LAPTOP PRICE PREDICTOR

A COURSE PROJECT REPORT

Submitted by

SHIVAM PAHARIYA (RA2011027010007)

ALANKRITI KALSI (RA2011027010023)

Under the guidance of

Dr. M. PRAKASH

In partial fulfilment for the Course

of

Machine Learning I (18CSE392T)

in

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

KATTANKULATHUR - 603 203

November, 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project titled "**Title**" is the bonafide work of **Shivam Pahariya (RA2011027010007), Alankriti Kalsi (RA2011027010023),** who carried out the project work under my supervision.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Dr. M. PRAKASH	Dr. M. LAKSHMI
Associate Professor	Professor & Head
Department of Data Science and Business Systems	Department of Data Science and Business Systems
SRM Institute of Science and Technology	SRM Institute of Science and Technology
Kattankulathur – 603 203	Kattankulathur – 603 203

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor** and **Course Coordinator** for their constant encouragement and support.

We are highly thankful to my Course project Faculty Dr. **M. Prakash, Associate Professor, Department of Data Science and Business Systems,** for his assistance, timely suggestion, and guidance throughout the duration of this course project.

We extend our gratitude to our **HoD, Dr. M. Lakshmi, Professor, Department of Data Science and Business Systems,** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

ABSTRACT

Machine learning is a branch of Artificial intelligence that deals with implementing applications that can make a future prediction based on past data. If you are a data science enthusiast or a practitioner then this article will help build your own end-to-end machine learning project from scratch. There are various steps involved in building a machine learning project but not all the steps are mandatory to use in a single project, and it all depends on the data. In this article, we will build a laptop price prediction project and learn about the machine learning project lifecycle.

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. Using multiple linear regression, there are multiple independent variables but one and only one dependent variable whose actual and predicted values are compared to find precision of results. This paper proposes a system where price is dependent variable which is predicted, and this price is derived from factors like Laptop’s model, RAM, ROM (HDD/SSD), GPU, CPU, IPS Display, and Touch Screen.

TABLE OF CONTENTS

1. INTRODUCTION

We will make a project for Laptop price prediction. The problem statement is that if any user wants to buy a laptop, then our application should be compatible to provide a tentative price of laptop according to the user configurations. Although it looks like a simple project or just developing a model, the dataset we have is noisy and needs lots of feature engineering, and preprocessing that will drive your interest in developing this project.

Laptop price prediction especially when the laptop is coming direct from the factory to Electronic Market/ Stores, is both a critical and important task. The mad rush that we saw in 2020 for laptops to support remote work and learning is no longer there. In India, demand of Laptops soared after the Nationwide lockdown, leading to 4.1-Million-unit shipments in the June quarter of 2021, the highest in the five years. Accurate Laptop price prediction involves expert knowledge, because price usually depends on many distinctive features and factors. Typically, most significant ones are brand and model, RAM, ROM, GPU, CPU, etc.

2. LITERATURE SURVEY

Author: Prof. Vaishali Surjuse; Sankalp Lohakare; Aayush Barapatre; Abhishek Chapke

(KDKCE & RTMNU University)

Published on International Journal of Computer Science and Mobile Computing

This paper presents a 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. Using multiple linear regression, there are multiple independent variables but one and only one dependent variable whose actual and predicted values are compared to find precision of results. This paper proposes a system where price is dependent variable which is predicted, and this price is derived from factors like Laptop's model, RAM, ROM (HDD/SSD), GPU, CPU, IPS Display, and Touch Screen.

https://www.researchgate.net/publication/241161008_Laptop_selection_using_data_mining_of_critical_features

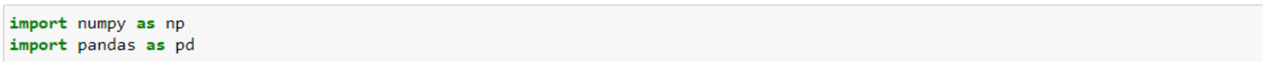
3. REQUIREMENTS ANALYSIS

- NumPy

  NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

- Pandas

  Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.



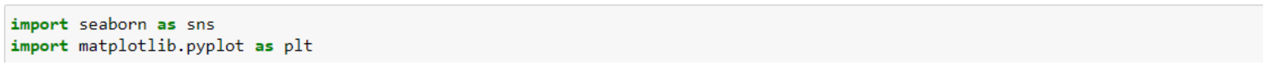
```
import numpy as np
import pandas as pd
```

- Seaborn

  Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- Matplotlib

  Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.
  One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.



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

- Sklearn

  Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

- Sklearn.compose

  Applies transformers to columns of an array or pandas DataFrame. This estimator allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space. This is useful for heterogeneous or columnar data, to combine several feature extraction mechanisms or transformations into a single transformer.

- Sklearn.pipeline

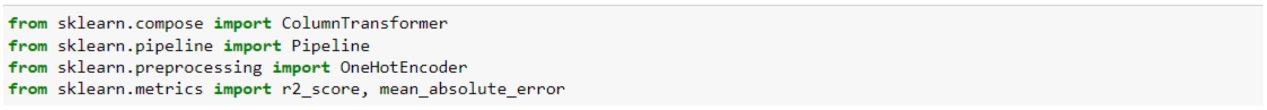
  Pipeline of transforms with a final estimator. The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. For this, it enables setting parameters of the various steps using their names and the parameter name separated by a '__', as in the example below. A step's estimator may be replaced entirely by setting the parameter with its name to another estimator, or a transformer removed by setting it to 'passthrough' or None.

- Sklearn.preprocessing

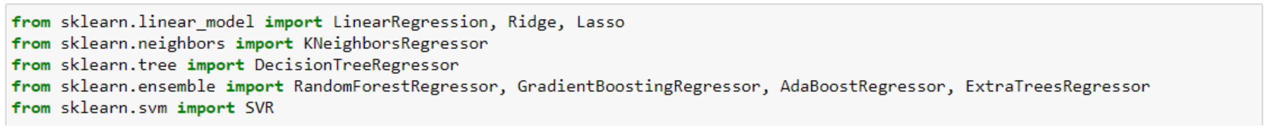
  The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate. The behaviors of the different scalers, transformers, and normalizers on a dataset containing marginal outliers is highlighted in Compare the effect of different scalers on data with outliers.

- Sklearn.metrics

  The sklearn. metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.



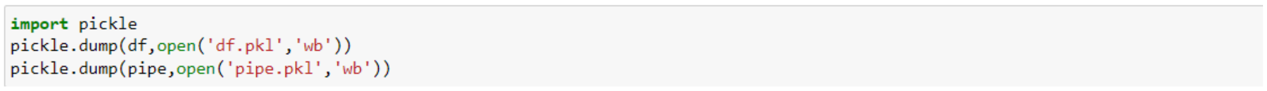
```python
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score, mean_absolute_error
```



```python
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
```

- Pickle

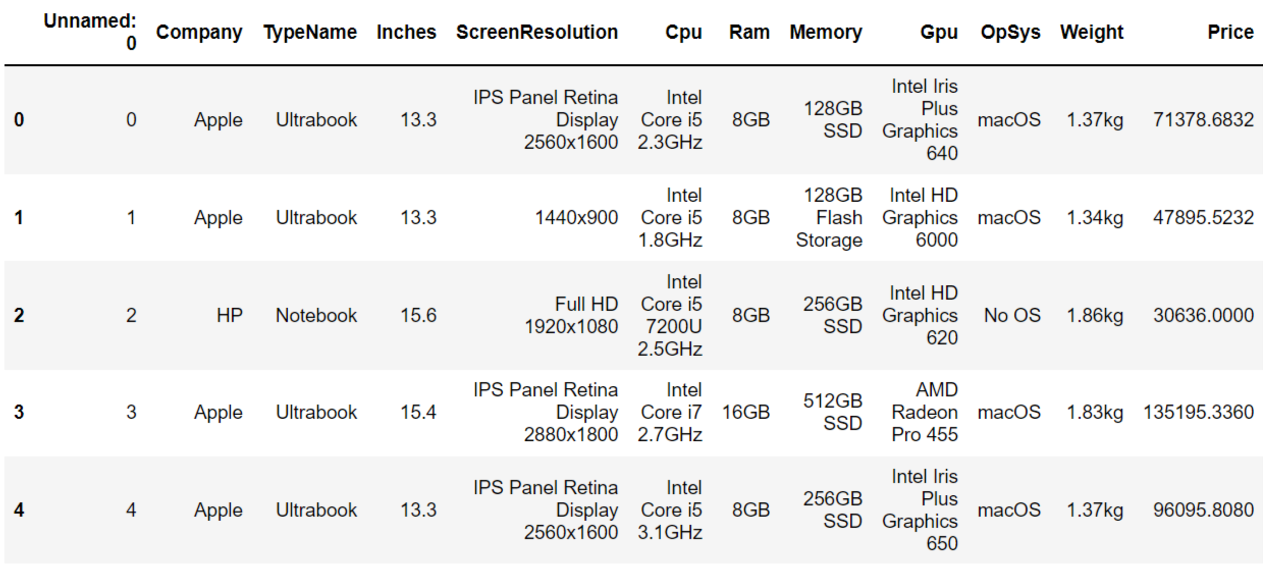
  Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.



```
import pickle
pickle.dump(df,open('df.pkl','wb'))
pickle.dump(pipe,open('pipe.pkl','wb'))
```

4. DATA SET DESCRIPTION

To support the application of machine learning using the Decision Tree algorithm, of course the sample data is needed. Table below contains data about various laptops and their prices depending on their configuration. Sample data were obtained from Kaggle.com



	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

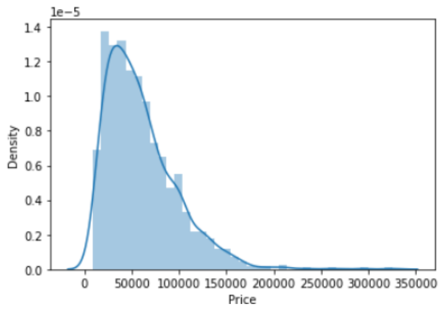
1. Company- String -Laptop Manufacturer
2. TypeName -String -Type (Notebook, Ultrabook, Gaming, etc.)
3. Inches -Numeric- Screen Size
4. ScreenResolution -String- Screen Resolution
5. Cpu- String -Central Processing Unit (CPU)
6. Ram -String- Laptop RAM
7. Memory -String- Hard Disk / SSD Memory
8. GPU -String- Graphics Processing Units (GPU)
9. OpSys -String- Operating System
10. Weight -String- Laptop Weight
11. Price - Numeric- Price (Rs)

5. EDA

```
sns.distplot(df['Price'])
```

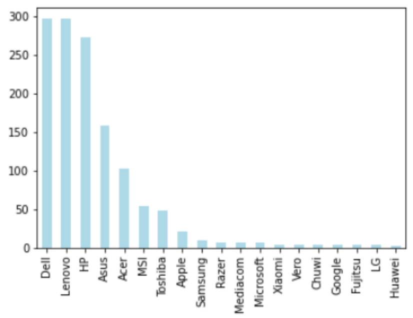
```
C:\Users\Alankriti Kalsi\Anaconda\lib\site-packages\seaborn\distribu
nction and will be removed in a future version. Please adapt your co
milar flexibility) or `histplot` (an axes-level function for histogr
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```

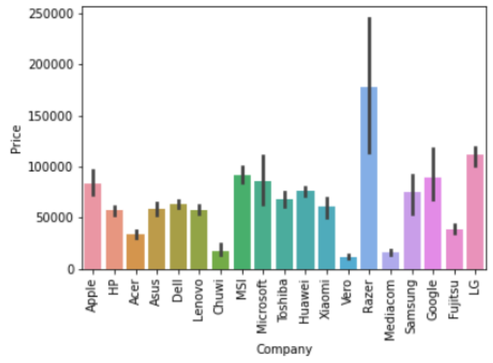


```
df['Company'].value_counts().plot(kind='bar', color='lightblue')
```

```
<AxesSubplot:>
```

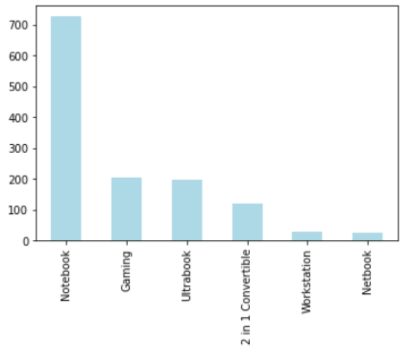


```
#Average price of each brand
sns.barplot(x=df['Company'], y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```

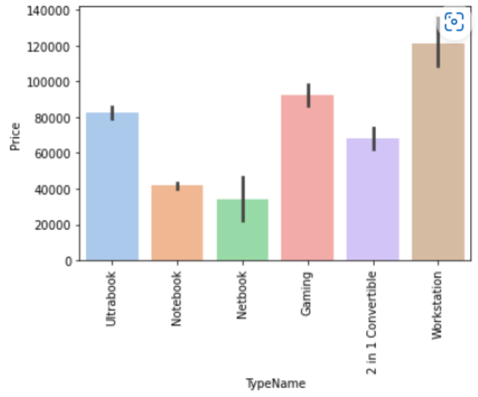


```
df['TypeName'].value_counts().plot(kind='bar', color='lightblue')
```

```
<AxesSubplot:>
```



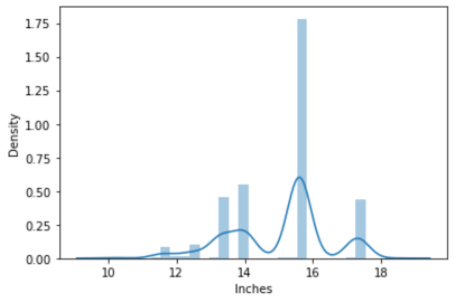
```
sns.barplot(x=df['TypeName'], y=df['Price'], palette='pastel')
plt.xticks(rotation='vertical')
plt.show()
```



```
sns.distplot(df['Inches'])
```

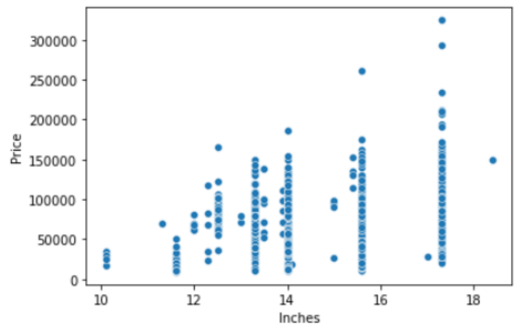
```
C:\Users\Alankriti Kalsi\Anaconda\lib\site-packages\seaborn\distribut
nction and will be removed in a future version. Please adapt your coc
milar flexibility) or `histplot` (an axes-level function for histogra
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='Inches', ylabel='Density'>
```

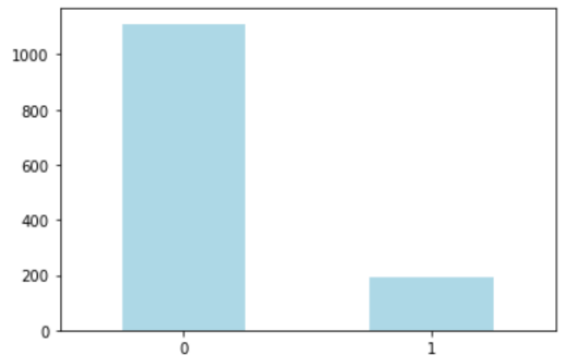


```
sns.scatterplot(x=df['Inches'], y=df['Price'])
```

```
<AxesSubplot:xlabel='Inches', ylabel='Price'>
```

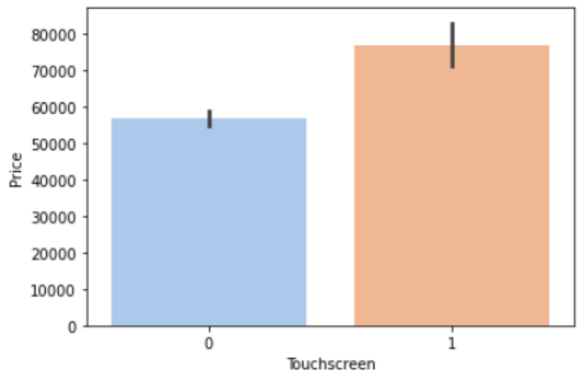


```
df['Touchscreen'].value_counts().plot(kind='bar', colo
plt.xticks(rotation='horizontal')
plt.show()
```

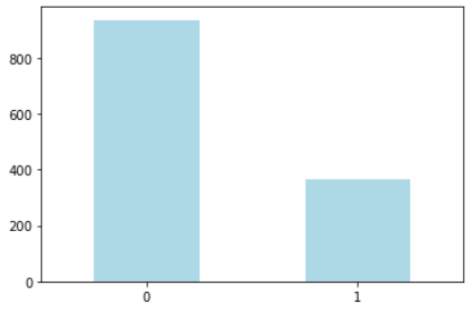


```
sns.barplot(x=df['Touchscreen'], y=df['Price'], palette='
```

```
<AxesSubplot:xlabel='Touchscreen', ylabel='Price'>
```

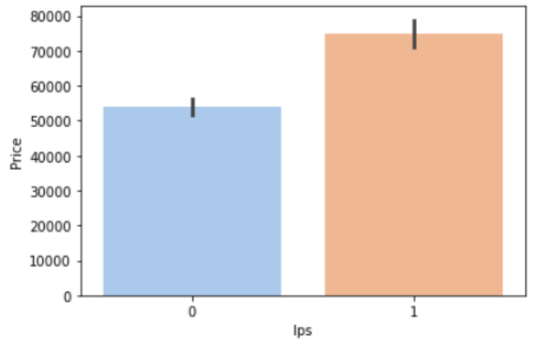


```
df['Ips'].value_counts().plot(kind='bar', color='lightblue')
plt.xticks(rotation='horizontal')
plt.show()
```



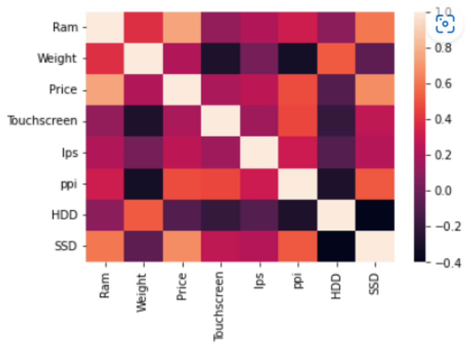
```
sns.barplot(x=df['Ips'], y=df['Price'], palette='pastel')
```

```
<AxesSubplot:xlabel='Ips', ylabel='Price'>
```



```
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```



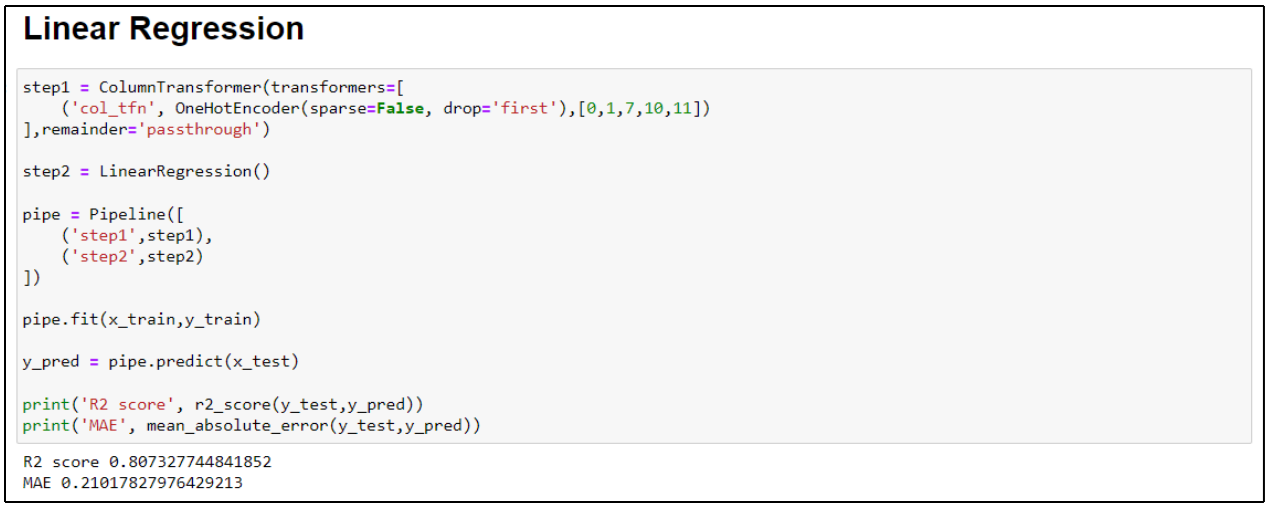
High correlation can be seen between:

- RAM and Price
- RAM and SSD
- Weight and HDD
- Price and SSD
- Price and RAM
- PPI and SSD, Touchscreen, and Price

6. ALGORITHMS USED

1. Linear Regression

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



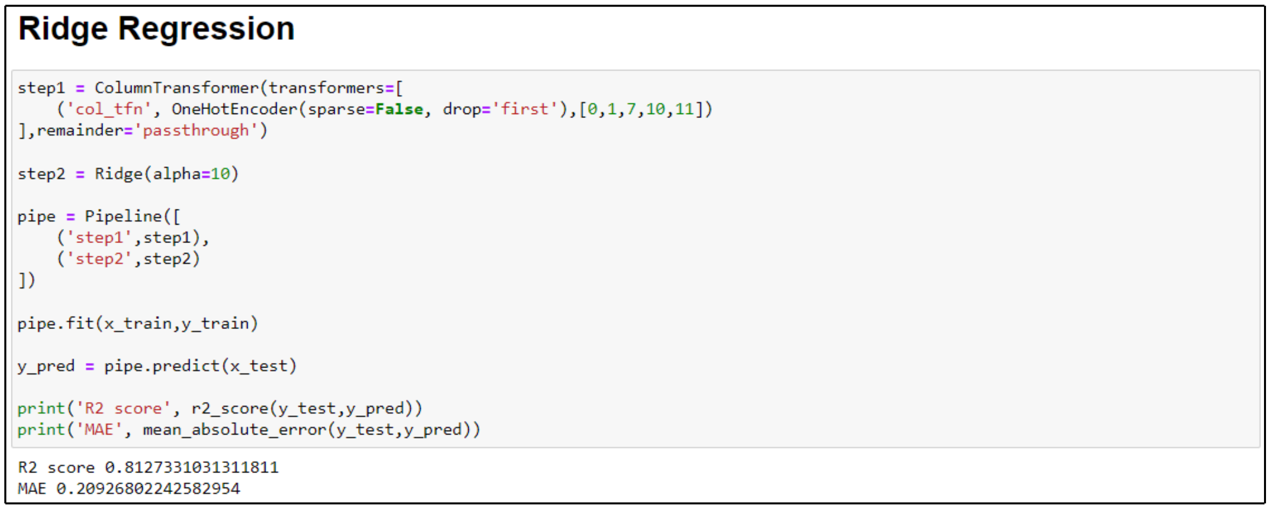
Linear Regression

```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = LinearRegression()

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))
```

```
R2 score 0.807327744841852
MAE 0.21017827976429213
```

2. Ridge Regression

Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. It has been used in many fields including econometrics, chemistry, and engineering.



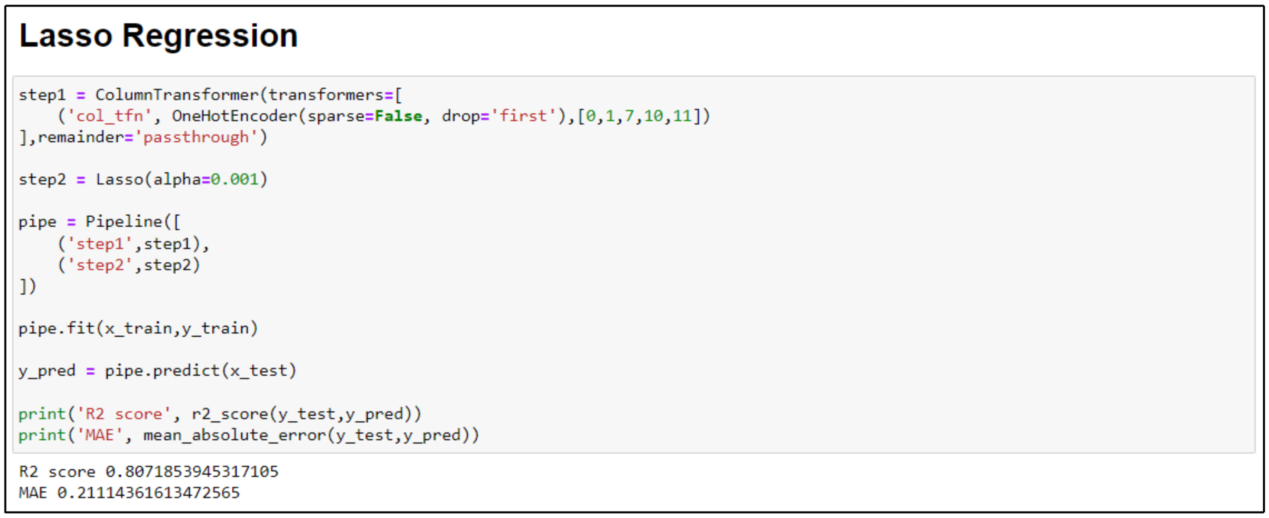
Ridge Regression

```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = Ridge(alpha=10)

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))
```

```
R2 score 0.8127331031311811
MAE 0.20926802242582954
```

3. Lasso Regression

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. The acronym "LASSO" stands for Least Absolute S.



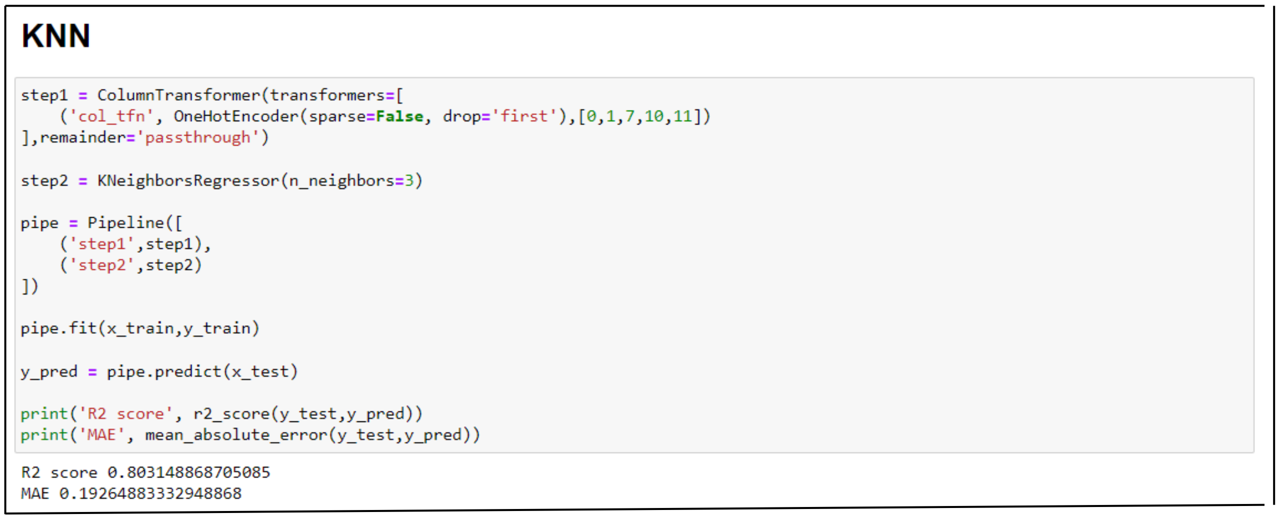
Lasso Regression

```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = Lasso(alpha=0.001)

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))
```

```
R2 score 0.8071853945317105
MAE 0.21114361613472565
```

4. KNN

The k-nearest neighbors' algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.



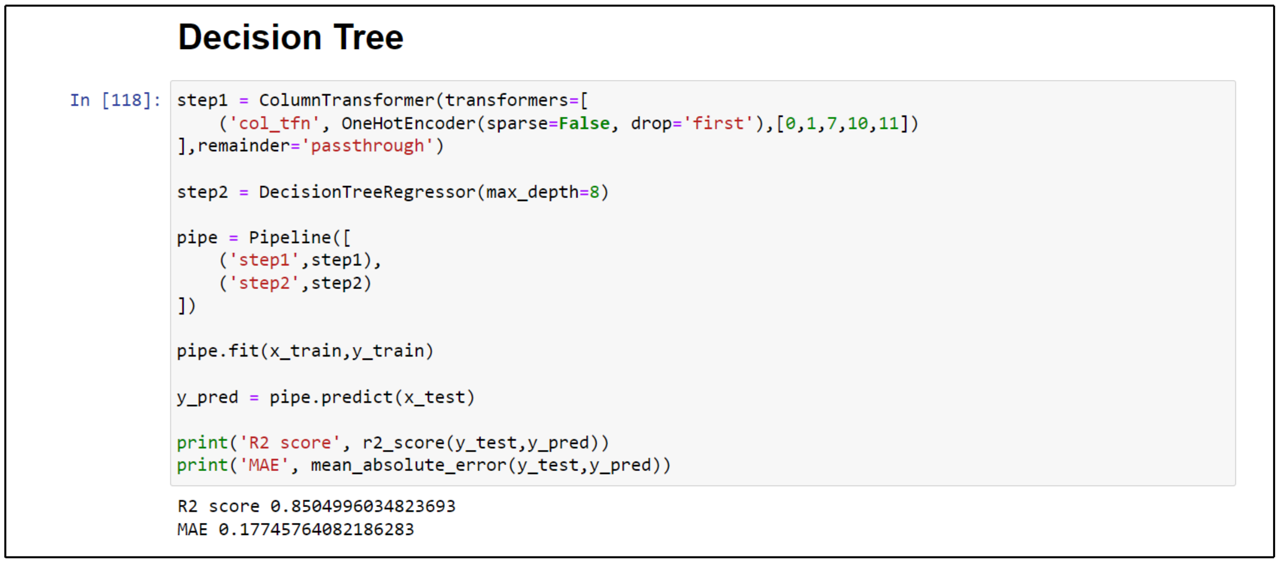
KNN

```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = KNeighborsRegressor(n_neighbors=3)

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))
```

```
R2 score 0.803148868705085
MAE 0.19264883332948868
```

5. Decision Tree

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

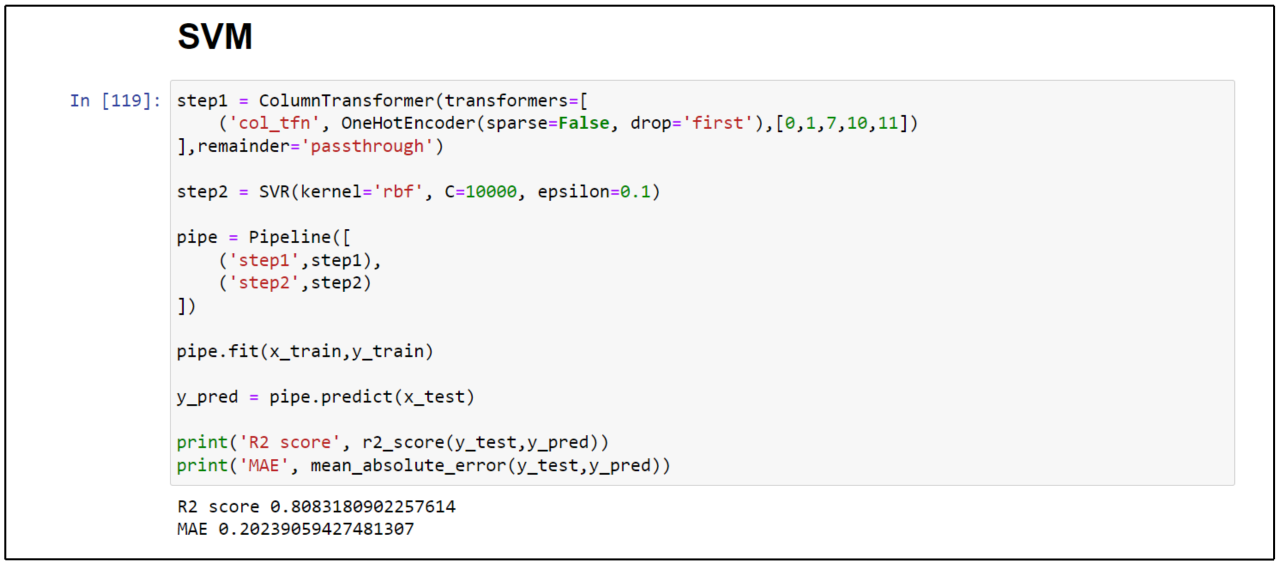


Decision Tree

In [118]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = DecisionTreeRegressor(max_depth=8)

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))
```
```
R2 score 0.8504996034823693
MAE 0.17745764082186283
```

6. SVM

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.



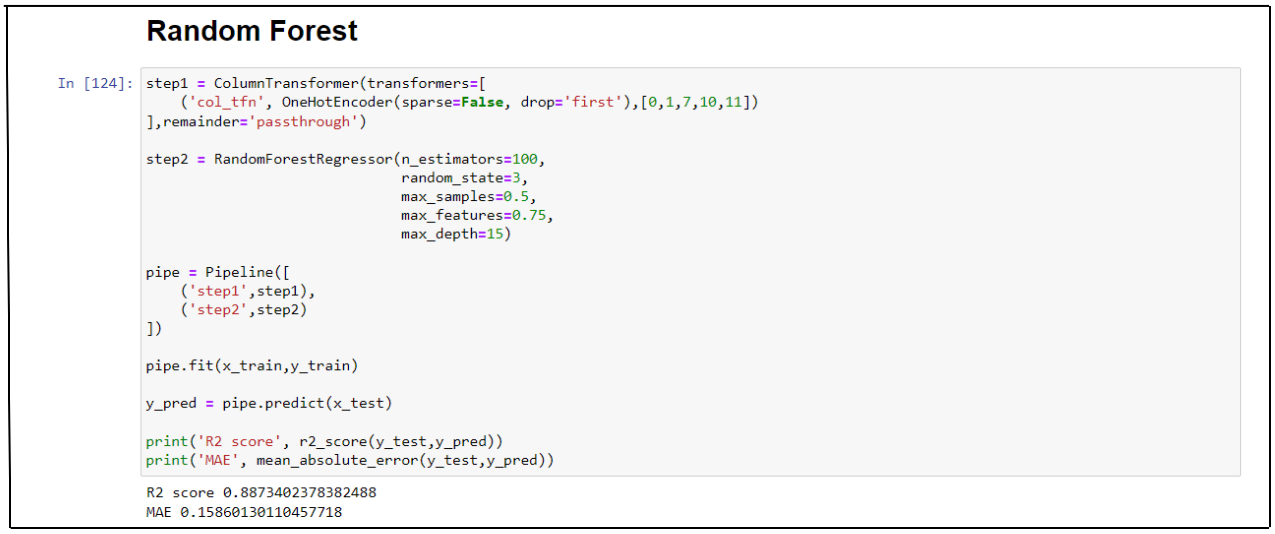
SVM

In [119]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2 = SVR(kernel='rbf', C=10000, epsilon=0.1)

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))
```
```
R2 score 0.8083180902257614
MAE 0.20239059427481307
```

7. Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.



Random Forest

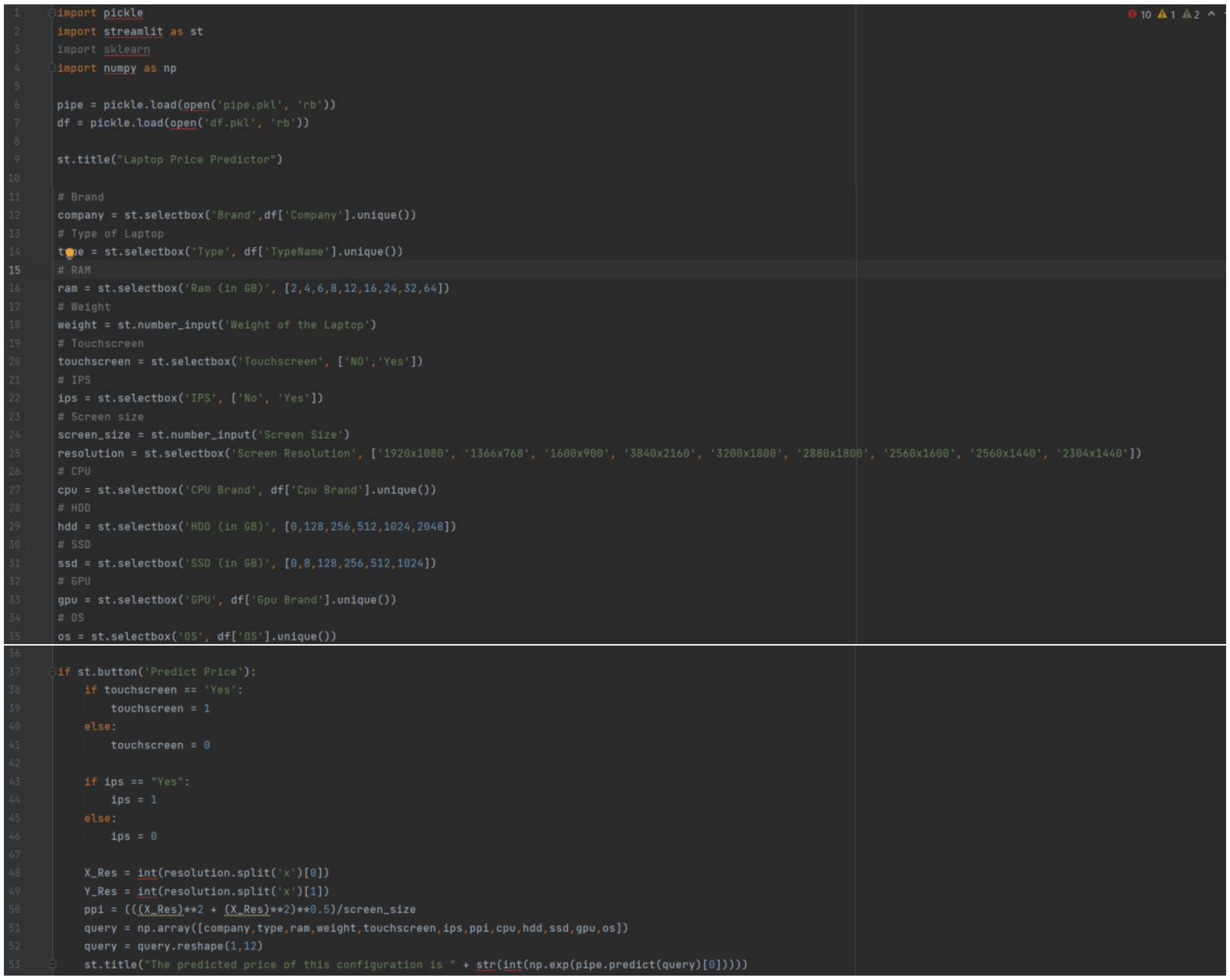
In [124]:
```
step1 = ColumnTransformer(transformers=[
    ('col_tfn', OneHotEncoder(sparse=False, drop='first'),[0,1,7,10,11])
],remainder='passthrough')

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))
```

```
R2 score 0.8873402378382488
MAE 0.15860130110457718
```

7. RESULTS AND DISCUSSION

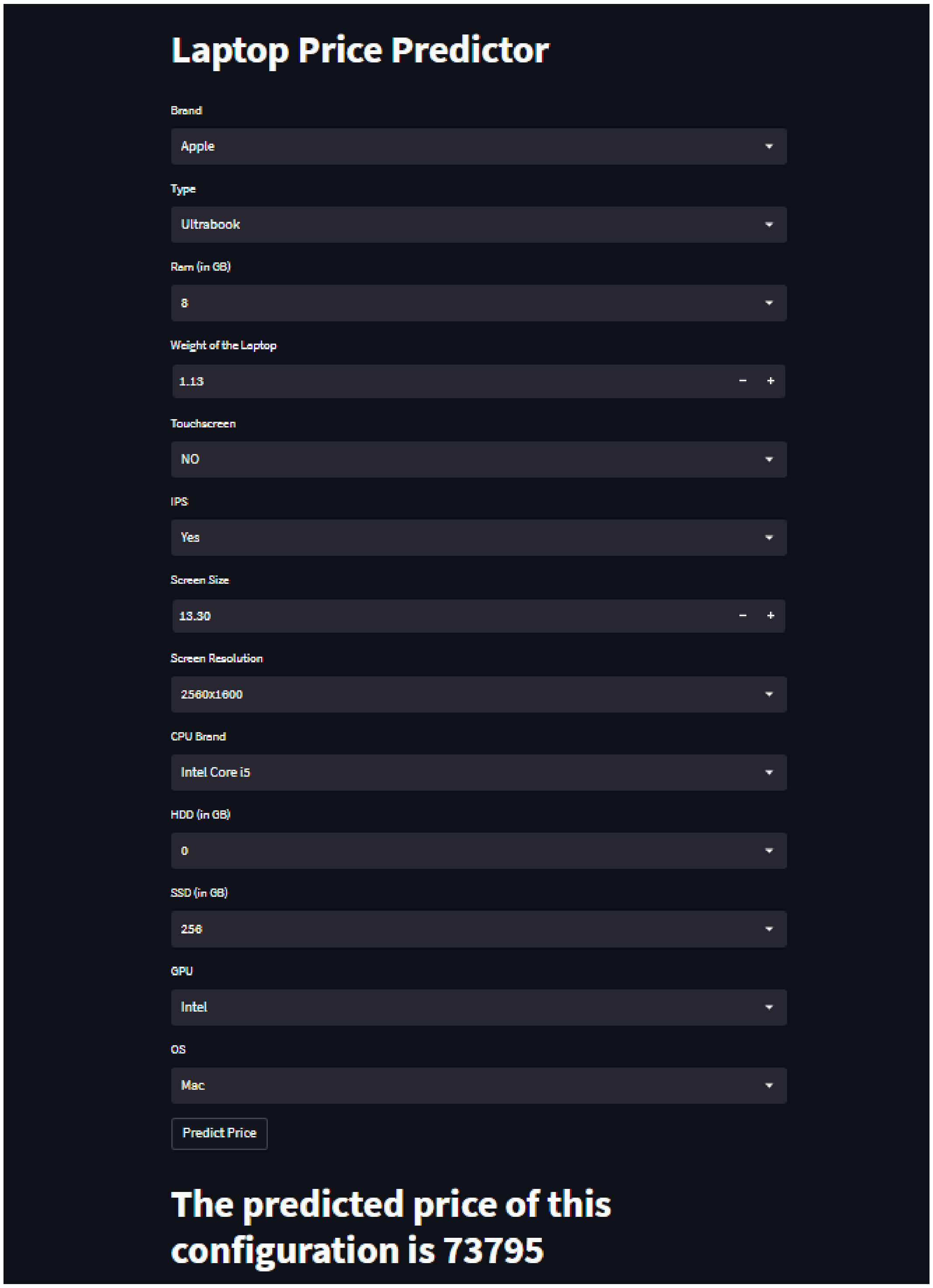
INTEGRATING ML MODEL WITH WEB APPLICATION:

Streamlit library is used to build this WebApp UI. Streamlit is an (open-source Python library) that makes it easy to create and share, custom web apps for machine learning and data science. Result with backend code is shown in following figures.



```python
import pickle
import streamlit as st
import sklearn
import numpy as np

pipe = pickle.load(open('pipe.pkl', 'rb'))
df = pickle.load(open('df.pkl', 'rb'))

st.title("Laptop Price Predictor")

# Brand
company = st.selectbox('Brand',df['Company'].unique())
# Type of Laptop
type = st.selectbox('Type', df['TypeName'].unique())
# RAM
ram = st.selectbox('Ram (in GB)', [2,4,6,8,12,16,24,32,64])
# Weight
weight = st.number_input('Weight of the Laptop')
# Touchscreen
touchscreen = st.selectbox('Touchscreen', ['NO','Yes'])
# IPS
ips = st.selectbox('IPS', ['No', 'Yes'])
# Screen size
screen_size = st.number_input('Screen Size')
resolution = st.selectbox('Screen Resolution', ['1920x1080', '1366x768', '1600x900', '3840x2160', '3200x1800', '2880x1800', '2560x1600', '2560x1440', '2304x1440'])
# CPU
cpu = st.selectbox('CPU Brand', df['Cpu Brand'].unique())
# HDD
hdd = st.selectbox('HDD (in GB)', [0,128,256,512,1024,2048])
# SSD
ssd = st.selectbox('SSD (in GB)', [0,8,128,256,512,1024])
# GPU
gpu = st.selectbox('GPU', df['Gpu Brand'].unique())
# OS
os = st.selectbox('OS', df['OS'].unique())

if st.button('Predict Price'):
    if touchscreen == 'Yes':
        touchscreen = 1
    else:
        touchscreen = 0

    if ips == "Yes":
        ips = 1
    else:
        ips = 0

    X_Res = int(resolution.split('x')[0])
    Y_Res = int(resolution.split('x')[1])
    ppi = (((X_Res)**2 + (X_Res)**2)**0.5)/screen_size
    query = np.array([company,type,ram,weight,touchscreen,ips,ppi,cpu,hdd,ssd,gpu,os])
    query = query.reshape(1,12)
    st.title("The predicted price of this configuration is " + str(int(np.exp(pipe.predict(query)[0]))))
```

OUTPUT SCREEN:



Laptop Price Predictor

Brand: Apple

Type: Ultrabook

Ram (in GB): 8

Weight of the Laptop: 1.13

Touchscreen: NO

IPS: Yes

Screen Size: 13.30

Screen Resolution: 2560x1600

CPU Brand: Intel Core i5

HDD (in GB): 0

SSD (in GB): 256

GPU: Intel

OS: Mac

The predicted price of this configuration is 73795

8. CONCLUSION AND FUTURE ENHANCEMENT

Predicting something through the application of machine learning using the Random Forest algorithm makes it easy for students, especially in determining the choice of laptop specifications that are most desirable for students to meet student needs and in accordance with the purchasing power of students. Students no longer need to look for various sources to find laptop specifications that are needed by them, because the laptop specifications from the results of the machine learning application have provided the most desirable specifications with their prices of laptops.

Using selenium, we will scrape the websites to increase our dataset for training and testing thus making the predictions more efficient.
We have used 11 parameters for making predictions, and for more efficient and accurate predictions the parameters that decide the price dependency can be increased.

REFERENCES

[1.] ARTICLE: Waqas Mirza (Iqra UniversityDepartment of Management Sciences, Pakistan), Irfan Manarv (IHI TEC University, Taxila Education City, Pakistan, July 2009.

[2.] NumPy: https://numpy.org/

[3.] Pandas: https://pandas.pydata.org/

[4.] Matplotlib: https://matplotlib.org/

[5.] Seaborn: https://seaborn.pydata.org/

[6.] Sklearn: https://scikit-learn.org/stable/

[7.] Dataset: https://www.kaggle.com/datasets/muhammetvarl/laptop-price