ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

Alliance College of Engineering and Design

Project Report

Master of Computer Application

Semester – II
Machine Learning Theory and Practice

[Laptops Price Prediction Dataset]

By
[Charan.S.R]
[2411022250055]

Department of Computer Application
Alliance University
Chandapura - Anekal Main Road, Anekal
Bengaluru - 562 106

March 2025

Project Report: Laptops Price Prediction dataset

1.Introduction

- This project aims to predict laptop prices using machine learning.
- The dataset includes specifications like Company, TypeName, Inches, Screen Resolution, Cpu, Ram, Memory, Gpu, OpSys, Weight, and Price.
- The process involves data preprocessing, exploratory analysis, feature engineering, and building a linear regression model.
- The dataset contains X rows and Y columns, with both numeric (Inches, Ram, Memory, Weight, Price) and categorical (Company, TypeName, ScreenResolution, Cpu, Gpu, OpSys) features.
- The target variable is Price, representing the cost of the laptop.

2. Methodology

- **Handling Missing Values:** Filled missing values with the mean for numeric columns and mode for categorical columns.
- **Outlier Removal:** Used the IQR method to remove extreme values in the Price column.
- **Ram & Memory Conversion:** Removed the GB suffix from Ram and summed multiple storage values in Memory.
- **Dropping Unnecessary Columns:** Removed non-numeric columns like Company, TypeName, ScreenResolution, Cpu, Gpu, and OpSys.
- **Correlation Heatmap & Price Distribution:** Plotted a heatmap to see relationships and a histogram to analyze Price distribution.
- **Feature Scaling**: Applied Min-Max Scaling and Z-score Standardization to normalize data.
- **Train-Test Split:** Divided the dataset into 80% training and 20% testing.
- **Linear Regression Model:** Trained the model and evaluated it using MSE and R^2 score

CODE:-

1. Import the necessary libraries

Defination:

- **pandas**: A Python library for data manipulation and analysis.
- **numpy:** A library for numerical computations and handling arrays.
- **matplotlib.pyplot:** A library for creating static, animated, and interactive visualizations.
- **seaborn:** A statistical data visualization library built on matplotlib.
- **LabelEncoder:** Converts categorical values into numerical values.
- **StandardScaler:** Standardizes features by removing the mean and scaling to unit variance.
- **MinMaxScaler:** Scales features to a fixed range, typically [0,1].
- **train_test_split:** Splits the dataset into training and testing sets.
- **LinearRegression:** A machine learning algorithm for linear regression models.

- **mean_squared_error:** Measures the average squared difference between actual and predicted values.
- **r2_score:** Evaluates the goodness of fit of a model.

Python Code:

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

Code Explaination:

Importing Libraries – Loads necessary tools for data analysis, visualization, and machine learning.

Data Preprocessing – Handles missing values, converts categorical data, and scales features for better model performance.

Model Building – Uses Linear Regression to predict laptop prices.

Model Evaluation – Checks accuracy using Mean Squared Error (MSE) and R-squared (R^2) score**.**

2. Load the dataset

Defination:

- **pd.read_csv():** Reads a CSV file and loads it into a Pandas DataFrame.
- **df:** The variable that stores the dataset as a DataFrame.

Python Code:

```
df = pd.read_csv('C:/MCA 2nd sem/ML/laptop_data.csv')
df
```

Code Explaination:

- pd.read_csv() is used to load the dataset from the given file path.
- 'C:/MCA 2nd sem/ML/laptop_data.csv' is the path to the dataset stored on your computer.
- df stores the data in a DataFrame, which is a table-like structure in pandas.
- df (last line) displays the entire dataset in a Jupyter Notebook or an interactive Python environment.

Output:-

[5]:

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080
...	...	...	...	...	...	...	...	...	...	...	...	...
1298	1298	Lenovo	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD	Intel HD Graphics 520	Windows 10	1.8kg	33992.6400
1299	1299	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD	Intel HD Graphics 520	Windows 10	1.3kg	79866.7200
1300	1300	Lenovo	Notebook	14.0	1366x768	Intel Celeron Dual Core N3050 1.6GHz	2GB	64GB Flash Storage	Intel HD Graphics	Windows 10	1.5kg	12201.1200
1301	1301	HP	Notebook	15.6	1366x768	Intel Core i7 6500U 2.5GHz	6GB	1TB HDD	AMD Radeon R5 M330	Windows 10	2.19kg	40705.9200
1302	1302	Asus	Notebook	15.6	1366x768	Intel Celeron Dual Core N3050 1.6GHz	4GB	500GB HDD	Intel HD Graphics	Windows 10	2.2kg	19660.3200

1303 rows × 12 columns

3. Finding Missing Values

Defination:

- **df.isnull():** Identifies missing values in the dataset.
- **sum():** Calculates the total number of missing values per column.

Python Code:

```
df.isnull().sum()
```

Code Explaination:

1. **Checks for missing values** – Identifies NaN (null) values in each column of the dataset.
2. **Counts missing values** – Tells how many values are missing per column.
3. **Helps identify issues** – Shows which columns need attention.
4. **Fixing missing values:**
   - **For numbers** → Fill with the **mean** (average).
   - **For categories** → Fill with the **most common value** (mode).
5. **Prepares data for analysis** – Ensures a complete dataset for machine learning or analysis.

Output:-

```
Unnamed: 0          0
Company             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price               0
dtype: int64
```

Output Explaination:- After performing the missing value check using the df.isnull().sum() command, it was found that there are no missing values in the dataset. This means that all columns contain complete data, with no empty or null (NaN) values.

4. Replace Missing Value By Mean

Defination:

- **select_dtypes(include=[np.number]):** Selects numerical columns in the dataset.
- **fillna():** Fills missing values with a specified method (mean in this case).
- **Mean():** The mean is the average of a set of numbers. It is calculated by adding all the values and dividing by the total number of values.

Python Code:

```
numeric_columns = df.select_dtypes(include=[np.number]).columns
df[numeric_columns]=df[numeric_columns].fillna(df[numeric_columns].mean())
df
```

Code Explaination:

This code fills missing values in numeric columns with the column's mean.

1. df.select_dtypes(include=[np.number]).columns selects only numeric columns from the dataset.
2. df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean()) replaces missing values in these columns with their respective mean.

Output:

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080
...	...	...	...	...	...	...	...	...	...	...	...	...
1298	1298	Lenovo	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD	Intel HD Graphics 520	Windows 10	1.8kg	33992.6400
1299	1299	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD	Intel HD Graphics 520	Windows 10	1.3kg	79866.7200
1300	1300	Lenovo	Notebook	14.0	1366x768	Intel Celeron Dual Core N3050 1.6GHz	2GB	64GB Flash Storage	Intel HD Graphics	Windows 10	1.5kg	12201.1200
1301	1301	HP	Notebook	15.6	1366x768	Intel Core i7 6500U 2.5GHz	6GB	1TB HDD	AMD Radeon R5 M330	Windows 10	2.19kg	40705.9200
1302	1302	Asus	Notebook	15.6	1366x768	Intel Celeron Dual Core N3050 1.6GHz	4GB	500GB HDD	Intel HD Graphics	Windows 10	2.2kg	19660.3200

1303 rows × 12 columns

5. Check Outlier

Defination:

- **Outlier:-** An outlier is a data point that is significantly different from other values in a dataset. It is either much higher or much lower than the rest of the data.
- **plt.figure(figsize=(10, 6)):** Sets the figure size for visualization.
- **sns.boxplot():** Creates a boxplot to visualize outliers.
- **plt.title():** Adds a title to the plot.
- **plt.show():** Displays the plot.

Python Code:

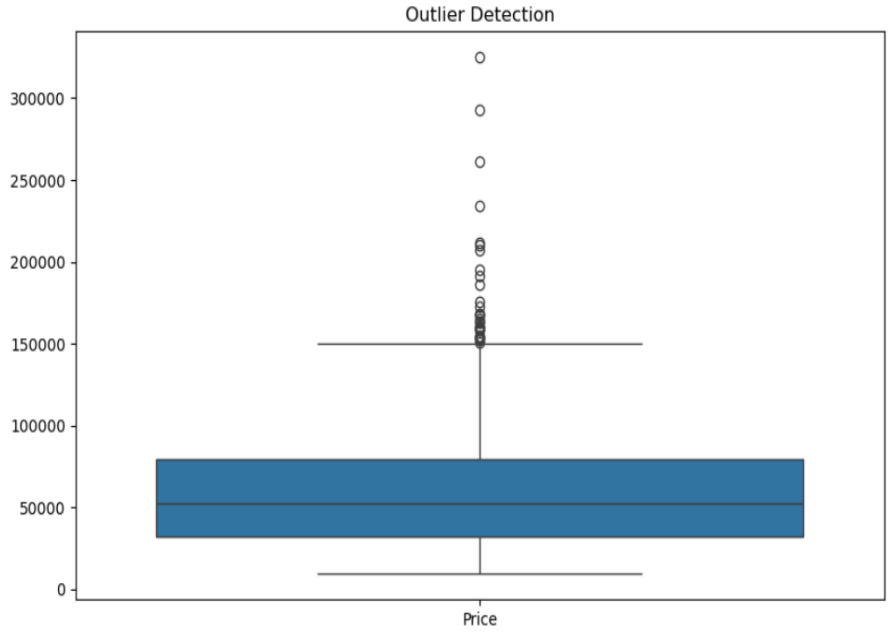
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['Price']])
plt.title("Outlier Detection")
plt.show()
```

Code Explaination:

This code creates a boxplot to detect outliers in the Price column.

1. plt.figure(figsize=(10, 6)) sets the figure size for better visibility.
2. sns.boxplot(data=df[['Price']]) generates a boxplot for the Price column, helping to identify extreme values.
3. plt.title("Outlier Detection") adds a title to the plot.
4. plt.show() displays the plot.

Output:



Output Explaination:-

Yes, based on the boxplot analysis in the code (sns.boxplot(data=df[['Price']])), there are outliers in the dataset, specifically in the "Price" column.

The outliers DataFrame stores all rows where the "Price" is outside the lower or upper bound.

And outliers.shape[0] gives the total number of outliers.

6. Drop Outliers

Definition:

- **quantile():** Computes quantiles of a dataset.
- **IQR (Interquartile Range):** Measures statistical dispersion to detect outliers.
- **sns.boxplot():** Displays a boxplot of Price values to visualize outliers.
- **Filtering Data:** Removes data points outside the lower and upper bounds.

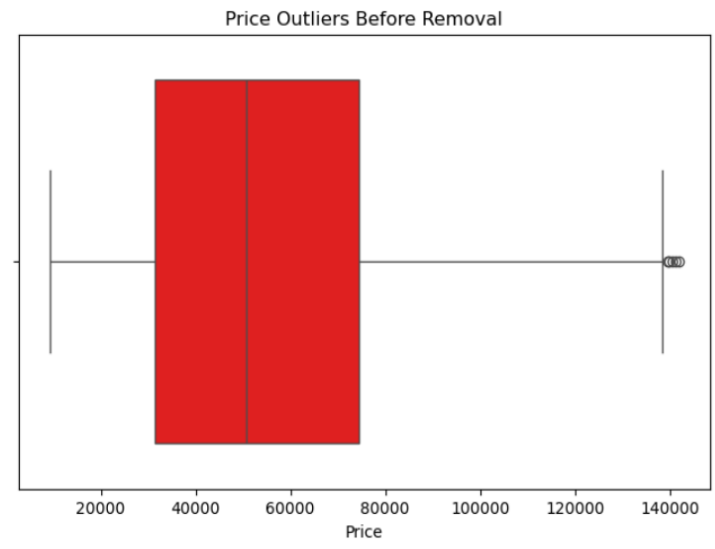
Python Code:

```
#Here i assumed Outliers are Above 1.5 * IQR
Q1 = df['Price'].quantile(0.25)
Q3 = df['Price'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

plt.figure(figsize=(8, 5))
sns.boxplot(x=df['Price'], color='red')
plt.title("Price Outliers Before Removal")
plt.xlabel("Price")
plt.show()
```

Code Explaination:

1. Calculate IQR
    - Q1 and Q3 represent the 25th and 75th percentiles of the Price column.
    - IQR = Q3 - Q1 calculates the interquartile range.
    - The lower bound and upper bound define the range within which most values fall.
    - Any value beyond 1.5 * IQR is considered an outlier.
2. Boxplot for Outlier Detection
    - plt.figure(figsize=(8, 5)) sets the figure size.
    - sns.boxplot(x=df['Price'], color='red') creates a boxplot for Price, highlighting outliers.
    - plt.title("Price Outliers Before Removal") adds a title.
    - plt.xlabel("Price") labels the x-axis.
    - plt.show() displays the plot.

Output:



7. Performing Label Encoding

Definition:

- **LabelEncoder():** Converts categorical labels into numerical values.
- **fit_transform():** Fits the encoder and transforms categorical values into numbers.

Python Code:

```
#Here I Performed label encoding for categorical variables
label_encoder = LabelEncoder()
categorical_columns = ['Company', 'TypeName', 'ScreenResolution', 'Cpu', 'Memory', 'Gpu', 'OpSys']
for col in categorical_columns:
    df[col] = label_encoder.fit_transform(df[col])
df
```

Code Explaination:

1. Label Encoder is created to convert text into numbers.
2. A list of columns with text values is selected.
3. Each column is changed so that every unique word gets a number.
    - Example: "Dell" → 0, "HP" → 1, "Apple" → 2.
4. The dataset is updated with these new numbers.

This step is needed because computers work with numbers, not words. Now, the data is ready for further processing.

Output:

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0	1	4	13.3	23	65	8GB	4	56	8	1.37kg	71378.6832
1	1	1	4	13.3	1	63	8GB	2	49	8	1.34kg	47895.5232
2	2	7	3	15.6	8	74	8GB	14	51	4	1.86kg	30636.0000
3	3	1	4	15.4	25	85	16GB	27	8	8	1.83kg	135195.3360
4	4	1	4	13.3	23	67	8GB	14	57	8	1.37kg	96095.8080
...	...	...	...	...	...	...	...	...	...	...	...	...
1298	1298	10	0	14.0	13	88	4GB	4	45	5	1.8kg	33992.6400
1299	1299	10	0	13.3	19	88	16GB	27	45	5	1.3kg	79866.7200
1300	1300	10	3	14.0	0	34	2GB	32	38	5	1.5kg	12201.1200
1301	1301	7	3	15.6	0	88	6GB	10	19	5	2.19kg	40705.9200
1302	1302	2	3	15.6	0	34	4GB	24	38	5	2.2kg	19660.3200

1274 rows × 12 columns

8. Check Cor-relation

Definition:

- **Correlation:** Measures the relationship between two variables. A high correlation indicates a strong relationship.
- **astype():** Converts a column to a specified data type.
- **str.replace():** Replaces specific characters in a string.
- **str.extract():** Extracts specific patterns from strings using regular expressions.

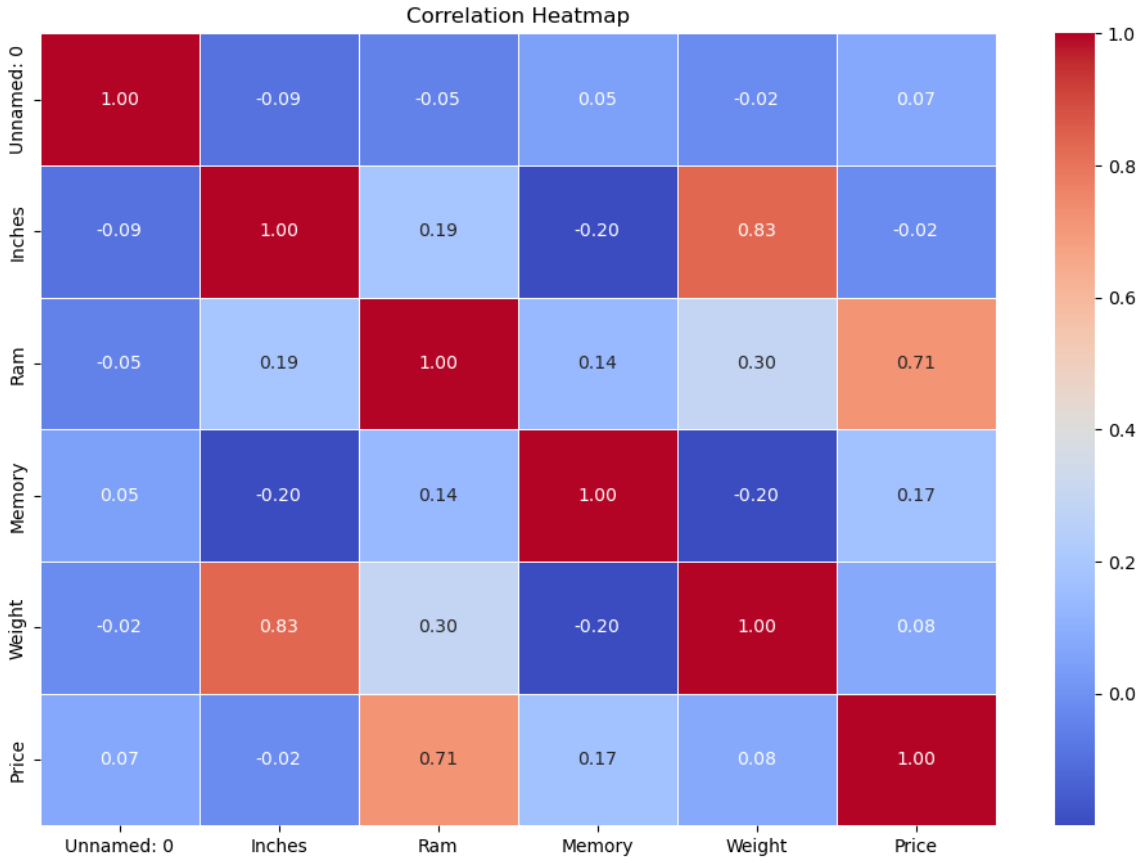
Python Code:

```
#To Do Correlation I am Converting 'Ram' to numeric
df['Ram'] = df['Ram'].astype(str).str.replace('GB', '', regex=True).astype(float)

# Same here I am Converting 'Memory' to numeric
df['Memory'] = df['Memory'].astype(str).str.extract('(\d+)').astype(float)

# I followed Same thing here to Convert 'Weight' to numeric
df['Weight'] = df['Weight'].astype(str).str.replace('kg', '', regex=True).astype(float)

# Now here i am  Dropping categorical data
drop1 = ['Company', 'TypeName', 'ScreenResolution', 'Cpu', 'Gpu', 'OpSys']
df_numeric = df.drop(columns=[col for col in drop1 if col in df.columns])

# Here i am writing code to Handle missing values and if any missing values found then i am replacing with mean
df_numeric.fillna(df_numeric.mean(), inplace=True)

# Plotting the correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```

Code Explaination:

1. **Convert text to numbers:** Removes "GB" from Ram, "kg" from Weight, and extracts numbers from Memory.
2. **Remove unnecessary columns:** Drops text-based columns like Company, Cpu, and OpSys since they are not needed.
3. **Handle missing values:** Fills missing values with the column’s average.
4. **Create a heatmap:** Shows how different features like Ram, Memory, Weight, and Price are related.

Output:



9. Check outcome proportionality

Definition:

- **Histogram (sns.histplot()):** A graphical representation showing the distribution of numerical data.
- **Kernel Density Estimation (KDE):** A smooth curve showing the probability density function of the dataset.
- **plt.title():** Sets the title of the graph

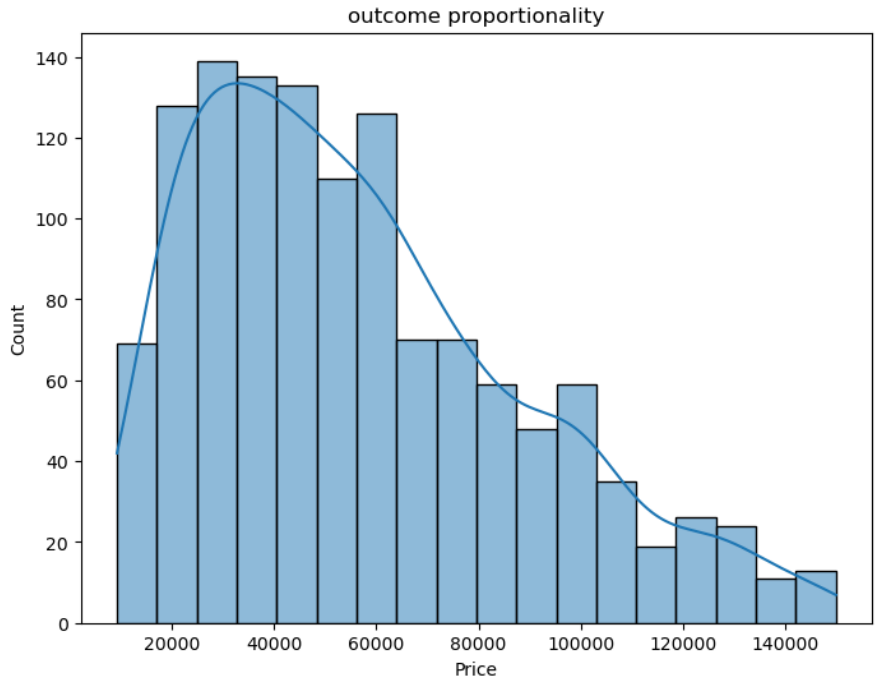
Python Code:

```
plt.figure(figsize=(8, 6))
sns.histplot(df['Price'], kde=True)
plt.title("proportionality")
plt.show()
```

Code Explaination:

1. **Histogram Plot:** Displays the frequency of different laptop prices.
2. **KDE (Kernel Density Estimation):** Adds a smooth curve to show the overall price trend.
3. **Title:** Labels the chart as "Proportionality" to indicate how prices are spread out.

Output:



10. Separate independent variable and target variable

Definition:

- **Independent Variables (X):** Features used to predict laptop prices.
- **Target Variable (y):** The price of laptops (dependent variable).
- **drop():** Removes the "Price" column from the dataset to create independent features

Python Code:

```
x=df.drop('Price', axis=1)  #I am declaring X variable as Independent variables
y=df['Price'] #I am declaring y variable as Target variables
y
```

Code Explaination:

1. **x (Independent Variables):** All columns except Price (features used to predict price).
2. **y (Target Variable):** Only the Price column (the value we want to predict).

Now, X will be used as input, and y as the output in the machine learning model.

Output:

```
0        71378.6832
1        47895.5232
2        30636.0000
3       135195.3360
4        96095.8080
            ...
1298     33992.6400
1299     79866.7200
1300     12201.1200
1301     40705.9200
1302     19660.3200
Name: Price, Length: 1261, dtype: float64
```

11.Apply Normalization and Standardization

Definition:

- **Normalization (MinMaxScaler):** Scales features between 0 and 1.
- **Standardization (StandardScaler):** Centers the data around mean = 0 and standard deviation = 1.

Python Code:

```
# Normalization
min_max_scaler = MinMaxScaler()
X_normalized = min_max_scaler.fit_transform(X)

# Standardization
standard_scaler = StandardScaler()
X_standardized = standard_scaler.fit_transform(X)
```

Code Explaination:

1. **Normalization (Min-Max Scaling):** Scales values between 0 and 1 to keep all features in the same range.
2. **Standardization (Z-score Scaling):** Centers the data around mean = 0 and standard deviation = 1 to handle different feature distributions.

12.Implementing Linear Regression

Definition:

- **train_test_split():** Splits data into training and testing sets.
- **LinearRegression():** A machine learning algorithm for regression problems.
- **fit():** Trains the model on the dataset.
- **predict():** Makes predictions using the trained model.
- **mean_squared_error (MSE):** Measures the average squared difference between actual and predicted values.
- **r2_score (R^2):** Measures how well the model explains the variance in the dataset.

Python Code:

```
#testing
X_train, X_test, y_train, y_test = train_test_split(X_standardized, y, test_size=0.2, random_state=42)

# Training the model
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Using mean sqaured error to evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:",r2)
```

```
# Plotting
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()])
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Linear Regression")
plt.show()
```

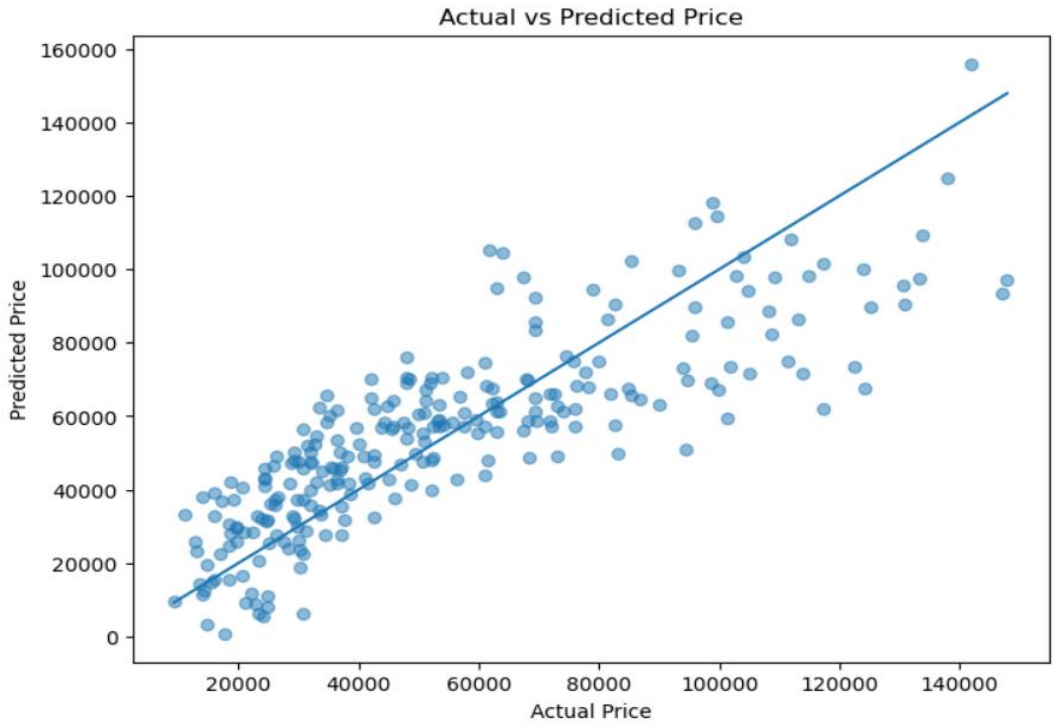
Code Explaination:

This code trains and evaluates a Linear Regression model to predict laptop prices:

1. **Splitting Data:** The dataset is divided into 80% training and 20% testing to evaluate performance.
2. **Training the Model:** A Linear Regression model is trained using the training data.
3. **Making Predictions:** The model predicts laptop prices for the test data.
4. **Evaluating the Model:**
    - Mean Squared Error (MSE): Measures prediction error. Lower is better.
    - R-squared (R^2): Shows how well the model fits. Closer to 1 is better.
5. **Plotting Results:** A scatter plot compares actual vs. predicted prices to check accuracy

Output:

```
Mean Squared Error: 314699098.16824865
R-squared: 0.6872530506340124
```



Conclusion

- The linear regression model predicted laptop prices reasonably well.
- Ram and Memory were the most significant factors influencing price.
- Outliers, especially in high-priced laptops, affected model accuracy.
- The model assumes a linear relationship, which may not always hold true.
- Limited features like brand reputation and build quality were not considered.
- Limited features like brand reputation and build quality were not considered.
- Hyperparameter tuning can further optimize model performance.

GitHub Link:-

https://github.com/charansr18/Machine-Learning-Assignment