A PROJECT REPORT ON

LAPTOP PRICE PREDICTOR

SUBMITTED TO

CHANDIGARH UNIVERSITY

By

Sweta Dey
24MCI10247

In partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATION

IN
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

UNIVERSITY INSTITUTE OF COMPUTING,

CHANDIGARH UNIVERSITY,

ACKNOWLEDGEMENT

In performing our project, I would like to show my gratitude to all the faculty members of University Institute of Computing, Chandigarh University for giving us a good guideline for the project throughout numerous consultations. I would also like to expand my deepest gratitude to all those who have directly and indirectly guided me in completion of this project.

In addition, a thank you to my Prof. Dr. Pooja Thakur, my mentor who introduced me to the Art of Computer Programming, and her passion for the "underlying structures" had lasting effect. I also thank the University of Chandigarh for consent to include copyright pictures as a part of our paper.

Many people, especially my classmates have made valuable comment suggestions on this project which gave me an inspiration to improve my application.

I thank all the people for their help directly and indirectly to complete my assignment.

CONTENT

PREFACE

Laptop pricing is a crucial aspect of the technology market, with numerous factors influencing the cost of a device. Consumers often struggle to determine fair pricing based on specifications such as processor type, RAM, storage, GPU, and display features. Given the vast variety of laptop models available, predicting a reasonable price can be a complex task.

This project utilizes a machine learning approach, specifically the Random Forest Regressor, to predict laptop prices based on various attributes. By training on a dataset containing laptop specifications and prices, the model learns to make accurate price estimations. The integration of this model into a Streamlit-based web application allows users to input specific laptop configurations and receive instant price predictions.

Although this project provides a fundamental framework for price prediction, future enhancements can include real-time market data integration, additional features, and alternative machine learning models for improved accuracy. This initiative serves as a stepping stone towards making price estimation more accessible and reliable for consumers and businesses alike.

PROBLEM STATEMENT

Given a dataset of laptop specifications and prices, the goal is to develop a predictive model that estimates the price of a laptop based on its attributes. The challenge is to preprocess and encode categorical variables, train an efficient machine learning model, and deploy an interactive interface for users.

In practice, laptop pricing depends on multiple factors such as hardware components, brand value, and market trends. While the Random Forest Regressor provides a robust foundation for price prediction, real-world applications require additional considerations, such as real-time pricing data integration and dynamic feature selection.

3. INTRODUCTION

Laptop pricing is a crucial aspect of the technology market, as prices vary widely based on brand, specifications, and market demand. Consumers often struggle to determine the fair price of a laptop due to the wide range of available models and configurations. Factors such as processor type, RAM capacity, storage options, GPU capabilities, and display size all contribute to pricing, making it difficult to assess whether a laptop is overpriced or offers good value for money. As technology advances and more models enter the market, an efficient method for estimating laptop prices becomes increasingly necessary for both consumers and retailers.

In recent years, machine learning has emerged as a powerful tool for predicting outcomes based on historical data. By leveraging machine learning techniques, we can analyze past laptop prices and their corresponding specifications to create a predictive model. This project employs a **Random Forest Regressor**, a widely used machine learning algorithm, to estimate laptop prices based on key attributes. The model is trained on a dataset containing various laptop specifications and their respective prices, enabling it to generate accurate price predictions for new configurations.

To ensure ease of use, the model is integrated into a **Streamlit web application**, allowing users to input laptop specifications and receive real-time price predictions. This interactive platform provides an accessible way for consumers to make informed purchasing decisions and for businesses to set competitive pricing. The application processes user inputs, encodes categorical data, and utilizes the trained machine learning model to generate price estimations instantly. Additionally, the model's accuracy is evaluated using **R^2 (R-Squared) Score**, ensuring reliable predictions.

While this project establishes a foundational approach to laptop price prediction, there is potential for future enhancements. Incorporating real-time market data, additional machine learning models, and dynamic feature selection can further refine accuracy and usability. By leveraging artificial intelligence, this project demonstrates how data-driven insights can improve decision-making in the laptop market.

3.1 Objective

- **Develop a laptop price prediction system**
  The system will estimate the price of a laptop based on key specifications.
  It will provide accurate price predictions using machine learning techniques.

- **Implement a Random Forest Regressor model**
  The Random Forest Regressor will be used for price estimation.
  It ensures reliable predictions by handling non-linear relationships in data.

- **Create a user-friendly interface**
  A Streamlit web application will allow users to input laptop specifications.
  The interface will display the predicted price instantly for easy decision-making.

- **Ensure model accuracy and efficiency**
  The model will be trained on a well-processed dataset for better precision.
  Performance metrics like R^2 Score will be used to evaluate accuracy.

- **Enhance feature selection and preprocessing**
  The system will preprocess and encode categorical variables efficiently.
  Feature selection will focus on relevant attributes to improve model performance.

- **Enable future scalability and improvements**
  The system will allow for the integration of real-time market data.
  Future improvements may include alternative machine learning models for enhanced accuracy.

3.2 Scope of the Project

- **Automated Laptop Price Prediction**
  The project focuses on developing a machine learning-based system that predicts laptop prices based on key hardware specifications. This system helps users estimate the cost of a laptop without manually comparing multiple models.

- **Machine Learning Model Integration**
  A Random Forest Regressor model is implemented to analyze and predict laptop prices. The model is trained on historical pricing data and considers factors such as processor type, RAM, storage, GPU, display size, and brand value.

- **User-Friendly Web Interface**
  The system includes a Streamlit-based web application where users can input laptop specifications and receive real-time price predictions. This enhances accessibility and ease of use for consumers and businesses.

- **Data Processing and Feature Engineering**
  The project involves data preprocessing techniques such as cleaning, encoding categorical variables, feature selection, and data transformation to improve model accuracy.

- **Model Evaluation and Performance Optimization**
  The project evaluates the model using performance metrics like R^2 Score to ensure accurate predictions. Techniques such as hyperparameter tuning and feature engineering are explored to enhance efficiency.

- **Future Enhancements and Scalability**
  The system is designed to support future improvements, including integration with real-time pricing data, additional feature selection methods, and alternative machine learning models to improve prediction accuracy.

3.3 Survey of Technology

Operating System: Windows Operating System

Software used: VS Code, Kaggle

Language used: Python

3.4 Requirements

Hardware requirements for running this project are as follows:

OS: - Windows XP and above

RAM: Ideally 2GB or above

Graphics: Integrated / Min 2GB (discrete)

Hard disk: Min 128 GB

Software requirements for running this application are as follows:
RAM: 1 GB or above.

Memory Space: 500 MB or above.

Languages and Platform used:

Platform: Output window

Language: Python

4. LITERATURE REVIEW

4.1 Overview of Random Forest

Random Forest is a widely used ensemble learning algorithm that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It operates using two key techniques: **Bagging (Bootstrap Aggregation)**, where multiple subsets of training data are randomly sampled to build individual trees, and **Feature Randomness**, where each tree is trained on a random subset of features to ensure diversity in predictions.

This algorithm is robust, handles both numerical and categorical data efficiently, and is highly resistant to overfitting compared to individual decision trees. It also provides feature importance scores, helping identify key attributes influencing predictions. Random Forest is used in various applications, including price prediction, medical diagnosis, fraud detection, and recommendation systems.

Key advantages of Random Forest include:

- **High Accuracy:** The aggregation of multiple trees improves prediction reliability.
- **Handles Missing Data:** Can process incomplete datasets without significant performance loss.
- **Reduces Overfitting:** The randomness in feature selection and data sampling prevents overfitting.
- **Feature Importance Analysis:** Identifies key variables affecting predictions.
- **Scalability:** Works well with large datasets and high-dimensional data.

In this project, Random Forest is employed for laptop price prediction due to its ability to handle complex feature relationships and provide accurate price estimations. Its ensemble nature ensures stability, making it an ideal choice for this application.

4.2 Applications of Random Forest in laptop Price Prediction

Random Forest is a powerful machine learning algorithm widely used for laptop price prediction due to its ability to handle multiple factors like brand, processor, RAM, storage, and GPU. Its ensemble learning approach ensures accurate and reliable predictions.

Key Applications:

- **Feature-Based Price Estimation:** Identifies key attributes influencing laptop prices, such as processor type and RAM size.
- **Handling Mixed Data Types:** Efficiently processes both categorical (brand, OS) and numerical (RAM, storage) variables.
- **Reducing Overfitting:** Averaging multiple tree predictions prevents overfitting and improves generalization.
- **High Accuracy with Large Datasets:** Learns complex patterns, maintaining strong predictive performance.
- **Market Trend Analysis:** Helps forecast future prices based on historical data.
- **User-Friendly Implementation:** Can be integrated into web applications like Streamlit for real-time price estimation.

Random Forest's versatility and robustness make it an ideal choice for developing an efficient laptop price prediction model.

4.3 Existing Solutions and Tools

Several existing solutions and tools are available for laptop price prediction, leveraging various machine learning models and online databases.

- **Online Price Comparison Websites:**

Platforms like Amazon, Flipkart, and Best Buy provide real-time laptop prices based on user-selected specifications. However, they do not predict prices for custom configurations.

- **Machine Learning-Based Models:**
  - **Linear Regression:** Simple and interpretable but struggles with complex relationships.
  - **Decision Trees:** Captures non-linear patterns but prone to overfitting.
  - **Random Forest:** Provides robust and accurate predictions by averaging multiple decision trees.
  - **Price Estimation Software & APIs:**
    Some platforms use APIs to fetch price trends and historical data for better price estimation. However, they may lack predictive capabilities for new configurations.

While these solutions offer price insights, our Random Forest-based model enhances prediction accuracy by learning from extensive datasets, providing a reliable and customizable price estimation tool.

5. SYSTEM DESIGN AND ARCHIECTURE

5.1 Data Structures

- **Feature Matrix (X) & Target (y)**:
  - X contains numerical and encoded categorical laptop specifications.
  - y stores log-transformed prices for better model accuracy.

- **Label Encoders (label_encoders)**:
  - A **dictionary** mapping categorical features to numerical values.

- **Training & Testing Data**:
  - Data is split into 80% training, 20% testing using train_test_split().

- **Random Forest Model (rf_model)**:
  - A RandomForestRegressor trained for price prediction.

- **Pickle Storage (rf_model.pkl, label_encoders.pkl)**:
  - Model and encoders are saved for future predictions.

- **Streamlit UI for User Input**:
  - Users select specs via dropdowns, which are encoded and fed into the model.

5.2 Algorithm Implementation

- **Data Processing**
  - Load and clean the dataset.
  - Encode categorical features.
  - Split data into training (80%) and testing (20%).

- **Model Training**
  - Train a **Random Forest Regressor** with 100 trees.
  - Apply **log transformation** to price for accuracy.
  - Save the model and encoders.

- **Web App (Streamlit)**
  - Load the trained model.
  - Take user inputs for laptop specs.
  - Convert inputs using encoders.

- **Prediction & Accuracy**
  - Predict price and apply exponentiation.
  - Show price in ₹ and display **R^2 score**.

6. METHODOLOGY

6.1 Step-by-Step Process of Random Forest

- **Initialize Data Structures** – Load the dataset, encode categorical variables, and apply a logarithmic transformation to the price for better model performance.
- **Train-Test Split** – Split the dataset into 80% training and 20% testing to ensure the model generalizes well to new data.
- **Train Random Forest Model** – Train a Random Forest Regressor with 100 decision trees to capture complex relationships between features and price. Save the trained model and encoders for later use.
- **Prediction and Evaluation** – Load the model in a Streamlit app, take user inputs, encode them, and predict the laptop price. Display the result along with the model's accuracy using the R^2 score.

6.2 Input and Output Specifications

Input: The user provides laptop specifications such as brand, processor, RAM, storage, GPU, display size, resolution, operating system, and warranty.

Output: The system predicts the estimated price of the laptop based on the provided specifications and displays the result along with model accuracy.

6.3 User Interface Design

Streamlit is an open-source Python framework for building interactive web applications with minimal code. It simplifies the integration of machine learning models with user-friendly interfaces, allowing real-time updates and interactive widgets. In the **Laptop Price Prediction** project, Streamlit enables users to input specifications and get instant price predictions.

Here, the Streamlit-based interface allows users to select laptop specifications and get a price estimate.

- **Title**: Displays "Laptop Price Predictor."
- **Inputs**: Dropdowns for selecting brand, model, processor, RAM, storage, GPU, display size, resolution, OS, and warranty.
- **Prediction**: Users click "Predict Price", and the trained model estimates the price.
- **Output**: Displays the predicted price in ₹ along with model accuracy.

7. CODE IMPLEMENTATION

7.1 Algorithm Code

home.py > ...

```python
import streamlit as st
import pickle
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score

# Load the dataframe
df = pd.read_csv("data.csv")
df = df.drop(df.columns[[0, 1, 9]], axis=1)

df.rename(columns={
    'brand': 'brand',
    'name': 'name',
    'price': 'price',
    'spec_rating': 'spec_rating',
    'processor': 'processor',
    'CPU': 'cpu',
    'Ram': 'ram',
    'ROM': 'rom',
    'ROM_type': 'rom_type',
    'GPU': 'gpu',
    'display_size': 'display_size',
    'resolution_width': 'resolution_width',
    'resolution_height': 'resolution_height',
    'OS': 'os',
    'warranty': 'warranty'
}, inplace=True)
```

```python
# Encode categorical variables
categorical_columns = ['brand', 'name', 'processor', 'cpu', 'ram', 'rom', 'rom_type', 'gpu', 'os']
label_encoders = {}

for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Train-test split
X = df[['brand', 'name', 'processor', 'cpu', 'ram', 'rom', 'rom_type', 'gpu', 'display_size', 'resolution_width',
y = np.log(df['price'])  # Log transformation for price
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Save the trained model and encoders
pickle.dump(rf_model, open("rf_model.pkl", "wb"))
pickle.dump(label_encoders, open("label_encoders.pkl", "wb"))

st.title("Laptop Price Predictor")
```

```python
# Load model and encoders
model_path = "rf_model.pkl"
encoder_path = "label_encoders.pkl"
if os.path.exists(model_path) and os.path.exists(encoder_path):
    model = pickle.load(open(model_path, "rb"))
    label_encoders = pickle.load(open(encoder_path, "rb"))
else:
    st.error("Error: Model or encoder file not found.")
    st.stop()

# User Inputs with placeholders
brand = st.selectbox("Brand", label_encoders['brand'].classes_, index=None, placeholder="Select brand")
laptop_name = st.selectbox("Laptop Model", label_encoders['name'].classes_, index=None, placeholder="Select model")
processor = st.selectbox("Processor", label_encoders['processor'].classes_, index=None, placeholder="Select processor")
cpu = st.selectbox("CPU", label_encoders['cpu'].classes_, index=None, placeholder="Select CPU")
ram = st.selectbox("RAM (in GB)", label_encoders['ram'].classes_, index=None, placeholder="Select RAM")
rom = st.selectbox("Storage (ROM)", label_encoders['rom'].classes_, index=None, placeholder="Select ROM")
rom_type = st.selectbox("Storage Type", label_encoders['rom_type'].classes_, index=None, placeholder="Select storage type")
gpu = st.selectbox("GPU", label_encoders['gpu'].classes_, index=None, placeholder="Select GPU")
display_size = st.selectbox("Display Size (in inches)", df['display_size'].unique(), index=None, placeholder="Select display
resolution_width = st.selectbox("Screen Resolution Width", df['resolution_width'].unique(), index=None, placeholder="Select w
resolution_height = st.selectbox("Screen Resolution Height", df['resolution_height'].unique(), index=None, placeholder="Selec
os = st.selectbox("Operating System", label_encoders['os'].classes_, index=None, placeholder="Select OS")
warranty = st.selectbox("Warranty (years)", df['warranty'].unique(), index=None, placeholder="Select warranty")
```

```python
# Function to safely encode categorical values
def safe_encode(label_encoder, value):
    if value in label_encoder.classes_:
        return label_encoder.transform([value])[0]
    else:
        return -1  # Assign unseen label a fixed number

# Prediction
if st.button('Predict Price'):
    try:
        ppi = ((resolution_width ** 2) + (resolution_height ** 2)) ** 0.5 / display_size
        query = np.array([
            safe_encode(label_encoders['brand'], brand),
            safe_encode(label_encoders['name'], laptop_name),
            safe_encode(label_encoders['processor'], processor),
            safe_encode(label_encoders['cpu'], cpu),
            safe_encode(label_encoders['ram'], ram),
            safe_encode(label_encoders['rom'], rom),
            safe_encode(label_encoders['rom_type'], rom_type),
            safe_encode(label_encoders['gpu'], gpu),
            display_size,
            resolution_width,
            resolution_height,
            safe_encode(label_encoders['os'], os),
            warranty
        ])
```

```python
        ])
        query = query.reshape(1, -1)
        predicted_price = np.exp(model.predict(query)[0])
        st.title(f"The predicted price of this configuration is ₹{int(predicted_price)}")

        # Calculate accuracy for the prediction
        y_test_pred = model.predict(X_test)
        prediction_accuracy = r2_score(y_test, y_test_pred)
        st.write(f"Model Accuracy: {prediction_accuracy:.2f}")
    except Exception as e:
        st.error(f"Error occurred during prediction: {str(e)}")
```

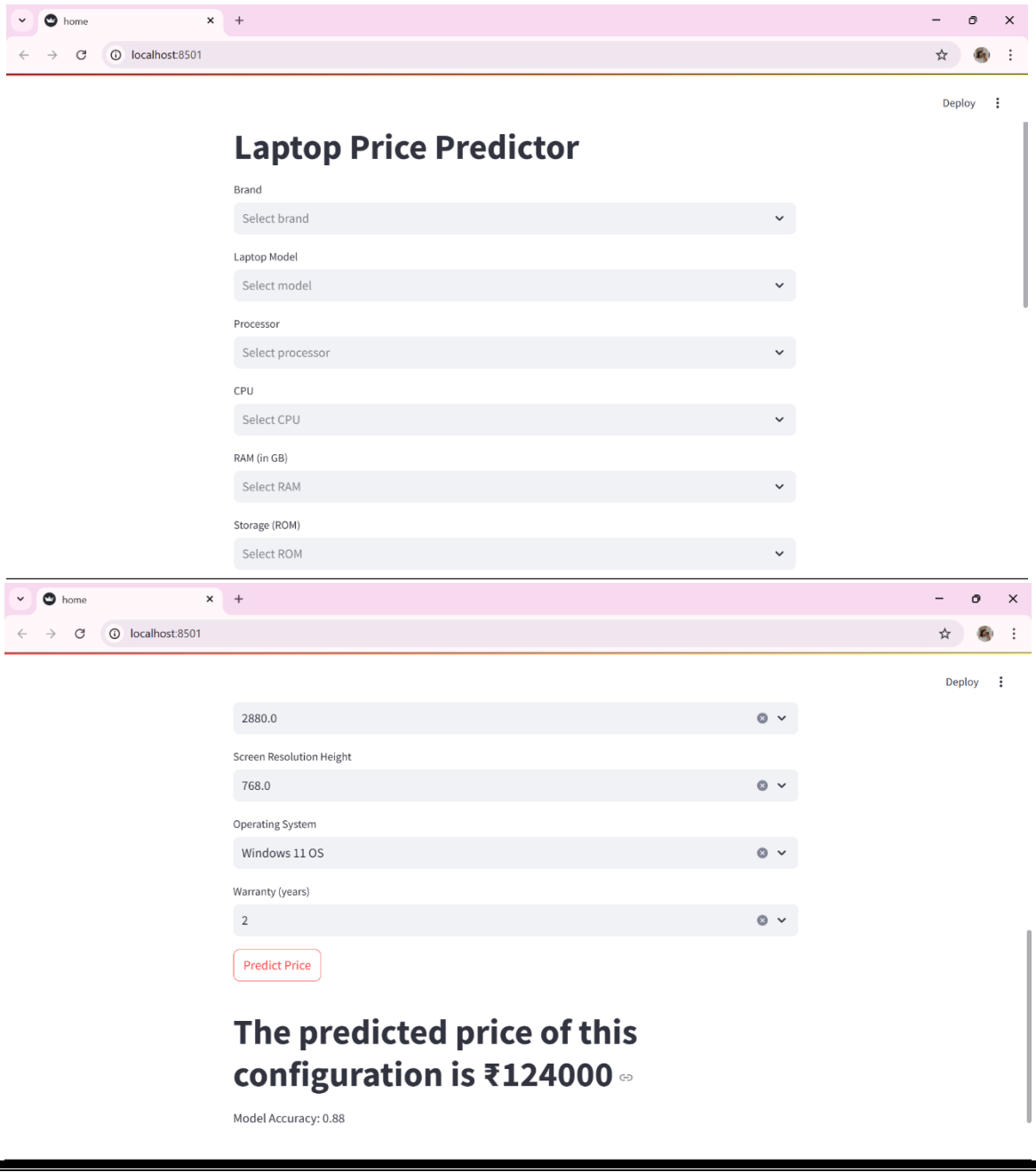
7.2 Explanation of Key Functions

- **safe_encode()** – Encodes categorical inputs, handling unseen values.
- **st.selectbox()** – Creates dropdowns for user input selection.
- **st.button('Predict Price')** – Triggers prediction when clicked.
- **model.predict(query)** – Uses the trained model to estimate the laptop price.
- **r2_score()** – Evaluates model accuracy.
- **pickle.load() & pickle.dump()** – Saves and loads the trained model for reuse.

7.3 Data Input

The code loads and cleans the dataset, encoding categorical features using LabelEncoder(). User inputs are collected through st.selectbox() in Streamlit, formatted for model prediction, and used to estimate laptop prices efficiently.

7.4 Snapshots



7.5 Performance Analysis

The Random Forest model efficiently predicts laptop prices with **O(n log n)** training time and **O(t d)** prediction time. It ensures accuracy, reduces overfitting, and handles diverse laptop configurations. While memory-intensive, optimizations help manage resources, providing real-time price estimates for users.

8. BENEFITS

- **Optimized Laptop Purchasing**
  The system predicts laptop prices based on specifications, helping buyers choose the best option within their budget while ensuring fair market value.

- **Cost Reduction for Sellers**
  By analyzing pricing trends, retailers can set competitive prices, optimize inventory, and reduce losses due to overpricing or underpricing.

- **Enhanced Decision-Making**
  The model provides data-driven insights, allowing consumers, businesses, and retailers to make informed purchasing and pricing decisions.

- **Easy Price Comparison**
  Users can compare different laptop models based on specifications and estimated prices, simplifying the decision-making process.

- **Scalability and Adaptability**
  The system can be expanded to include real-time market data, additional product categories, and integration with e-commerce platforms for a more dynamic experience.

9. CHALLENGES AND LIMITATIONS

- **Data Dependency**
  The model's accuracy is limited by the quality and completeness of the dataset, lacking real-time price updates.

- **Categorical Encoding Issues**
  Unseen laptop brands or specifications may not be correctly encoded, affecting prediction accuracy.

- **Scalability Challenges**
  Handling large datasets or expanding to global markets may reduce efficiency and increase computation time.

- **User Input Limitations**
  Incorrect or missing inputs in the Streamlit interface can lead to errors or unreliable predictions.

10. CONCLUSION

The Laptop Price Prediction project effectively utilizes machine learning to estimate laptop prices based on specifications. By integrating a Random Forest Regressor with a Streamlit interface, it provides accurate and user-friendly predictions. While the model performs well, it has limitations like data dependency and lack of real-time pricing. Future enhancements can improve accuracy with dynamic data integration and advanced algorithms, making it a valuable tool for buyers and businesses.

10.1 Future Scope

- **Real-Time Price Updates**
  Integrating live market data will improve prediction accuracy by reflecting real-time pricing trends.

- **Enhanced Model Performance**
  Exploring deep learning techniques and advanced ML models can further refine predictions.

- **Expanded Dataset**
  Including more laptop brands, models, and configurations will improve model generalization.

- **Feature Optimization**
  Incorporating additional factors like user reviews, demand trends, and seasonal pricing will enhance accuracy.

- **Deployment as a Web Service**
  Making the model accessible via a cloud-based API will allow seamless integration with e-commerce platforms.

11. REFERENCES

- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR, 12*, 2825-2830.

- Wikipedia contributors. (2024). Random Forest. *Wikipedia, The Free Encyclopedia.*

- Streamlit Documentation. (2024). Streamlit: The Fastest Way to Build and Share Data Apps.

GITHUB LINK: **https://github.com/swetaaa09/LAPTOP-PRICE-PREDICTION-USING-RANDOM-FOREST.git**