Hands-on practice lab: Model Development

Estimated time needed: **45** minutes

In this lab, you will use the skills acquired in throughout the module, and use linear regression principles to create a model that predicts the Price of the laptop, based on one or more attributes of the dataset.

Objectives

After completing this lab you will be able to:

- Use Linear Regression in one variable to fit the parameters to a model
- Use Linear Regression in multiple variables to fit the parameters to a model
- Use Polynomial Regression in single variable tofit the parameters to a model
- Create a pipeline for performing linear regression using multiple features in polynomial scaling
- Evaluate the performance of different forms of regression on basis of MSE and R^2 parameters

Setup

For this lab, we will be using the following libraries:

- `pandas` for managing the data.
- `numpy` for mathematical operations.
- `sklearn` for machine learning and machine-learning-pipeline related functions.
- `seaborn` for visualizing the data.
- `matplotlib` for additional plotting tools.

The following required libraries are **not** pre-installed in the Skills Network Labs environment. **You will need to run the following cell** to install them:

In [1]:
```
import piplite
await piplite.install('seaborn')
```

Importing Required Libraries

We recommend you import all required libraries in one place (here):

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
%matplotlib inline
```

```
<ipython-input-2-c5c9fedd4c6a>:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

Importing the dataset

Run the cell below to download the dataset into this environment.

This function will download the dataset into your browser

In [3]:
```
#This function will download the dataset into your browser

from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

We put the file path along with a quotation mark so that pandas will read the file into a dataframe from that address. The file path can be either an URL or your local file address.

In [4]:
```
path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/laptop_pricing_dataset_mod2.csv"
```

You will need to download the dataset using the download() function:

In [5]:
```
#you will need to download the dataset;
await download(path, "laptops.csv")
file_name="laptops.csv"
```

Load the dataset into a pandas dataframe

In [6]:
```
df = pd.read_csv(file_name, header=0)
```

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface.While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply skip the steps above, and simply use the URL directly in the pandas.read_csv() function. You can uncomment and run the statements in the cell below.

In [ ]:
```
#https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/laptop_pricing_dataset_mod2.csv"
#df = pd.read_csv(filepath, header=None)
```

In [ ]:
```
# show the first 5 rows using dataframe.head() method
print("The first 5 rows of the dataframe")
df.head(5)
```

Task 1 : Single Linear Regression

You have learnt that "CPU_frequency" is the parameter with the lowest p-value among the different features of the dataset. Create a single feature Linear Regression model that fits the pair of "CPU_frequency" and "Price" to find the model for prediction.

In [7]:
```
# Write your code below and press Shift+Enter to execute
lm = LinearRegression()
lm.fit(df[["CPU_frequency"]],df["Price"])
yhat = lm.predict(df[["CPU_frequency"]])
```

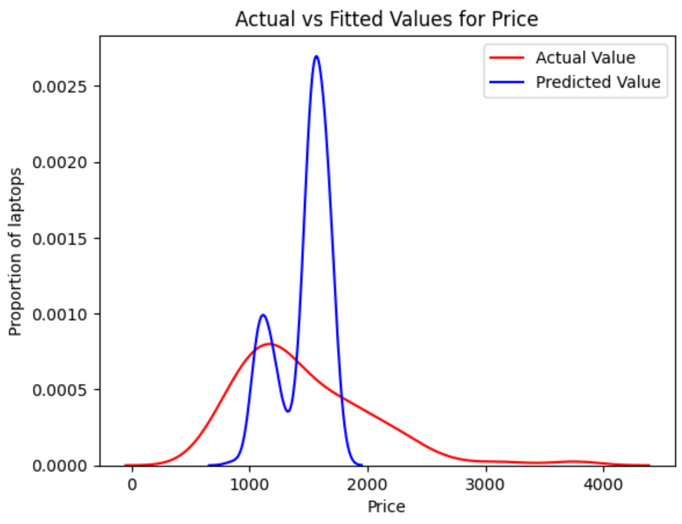
▸ Click here for Solution

Generate the Distribution plot for the predicted values and that of the actual values. How well did the model perform?

In [9]:
```
# Write your code below and press Shift+Enter to execute
ax1 = sns.distplot(df['Price'], hist=False, color="r", label="Actual Value")

# Create a distribution plot for predicted values
sns.distplot(yhat, hist=False, color="b", label="Fitted Values" , ax=ax1)

plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price')
plt.ylabel('Proportion of laptops')
plt.legend(['Actual Value', 'Predicted Value'])
plt.show()
```



▸ Click here for Solution

Evaluate the Mean Squared Error and R^2 score values for the model.

In [12]:
```
# Write your code below and press Shift+Enter to execute
print("Mean Squared Error : ", mean_squared_error(df['Price'],yhat))
print("R^2 Error : ", lm.score(df[["CPU_frequency"]],df["Price"]))
```

```
Mean Squared Error :  284583.44058686297
R^2 Error :  0.13444363210243238
```

▸ Click here for Solution

Task 2 - Multiple Linear Regression

The parameters which have a low enough p-value so as to indicate strong relationship with the 'Price' value are 'CPU_frequency', 'RAM_GB', 'Storage_GB_SSD', 'CPU_core', 'OS', 'GPU' and 'Category'. Use all these variables to create a Multiple Linear Regression system.

```
In [14]: # Write your code below and press Shift+Enter to execute
Z = df[['CPU_frequency', 'RAM_GB', 'Storage_GB_SSD', 'CPU_core', 'OS', 'GPU', 'Category']]
lm1 = LinearRegression()
lm1.fit(Z,df['Price'])
yhat1 = lm1.predict(Z)
```

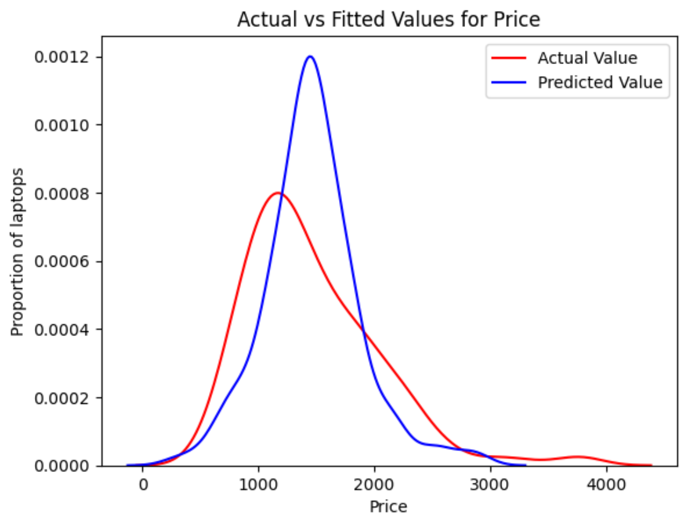
▸ Click here for Solution

Plot the Distribution graph of the predicted values as well as the Actual values

```
In [15]: # Write your code below and press Shift+Enter to execute
ax1 = sns.distplot(df['Price'], hist=False, color="r", label="Actual Value")

# Create a distribution plot for predicted values
sns.distplot(yhat1, hist=False, color="b", label="Fitted Values" , ax=ax1)

plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price')
plt.ylabel('Proportion of laptops')
plt.legend(['Actual Value', 'Predicted Value'])
plt.show()
```



▸ Click here for Solution

Find the R^2 score and the MSE value for this fit. Is this better or worst than the performance of Single Linear Regression?

```
In [16]: # Write your code below and press Shift+Enter to execute
print("Mean Squared Error : ", mean_squared_error(df['Price'],yhat1))
print("R^2 Error : ", lm.score(df[["CPU_frequency"]],df["Price"]))

Mean Squared Error :  161680.57263893107
R^2 Error :  0.13444363210243238
```

▸ Click here for Solution

Task 3 - Polynomial Regression

Use the variable "CPU_frequency" to create Polynomial features. Try this for 3 different values of polynomial degrees. Remember that polynomial fits are done using `numpy.polyfit` .

```
In [18]: #  Write your code below and press Shift+Enter to execute
f1 = np.polyfit(df["CPU_frequency"].to_numpy().flatten(),df["Price"],1)
p1 = np.poly1d(f1)

f3 = np.polyfit(df["CPU_frequency"].to_numpy().flatten(),df["Price"],3)
p3 = np.poly1d(f3)

f5 = np.polyfit(df["CPU_frequency"].to_numpy().flatten(),df["Price"],5)
p5 = np.poly1d(f5)
```

▸ Click here for Solution

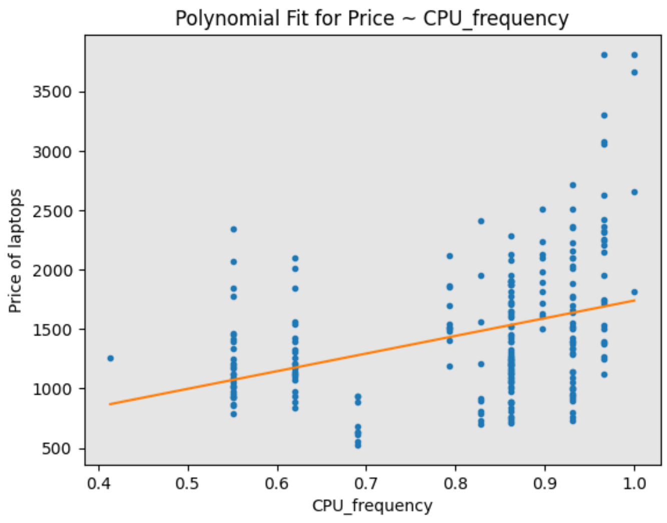
Plot the regression output against the actual data points to note how the data fits in each case. To plot the polynomial response over the actual data points, you have the function shown below.

```
In [19]: def PlotPolly(model, independent_variable, dependent_variabble, Name):
             x_new = np.linspace(independent_variable.min(),independent_variable.max(),100)
             y_new = model(x_new)

             plt.plot(independent_variable, dependent_variabble, '.', x_new, y_new, '-')
             plt.title(f'Polynomial Fit for Price ~ {Name}')
             ax = plt.gca()
             ax.set_facecolor((0.898, 0.898, 0.898))
             fig = plt.gcf()
             plt.xlabel(Name)
             plt.ylabel('Price of laptops')
```

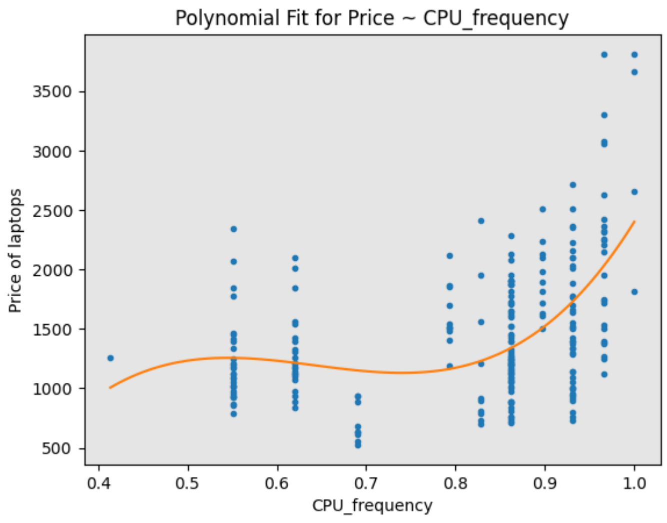
Call this function for the 3 models created and get the required graphs.

```
In [20]: #  Write your code below and press Shift+Enter to execute
         # Call for function of degree 1
         PlotPolly(p1,df["CPU_frequency"],df["Price"],"CPU_frequency")
```



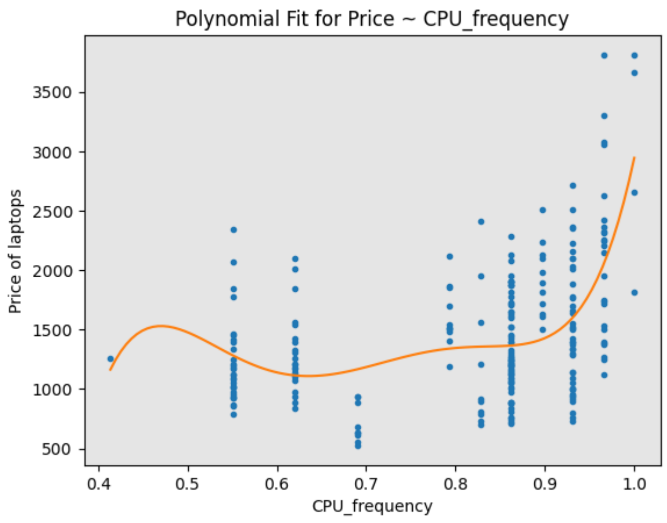
▸ Click here for Solution

```
In [21]: #  Write your code below and press Shift+Enter to execute
         # Call for function of degree 3
         PlotPolly(p3,df["CPU_frequency"],df["Price"],"CPU_frequency")
```



▸ Click here for Solution

```
In [22]: #  Write your code below and press Shift+Enter to execute
         # Call for function of degree 5
         PlotPolly(p5,df["CPU_frequency"],df["Price"],"CPU_frequency")
```



▸ Click here for Solution

Also, calculate the R^2 and MSE values for these fits. For polynomial functions, the function sklearn.metrics.r2_score will be used to calculate R^2 values.

In [23]:
```
#  Write your code below and press Shift+Enter to execute
print("MSE of 1st degree polynomial : ",mean_squared_error(df["Price"],p1(df["CPU_frequency"])))
print("R2 Score of 1st degree polynomial : ",r2_score(df["Price"],p1(df["CPU_frequency"])))

print("MSE of 3rd degree polynomial : ",mean_squared_error(df["Price"],p3(df["CPU_frequency"])))
print("R2 Score of 3rd degree polynomial : ",r2_score(df["Price"],p3(df["CPU_frequency"])))

print("MSE of 5th degree polynomial : ",mean_squared_error(df["Price"],p5(df["CPU_frequency"])))
print("R2 Score of 5th degree polynomial : ",r2_score(df["Price"],p5(df["CPU_frequency"])))
```

```
MSE of 1st degree polynomial :  284583.4405868628
R2 Score of 1st degree polynomial :  0.13444363210243282
MSE of 3rd degree polynomial :  241024.8630384881
R2 Score of 3rd degree polynomial :  0.26692640796530986
MSE of 5th degree polynomial :  229137.29548053825
R2 Score of 5th degree polynomial :  0.3030822706443803
```

▸ Click here for Solution

Task 4 - Pipeline

Create a pipeline that performs parameter scaling, Polynomial Feature generation and Linear regression. Use the set of multiple features as before to create this pipeline.

In [25]:
```
#  Write your code below and press Shift+Enter to execute
Z = df[["CPU_frequency","GPU","OS","Screen_Size_inch"]]
Input = [('scale',StandardScaler()),('ploynomial',PolynomialFeatures()),('model',LinearRegression())]
pipe = Pipeline(Input)
pipe.fit(Z,df["Price"])
ypipe = pipe.predict(Z)
```

▸ Click here for Solution

Evaluate the MSE and R^2 values for the this predicted output.

In [26]:
```
#  Write your code below and press Shift+Enter to execute
print("MSE Score : ",mean_squared_error(df["Price"],ypipe))
print("R2 Score : ",r2_score(df["Price"],ypipe))
```

```
MSE Score :  199686.19327731093
R2 Score :  0.39265736679557306
```

▸ Click here for Solution

You should now have seen that the values of R^2 increase as we go from Single Linear Regression to Multiple Linear Regression. Further, if we go for multiple linear regression extended with polynomial features, we get an even better R^2 value.

Congratulations! You have completed the lab

Authors

Abhishek Gagneja

Vicky Kuo