Mahatma Education Society's

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE

(Autonomous)

Re-accredited "A" Grade by NAAC (3rd Cycle)

Project Completion Certificate

THIS IS TO CERTIFY THAT

SURAJ SOMNATH SHETE

of **M.Sc. Data Analytics Part – I** has completed the project titled **Laptop Price Prediction** of subject **Big Data Analytics** under our guidance and supervision during the academic year 2023-24 in the department of Computer Science.

Project Guide	Course Coordinator	Head of the Department

Mahatma Education Society's
Pillai College of Arts, Commerce & Science
(Autonomous)
Affiliated to University of Mumbai
NAAC Accredited 'A' grade (3 cycles)
Best College Award by University of Mumbai
ISO 9001:2015 Certified

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE (Autonomous)

NEW PANVEL

PROJECT REPORT ON

"Laptop Price Prediction"

IN PARTIAL FULFILLMENT OF

MASTER OF SCIENCE IN DATA ANALYTICS

SEMESTER 1st– 2023-24

PROJECT GUIDE

Prof. Omkar Sherkhane

SUBMITTED BY: SURAJ SHETE

ROLL NO: 3143

CA-2 PROJECT

Dataset Description: -

- The Dataset contains 11 columns and 1000 rows.
- It contains data of laptops sales with detailed information
- It tells us about the CompanyName, TypeOfLaptop, Inches, ScreenResolution, Cpu, Ram, Memory, Gpu, OpSys, Weight and Price
- It may contain null values, unique values, categorical and numerical values.
- It may also contain false values or null rows and columns.
- In this model the predicted output will be “Price”

Analysis Tasks: -

- Using the above-mentioned dataset, we will perform all the phases of Data Science life cycle i.e., Data Understanding, Data Preparation, Data Visualization, Data Modelling and Model Evaluation

- The below table specifies the name of the columns, their data types, the feature is categorical or numerical.

Tools & Techniques used: -

- Tools: Google Collab
- Language used Python
- Exploratory Data Analysis
- Data Visualization
- Machine Learning for Model building

Table:

Column Name	Datatype	Categorical/N numerical/Unique
CompanyName	object	Categorical
TypeOfLaptop	object	Categorical
Inches	Float64	Numerical
ScreenResolution	object	Categorical
Cpu	object	Categorical
Ram	object	Categorical
Memory	object	Categorical
Gpu	object	Categorical
OpSys	object	Categorical
Weight	float64	Numerical
Price	float64	Numerical

Code and Output with their Explanation:

Importing the required libraries and reading the dataset:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Data Cleaning and Pre-processing:

```
df['CompanyName'].unique()
df['TypeOfLaptop'].unique()

df['ScreenResolution'].unique()
```

1. Data Cleaning and Preprocessing:

1.1 Missing Values:

- Check for missing values in each column.
- Decide on a strategy for handling missing data (e.g., imputation or removal)

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

	CompanyName	TypeOfLaptop	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	MSI	Business Laptop	17.040680	IPS Panel Retina Display 2560x1600	Intel Core i7	12GB	512GB SSD	Intel Iris Xe Graphics	Linux	2.064834	35844.099371
1	Chuwi	2 in 1 Convertible	16.542395	Full HD	Intel Core i5	12GB	128GB PCIe SSD	Intel Iris Xe Graphics	No OS	4.060656	37019.059051
2	hp	WorkStation	17.295294	Full HD	Intel Xeon E3-1505M	8GB	1TB HDD	Intel Iris Xe Graphics	Linux	2.901689	33329.360341
3	MSI	2 in 1 Convertible	11.526203	2K	Intel Core i7	16GB	512GB NVMe SSD	Intel Iris Xe Graphics	Windows 10	2.914843	68631.102486
4	Microsoft	Gaming	12.649634	Full HD	Intel Core i5	8GB	512GB SSD	AMD Radeon RX 5600M	Windows 10	4.341995	33842.479566
5	Apple	WorkStation	15.543249	HD 1920x1080	Intel Atom x5-Z8550	16GB	1TB NVMe SSD	NVIDIA GeForce GTX 1650	macOS	3.580368	83937.484249
6	lenevo	NoteBook	15.559451	2K	Intel Xeon E3-1505M	8GB	512GB SSD	Intel Iris Xe Graphics	Windows 11	4.050894	62896.017385
7	Asus	UltraBook	17.806917	IPS Panel Full HD / Touchscreen 1920x1080	Intel Celeron Dual Core 3855U	12GB	256GB PCIe SSD	NVIDIA GeForce GTX 1650	macOS	4.402627	35919.072831
8	Microsoft	Business Laptop	12.846039	IPS Panel Retina Display 2560x1600	Intel Core i9	12GB	128GB SSD	NVIDIA GeForce GTX 1650	Linux	3.700557	45011.851908
9	Microsoft	UltraBook	15.204180	Full HD	AMD A9-Series 9420	8GB	128GB PCIe SSD	NVIDIA GeForce GTX 1650	No OS	3.656769	52669.310830

Checking for Unique Values using .unique() function

```
df['CompanyName'].unique()

array(['MSI', 'Chuwi', 'hp', 'Microsoft', 'Apple', 'lenevo', 'Asus',
       'Acer', 'Dell'], dtype=object)

[5] df['TypeOfLaptop'].unique()

array(['Business Laptop', '2 in 1 Convertible', 'WorkStation', 'Gaming',
       'NoteBook', 'UltraBook'], dtype=object)

[6] df['ScreenResolution'].unique()

array(['IPS Panel Retina Display 2560x1600', 'Full HD', '2K',
       'HD 1920x1080 ', 'IPS Panel Full HD / Touchscreen 1920x1080', '4K'],
      dtype=object)

[7] df['Cpu'].unique()

array(['Intel Core i7', 'Intel Core i5', 'Intel Xeon E3-1505M ',
       'Intel Atom x5-Z8550', 'Intel Celeron Dual Core 3855U ',
       'Intel Core i9', 'AMD A9-Series 9420', 'AMD Ryzen 5',
       'AMD Ryzen 7', 'Intel Pentium Quad Core N4200'], dtype=object)
```

```
df.info()
df.shape
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   CompanyName       1000 non-null   object
 1   TypeOfLaptop      1000 non-null   object
 2   Inches            1000 non-null   float64
 3   ScreenResolution  1000 non-null   object
 4   Cpu               1000 non-null   object
 5   Ram               1000 non-null   object
 6   Memory            1000 non-null   object
 7   Gpu               1000 non-null   object
 8   OpSys             1000 non-null   object
 9   Weight            1000 non-null   float64
 10  Price             1000 non-null   float64
dtypes: float64(3), object(8)
memory usage: 86.1+ KB

[ ] df.shape

(1000, 11)
```

Checking for null values: As no null values were detected

```
df.isna().sum()
```

```
[ ] df.isna().sum()

CompanyName          0
TypeOfLaptop         0
Inches               0
ScreenResolution     0
Cpu                  0
Ram                  0
Memory               0
Gpu                  0
OpSys                0
Weight               0
Price                0
dtype: int64
```

Exploratory Data Analysis:

```
df.describe()
```

2. Exploratory Data Analysis (EDA):

2.1 Descriptive Statistics:

- Calculate basic statistics (mean, median, standard deviation) for numerical columns.
- Explore the distribution of categorical variables.

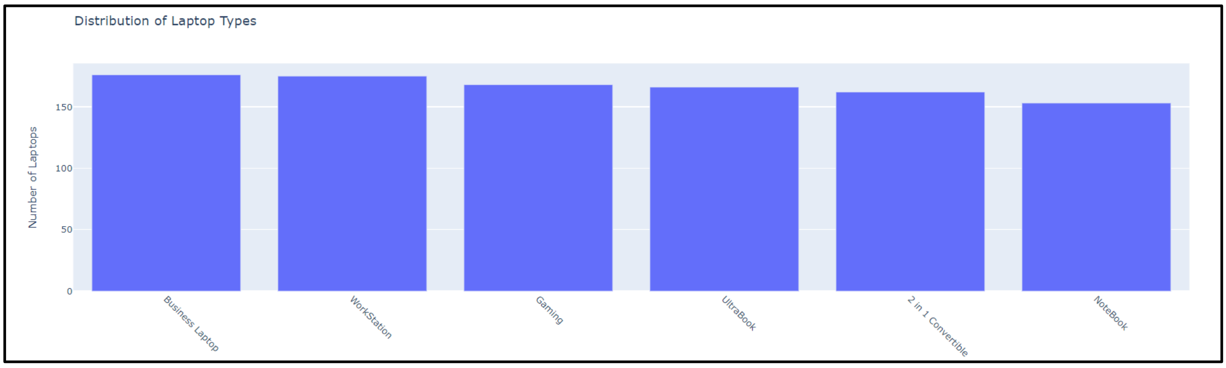
```
df.describe()
```

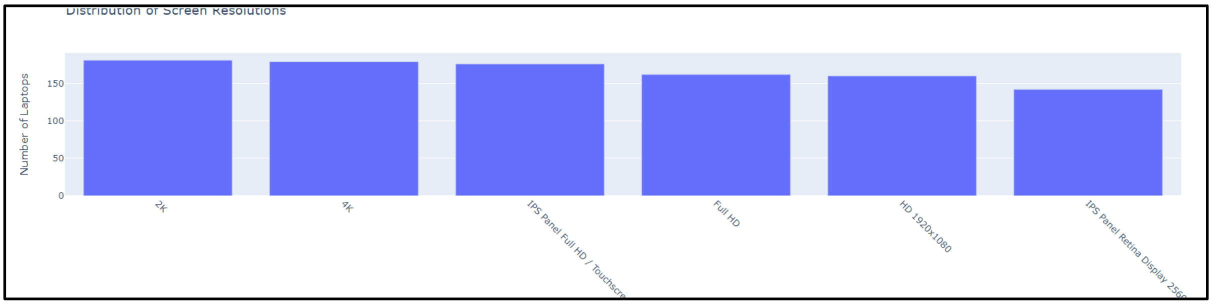
	Inches	Weight	Price
count	1000.000000	1000.000000	1000.000000
mean	14.496646	3.469800	51602.255339
std	2.066624	0.857112	13802.833231
min	11.005842	2.000819	30060.275100
25%	12.677791	2.720228	40376.617670
50%	14.509298	3.477824	50683.971717
75%	16.313026	4.189891	61897.280126
max	17.998786	4.994556	115137.368077

Count Plot for TypeOfLaptop and Screen Resolution:

```
# 1. Type of Laptop
type_count = df['TypeOfLaptop'].value_counts().reset_index()
type_count.columns = ['TypeOfLaptop', 'Count']

fig = px.bar(type_count, x='TypeOfLaptop', y='Count', title='Distribution of Laptop Types', labels={'Count': 'Number of Laptops', 'TypeOfLaptop': 'Type of Laptop'})
fig.update_layout(xaxis=dict(tickangle=45))
fig.show()

# 2. Screen Resolution
resolution_count = df['ScreenResolution'].value_counts().reset_index()
resolution_count.columns = ['ScreenResolution', 'Count']

fig = px.bar(resolution_count, x='ScreenResolution', y='Count', title='Distribution of Screen Resolutions', labels={'Count': 'Number of Laptops', 'ScreenResolution': 'Screen Resolution'})
fig.update_layout(xaxis=dict(tickangle=45))
fig.show()
```

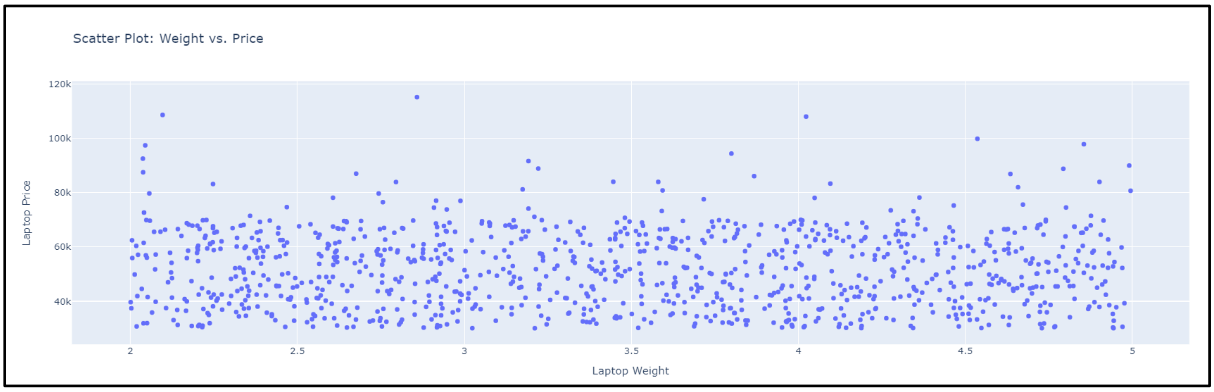

```
# 1. Type of Laptop
type_count = df['TypeOfLaptop'].value_counts().reset_index()
type_count.columns = ['TypeOfLaptop', 'Count']

fig = px.bar(type_count, x='TypeOfLaptop', y='Count', title='Distribution of Laptop Types', labels={'Count': 'Number of Laptops', 'TypeOfLaptop': 'Type of Laptop'})
fig.update_layout(xaxis=dict(tickangle=45))
fig.show()

# 2. Screen Resolution
resolution_count = df['ScreenResolution'].value_counts().reset_index()
resolution_count.columns = ['ScreenResolution', 'Count']

fig = px.bar(resolution_count, x='ScreenResolution', y='Count', title='Distribution of Screen Resolutions', labels={'Count': 'Number of Laptops', 'ScreenResolution': 'Screen Resolution'})
fig.update_layout(xaxis=dict(tickangle=45))
fig.show()
```

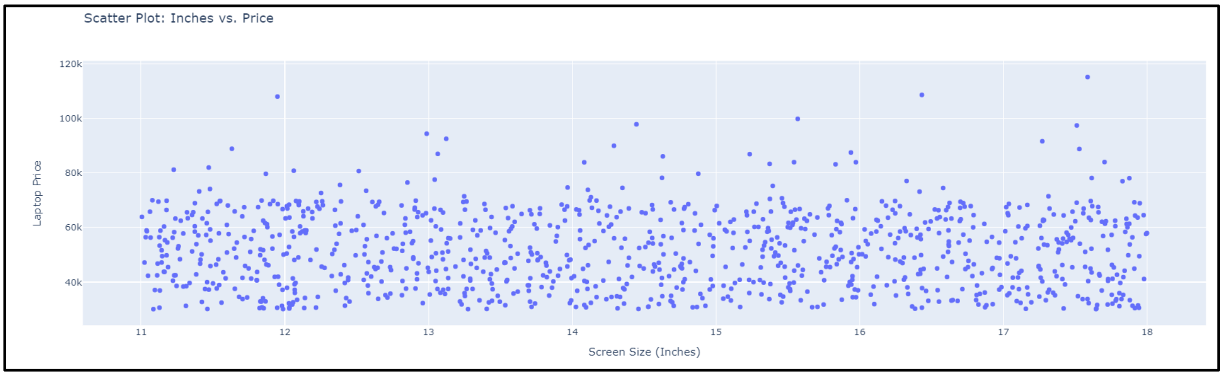





Scatter Plot for Numerical Values:

```
# Scatter Plot: Price vs. Weight
fig = px.scatter(df, x='Weight', y='Price', title='Scatter Plot: Weight vs. Price', labels={'Weight': 'Laptop Weight', 'Price': 'Laptop Price'})
fig.show()

# Scatter Plot: Price vs. Inches
fig = px.scatter(df, x='Inches', y='Price', title='Scatter Plot: Inches vs. Price', labels={'Inches': 'Screen Size (Inches)', 'Price': 'Laptop Price'})
fig.show()
```





Model Building and Label Encoding for

categorical attributes:

```
label_encoder = LabelEncoder()
df['CompanyName'] = label_encoder.fit_transform(df['CompanyName'])
df['TypeOfLaptop'] = label_encoder.fit_transform(df['TypeOfLaptop'])
df['ScreenResolution'] = label_encoder.fit_transform(df['ScreenResolution'])
df['Cpu'] = label_encoder.fit_transform(df['Cpu'])
df['Ram'] = label_encoder.fit_transform(df['Ram'])
df['Memory'] = label_encoder.fit_transform(df['Memory'])
df['Gpu'] = label_encoder.fit_transform(df['Gpu'])
df['OpSys'] = label_encoder.fit_transform(df['OpSys'])
```

	CompanyName	TypeOfLaptop	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	5	1	17.040680	5	6	0	15	1	0	2.064834	35844.099371
1	3	0	16.542395	2	5	0	0	1	1	4.060656	37019.059051
2	7	5	17.295294	2	9	3	3	1	0	2.901689	33329.360341
3	5	0	11.526203	0	6	1	14	1	2	2.914843	68631.102486
4	6	2	12.649634	2	5	3	15	0	2	4.341995	33842.479566
5	1	5	15.543249	3	3	1	4	2	4	3.580368	83937.484249
6	8	3	15.559451	0	9	3	15	1	3	4.050894	62896.017385
7	2	4	17.806917	4	4	0	7	2	4	4.402627	35919.072831
8	6	1	12.846039	5	7	0	1	2	0	3.700557	45011.851908
9	6	4	15.204180	2	0	3	0	2	1	3.656769	52669.310830

Selecting Target attributes for training and testing dataset:

```
x = df.iloc[:, :-1].values
x
```

```
y = df.iloc[:, 10].values
y
```

```
array([[ 5.        , 1.        , 17.04067994, ...,  1.        ,
         0.        , 2.06483418],
       [ 3.        , 0.        , 16.54239484, ...,  1.        ,
         1.        , 4.06065604],
       [ 7.        , 5.        , 17.29529434, ...,  1.        ,
         0.        , 2.90168919],
       ...,
       [ 8.        , 3.        , 13.76128764, ...,  2.        ,
         2.        , 4.04746848],
       [ 2.        , 4.        , 11.03799989, ...,  1.        ,
         1.        , 3.66982456],
       [ 4.        , 3.        , 11.00584228, ...,  1.        ,
         1.        , 4.79967463]])
```

Training and Testing of dataset:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=30)
```

```
x_train.size
```

```
7000
```

Building Linear Regression Model:

```
lr = LinearRegression()
lr.fit(x_train, y_train)
y_predict = lr.predict(x_test)
```

```
lr = LinearRegression()
lr.fit(x_train, y_train)
y_predict = lr.predict(x_test)
```

Taking random Values for Predicting Profit:

```
#taking random values for prediction
#{'CompanyName': 7, 'TypeOfLaptop': 2, 'Inches': 17.295294, 'ScreenResolution':
2, 'Cpu': 9, 'Ram': 3, 'Memory': 3, 'Gpu': 0, 'OpSys': 2, 'Weight': 2.914843}
y_pred= lr.predict([[7, 2, 17.295294, 2, 9, 3, 3, 0, 2, 2.914843]])
print(y_pred) #predicting laptop price
```

```
print(y_pred) #predicting laptop price
```

```
[50357.05470517]
```

R2 Score for X_Train and X_Test:

```
r2_score = lr.score(x_train, y_train)
print("Training Score:", r2_score*100, "%")
r2_score = lr.score(x_test, y_test)
print("Testing Score:", r2_score*100, "%")
```

Evaluation of actual profit and the predicted profit:

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Assuming y_true contains the actual prices
y_true = df['Price']

# Make predictions and assume y_pred contains the predicted prices
y_pred = lr.predict(x)

# Calculate evaluation metrics
mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_true, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
```

```
Mean Absolute Error (MAE): 11419.5391194173
Mean Squared Error (MSE): 189327690.67249292
Root Mean Squared Error (RMSE): 13759.639917980881
R-squared (R2): 0.0052540770276375826
```

Interpretation for Multi-linear Regression model:

Mean Absolute Error (MAE)	11419.5391194173
Mean Squared Error (MSE)	189327690.67249292
Root Mean Squared Error (RMSE)	13759.639917980881
R-squared (R2)	0.0052540770276375826