Capstone Project Data Science Nanodegree March 10th , 2024

Laptop price predictions

Project overview:-

This project aims to develop a machine learning algorithm to predict laptop prices based on specific criteria. The dataset comprises information such as manufacturer, laptop type, screen size and resolution, RAM, memory, GPU, weight, and price. Price variation is influenced by these specifications. The project involves conducting

Problem Statement:-

Predicting the price of the laptop which will be possible higher level of response from user based on Manufacturer, type of the laptop, screen size and resolution, Ram, memory, GPU, weight and price.

Description of input data ;- The provided dataset contains information about laptops from different companies, including their specifications and prices. Here's a description of the data:

- Company: The manufacturer of the laptop (e.g., Apple, HP).
- TypeName: The type or category of the laptop (e.g., Ultrabook, Notebook).
- Inches: The screen size of the laptop in inches.
- ScreenResolution: The resolution of the laptop screen (e.g., IPS Panel Retina Display 2560x1600, Full HD 1920x1080).
- Cpu: The processor (CPU) of the laptop, including its model and clock speed (e.g., Intel Core i5 2.3GHz, Intel Core i7 2.7GHz).
- Ram: The amount of random-access memory (RAM) in the laptop.
- Memory: The storage capacity of the laptop (e.g., SSD, Flash Storage).
- Gpu: The graphics processing unit (GPU) of the laptop (e.g., Intel Iris Plus Graphics 640, AMD Radeon Pro 455).
- OpSys: The operating system installed on the laptop (e.g., macOS, No OS).
- Weight: The weight of the laptop in kilograms.
- Price: The price of the laptop.

Overall, this dataset provides detailed information about various laptops, including their hardware specifications, operating systems, and prices. Such data can be utilized for tasks such as analyzing market trends, comparing laptop features, and predicting laptop prices based on their specifications.

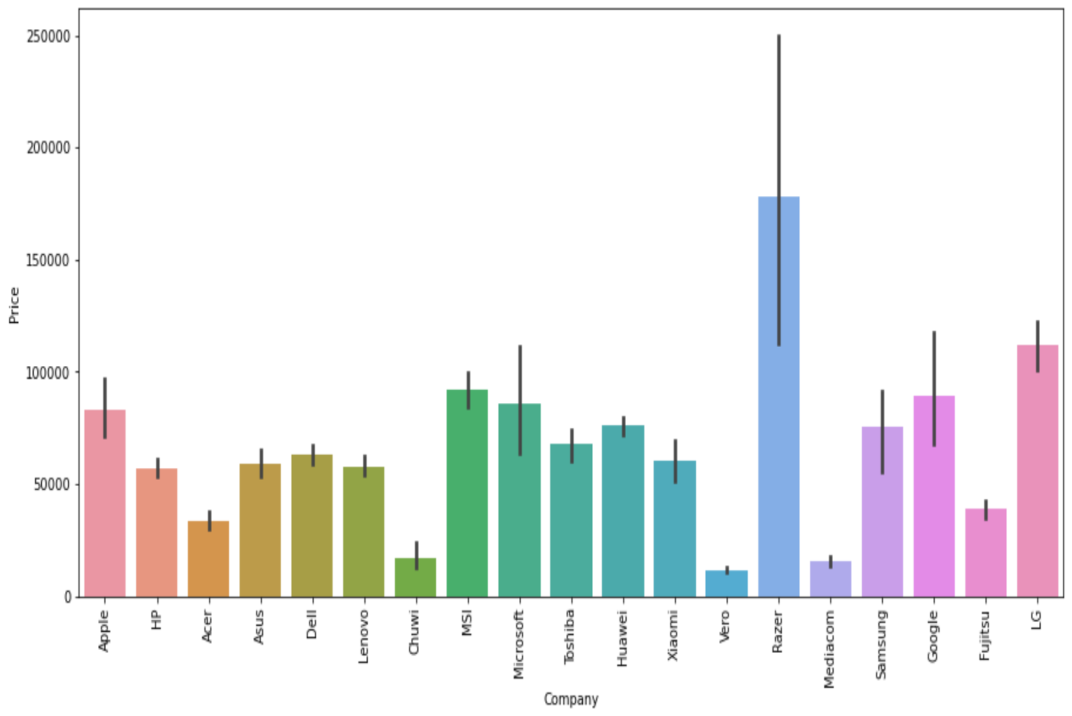
Steps to solve the problem

The proposed solution involves utilizing multiple regression and decision tree-based machine learning algorithms to predict laptop prices based on their specifications. Let's discuss the overall architecture or workflow of the solution and how the various components fit together to achieve the desired outcome:

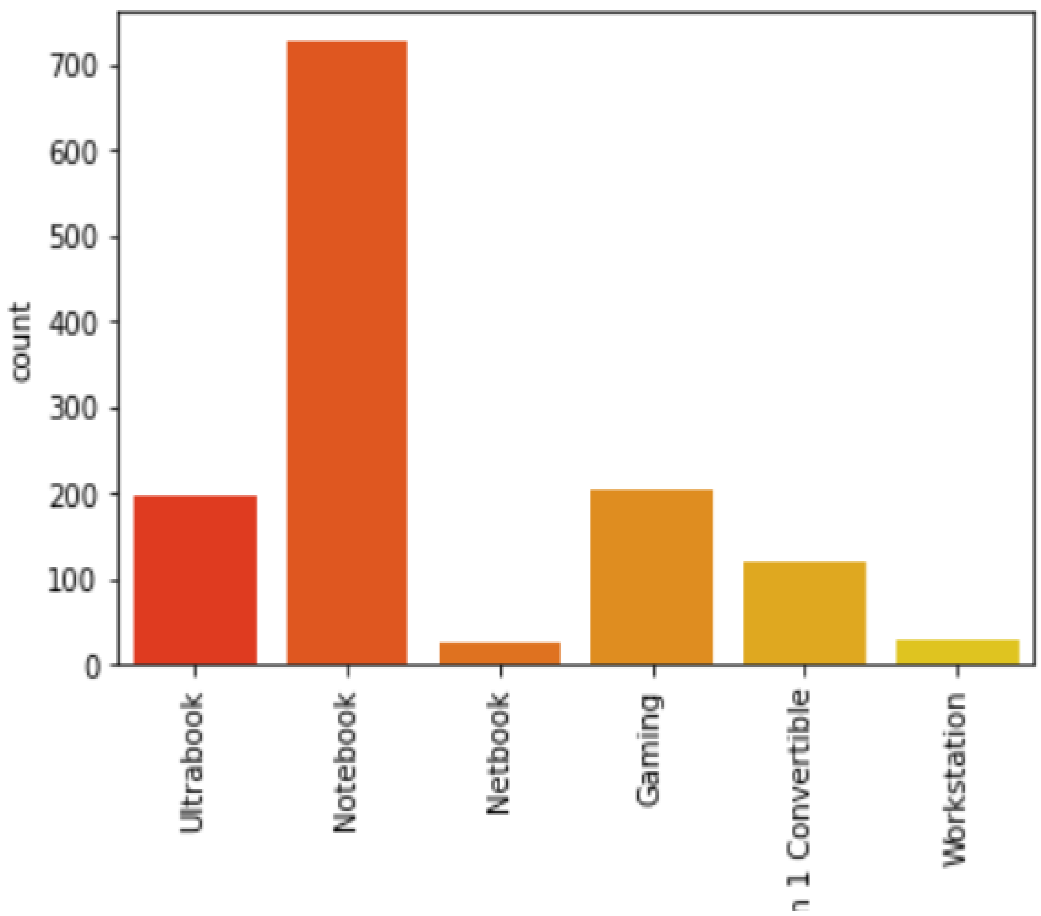
1. *Data Preprocessing*:

- The first step involves preprocessing the dataset. This includes handling missing values, encoding categorical variables, scaling numerical features, and splitting the data into training and testing sets.

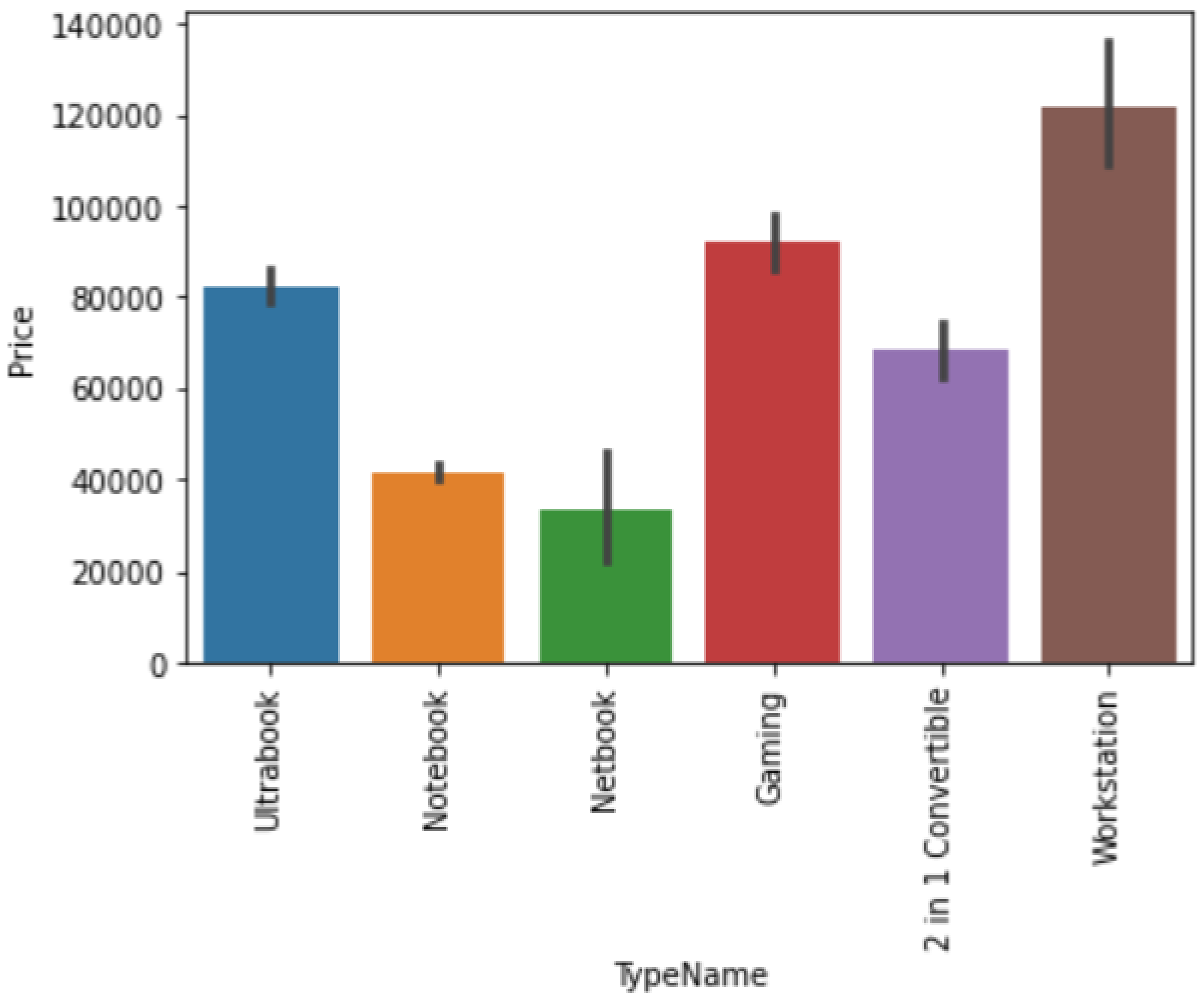
1. Checking the distribution of price so see the price ranges.
2. Checking the count of laptops against company,Typename,Ram,OpSys.
3. Check for the price ranges against each company.



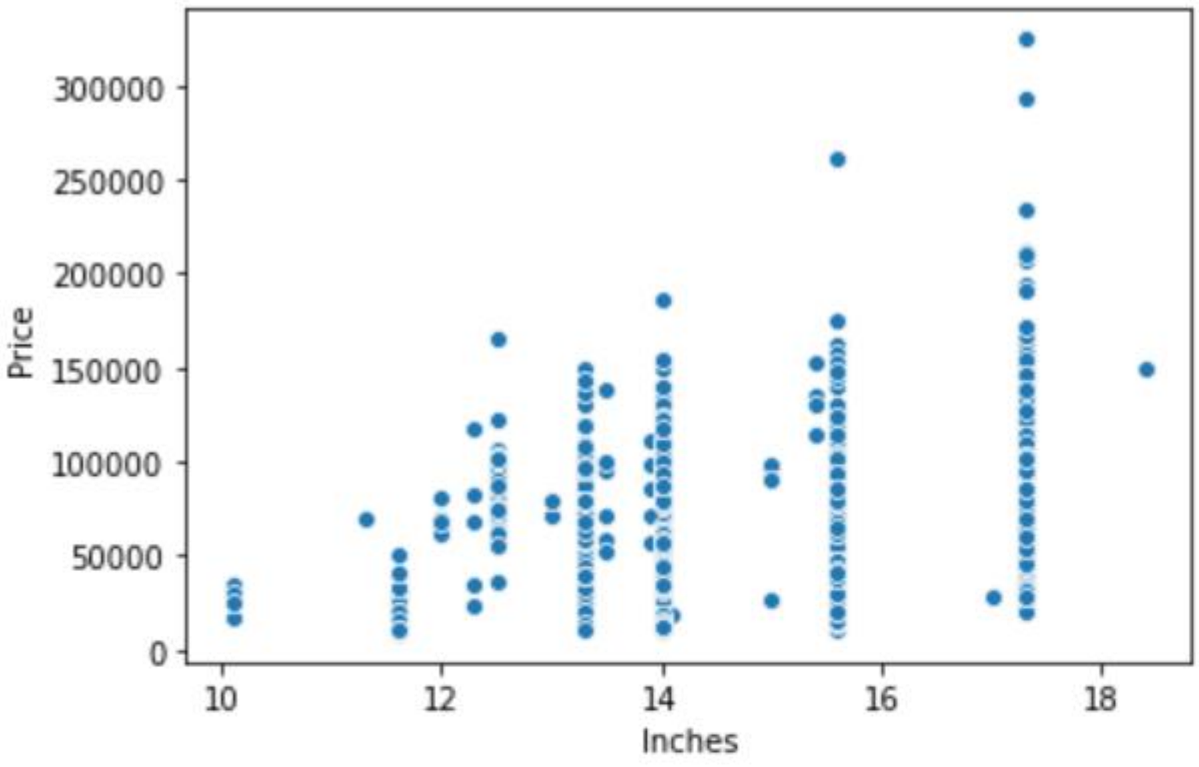
4. Check for the count of various type of laptop.



5. Similarly check for the variation of price against screen, laptop type and price.



140000
120000
100000
80000
60000
40000
20000
0
Price
Ultrabook
Notebook
Netbook
Gaming
2 in 1 Convertible
Workstation
TypeName



300000
250000
200000
150000
100000
50000
0
Price
10
12
14
16
18
Inches

6. Convert the Screen-resolution column into X&Y and create another KPI named as PPI by using below formulae and remove screen size and resolution columns.

$$PPI(pixelsperinch) = \frac{\sqrt{X_resolution^2 + Y_resolution^2}}{inches}$$

7. Create columns named as IPS, Touchscreen and fill bainary values in it to use further.
8. Check for the correlation of each column against price.
9. Remove those columns which are not contributing much in the price.
10. Convert the memory into columns (e.g. HDD, SDD, Storage & Flash storage) to use it further in the modeling.
11. Apply the same step in GPU & OpSys columns as well.

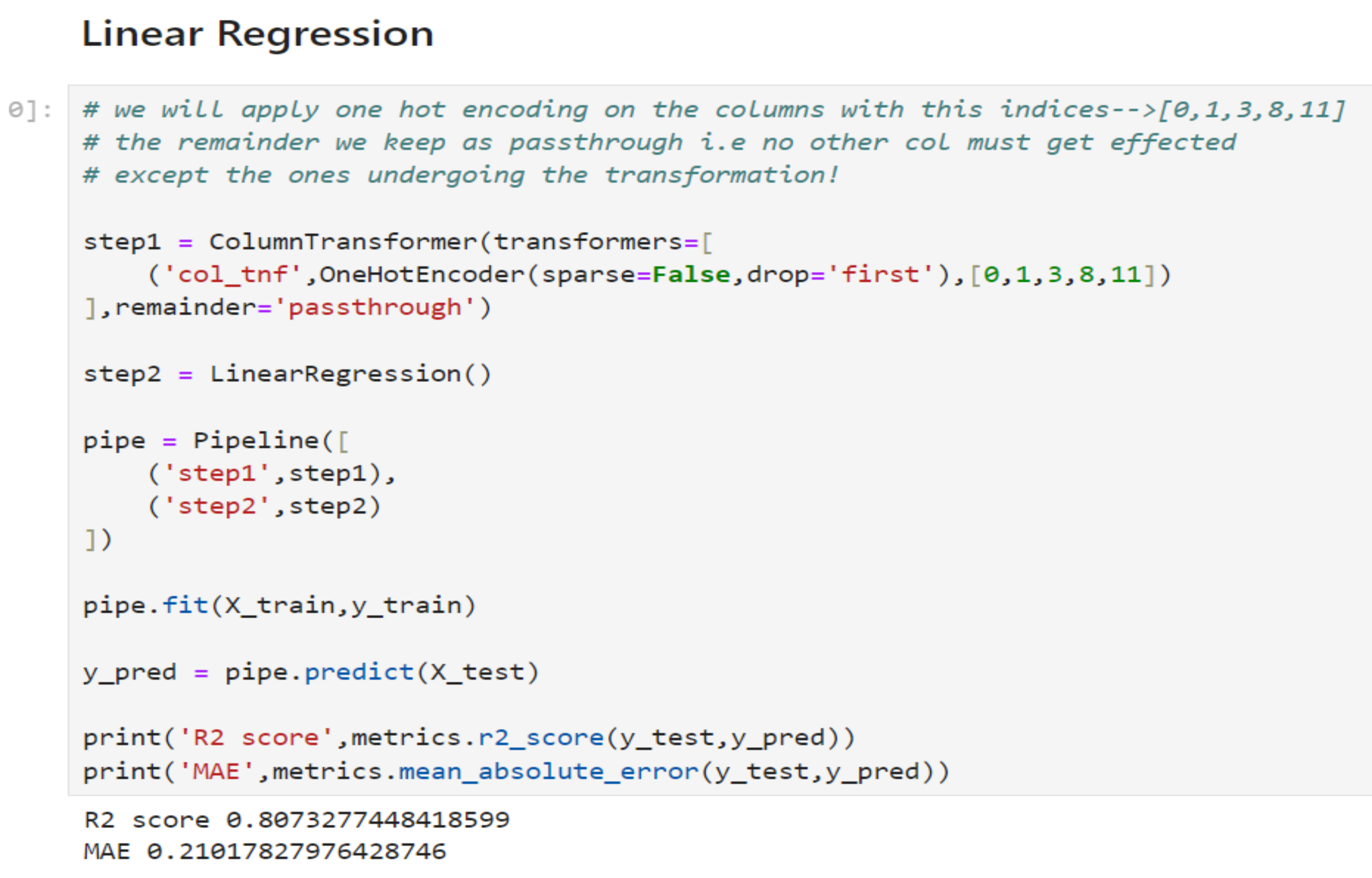
2. *Feature Engineering*:

- Next, feature engineering techniques may be applied to extract useful information from the existing features or create new features that may enhance the predictive performance of the models. This could involve feature transformation, dimensionality reduction, or interaction features.

3. *Model Training*:

- Once the data is preprocessed and features are engineered, the next step is to train multiple regression and decision tree-based models on the training data. The models considered in this solution are:

  - Linear Regression



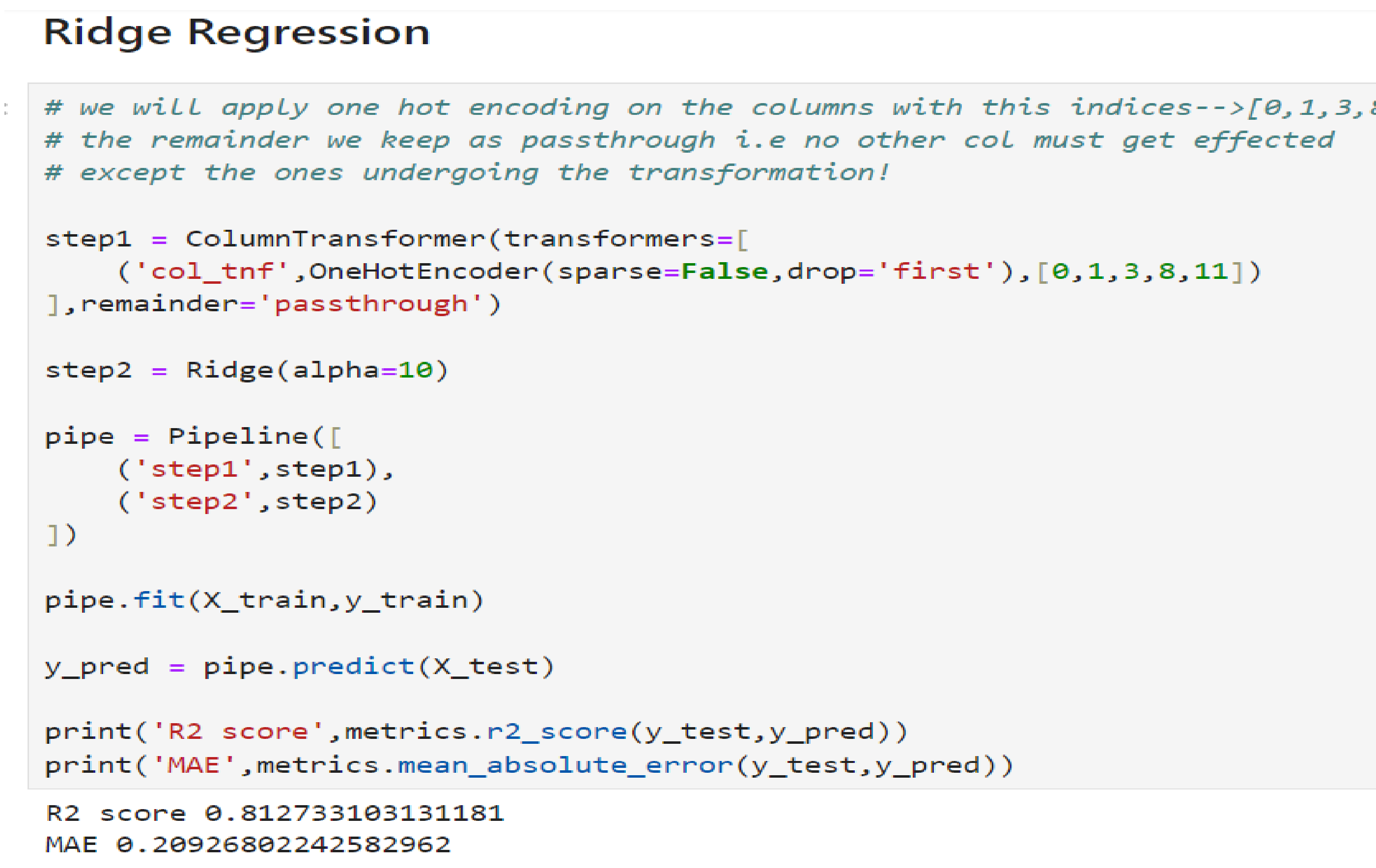
Linear Regression

0]:

```python
# we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],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',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8073277448418599
MAE 0.21017827976428746
```

- Ridge Regression



Ridge Regression

```
# we will apply one hot encoding on the columns with this indices-->[0,1,3,
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],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',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.812733103131181
MAE 0.20926802242582962
```

- Lasso Regression

LassoRegression

```
# we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],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',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8071857196899418
MAE 0.21114350716913166
```

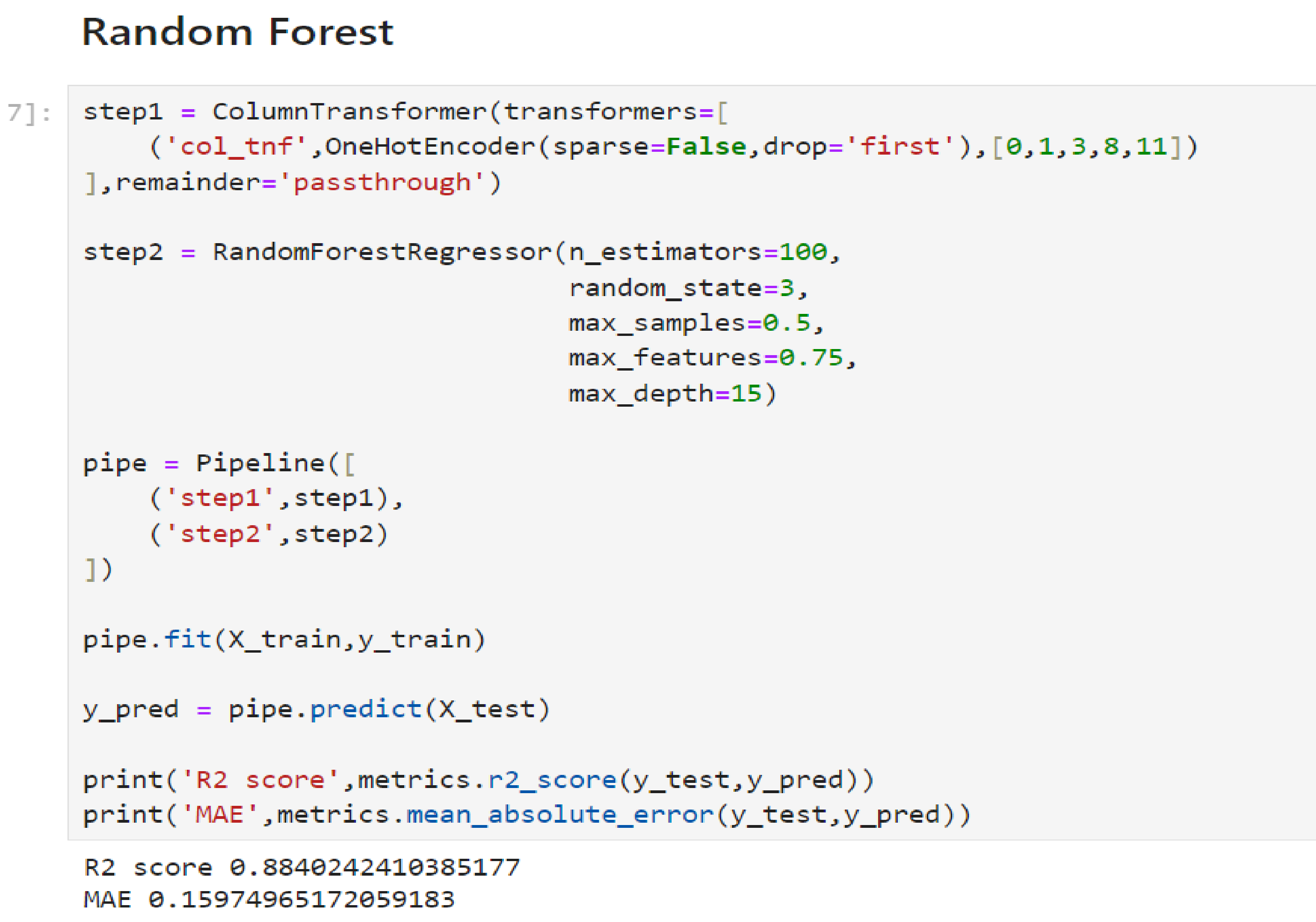
- Decision Tree

Decision Tree

```
.04]: # we will apply one hot encoding on the columns with this indices-->[0,1,3,8,11]
# the remainder we keep as passthrough i.e no other col must get effected
# except the ones undergoing the transformation!

step1 = ColumnTransformer(transformers=[
    ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
],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',metrics.r2_score(y_test,y_pred))
print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8437664803528917
MAE 0.1808159839491631
```

- Random Forest

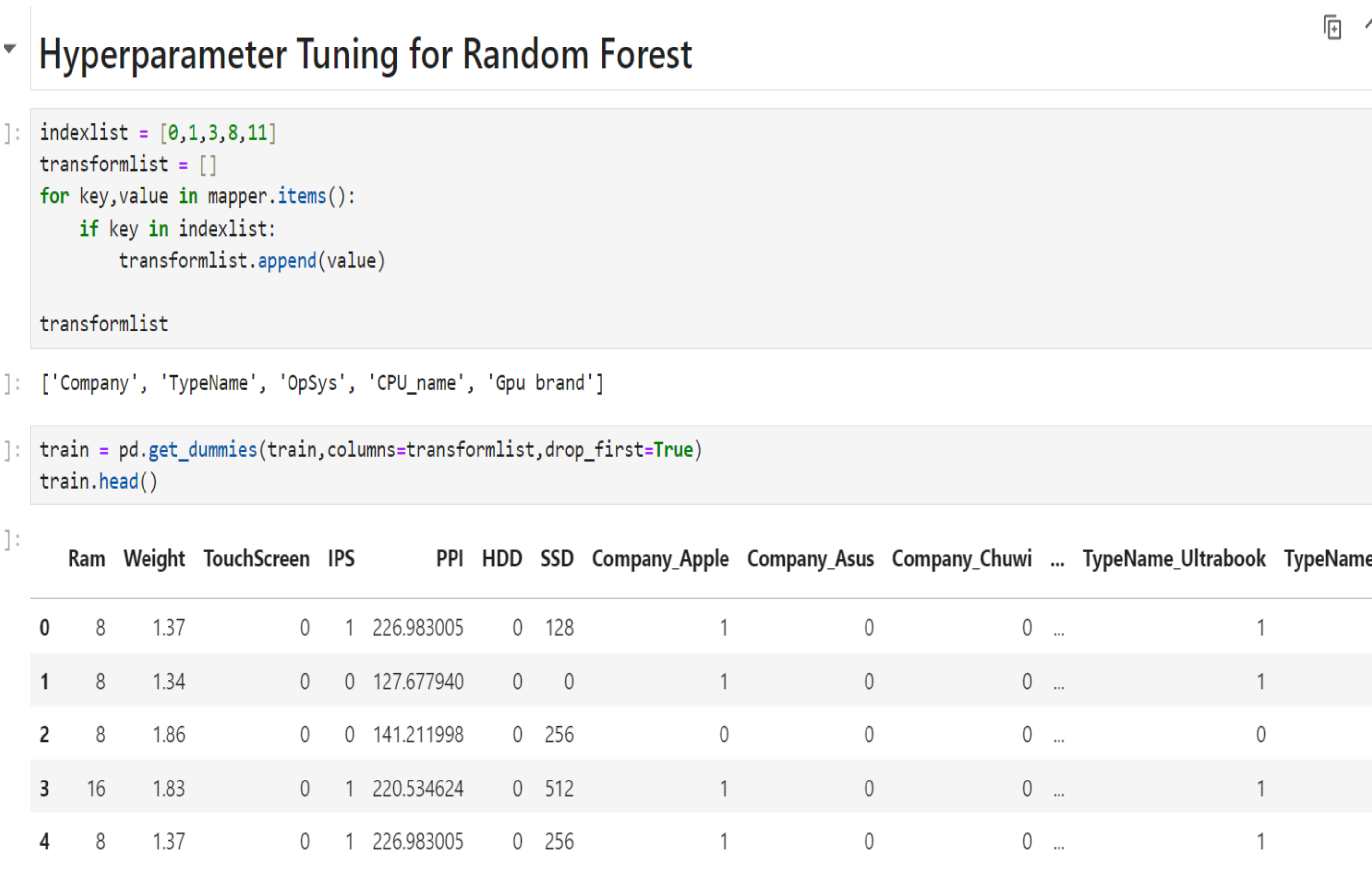


Random Forest

```
7]: step1 = ColumnTransformer(transformers=[
        ('col_tnf',OneHotEncoder(sparse=False,drop='first'),[0,1,3,8,11])
    ],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',metrics.r2_score(y_test,y_pred))
    print('MAE',metrics.mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8840242410385177
MAE 0.15974965172059183
```

4. *Hyperparameter Tuning*:



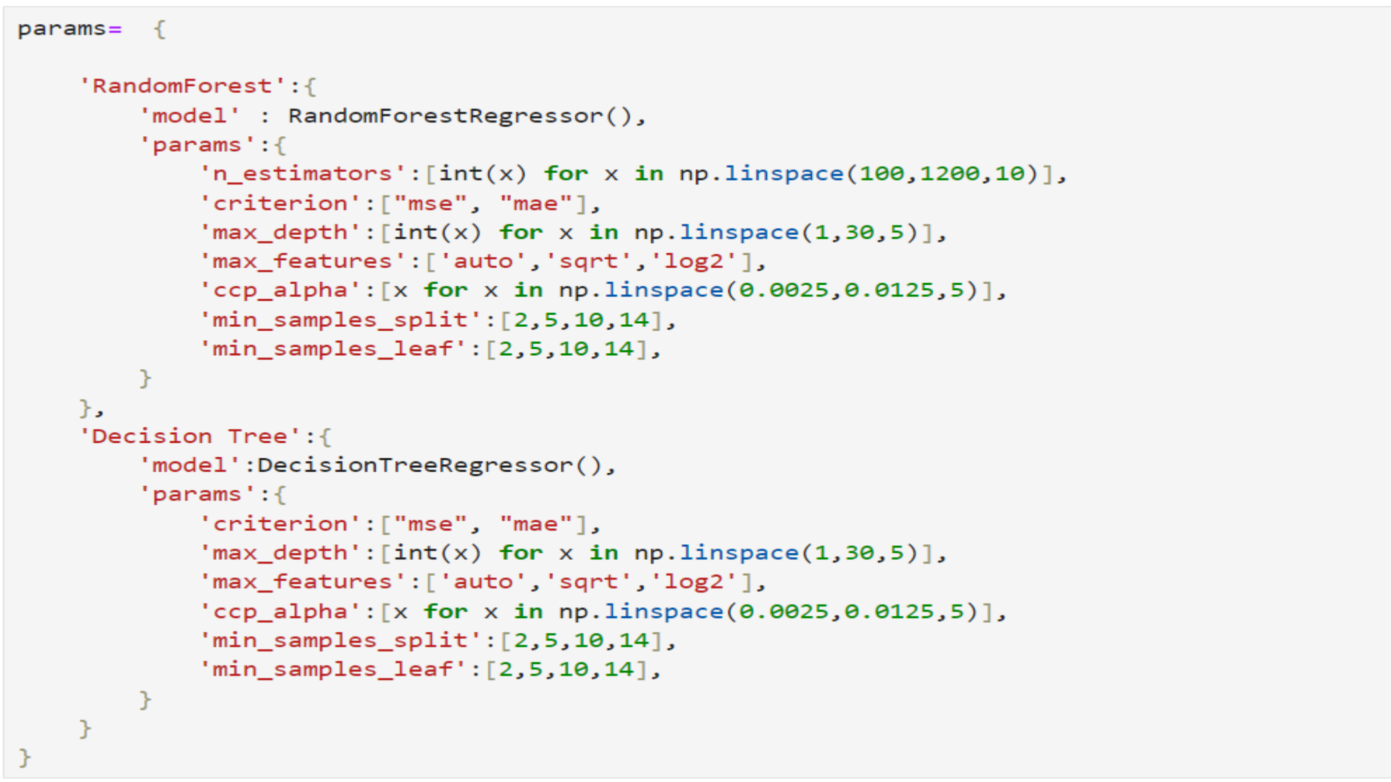
Hyperparameter Tuning for Random Forest

```python
indexlist = [0,1,3,8,11]
transformlist = []
for key,value in mapper.items():
    if key in indexlist:
        transformlist.append(value)

transformlist
```

```
['Company', 'TypeName', 'OpSys', 'CPU_name', 'Gpu brand']
```

```python
train = pd.get_dummies(train,columns=transformlist,drop_first=True)
train.head()
```

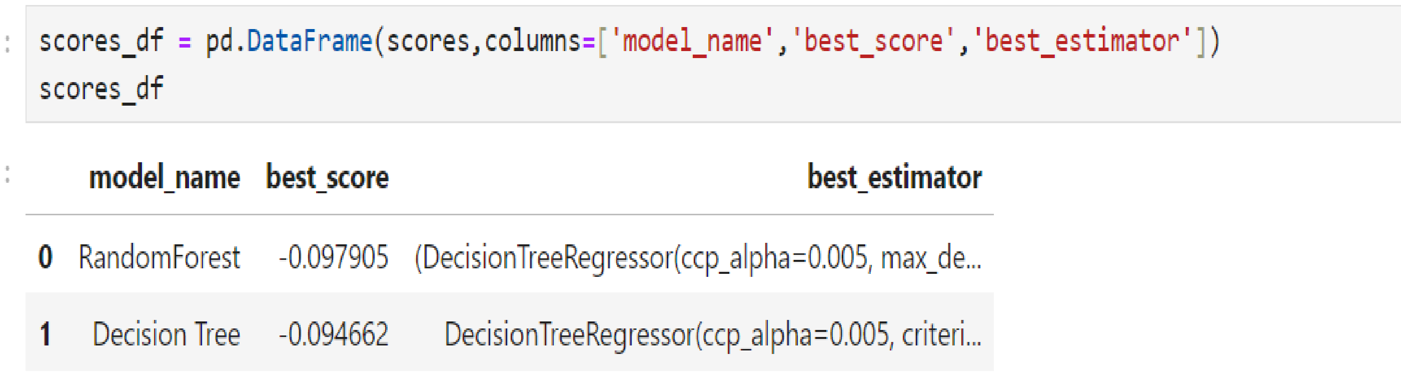
	Ram	Weight	TouchScreen	IPS	PPI	HDD	SSD	Company_Apple	Company_Asus	Company_Chuwi	...	TypeName_Ultrabook	TypeName
0	8	1.37	0	1	226.983005	0	128	1	0	0	...	1	
1	8	1.34	0	0	127.677940	0	0	1	0	0	...	1	
2	8	1.86	0	0	141.211998	0	256	0	0	0	...	0	
3	16	1.83	0	1	220.534624	0	512	1	0	0	...	1	
4	8	1.37	0	1	226.983005	0	256	1	0	0	...	1	



```python
params=  {

    'RandomForest':{
        'model' : RandomForestRegressor(),
        'params':{
            'n_estimators':[int(x) for x in np.linspace(100,1200,10)],
            'criterion':["mse", "mae"],
            'max_depth':[int(x) for x in np.linspace(1,30,5)],
            'max_features':['auto','sqrt','log2'],
            'ccp_alpha':[x for x in np.linspace(0.0025,0.0125,5)],
            'min_samples_split':[2,5,10,14],
            'min_samples_leaf':[2,5,10,14],
        }
    },
    'Decision Tree':{
        'model':DecisionTreeRegressor(),
        'params':{
            'criterion':["mse", "mae"],
            'max_depth':[int(x) for x in np.linspace(1,30,5)],
            'max_features':['auto','sqrt','log2'],
            'ccp_alpha':[x for x in np.linspace(0.0025,0.0125,5)],
            'min_samples_split':[2,5,10,14],
            'min_samples_leaf':[2,5,10,14],
        }
    }
}
```

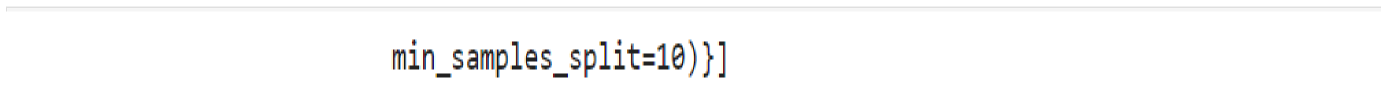
- For models that have hyperparameters (such as Ridge Regression, Lasso Regression, Decision Tree, and Random Forest), hyperparameter tuning techniques like grid search or randomized search may be employed to find the optimal hyperparameters that maximize model performance.

5. *Model Evaluation*:

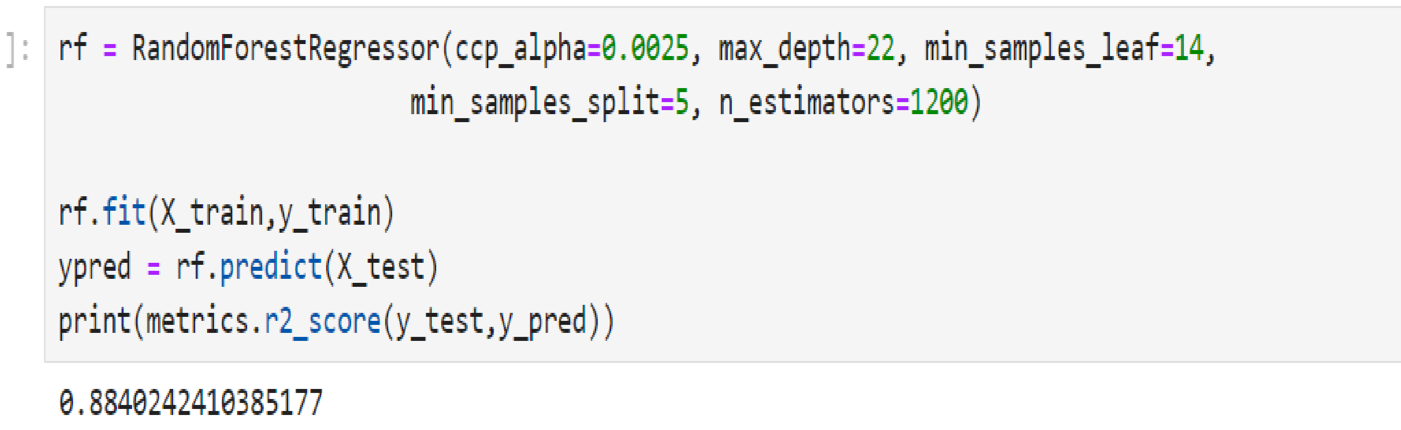


```python
scores_df = pd.DataFrame(scores,columns=['model_name','best_score','best_estimator'])
scores_df
```

	model_name	best_score	best_estimator
0	RandomForest	-0.097905	(DecisionTreeRegressor(ccp_alpha=0.005, max_de...
1	Decision Tree	-0.094662	DecisionTreeRegressor(ccp_alpha=0.005, criteri...



```
                        min_samples_split=10)}]
```



```python
rf = RandomForestRegressor(ccp_alpha=0.0025, max_depth=22, min_samples_leaf=14,
                           min_samples_split=5, n_estimators=1200)

rf.fit(X_train,y_train)
ypred = rf.predict(X_test)
print(metrics.r2_score(y_test,y_pred))
```

```
0.8840242410385177
```

- Once the models are trained and tuned, they are evaluated on the testing dataset using appropriate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). This step helps assess the predictive performance of each model and compare their effectiveness in predicting laptop prices.

6. *Model Selection*:

- Based on the evaluation results, the best-performing model or model is **Random forest regressor** and which is selected for deployment. This decision is based on factors such as predictive accuracy, computational efficiency, and interpretability.