6302 Project 2
Austin Steel

For this project I am using the dataset *Elite Sports Cars in Data* from Kaggle. This dataset contains synthetic information on 5,000 sports cars. Each entry has 27 characteristics for the vehicle.

Problem 1 Training a Classifier:

First I imported the data set and checked the type of entries to identify which are categorical and which are numerical. Out of the categorical variables, none were particularly imbalanced. Then I looked at the *Price* column and noticed most of the vehicles are very expensive. This prompted me to add a column of *Price_Category* where I split the prices into two groups, (1) price greater than $100,000 and (0) price less than $100,000. This gave me a target variable which was highly imbalnced.

```
data['Price_Category'].value_counts()
```

Price_Category	count
1	4190
0	810

I checked each categorical column to see how many unique values there were. Given there was less than 50 total unique values, I decided it would be best to use one hat encoding to include these values in my analysis. The columns *Market_Demand,* and *Popularity* were values of *Low, Medium, High*. I thought these values would best be replaced with 1, 2, and 3 respectively. There were also two columns which seemed repative; *Log_Price,* and *Log_Mileage*. While these columns could be helpful for some analysis, they don't seem appropriate here, so I removed them. Now I have a dataset that is suitable for analysis.

Without handling the imbalance, I trained a basic classifier on the dataset with the target variable of *Price_Category*. Even so, the performance of my model is very good by the standard metrics.

```
Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       158
           1       0.99      0.99      0.99       842

    accuracy                           0.99      1000
   macro avg       0.98      0.98      0.98      1000
weighted avg       0.99      0.99      0.99      1000
```

The accuracy is 0.99 which is very high. The precision for 0 (less than $100,000) i s 0.97, and for 1 (more than $100,000) 0.99. Both of these are very high and tell us the model is 97% correct at predicting 0, and 99% correct at predicting 1. The recall, or sensitivity of the model is the same at 0.97 and 0.99 respectively. And the F1-score has the same values as well. All in all, this model is very accurate at predicting the *Price_Category* by every metric.

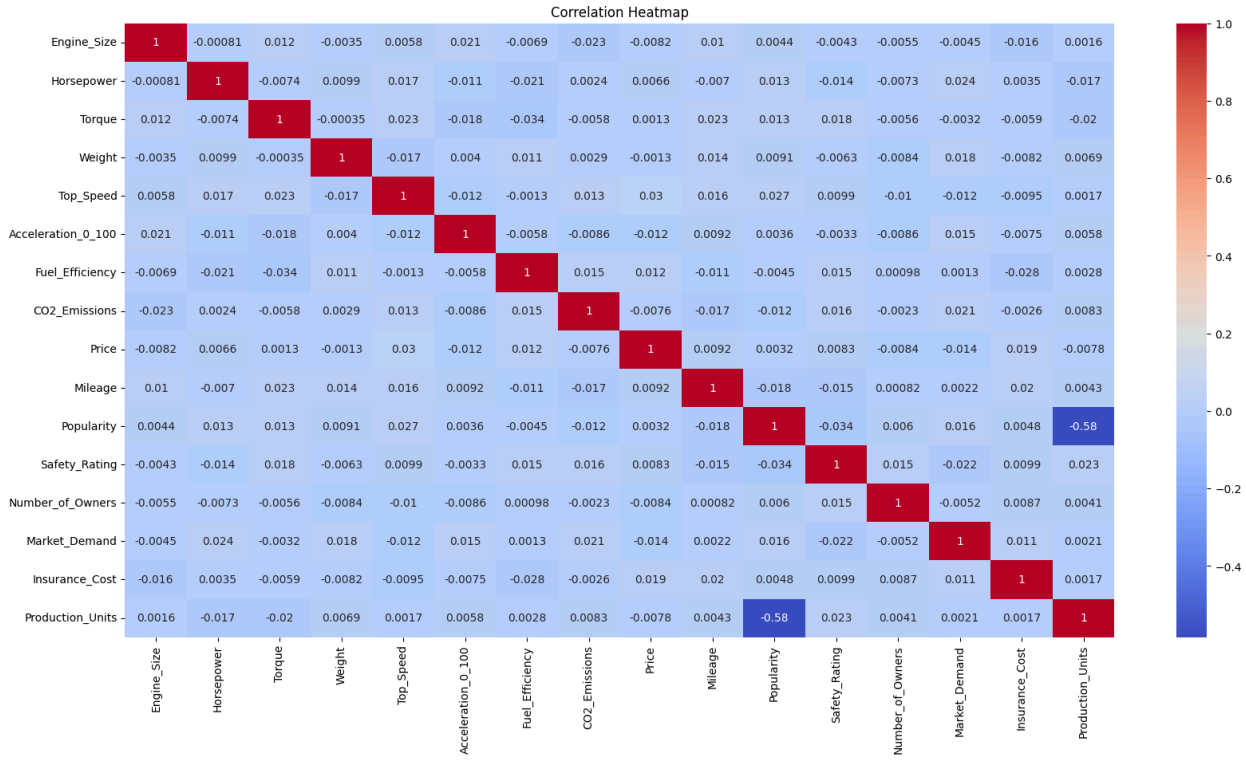
Next, I applied oversampling, and undersampling techniques to compare the models performance without handling the imbalance to handling the imbalance. First, using SMOTE, I balanced the data and then created a new model. The metrics of this model are even better.

```
Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       817
           1       0.99      0.99      0.99       859

    accuracy                           0.99      1676
   macro avg       0.99      0.99      0.99      1676
weighted avg       0.99      0.99      0.99      1676
```

Across the board here, we have 0.99. Accuracy, Precision, Recall, and F1-Score are all as high as they can be. This model tells me I have built a phenomenally good model and should be given a raise. This model is better than the imbalanced model, even though that model was very good to begin with.

Problem 2 Correlation Analysis:

For this part of the project we want to focus on the numerical categories to analysis correlations within the dataset. I imported the dataset again, replaced Low, Medium, and High with 1, 2, 3, and dropped the other categorical columns. I decided to remove these entries instead of using one hat encoding for this part of the project because this data would not be helpful in correlation analysis. I then created a correlation matrix and a heatmap to help visualize the data.



Most of the correlations are very weak, so to help identify the strongest and weakest correlations I unstacked the correlation matrix, ordered them, deleted duplicates, and printed the 3 highest and 3 lowest correlations.

```
sorted_corr_pairs.head(3)
```

		0
Price	Top_Speed	0.029831
Popularity	Top_Speed	0.026731
Horsepower	Market_Demand	0.024177

```
sorted_corr_pairs.tail(3)
```

		0
Fuel_Efficiency	Torque	-0.033617
Safety_Rating	Popularity	-0.034426
Production_Units	Popularity	-0.583622

The three highest positive correlations, while all very weak (0.02 - 0.03) would make sense. They are between the *Top_Speed* and *Price*, *Top_Speed* and *Popularity*, and *Horsepower* and *Market_Demand*. These are all correlations I would expect to see a much stronger relationship between, but still to be the strongest. This could be because the data is synthetic in nature. Out of the three strongest negative correlations, again two are very weak, but one is moderately strong at -0.58. This is significantly stronger than any other correlation (positive or negative) in the entire dataset. This correlation is between *Production_Units* and *Popularity*. *Production_Units* is a measure of how many of the car are produced. This correlation tells us the more cars that are produced the less popular they are.

Problem 3 Multiple Linear Regression Model:

For this data set, the target variable we are interested in predicting is the *Price.* To build a multiple linear regression model, I began with importing the original data again. I preprocessed the data by making sure there were no missing values, and replacing categorical values with numerical entries through one hat encoding. I then normalized the data using min-max scaling. I choose this method of standardization because I thought it best the variables remain positive for the model. After this, I was able to build the model.

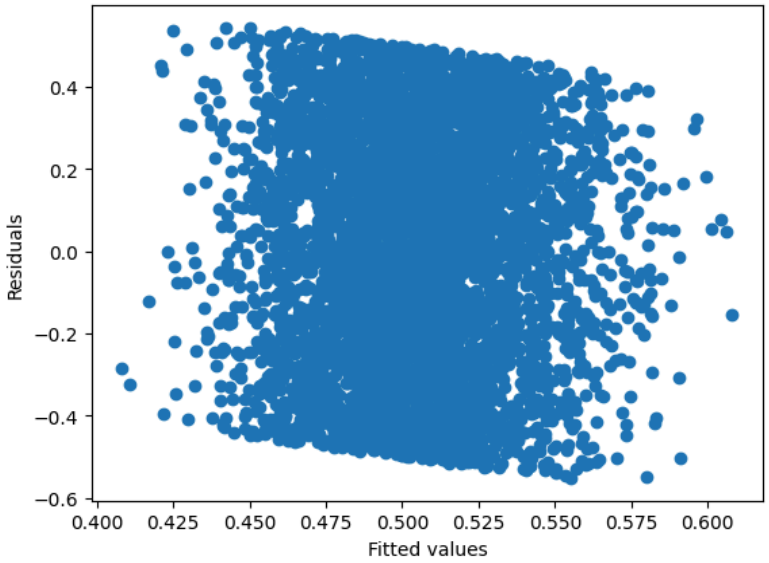
```
# regression coefficients
print('Coefficients: ', model.coef_)

# variance score
print('Variance score: {}'.format(model.score(X_test, y_test)))
```

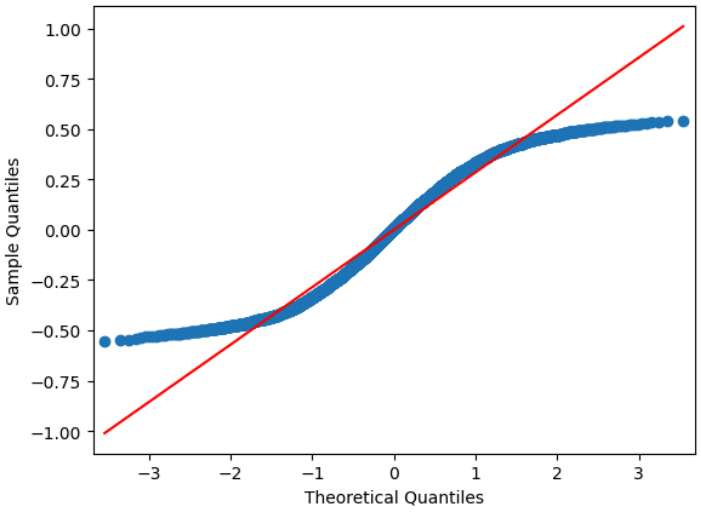
```
Coefficients:  [-1.03757909e-02 -1.50451699e-02 -3.14311837e-03  4.59453178e-03
  4.87619714e-03  3.05596689e-02 -2.06459519e-02  1.88210801e-02
  1.20306520e-02  1.00182688e-02 -3.26282423e-03  4.66237728e-03
 -6.81580769e-03 -1.22074177e-02  2.66029799e-02 -2.01808075e-03
 -8.29719555e-03  2.56872868e-02  2.46933034e-03  4.48480204e-03
 -7.19748529e-03 -1.30665198e-02 -8.79783972e-03  2.21682515e-02
 -5.58233929e-03 -1.18682910e-02 -9.75639194e-04  4.10142501e-02
 -3.15329721e-02 -8.57741485e-03  2.19931038e-02 -2.29879303e-03
 -4.27245278e-03 -1.81111980e-02  2.86049555e-03 -9.93795029e-05
 -6.64614465e-03  3.32367177e-04  6.31377747e-03 -8.26120869e-03
  5.32080272e-03 -2.90507505e-03  5.84548102e-03  5.31149029e-03
 -4.69702305e-04 -4.84178798e-03 -4.60149400e-03  3.01088402e-03
  1.59060999e-03 -9.50253546e-03 -1.28555438e-03  9.37616038e-03
  1.41192946e-03 -3.06683764e-02  3.09311913e-03  6.61135356e-05
 -2.01298577e-02 -6.10154953e-03  2.27896390e-02  3.19828305e-02
  2.96125439e-02]
Variance score: -0.016464725215062925
```

When building a multiple linear regression model, it is important to check the assumptions.

- **Linearity**: First I checked the linearity by creating scatter plots of each independent variable compared to the dependent variable (*Price)*. I did not observe any clear linear relationships from these scatterplots. (With so many variables, it is not practical to include them all here.

- **Homoscedasticity**: Next I checked the homoscedasticity by plotting the residuals versus the fitted values to ensure equal variance. The variance is randomly distributed.





- **Normality of Residuals**: Then I checked the normality of the residuals through the Q-Q plot of residuals. The residuals are fairly normal, except for the tails on both sides where they stray.

- **Multicollinearity**: Lastly, I checked for multicollinearity issues by calculating the Variance Inflation Factor. All of the variables that were one hat encoded have a VIF of infinity, which can be ignored because of the one hat encoding. The rest of the variables all have VIF scores of close to 1 and therefore indicates I do not have strong multicollinearity in this dataset.

Lastly, we want to evaluate the model to see how good it is. I will use four main metrics to do this: R-squared, Adjusted R-squared, Mean Squared Error, and Root Mean Squared Error. The R-squared value is 0.0074. This is very low, and tells us the model is not very accurate at all in

predicting the price. The Adjusted R-squared value is -0.0826. Since the Adjusted R-squared value is negative, this tells us the model explains very little variance in the Price and includes many predictors that result in a negative value. The Mean Squared Error is 0.0835 and the Root Mean Squared Error is 0.28898. Again, these two values tell us the multiple linear regression model is not very good at all in predicting the price and should not be used. One of the reasons for this could be the high number of variables, particularly the ones which were added through one hat encoding.