



**CHANDIGARH  
UNIVERSITY**

Discover. Learn. Empower.

End semester final project  
(1st) - 2022

# **DISRUPTIVE TECHNOLOGY**

---

**Mayank Singh Rana**

21BCS2409

**Joydeep Sen**

21BCS3234



## Introduction

A laptop, which is sometimes known as a notebook, is a portable computer that is more efficient and powerful than a desktop computer. For several hours, portable computers run on AC power or batteries such as NiMH, NiCad, or Li-ion packs.

To compete with alternative devices, such as tablets, most laptop computers are growing slimmer and lighter. Some laptops, particularly those with smaller form factors, are referred to as notebook computers.

## Personal Details

### Team member 1

- ❖ Name: Mayank Singh Rana
- ❖ UID: 21BCS2409
- ❖ GitHub: [github.com/mayankrana29](https://github.com/mayankrana29)
- ❖ Telephone : +91 9871872352
- ❖ Language: English, Hindi
- ❖ Email: [mayankrana0078@gmail.com](mailto:mayankrana0078@gmail.com)
- ❖ LinkedIn: [linkedin.com/in/mayankrana29](https://linkedin.com/in/mayankrana29)

### Team member 2

- Name: Joydeep Sen
- UID: 21BCS3234
- GitHub: [github.com/Joyyojpeed](https://github.com/Joyyojpeed)
- Telephone: +91 75969 23557
- Languages: English, Hindi
- Email: [Jds472016@gmail.com](mailto:Jds472016@gmail.com)



# Project Proposal

## Laptop Insider

### Problem statement and overview

We want to help people that have less closure to the technical field and want to choose a device for themselves so they can use our application called LAPTOP INSIDER.

So, the problem is that if any user wants to buy a laptop or any other electronic accessories, then our application should be compatible to provide a tentative price of laptop according to 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 pre-processing that will drive your interest in our project

### Key features and benefits

#### ❖ Save time and energy

Basically, it will save the time for the research part of the customer that we will provide all the information of price and other things user just have to put his needs in the site and he is good to go.

#### ❖ Building a custom laptops and pc

As most the customers need their custom laptop and pc according to the work they do such as some do graphic designing video editing rendering high-end games and some do the basic task on their pc so our model can help the customer and also the company to build custom laptops and pc for them.

### ❖ Used as a Guide

Our model can also give suggestions to customers that how will you buy this laptop instead of this laptop by MACHINE LEARNING model by comparing configuration and price of other laptops and suggest the user/customer buy this item.

## Methodology

### ❖ Description of the dataset

The dataset used for this research purpose was the dataset which is taken from Kaggle and further filtered by JOYDEEP and it consists of 12 attributes (Company, TypeName, Inches, screen resolution, CPU, RAM, Memory, GPU, Opsys, weight, price)

### ❖ Pre-processing of dataset

No null values exist in the dataset. However, there were numerous outliers to deal with, and the dataset was not distributed evenly. We employed two different ways. One that is free of outliers and uses a feature selection method instead of applying the results directly. inputs to machine learning algorithms, as well as the outcomes, weren't looking good. However, following the use of the dataset's normal distribution for

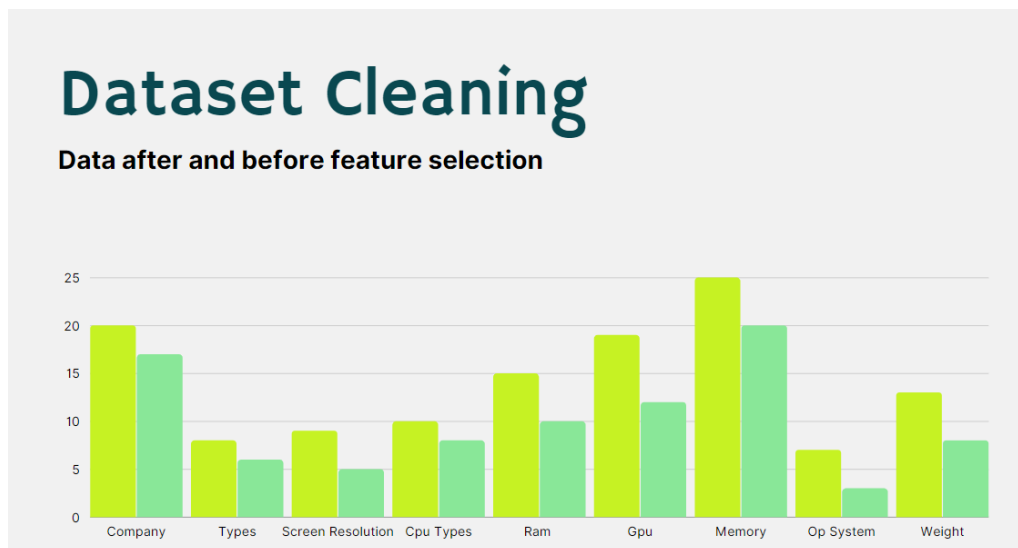
Overcoming the problem of overfitting and then using Isolation Forest to solve the problem findings obtained are pretty promising in terms of outlier detection. Plotting techniques. Checking the data's skewness, detecting outliers, and other procedures were applied. the way the info is dispersed. All of these methods of pre-processing are crucial. When data is passed for categorization or prediction, the function of the data is important.

### ❖ Checking the distribution of the data

When it comes to predicting or classifying a problem, the distribution of the data is crucial. As a result, we must balance the dataset, or it will get overfit. This will aid the model in identifying a pattern in the dataset that can be used to predict prices.

## ❖ Dataset Cleaning

As in every machine learning or any other project, first thing is to do is clean data according to the need of the projects. So in our projects, we have to clean data according to technical terms such as IPS display panel, RAM, ROM, SSD, etc.



## ❖ Feature Selection

When doing feature selection, the Lasso algorithm, which is a part of embedded techniques, is used to pick the features and just the significant ones. It outperforms the filter approaches in terms of predicted accuracy. It generates the best feature subsets for the algorithm in use. Select the selected features from the model in the scikit-learn library, which is a part of feature selection.

## ❖ Checking Duplicate Values in the Data

Duplicate objects must be handled, else the generalisation of the constructed model will be harmed. There's a potential that if duplicates aren't handled properly, they'll show up in the test dataset as well as the training dataset.

## Software used and libraries:



### ❖ Sklearn:

It is free software ML Library (pre defined dataset). It features Various classification, regression, and clustering algorithms include vector machines

### ❖ Pickle:

Library is used for serializing and de-serializing a Python object structure. It is a way to convert a python object (list, dict, etc.) into a character stream. We connected our application to model by using this.

### ❖ PyCharm:

IDE Environment.

### ❖ Google Collab & Jupyter:

Notebooks used for combining executable code and rich text in a single document.

---

❖ **Github:**

The open-source app is used for hosting and version control.

❖ **Heroku:**

Heroku is the quickest way for a company to become an apps company. Heroku is a service that enables companies to spend their time developing and deploying apps that immediately start producing value. An app starts impacting the world when customers start interacting with it.

❖ **Seaborn:**

Seaborn works best with Pandas DataFrames and arrays that contain a whole data set. Remember that DataFrames are a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable.

❖ **Pandas:**

Pandas is a high-level data manipulation tool. It is built on the NumPy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables.

❖ **Matplotlib:**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose. Matplotlib consists of several plots like line, bar, scatter, histogram, etc.

❖ **PickleDB:**

PickleDB is a simple and lightweight data store. It stores the data as a key-value store. It is based on the Python module named SimpleJSON, which allows

developers to handle JSON (JavaScript Object Notation) in a simple and faster manner.

❖ **Kaggle:**

Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

## CODE

Link of Jupyter notebook: [rb.gy/2axblp](https://rb.gy/2axblp)

### → TRAINING MODEL CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('laptop_data.csv')

df.head()

df.shape

df.info()

df.duplicated().sum()

df.isnull().sum()

df.drop(columns=['Unnamed: 0'],inplace=True)

df.head()

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
```

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

```
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.sample(5)
```

```
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['lps'].value_counts().plot(kind='bar')
sns.barplot(x=df['lps'],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']))
```



```
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
```

## **Linear regression**

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf',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))
```

## ***Ridge Regression***

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf', 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))
```

## ***Lasso Regression***

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf', 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))
```

## **KNN**

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf',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))
```

## **Decision Tree**

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf',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))
```

## **SVM**

```
step1 = ColumnTransformer(transformers=[

    ('col_tnf',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))
```

## **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))
```

## **ExtraTrees**

```
step1 = ColumnTransformer(transformers=[  
  
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])  
  
],remainder='passthrough')  
  
step2 = ExtraTreesRegressor(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))
```

## AdaBoost

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

step2 = AdaBoostRegressor(n_estimators=15,learning_rate=1.0)

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))
```



## Gradient Boost

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])  
],remainder='passthrough')  
  
step2 = GradientBoostingRegressor(n_estimators=500)  
  
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))
```

## XgBoost

```
step1 = ColumnTransformer(transformers=[  
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])  
],remainder='passthrough')  
  
step2 = XGBRegressor(n_estimators=45,max_depth=5,learning_rate=0.5)  
  
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))
```

## Voting Regressor

```
from sklearn.ensemble import VotingRegressor,StackingRegressor  
  
step1 = ColumnTransformer(transformers=[  
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])  
],remainder='passthrough')  
  
rf =  
RandomForestRegressor(n_estimators=350,random_state=3,max_samples=0.5,max_features=0.75,max_depth=15)  
  
gbdt = GradientBoostingRegressor(n_estimators=100,max_features=0.5)  
  
xgb = XGBRegressor(n_estimators=25,learning_rate=0.3,max_depth=5)  
  
et =  
ExtraTreesRegressor(n_estimators=100,random_state=3,max_samples=0.5,max_features=0.75,max_depth=10)  
  
step2 = VotingRegressor([('rf', rf), ('gbdt', gbdt), ('xgb',xgb), ('et',et)],weights=[5,1,1,1])  
  
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))
```

## Stacking

```
from sklearn.ensemble import VotingRegressor, StackingRegressor

step1 = ColumnTransformer(transformers=[

    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0,1,7,10,11])

], remainder='passthrough')

estimators = [

    ('rf',
     RandomForestRegressor(n_estimators=350, random_state=3, max_samples=0.5, max_features=0.75, max_depth=15)),

    ('gbdt', GradientBoostingRegressor(n_estimators=100, max_features=0.5)),

    ('xgb', XGBRegressor(n_estimators=25, learning_rate=0.3, max_depth=5))

]

step2 = StackingRegressor(estimators=estimators, final_estimator=Ridge(alpha=100))

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))
```

### **Exporting the Model**

```
import pickle
```

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

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

```
df'
```

```
X_train
```

**“END OF TRAINING OF MODEL”**

Link of APP: [//rb.gy/qsoemk](https://rb.gy/qsoemk)

### **→ CODE OF APP.py**

```
import streamlit as st
```

```
import pickle
```

```
import numpy as np
```

```
# import the model
```

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

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

```
st.title("Laptop 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

resolution = st.selectbox('Screen
Resolution',['1920x1080','1366x768','1600x900','3840x2160','3200x1800','2880x1800','2560x1
600','2560x1440','2304x1440'])

#cpu

cpu = st.selectbox('CPU',df['Cpu brand'].unique())

hdd = st.selectbox('HDD(in GB)',[0,128,256,512,1024,2048])

ssd = st.selectbox('SSD(in GB)',[0,8,128,256,512,1024])

gpu = st.selectbox('GPU',df['Gpu brand'].unique())

os = st.selectbox('OS',df['os'].unique())

if st.button('Predict Price'):
```



```
# query

ppi = None

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) + (Y_res**2))**0.5/screen_size

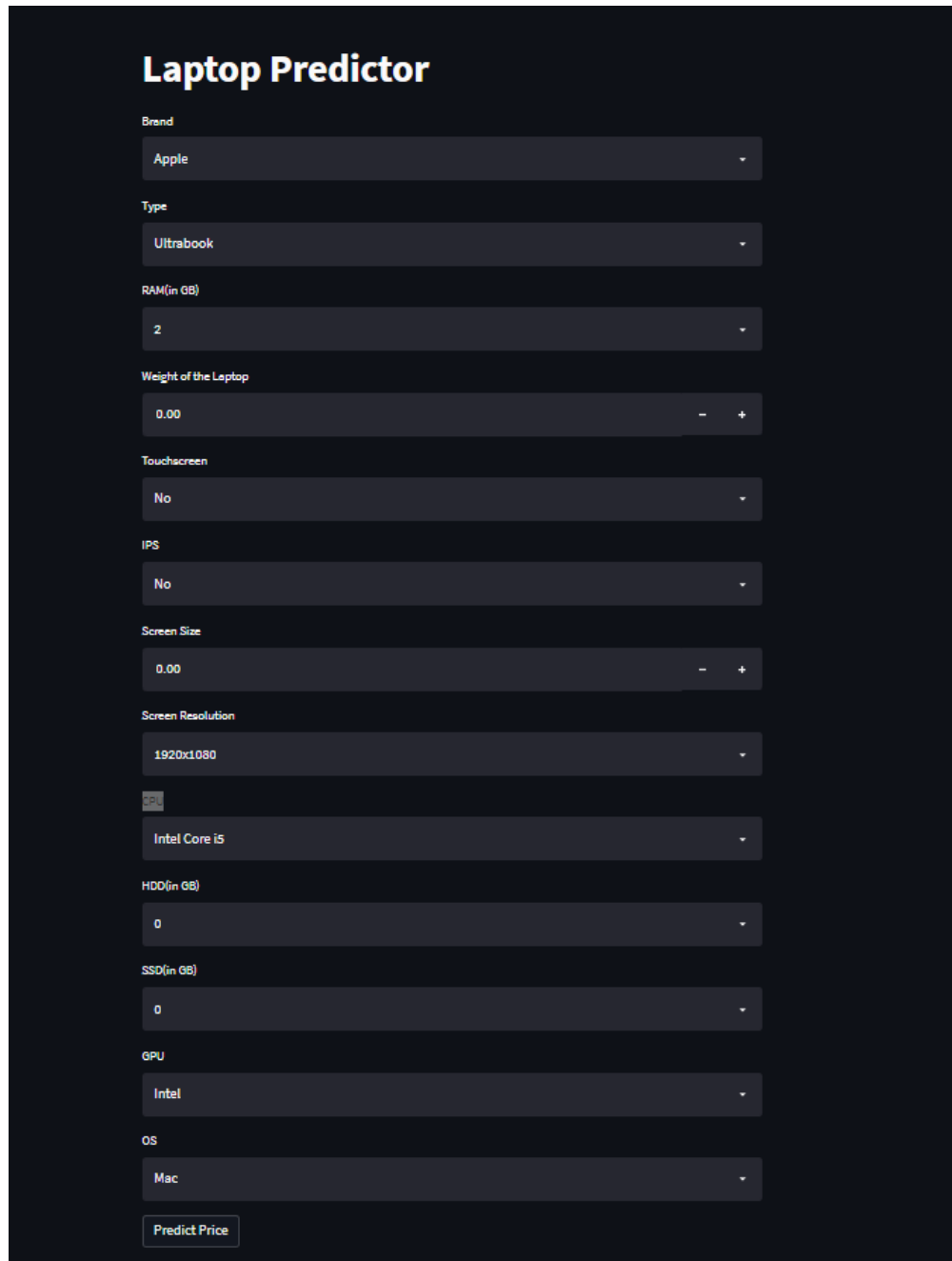
query = np.array([company,type,ram,weight,touchscreen,ips,ppi,cpu,hdd,ssd,gpu,os])

query = query.reshape(1,12)

st.title("The predicted price of this configuration is " + str(int(np.exp(pipe.predict(query)[0]))))
```

## OUTPUT

Link for the live project: <https://lpp-kaigon.herokuapp.com/>



The screenshot displays a web application titled "Laptop Predictor" with a dark theme. It features a series of input fields for various laptop specifications, each with a label and a dropdown menu. The specifications include Brand (Apple), Type (Ultrabook), RAM (2 GB), Weight of the Laptop (0.00), Touchscreen (No), IPS (No), Screen Size (0.00), Screen Resolution (1920x1080), CPU (Intel Core i5), HDD (0 GB), SSD (0 GB), GPU (Intel), and OS (Mac). At the bottom, there is a "Predict Price" button.

Brand	Type	RAM(in GB)	Weight of the Laptop	Touchscreen	IPS	Screen Size	Screen Resolution	CPU	HDD(in GB)	SSD(in GB)	GPU	OS
Apple	Ultrabook	2	0.00	No	No	0.00	1920x1080	Intel Core i5	0	0	Intel	Mac

Predict Price

## ***LET TEST THE MODEL BY ENTERING MY LAPTOP SPECS AND VERIFYING THE SAME***

### Laptop Predictor

Brand

Asus

Type

Ultrabook

RAM(= GB)

8

Weight of the Laptop

1.70

Touchscreen

No

IPS

Yes

Screen Size

15.00

Screen Resolution

1920x1080

CPU

Intel Core i5

HDD(= GB)

5004

SSD(= GB)

256

GPU

Intel

OS

Windows

Predict Price


The predicted price of this configuration is 57454

## PREDICTED VALUE

The predicted price of this configuration is 57454


## ACTUAL VALUE


[Back to results](#)




ASUS VivoBook Ultra 15 (2020) Intel Core i5-1135G7 11th Gen, 15.6-inch FHD Thin and Light Laptop (8GB/1TB HDD + 256GB SSD/Windows 10/Office 2019/Iris Xe Graphics/Dreamy White/1.8 Kg), X513EA-BQ503TS

Visit the ASUS Store  
★★★★★ 351 ratings | 141 answered questions


M.R.P.: ₹73,990.00  
Price: **₹55,990.00**   
You Save: ₹18,000.00 (24%)  
Inclusive of all taxes





EMI starts at ₹2,636. No Cost EMI available EMI options 

 Save Extra with 5 offers


No Cost EMI: Avail No Cost EMI on select cards for orders above ₹3... | Details

Exchange Offer: Up to ₹18,100.00 off on Exchange

 See 3 more

M.R.P.: ₹73,990.00

Price: **₹55,990.00** 

You Save: ₹18,000.00 (24%)

Inclusive of all taxes

EMI starts at ₹2,636. No Cost EMI  
available [EMI options](#) 

*“Hence our model is correct and working properly “*

---

## Deliverables

A laptop, also known as a laptop computer or notebook computer, is a tiny, portable computer featuring a screen and an alphanumeric keypad. Although 2-in-1 PCs with a detachable keyboard are commonly marketed as laptops or as having a laptop mode, they typically have a clamshell form factor, with the screen mounted on the inside of the higher lid and the keyboard mounted on the inside of the lower lid. Laptops are suited for mobile use because they can be folded shut for traveling. Its name originates from the fact that it was originally intended to be used on a person's lap. Laptops are now utilized in a number of scenarios, including at work, in schools, for gaming, web browsing, personal multimedia, and more.

As of 2021, the phrases laptop computer and notebook computer are interchangeable in American English; nevertheless, in other dialects of English, one or the other may be preferred. Originally, the names 'notebook computers' and 'notebooks' referred to a specific size of the laptop (smaller and lighter than popular laptops at the time), but the terms have since come to signify the same thing, and notebook no longer refers to any specific size.

## How to use and run this prediction model on your pc or anywhere else.

**“Make sure git is installed in your computer”**

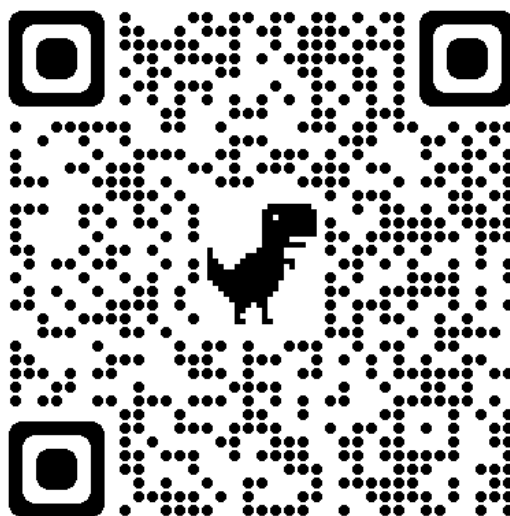
→ GIT Clone: [Kaigon0/Distructive-techonology-project: In this repository i am uploading my project code that i related to machine learning. In project i made a laptop price prediction model that is made on the basics of python classification and regression model. I hope you will like it. \(github.com\)](#)

→ Live working of model: <https://lpp-kaigon.herokuapp.com/>

***“YOU CAN ALSO SCAN THIS QR TO REACH THE MODEL”***



→ GITHUB Profile link: [Kaigon0 \(Mayank singh rana\) \(github.com\)](https://github.com/Kaigon0)



## References

- Real-time automatic price prediction for eBay online trading  
<https://axon.cs.byu.edu/papers/raykhel.iaai09.pdf>
- Web scrapping laptop prices to estimate hedonic models and extensions to other predictive methods.  
[https://eventos.fgv.br/sites/eventos.fgv.br/files/arquivos/u161/ottawa\\_insee\\_v1.pdf](https://eventos.fgv.br/sites/eventos.fgv.br/files/arquivos/u161/ottawa_insee_v1.pdf)
- YouTube:
  - <https://www.youtube.com/c/CodeWithHarry>
  - [https://www.youtube.com/channel/UCBwmMxybNva6P\\_5VmxjzwqA](https://www.youtube.com/channel/UCBwmMxybNva6P_5VmxjzwqA)
  - <https://www.youtube.com/c/CampusX-official>
- Articles:
  - <https://www.analyticsvidhya.com/blog/2021/11/laptop-price-prediction-practical-understanding-of-machine-learning-project-lifecycle/>