Statement of Work

Jack Lemere

Project: Diamond Value Prediction

AIDI 1002 AI Algorithms

Course Facilitator: Marcos Bittencourt

Date: Dec 18, 2020

Executive Summary and Rationale

Diamonds are well known to be one of the most expensive and sought-after gemstones in the world, making the diamond business extremely lucrative. Diamonds are a form of elemental carbon created deep in the earth’s crust under extreme temperature and pressure. Once mined, the diamonds are carefully examined and then cut into various unique shapes. Once the diamond has been cut, it is then examined and graded in order to determine the diamond’s price value. The value of each gemstone is based on the variables used in the grading process. These variables primarily include carat, clarity, color, and cut; also known as the four Cs. For this project, we will be creating an AI algorithm to predict the price values of various diamonds based on these variables, as well as other variables. Having an AI be able to assess the value of diamonds is very useful as this can be a very mundane task for humans. By being able to recognize patterns in characteristics that give diamonds their value through AI, human resources may be spared, making the process much faster, more efficient, and possibly more accurate.

The dataset used will contain a list of several diamonds each have variables including carat, clarity, color, cut, and more. Each datapoint will also include the evaluated price of the diamond determined manually with the given variables. The algorithm will be trained using a sample of this data, which will begin to correlate a relationship between the variables and the given price. The algorithm will then be tested to predict the value of numerous other diamonds. The predicted price and the actual determined price will then be compared to determine the accuracy of the model.

Data and Algorithm

The training and testing of the prediction model will require a dataset acquired from Kaggle. The dataset is very large, so many samples will be used to train and test the model. The main variables of the dataset to be examined will be the four Cs mentioned earlier since those variables will have the most impact on the value of each diamond. There are some other values which may have an influence on the pricing of diamonds such as the various geometric characteristics included in the data. This relationship between price and geometric will be explored to determine if there is any noticeable influence. There are some variables not in the dataset that may likely have an influence on the value of the diamond. These variables include non-numerical values such as the history or uniqueness of each diamond, or the location of where each diamond was mined. Not having access to these variables may hinder the accuracy of the prediction model. However, these inaccessible values may be reflected as outliers in the dataset which can be removed to account for special cases. As for the algorithm itself, the goal is to predict price values based on known information. The best choice of AI architecture in this case would be a regression based algorithm.

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...	...

Fig 1. The first four data points of the diamond dataset. The main characteristics which will determine price will be the columns named carat, cut, color, and clarity. Carat is a quantitative value representing the mass of the diamond in carats, while cut, color, and clarity are all categorical values. “Cut” describes the cut quality of the diamond and is given a value of either Fair, Good, Very Good, Premium, or Ideal. “Color” is given a letter between D and J with D being best and J being worst. “Clarity” gives each diamond a value from the *GIA diamond clarity grading scale* which gives each diamond a rating between I_3 (Included) and FL (Flawless) with various values in between.

EDA and Data Cleaning

The data we are using has both numerical and categorical data. We will check to ensure the data is clean then analyze the clean data.

	carat	depth	table	price	x	y	z
count	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000	53920.000000
mean	0.797698	61.749514	57.456834	3930.993231	5.731627	5.734887	3.540046
std	0.473795	1.432331	2.234064	3987.280446	1.119423	1.140126	0.702530
min	0.200000	43.000000	43.000000	326.000000	3.730000	3.680000	1.070000
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5323.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

```
df.cut.unique()
```

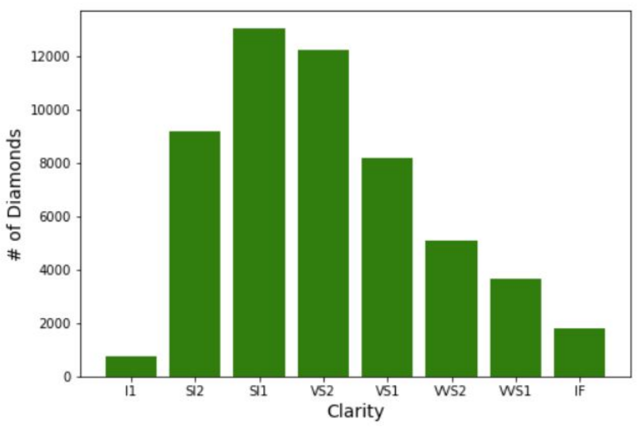
```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

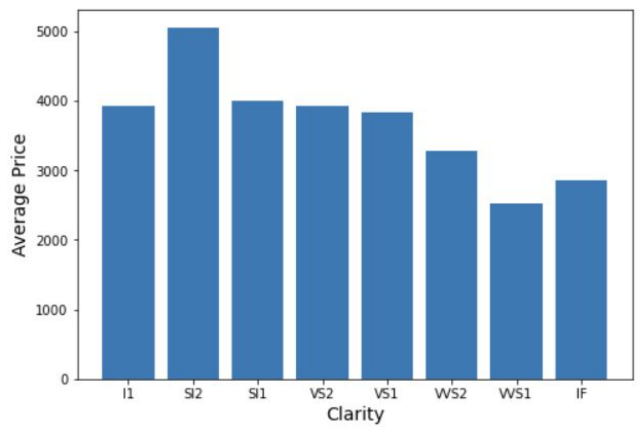
```
df.color.unique()
```

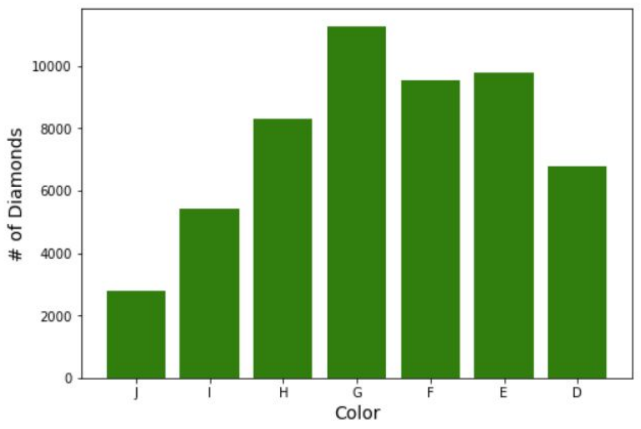
```
array(['E', 'I', 'J', 'H', 'F', 'G', 'D'], dtype=object)
```

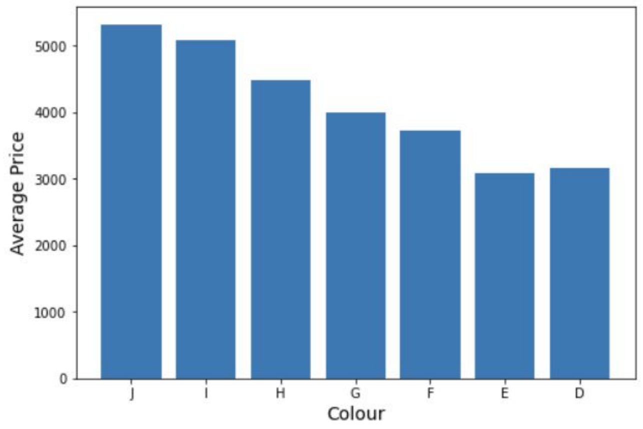
```
df.clarity.unique()
```

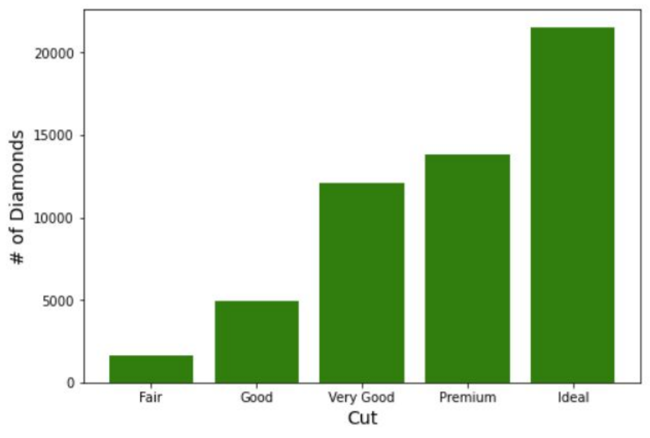
```
array(['SI2', 'SI1', 'VS1', 'VS2', 'VVS2', 'VVS1', 'I1', 'IF'],
      dtype=object)
```

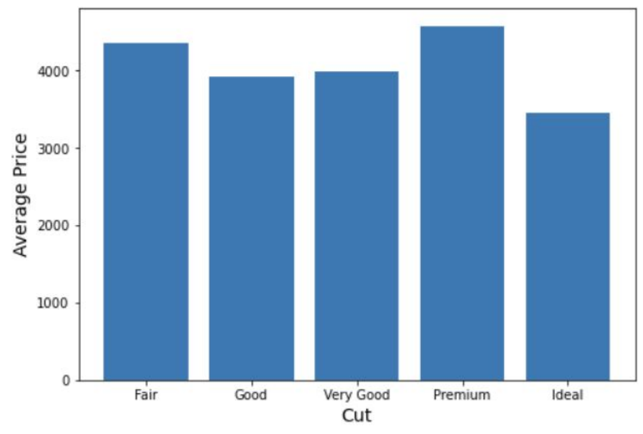






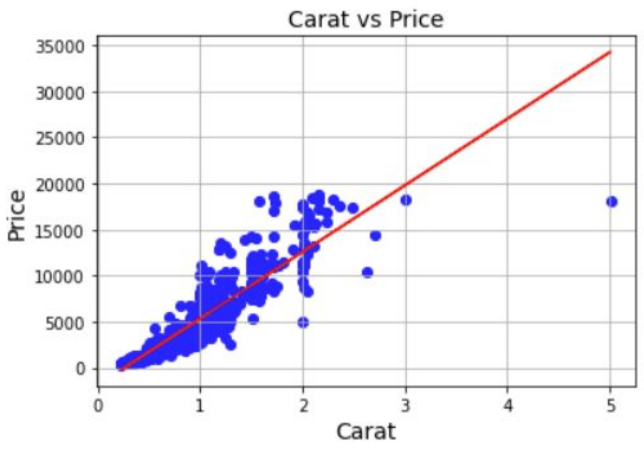


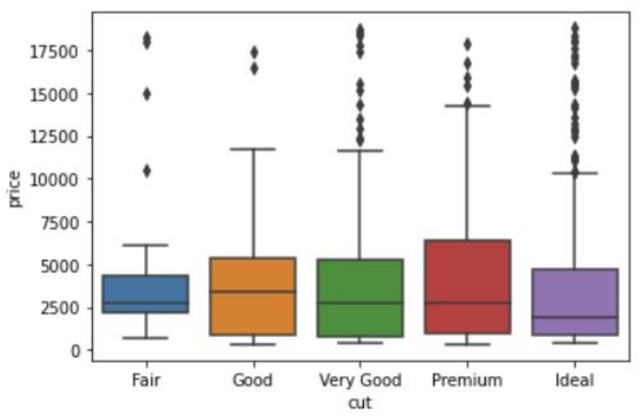


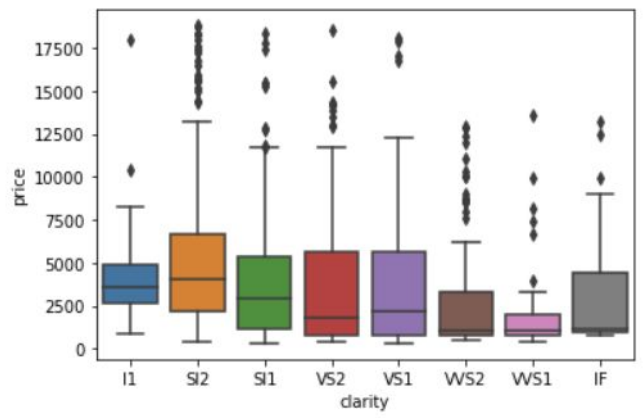


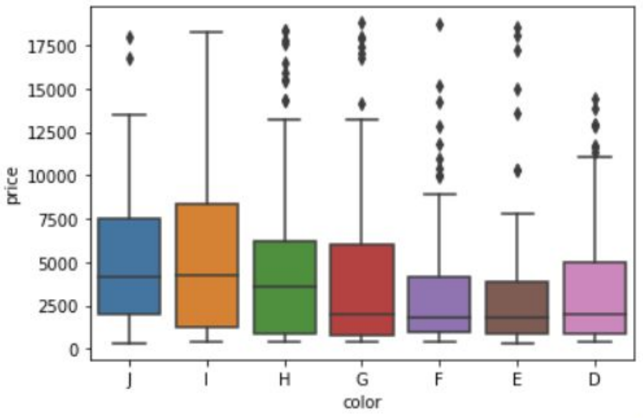
Data Features

As stated previously, the main features we will be examining to determine the value of the diamond are ‘carat’, ‘cut’, ‘clarity’, and ‘color’. We will perform a statistical analysis on these features to determine how heavily they influence the price of the diamond.









Based on the analysis of the data, it appears that carat has the most substantial impact on the price of the diamond. Cut also appears to have the second largest impact on price, whilst clarity and color appear to have seem to not have much impact in comparison. This makes sense as carat and cut are much more noticeable to the naked eye compared to clarity and color.

Test Process

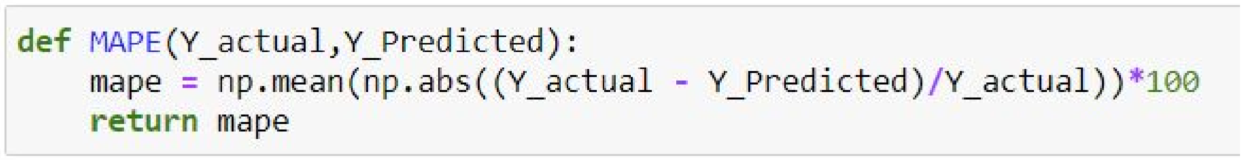
The final testing process will be where the model is evaluated to determine how accurately the prices of the diamonds are predicted. This will be measured by determining the degree of error the model produces. The one method that will be examined for determining the magnitude of error will be to measure the mean-absolute error. This will take the average of the differences between each prediction and actual value, giving us a metric to determine what range we can expect the predicted value to differ from the actual value.

$$\frac{1}{n}\sum_{n}^{t=1}|A_t - F_t|$$

An issue with using mean-absolute error as a way to measure inaccuracy is that calculating the difference may not provide an accurate understanding of how accurate the model is since we are potentially dealing with a vast range of numbers. For example, if a price difference between the predicted price and actual price of a diamond was $5,000, that does not really tell us much. If the diamond was worth $400 and the model predicted $5,400, that would suggest the model made a very poor prediction. Though if the diamond was worth $1,200,000 and the model predicted $1,205,000, that would suggest the model made a very good prediction. Since we only see the final difference when using this evaluation method, determining if the prediction was good or not is not possible. To avoid this, if we are dealing with a wide range of numbers it would be best to use a percentage based value. Such as Mean Absolute Percentage Error (MAPE).

$$\left\{\frac{1}{n}\sum\frac{|A_t - F_t|}{|A_t|}\right\} * 100$$

By using this method, more background information is seen which allows us to more easily compare how accurately the model predicted. Applying this method on the earlier example, the poor prediction would have an error of 1250% while the good prediction would have an error of 0.42%. By using this example, we can see that MAPE provides much more context to the error, which will be better for our purposes.



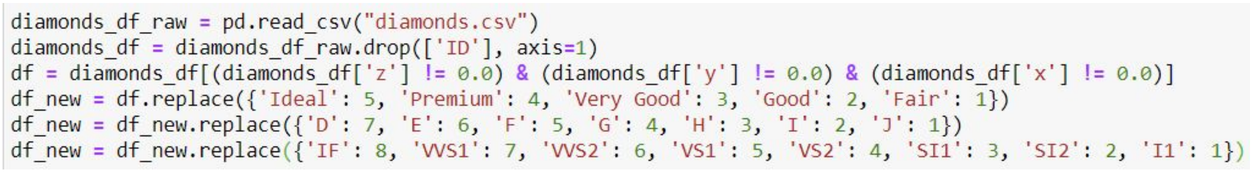
```
def MAPE(Y_actual,Y_Predicted):
    mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return mape
```

The function we will use to calculate MAPE.

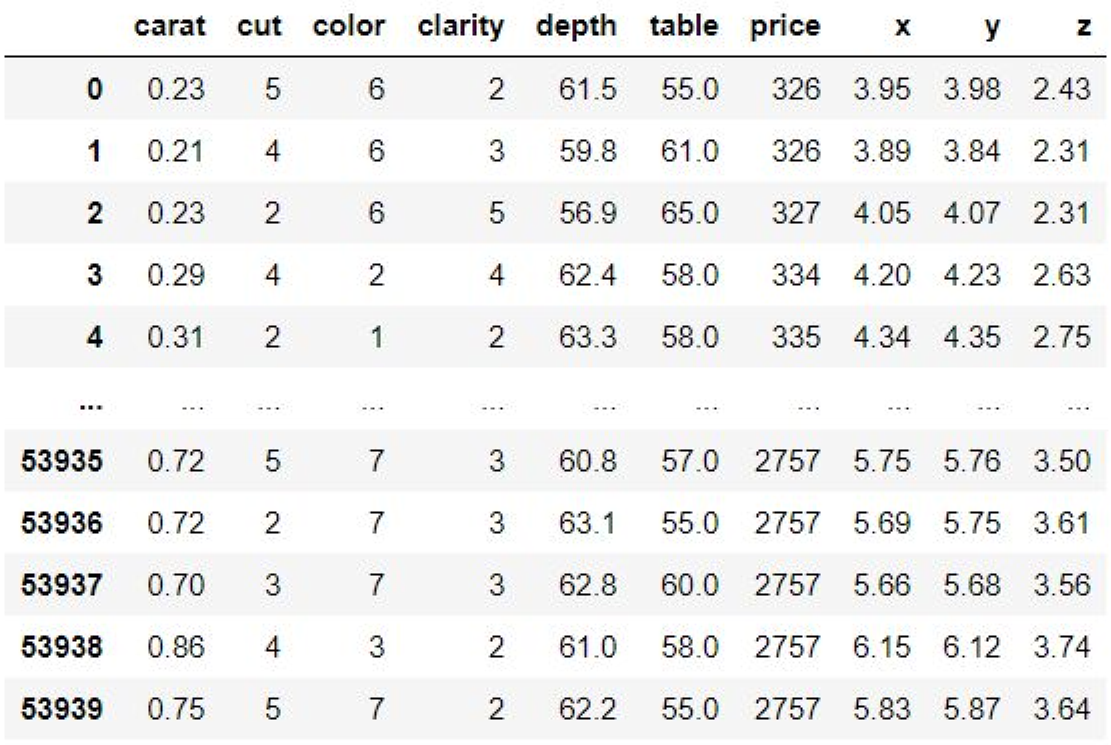
Model

As we are trying to predict the value of something dependent on variables, and that the only non-categorical data point has a linear trend with price, it is clear that a linear regression algorithm should be an obvious choice of model for this problem. As we have multiple variables influencing price, we will be using multiple regression; an extension of linear regression that utilizes multiple dependent variables.

The first step of the model will be to convert the categorical data into a usable form. Rather than represent each grade of cut, colour, and clarity with a string variable, we will convert this to numerical. This will be done rather simply by replacing the ranking ranking system into numerical ranking. For cut, the ranking will be 1-5 with 1 being worst and 5 being best, 1-7 with 1 being worst and 7 being best for colour, and 1-8 with 1 being worst and 8 being best for clarity.

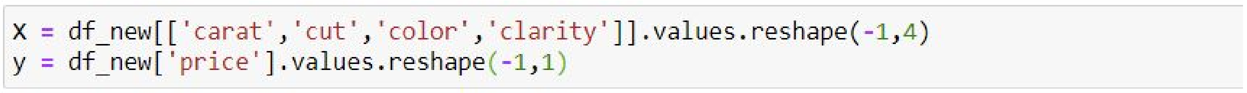


```
diamonds_df_raw = pd.read_csv("diamonds.csv")
diamonds_df = diamonds_df_raw.drop(['ID'], axis=1)
df = diamonds_df[(diamonds_df['z'] != 0.0) & (diamonds_df['y'] != 0.0) & (diamonds_df['x'] != 0.0)]
df_new = df.replace({'Ideal': 5, 'Premium': 4, 'Very Good': 3, 'Good': 2, 'Fair': 1})
df_new = df_new.replace({'D': 7, 'E': 6, 'F': 5, 'G': 4, 'H': 3, 'I': 2, 'J': 1})
df_new = df_new.replace({'IF': 8, 'VVS1': 7, 'VVS2': 6, 'VS1': 5, 'VS2': 4, 'SI1': 3, 'SI2': 2, 'I1': 1})
```

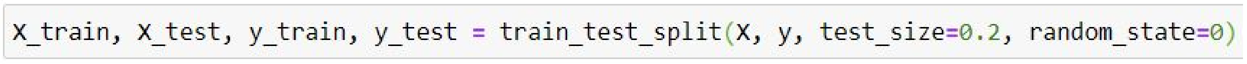


	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	5	6	2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	4	6	3	59.8	61.0	326	3.89	3.84	2.31
2	0.23	2	6	5	56.9	65.0	327	4.05	4.07	2.31
3	0.29	4	2	4	62.4	58.0	334	4.20	4.23	2.63
4	0.31	2	1	2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	5	7	3	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	2	7	3	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	3	7	3	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	4	3	2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	5	7	2	62.2	55.0	2757	5.83	5.87	3.64

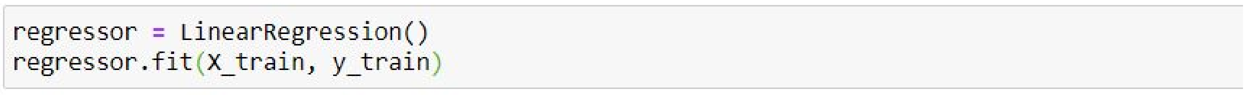
Now to implement the model, we will set our dependent variables to the four main columns that determine price (‘carat’, ‘cut’, ‘color’, and ‘clarity’). The independent variable will be price. With this, we will split the dataset into train and test sets, and train the model.



```
X = df_new[['carat','cut','color','clarity']].values.reshape(-1,4)
y = df_new['price'].values.reshape(-1,1)
```

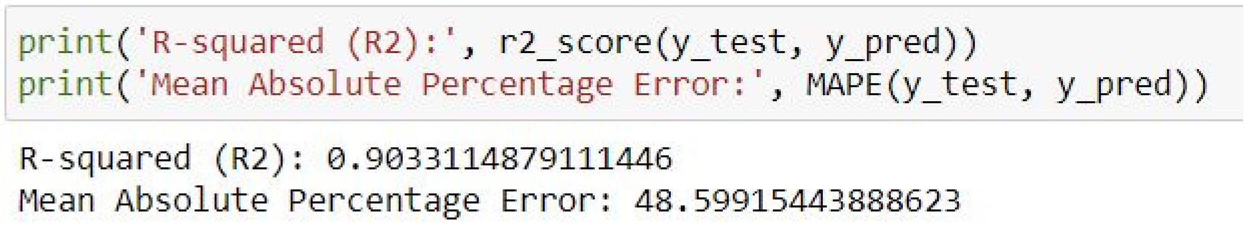


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```



```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

With our model trained, we can determine the accuracy by examining the R2 value along with the MAPE as discussed earlier.



```
print('R-squared (R2):', r2_score(y_test, y_pred))
print('Mean Absolute Percentage Error:', MAPE(y_test, y_pred))
```

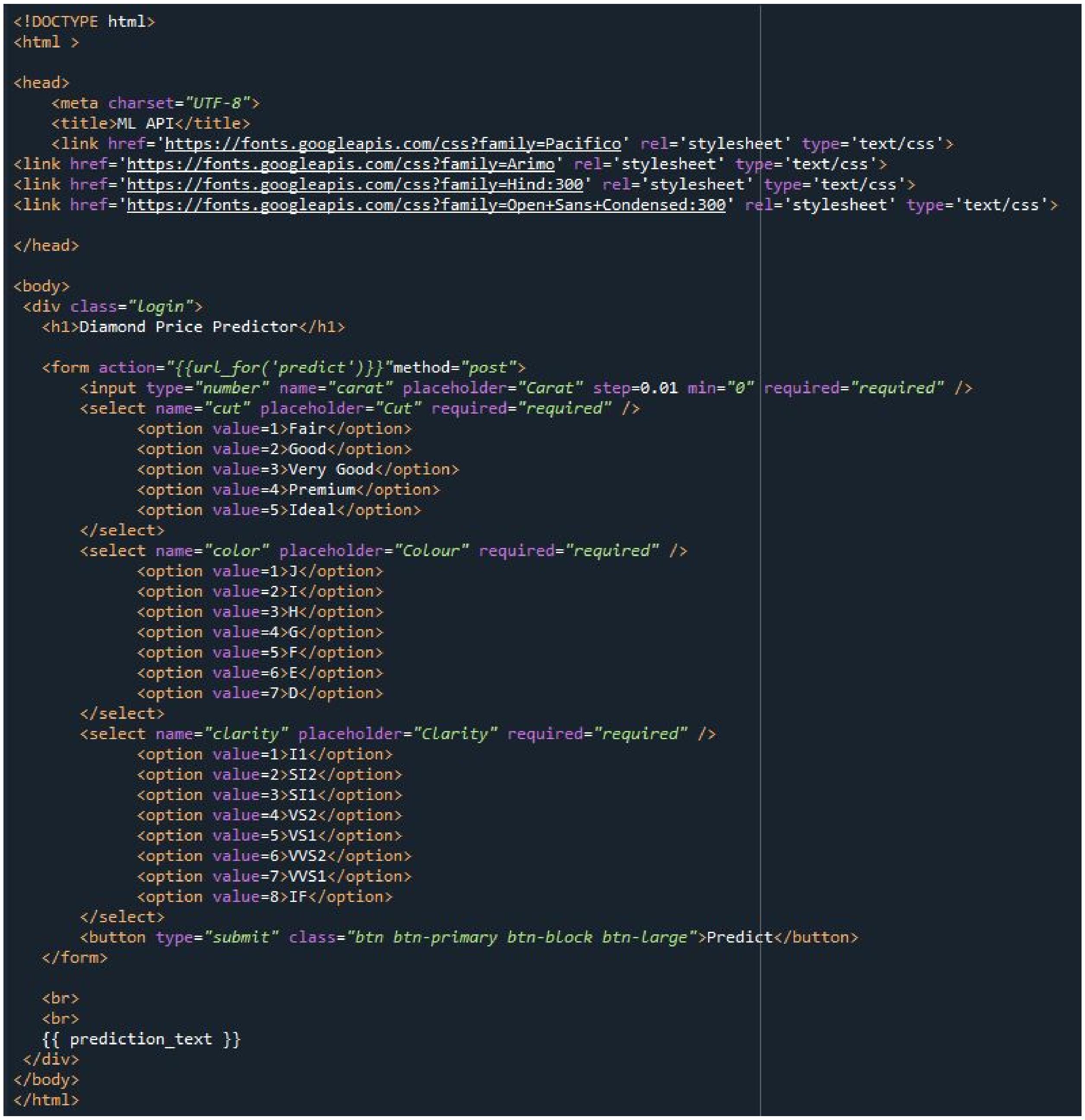
```
R-squared (R2): 0.9033114879111446
Mean Absolute Percentage Error: 48.59915443888623
```

Here we can see that the model has a good R2 value of 0.90 meaning there is a strong linear correlation, and we have a MAPE of 48% (meaning that on average, the values are within 48% of the actual price) which isn't ideal but also not terrible.

Interface

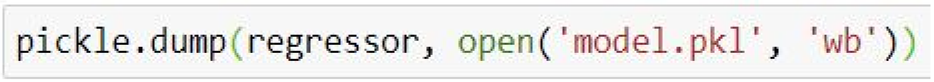
With our model now ready, we can develop the software pipeline to link with a user-interface for professional use.

First we will develop the app, this app first requires a format for the application to run in. A simple interface is developed using html code to create an interface with one numerical input (carat) and three categorical inputs (cut, colour, and clarity) and an output line that will print the predicted price of the diamond with the inputted parameters.

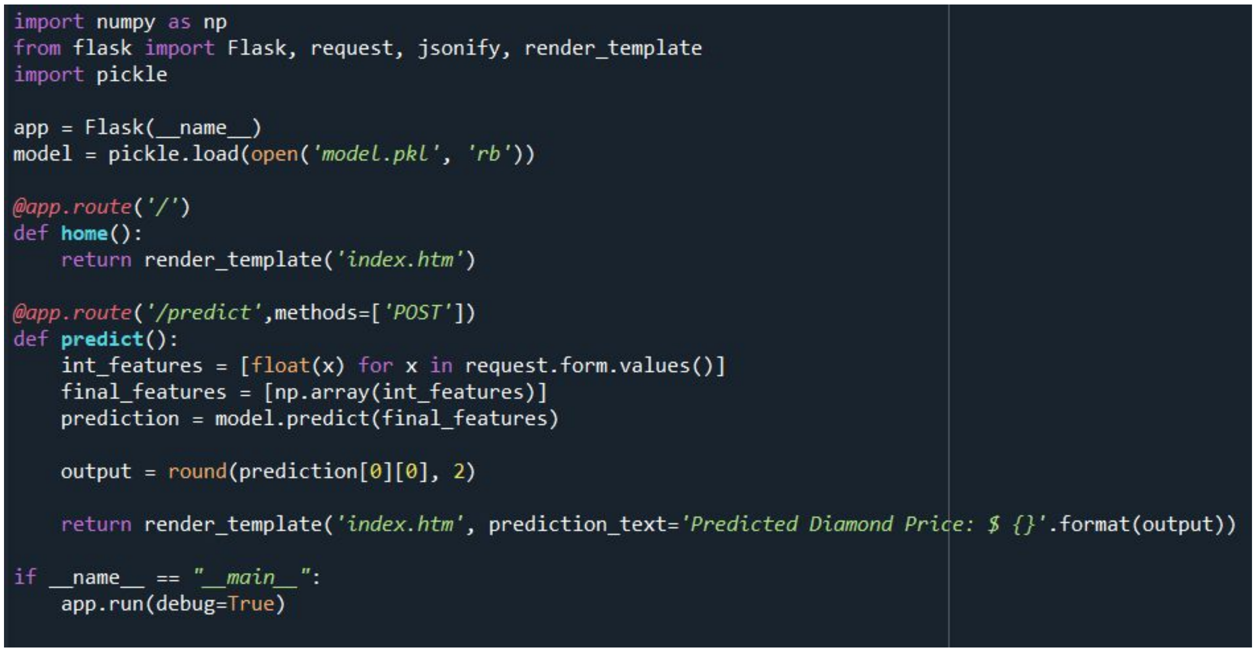


```html
<!DOCTYPE html>
<html >

<head>
    <meta charset="UTF-8">
    <title>ML API</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

</head>

<body>
 <div class="login">
   <h1>Diamond Price Predictor</h1>

   <form action="{{url_for('predict')}}"method="post">
       <input type="number" name="carat" placeholder="Carat" step=0.01 min="0" required="required" />
       <select name="cut" placeholder="Cut" required="required" />
             <option value=1>Fair</option>
             <option value=2>Good</option>
             <option value=3>Very Good</option>
             <option value=4>Premium</option>
             <option value=5>Ideal</option>
       </select>
       <select name="color" placeholder="Colour" required="required" />
             <option value=1>J</option>
             <option value=2>I</option>
             <option value=3>H</option>
             <option value=4>G</option>
             <option value=5>F</option>
             <option value=6>E</option>
             <option value=7>D</option>
       </select>
       <select name="clarity" placeholder="Clarity" required="required" />
             <option value=1>I1</option>
             <option value=2>SI2</option>
             <option value=3>SI1</option>
             <option value=4>VS2</option>
             <option value=5>VS1</option>
             <option value=6>VVS2</option>
             <option value=7>VVS1</option>
             <option value=8>IF</option>
       </select>
       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
   </form>

   <br>
   <br>
   {{ prediction_text }}
 </div>
</body>
</html>
```

Next we must link the interface with the model. We first save our model to a pkl file called model.pkl, which we then use in the application. A function ‘predict’ is created, which takes the values inputted into the four boxes made earlier, converts those numbers into a numpy array, and sends that array to the model. When the model makes a prediction, the value is returned and printed in the webpage.

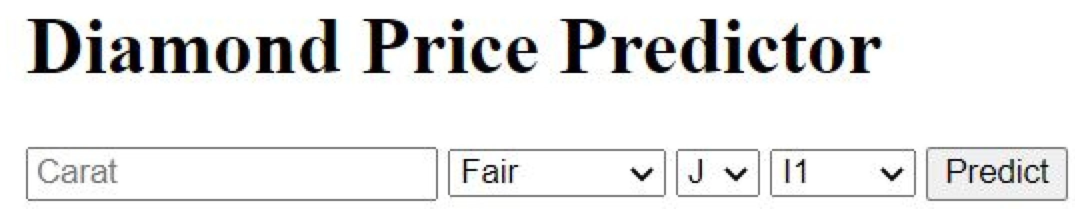


```python
pickle.dump(regressor, open('model.pkl', 'wb'))
```



```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.htm')

@app.route('/predict',methods=['POST'])
def predict():
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0][0], 2)

    return render_template('index.htm', prediction_text='Predicted Diamond Price: $ {}'.format(output))

if __name__ == "__main__":
    app.run(debug=True)
```

In order to make the application publicly usable, these files are all uploaded to Github, and linked to cloud based Heroku. With all this done, the software is ready for use.



Diamond Price Predictor

You can use the software yourself here: https://diamond-price-api.herokuapp.com/

Improvements to Make

The model built here is by no means perfect and could be improved with some adjustments. The main area of concern is for low carat diamonds with low quality cut, colour, and clarity. When smaller values are used it often predicts a negative value. This is likely to do with the line of best fit for the carat variable. At low carat values (roughly < 0.5), the prediction line can be seen to dip into the negative, which means that the overall prediction is likely being influenced by this. Obviously a diamond cannot have a negative price, so this is a flaw in our model. We could further adjust it by ensuring the prediction line has a y-intercept of 0 either through linear fitting or polynomial fitting.

The model could also be improved by changing the influence of each variable. Many experts suggest that cut is more likely to influence price than colour or clarity. Currently, our model suggests that these influence price equally. We could adjust our model so that certain features have more influence on price than others, thus improving accuracy and more closely reflecting real world price.

The model could also be improved by examining other variables besides the 4 Cs mentioned earlier. In this dataset, we are also given values for the dimensions of the diamond. These include table size, depth, and dimensions in x, y, and z. Experts have also suggested that these values can influence price, with the ratio between these values being important for determining the diamond's 'brilliance' and thus it's value.

Overall, there are likely many small adjustments we can make to this model to more accurately reflect real world pricing. As it stands, the model is able to predict the price of many diamonds with relatively decent accuracy. With these added adjustments, determining the price of diamonds should be much easier, and would no longer require an expert to analyse the characteristics. This software has the potential to help many businesses in the diamond industry to project revenue and save time and money on human labour in the process.