

Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

Laptop Price Prediction



Supervised By:

Mrunal Paliwal

Submitted By:

Kashvi Sharma(2210990499) G7

Khushi Gupta(2210990512)G7

Keshav Wadhwa(2210990506)G7

**Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab**

ABSTRACT

In recent years, the demand for laptops has surged exponentially, driven by advancements in technology, remote work trends, and the growing reliance on digital tools for various purposes. Consequently, accurately predicting laptop prices has become paramount for both consumers and retailers. This abstract delves into the application of machine learning techniques for predicting laptop prices.

The proposed model utilizes a dataset comprising various features such as brand, specifications, processor type, RAM, storage capacity, display size, and other relevant attributes to predict the price of laptops.

Several machine learning algorithms are evaluated to determine the most suitable approach for price prediction. These algorithms include linear regression, decision trees, random forests, support vector machines, and gradient boosting techniques. Hyperparameter tuning and cross-validation are utilized to optimize the performance of the selected algorithms.

Furthermore, the model's performance is assessed using appropriate evaluation metrics such as mean absolute error, mean squared error, and R-squared score.

The findings of this study provide valuable insights into the factors influencing laptop prices and demonstrate the efficacy of machine learning in addressing pricing challenges in the consumer electronics industry. The developed model can assist consumers in making informed purchasing decisions and aid retailers in pricing strategies and inventory management. Additionally, it lays the groundwork for future research in price prediction and demand forecasting for electronic devices.

Table of Contents

<u>Sr.No</u>	<u>TITLE</u>	<u>Page no.</u>
1	Introduction 1. Background 2. Objective 3. Introduction to Blockchain Technology 4. Overview of Machine Learning in Fraud Detection 5. Significance	1
2	Problem Definition and Requirement 1. Problem Statement 2. Software Requirements 3. Hardware Requirements 4. Data Set	4
3	Proposed Design/Methodology 1. Project Directory 2. Methodology 3. Algorithms Used	8
4	Results 1. Data Preprocessing 2. Model Training 3. Model Evaluation 4. Result Analysis	11
5	References 1. Data Set 2. Study Material	28

1. INTRODUCTION

1.1 BACKGROUND:

In recent years, the demand for laptops has surged, driven by factors such as remote work, online learning, and digital entertainment. As a result, understanding the pricing dynamics of laptops has become crucial for both consumers and retailers. Traditional methods of pricing estimation often lack accuracy and fail to capture the nuanced relationship between various laptop specifications and their prices. Leveraging machine learning techniques can provide more precise and reliable predictions by analyzing large datasets and identifying patterns that influence laptop prices

1.2 OBJECTIVE:

The project's goal is to develop a precise machine learning model for laptop price prediction. It begins with compiling and preprocessing a comprehensive dataset of laptop attributes. Various machine learning algorithms are explored to find the most suitable for the task. Multiple models are trained and evaluated using metrics like mean absolute error and R-squared score to improve accuracy. Through iterative refinement, including parameter tuning and feature selection, the top-performing model is selected. Deployment in real-world scenarios involves ensuring the model's resilience, scalability, and adaptability to different data formats. Continuous monitoring and validation are essential to maintain accuracy and reliability over time. The project aims to provide valuable insights for consumers, retailers, and stakeholders in the consumer electronics industry by enabling accurate price prediction

1.3 INTRODUCTION TO LAPTOP PRICE PREDICTION TECHNOLOGY:

Laptop price prediction technology utilizes machine learning algorithms to forecast laptop prices accurately. By analyzing diverse attributes such as brand, specifications, and market trends, these algorithms offer valuable insights for consumers and retailers. The technology's emergence marks a significant advancement in pricing strategies within the consumer electronics industry. Its ability to discern complex patterns enables informed decision-making and enhances market competitiveness. This introduction provides a glimpse into the profound impact of laptop price prediction technology on market dynamics and consumer behavior.

1.4 OVERVIEW OF MACHINE LEARNING IN FRAUD DETECTION:

Machine learning revolutionizes laptop price prediction, leveraging advanced algorithms to analyze diverse data and offer precise forecasts. By considering brand, specifications, and market trends, these models empower consumers and retailers with valuable insights, shaping pricing strategies and enhancing competitiveness in the market.

1. Algorithm Utilization: Machine learning serves as the backbone of laptop price prediction, employing a spectrum of algorithms to sift through extensive datasets and uncover meaningful patterns. These algorithms range from conventional regression techniques to more sophisticated ensemble methods, each offering unique advantages in modeling price trends.

2. Consideration of Attributes: Central to the efficacy of machine learning in this context is the comprehensive consideration of laptop attributes. Factors such as brand reputation, technical specifications, market trends, and consumer preferences are meticulously analyzed to inform the predictive models. This holistic approach ensures that the resulting price forecasts are grounded in a nuanced understanding of the factors influencing market dynamics.

3. Feature Engineering: Feature engineering is paramount in extracting relevant insights from raw data. By carefully selecting and transforming input variables, machine learning models can better capture the underlying relationships between different attributes and their impact on laptop prices. This process is instrumental in enhancing the predictive accuracy of the models.

4. Model Optimization: Model selection and hyperparameter tuning are essential steps in optimizing the performance of machine learning models for laptop price prediction. Through rigorous experimentation and validation, the most suitable algorithms are identified, and their parameters are fine-tuned to achieve optimal predictive outcomes. This iterative refinement process is key to ensuring that the models can effectively generalize to unseen data..

5. Evaluation Metrics: Evaluation metrics such as mean absolute error and R-squared score are employed to assess the accuracy and reliability of the machine learning models.

1.5 SIGNIFICANCE:

Accurate price prediction of laptops has significant implications for both consumers and retailers. For consumers, it enables informed decision-making by providing insights into the value proposition of different laptop models based on their features and specifications. This helps consumers optimize their purchasing decisions according to their budget constraints and desired functionalities. For retailers, accurate price prediction facilitates dynamic pricing strategies based on market demand, competitor analysis, and inventory management. By aligning prices with consumer preferences and market trends, retailers can enhance their competitiveness, improve sales performance, and maximize profitability.

2. PROBLEM DEFINITION AND REQUIREMENT

2.1 PROBLEM STATEMENT:

In the realm of consumer electronics, accurately predicting laptop prices presents a significant challenge due to the intricate interplay of various factors influencing market dynamics. With a multitude of laptop models offering diverse specifications and features, coupled with fluctuating consumer preferences and technological advancements, the task of determining optimal pricing strategies becomes increasingly complex. The primary challenge at hand is to develop a robust predictive model capable of forecasting laptop prices with a high degree of accuracy. This entails leveraging machine learning algorithms to analyze a plethora of input features, including but not limited to processor type, RAM size, storage capacity, display resolution, brand reputation, customer reviews, and market trends. The model must effectively capture the nuanced relationships between these features and their impact on laptop prices, while also adapting to evolving market conditions and consumer behavior.

2.2 SOFTWARE REQUIREMENT:

Our solution will be implemented using Python, a versatile and widely adopted programming language suitable for data analysis and machine learning tasks. In addition to Python, we will leverage several libraries to facilitate various aspects of our project:

1. **NumPy:** Utilised for efficient numerical operations and array manipulation.
2. **Matplotlib:** Employed for creating visualisations and graphs to analyse data.
3. **Seaborn:** Used to enhance the aesthetics of visualisations and statistical data exploration.
4. **Pandas:** Utilised for data manipulation and analysis, offering powerful data structures and functions.

5. **Scikit-learn (sklearn):** Utilised for implementing machine learning algorithms, model evaluation, and preprocessing.
6. **Imbalanced-learn (imblearn):** Used for handling imbalanced datasets, a common challenge in fraud detection.

Methodologies:-

1. **Logistic Regression:** Applied for binary classification tasks, effectively identifying patterns in categorical data.
2. **Decision Tree:** Employed for both classification and regression tasks, partitioning data into subsets based on feature values.
3. **Random Forest:** Utilised as an ensemble learning method, constructing multiple decision trees to improve accuracy and robustness.

These software components provide the essential framework for developing our fraud detection algorithms effectively. By leveraging these libraries, we can streamline the development process and ensure the scalability and reliability of our solution.

2.3 HARDWARE REQUIREMENT:

In terms of hardware, we require a computer with sufficient processing power and memory to support the computational demands of our algorithms.

While the specific hardware specifications may vary depending on the scale of the project and the size of the dataset, a standard computer with a multi-core processor and ample RAM should suffice for development and testing purposes.

2.4 DATA SET:

For training and evaluating our fraud detection algorithms, we will utilise the Ethereum fraud detection dataset sourced from Kaggle. This dataset contains label transaction data, including features such as sender address, receiver address, transaction amount, etc. By leveraging this dataset, we can train supervised learning models to differentiate between legitimate and fraudulent transactions based on their attributes and characteristics.

We've got everything we need to build a strong fraud detection system for Ethereum transactions. With the right software, hardware, and data, we're ready to create smart algorithms that can spot

and stop fraudulent activities. Our aim is to make Ethereum safer and more trustworthy for everyone who uses it.

3. PROPOSED DESIGN / METHODOLOGY

Our proposed design and methodology focus on utilising logistic regression, decision tree, and random forest algorithms to develop a fraud detection system tailored for Ethereum transactions. As we have only one file containing all the work and a CSV file containing the data, our approach will be streamlined and contained within this single file.

3.1 PROJECT DIRECTORY:

Laptop_price_predictor_dataset.csv

Laptop_detection_system.py: This Python script contains all the code for data preprocessing, model training, evaluation, and deployment. It encompasses the entire workflow of our laptop detection system. **Laptop_price_dataset.csv:** The laptop price detection dataset sourced from Kaggle, containing transaction data.

3.2 METHODOLOGY:

Data Preprocessing:

- Load the Ethereum fraud detection dataset (laptop_price_predictor_dataset.csv) into memory.
- Perform data cleaning, handling missing values, and encoding categorical variables if necessary.
- Split the dataset into training and testing sets.

Model Training:

- Implement logistic regression, decision tree, and random forest algorithms using scikit-learn.
- Train each model on the training dataset.

Model Evaluation:

- Evaluate the performance of each model using metrics such as accuracy, precision, recall, and F1-score on the testing dataset.
- Compare the performance of logistic regression, decision tree, and random forest algorithms to identify the most effective approach for fraud detection

3.3 ALGORITHM USED:

Linear Regression :

Linear regression is a statistical algorithm used in predictive analysis to establish a linear relationship between an independent variable (predictor) and a dependent variable (outcome). It predicts continuous outcomes like sales, salary, or product price based on fluctuations in the independent variable.

Decision Tree: Decision tree classification is a method in machine learning where data is split into smaller subsets based on feature values, creating a tree-like structure of decision rules for predicting the class of new data points.

Random Forest: Random forest is an ensemble learning technique that constructs multiple decision trees during training and outputs the mode of the classes (for classification tasks) of the individual trees. It is known for its robustness and ability to handle large datasets with high dimensionality.

KNN algorithm : The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping

of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

Lasso Regression:

By controlling the regularization parameter (λ), it can eliminate irrelevant features, avoiding multicollinearity and overfitting. Lasso regression is particularly useful for high-dimensional data and enhances model interpretability compared to other regularization techniques like ridge regression. It's also applicable in logistic regression for similar regularization benefits.

4. RESULTS

In this section, we present the detailed results of our fraud detection system using linear regression, decision tree, and random forest algorithms, SVM, lasso. We outline the steps taken in data preprocessing, discuss the handling of missing values, class imbalance correction, and the creation of correlation matrices. Then, we delve into the model training process and provide comprehensive evaluations of each model's performance on the Laptop price detection dataset.

4.1 Data Preprocessing:

Handling Missing Values:

- We addressed missing values in the dataset using appropriate techniques such as imputation or deletion. The goal was to ensure that the dataset was complete and suitable for model training.
- We created a heat map to visualise the missing values in the dataset.
-
- The null values are indicated by the red colour, to handle these values we dropped the null values using the `dropna()` function.
- We then again visualised the dataset with the help of a heat map, to check whether all the missing values have been handled.

Dropping Features with zero Variance:

- Dropping features with zero variance involves removing variables from a dataset that have the same value for all observations. This means that these features do not provide any useful information for predicting or explaining the target variable. By eliminating these constant features, we can simplify the dataset and improve the efficiency of machine learning algorithms.
- We listed the features that had zero variance and dropped them.

Bar plot to visualise the distribution of transactions categorised as normal versus fraudulent:

- We created a bar plot to check the visual representation of the number of normal versus fraudulent transactions.

Handling Skewness:

- Skewness in the data was corrected to ensure that the distribution of features was approximately symmetrical, which is beneficial for the performance of machine learning algorithms.
- We first checked the skewness of features using the skew() function.
- After identifying the features with high skewness we applied logarithmic transformations on those features to reduce their skewness, and listed the skewness of the features again to verify.

Graphical representation of randomly picked feature showing skewness before the logarithmic transformation:

Graphical representation of randomly picked feature showing skewness after the logarithmic transformation:

Creating Correlation Matrix:

- We created a correlation matrix to identify relationships between features and determine which features were most strongly correlated with fraudulent transactions. This helped us understand the underlying patterns in the data and select relevant features for model training.
- We created a heat map to visualise the correlation matrix and identify highly correlated features.
-
- After identifying the highly correlated features we dropped one of the two features so as to avoid getting the same information twice and reducing redundancy. This ensures that our analysis remains clear, accurate, and efficient.

Handling Class Imbalance:

- Class imbalance was addressed by employing techniques such as oversampling or undersampling to ensure that both fraudulent and legitimate transactions were adequately represented in the training data.
 - We used SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority class.
 - Another method that we used to address class imbalance was hyper parameter tuning in all the models that we used.
- We set the “class_weights” parameter as balanced to add appropriate weights to each class.

4.2 Model Training:

Lasso Regression:

Lasso regression, or L1 regularization, applies a penalty to prevent overfitting in linear regression models. Introduced by Tibshirani in 1996, it shrinks coefficients towards zero, promoting sparsity and automatic feature selection. By controlling the regularization parameter (λ), it can eliminate irrelevant features, avoiding multicollinearity and overfitting. Lasso regression is particularly useful for high-dimensional data and enhances model interpretability compared to

other regularization techniques like ridge regression. It's also applicable in logistic regression for similar regularization benefits.

:

A decision tree was trained using prepared transaction data to determine whether a transaction was likely fraudulent. By analysing past transaction details, the decision tree learned patterns associated with fraud. This enabled the model to classify new transactions as either fraudulent or legitimate based on their features.

Decision Tree :

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model. 4.3 Model Evaluation:

Linear Regression :

Linear regression is a statistical algorithm used in predictive analysis to establish a linear relationship between an independent variable (predictor) and a dependent variable (outcome). It predicts continuous outcomes like sales, salary, or product price based on fluctuations in the independent variable. This supervised learning method is advantageous when multiple variables are available, as seen in stock market forecasting and scientific analysis. The regression model utilizes a sloped straight line to represent the relationship between variables, facilitating prediction and analysis in various fields.

KNN algorithm : The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

Random forest: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Trees in the forest use the best split strategy, i.e. equivalent to passing `splitter="best"` to the underlying `DecisionTreeRegressor`. The sub-sample size is controlled with

the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Support Vector machine: A support vector machine (SVM) is defined as a machine learning algorithm that uses supervised learning models to solve complex classification, regression, and outlier detection problems by performing optimal data transformations that determine boundaries between data points based on predefined classes, labels, or outputs. This article explains the fundamentals of SVMs, their working, types, and a few real-world examples.

4.4 Results Analysis:

In the analysis, we'll evaluate the performance of different machine learning models—linear regression, SVM, decision trees, and random forest—for predicting laptop prices. We'll compare their accuracy, precision, and recall scores to determine the most effective model. Additionally, we'll assess factors like computational efficiency and model complexity to identify the most suitable approach for our price prediction project. This comprehensive analysis will guide us in selecting the optimal model for accurately forecasting laptop prices.

In our laptop price predictor project, we conducted a comparative analysis of several machine learning models, including linear regression, SVM, decision trees, and random forest. Through this analysis, we assessed various performance metrics such as mean absolute error, mean squared error, and R-squared values. Additionally, we scrutinized the models' robustness to outliers, scalability, and interpretability. By delving into these aspects, we aim to pinpoint the model that offers the best balance of accuracy, generalization ability, and computational efficiency for predicting laptop prices reliably.

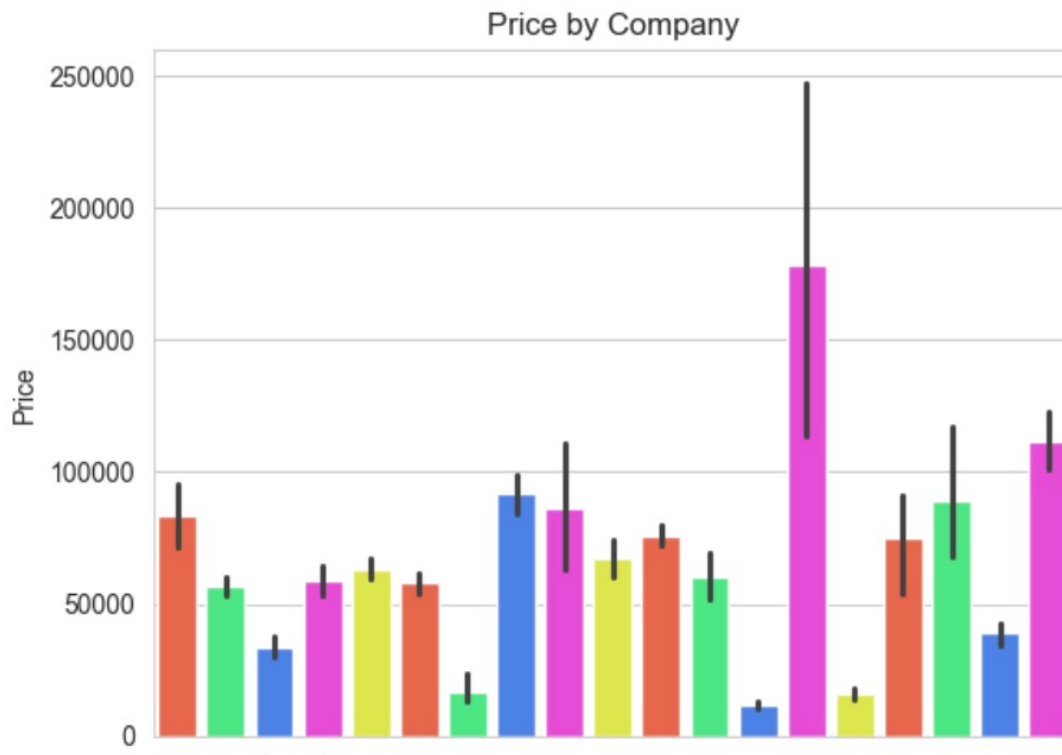


Fig 1. represents the distribution of prices within the dataset. The x-axis typically represents the range of prices, while the y-axis represents the frequency or density of occurrence of those prices.

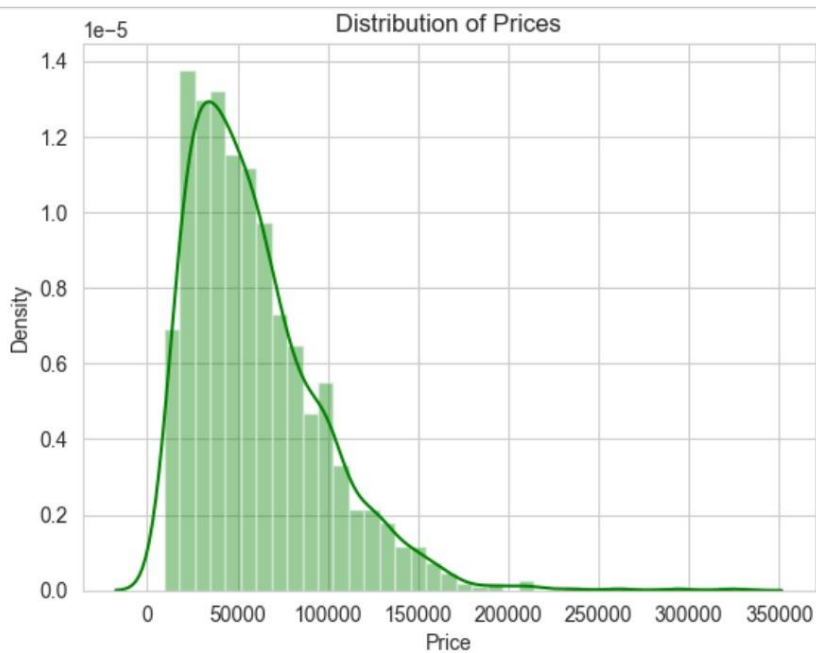


Fig 2. Represents the resulting plot will have bars representing each unique company, and the height of each bar indicates how many times that company appears in the 'Company' column of the DataFrame df.

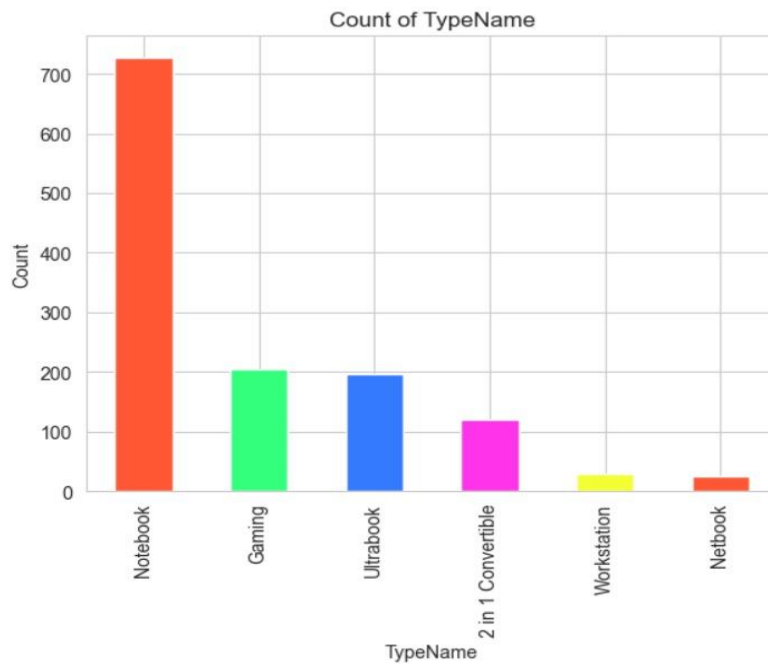


Fig 3.graph will have bars for each unique company along the x-axis, and the height of each bar represents the average price associated with that company. This visualization helps in comparing the average prices across different companies in your dataset

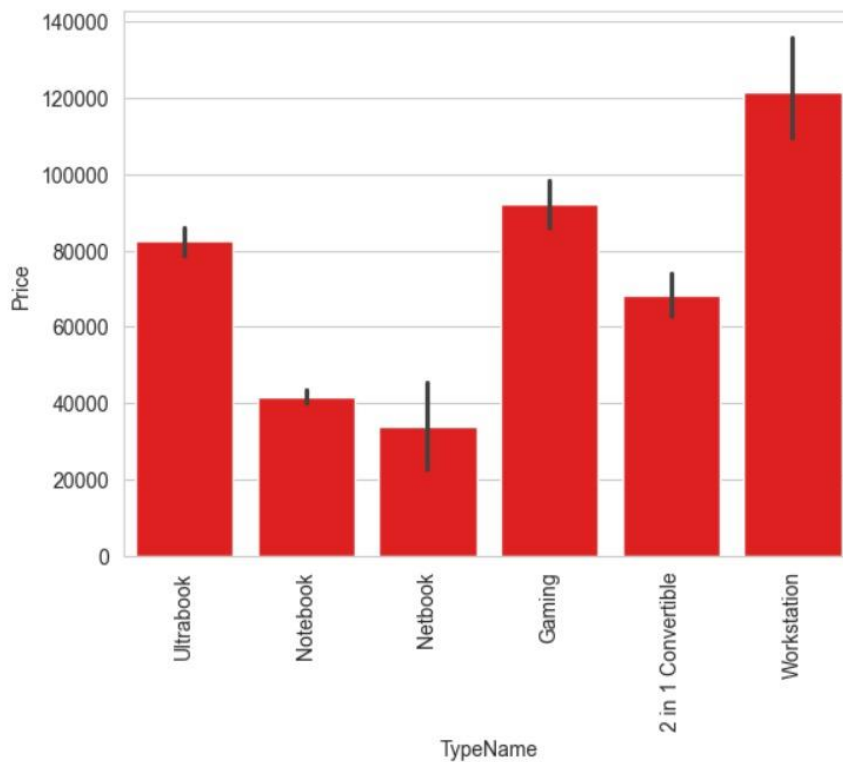


Fig 4. Represents the resulting plot will have bars representing each unique type, and the height of each bar indicates how many times that type appears in the 'TypeName' column of the DataFrame df

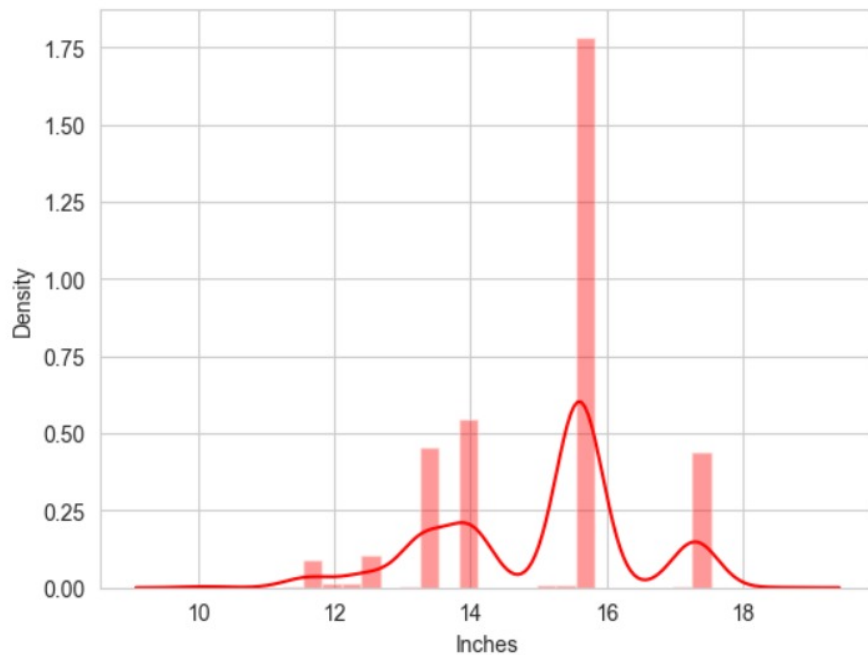


Fig 5. graph have bars for each unique type along the x-axis, and the height of each bar represents the average price associated with that type. This visualization helps in comparing the average prices across different types in your dataset.

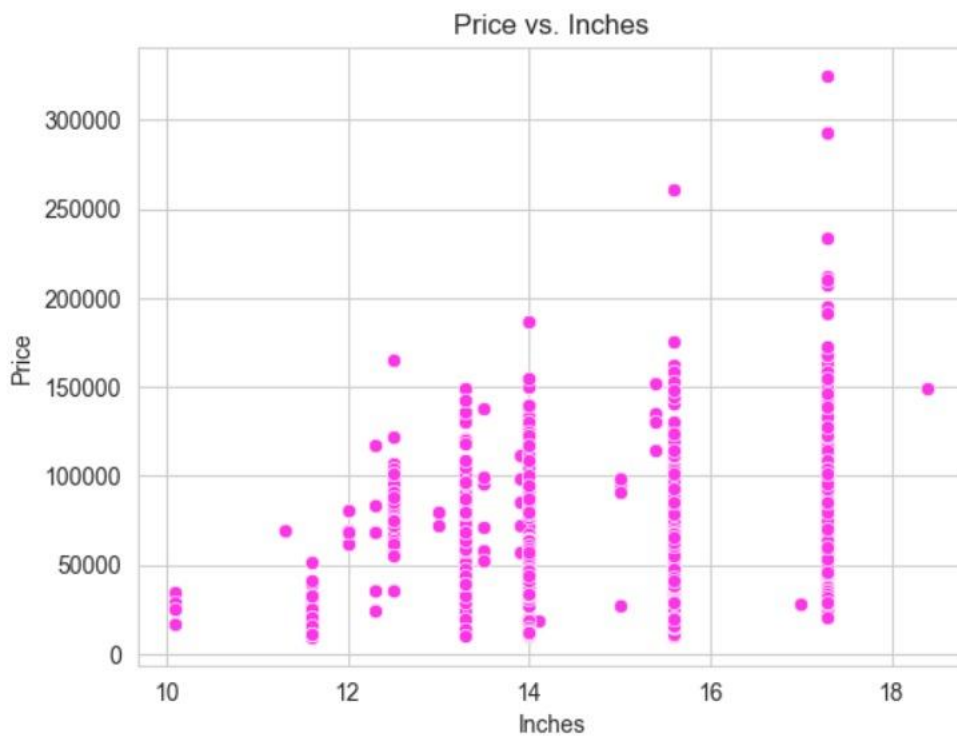


Fig 6. This visualization helps in understanding the distribution of screen sizes in the dataset, including the central tendency, spread, and any patterns or outliers present in the data.

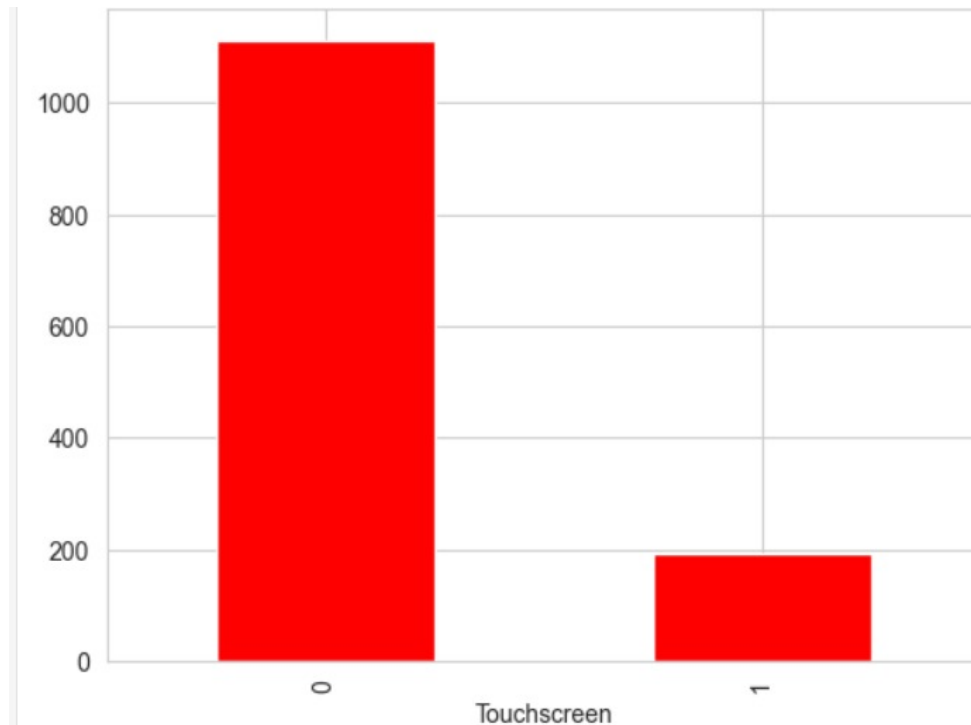


Fig 7. Display a scatter plot where each point represents a data entry from your DataFrame. The x-axis will represent the 'Inches' column, likely indicating screen size in inches, and the y-axis will represent the 'Price' column, indicating the price of devices

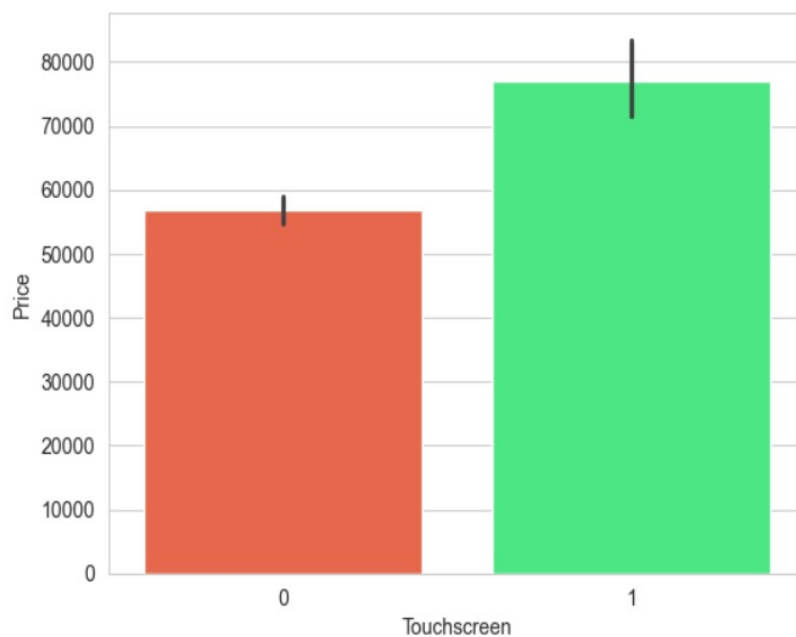


Fig 8. This visualization helps in understanding the distribution of touchscreen and non-touchscreen devices in your dataset

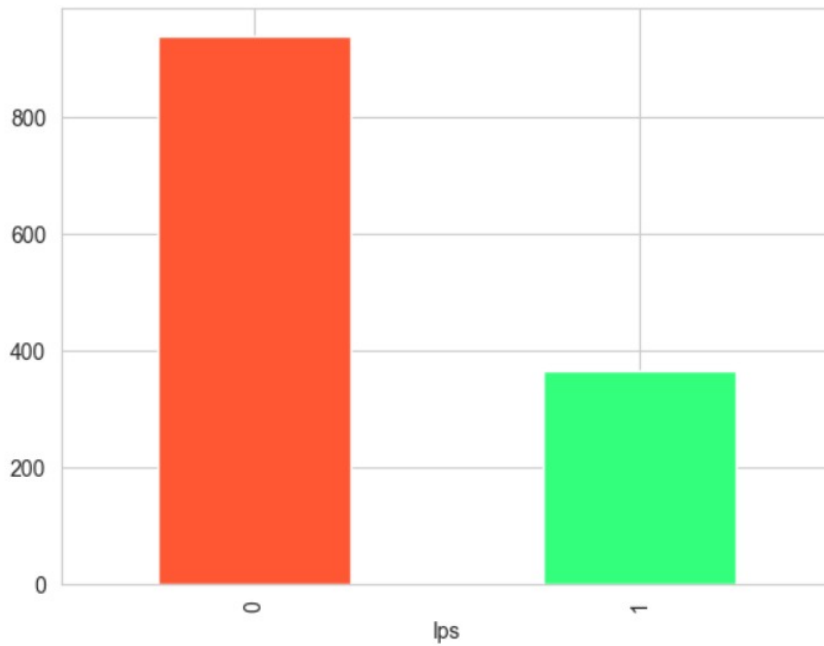


Fig 8. the resulting graph will have bars for both 'Touchscreen' categories ('True' and 'False') along the x-axis, and the height of each bar represents the average price associated with that category. This visualization helps in comparing the average prices between touchscreen and non-touchscreen devices in your dataset.

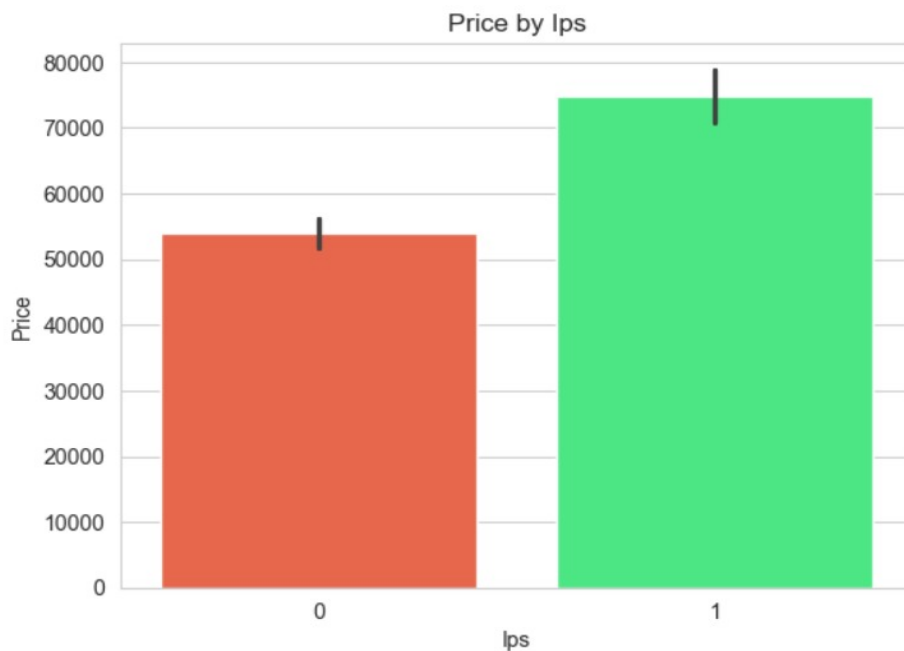


Fig 9. This visualization helps in understanding the distribution of IPS (In-plane switching) and non-IPS displays in your dataset.

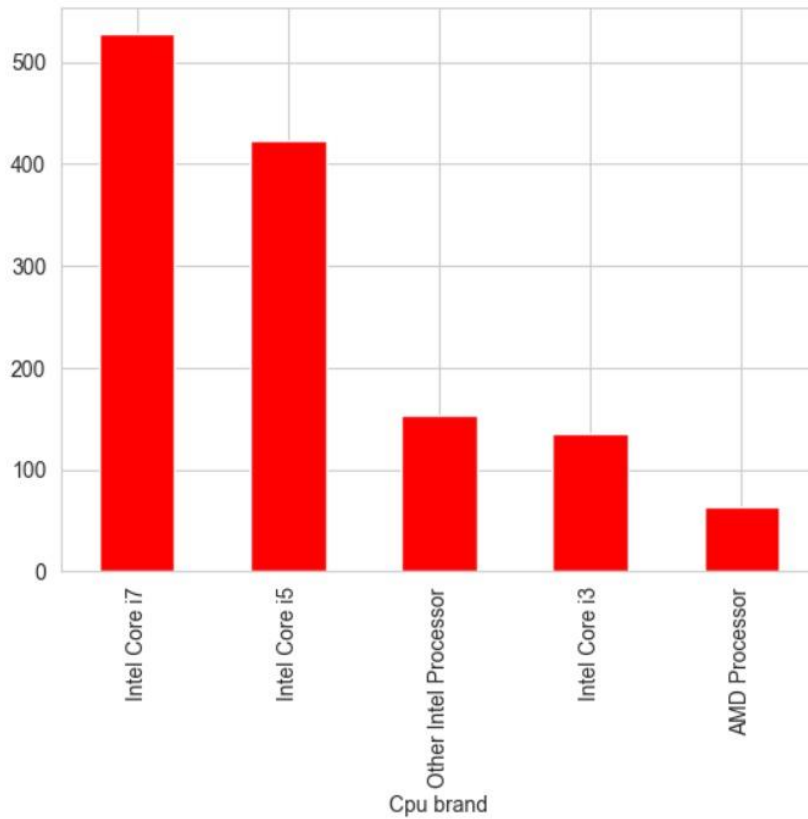


Fig 10.. This visualization helps in understanding the distribution of CPU brands in your dataset and identifying the prevalence of different brands

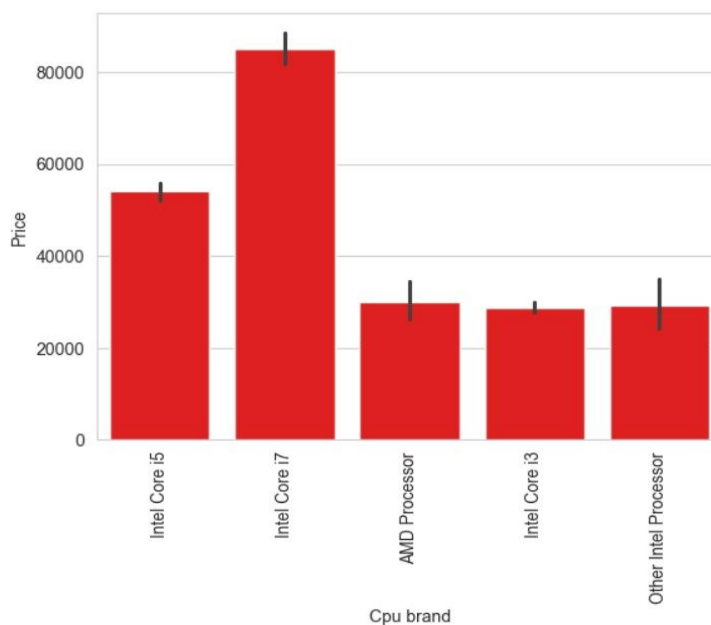


Fig 11. So, the resulting graph will have bars for each unique CPU brand along the x-axis, and the height of each bar represents the average price associated with that brand. This visualization helps in comparing the average prices across different CPU brands in your dataset.

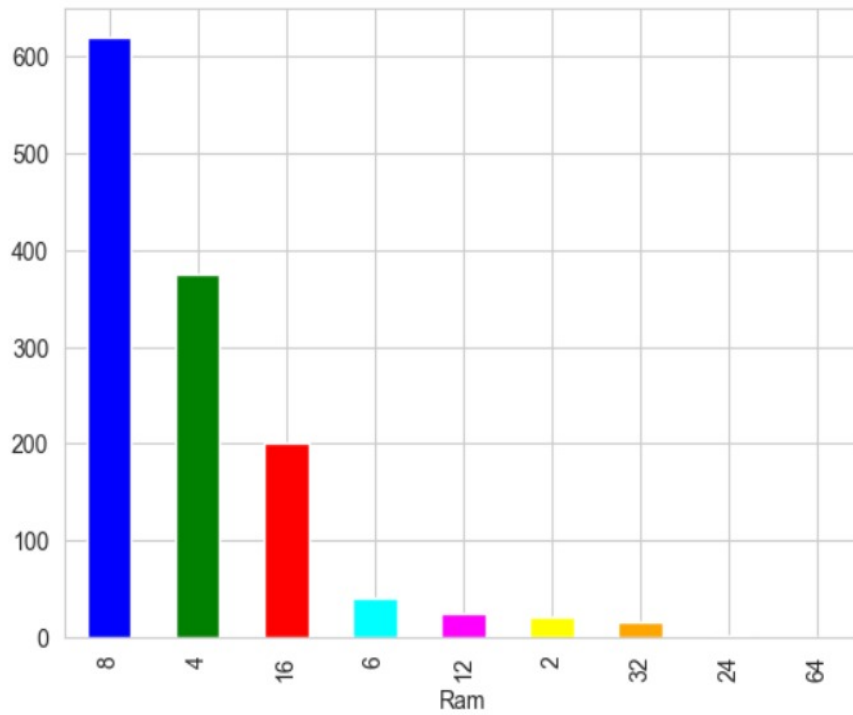


Fig 12. a bar plot to visualize the frequency of each unique value in the 'Ram' column of the DataFrame df.

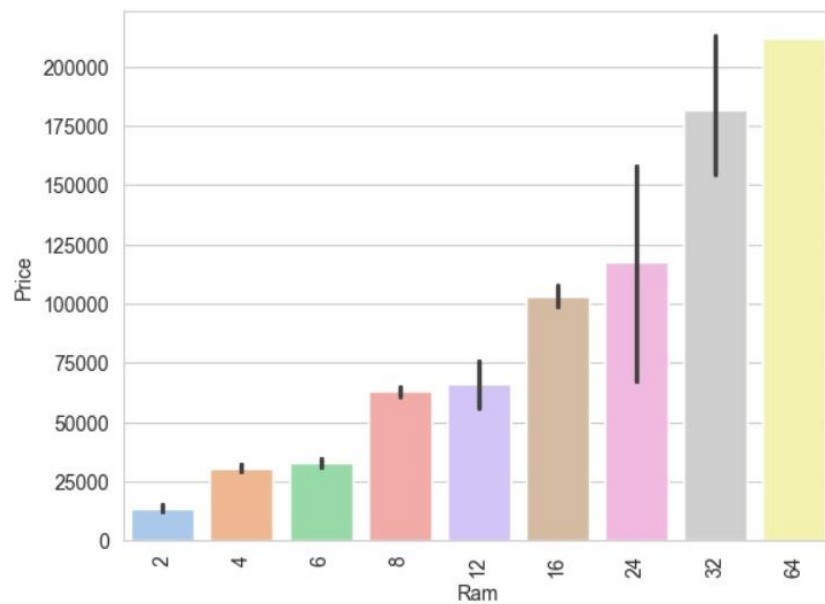


Fig 13. This visualization helps in comparing the average prices across different RAM configurations in your dataset.

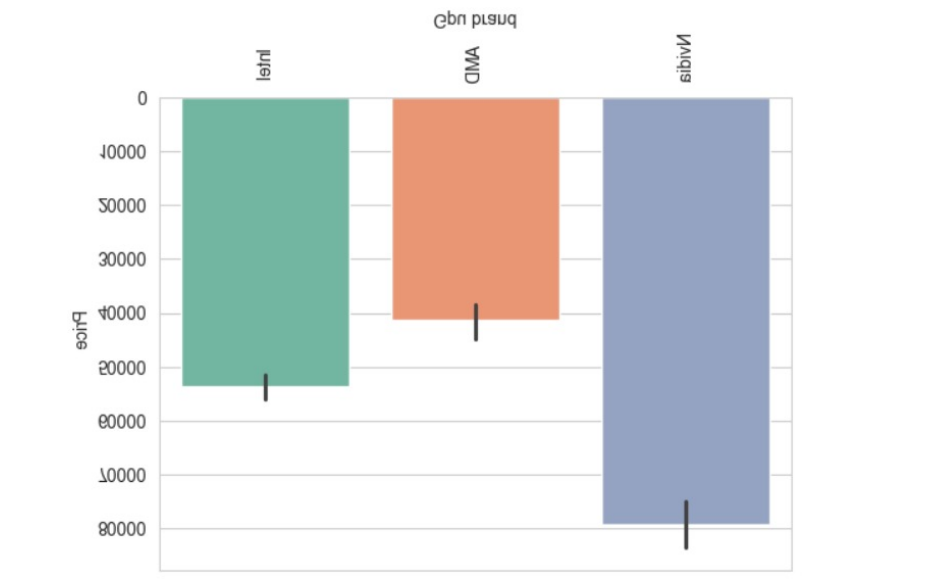


Fig 14. This visualization helps in comparing the average prices across different GPU brands in your dataset.

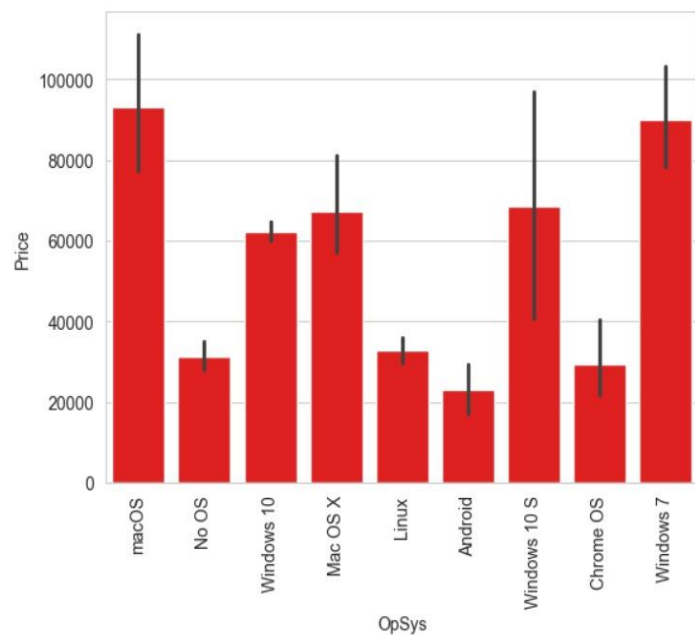


Fig 15. This visualization helps in comparing the average prices across different operating systems in your dataset..

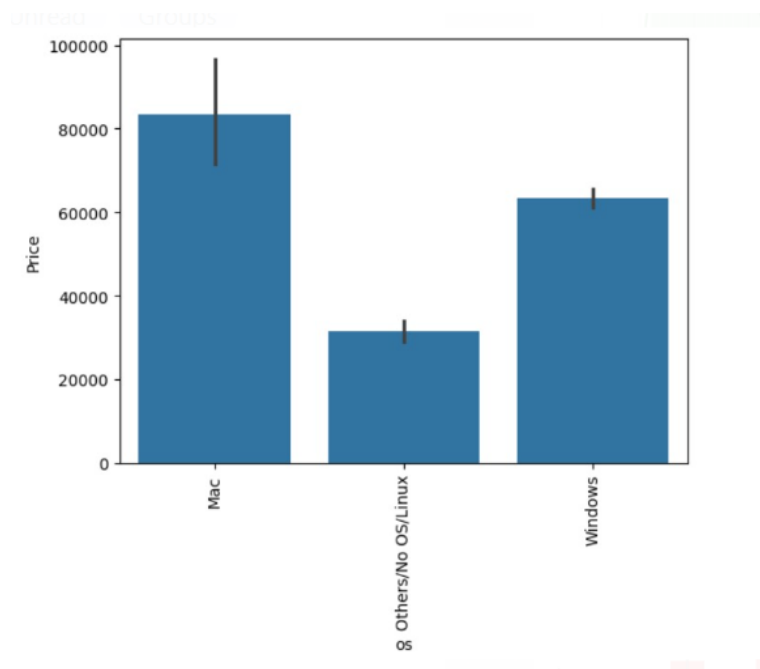


Fig 16. This visualization helps in comparing the average prices across different operating systems in your dataset.

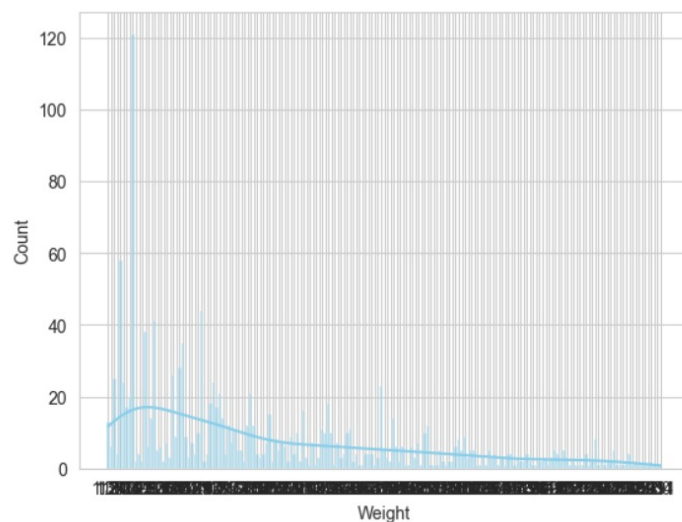


Fig 17. This visualization helps in understanding the distribution of weights in your dataset, including the central tendency, spread, and shape of the weight distribution. It can also help in identifying any outliers or patterns in the data.



Fig 18. Overall, the correlation heatmap serves as a visual aid to understand the relationships between variables in your dataset and to guide further analysis

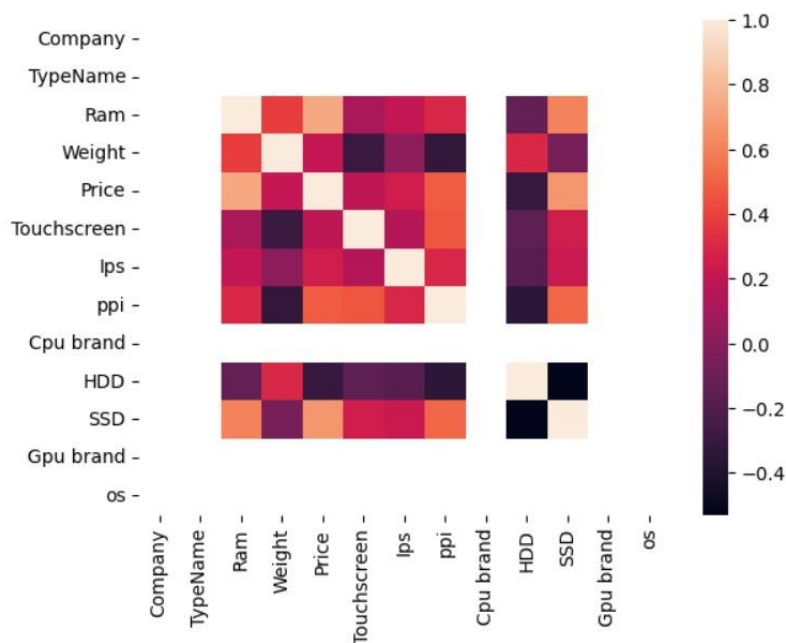


Fig 19. Overall, the correlation heatmap serves as a visual aid to understand the relationships between variables in your dataset and to guide further analysis

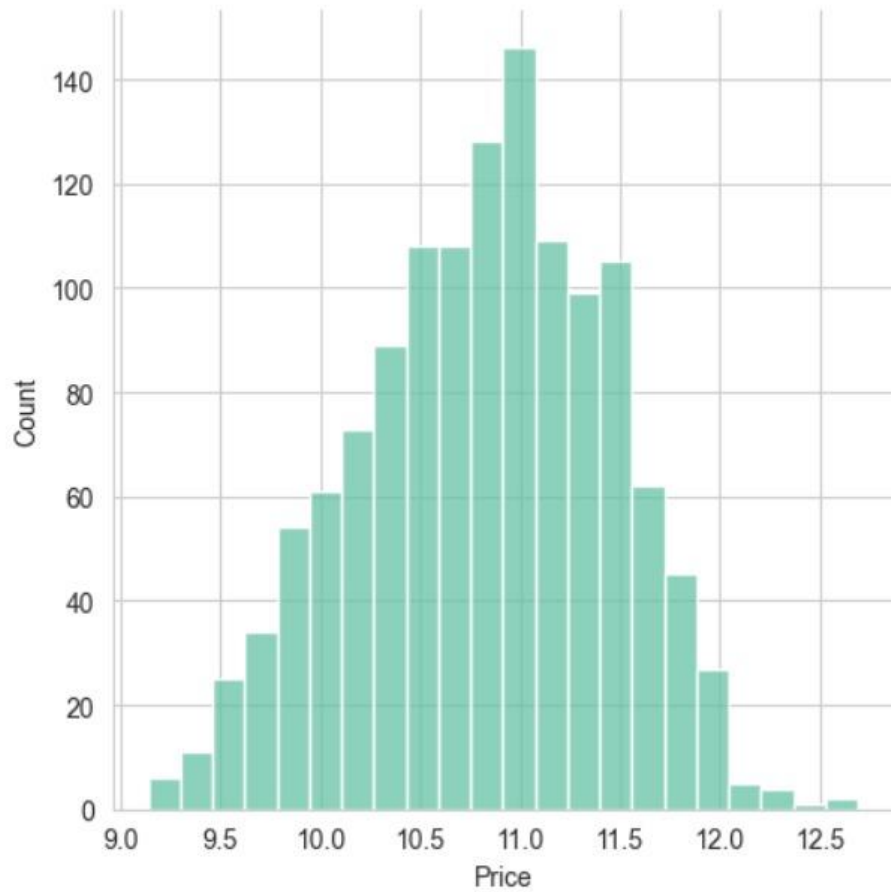
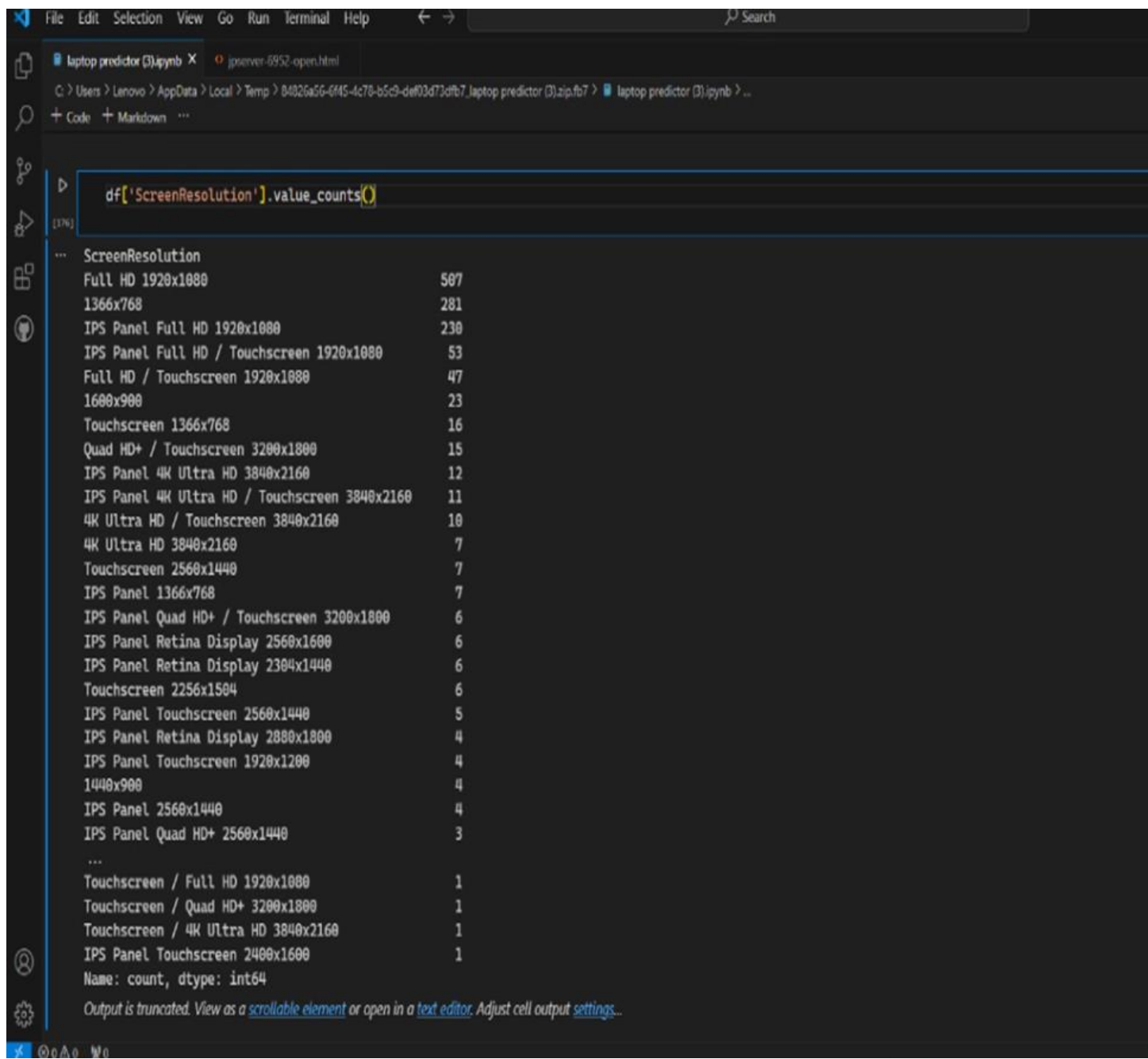


Fig 20. This visualization helps in understanding the distribution of the logarithm of prices in your dataset, which might be useful for certain statistical analyses or for visualizing data that has a wide range of values



The screenshot shows a Jupyter Notebook interface with a code cell containing the command `df['ScreenResolution'].value_counts()`. The output is a pandas Series showing the frequency of each screen resolution. The resolutions are listed on the left, and their corresponding counts are on the right. The output is truncated at the bottom, indicating there are more results.

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

ScreenResolution	count
Full HD 1920x1080	507
1366x768	281
IPS Panel Full HD 1920x1080	230
IPS Panel Full HD / Touchscreen 1920x1080	53
Full HD / Touchscreen 1920x1080	47
1600x900	23
Touchscreen 1366x768	16
Quad HD+ / Touchscreen 3200x1800	15
IPS Panel 4K Ultra HD 3840x2160	12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160	11
4K Ultra HD / Touchscreen 3840x2160	10
4K Ultra HD 3840x2160	7
Touchscreen 2560x1440	7
IPS Panel 1366x768	7
IPS Panel Quad HD+ / Touchscreen 3200x1800	6
IPS Panel Retina Display 2560x1600	6
IPS Panel Retina Display 2304x1440	6
Touchscreen 2256x1504	6
IPS Panel Touchscreen 2560x1440	5
IPS Panel Retina Display 2880x1800	4
IPS Panel Touchscreen 1920x1200	4
1440x900	4
IPS Panel 2560x1440	4
IPS Panel Quad HD+ 2560x1440	3
...	
Touchscreen / Full HD 1920x1080	1
Touchscreen / Quad HD+ 3200x1800	1
Touchscreen / 4K Ultra HD 3840x2160	1
IPS Panel Touchscreen 2400x1600	1

Name: count, dtype: int64

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

Fig. The screenshot of code snippet displayed above will output the counts of unique screen resolutions present in the 'ScreenResolution' column of your DataFrame df

5. REFERENCES

5.1 Data Set:

<https://www.kaggle.com/datasets/mohidabdulrehman/laptop-price-dataset>

5.2 Study Material:

<https://www.analyticsvidhya.com/blog/2021/11/laptop-price-prediction-practical-understanding-of-machine-learning-project-lifecycle/>

https://www.researchgate.net/publication/377721653_Machine_Learning-Based_Price_Prediction_for_Laptops

<https://youtu.be/BgpM2IiCH6k?si=neGsObQVmONG0Ht>