A PROJECT REPORT

of

Big Data Analysis

(Laptop Price Prediction)

Submitted by:

Ayush Gurjar (220466)

Jatin Singh (220347)

Abhinav (220619)

Himanshu (220593)

Vishal Yadav (220621)

under mentorship of

Dr. Meenakshi Malik

Department of Computer Science Engineering
School of Engineering and Technology
BML MUNJAL UNIVERSITY, GURUGRAM (INDIA)

November 2024

INDEX

CANDIDATES DECLARATION

We hereby certify that this report, named **Laptop Price Prediction** is our own work, thatit has not been submitted in whole or in part for any other degree or qualification at this university or any other institution and that all references have been properly cited. We take full responsibility for the integrity and correctness of the data included in this report.

Ayush Gurjar (220466)
Jatin Singh (220347)
Abhinav (220619)
Himanshu (220593)
Vishal Yadav(220621)

SUPERVISOR DECLARATION

This report, named **Laptop Price Prediction** was prepared under my supervision, and I Dr. Meenakshi Malik certify that the candidate conducted the most of the work on it. I certify that the right kind of supervision was given in compliance with the rules of the institution.

Dr. Meenakshi Malik

ACKNOWLEDGEMENT

We would like to take this opportunity to express our heartfelt gratitude to all those who contributed to the completion of this report. Firstly, we extend our deepest appreciation to our faculty mentor, **Dr. Meenakshi Malik**, for her exceptional guidance, motivating ideas, and unwavering support throughout the development of this project and the drafting of this report. Her insights and feedback have been invaluable in shaping the direction of this project.

We would also like to thank all the individuals who participated, contributed, and will continue to contribute to this project. Their advice and assistance have been instrumental in ensuring its success, and we are truly grateful for their ongoing support.

Furthermore, we wish to extend our appreciation to our colleagues and peers for their constructive feedback, discussions, and assistance during various stages of this project. Their input has enriched the quality of this report and contributed to its refinement.

We are immensely grateful to all those who played a part in this journey. Any shortcomings or omissions in the report are solely our responsibility.

ABSTRACT

Accurately predicting laptop prices is essential for market analysis, strategic pricing, and informed consumer decision-making. This project leverages machine learning techniques to build a robust model for **laptop price prediction** based on features such as Brand, Processor, RAM, Storage, Operating System, and User Ratings. Data was collected via web scraping from Flipkart, encompassing real-world laptop specifications and pricing information. The dataset was meticulously cleaned and pre-processed to ensure quality and reliability for subsequent analysis.

Comprehensive Exploratory Data Analysis (EDA) was conducted to uncover patterns and relationships within the data, such as the influence of brand and specifications on pricing. Multiple machine learning algorithms, including **Lasso Regression, KNN Regression, Decision Tree, Support Vector Regression and Random Forest** were applied and evaluated using performance metrics like R-squared and Mean Absolute Error (MAE). Random Forest demonstrated superior performance, making it the most reliable model for this application.

To enhance usability, a web application was developed using Flask, enabling users to input specifications manually and receive real-time price predictions. This project showcases the application of machine learning in addressing complex pricing challenges and offers a scalable solution for retailers and consumers alike. Future improvements may involve incorporating additional features, expanding the dataset, and integrating real-time data updates for even greater accuracy and relevance.

INTRODUCTION

The exponential growth in the electronics market, especially laptops, has brought about a wide variety of products catering to diverse consumer needs. Laptops differ significantly in features such as processing power, RAM, storage, screen size, and additional functionalities, leading to considerable variations in price. For consumers, understanding how these features influence pricing is crucial for making informed purchasing decisions. Similarly, manufacturers and retailers benefit from predictive models that help set competitive prices while maintaining profitability.

This project focuses on leveraging machine learning techniques to develop a reliable and efficient model for laptop price prediction. Using publicly available data sourced from Flipkart, a leading e-commerce platform, we gathered detailed information about laptop specifications, ratings, and prices. The aim was to explore the relationships between hardware attributes (e.g., Processor Brand, RAM Size, SSD Capacity) and prices, uncovering patterns that drive consumer preferences and pricing strategies in the market.

Laptop price prediction is a challenging task due to the dynamic nature of pricing and the non-linear relationships between features. Factors such as brand reputation, processor generation, and storage capacity exhibit varying degrees of influence on price. Machine learning provides a robust framework to model these complexities, offering insights into pricing trends and enabling accurate predictions. By implementing and evaluating multiple algorithms, this project identifies the most effective approach for price prediction and translates it into a practical application through a user-friendly web interface. This work bridges the gap between data-driven insights and real-world usability, benefiting both consumers and industry stakeholders.

LITERATURE REVIEW

Laptop price prediction has gained considerable attention in the field of data science due to its practical applications in market analysis and consumer decision-making. Existing studies have explored various machine learning approaches for predicting product prices, with a focus on handling large datasets and identifying significant features that influence pricing. Researchers have demonstrated the potential of algorithms such as Lasso Regression, KNN Regression, Decision Tree, Support Vector Regression and Random Forest in modelling the relationships between product specifications and their market prices.

Linear Regression is widely used for price prediction tasks due to its simplicity and interpretability. However, it struggles with capturing non-linear relationships between features, which are often prevalent in laptop pricing. Decision Trees, on the other hand, offer better flexibility and can handle non-linear relationships effectively, but they are prone to overfitting. Ensemble methods like Random Forest and Support Vector Regression address these issues by combining multiple decision trees to improve robustness and accuracy. Random Forest, in particular, has been highlighted for its ability to handle high-dimensional data and identify feature importance, making it a strong candidate for price prediction tasks.

Despite the advancements in machine learning, challenges remain in achieving high accuracy for laptop price prediction. Factors such as the dynamic nature of the market, variations in brand reputation, and the impact of external factors like discounts and promotions add complexity to the task. Additionally, integrating user ratings and reviews as features has shown promise in capturing consumer preferences, but it also introduces noise into the data. This project builds on these insights, leveraging the strengths of Random Forest while addressing challenges through data preprocessing, feature engineering, and model optimization techniques to deliver a robust and accurate prediction system.

PROBLEM STATEMENT

Laptop pricing is influenced by a variety of factors, including hardware specifications, brand reputation, operating system, and user feedback. Consumers often find it challenging to compare and evaluate laptops based on these attributes, while manufacturers and retailers need data-driven insights to set competitive and profitable prices. The absence of a reliable and accessible system for estimating laptop prices complicates decision-making for both buyers and sellers.

The goal of this project is to develop a robust machine learning model that predicts laptop prices based on key features such as processor type, RAM, storage capacity, brand, and user ratings. This involves addressing several challenges, including the non-linear relationships between features and prices, the dynamic nature of pricing in the electronics market, and the need for a user-friendly deployment platform.

To achieve this, the project encompasses the following objectives:

1. **Collecting real-world data through web scraping to ensure relevance and diversity.**

2. **Preprocessing the data to handle missing values, remove duplicates, and encode categorical variables.**

3. **Exploring and visualizing relationships between features and pricing through Exploratory Data Analysis (EDA).**

4. **Implementing and evaluating multiple machine learning algorithms to identify the most effective model.**

5. **Developing a web-based interface for real-time price prediction, enabling users to input specifications and receive instant estimates.**

By addressing these objectives, this project aims to bridge the gap between data analysis and practical application, providing stakeholders with a reliable tool for price prediction and decision-making.

METHODOLOGY

The methodology for this project is divided into five major phases, encompassing data collection, preprocessing, exploratory data analysis, model implementation, and deployment. Each phase contributes to building a robust machine learning pipeline for accurate laptop price prediction.

1. Data Collection -

Data was collected through web scraping from Flipkart, one of India's largest e-commerce platforms. A Python-based web scraper was used to extract laptop details such as brand, model, processor, RAM, storage, operating system, user ratings, reviews, and prices. This ensured the dataset represented real-world scenarios and included diverse laptop configurations.

In [84]:
```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import time
```

```
        next_button.click()
        time.sleep(3)  # Wait for the next page to load
    except:
        print("No more pages or unable to find the Next button.")
        break

# Close the driver
driver.quit()

# Convert the list of products into a DataFrame
df = pd.DataFrame(products)

# Save the DataFrame to an Excel file
df.to_excel('flipkart_gaming_laptops.xlsx', index=False)

print("Data has been saved to 'flipkart_laptops.xlsx'.")
```

```
Product Name: ASUS TUF Gaming A15 AMD Ryzen 7 Octa Core 7435HS - (8 G...
Product Price: ₹58,990
Product Description: 15.6 inch, Graphite Black, 2.30 Kg
Product Rating: 4.1
No. of Product Reviews: (198)
--------------------------------------------------------------------------------
Product Name: ASUS TUF Gaming F17 - AI Powered Gaming Intel Core i5 1...
Product Price: ₹57,990
Product Description: 17.3 Inch, Graphite Black, 2.60 kg
Product Rating: 4.3
No. of Product Reviews: (5,955)
--------------------------------------------------------------------------------
Product Name: Acer Aspire 7 Intel Core i5 13th Gen 13420H - (16 GB/51...
Product Price: ₹61,990
Product Description: 15.6 Inch, Black, 1.99 Kg
Product Rating: 4.4
No. of Product Reviews: (245)
--------------------------------------------------------------------------------
Product Name: MSI Thin 15 Intel Core i5 12th Gen 12450H - (16 GB/512 ...
```

2. Dataset –

```
df = pd.read_csv("D:\\SEM 5\\BDA\\Laptop Analysis Project\\cleaned_file.csv")
df.tail()
```

Python

	Brand	Model	Processor	Generation	Operating System	RAM	Storage	Size	Warranty	Color	Star Rating	No. of Ratings	No. of Reviews	Price
8905	MSI	MSI Modern	Intel Core i5	13th Gen	Windows 11 Operating System	16 GB	512	39.62	2	Silver	4.3	1963	284	44990
8906	DELL	DELL Inspiron	Intel Core i3	12th Gen	Windows 11 Operating System	8 GB	512	39.62	1	Blue	4.2	1630	300	42490
8907	MSI	MSI Thin	AMD Ryzen 5	11th Gen	Windows 11 Operating System	8 GB	512	39.62	2	Black	4.3	1689	233	47990
8908	MSI	MSI Modern	Intel Core i5	12th Gen	Windows 11 Operating System	8 GB	512	35.56	1	Blue	4.3	764	112	36990
8909	Lenovo	Lenovo IdeaPad Slim	Intel Core i5	12th Gen	Windows 11 Operating System	16 GB	512	100.63	2	Black	4.2	463	286	36990

3. Data Preprocessing -

The raw data required substantial preprocessing to prepare it for analysis and model training:

- **Handling Missing Values**: Rows with incomplete or missing key attributes, such as price or processor details, were removed.
- **Removing Duplicates**: Duplicate entries were identified and eliminated to avoid bias.
- **Data Transformation**: Categorical features (e.g., brand, color) were encoded into numerical values using one-hot encoding or label encoding.
- **Standardization**: Features such as weight and screen size were standardized to a common scale.
- **Outlier Detection**: Extreme values in features like price were detected and either corrected or excluded to ensure model reliability.

```
# Calculate IQR (Interquartile Range)
Q1 = df['Price'].quantile(0.25)
Q3 = df['Price'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds to remove outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers from the Price column using the IQR method
df = df[(df['Price'] >= lower_bound) & (df['Price'] <= upper_bound)]

# Check the shape and info of the cleaned DataFrame
print(df.shape)
df.info()
```

```
(8210, 14)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8210 entries, 1 to 8909
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Brand             8210 non-null   object
 1   Model             8210 non-null   object
 2   Processor         8210 non-null   object
 3   Generation        8210 non-null   object
 4   Operating System  8210 non-null   object
 5   RAM               8210 non-null   int32
 6   Storage           8210 non-null   int64
 7   Size              8210 non-null   float64
 8   Warranty          8210 non-null   int64
 9   Color             8210 non-null   object
 10  Star Rating       8210 non-null   float64
 11  No. of Ratings    8210 non-null   int64
 12  No. of Reviews    8210 non-null   int64
 13  Price             8210 non-null   int64
dtypes: float64(2), int32(1), int64(5), object(6)
memory usage: 930.0+ KB
```

4. Exploratory Data Analysis (EDA) -

EDA was performed to gain insights into the relationships between features and their impact on price:

- **Correlation Analysis**: Examined correlations between features (e.g., RAM, processor generation, SSD capacity) and price to identify significant predictors.
- **Visualization**: Created graphs such as histograms, box plots, and scatter plots to visualize price distribution and feature relationships.
- **Insights**: Derived actionable insights, such as higher RAM and newer processor generations generally resulting in higher prices.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting Price Distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], bins=50, kde=True)
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='RAM', y='Price', data=df)
plt.title('Price vs RAM')
plt.xlabel('RAM')
plt.ylabel('Price')
plt.show()
```

```
plt.figure(figsize=(7,10))
sns.scatterplot(df, x='Price', y='Processor')
plt.show()
```

5. Model Implementation -

Several machine learning algorithms were implemented and evaluated to predict laptop prices:

- **Algorithms Used**:
- Lasso Regression
- KNN Regression
- Decision Tree
- Support Vector Regression
- Random Forest

```
# Create a pipeline with StandardScaler and Random Forest Regressor
pipeline = Pipeline(steps=[
    ('scaler', StandardScaler()),  # Standardize the features
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))  # Use Random Forest Regressor
])

# Fit the model on the training data
pipeline.fit(X_train, y_train)

# Make predictions on the test data
y_pred = pipeline.predict(X_test)

# Evaluate the model
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Print the evaluation metrics
print('R2 Score:', r2)
print('Mean Absolute Error (MAE):', mae)
print('Root Mean Squared Error (RMSE):', rmse)
```

- **Evaluation Metrics**:
  - R-squared (R^2): Measured the proportion of variance explained by the model.
  - Mean Absolute Error (MAE): Assessed the average deviation of predictions from actual prices.

- **Best Model**:
  Random Forest emerged as the best-performing algorithm due to its ability to handle non-linear relationships and complex feature interactions effectively. Hyperparameter tuning was performed using Grid Search Cross-Validation to optimize model performance.

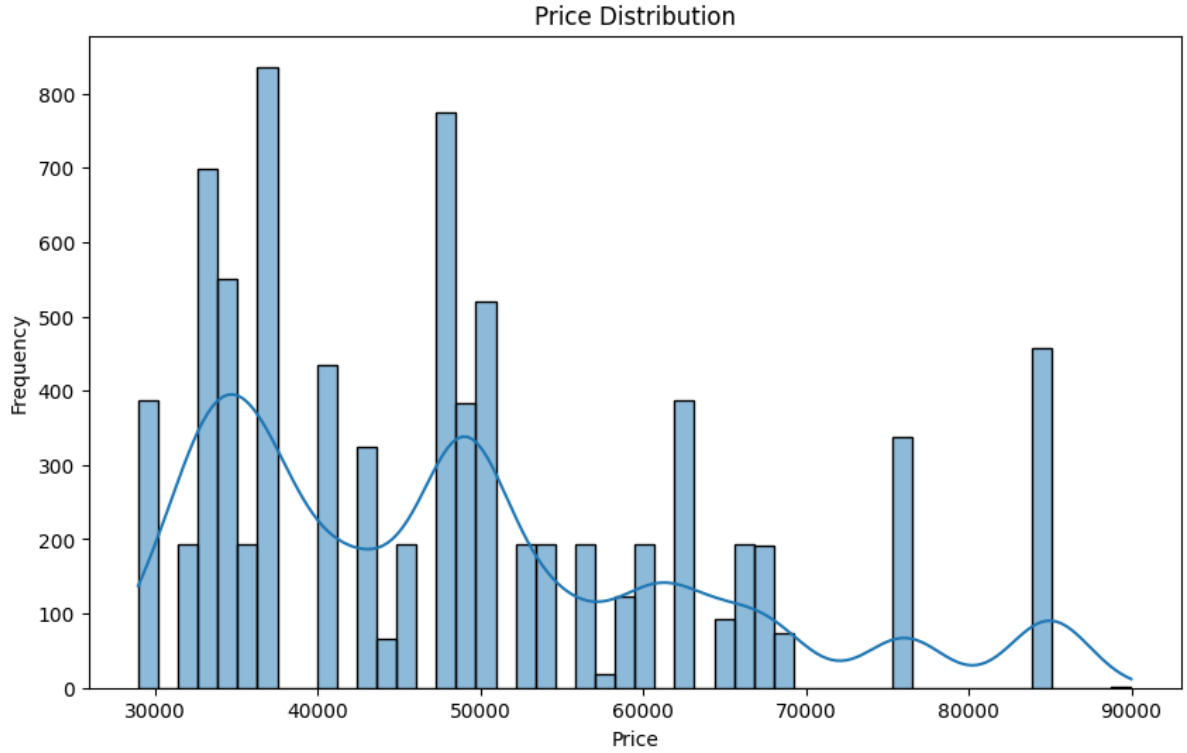
6. Deployment -

To make the model accessible, a web application was developed using Flask:

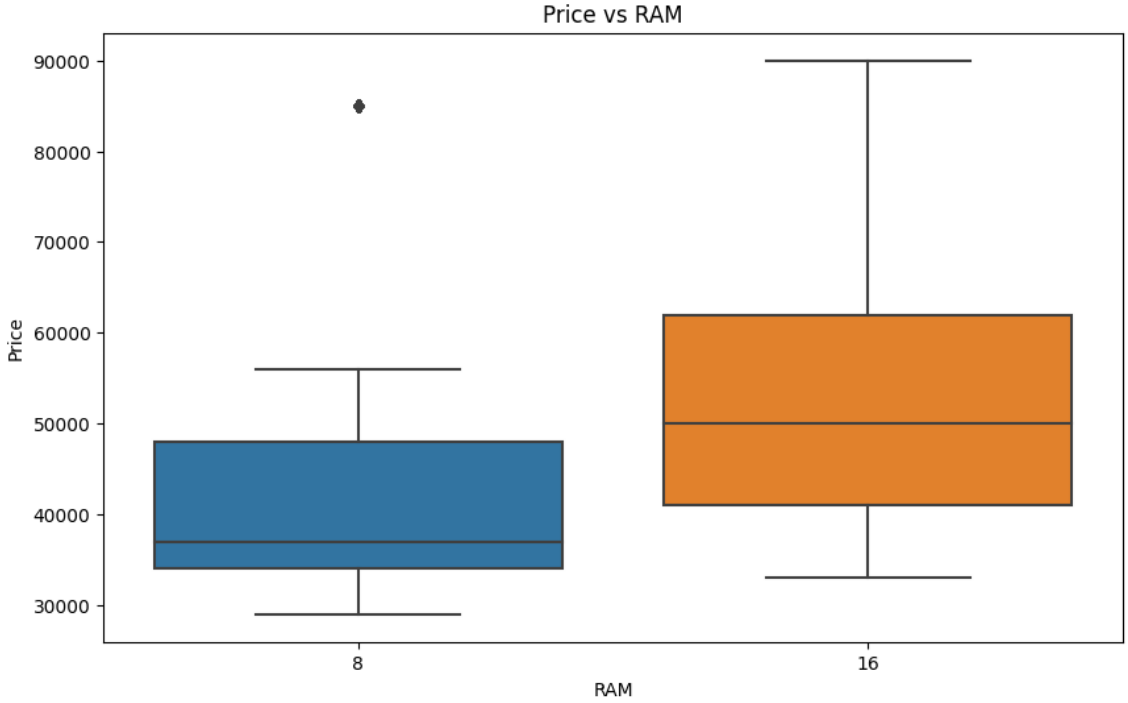
- **Interface Design**: The application enables users to input laptop specifications manually (e.g., RAM, processor, brand) through a user-friendly interface.
- **Prediction Engine**: The trained Random Forest model was integrated to provide real-time price predictions based on user inputs.
- **Testing and Validation**: The application was tested on sample inputs to ensure accuracy and usability.

EXPLORATORY DATA ANALYSIS

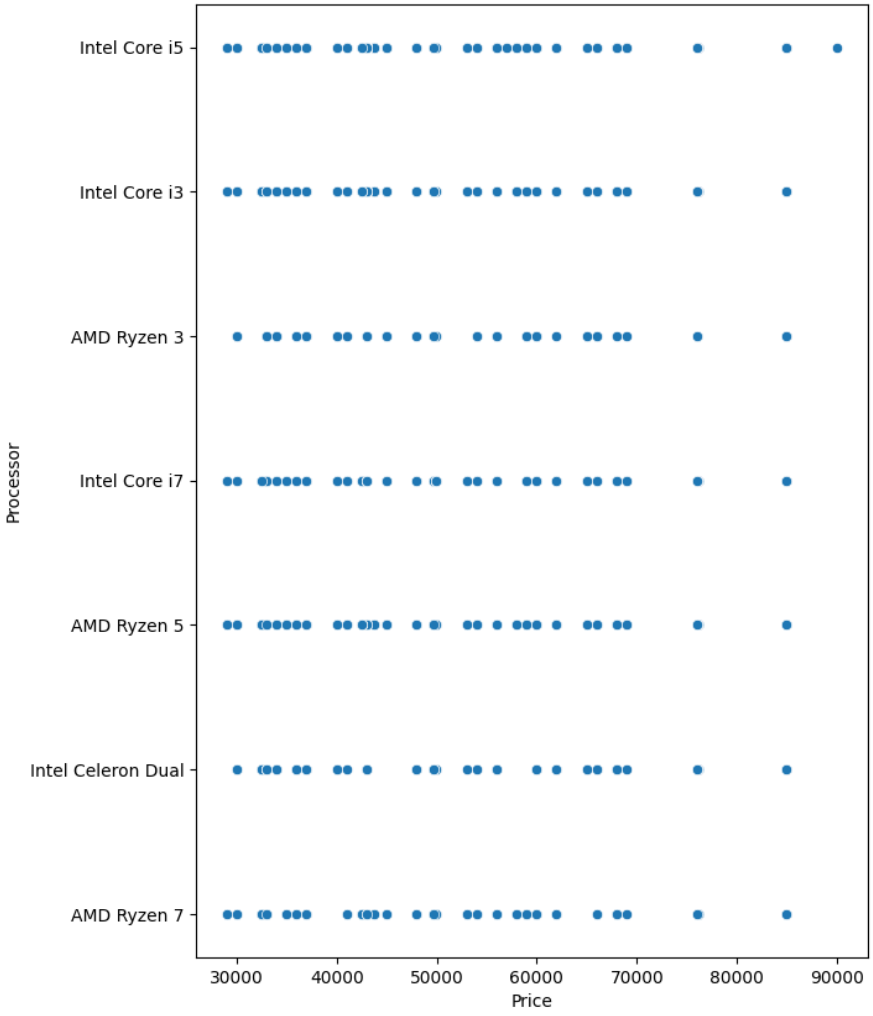
- **Price Distribution Graph -**



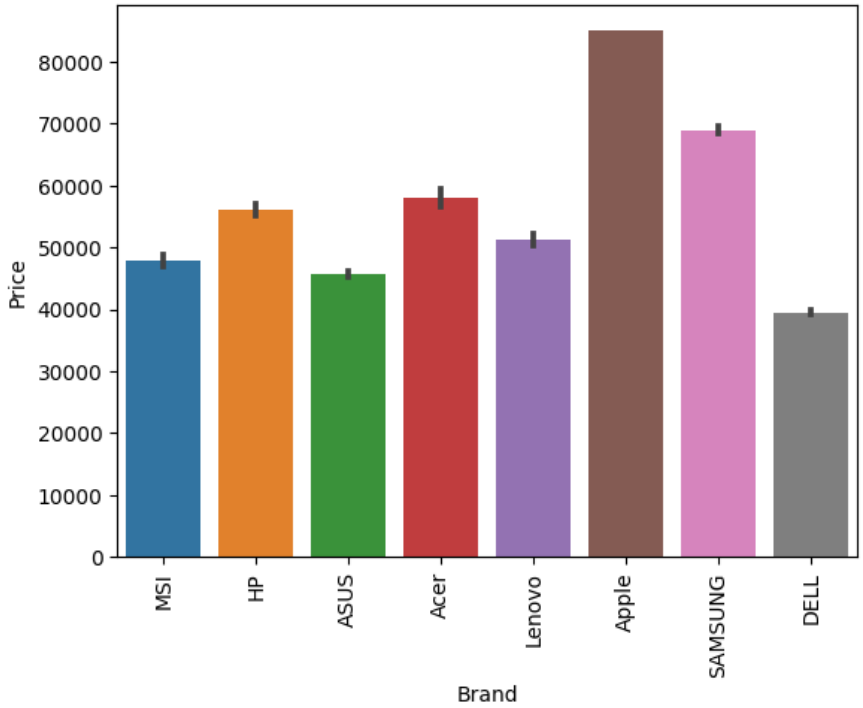
- **Price Vs RAM Graph -**



- **Price Vs Processor Graph-**



- **Brand Vs Price Graph -**



MODEL IMPLEMENTATION AND PERFORMANCE

- **Algorithms Tried:**

- **Lasso Regression:** Linear regression with L1 regularization to shrink coefficients and select features.
- **KNN Regression:** Predicts target values based on the average of the KKK nearest neighbours.
- **Decision Tree Regression:** Splits data recursively to predict target values at leaf nodes.
- **Support Vector Regression (SVR):** Uses support vectors and a margin of error to predict values, often with kernels for non-linearity.
- **Random Forest Regression:** Combines multiple decision trees to improve prediction accuracy and reduce overfitting.

Random Forest proved to be the best model in this case, delivering the highest accuracy and lowest error metrics. It is a powerful and reliable choice for both linear and non-linear relationships.

2. Performance Metrics:

- **R-squared (R^2)** measures how well the model explains the variance in the data, with values closer to 1 indicating better fit.
- **Mean Absolute Error (MAE)** calculates the average magnitude of errors in predictions without considering direction.
- **Random Forest** achieved the highest performance with an R^2 of **0.97** and the lowest MAE of **0.0228**, showing it was the most accurate model.

3. Hyperparameter Tuning:

- **Grid Search Cross-Validation** was used to tune key Random Forest parameters, such as the number of trees (n_estimators), maximum depth (max_depth), and minimum samples split (min_samples_split).
- This technique systematically tested all combinations of parameters using cross-validation, ensuring the model's optimal settings.
- After tuning, the model showed improved accuracy and reduced overfitting, leading to better performance on new data.

DEPLOYMENT

The **deployment** of the Random Forest model using **Flask** involves creating a web application that enables users to input laptop specifications and receive real-time price predictions.

Key Features:

- **Flask Framework**: Flask is used to build the backend of the app, handling user requests, running the model, and sending predictions. It's lightweight, making it ideal for small-scale machine learning deployments.

- **User Interface**: A simple, intuitive web interface is built where users can manually enter laptop specifications like brand, processor, RAM, storage, and color. Once the user submits the data, the Flask app processes it and uses the Random Forest model to predict the price.

- **Real-Time Price Prediction**: The app provides instant price estimates based on the user's input. The backend model (Random Forest) processes the input features, generates predictions, and sends the results to the front end for display.

- **Model Deployment**: The trained Random Forest model is saved using libraries like **pickle** and loaded by the Flask app for prediction. This allows the model to be reused in the deployed environment.

- **Scalability & Accessibility**: The web app is hosted on **Streamlit** for easy access and to handle multiple users.

Workflow:

1. Users visit the web app and input laptop details.
2. Flask processes these inputs and feeds them to the trained Random Forest model.
3. The model predicts the price, which is displayed instantly on the web interface.

Benefits:

- **Practical**: It provides users with real-time, accurate price estimates.
- **User-Friendly**: The simple design makes it accessible to both technical and non-technical users.
- **Scalable and Efficient**: Hosted in the cloud, it's ready to handle many users at once.

STREAMLIT IMPLEMENTATION

```
import streamlit as st
import pickle
import numpy as np
import pandas as pd

# Page Configuration
st.set_page_config(
    page_title="Laptop Price Predictor",
    page_icon="💻",
    layout="wide"
)

# Title
st.title("Laptop Price Predictor 💻")

# Load the model and preprocessed dataframe
try:
    pipe = pickle.load(open("pipe.pkl", "rb"))
    df = pickle.load(open("df.pkl", "rb"))
except:
    st.error("Error: Unable to load model files. Make sure pipe.pkl and df.pkl exist in the same directory.")
    st.stop()

# Create the input form
col1, col2, col3 = st.columns(3)

with col1:
    brand = st.selectbox("Brand",
        options=['Select Brand', 'Apple', 'Lenovo', 'HP', 'DELL', 'SAMSUNG', 'Acer', 'MSI'],
        index=0
    )

with col2:
    ram = st.selectbox("RAM (in GB)",
        options=['Select RAM', 8, 16, 24],
        index=0
    )
with col3:
    size = st.selectbox("Screen Size (in Inches)",
        options=['Select Screen Size', 35.56, 39.62, 34.54, 40.64, 100.63, 100.58, 40.89, 43.94],
        index=0
    )

col4, col5, col6 = st.columns(3)

with col4:
    processor = st.selectbox("Processor",
        options=['Select Processor', 'Intel Core i5', 'Intel Core i3', 'AMD Ryzen 5',
                 'Intel Core i7', 'AMD Ryzen 7', 'Intel Celeron Dual', 'AMD Ryzen 3'],
        index=0
    )

with col5:
    storage = st.selectbox("Storage (in GB)",
        options=['Select Storage', 256, 512, 1024],
        index=0
    )

with col6:
    generation = st.selectbox("Generation",
        options=['Select Generation', 10, 11, 12, 13, 14],
        index=0
    )
```

```python
with col6:
    generation = st.selectbox("Generation",
        options=['Select Generation', 10, 11, 12, 13, 14],
        index=0
    )

col7, col8 = st.columns(2)

with col7:
    os = st.selectbox("Operating System",
        options=['Select OS', 'Windows 11 Operating System', '64 bit Windows 11 Operating System',
                 'Mac OS Operating System', 'Windows 11 Home Operating System'],
        index=0
    )

with col8:
    rating = st.slider("Star Rating", min_value=1.0, max_value=5.0, value=None, step=0.1)
    if rating is not None:
        stars = "⭐" * int(rating)
        if rating % 1 >= 0.5:
            stars += "★"
        st.write(f"Selected Rating: {stars}")

# Validation function
def validate_inputs():
    if brand == 'Select Brand' or \
       ram == 'Select RAM' or \
       size == 'Select Screen Size' or \
       processor == 'Select Processor' or \
       storage == 'Select Storage' or \
       generation == 'Select Generation' or \
       os == 'Select OS' or \
       rating is None:
        return False
    return True
# Create prediction button
if st.button("Predict Price"):
    if not validate_inputs():
        st.error("Please select all required fields before prediction.")
    else:
        try:
            # Create input features
            features = np.zeros(len(pipe.feature_names_in_))
            feature_dict = {}

            # Add numerical features
            feature_dict['RAM'] = ram
            feature_dict['Storage'] = storage
            feature_dict['Size'] = size
            feature_dict['Generation'] = generation
            feature_dict['Rating'] = rating

            # Add one-hot encoded features
            feature_dict[f'Brand_{brand}'] = 1
            feature_dict[f'Processor_{processor}'] = 1
            feature_dict[f'Operating System_{os}'] = 1

            # Create DataFrame with all possible columns initialized to 0
            input_df = pd.DataFrame(columns=pipe.feature_names_in_)
            input_df.loc[0] = 0

            # Fill in the actual values
            for feature, value in feature_dict.items():
                if feature in input_df.columns:
                    input_df.loc[0, feature] = value
```

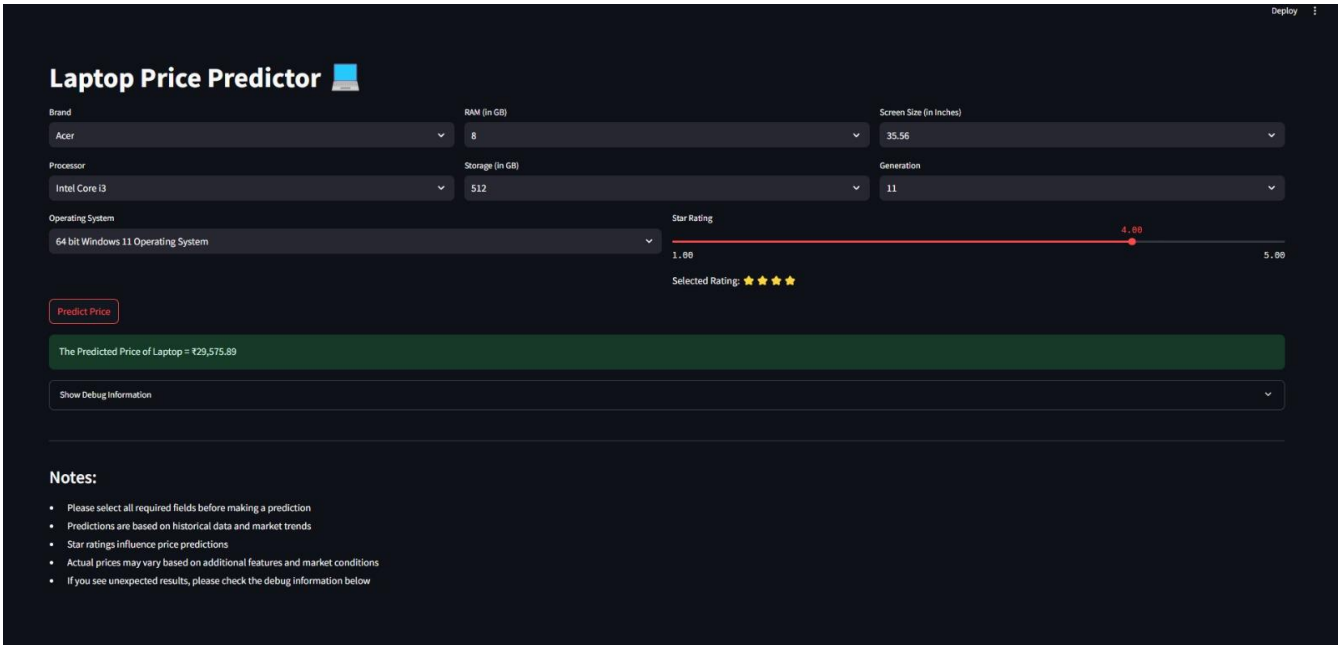
```
        # Make prediction
        log_price = pipe.predict(input_df)[0]

        # Convert log price to actual price with validation
        if log_price > 20:
            st.error("Prediction error: Value out of expected range")
        else:
            prediction = np.exp(log_price)
            if np.isinf(prediction) or prediction > 1000000:
                st.error("Prediction error: Value too large")
            else:
                st.success(f"The Predicted Price of Laptop = ₹{prediction:,.2f}")

        # Debug information
        with st.expander("Show Debug Information"):
            st.write("Log Price:", log_price)
            st.write("Features Used:")
            for col in input_df.columns:
                if input_df.loc[0, col] != 0:
                    st.write(f"{col}: {input_df.loc[0, col]}")

    except Exception as e:
        st.error(f"An error occurred during prediction: {str(e)}")
        st.write("Error details:", str(e))
```

WEB INTERFACE

ACTUAL VALUE –

3	Acer	Acer One	Intel Core i3	11th Gen	64 bit Win	8 GB	512	35.56	1	Black	4.2	1974	277	28990

PREDICTED VALUE –



CONCLUSION

This project successfully predicted laptop prices using various machine learning techniques. The key outcomes are:

- **Random Forest Regression** outperformed all other models, demonstrating its ability to handle complex feature interactions effectively. It provided the best performance with the highest R^2 (0.97), lowest MAE (0.0211), and RMSE (0.046), proving to be the most accurate and robust model for predicting laptop prices.

```
R2 Score: 0.9744555921581622
Mean Absolute Error (MAE): 0.021138412225254852
Root Mean Squared Error (RMSE): 0.04621900449912219
```

- **Deployment via Flask**: The model was successfully deployed using Flask, showcasing the practical utility of the prediction system in a real-time web application. The simple, user-friendly interface allowed for easy interaction, making it accessible to a wide audience.

Model Comparisons:

- **Lasso Regression**: A linear model with L1 regularization for feature selection. While it's simple and interpretable, it had the lowest R^2 (0.92), higher MAE (0.053), and higher RMSE (0.077). This made it less accurate than the other models for this dataset.

```
R2 Score: 0.927538453857948
Mean Absolute Error (MAE): 0.053850776848557505
Root Mean Squared Error (RMSE): 0.07784423688505154
```

- **KNN Regression**: A flexible, non-parametric model that predicts based on the average of nearest neighbours. While it performed well (R^2 = 0.93, MAE = 0.04, RMSE = 0.072), it was still slightly less accurate than Random Forest.

```
R2 Score: 0.9365686113076417
Mean Absolute Error (MAE): 0.04082410496630114
Root Mean Squared Error (RMSE): 0.07283242612857899
```

- **Decision Tree Regression**: A model that recursively splits data to capture complex relationships. Although it performed better than Lasso and KNN (R^2 = 0.95, MAE = 0.021, RMSE = 0.064), it's prone to overfitting, which slightly affects its accuracy compared to Random Forest.

```
R2 Score: 0.9509498583743491
Mean Absolute Error (MAE): 0.021659895352501388
Root Mean Squared Error (RMSE): 0.06404611637432991
```

- **Support Vector Regression (SVR)**: A non-linear model using kernel methods to model complex relationships. However, it had the lowest R^2 (0.91) and higher error metrics (MAE = 0.063, RMSE = 0.083), making it less effective for this specific dataset.

```
R2 Score: 0.9161009549478853
Mean Absolute Error (MAE): 0.06318344715410998
Root Mean Squared Error (RMSE): 0.08376279802331756
```

Summary:

- **Random Forest** proved to be the best model in this case, delivering the highest accuracy and lowest error metrics. It is a powerful and reliable choice for both linear and non-linear relationships. While other models like **SVR** and **Lasso** offer good performance, they fall short in accuracy compared to **Random Forest**.

This project highlights the importance of choosing the right model based on the dataset and problem at hand, with **Random Forest** being the optimal choice for predicting laptop prices in this case.

REFERENCES

- https://www.ijnrd.org/papers/IJNRD2303124.pdf
- https://www.researchgate.net/publication/377721653_Machine_Learning-Based_Price_Prediction_for_Laptops
- https://ijirt.org/publishedpaper/IJIRT158273_PAPER.pdf
- https://ijcsmc.com/docs/papers/January2022/V11I1202229.pdf
- https://www.researchgate.net/publication/374120586_Laptop_Price_Prediction_with_Machine_Learning_Using_Regression_Algorithm
- https://ieeexplore.ieee.org/document/10085473

PLAGIARISM REPORT

BDA Report Group - 4.docx

ORIGINALITY REPORT

11%	7%	3%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	1%
2	fastercapital.com Internet Source	1%
3	dokumen.pub Internet Source	1%
4	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
5	www.researchgate.net Internet Source	1%
6	Kode, Kiran Chandrakant. "Machine Learning-Based Advanced Predictive Models for Analyzing Passive Component Shifts in Surface Mount Technology", State University of New York at Binghamton, 2024 Publication	1%
7	Submitted to University of Keele Student Paper	1%