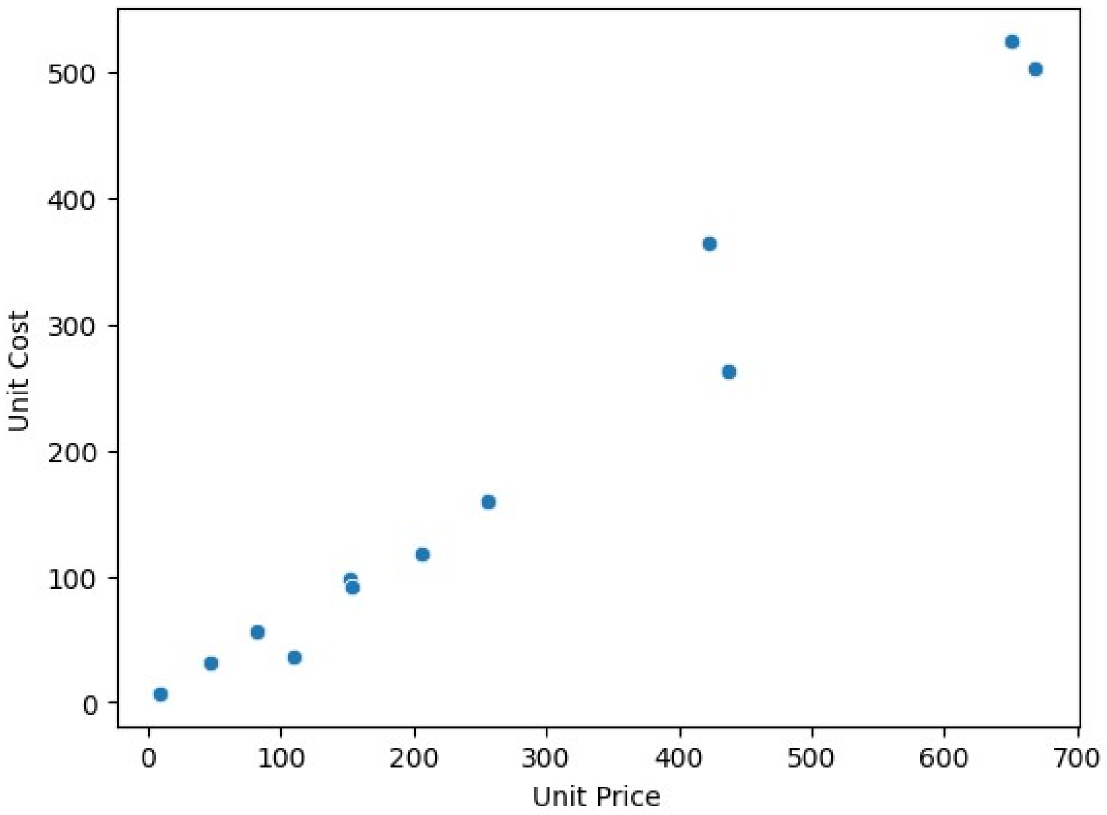


```
sns.countplot(x = 'Order Priority', data = df)
mplcyberpunk.add_glow_effects()
```



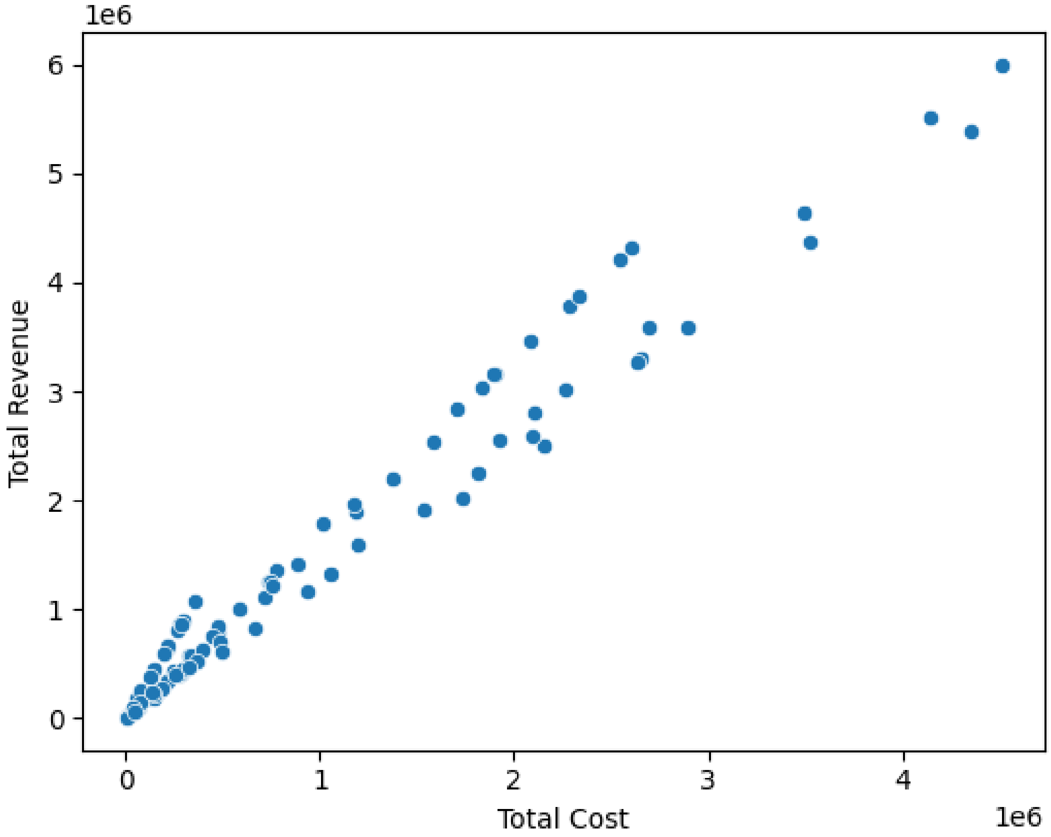
```
sns.scatterplot(x = 'Unit Price', y = 'Unit Cost', data = df)
```

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



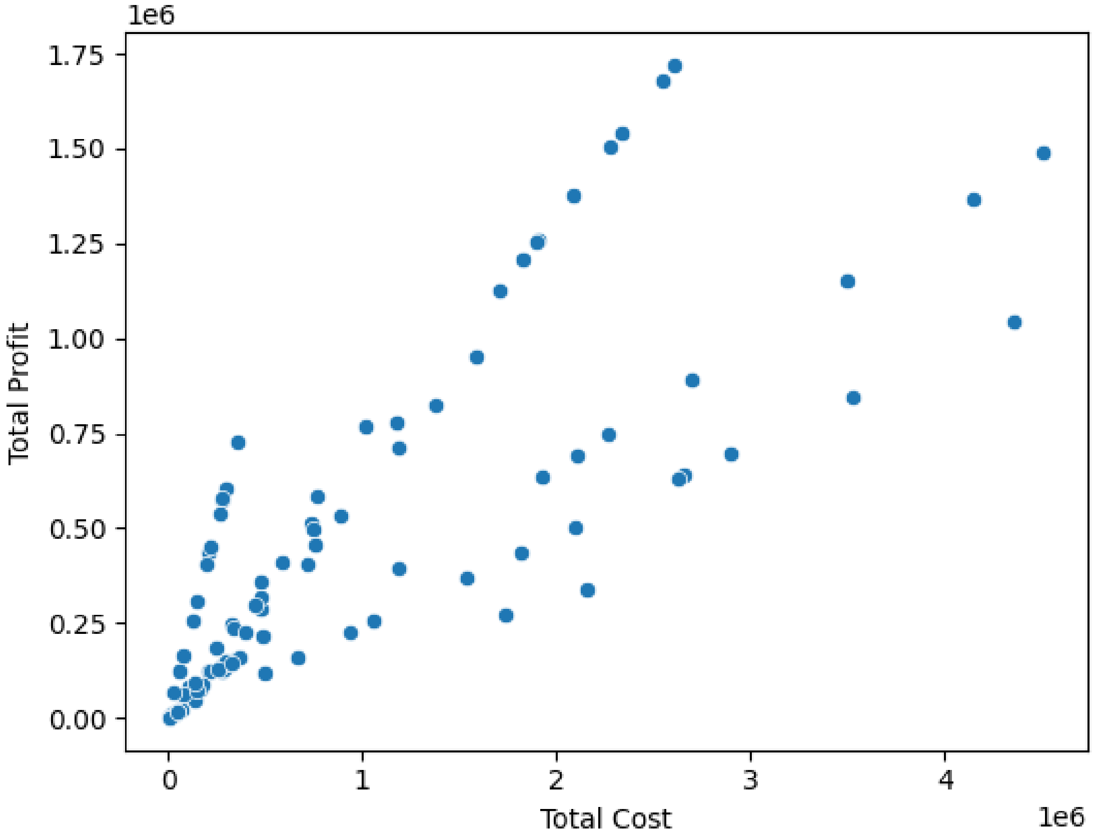
```
sns.scatterplot(x = 'Total Cost', y = 'Total Revenue', data = df)
```

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



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

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



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

```
Total revenue generated:  137348768.31
```

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

```
Total Cost:  93180569.91000001
```

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

```
Total profit generated:  44168198.39999999
```

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

```
Max profit: 1719922.04
```

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

```
Minimum profit: 1258.02
```

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

```
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
40.977541217770739,\n          13.558036455000117\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

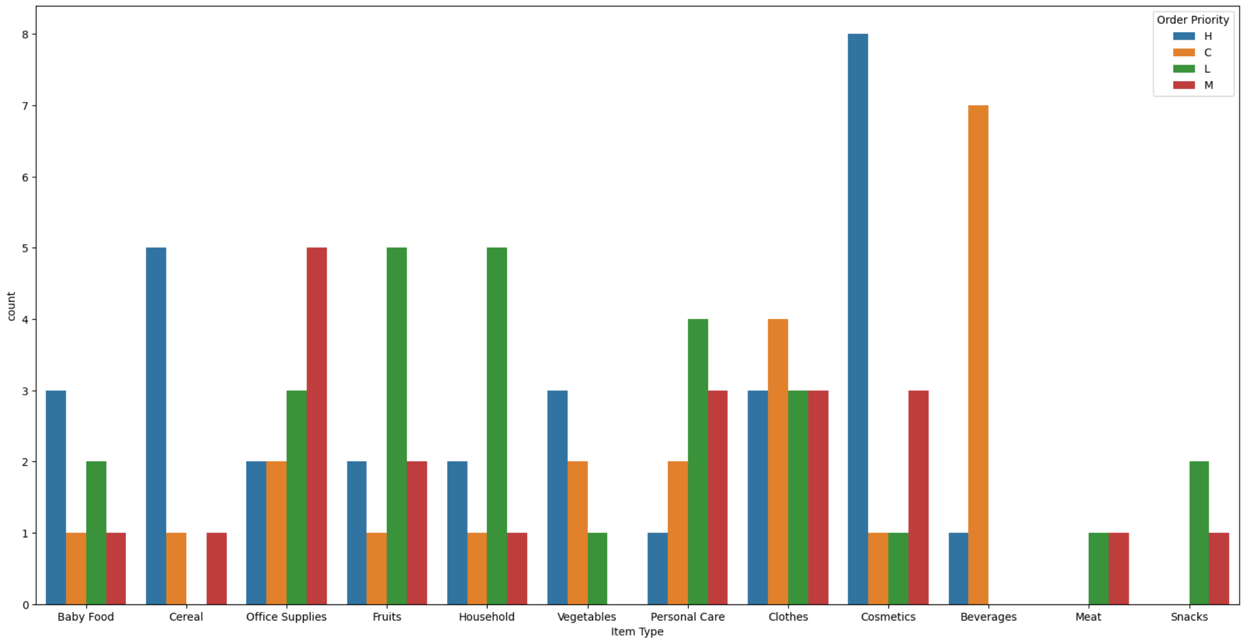
```
df['Average Revenue per Unit'] = df['Total Revenue']/df['Units Sold']

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\":
```

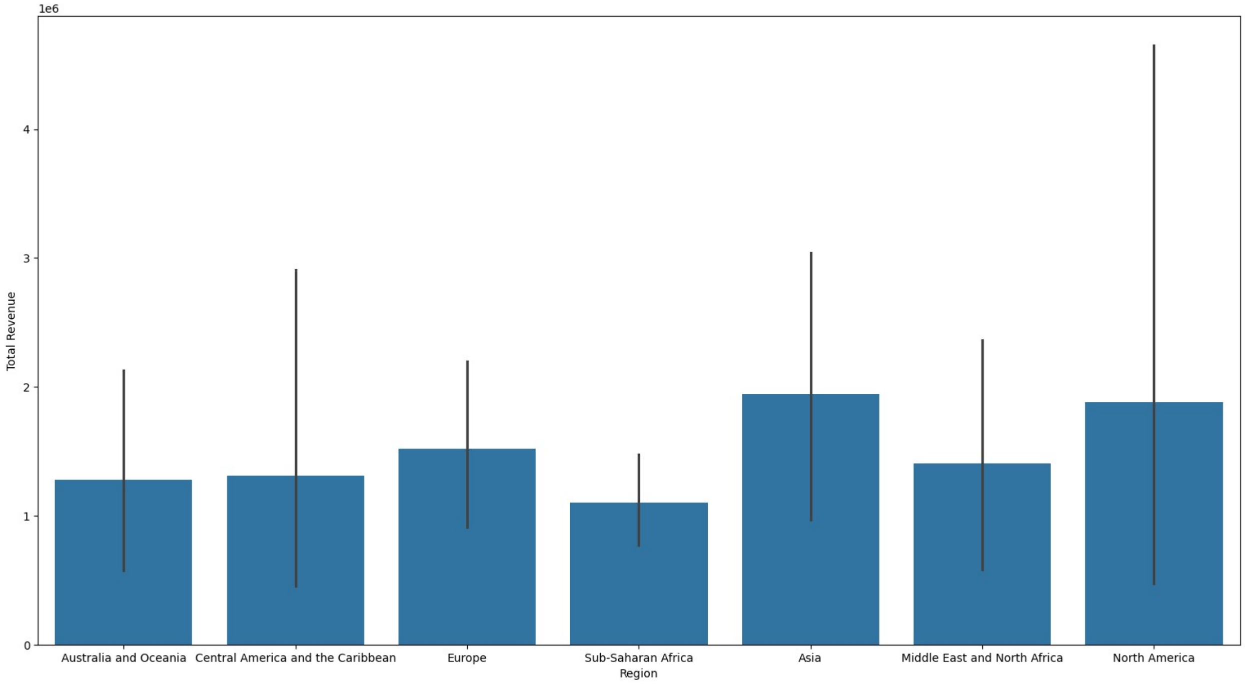
```
\"category\",\n        \"num_unique_values\": 12,\n
\"samples\": [\n          \"Meat\",\n          \"Beverages\",\n
\"Baby Food\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sales Channel\",\n      \"properties\": {\n        \"dtype\":
\"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"}
```

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

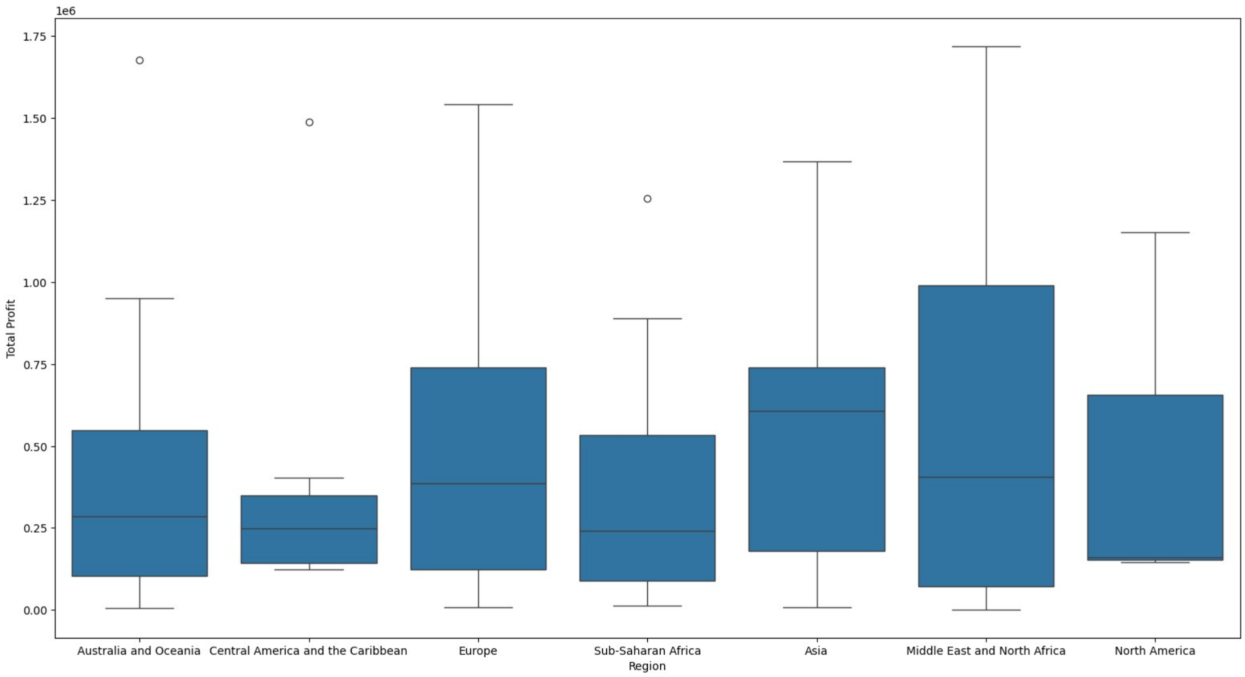
```
<Axes: xlabel='Item Type', ylabel='count'>
```



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

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



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

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



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

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

```
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
```

```
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')
```

```
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"}
```

```
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"}
```

```
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"}
```

```
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"}
```

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

```
new_df
```

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

```
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"}

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

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

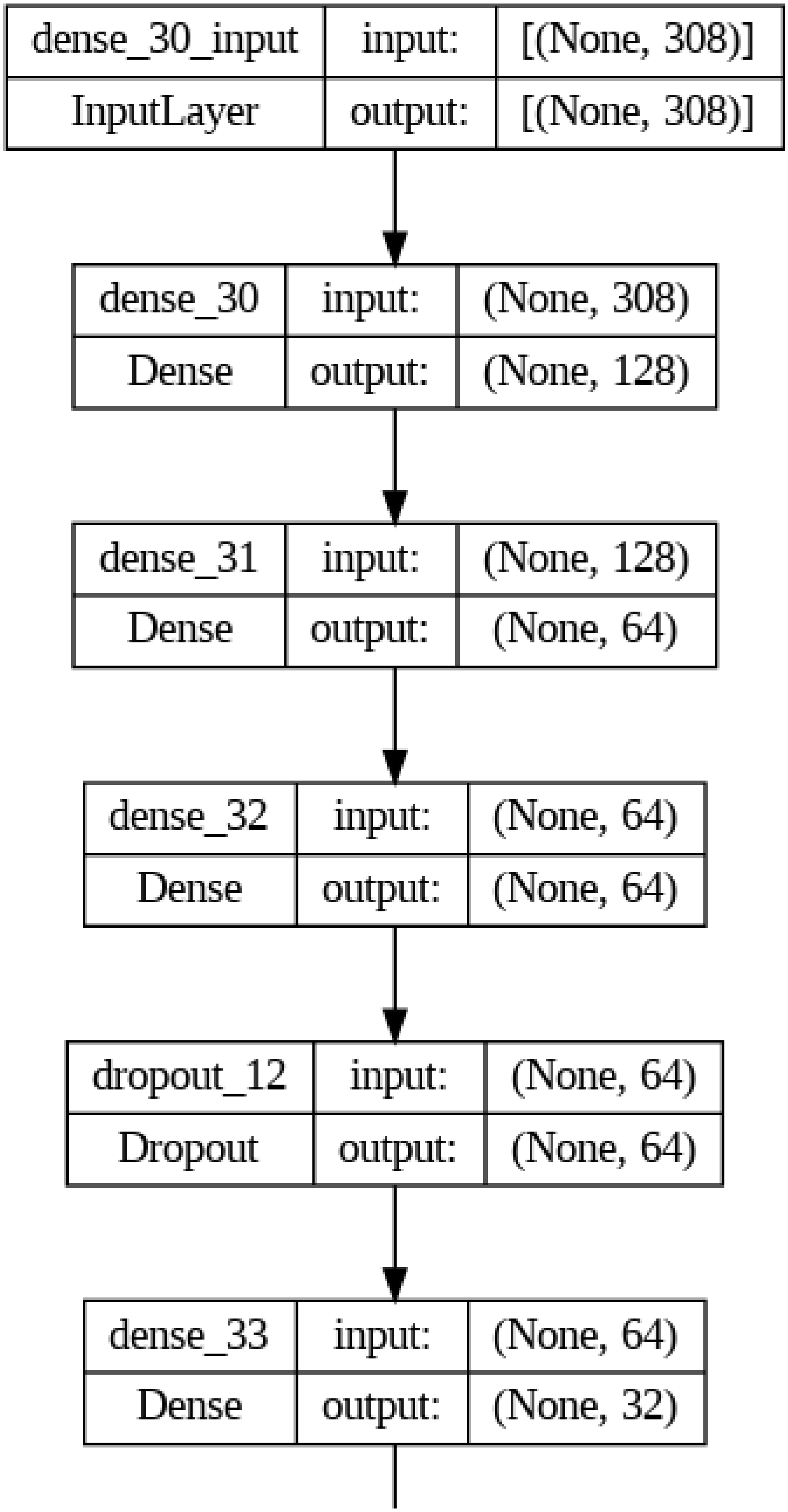
```
(100, 308)
(100, 1)
```

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

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

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

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

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units = 128, activation = 'relu',
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_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_30 (Dense)            (None, 128)               39552

 dense_31 (Dense)            (None, 64)                8256

 dense_32 (Dense)            (None, 64)                4160

 dropout_12 (Dropout)        (None, 64)                0

 dense_33 (Dense)            (None, 32)                2080

 dense_34 (Dense)            (None, 32)                1056

 dropout_13 (Dropout)        (None, 32)                0

 dense_35 (Dense)            (None, 16)                528

 dense_36 (Dense)            (None, 16)                272

 dropout_14 (Dropout)        (None, 16)                0

 dense_37 (Dense)            (None, 8)                 136

 dropout_15 (Dropout)        (None, 8)                 0

 dense_38 (Dense)            (None, 4)                 36

 dense_39 (Dense)            (None, 1)                 5

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

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


dense_30_input | input: | [(None, 308)]
InputLayer | output: | [(None, 308)]
dense_30 | input: | (None, 308)
Dense | output: | (None, 128)
dense_31 | input: | (None, 128)
Dense | output: | (None, 64)
dense_32 | input: | (None, 64)
Dense | output: | (None, 64)
dropout_12 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_33 | input: | (None, 64)
Dense | output: | (None, 32)

```
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])
```

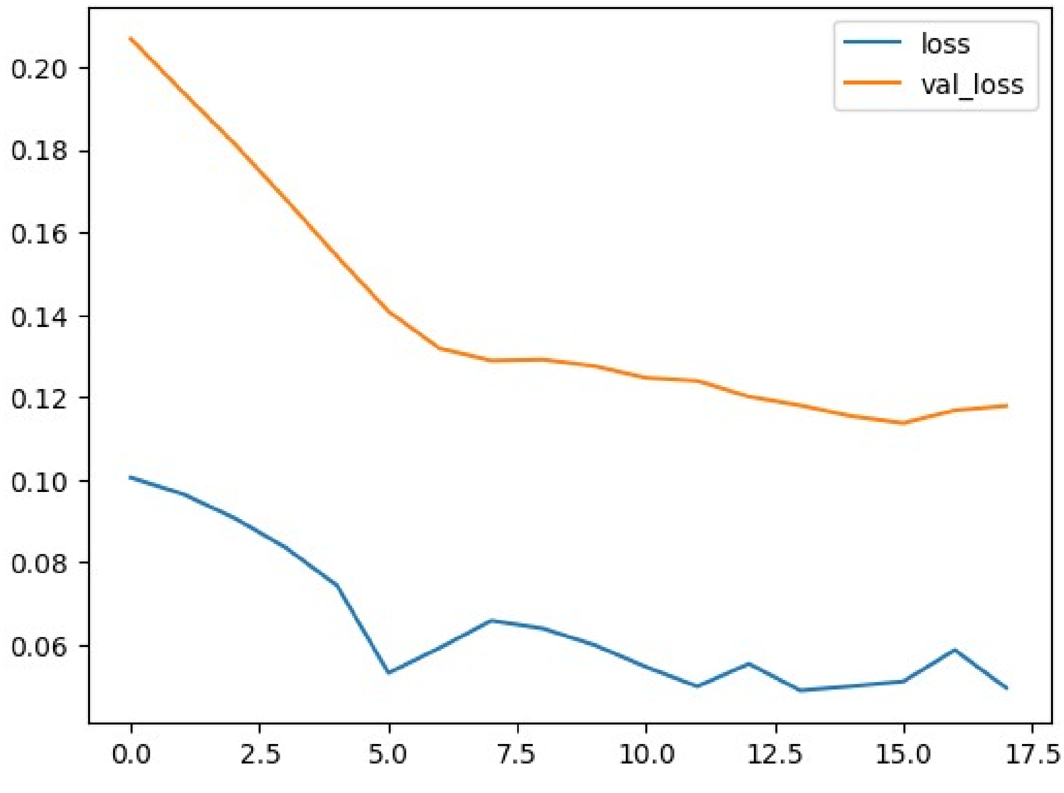
```
Epoch 1/25
8/8 [==============================] - 2s 36ms/step - loss: 0.1005 -
val_loss: 0.2069
Epoch 2/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0966 -
val_loss: 0.1941
Epoch 3/25
8/8 [==============================] - 0s 17ms/step - loss: 0.0908 -
val_loss: 0.1816
Epoch 4/25
8/8 [==============================] - 0s 13ms/step - loss: 0.0836 -
val_loss: 0.1681
Epoch 5/25
8/8 [==============================] - 0s 13ms/step - loss: 0.0744 -
val_loss: 0.1542
Epoch 6/25
8/8 [==============================] - 0s 11ms/step - loss: 0.0531 -
val_loss: 0.1408
Epoch 7/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0592 -
val_loss: 0.1318
Epoch 8/25
8/8 [==============================] - 0s 8ms/step - loss: 0.0658 -
val_loss: 0.1289
Epoch 9/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0639 -
val_loss: 0.1291
Epoch 10/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0599 -
val_loss: 0.1275
Epoch 11/25
8/8 [==============================] - 0s 8ms/step - loss: 0.0546 -
val_loss: 0.1247
Epoch 12/25
8/8 [==============================] - 0s 10ms/step - loss: 0.0499 -
val_loss: 0.1240
Epoch 13/25
8/8 [==============================] - 0s 9ms/step - loss: 0.0554 -
val_loss: 0.1201
Epoch 14/25
8/8 [==============================] - 0s 11ms/step - loss: 0.0489 -
val_loss: 0.1180
Epoch 15/25
8/8 [==============================] - 0s 8ms/step - loss: 0.0499 -
```

```
val_loss: 0.1154
Epoch 16/25
8/8 [==============================] - 0s 8ms/step - loss: 0.0511 -
val_loss: 0.1137
Epoch 17/25
8/8 [==============================] - 0s 12ms/step - loss: 0.0587 -
val_loss: 0.1168
Epoch 18/25
8/8 [==============================] - 0s 14ms/step - loss: 0.0495 -
val_loss: 0.1179
```

```
<keras.src.callbacks.History at 0x7ae081092740>
```

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

```
<Axes: >
```



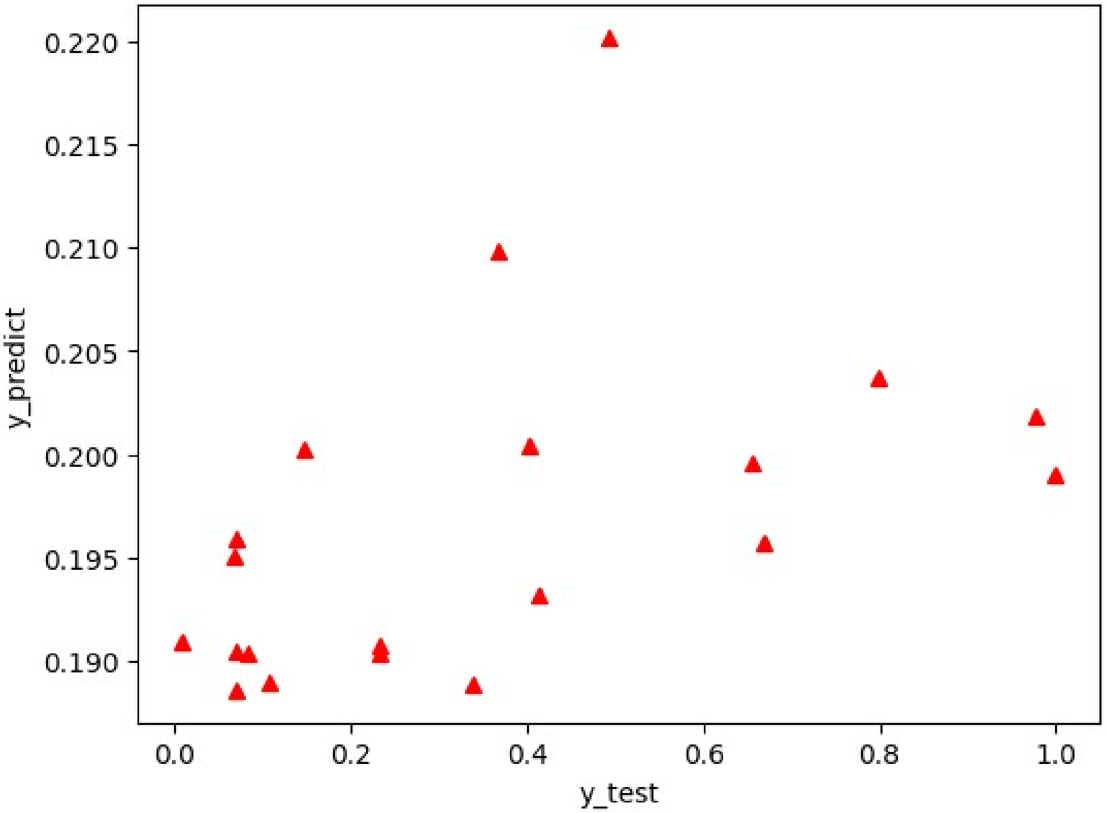
```
  y_predict = model.predict(X_test)
  y_predict
```

```
1/1 [==============================] - 0s 164ms/step
```

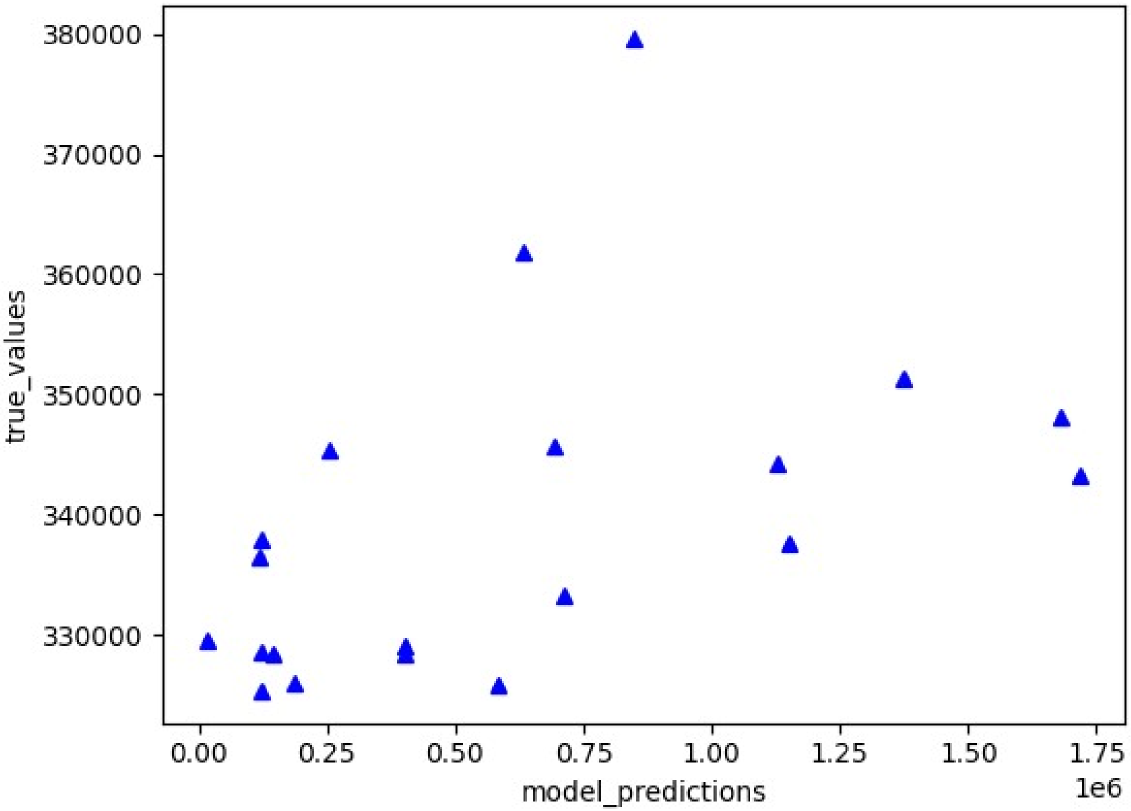
```
array([[0.18859655],
       [0.19958779],
       [0.19097182],
       [0.19319476],
       [0.19038466],
       [0.20367971],
       [0.19034201],
       [0.19897525],
       [0.20045404],
       [0.20978141],
       [0.1907322 ],
       [0.19045499],
       [0.18894072],
       [0.20020184],
       [0.20182079],
       [0.19589257],
       [0.1950508 ],
       [0.19572142],
       [0.220163  ],
       [0.18890244]], dtype=float32)
```

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

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



```
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')
```



```
k = X_test.shape
k
n = len(X_test)
print('\n')
n




20

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)
```

```
590152.018
```

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

```
348279404558.6156
```

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

```
438665.3042499999
```

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

```
-0.26509112819599023
```

```
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(
```