Amrit Science Campus

Tribhuvan University

Institute of Science and Technology

A PROJECT REPORT

ON

Laptop Price Prediction using different Machine Learning Algorithms

Under the supervision of

Dhirendra Kumar Yadav

Submitted to

Institute of Science and Information Technology

Tribhuvan University

In Partial Fulfillment of the Requirement for the Bachelor's Degree in Computer Science and Information Technology

Submitted by

Anjala Bhatta (23118/076)
Prapanna Bista (23173/076)
Mahesh Chandra Regmi (23153/076)

SUPERVISOR'S RECOMMENDATION

This report, titled "Laptop Price Prediction using different Machine Learning Algorithms" was prepared under my supervision by Anjala Bhatta (23118/076), Prapanna Bista (23173/076), and Mahesh Chandra Regmi (23153/076). It partially fulfills the requirements for the degree of B.Sc. in Computer Science and Information Technology (B. Sc. CSIT), and I hereby recommend that it be processed for evaluation.

……..

Asst. Prof. Dhirendra Kumar Yadav

Supervisor

Department of Computer Science and Information Technology

Amrit Science Campus

Kathmandu, Nepal

LETTER OF APPROVAL

This is to certify that this project **Anjala Bhatta (23118/076), Prapanna Bista (23173/076) and Mahesh Chandra Regmi (23153/076)** entitled "**Laptop Price Prediction using different Machine Learning Algorithms"** in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT) has been well studied. In our opinion, it is satisfactory in the scope and quality for the required degree.

Signature of Supervisor ……. **Asst. Prof. Dhirenda Kumar Yadav** Amrit Science Campus	Signature of Co-Ordinator ……. **Asst. Prof. Dhirenda Kumar Yadav** Amrit Science Campus
Signature of Internal Examiner ……. Amrit Science Campus	Signature of External Examiner ……. IOST, Tribhuvan University

ACKNOWLEDGEMENT

This project would not have been able to be completed successfully without the help and cooperation of numerous people and organizations. We are incredibly grateful that we received this during our study. We would want to use this occasion to express our sincere admiration for each and every one of them.

First and foremost, we would like to thank **Asst. Prof. Dhirendra Kumar Yadav**, our project supervisor and mentor, for his guidance and support in helping us with the project. His direction has been essential to the development of this project. With his assistance, we successfully completed this project thanks to his insightful advice and recommendations.

We would especially like to thank the Department of Computer Science and Information Technology for giving us the space to explore the newest technological advancements and conduct in-depth study in those areas as part of our main project.

Not to mention, we would want to express our gratitude to all of our instructors, seniors, and friends for their unwavering support and ongoing inspiration, which enabled us to overcome our dreams and complete our project.

ABSTRACT

This study explores the prediction of laptop prices using Support Vector Machine (SVM), Decision Tree, and Linear Regression algorithms. The system is built using React for the frontend, Flask for the backend and Jupyter notebook for data preprocessing the Kaggle dataset. SVM aids in determining optimal separation between various laptop types based on features such as RAM, processor speed, and screen size. Decision Tree facilitates the breakdown of features into interpretable segments, elucidating their impact on pricing. Linear Regression establishes a linear relationship between features and price. By integrating these algorithms, a robust prediction model is developed, which helps to predict the price of laptop and analysis of algorithm.

Keywords: *Machine Learning, Price Prediction, Decision Tree, Data Preprocessing, Flask, React, Kaggle Dataset*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

API	Application Programming Interface
CASE	Computer-Aided Software Engineering
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DFD	Data Flow Diagram
GUI	Graphical User Interface
MAE	Mean Absolute Error
RAM	Random Access Memory
VS Code	Visual Studio Code

CHAPTER 1: INTRODUCTION

1.1 Introduction

This project focuses on comparing the effectiveness of three machine learning algorithms Linear Regression, Decision Tree, and Support Vector Machine (SVM) in predicting laptop prices. Through a user-friendly web application, individuals can input various technical specifications of a laptop, such as RAM, processor speed, screen size, and storage capacity. These specifications serve as the features for our predictive models. Utilizing the trained models, the system then generates price predictions based on the provided specifications. Subsequently, the web application presents graphical representations showcasing the performance of each algorithm in terms of prediction accuracy. By analyzing these visualizations and associated metrics, the project aims to determine the algorithm that offers the most precise predictions for laptop prices, thereby assisting users in making informed decisions during laptop transactions. Through the amalgamation of advanced machine learning methodologies and intuitive web development technologies, this project seeks to provide a practical solution for predicting laptop prices, aiding users in assessing the value of their prospective purchases.

1.2 Problem Statement

The problem statement revolves around two key challenges. Firstly, the prices of laptops can vary significantly between different stores, creating uncertainty for consumers who want to make informed purchasing decisions. Secondly, various machine learning algorithms produce different predictions for laptop prices, making it essential to identify the most effective algorithm for this specific task. Therefore, the goal is to develop a reliable method for predicting laptop prices accurately across multiple retailers, while also determining which algorithm performs best in this context. This will empower consumers with valuable insights to navigate the diverse laptop market confidently.

1.3 Objectives

This study's primary goal is to identify, measure, and examine the critical variables influencing laptop prices using three different machine learning algorithms. The final objective is to create a reliable predictive model that can calculate a laptop's pricing using its parameters. The following is an outline of the main goal

- To develop a GUI to predict laptop price on the basis of different features entered by user.
- To develop a predicted price visualizer on the basis of different algorithm's evaluation metrices.

1.4 Scope and Limitation

The scope of this project is to determine a fair and reasonable price for a laptop. The primary focus of the project "Laptop Price Prediction using different machine learning algorithms" is to identify the factors that impact the price of a laptop and utilize these genuine factors to predict the appropriate price using regression algorithms. The model will be developed using machine learning techniques to accurately predict the price of a laptop based on various factors such as weight, size and other important features. By doing so, the project aims to provide a useful tool for both buyers and sellers in the laptop market.

Some of the scopes are:

i. This system can be predicting laptop price.
ii. This system can be used for research and analysis.
iii. This system can be used for buying and selling laptops.
iv. This system can be used in integration with laptop's ecommerce websites and application.

Some of limitations are:

i. Limited availability of dataset.
ii. It is difficult to address all available factor.
iii. Prediction failed with mac price prediction, as we have considered only Intel and AMD processor during feature extraction.

1.5 Development Methodology

For this laptop price prediction project, we adopted the Waterfall development methodology, characterized by its linear and sequential approach. In this method, each phase is completed in its entirety before progressing to the next, with minimal overlap

between phases. The Waterfall model for our project encompasses the following key phases:

1.5.1 Requirement Gathering:

This phase involved gathering detailed requirements for the laptop price prediction system, including understanding the laptop specifications affecting pricing, desired prediction accuracy, user interface requirements, and performance benchmarks.

1.5.2 System Design:

After collecting requirements, the system's architecture was designed. This includes selecting the Decision Tree algorithm, Linear regression and SVM for modeling, planning data preprocessing steps, and conceptualizing user interactions. Design documents like system architecture and data flow diagrams were created for visualization.

1.5.3 Implementation:

The system was developed during this phase. Tasks include implementing the Decision Tree model, Linear regression and SVM for creating data cleaning/preprocessing scripts, and building Flask backend and React frontend components based on the design specifications.

1.5.4 Testing:

Rigorous testing was conducted to ensure system reliability and effectiveness. This includes unit testing for individual components and integration testing to confirm seamless system operation.

1.5.5 Deployment:

Once testing was successful, the system is deployed to a production environment. This phase involves setting up server infrastructure, deploying code, and ensuring system stability and security.

1.5.6 Maintenance:

The final phase involved ongoing maintenance where the system is monitored for issues during usage. User feedback was collected, and updates and fixes are implemented to maintain accuracy and user-friendliness.

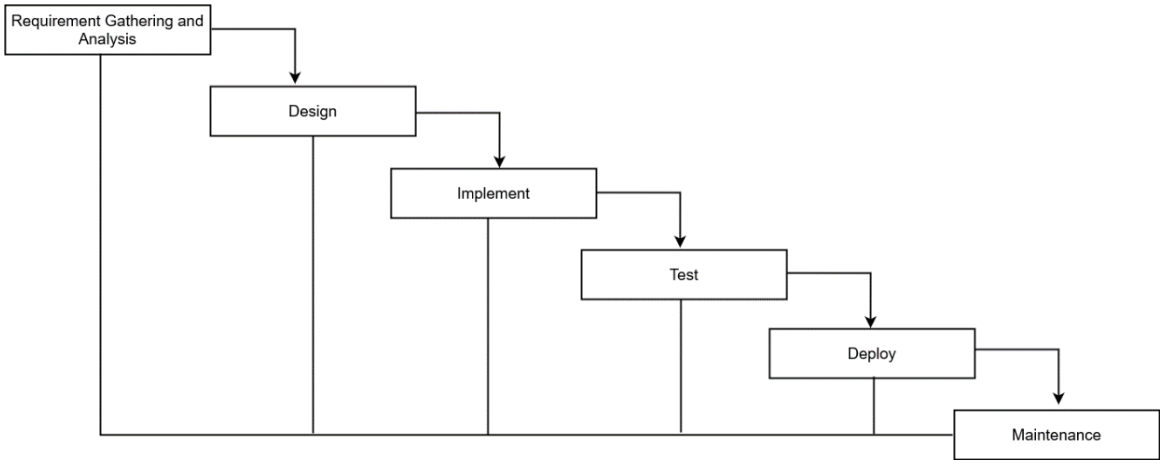


Figure 1.1: Waterfall Model

1.6 Report Organization

The project report adheres to the prescribed format and guidelines established by the Central Department of Computer Science and Information Technology, comprising six distinct chapters:

Chapter 1: Introduction

This section introduces the project, outlining the problem statements aimed to be addressed, the defined objectives, scope, limitations, and the chosen development methodology.

Chapter 2: Background Study and Literature Review

This chapter delves into the background research and literature review, exploring related works and key terminologies relevant to the project.

Chapter 3: System Analysis

System Analysis is detailed in this chapter, covering requirement analysis and feasibility study to ascertain the project's viability and functional requirements.

Chapter 4: System Design

This chapter elucidates the system design, providing insights into the design architecture and the selection of algorithms utilized.

Chapter 5: Implementation and Testing

Implementation and Testing are discussed in this chapter, focusing on the tools employed for system development and the methodology adopted for testing the system's functionality.

Chapter 6: Conclusion

The final chapter, Conclusion, summarizes the project's outcomes, offers future recommendations, and provides an overall project summary.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

The laptop market has experienced remarkable growth, fueled by technological advancements and a growing reliance on digital tools for work, learning, and entertainment. This growth has led to a diverse range of laptop options, each with unique specifications and prices, catering to various consumer needs and budgets. Predicting laptop prices accurately presents a challenge due to the intricate interplay of hardware specifications, brand perceptions, and dynamic market trends that collectively shape a laptop's value.

To address this complexity, machine learning and data analytics have emerged as invaluable tools for understanding and forecasting pricing strategies. This project employs Decision Tree algorithms, Linear Regression, and Support Vector Machines (SVM) to construct a predictive model capable of navigating the multifaceted factors influencing laptop prices. The objective is to improve the accuracy of price predictions using different algorithm analysis.

2.2 Literature Review

Prof. Vaishali Surjuse and her team have made significant contributions to the field of laptop price prediction through their research utilizing machine learning techniques, such as multiple linear regression and decision tree algorithms. Their work showcases a high level of prediction precision, reaching 81%, and emphasizes the importance of feature engineering and exploratory data analysis in understanding the factors influencing laptop prices. By integrating machine learning models with web applications, they have created user-friendly interfaces for predicting laptop prices, making it easier for consumers, particularly students, to make informed decisions based on desired specifications and budget constraints. The research also acknowledges the valuable insights provided by entrepreneurs in facilitating the completion of the study, highlighting the potential usefulness of the developed laptop price prediction system for guiding consumer choices in the tech industry [1].

Siburian et al.'s research employs Machine Learning and regression algorithms to predict laptop prices effectively, catering to the rising need for laptops in Work From Home scenarios amid the COVID-19 pandemic. By utilizing a structured methodology that includes data acquisition, cleaning, feature engineering, exploratory data analysis, and model building with AutoML, the study demonstrates the significance of regression algorithms in price prediction. Through feature engineering and exploratory data analysis, the authors enhance the model's predictive capabilities by extracting valuable insights from the dataset. The high accuracy achieved by the XGBoost algorithm underscores the potential of advanced machine learning techniques in accurately forecasting laptop prices, providing valuable insights for individuals making informed purchasing decisions for their WFH setups [2].

The literature review based on the works presented in the "Laptop Price Prediction" emphasizes the importance of machine learning algorithms, such as Multinomial Naïve Bayes, Support Vector Machines, Decision Trees, and Artificial Neural Networks, in predicting laptop prices. These algorithms are utilized for classification tasks, enhancing the management of laptop products on e-commerce platforms and providing customers with efficient shopping experiences. The research showcases the application of improved KNN classification and decision tree regression for predicting commodity prices, indicating the potential for accurate price predictions in the laptop market. Overall, the studies underscore the significance of machine learning in optimizing pricing strategies, improving classification accuracy, and aiding decision-making processes for both customers and businesses in the laptop industry [3].

The authors Pragnatha et al. present a Laptop Price Prediction Model in the "Laptop Price Estimation Using Machine Learning" that leverages machine learning techniques, specifically the Random Forest algorithm, to enhance decision-making in laptop purchases. Through rigorous testing, the model achieves a high level of accuracy with an R^2 score of approximately 0.88 and minimal Mean Absolute Error, showcasing its precision in predicting laptop prices. The system addresses the limitations of manual pricing methods by introducing a data-driven, automated approach for precise price estimation, supported by feature engineering processes and an intuitive web application built with Streamlit. This collaborative effort, with the support of the Department of Computer Science and Information Technology at Lendi Institute of Engineering and Technology, Vizianagaram, underscores the successful implementation of the Laptop

Price Prediction System, providing users with a valuable tool for making informed and budget-conscious decisions in laptop purchases [4].

Lokmane Akkouh's study on machine learning-based price prediction for laptops contributes to the e-commerce domain by developing a predictive model that utilizes regression algorithms to forecast laptop prices accurately and efficiently based on features like processor type, RAM, and brand. By collecting data from a Moroccan e-commerce platform and Kaggle, Akkouh's comprehensive dataset enables a robust analysis of the relationships between laptop specifications and prices. Despite the limited dataset, the model, particularly the GradientBoostingRegressor, shows promising results with high accuracy metrics. The study emphasizes the importance of data quantity in model performance and highlights the practical value of the model for both sellers and buyers in the laptop market. Akkouh's future research directions include deploying the model on a platform for practical use and continuous refinement to improve prediction accuracy, showcasing the evolving potential of machine learning in enhancing decision-making processes and user experiences in e-commerce settings [5].

CHAPTER 3: SYSTEM ANALYSIS

3.1 System Analysis

System analysis, the initial phase in the software development life cycle, involves understanding the requirements of a system and its environment. It entails breaking down the system into its component parts and grasping how each part interacts to fulfill the system's primary goal.

3.1.1 Requirement Analysis

The requirement analysis for the laptop price prediction project involved determining the essential functionalities required for users to input laptop specifications and obtain precise market valuations. This process included seeking guidance from industry professionals to identify the key features that impact laptop prices significantly. Additionally, it encompassed outlining the system's capabilities to guarantee an intuitive interface that meets the technical requirements of powerful machine learning algorithms.

System Requirements

The requirements and specifications of hardware and software components necessary to operate this voice assistant system are as follows:

3.1.2 Software Requirement

- Operating System: Windows/Linux
- Front-End Technologies: HTML, CSS, JavaScript, React
- Back-End Technologies: Flask, Python
- Data Science Platform: Jupyter Notebook
- IDE: Visual Studio Code

3.1.3 Hardware Requirement

- System: Multi-core Processor 2.4GHz or above
- Hard disk: 25GB above as per needed
- RAM: 4GB or above

3.1.3.1 Functional Requirements

A functional requirement is something a system must do. The functional requirement of this project is explained though Use Case Diagram which captures the service, task or function of the system. Use Case Diagram are employed in UML, a standard notation for the modeling of real-world objects and systems

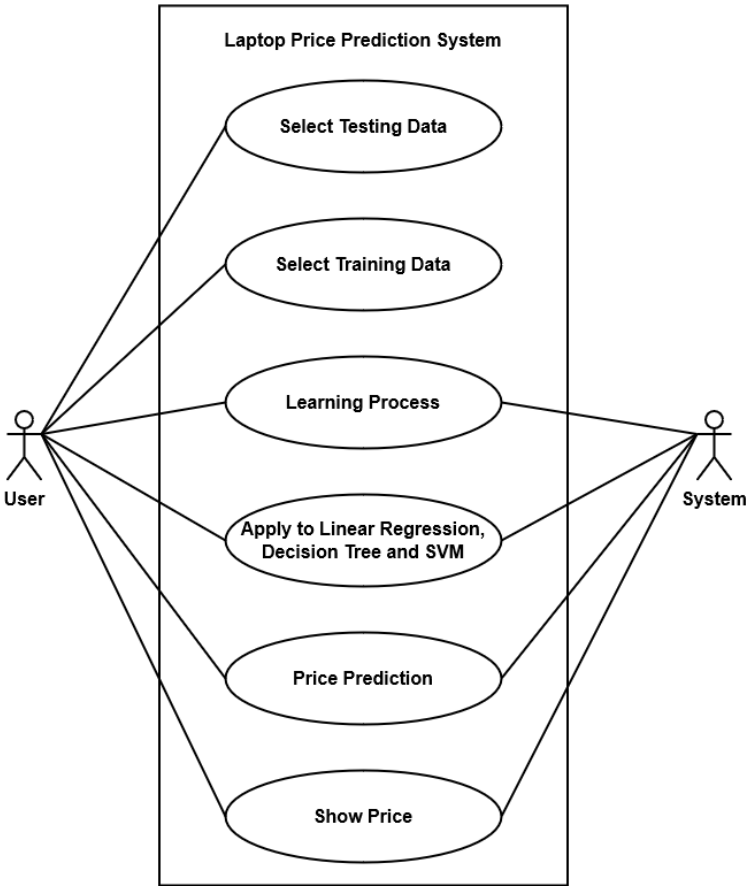


Figure 3.1 Use Case Diagram for Laptop Price Prediction

User-entered data is preprocessed by the System via removing outliers and replacing missing values, extracting important features and then applied to linear regression, decision tree and support vector machine model to predict and demonstrate user, the price of the laptop.

3.1.3.2 Non-functional requirement

A non-functional requirement is a specification that describes the system's operation capabilities and constraints that enhance its functionality. The non-functional requirements in content to the project are as follows

a. **Performance**

The system is able to provide price predictions within a reasonable response time, even with a large volume of concurrent users or high-dimensional input data.

b. **Scalability**

The system is scalable to accommodate an increasing number of users, data sources, and computational resources as the user base grows or additional features are added.

c. **Reliability**

The system is reliably under normal and peak load conditions, minimizing downtime and ensuring continuous availability to users.

d. **Accuracy**

The price predictions are generated by the system should exhibit a high degree of accuracy, minimizing errors and deviations from actual market prices.

e. **Security**

The system is implementing robust security measures to protect sensitive data, including user inputs and model parameters, from unauthorized access, modification, or disclosure.

f. **Privacy**

User privacy is respected, and the system should adhere to privacy regulations and best practices for handling personal information, such as anonymizing user data where possible.

g. **Maintainability**

The system is designed and implemented in a modular and maintainable manner, allowing for easy updates, bug fixes, and enhancements over time.

h. **Compatibility**

The system is compatible with a wide range of devices, browsers, and operating systems to ensure accessibility for users with diverse preferences and requirements.

3.1.4 Feasibility Analysis

A feasibility analysis for a laptop price prediction system involves assessing various aspects to determine the viability and practicality of implementing such a system. Here are some feasibility studies done in our system are as follows:

3.1.4.1 Technical Feasibility

All the tools and software product required to construct this project is easily available in the web. The application requires simple user interfaces but implementation real calculation of algorithm is complex. It can be done with assistance from our supervisor. The system requires technical equipment such as Laptops, Internet and Desktop. Any current device with Wi-Fi or data connectivity is essential to implement this project. Hence the system is technically feasible.

3.1.4.2 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. Statistics and report generated from this system are easier to read and understand. Hence, the system is feasible operationally.

3.1.4.3 Economic Feasibility

The purpose of economic feasibility is to determine the positive economic benefits. The system does not require additional software and hardware. The cost to conduct full system is negotiable because required information was collected from internet. The Python platform enables the use of free and open source which is free of cost. Hence, this system doesn’t require economic expenses while developing the system so it satisfies economic feasibility.

3.1.4.4 Schedule Feasibility

Schedule feasibility is the determination of whether a proposed project can be implemented fully within an estimated time frame. It helps to provide the possibility of the project to be completed within the specific time. If the project is completed within the estimated time, then the project is considered to be schedulable feasible.

Table 3.1: Project Schedule

Task Name	Duration	Start	Finish
Project Planning	7 days	Sun 12/3/23	Sun 12/10/23
Research	19 days	Mon 12/11/23	Thu 1/4/24
Analysis	12 days	Sun 1/7/24	Sat 1/20/24
Designing	12 days	Sat 1/27/24	Sun 2/11/24
Coding and Implementation	8 days	Thu 2/8/24	Mon 2/19/24
Testing and Debugging	8 days	Tue 2/20/24	Thu 2/29/24
Document	3 days	Fri 3/1/24	Tue 3/5/24

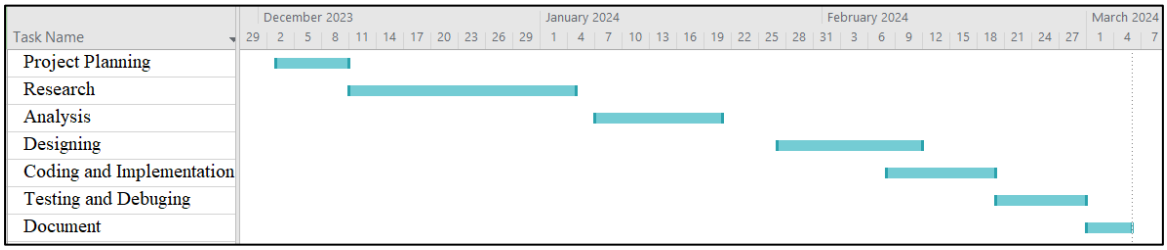


Figure 3.2: Gantt Chart

3.1.5 Analysis

3.1.5.1 Object modelling

A class diagram illustrates the structure of a system by showing the classes of objects, their attributes, operations, and the relationships among them. It provides a high-level overview of the system's design and the interactions between its components. Object modeling is a process used in software engineering to represent the real-world entities (objects) that exist in a system and the interactions between them. It involves creating a conceptual model of the system's structure and behavior, focusing on the objects that make up the system and their relationships. The object model representing using Class diagram is shown below:

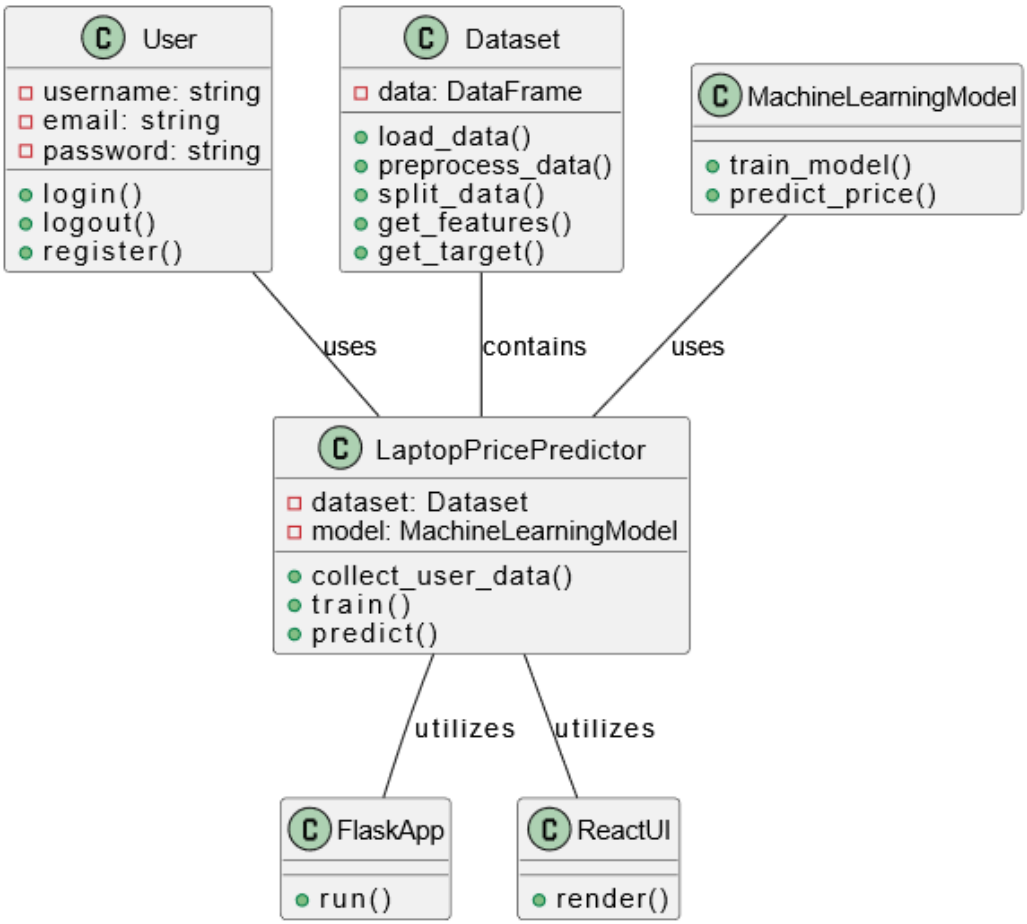


Figure 3.3: Class Diagram of the Prediction System

3.1.5.2 Dynamic modelling

Dynamic modeling using State and Sequence diagrams is a fundamental aspect of software design within the realm of object-oriented analysis and design (OOAD). These diagrams help depict the behavior of a system over time and during interactions between objects, respectively.

State Diagram

A state diagram, also known as a state machine diagram, represents the various states that an object or system can exist in, as well as the transitions between those states. States are represented as nodes, and transitions between states are depicted as arrows. State diagrams help visualize the behavior of a system over time by showing how it responds to various inputs and events.

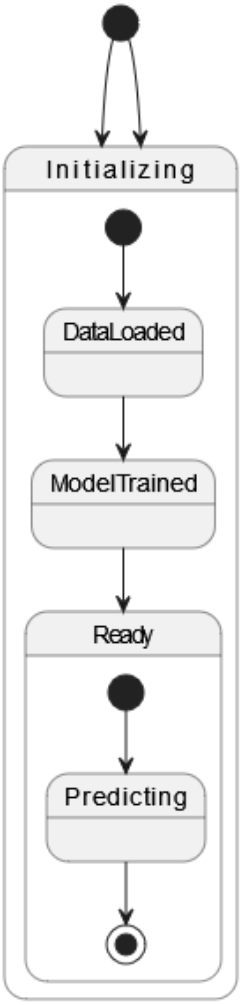


Figure 3.4: State Diagram of the Prediction System

Sequence Diagram

A sequence diagram is a type of interaction diagram that shows how processes operate with one another and in what order. It illustrates the flow of messages between objects or components of a system over time. Each vertical line in a sequence diagram represents an object or participant, and horizontal arrows depict the messages exchanged between them. Sequence diagrams are valuable for understanding the dynamic behavior of a system, including the order of interactions and the flow of control between components.

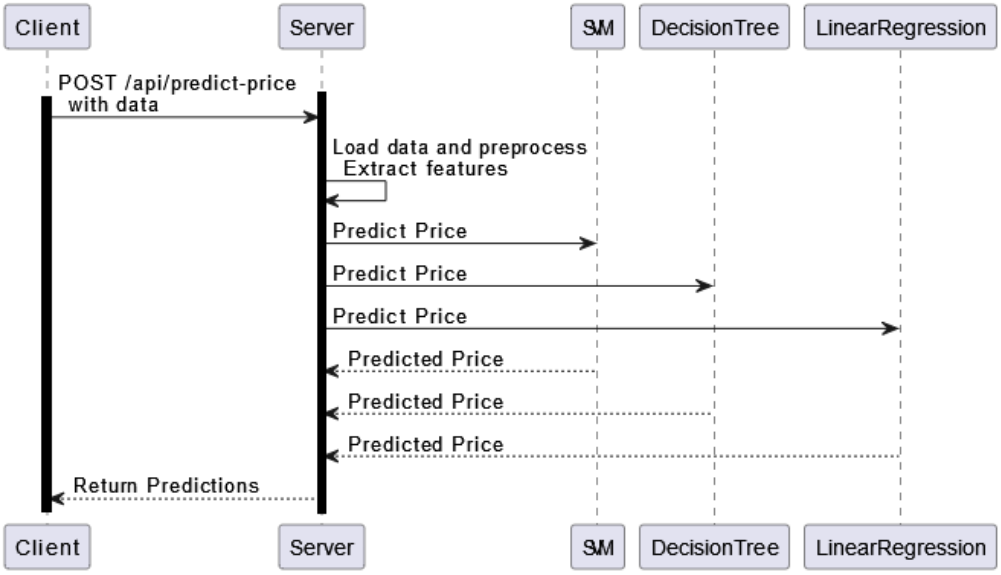


Figure 3.5: Sequence Diagram of the Prediction System

3.1.6 Process modelling

An activity diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that depicts the flow of activities or actions within a system or business process. It visually represents the sequence of activities, decisions, and interactions among components or actors in a system.

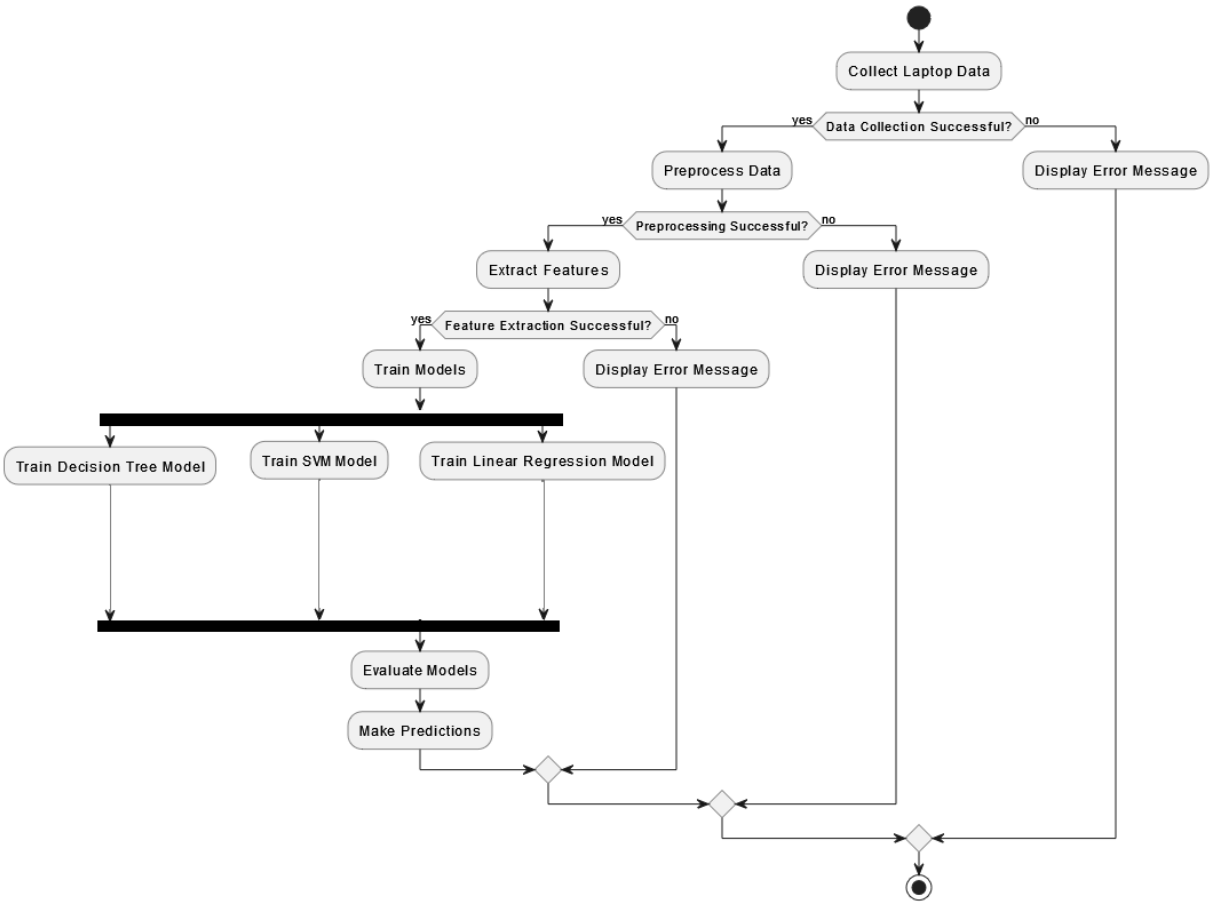


Figure 3.6: Activity Diagram for Prediction System

CHAPTER 4: SYSTEM DESIGN

4.1 Design

The project's detailed design takes into account the requirements established early in the application development process. To facilitate the development process, high level system design and system flowcharts were made during the design stage. These illustrations provide a thorough understanding of the structural and functional elements of the system. However, it's crucial to remember that the design component runs on its own and isn't connected to the code that is created as the project progresses.

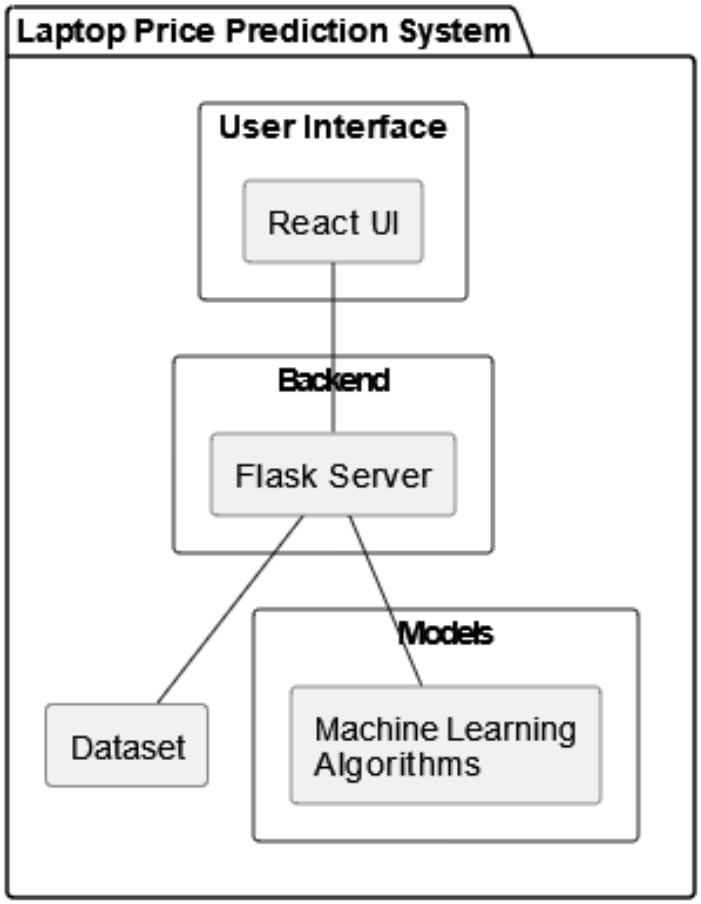


Figure 4.1: System Flow

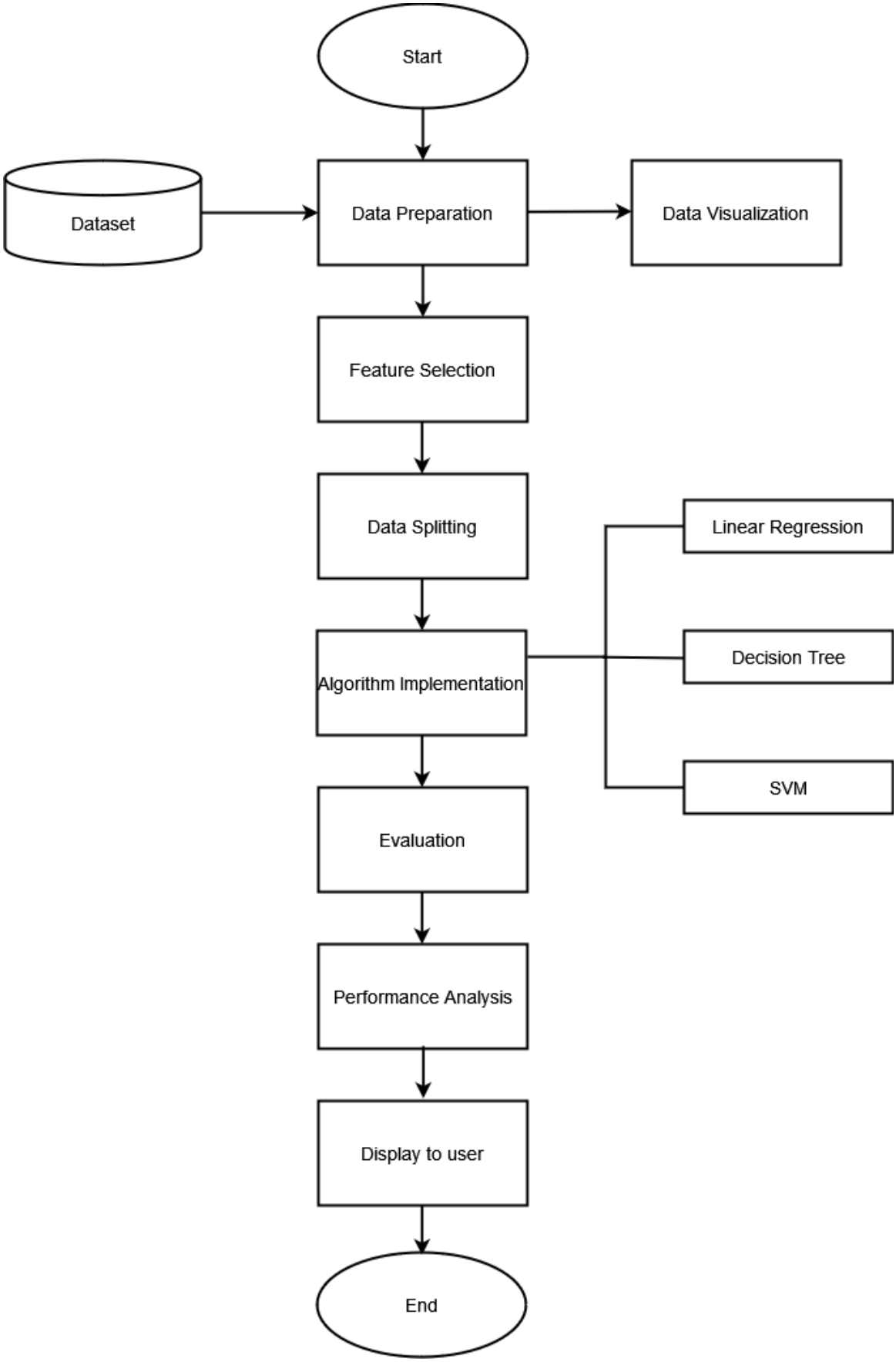


Figure 4.2: Flowchart of the System

4.2 Phases of Prediction System

The prediction system took the input from user in the form of text. System included data cleaning/preprocessing, column transformation, scaling, making pipeline and fitting. The phases of the prediction system are as follow:

4.2.1 Data Collection

In the data collection phase, the prediction system retrieves relevant datasets from Kaggle, a renowned platform for data science resources and competitions. Leveraging Kaggle's vast repository, the system obtains datasets pertinent to the prediction task, such as information about laptops and related features. The dataset is in tabular textual format consisting of 12 columns and 1303 rows.

4.2.2 Data Preprocessing

Data preprocessing is a crucial step in the data analysis and machine learning pipeline that involves transforming raw data into a clean and structured format suitable for further analysis and modeling. It encompasses a variety of techniques to address common issues and improve the quality of the data.

Steps performed for Data Preprocessing:

1. Identified and handled missing values in the dataset by imputing them with estimated values or removing them altogether.
2. Cleaned the data by removing duplicate records, correcting errors, and filtering out outliers that could distort analysis or modeling results.
3. Transformed categorical variables into numerical representations using techniques like one-hot encoding or label encoding.
4. Scaled numerical features to a similar range to prevent dominance of features with large magnitudes by normalization or standardization.
5. Performed feature engineering by creating new features or modifying existing ones to capture relevant information and enhance model performance.
6. Partitioned the dataset into training and testing sets while maintaining the distribution of target variables to ensure unbiased model evaluation.

```
In [2]: df = pd.read_csv('data.csv')

In [3]: df.head()

Out[3]:
```

	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

Figure 4.3: Data before Preprocessing

```
In [67]: df.head()

Out[67]:
```

	Company	TypeName	Ram	Weight	Price	Touchscreen	Ips	ppi	Cpu brand	HDD	SSD	Gpu brand	os
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983005	Intel Core i5	0	128	Intel	Mac
1	Apple	Ultrabook	8	1.34	47895.5232	0	0	127.677940	Intel Core i5	0	0	Intel	Mac
2	HP	Notebook	8	1.86	30636.0000	0	0	141.211998	Intel Core i5	0	256	Intel	Linux/Other/No OS
3	Apple	Ultrabook	16	1.83	135195.3360	0	1	220.534624	Intel Core i7	0	512	AMD	Mac
4	Apple	Ultrabook	8	1.37	96095.8080	0	1	226.983005	Intel Core i5	0	256	Intel	Mac

Figure 4.4: Data after Preprocessing

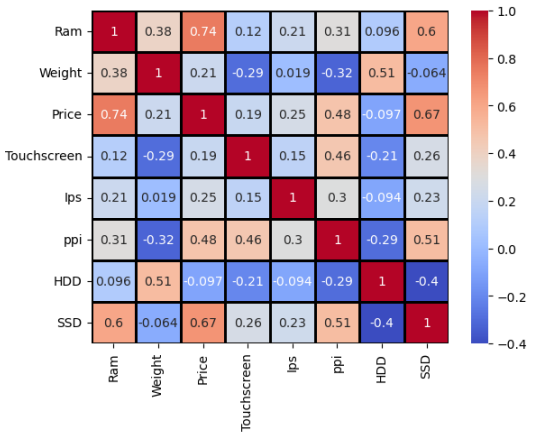


Figure 4.5: Correlation matrix of extracted features

4.2.3 User Interface Design

An interface has been developed using React.js. Within this React interface, users can find 13 select fields covering various parameters such as weight, screen size, company,

type, RAM, and more. This comprehensive setup ensures users have the flexibility to customize their selections according to their preferences.

To streamline the user experience, a predict button has been incorporated. Upon clicking this button, the React interface initiates an API call, which processes the selected parameters to generate a prediction. This seamless integration of user interaction and backend processing ensures a smooth and efficient prediction process.

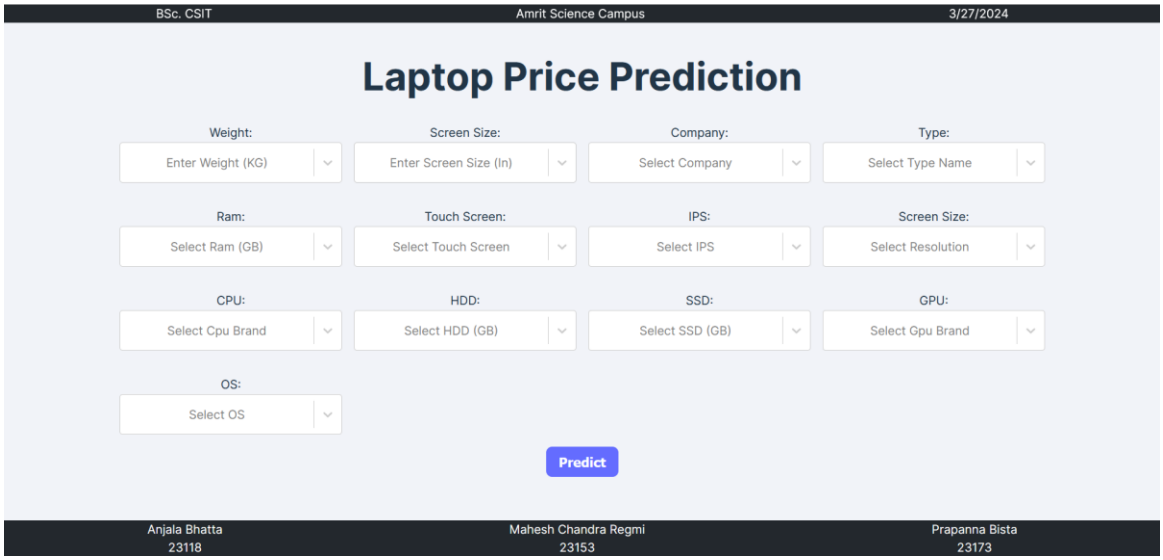


Figure 4.6: User Interface for Laptop Price Prediction

4.3 Component Diagram

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.

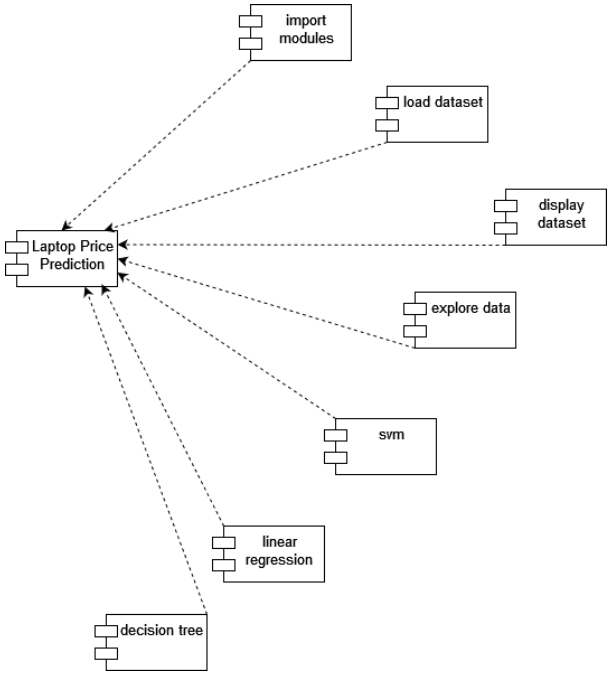


Figure 4.7: Component Diagram

4.4 Deployment Diagram

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

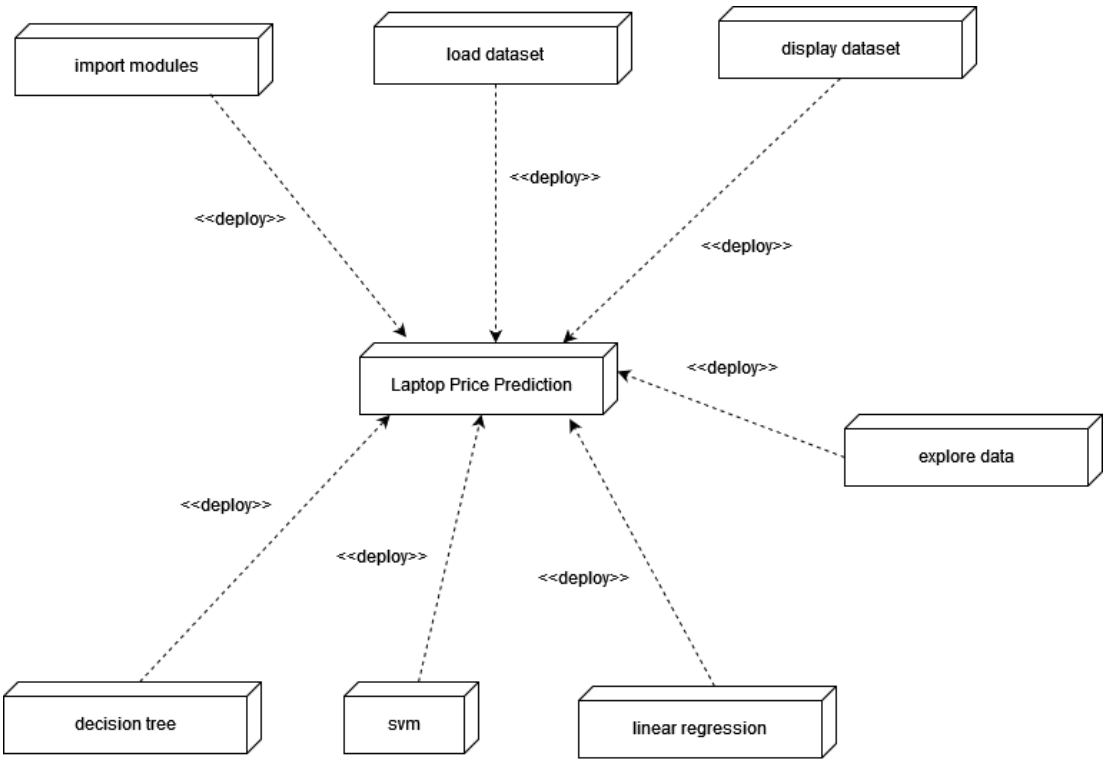


Figure 4.8: Deployment Diagram

4.5 Algorithm Details

Model implementation transforms a conceptual design into executable code, incorporating data preprocessing, model training, and evaluation stages to address a particular problem within a domain. Achieving success demands careful navigation through algorithm selection, data manipulation, and performance enhancement, as evidenced by the integration of SVM, decision tree, and support vector machine algorithms.

Linear Regression

Linear regression is a statistical method used for modeling the relationship between a dependent variable (often denoted as "Y") and one or more independent variables (often denoted as "X"). It assumes that the relationship between the variables can be approximated by a linear function.

The linear regression model can be represented by the equation:

$$Y=\beta_0+\beta_1X_1+\beta_2X_2+...+\beta_nX_n+\epsilon$$

Where:

- Y is the dependent variable.
- X_1, X_2,…, Xn are the independent variables.
- β_0 is the intercept (the value of Y when all independent variables are zero).
- β_1,β_2,...,βn are the coefficients (also called slopes) representing the change in Y for a unit change in each X variable, holding other variables constant.
- ϵ is the error term, representing the difference between the observed and predicted values of Y.

The goal of linear regression is to find the best-fitting line (or hyperplane, in higher dimensions) that minimizes the sum of the squared differences between the observed and predicted values of Y. This process is often done using the method of least squares.

Linear regression is widely used for various purposes, including prediction, forecasting, and understanding the relationship between variables.

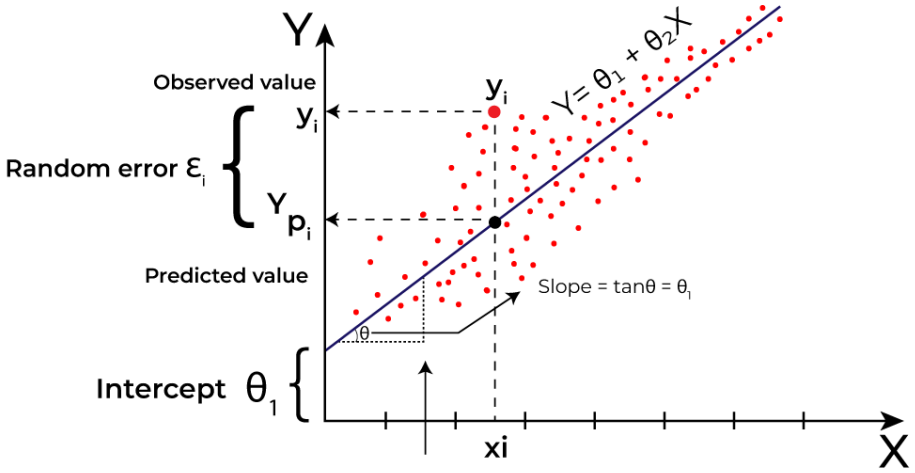


Figure 4.9: Linear Regression

Assumptions of Linear Regression

The basic assumptions of Linear Regression are as follows:

- **Linearity**: The relationship between the independent variables and the dependent variable is linear.

- **Independence of errors**: The errors of the model are independent of each other.
- **Homoscedasticity**: The variance of the errors is constant across all levels of the independent variables.
- **Normality of residuals**: The residuals are normally distributed.
- **No multicollinearity**: The independent variables are not highly correlated with each other.
- **No perfect multicollinearity**: There is no perfect linear relationship among the independent variables.
- **Additivity and linearity in parameters**: The effect of changes in the independent variables on the dependent variable is constant across all levels of the independent variables.

Decision Tree

A decision tree can be represented as a binary tree, where each internal node corresponds to a decision rule based on a particular feature, and each leaf node represents the predicted outcome or value.

1. **Tree Structure:** Let T denote the decision tree. T consists of a set of nodes, N, and a set of edges, E. Each node $n \in N$ has associated attributes:
    - feature(n): The feature used for splitting at node n.
    - threshold(n): The threshold value used for the splitting condition.
    - left(n) and right(n): Child nodes resulting from the splitting condition.
2. **Decision Rule at Node:** At each internal node n, a decision rule is applied based on the feature and threshold: If $\text{feature}(n) \leq \text{threshold}(n)$ then go to left(n) else go to right(n)
3. **Leaf Nodes:** Leaf nodes contain the predicted outcome or value. For a classification task, a leaf node may represent a class label c, while for regression, it may represent a predicted value y.
4. **Splitting Criterion:** The splitting criterion measures the impurity or uncertainty of the data at each node. For example, Gini impurity or entropy for classification, and mean squared error for regression.
5. **Tree Construction:** Given a dataset D with N samples and M features, the decision tree algorithm recursively selects the best feature and threshold to split

the data. This process minimizes impurity or error at each node until a stopping criterion is met.

6. **Pruning:** After tree construction, pruning techniques may be applied to reduce overfitting. This involves removing branches that do not significantly improve the tree's performance on validation data.

Mathematically, the decision tree algorithm aims to partition the feature space into regions R_1, R_2,...,R_K such that each region corresponds to a unique predicted outcome or value. The optimal partitioning is determined by minimizing a cost function, which depends on the task (classification or regression) and the chosen splitting criterion.

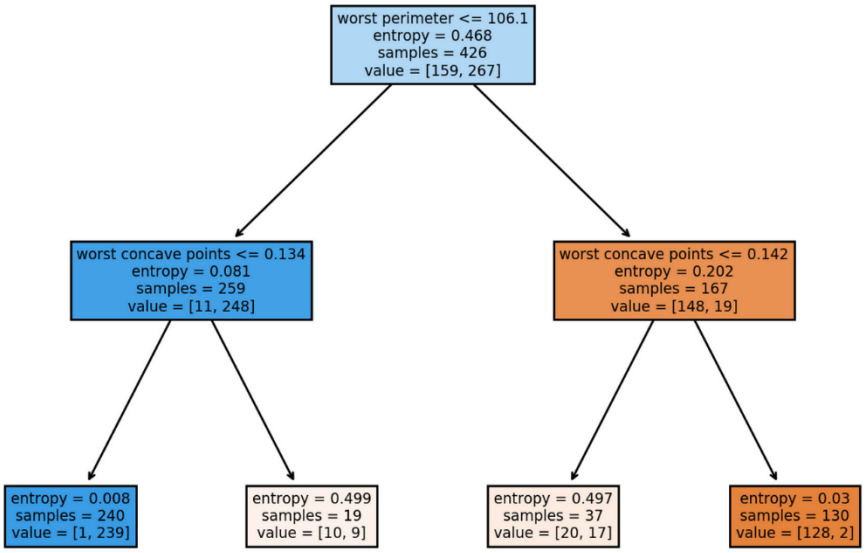


Figure 4.10: Decision Tree

Decision trees have fewer assumptions compared to some other machine learning algorithms like linear regression, but there are still some implicit assumptions and considerations to be aware of:

Partitioning Assumption: The algorithm assumes the dataset can be effectively partitioned into distinct regions based on feature values to predict the target variable accurately.

Feature Importance: It is assumed that features used for splitting are informative and relevant for predicting the target variable.

Orthogonal Decision Boundaries: Decision trees assume that the feature space can be partitioned using orthogonal decision boundaries.

Non-parametric Nature: Unlike parametric models, decision trees do not impose explicit assumptions about the functional form of the relationship between features and the target variable.

Data Representation: Decision trees can handle both numerical and categorical data types effectively without requiring transformation.

Overfitting Concerns: There's an implicit assumption that pruning techniques or ensemble methods will be used to address overfitting and improve generalization performance.

Sample Representativeness: The algorithm assumes that the dataset is representative of the population, without significant biases or missing information.

Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates different classes in the feature space.

Given a training dataset (x_i,y_i), where x_i is the feature vector and y_i is the class label ($y_i \in \{-1,+1\}$ for binary classification), the goal of SVM is to find the hyperplane $w^T x+b=0$ that best separates the data points into different classes.

Hyperplane: The hyperplane is defined by the weights vector w and the bias term b, where w is perpendicular to the hyperplane and determines its orientation, and b shifts the hyperplane away from the origin.

Margin: The distance between the hyperplane and the closest data points (support vectors) is the margin. Mathematically, the margin is $2/\|w\|$, where $\|w\|$ is the Euclidean norm of the weight vector w.

Optimization Objective: SVM aims to maximize the margin while minimizing the classification error. This is typically formulated as the optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$$

Subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \text{for } i = 1, 2, ..., N$$

where ξi are slack variables that allow for some misclassification, and C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error.

Kernel Trick: To handle non-linearly separable data, SVM employs the kernel trick, which implicitly maps the input features into a higher-dimensional space. The optimization problem is then solved in this higher-dimensional space using the kernel function K(xi,xj), where K computes the dot product between the mapped feature vectors.

Decision Function: The decision function of SVM is given by $f(x)=\text{sign}(w^Tx+b)$, where signs return the class label (+1 or -1) based on whether the input x lies on the positive or negative side of the hyperplane.

SVM seeks to find the optimal hyperplane that maximizes the margin between the classes while minimizing the classification error, thus providing a robust and effective solution for classification tasks.

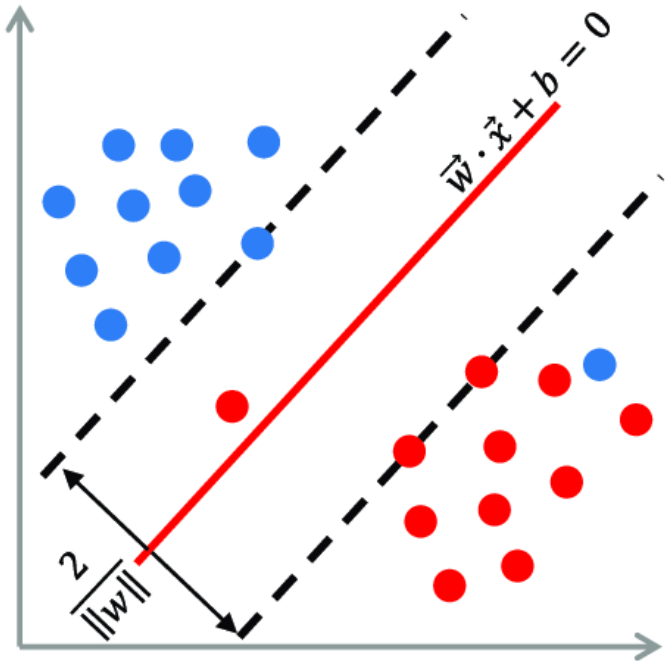


Figure 4.11: Support Vector Machine

Assumptions of Support Vector Machines (SVM):

- **Linear Separability:** SVM assumes linear separability of classes or the ability to find a hyperplane that separates classes in the feature space.
- **Feature Scaling:** Feature scaling is assumed for optimal SVM performance, ensuring no single feature dominates the optimization process.
- **Margin Maximization:** SVM aims to maximize the margin between support vectors of different classes, promoting a robust decision boundary that generalizes well to unseen data.
- **Kernel Function Choice:** The choice of kernel function is critical in SVM, and its performance can be sensitive to the selection and tuning of kernel parameters.
- **Binary Classification:** SVM is inherently a binary classification algorithm, although it can be extended to handle multi-class classification problems through various strategies.
- **Sparse Solutions:** Sparse solutions are favored in SVM, meaning only a subset of training instances (support vectors) influences the decision boundary, aiding computational efficiency.

- **Statistical Assumptions:** SVM does not rely on explicit probabilistic assumptions about data distribution, instead focusing on finding an optimal decision boundary based on the geometry of the feature space.

Regression Evaluation Metrics:

Regression evaluation metrics are used to assess the performance of regression models in predicting continuous outcomes. Here are some commonly used regressions evaluation metrics:

Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted values and the actual values. It is calculated as the average of the absolute differences between predicted and actual values:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|$$

where n is the number of samples, y_i is the actual value, and $\widehat{y_i}$ is the predicted value.

R-squared (Coefficient of Determination): R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 1 indicates a perfect fit:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y}$ is the mean of the observed values.

These regression evaluation metrics provide different perspectives on the performance of the regression model and help assess its accuracy, precision, and goodness of fit. It's important to choose the appropriate metric based on the specific requirements of the problem and the desired interpretation of the model's performance.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Tool Used

- **Anaconda and Jupyter Notebook:** The project utilizes Anaconda and Jupyter Notebook for an integrated data science environment. Anaconda streamlines Python package management, while Jupyter Notebook facilitates interactive data exploration and analysis.

- **HTML, CSS and React:** HTML and CSS are employed for structuring and styling web pages, while React.js enhances the user interface with dynamic and interactive components, contributing to an engaging user experience.

- **Python and Flask:** Python serves as the primary programming language for implementing server-side logic, while Flask, a lightweight web framework, handles HTTP requests and responses. This combination enables the development of RESTful APIs and dynamic content delivery.

- **IDE: Visual Studio Code (VSCode):** VSCode is utilized as the Integrated Development Environment (IDE), offering essential tools for coding, debugging, and version control. Its extensive extension ecosystem further enhances productivity with tailored functionalities.

Beside these tools we've used tools like github for tracking and sync code of one another, draw.io to create diagram required for project and thunder client for API testing. By leveraging these tools and technologies, the project achieves a comprehensive application stack, facilitating efficient development and seamless integration between front-end and back-end components. This setup ensures scalability, maintainability, and an optimal user experience throughout the project lifecycle.

5.2 Testing

Software testing is a process of evaluating a software application or system to ensure that it meets specified requirements and works correctly. It involves identifying defects or

errors in the software and verifying that the software behaves as expected under various conditions.

The primary objectives of software testing include:

1. **Finding Defects:** Software testing helps identify defects or bugs in the software, such as incorrect functionality, unexpected behavior, or performance issues.
2. **Ensuring Quality:** Testing aims to ensure that the software meets quality standards and satisfies user expectations. It helps in delivering a reliable, robust, and user-friendly product.
3. **Validating Requirements:** Testing verifies that the software meets the specified requirements and functions as intended. It ensures that the software fulfills the needs of its users and stakeholders.
4. **Preventing Defects:** By identifying defects early in the development process, testing helps prevent issues from reaching production and minimizes the cost and effort of fixing them later.
5. **Improving Maintainability:** Testing contributes to the maintainability of the software by providing feedback on code quality, architecture, and design. It helps in identifying areas for improvement and refactoring.

5.2.1 Unit Testing

Unit testing is a software testing technique where individual units or components of a software application are tested in isolation. A unit is the smallest testable part of an application, such as a function, method, or class. The purpose of unit testing is to validate that each unit of the software performs as expected and behaves correctly according to its design. In unit testing, the entire system is structured into modular components, and each module undergoes individual testing. The process involves focusing on one module at a time, thoroughly testing it until the desired and accurate output is achieved. This iterative approach allows developers to refine and fine-tune each module until it consistently produces the expected results.

Table 5.1: Unit Testing

Test Cases	Steps to be Executed	Expected Results	Actual Results	Pass/Fail
Estimated Price	-Enters incomplete or negative value of weight -Click Estimate Price	Show the alert box with message "please enter a valid weight"	Price not Estimated	Fail
Estimated Price	-Enters incomplete or negative value of weight -Click Estimate Price	Estimates the price	Price is Estimated	Pass
User Login	-Enter wrong credentials -Click login button	Show the alert box with message "Invalid email or password"	User is not logged in	Fail
User Login	-Enter current credentials -Click login button	Logged in user	User is logged in	Pass

5.2.2 System Testing

System testing is a level of software testing where the complete and integrated software application is tested as a whole. It aims to verify that the entire system meets specified requirements and functions correctly in its intended environment. System testing is typically performed after integration testing, where individual components or modules are combined and tested together.

We've used Thunder Client (VS code integrated API testing tool) for system testing. Http GET and POST end points have been tested successfully.

Demonstration of API testing:

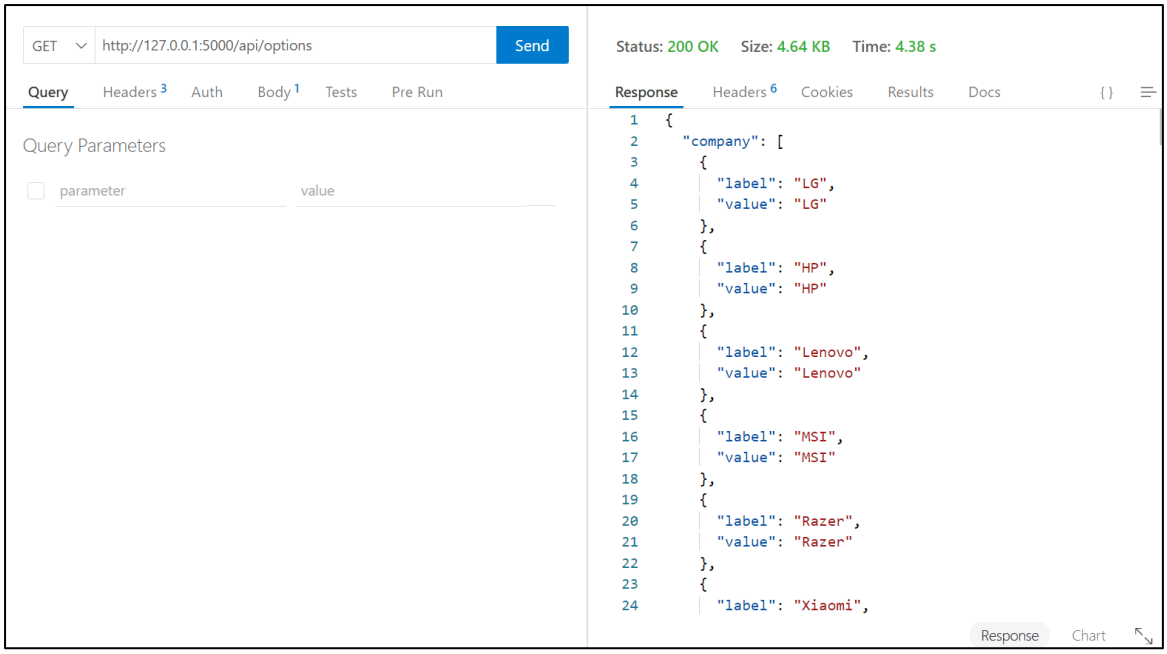


Figure 5.1: Http GET Protocol is tested

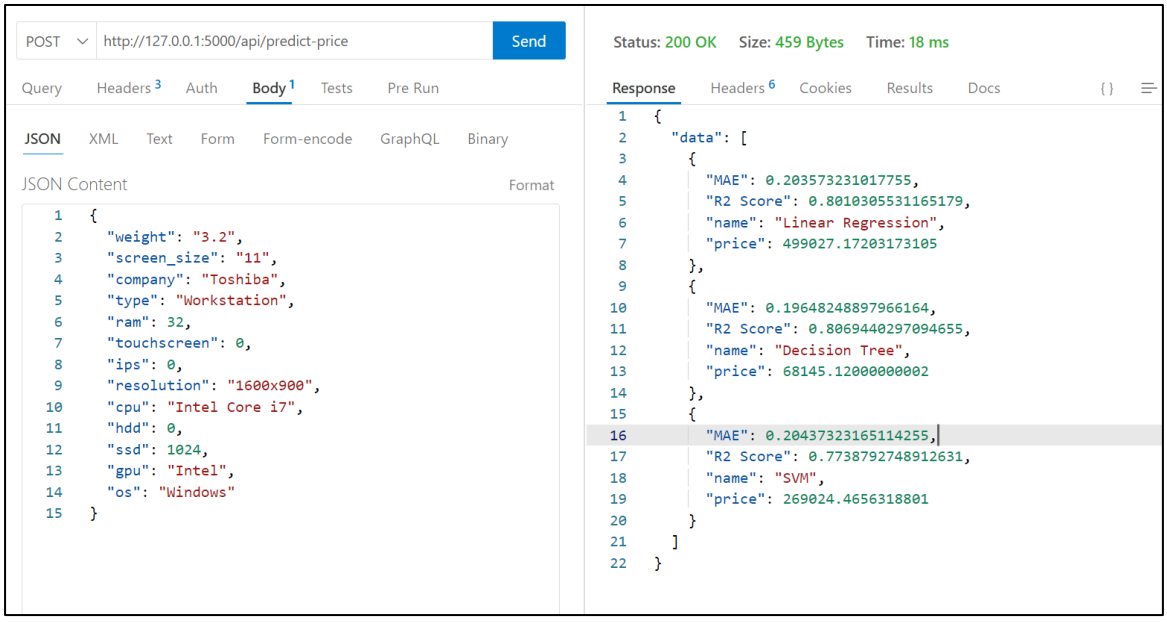


Figure 5.2: Http POST protocol is tested

5.2.3 Result Analysis

Mean absolute error is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.

R square (R^2) or Coefficient of Determination It shows percentage variation in y which is explained by all the x variables together. Higher the better. It is always between 0 and 1. It can never be negative – since it is a squared value.

Table 5.2: Result Analysis of algorithms

S.N.	**Methods**	**MAE**	**R^2 Score**
1	Linear Regression	0.203573	0.801030
2	Decision Tree	0.201348	0.802387
3	SVM	0.204373	0.773879

Note:

MAE = 10 implies that, on average, the Prediction's distance from the true value is 10 (e.g., true value is 100 and forecast is 90 or true value is 100 and forecast is 110 would be a distance of 10).

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1.1 Conclusion

In this study, we employed Support Vector Machine (SVM) with linear regression and Decision Tree Regression as alternative approaches to predicting laptop prices, with Linear Regression serving as our baseline algorithm. Through comprehensive experimentation and analysis, we gained valuable insights into the relative performance of these algorithms.

Linear Regression, our baseline algorithm, demonstrated a solid performance in predicting laptop prices, achieving a reasonable Mean Absolute Error (MAE) and R-squared (R2) score. Its simplicity and interpretability make it an attractive choice for straightforward regression tasks. However, Linear Regression's performance may be limited by its assumption of linear relationships between features and target variables, potentially leading to suboptimal predictions for datasets with complex relationships.

Support Vector Machine (SVM) with linear regression offered competitive performance compared to Linear Regression, with similar MAE and R2 scores. SVM with linear regression demonstrated robustness against outliers and noise in the dataset and was able to capture linear relationships effectively. Additionally, SVM with linear regression exhibited flexibility in handling high-dimensional data, making it a suitable choice for laptop price prediction tasks with a large number of features.

Decision Tree Regression, while providing comparable results to Linear Regression and SVM with linear regression, showed a tendency to overfit the training data, especially in the presence of complex feature interactions. Despite this, Decision Tree Regression offered interpretability and simplicity in model understanding, allowing for clear insights into the decision-making process behind price predictions. However, addressing overfitting through techniques such as pruning or ensemble methods is essential to enhance the generalization capability of Decision Tree Regression.

Overall, our analysis suggests that Linear Regression, SVM with linear regression, and Decision Tree Regression each offer unique strengths and weaknesses in the context of

laptop price prediction. Linear Regression serves as a solid baseline with simplicity and interpretability, while SVM with linear regression provides flexibility and robustness against outliers. Decision Tree Regression offers interpretability but requires careful handling to prevent overfitting.

In conclusion, the selection of the most suitable algorithm depends on factors such as the dataset characteristics, desired level of interpretability, and computational resources available. Future research may explore ensemble techniques or hybrid models combining the strengths of these algorithms to further enhance prediction accuracy and robustness in laptop price prediction tasks.

6.1.2 Future Recommendation

Based on the analysis of our study on predicting laptop prices using Linear Regression, Support Vector Machine (SVM) with linear regression, and Decision Tree Regression, the following recommendations are proposed:

1. **Model Evaluation and Selection**: Before integrating predictive models into e-commerce platforms like Daraz, Neostore, and Itti, it is essential to conduct thorough evaluation and comparison of the models' performance. Assess metrics such as Mean Absolute Error (MAE), R-squared (R2) score, and computational efficiency to determine the most suitable model for deployment.
2. **Data Quality and Feature Engineering**: Invest in data quality assurance measures and feature engineering techniques to improve the accuracy and robustness of predictive models. Ensure that the dataset used for training the models is comprehensive, clean, and representative of the target market. Explore additional features or transformations that could enhance predictive performance.
3. **Real-Time Integration**: Develop mechanisms for real-time integration of predictive models with e-commerce platforms. Implement APIs or web services that allow for seamless communication between the predictive models and the platforms, enabling dynamic pricing updates and personalized recommendations for users.
4. **User Experience Enhancement**: Prioritize user experience (UX) by designing intuitive interfaces and interactive features that showcase price predictions and product recommendations derived from the integrated models. Ensure that the

integration enhances the overall shopping experience and provides valuable insights to users.

5. **Continuous Monitoring and Optimization**: Establish processes for continuous monitoring and optimization of integrated predictive models. Regularly track key performance indicators (KPIs) such as conversion rates, average order value (AOV), and customer satisfaction scores to identify areas for improvement and refinement.
6. **Feedback Mechanism Implementation**: Implement feedback mechanisms within the e-commerce platforms to gather user feedback and preferences. Leverage this feedback to refine the predictive models and tailor price predictions and recommendations to better meet user needs and preferences.
7. **Security and Privacy Considerations**: Prioritize security and privacy considerations when integrating predictive models with e-commerce platforms. Implement robust security measures to protect sensitive customer data and ensure compliance with data protection regulations such as GDPR and CCPA.
8. **Collaboration with Stakeholders**: Foster collaboration with stakeholders including e-commerce platform providers, data scientists, marketers, and business analysts. By working collaboratively, leverage domain expertise and insights to optimize the integration of predictive models and drive business outcomes.
9. **Performance Tracking and Reporting**: Implement mechanisms for tracking and reporting the performance of integrated predictive models. Generate regular reports that highlight key metrics, trends, and insights derived from the models' predictions to facilitate data-driven decision-making.
10. **Scalability and Flexibility**: Design the integration architecture to be scalable and flexible, capable of accommodating future growth and changes in data volume and complexity. Ensure that the integrated solution can adapt to evolving business requirements and technological advancements.

In summary, by following these recommendations, businesses can effectively integrate predictive models into e-commerce platforms to enhance pricing strategies, improve user experience, and drive business growth. By leveraging data-driven insights and fostering collaboration across teams, organizations can unlock the full potential of predictive modeling in the e-commerce domain.

REFERENCES

[1] V. Surjuse, S. Lohakare, A. Barapatre, and A. Chapke, “Laptop Price Prediction using Machine Learning,” International Journal of Computer Science and Mobile Computing, vol. 11, no. 1, pp. 164–168, Jan. 2022, doi: https://doi.org/10.47760/ijcsmc.2022.v11i01.021.

[2] A. D. Siburian et al., “Laptop Price Prediction with Machine Learning Using Regression Algorithm,” Jurnal Sistem Informasi dan Ilmu Komputer Prima(JUSIKOM PRIMA), vol. 6, no. 1, pp. 87–91, Sep. 2022, doi: https://doi.org/10.34012/jurnalsisteminformasidanilmukomputer.v6i1.2850.

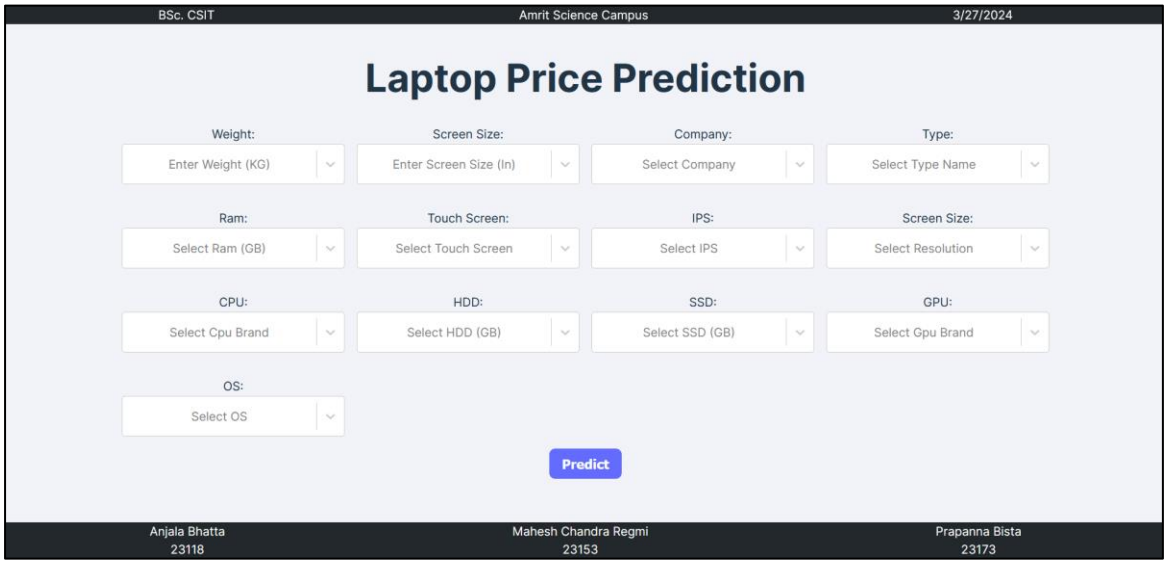
[3] V. Sarala, P. Sai, and P. Kumar, “Laptop Price Prediction.” Available: https://ijesat.com/ijesat/files/V23I9047_1694756701.pdf

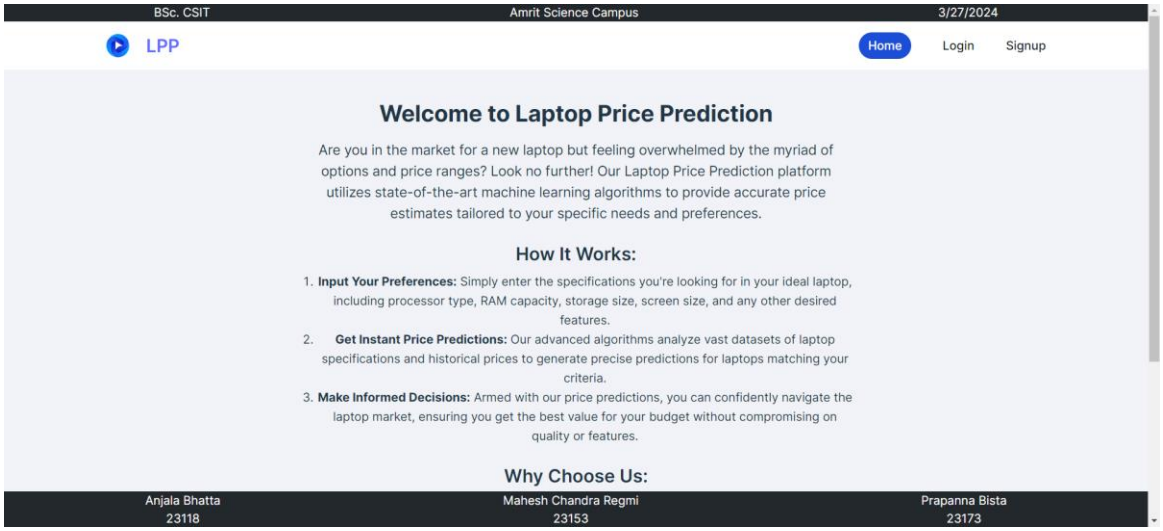
[4] “Laptop Price Estimation Using Machine Learning | International Journal of Research in Engineering, Science and Management,” journal.ijresm.com, Mar. 2024, Accessed: Mar. 28, 2024. [Online]. Available: https://journal.ijresm.com/index.php/ijresm/article/view/2961

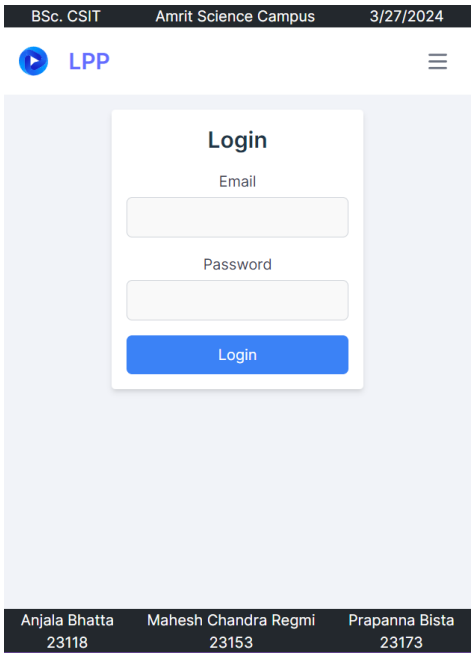
[5] L. Akkouh, "Machine Learning-Based Price Prediction for Laptops," Article, Jan. 2024. [Online]. Available: https://www.researchgate.net/publication/377721653.

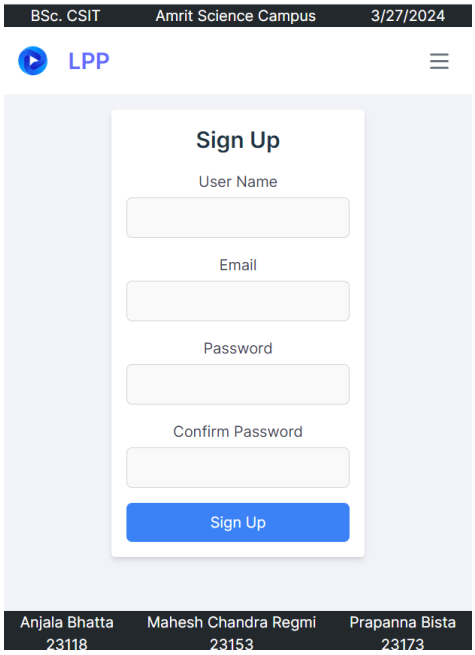
Appendix

Price Prediction with different input to the system:




BSc. CSIT
Amrit Science Campus
3/27/2024
LPP
Home
Login
Signup
Welcome to Laptop Price Prediction
Are you in the market for a new laptop but feeling overwhelmed by the myriad of options and price ranges? Look no further! Our Laptop Price Prediction platform utilizes state-of-the-art machine learning algorithms to provide accurate price estimates tailored to your specific needs and preferences.
How It Works:
1. Input Your Preferences: Simply enter the specifications you're looking for in your ideal laptop, including processor type, RAM capacity, storage size, screen size, and any other desired features.
2. Get Instant Price Predictions: Our advanced algorithms analyze vast datasets of laptop specifications and historical prices to generate precise predictions for laptops matching your criteria.
3. Make Informed Decisions: Armed with our price predictions, you can confidently navigate the laptop market, ensuring you get the best value for your budget without compromising on quality or features.
Why Choose Us:
Anjala Bhatta 23118
Mahesh Chandra Regmi 23153
Prapanna Bista 23173


BSc. CSIT
Amrit Science Campus
3/27/2024
LPP
Login
Email
Password
Login
Anjala Bhatta 23118
Mahesh Chandra Regmi 23153
Prapanna Bista 23173


BSc. CSIT
Amrit Science Campus
3/27/2024
LPP
Sign Up
User Name
Email
Password
Confirm Password
Sign Up
Anjala Bhatta 23118
Mahesh Chandra Regmi 23153
Prapanna Bista 23173

Source Code:

Linear Regression Code:

```
class CustomLinearRegression:
    def __init__(self):
        self.theta = None

    def fit(self, X, y):
        """Fit the linear regression model."""
        # Get the number of samples and features
        n_samples = len(X)
        n_features = len(X[0])


        X_augmented = [[1] + list(x) for x in X]
X_transpose = [[row[i] for row in X_augmented] for i in
range(n_features + 1)]


        theta_numerator = [sum(x * y_i for x, y_i in zip(x_row, y)) for
x_row in X_transpose]
        theta_denominator = [[sum(x_i * x_j for x_i, x_j in
zip(x_row_i, x_row_j)) for x_row_j in X_transpose] for x_row_i in
X_transpose]
        self.theta = self._solve_linear_system(theta_denominator,
theta_numerator)

    def _solve_linear_system(self, A, b):
        """Solve the system of linear equations Ax = b"""
        n = len(A)
        x = [0] * n

        for i in range(n):
            s = sum(-A[i][j] * x[j] for j in range(i))
            x[i] = (b[i] + s) / A[i][i]

        return x

    def predict(self, X):
        """Make predictions using the trained model."""
        # Add a column of ones to X for the bias term
        X_augmented = [[1] + list(x) for x in X]

        # Make predictions using the learned parameters
        y_pred = [sum(x_row[i] * theta_i for i, theta_i in
enumerate(self.theta)) for x_row in X_augmented]
        return y_pred
```

Linear Regression Pipeline:

```
# Define the ColumnTransformer
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse_output=False, drop='first'), [0,
1, 7, 10, 11])
], remainder='passthrough')

# Create an instance of LinearRegressionCustom
step2 = CustomLinearRegression()

# Create the Pipeline
pipe = Pipeline([
    ('step1', step1),
    ('step2', step2)
])

pipe_lr = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

# Fit the Pipeline to the training data
pipe.fit(X_train, y_train)


# Make predictions
y_pred = pipe.predict(X_test)

# Evaluate the model using custom functions
r2 = r2_score_custom(y_test, y_pred)
mae = mean_absolute_error_custom(y_test, y_pred)

# Print the evaluation metrics
print('R2 score:', r2)
print('MAE:', mae)
```

Decision Tree Pipeline

```
# Define the ColumnTransformer
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse_output=False, drop='first'), [0,
1, 7, 10, 11])
], remainder='passthrough')

step2 = DecisionTreeRegressor(max_depth=8)
# Create the Pipeline
pipe = Pipeline([
    ('step1', step1),
    ('step2', step2)
])

pipe_dt = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

# Fit the Pipeline to the training data
pipe.fit(X_train, y_train)


# Make predictions
y_pred = pipe.predict(X_test)

# Evaluate the model using custom functions
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Print the evaluation metrics
print('R2 score:', r2)
print('MAE:', mae)
```

SVM Pipeline

```
# Define the ColumnTransformer
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse_output=False, drop='first'), [0,
1, 7, 10, 11])
], remainder='passthrough')


step2 = SVR(kernel='rbf',C=10000,epsilon=0.1)
# Create the Pipeline
pipe = Pipeline([
    ('step1', step1),
    ('step2', step2)
])

pipe_svm = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

# Fit the Pipeline to the training data
pipe.fit(X_train, y_train)


# Make predictions
y_pred = pipe.predict(X_test)

# Evaluate the model using custom functions
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Print the evaluation metrics
print('R2 score:', r2)
print('MAE:', mae)
```