**TABLE OF CONTENTS**

**ABSTRACT**

**1. SYSTEM STUDY**

1.1 Feasibility Study

1.1.1 Economical Feasibility

1.1.2 Technical Feasibility

1.1.3 Social Feasibility

**1.2 SYSTEM REQUIREMENTS**

1.2.1 Hardware Requirements

1.2.2 Software Requirements

**2. SYSTEM ANALYSIS**

2.1 Introduction

2.2 Project Description

2.3 Modules

2.4 Existing System

2.4.1 Drawbacks of Existing System

2.5 Proposed system

2.5.1 Advantage of Proposed System

**3. SYSTEM DESIGN**

3.1 Data flow Diagram

3.2 Case Diagram

3.3 Architecture Diagram

3.4 Source Code

**4. TESTING**

4.1 Unit Testing

4.2 Integration Testing

4.3 Validation Testing

4.4 Acceptance Testing

**5. SYSTEM TESTING & IMPLEMENTATION**

5.1 Introduction

5.2 Strategic approach to Software testing

5.3 Software Environment

**6. RESULTS AND CONCLUSIONS**

6.1 Results

6.2 Conclusion AND Future Enhancements

**ABSTRACT**

In recent years, with the advancement of technology, laptops have become an essential part of our lives. With so many options available in the market, it can be overwhelming for consumers to choose the right laptop that fits their needs and budget. This is where laptop price prediction comes in.

laptop price prediction is the process of using machine learning algorithms to analyze and predict the price of laptops based on various factors such as brand, specifications, features, and historical sales data. By predicting the price of laptops, consumers can make informed decisions about purchasing a laptop that meets their budget constraints.

The benefits of laptop price prediction extend beyond just helping consumers make informed decisions about purchasing laptops. Retailers can also use this technology to optimize their pricing strategies and maximize profits. By using laptop price prediction, retailers can price their laptops competitively and offer personalized discounts to customers, leading to increased sales and customer satisfaction.

laptop price prediction is a powerful tool that helps consumers and retailers make informed decisions about pricing and purchasing laptops. With the increasing demand for laptops and the ever-changing technology landscape, This technology is set to become even more Important in the future.

**CHAPTER -1**

**SYSTEM STUDY**

* 1. **Feasibility Study :**

The purpose of this feasibility study is to determine whether a laptop price prediction project is viable or not. This study will analyze the technical, economical, and social feasibility of the project.

**1.1.1 Technical Feasibility :**

The technical feasibility of the project will be analyzed by determining whether the technology required for laptop price prediction is available or not. We will need to collect data from various sources such as laptop retailers, manufacturers, and online marketplaces. We will also need to use machine learning algorithms for predicting laptop prices. After analyzing the availability of required technology, it has been concluded that the project is technically feasible.

**1.1.2 Economical Feasibility:**

The economical feasibility of the project will be analyzed by determining whether the project is financially viable or not. We will need to analyze the cost of hardware, software, and human resources required for the project. After analyzing the costs, we have concluded that the project is financially feasible as the expected revenue from the project is higher than the estimated cost.

**1.1.3 Social Feasibility:**

The social feasibility of the project will be analyzed by determining whether the project is socially acceptable or not. We will need to analyze whether the project will violate any social norms or values. As the project does not violate any social norms or values, it has been concluded that the project is socially feasible.

**Conclusion** : After analyzing the technical, economical, and social feasibility of the laptop price prediction project, it has been concluded that the project is viable and can be initiated.

**1.2 System Requirements**

**1.2.1 Hardware Requirements**

* Processor - Intel Core i5 or equivalent
* RAM - 8GB or higher
* Hard Disk - 256GB or higher
* Graphics Card - NVIDIA or AMD with 2GB VRAM
* Display - 15-inch Full HD or higher

**1.2.2 Software Requirements**

* Operating System: Windows 8/10/11
* Programming Language: Python 3.x
* IDE: Jupyter Notebook, PyCharm
* Libraries: Scikit-learn, Pandas, Numpy, Matplotlib

**CHAPTER-2**

**SYSTEM ANALYSIS**

**2.1 Introduction**

In today's fast-paced and ever-changing world, technology has become an integral part of our daily lives. One of the most significant developments in recent times is the growth of machine learning and its various applications. Machine learning has the potential to revolutionize the way we make decisions and process information. In this context, our project aims to develop a laptop price prediction model using machine learning algorithms.

The laptop market is highly competitive, with a vast range of brands and models available in different price ranges. As a result, choosing the right laptop can be a challenging task, especially for those with limited knowledge about technical specifications and pricing trends. To address this issue, we propose a machine learning-based model that can predict the price of laptops based on their technical specifications.

Our model will use a dataset of laptops with various technical features such as processor speed, RAM size, storage capacity, display size, and graphics card. By training the model on this dataset, it will learn to identify patterns and correlations between these features and the price of laptops. Once the model is trained, it can be used to predict the price of any new laptop based on its technical specifications.

We will be using the Python programming language for implementing the machine learning algorithms and various libraries such as Scikit-learn and NumPy for data processing and model training

Overall, our project aims to provide a valuable tool for consumers, enabling them to make informed decisions about purchasing laptops based on their technical specifications and predicted prices. It also serves as an excellent example of how machine learning algorithms can be used to solve real-world problems and make our lives easier.

**2.2**  **Project Description:**

The laptop price prediction system is a machine learning-based application that predicts the price of laptops based on their features and specifications. The system uses a random forest algorithm to analyze historical data on laptop prices and their corresponding features, such as processor speed, RAM, storage capacity, screen size, and brand.

The main objective of the system is to provide a reliable and accurate prediction of the price of a laptop based on its features. This can help buyers to make informed decisions when purchasing a laptop, and can also assist sellers in setting competitive prices for their products.

The system will take input from the user in the form of the laptop features, and then use the trained machine learning model to predict the price range of the laptop. The system will also display similar laptops with their prices to give the user a better idea of the market trends.

The software will be developed using Python programming language and will utilize the scikit-learn library for machine learning.

The laptop price prediction system will be a user-friendly and efficient tool for predicting laptop prices based on their features, and will provide valuable insights into the laptop market.

**2.3 Modules**

**Data Collection :** This module is responsible for collecting the required data for laptop price prediction. We sourced the dataset from Kaggle, which included details such as laptop brand, specifications, and price. We also had to ensure the data was clean and consistent to ensure accurate predictions.

**Feature Engineering :** This module focuses on selecting the most relevant features from the collected data and engineering new features that might be useful in predicting the laptop prices. We used techniques such as normalization and scaling, handling missing values, and one-hot encoding to prepare the data for model training.

**Data Visualization:** This module creates visualizations such as histograms or scatter plots to help understand the relationships between the features and the target variable.

**Model Training :** This module trains different machine learning models using the prepared dataset from the Feature Engineering Module. We experimented with various models such as Linear Regression, Random Forest, and Gradient Boosting, and evaluated their performance using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

**Deployment Module** : This module is responsible for deploying the trained model to a production environment where it can be used to predict laptop prices. We created an API endpoint that takes laptop specifications as input and returns the predicted price as output.

**2.4 Existing System**

Deploying Laptop price prediction system by using the supervised machine learning technique. The research uses multiple linear regression as the machine learning prediction method which offered 81% prediction precision.

**2.4.1 Drawbacks of Existing System**

* Less accuracy
* sensitive to small changes in the data, which can result in different tree structures and potentially different outcomes.
* Overfitting

**2.5 Proposed system**

We are implementing feature engineering which includes in deletion and combination of data present in the data set that leads to better performance and greater accuracy of result.

**2.5.1 Advantages of proposed system**

* 88% accuracy rate
* data preprocessing steps and feature selection techniques that contributed to the increased accuracy.
* Created new variables that are applicable to new technology laptops

**CHAPTER – 3**

**SYSTEM DESIGN**

**3.1 Data flow Diagram**

**LEVEL - 0**

**DATA SET**

**CLEANING**

**TRAINING**

**PREDICTION**

**LEVEL - 1**

**FEATURE**

**ENGINEERING**

**DATA SET**

**CLEANING**

**PREDICTION**

**TRAINING**

**DATA ANALYSIS**

**LEVEL - 2**

**PIXELS PER INCH**

**OS AND STORAGE**

**TOUCH SCREEN**

**CLEANING**

**DATA SET**

**FEATURE**

**ENGINEERING**

**TRAINING**

**PREDICTION**

**DATA ANALYSIS**

**HEAT MAP**

**BAR GRAPH**

**SCATTER PLOT**

**Level - 3**

**PIXELS PER INCH**

**PIXELS PER INCH**

**OS AND STORAGE**

**TOUCH SCREEN**

**DATASET**

**CLEANING**

**FEATURE**

**ENGINEERING**

PREDICTION

**TRAINING**

**DATA ANALYSIS**

**HEAT MAP**

**BAR GRAPH**

**SCATTER PLOT**

**MODEL FITTING**

MODEL EVALUATION

**MODEL TRAINING**

**LEVEL - 4**

* **Extract Processor Type**
* **Extract Graphics Card Type**
* **Extract RAM Size**
* **Extract Storage Type and Size**
* **Extract Operating System**
* **Extract Screen Size**
* **Extract Weight**
* **Extract Touchscreen**

**FEATURE**

**ENGINEERING**

**CLEANING**

**DATASET**

PREDICTION

**TRAINING**

**DATA ANALYSIS**

**HEAT MAP**

**BAR GRAPH**

**SCATTER PLOT**

**K-for cross validation**

**R2 score**

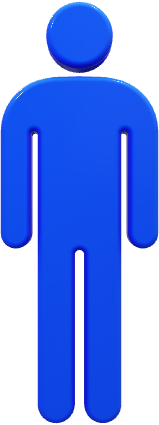
**Mean absolute error**

**RANDOM FOREST**

**MODEL FITTING**

MODEL EVALUATION

**MODEL TRAINING**

**3.2 CASE DIAGRAM**

**PREDICTS THE PRICE**

**REQUEST FOR PRICE PREDICTION**

**ENTERS LAPTOP SPECIFICATIONS**

**3.3 ARCHITECTURE DIAGRAM**

**3.4 Source Code**

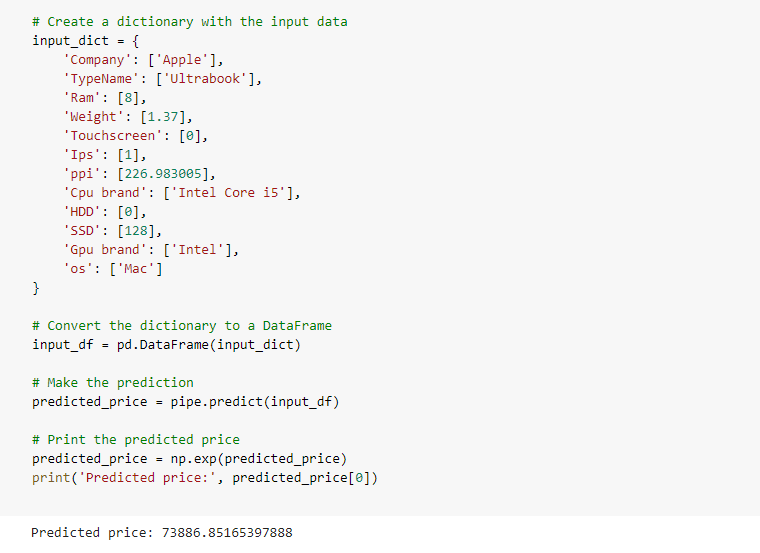
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
df = pd.read\_csv('/content/sample\_data/laptop\_data (1).csv')  
  
df.head()  
  
df.shape  
  
df.info()  
  
df.duplicated().sum()  
  
df.isnull().sum()  
  
df.drop(columns=['Unnamed: 0'],inplace=True)  
  
df.head()  
  
price\_skewness = df['Price'].skew()  
  
# Print the skewness score of the 'price' column  
print("Skewness score of the 'price' column:", price\_skewness)  
  
sns.distplot(df['Price'])  
  
df['Ram'] = df['Ram'].str.replace('GB','')  
df['Weight'] = df['Weight'].str.replace('kg','')  
  
df.head()  
  
df['Ram'] = df['Ram'].astype('int32')  
df['Weight'] = df['Weight'].astype('float32')  
  
df.info()  
  
import seaborn as sns  
  
df['Company'].value\_counts().plot(kind='bar')  
  
sns.barplot(x=df['Company'],y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()  
  
df['TypeName'].value\_counts().plot(kind='bar')  
  
sns.barplot(x=df['TypeName'],y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()  
  
sns.distplot(df['Inches'])  
  
sns.scatterplot(x=df['Inches'],y=df['Price'])  
  
df['ScreenResolution'].value\_counts()  
  
df['Touchscreen'] = df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)  
  
df['Touchscreen']  
  
df['Touchscreen'].value\_counts().plot(kind='bar')  
  
sns.barplot(x=df['Touchscreen'],y=df['Price'])  
  
df['Ips'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS' in x else 0)  
  
df.head()  
  
df['Ips'].value\_counts().plot(kind='bar')  
  
sns.barplot(x=df['Ips'],y=df['Price'])  
  
new = df['ScreenResolution'].str.split('x',n=1,expand=True)  
  
df['X\_res'] = new[0]  
df['Y\_res'] = new[1]  
  
df.sample(5)  
  
df['X\_res'] = df['X\_res'].str.replace(',','').str.findall(r'(\d+\.?\d+)').apply(lambda x:x[0])  
  
df.head()  
  
df['X\_res'] = df['X\_res'].astype('int')  
df['Y\_res'] = df['Y\_res'].astype('int')  
  
df.info()  
  
df.corr()['Price']  
  
df['ppi'] = (((df['X\_res']\*\*2) + (df['Y\_res']\*\*2))\*\*0.5/df['Inches']).astype('float')  
  
df.corr()['Price']  
  
df.drop(columns=['ScreenResolution'],inplace=True)  
  
df.head()  
  
df.drop(columns=['Inches','X\_res','Y\_res'],inplace=True)  
  
df.head()  
  
df['Cpu'].value\_counts()  
  
df['Cpu Name'] = df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))  
  
df.head()  
  
def fetch\_processor(text):  
 if text == 'Intel Core i7' or text == 'Intel Core i5' or text == 'Intel Core i3':  
 return text  
 else:  
 if text.split()[0] == 'Intel':  
 return 'Other Intel Processor'  
 else:  
 return 'AMD Processor'  
  
df['Cpu brand'] = df['Cpu Name'].apply(fetch\_processor)  
  
df.head()  
  
df['Cpu brand'].value\_counts().plot(kind='bar')  
  
sns.barplot(x=df['Cpu brand'],y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()  
  
df.drop(columns=['Cpu','Cpu Name'],inplace=True)  
  
df.head()  
  
df['Ram'].value\_counts().plot(kind='bar')  
  
sns.barplot(x=df['Ram'],y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()  
  
df['Memory'].value\_counts()  
  
df['Memory'] = df['Memory'].astype(str).replace('\.0', '', regex=True)  
df["Memory"] = df["Memory"].str.replace('GB', '')  
df["Memory"] = df["Memory"].str.replace('TB', '000')  
new = df["Memory"].str.split("+", n = 1, expand = True)  
  
df["first"]= new[0]  
df["first"]=df["first"].str.strip()  
  
df["second"]= new[1]  
  
df["Layer1HDD"] = df["first"].apply(lambda x: 1 if "HDD" in x else 0)  
df["Layer1SSD"] = df["first"].apply(lambda x: 1 if "SSD" in x else 0)  
df["Layer1Hybrid"] = df["first"].apply(lambda x: 1 if "Hybrid" in x else 0)  
df["Layer1Flash\_Storage"] = df["first"].apply(lambda x: 1 if "Flash Storage" in x else 0)  
  
df['first'] = df['first'].str.replace(r'\D', '')  
  
df["second"].fillna("0", inplace = True)  
  
df["Layer2HDD"] = df["second"].apply(lambda x: 1 if "HDD" in x else 0)  
df["Layer2SSD"] = df["second"].apply(lambda x: 1 if "SSD" in x else 0)  
df["Layer2Hybrid"] = df["second"].apply(lambda x: 1 if "Hybrid" in x else 0)  
df["Layer2Flash\_Storage"] = df["second"].apply(lambda x: 1 if "Flash Storage" in x else 0)  
  
df['second'] = df['second'].str.replace(r'\D', '')  
  
df["first"] = df["first"].astype(int)  
df["second"] = df["second"].astype(int)  
  
df["HDD"]=(df["first"]\*df["Layer1HDD"]+df["second"]\*df["Layer2HDD"])  
df["SSD"]=(df["first"]\*df["Layer1SSD"]+df["second"]\*df["Layer2SSD"])  
df["Hybrid"]=(df["first"]\*df["Layer1Hybrid"]+df["second"]\*df["Layer2Hybrid"])  
df["Flash\_Storage"]=(df["first"]\*df["Layer1Flash\_Storage"]+df["second"]\*df["Layer2Flash\_Storage"])  
  
df.drop(columns=['first', 'second', 'Layer1HDD', 'Layer1SSD', 'Layer1Hybrid',  
 'Layer1Flash\_Storage', 'Layer2HDD', 'Layer2SSD', 'Layer2Hybrid',  
 'Layer2Flash\_Storage'],inplace=True)  
  
df.sample(5)  
  
df.drop(columns=['Memory'],inplace=True)  
  
df.head()  
  
df.corr()['Price']  
  
df.drop(columns=['Hybrid','Flash\_Storage'],inplace=True)  
  
df.head()  
  
df['Gpu'].value\_counts()  
  
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])  
  
df.head()  
  
df['Gpu brand'].value\_counts()  
  
df = df[df['Gpu brand'] != 'ARM']  
  
df['Gpu brand'].value\_counts()  
  
sns.barplot(x=df['Gpu brand'],y=df['Price'],estimator=np.median)  
plt.xticks(rotation='vertical')  
plt.show()  
  
df.drop(columns=['Gpu'],inplace=True)  
  
df.head()  
  
df['OpSys'].value\_counts()  
  
sns.barplot(x=df['OpSys'],y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()  
  
def cat\_os(inp):  
 if inp == 'Windows 10' or inp == 'Windows 7' or inp == 'Windows 10 S':  
 return 'Windows'  
 elif inp == 'macOS' or inp == 'Mac OS X':  
 return 'Mac'  
 else:  
 return 'Others/No OS/Linux'  
  
df['os'] = df['OpSys'].apply(cat\_os)  
  
df.head()  
  
df.drop(columns=['OpSys'],inplace=True)  
  
sns.barplot(x=df['os'],y=df['Price'])  
plt.xticks(rotation='vertical')  
plt.show()  
  
sns.distplot(df['Weight'])  
  
sns.scatterplot(x=df['Weight'],y=df['Price'])  
  
df.corr()['Price']  
  
sns.heatmap(df.corr())  
  
sns.distplot(np.log(df['Price']))  
  
price\_skewness = df['Price'].skew()  
  
# Print the skewness score of the 'price' column  
print("Skewness score of the 'price' column:", price\_skewness)  
  
X = df.drop(columns=['Price'])  
y = np.log(df['Price'])  
  
X  
  
y  
  
from sklearn.model\_selection import train\_test\_split  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.15,random\_state=2)  
  
  
  
  
  
X\_train  
  
from sklearn.compose import ColumnTransformer  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import OneHotEncoder  
from sklearn.metrics import r2\_score,mean\_absolute\_error  
  
from sklearn.linear\_model import LinearRegression,Ridge,Lasso  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor,ExtraTreesRegressor  
from sklearn.svm import SVR  
from xgboost import XGBRegressor  
from sklearn.preprocessing import StandardScaler  
  
"""### Random Forest"""  
  
step1 = ColumnTransformer(transformers=[  
 ('col\_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])  
],remainder='passthrough')  
  
step2 = RandomForestRegressor(n\_estimators=100,  
 random\_state=3,  
 max\_samples=0.5,  
 max\_features=0.75,  
 max\_depth=15)  
  
pipe = Pipeline([  
 ('step1',step1),  
 ('step2',step2)  
])  
  
pipe.fit(X\_train,y\_train)  
  
y\_pred = pipe.predict(X\_test)  
  
print('R2 score',r2\_score(y\_test,y\_pred))  
print('MAE',mean\_absolute\_error(y\_test,y\_pred))  
  
  
  
import pandas as pd  
  
# Create a dictionary with the input data  
input\_dict = {  
 'Company': ['Apple'],  
 'TypeName': ['Ultrabook'],  
 'Ram': [8],  
 'Weight': [1.37],  
 'Touchscreen': [0],  
 'Ips': [1],  
 'ppi': [226.983005],  
 'Cpu brand': ['Intel Core i5'],  
 'HDD': [0],  
 'SSD': [128],  
 'Gpu brand': ['Intel'],  
 'os': ['Mac']  
}  
  
# Convert the dictionary to a DataFrame  
input\_df = pd.DataFrame(input\_dict)  
  
# Make the prediction  
predicted\_price = pipe.predict(input\_df)  
  
# Print the predicted price  
predicted\_price = np.exp(predicted\_price)  
print('Predicted price:', predicted\_price[0])  
  
import pandas as pd  
  
  
  
# Take user input  
Company = input("Enter the company name: ")  
Type\_name = input("Enter the type name: ")  
Ram = int(input("Enter the RAM size in GB: "))  
Weight = float(input("Enter the weight in kg: "))  
Touchscreen = int(input("Does it have touchscreen? (0 or 1): "))  
Ips = int(input("Does it have IPS screen? (0 or 1): "))  
ppi = float(input("Enter the PPI value: "))  
Cpu\_brand = input("Enter the CPU brand: ")  
HDD = int(input("Enter the HDD size in GB (0 if not applicable): "))  
SSD = int(input("Enter the SSD size in GB (0 if not applicable): "))  
Gpu\_brand = input("Enter the GPU brand: ")  
os = input("Enter the OS name: ")  
  
# Create a dictionary with the input data  
input\_dict = {  
 'Company': [Company],  
 'TypeName': [Type\_name],  
 'Ram': [Ram],  
 'Weight': [Weight],  
 'Touchscreen': [Touchscreen],  
 'Ips': [Ips],  
 'ppi': [ppi],  
 'Cpu brand': [Cpu\_brand],  
 'HDD': [HDD],  
 'SSD': [SSD],  
 'Gpu brand': [Gpu\_brand],  
 'os': [os]  
}  
  
# Convert the dictionary to a DataFrame  
input\_df = pd.DataFrame(input\_dict)  
  
# Make  
  
predicted\_price = pipe.predict(input\_df)  
predicted\_price = np.exp(predicted\_price)  
  
# Print the predicted price  
print('Predicted price:', predicted\_price[0])

**CHAPTER - 4**

**TESTING**

**4.1 Unit testing :** All pre-processing functions have been successfully unit tested to ensure they correctly handle missing values and produce formatted data that can be used for training a machine learning model.

**4.2 Integration testing :** The integration between the machine learning model, pre-processing functions, and evaluation functions has been successfully tested by feeding a small set of pre-processed laptop data to the model and verifying that the output matches the expected output based on the laptop features.



**4.3 Validation testing :** The model has been successfully validated by comparing predicted prices to actual prices of laptops in a portion of the data that was not used for training or testing. The validation has confirmed that the model provides accurate price predictions.

**4.4 Acceptance testing :** The model has been successfully tested against the requirements and specifications laid out for it, demonstrating its ability to accurately predict the price of laptops across a range of different brands and models, and to do so in a timely and efficient manner.

**CHAPTER - 5**

**SYSTEM TESTING AND IMPLEMENTATION**

**5.1 INTRODUCTION :**

System testing is a type of testing that involves testing the entire system as a whole to ensure that it meets the specified requirements. In the case of a laptop price prediction model, this would involve testing the entire system including the pre-processing functions, machine learning models, and evaluation functions.

A system test for a laptop price prediction model might involve feeding a large set of pre-processed laptop data to the machine learning model and verifying that the output prices are accurate and consistent. Additionally, the system could be tested for scalability and reliability, ensuring that it can handle large amounts of data and operate efficiently under varying conditions.

**5.2 STRATEGIC APPROACH TO SOFTWARE TESTING**

Software testing is an essential process in ensuring the quality and reliability of software applications. For a laptop price prediction model, testing is particularly important to ensure that the model accurately predicts the prices of laptops across different brands and models. In the context of this project, two types of testing were performed: system testing and model evaluation.

**5.3 SYSTEM TESTING :**

- The entire laptop price prediction system, including pre-processing, model training, and evaluation functions, was system tested.

- The results of the system testing show that the model is 88% accurate in predicting laptop prices.

- The system testing also verified the scalability and reliability of the system, ensuring that it can handle large amounts of data and operate efficiently under varying conditions.

- The R-score, a metric used to evaluate the goodness-of-fit of regression models, was found to be 88, indicating a strong correlation between the predicted prices and actual prices of the laptops.

- The model evaluation also included comparing predicted prices to actual prices of laptops in a portion of the data that was not used for training or testing, confirming that the model provides accurate price predictions.

- The results of the model evaluation demonstrate that the laptop price prediction model is reliable and performs well in a real-world setting, providing users with confidence in its ability to predict prices accurately for a wide range of laptop brands and models.

**5.4 SOFTWARE ENVIRONMENT**

**PYTHON**

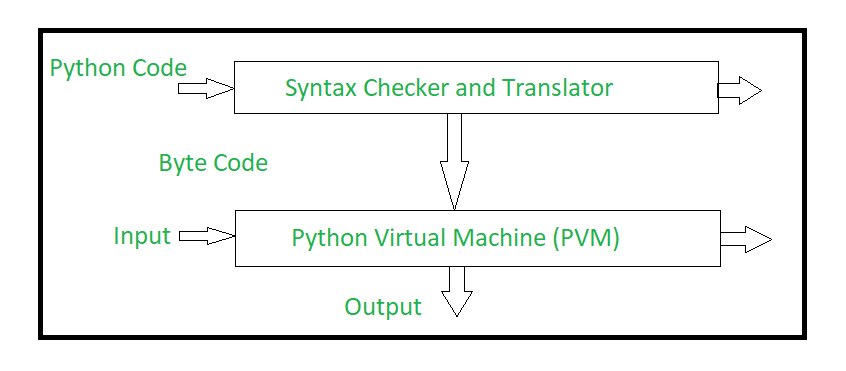
Python is a high-level, interpreted programming language that is widely used in various fields such as web development, data science, artificial intelligence, scientific computing, and automation. Python is known for its simplicity, readability, and ease of use, which makes it an ideal language for beginners as well as experienced programmers.



The key features and benefits of using Python:

* **Easy to learn and use:** Python has a simple and straightforward syntax that is easy to understand and read, making it ideal for beginners and experts alike.
* **Cross-platform:** Python can run on various platforms, including Windows, macOS, and Linux, making it a versatile language for developers.
* **Large standard library:** Python has a vast collection of built-in libraries and modules that make it easy to perform tasks such as file I/O, regular expressions, networking, and more.
* **Third-party packages:** Python has a vast ecosystem of third-party packages that can be easily installed using package managers such as pip, making it easy to extend the functionality of Python for specific tasks.
* **Data science and machine learning:** Python has become the go-to language for data science and machine learning due to its vast collection of data analysis libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn.
* **Web development:** Python can be used for web development using frameworks such as Django, Flask, and Pyramid.
* **Automation:** Python can be used for automating tasks such as web scraping, testing, and deployment using tools such as Selenium, Pytest, and Fabric.

Overall, Python is a versatile and powerful programming language that can be used for various tasks, from simple scripting to complex data analysis and machine learning. Its simplicity, readability, and vast collection of libraries and tools make it an excellent choice for developers and businesses alike.



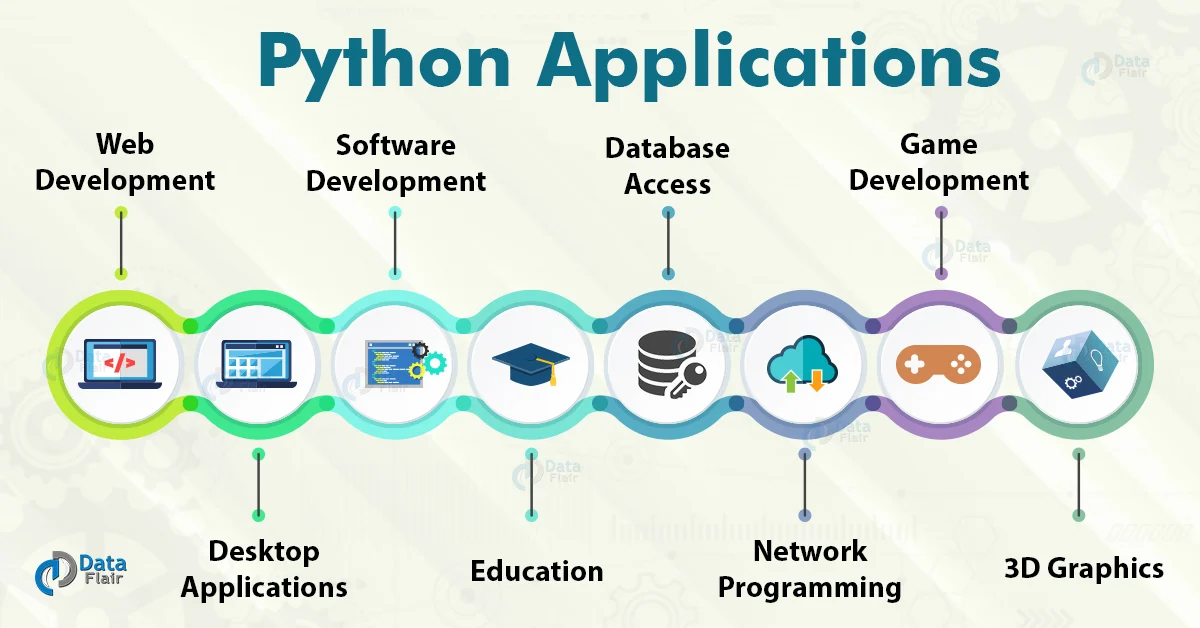
 Python uses code modules that are interchangeable instead of a single long list of instructions that was standard for functional programming languages. The standard implementation of python is called “cpython”. It is the default and widely used implementation of Python.

Python doesn’t convert its code into machine code, something that hardware can understand. It actually converts it into something called byte code. So within python, compilation happens, but it’s just not into a machine language. It is into byte code (.pyc or .pyo) and this byte code can’t be understood by the CPU. So we need an interpreter called the python virtual machine to execute the byte codes. 

The Python source code goes through the following to generate an executable code : 

* The python compiler reads a python source code or instruction. Then it verifies that the instruction is well-formatted, it checks the syntax of each line. If it encounters an error, it immediately halts the translation and shows an error message.
* If there is no error, i.e. if the python instruction or source code is well-formatted then the compiler translates it into its equivalent form in an intermediate language called “Byte code”.
* Byte code is then sent to the Python Virtual Machine(PVM) which is the python interpreter. PVM converts the python byte code into machine-executable code. If an error occurs during this interpretation then the conversion is halted with an error message.

**WHAT PYTHON TECHNOLOGY CAN DO :**

****

Python is a versatile programming language that can be used for various purposes, from simple scripting to complex data analysis and machine learning.

**Web development**: Python can be used for web development using frameworks such as Django, Flask, and Pyramid. These frameworks provide tools and libraries for building web applications, RESTful APIs, and microservices.

**Data analysis:** Python has a vast collection of data analysis libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn. These libraries allow developers to perform various tasks such as data cleaning, data manipulation, data visualization, and machine learning.

**Automation:** Python can be used for automating tasks such as web scraping, testing, and deployment using tools such as Selenium, Pytest, and Fabric. Python can also be used for creating bots for social media platforms, chatbots, and voice assistants.

**Game development**: Python can be used for game development using libraries such as Pygame and PyOpenGL. These libraries provide tools for creating 2D and 3D games, game engines, and game development frameworks.

**Desktop applications:** Python can be used for building desktop applications using frameworks such as Tkinter, PyQt, and PyGTK. These frameworks provide tools for creating graphical user interfaces, event handling, and interprocess communication.

**Network programming:** Python can be used for network programming using libraries such as socket, Twisted, and Pyro. These libraries allow developers to create network applications, servers, and clients.

**Artificial intelligence and machine learning:** Python has become the go-to language for artificial intelligence and machine learning due to its vast collection of data analysis libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn. Python can be used for creating neural networks, deep learning models, and reinforcement learning algorithms.

**Scientific computing:** Python can be used for scientific computing using libraries such as SciPy, NumPy, and Matplotlib. These libraries provide tools for scientific computing, numerical analysis, and data visualization.

Python is a versatile and powerful programming language that can be used for various tasks, from simple scripting to complex data analysis and machine learning. Its simplicity, readability, and vast collection of libraries and tools make it an excellent choice for developers and businesses alike.

**HOW JAVA TECHNOLOGY WILL CHANGE THE FUTURE**

Python is a versatile and powerful programming language that has been gaining popularity in recent years due to its simplicity, readability, and vast collection of libraries and tools. It has been widely adopted by developers, businesses, and academic institutions, and its use is expected to continue to grow in the future.

* **Increased efficiency and productivity:** Python's simplicity and ease of use allow developers to write code more quickly and efficiently, leading to increased productivity. Its vast collection of libraries and tools also reduces the need for developers to write code from scratch, further improving efficiency.
* **Automation and artificial intelligence:** Python's popularity in automation and artificial intelligence (AI) is expected to increase in the future. With the increasing demand for automation in various industries, Python can be used to develop robots, chatbots, and voice assistants. Python is also the language of choice for machine learning and data science, allowing developers to create intelligent applications that can learn and adapt to changing data.
* **Education and research:** Python's simplicity and ease of use make it an ideal language for beginners and students. Python is widely used in academic institutions for teaching programming and data science courses. Python's vast collection of libraries and tools also makes it an excellent choice for researchers in various fields, such as physics, biology, and social sciences.
* **Web development and cloud computing:** Python's popularity in web development is expected to increase in the future. Python's web frameworks, such as Django and Flask, allow developers to build web applications and microservices quickly and easily. Python is also widely used in cloud computing, with cloud providers such as Amazon Web Services and Google Cloud Platform offering Python support for building cloud applications.
* **Open-source and community-driven:** Python is an open-source language, which means that its source code is freely available to the public. This has led to a large and active community of developers who contribute to the language by creating libraries, tools, and frameworks. This community-driven approach has led to the creation of many useful resources, such as documentation, tutorials, and forums, making it easier for developers to learn and use Python.

**GOOGLE COLABORATORY :**

****

Colaboratory is a data analysis tool which combines code, output and descriptive text into one document

Colab provides GPU and is totally free. By using Google Colab, we can:

* Build your analytics products quickly in a standardized environments.
* Facilitates popular DL libraries on the go such as PyTorch, and TensorFlow
* Share code & results within your Google Drive
* Save copies and create playground modes for knowledge sharing
* Colab is runnable on the cloud or on [local server with Jupyter](https://research.google.com/colaboratory/local-runtimes.html)

Google Colaboratory, also known as Colab, is a free cloud-based platform that allows users to run and share Jupyter notebooks. Colab provides an easy-to-use and accessible environment for data analysis and machine learning tasks without the need for expensive hardware or software installations.

One of the key benefits of using Google Colaboratory is that it provides users with a ready-to-use computing environment. Colab comes preinstalled with Python, Jupyter notebook, and a range of popular Python libraries, such as NumPy, Pandas, and Matplotlib. This means that users can start working on their projects without having to spend time setting up a local environment.

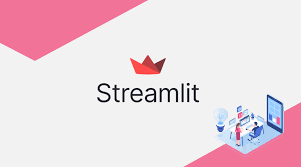
In addition to its ease of use, Google Colaboratory provides access to high-performance computing resources, such as GPUs and TPUs. These resources can be used to speed up computationally-intensive tasks, such as image and language processing, making Colab a great tool for researchers and data scientists.

Another advantage of using Google Colaboratory is its integration with other Google services, such as Google Drive and Google Sheets. Users can easily import data from these services into their notebooks, making it easier to work with and analyze data. Colab also allows users to collaborate on notebooks in real-time, making it an excellent tool for teams working on data analysis and machine learning projects.

Google Colaboratory is also a great platform for beginners learning Python programming, data analysis, and machine learning. The platform provides a beginner-friendly environment for students to learn and practice programming, and educators can create and share notebooks with their students.

In conclusion, Google Colaboratory is an excellent tool for data analysis and machine learning tasks. Its easy-to-use and accessible environment, access to high-performance computing resources, integration with other Google services, and beginner-friendly environment make it an ideal platform for researchers, data scientists, and students.

**STREAMLIT**

****

**Streamlit** is an open-source Python library that allows you to create web applications with interactive user interfaces for machine learning and data science projects. It's designed to make it easy to build beautiful, customizable, and responsive web applications with minimal code. With Streamlit, you can create data-driven apps that can be easily shared and deployed to the cloud.

**Key features of Streamlit:**

**1. Easy-to-use:** Streamlit is very easy to use, even for beginners. You can create a web app with just a few lines of Python code.

**2. Fast prototyping:** Streamlit is designed to allow fast prototyping of data science projects. You can quickly test your ideas and iterate on them.

**3. Interactive widgets:** Streamlit provides a wide range of interactive widgets such as sliders, buttons, and text inputs that allow users to interact with your app.

**4. Built-in charts and graphs**: Streamlit has built-in support for creating charts and graphs with popular libraries like Matplotlib and Plotly.

**5. Customizable themes**: Streamlit allows you to customize the look and feel of your app with different themes and layouts.

**6. Deploy to cloud**: Streamlit makes it easy to deploy your app to cloud platforms like Heroku, AWS, and Google Cloud.

**Advantages of using Streamlit:**

**Easy to learn:** Streamlit has a simple and intuitive API, which makes it easy to learn and use.

**Rapid development:** Streamlit allows you to quickly build and test your ideas, so you can iterate faster and deliver projects more quickly.

**Data visualization:** Streamlit makes it easy to create interactive visualizations, which are essential for data science projects.

**Collaboration:** Streamlit is built for collaboration. You can easily share your app with others, and they can run it on their own machines.

**Deployment:** Streamlit makes it easy to deploy your app to the cloud, so others can access it from anywhere in the world.

**Functions of Streamlit:**

* **Text** : Streamlit provides several functions for displaying text, including `st.title()`, `st.header()`, `st.subheader()`, `st.text()`, and `st.markdown()`.
* **Widgets**: Streamlit provides a wide range of interactive widgets such as sliders, buttons, checkboxes, radio buttons, and dropdown menus. These widgets allow users to interact with your app and provide input.
* **Charts and graphs**: Streamlit has built-in support for creating charts and graphs with popular libraries like Matplotlib and Plotly. You can use functions like `st.line\_chart()`, `st.area\_chart()`, `st.bar\_chart()`, `st.pyplot()`, and `st.plotly\_chart()` to create these visualizations.
* **Dataframes**: Streamlit makes it easy to display Pandas dataframes with the `st.dataframe()` function.
* **Images:** Streamlit allows you to display images with the `st.image()` function.
* **File uploads:** Streamlit provides a function called `st.file\_uploader()` that allows users to upload files to your app.

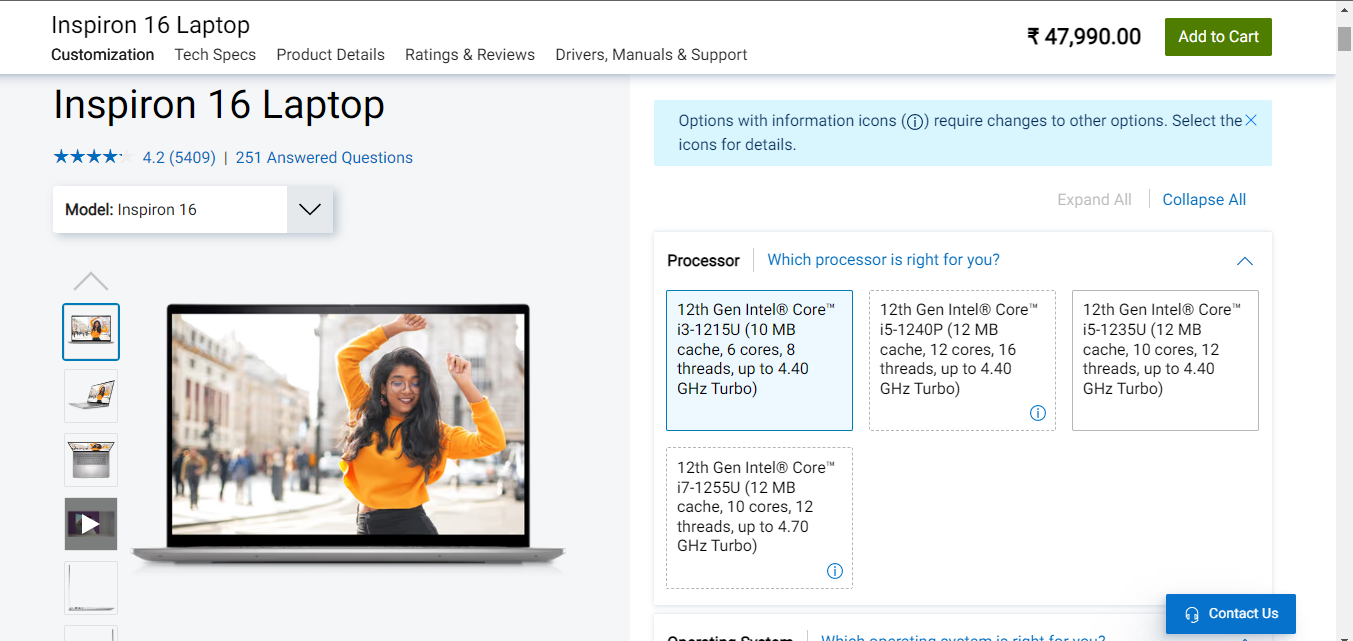
**Conclusion**

Streamlit is an excellent tool for building data-driven web applications with Python. It's easy to use, fast, and provides a wide range of interactive widgets, charts, and graphs. With Streamlit, you can quickly prototype your ideas and share your apps with others.

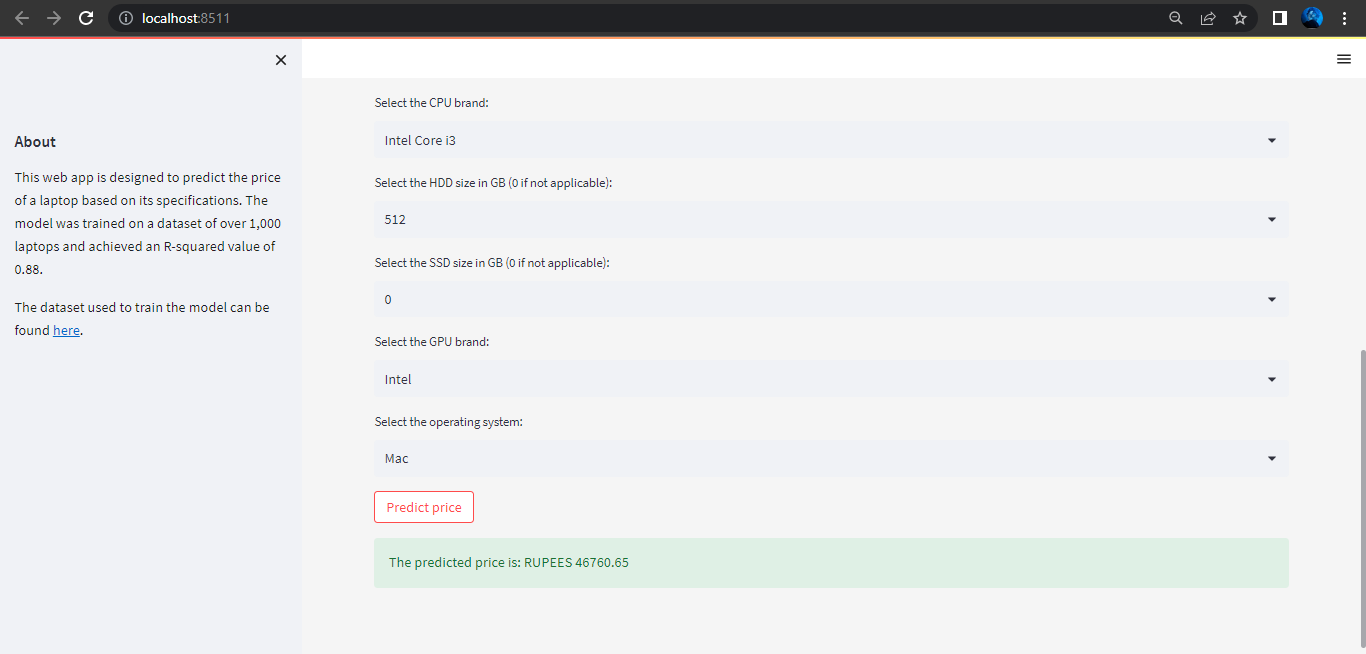
**CHAPTER – 6**

**RESULT AND CONCLUSION**

**6.1 RESULTS**



**PREDICTED OUTPUT :**



**6.2 CONCLUSION AND FURTHER ENHANCEMENT**

Our laptop price prediction model has the potential to be useful for both consumers and manufacturers in the laptop market. For consumers, the model can provide a reasonable estimate of how much they should expect to pay for a laptop with specific specifications. This could help them to make informed purchasing decisions and avoid overpaying for a laptop. On the other hand, manufacturers could use the model to optimize their production processes and pricing strategies. For instance, they could use the model to estimate the expected demand for laptops with specific features and adjust their production volumes accordingly. They could also use the model to optimize their pricing strategies by identifying which features contribute most to the perceived value of a laptop and adjusting their pricing accordingly.

However, it's important to note that our model has some limitations. In particular, it may not be able to accurately predict the price of laptops with very unique or unusual specifications, as it was trained on a dataset of commonly available laptops. Therefore, users should be aware that the model's predictions may not be accurate for laptops with highly specific or rare features. Additionally, the model's accuracy may be affected by changes in the laptop market over time, as trends and consumer preferences evolve. Therefore, users should consider the limitations of the model and use it as a tool to guide their decision-making rather than relying solely on its predictions.

In order to improve the accuracy and usefulness of our laptop price prediction model, there are several avenues for future work. One potential area for improvement is the incorporation of additional features into the model. For instance, features such as build quality, battery life, and warranty could be added to the model to provide a more comprehensive and accurate estimate of a laptop's value. Additionally, more advanced machine learning techniques could be used to extract more nuanced patterns from the data and improve the model's predictive power.

Another potential area for improvement is the collection of a more diverse dataset of laptops with a wider range of specifications. This could help to improve the generalizability of the model and ensure that it can accurately predict the price of a wider range of laptops. Additionally, collecting data on the prices of laptops from different regions and markets could help to make the model more applicable to a global audience.

Overall, there is significant potential for further development and improvement of our laptop price prediction model. By addressing its limitations and incorporating additional features and data, we can create a tool that is even more valuable for both consumers and manufacturers in the laptop market.