Predicting BMW Used Car Sale Prices

Roxanne García
August 17, 2021

1 Project Motivation

When a dealership is pricing a used vehicle, the biggest concerns are in making sure the vehicle is priced correctly and fairly. If the car is underpriced, the vehicle will sell quickly, and the dealer will not maximize the value of the sale. If it is overpriced, it can sit on the lot too long and depreciate. It is important to be able to predict a fair price. In addition, if a buyer is interested in a vehicle, but is uncertain about purchasing it, it could be helpful to provide similar vehicles as options to ensure the chances that the buyer leaves the lot purchasing from the dealership. A recommendation engine will help find vehicles that are like a target vehicle.

2 The Dataset

The dataset used in this project contains information for used BMW cars ranging from 1996 to 2020 and includes *model*, *year*, *price*, *transmission*, *mileage*, *fuel type*, *road tax*, *miles per gallon*, and *engine size* information for each vehicle. The target variable is *price* and there are eight independent variables, three of which are categorical. The dataset contains 24 different models and 25 different years, with a total of 10,781 entries.

3 Analysis Plan

The goal is to build a model that determines a reasonable price for a used BMW car and an engine that recommends related vehicles with a lower price, lower miles, and one with a slightly higher price. The following steps will be followed for this analysis:

1. Examine dataset to identify data problems, determine whether there are missing values, the datatype each feature contains, unique values in each feature, and get a general sense of the data.
2. Select a suitable algorithm based on the information gained during exploratory data analysis.
3. Fit and evaluate the models.

4 Performance Metrics

Mean Absolute Error

The mean absolute error measures the distance between the predicted and target values and averages them. It is good to use when there are a lot of outliers in the training set since it will not penalize them as much.

Mean Squared Error

The mean squared error shows how close a regression line is to a set of points by taking the average of a set of errors, which are calculated by taking the square of the distance between the values and the regression line. It can penalize large errors as a result.

Root Mean Squared Error

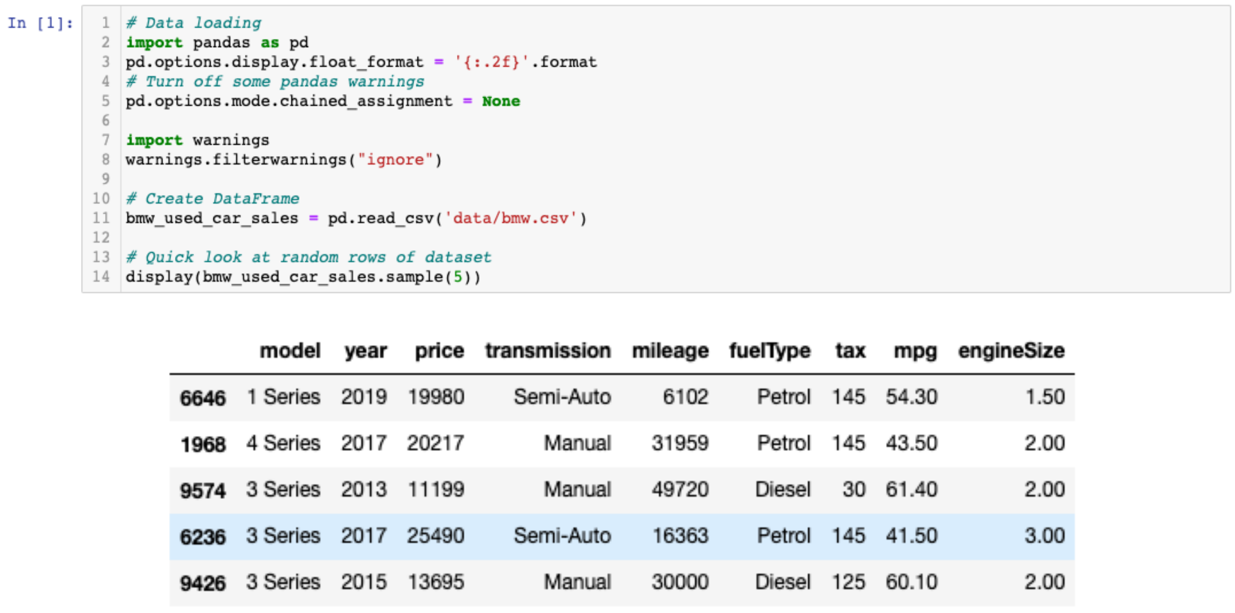
The root mean squared error is also a way to measure the distance between the predicted and target values. It is the standard deviation of the residuals, or prediction errors. It is a good measure of accuracy to compare prediction errors of different models for a particular dataset rather than between datasets since it is scale-dependent. Residuals are a measure of how far the set of points are from the regression line and RMSE is a measure of how spread out these residuals are, or how the data are concentrated around the regression line.

Mean Absolute Percentage Error

The mean absolute percentage error is the average of the absolute percentage errors (the difference between the actual value and the predicted value).

5 Exploratory Data Analysis

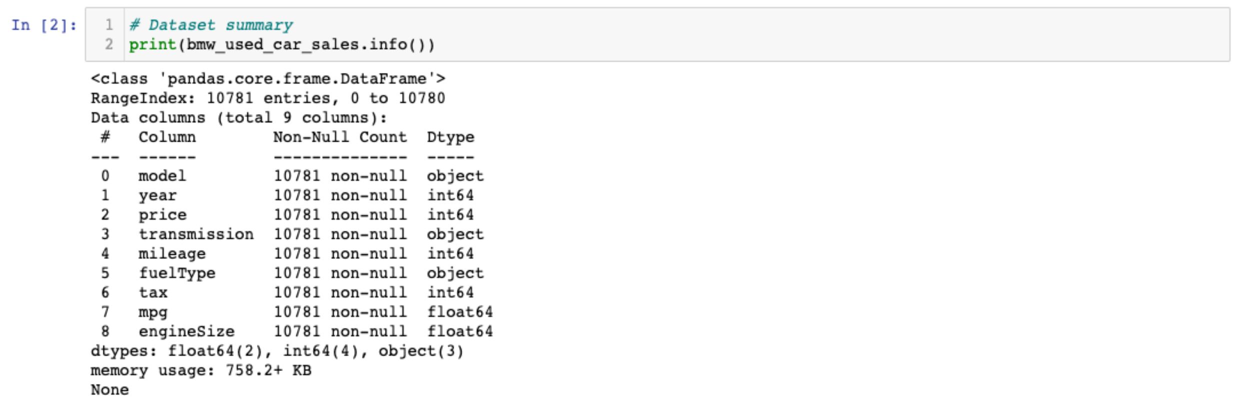
The dataset will be evaluated for any data quality issues and summarized to draw initial insights.



```
In [1]:
# Data loading
import pandas as pd
pd.options.display.float_format = '{:.2f}'.format
# Turn off some pandas warnings
pd.options.mode.chained_assignment = None

import warnings
warnings.filterwarnings("ignore")

# Create DataFrame
bmw_used_car_sales = pd.read_csv('data/bmw.csv')

# Quick look at random rows of dataset
display(bmw_used_car_sales.sample(5))
```

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
6646	1 Series	2019	19980	Semi-Auto	6102	Petrol	145	54.30	1.50
1968	4 Series	2017	20217	Manual	31959	Petrol	145	43.50	2.00
9574	3 Series	2013	11199	Manual	49720	Diesel	30	61.40	2.00
6236	3 Series	2017	25490	Semi-Auto	16363	Petrol	145	41.50	3.00
9426	3 Series	2015	13695	Manual	30000	Diesel	125	60.10	2.00

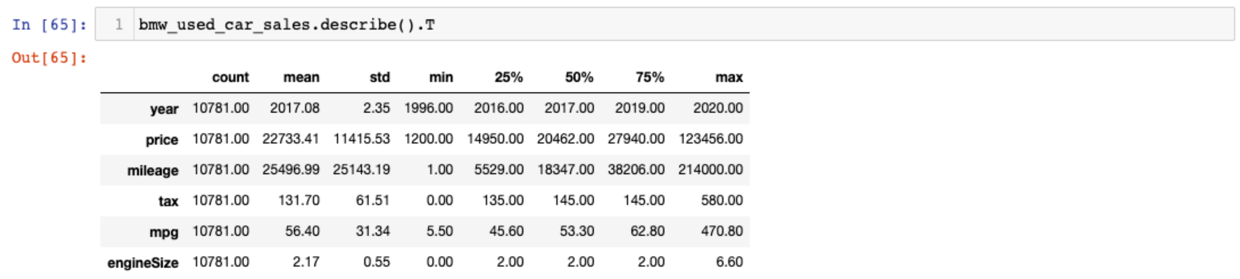
Figure 5-1 : The dataset is loaded into a DataFrame and inspected.



```
In [2]:
# Dataset summary
print(bmw_used_car_sales.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10781 entries, 0 to 10780
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   model         10781 non-null  object
 1   year          10781 non-null  int64
 2   price         10781 non-null  int64
 3   transmission  10781 non-null  object
 4   mileage       10781 non-null  int64
 5   fuelType      10781 non-null  object
 6   tax           10781 non-null  int64
 7   mpg           10781 non-null  float64
 8   engineSize    10781 non-null  float64
dtypes: float64(2), int64(4), object(3)
memory usage: 758.2+ KB
None
```

Figure 5-2: There are eight attributes, three of which are categorical.

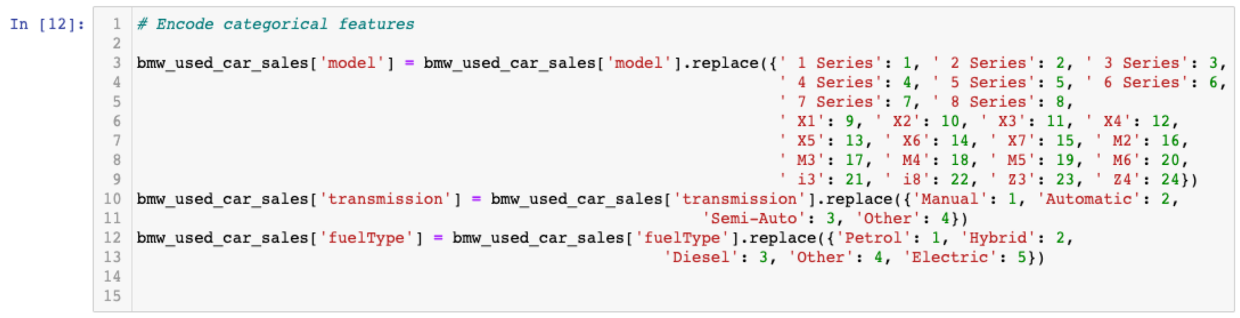


```
In [65]:
bmw_used_car_sales.describe().T
```

Out[65]:

	count	mean	std	min	25%	50%	75%	max
year	10781.00	2017.08	2.35	1996.00	2016.00	2017.00	2019.00	2020.00
price	10781.00	22733.41	11415.53	1200.00	14950.00	20462.00	27940.00	123456.00
mileage	10781.00	25496.99	25143.19	1.00	5529.00	18347.00	38206.00	214000.00
tax	10781.00	131.70	61.51	0.00	135.00	145.00	145.00	580.00
mpg	10781.00	56.40	31.34	5.50	45.60	53.30	62.80	470.80
engineSize	10781.00	2.17	0.55	0.00	2.00	2.00	2.00	6.60

Figure 5-3: A statistical summary of the dataset's numerical features.



```
In [12]:
# Encode categorical features

bmw_used_car_sales['model'] = bmw_used_car_sales['model'].replace({' 1 Series': 1, ' 2 Series': 2, ' 3 Series': 3,
                                                                   ' 4 Series': 4, ' 5 Series': 5, ' 6 Series': 6,
                                                                   ' 7 Series': 7, ' 8 Series': 8,
                                                                   ' X1': 9, ' X2': 10, ' X3': 11, ' X4': 12,
                                                                   ' X5': 13, ' X6': 14, ' X7': 15, ' M2': 16,
                                                                   ' M3': 17, ' M4': 18, ' M5': 19, ' M6': 20,
                                                                   ' i3': 21, ' i8': 22, ' Z3': 23, ' Z4': 24})
bmw_used_car_sales['transmission'] = bmw_used_car_sales['transmission'].replace({'Manual': 1, 'Automatic': 2,
                                                                                 'Semi-Auto': 3, 'Other': 4})
bmw_used_car_sales['fuelType'] = bmw_used_car_sales['fuelType'].replace({'Petrol': 1, 'Hybrid': 2,
                                                                         'Diesel': 3, 'Other': 4, 'Electric': 5})
```

Figure 5-4: Dictionaries were created for each of the unique values of each categorical attribute to encode them and make them usable in our analyses.

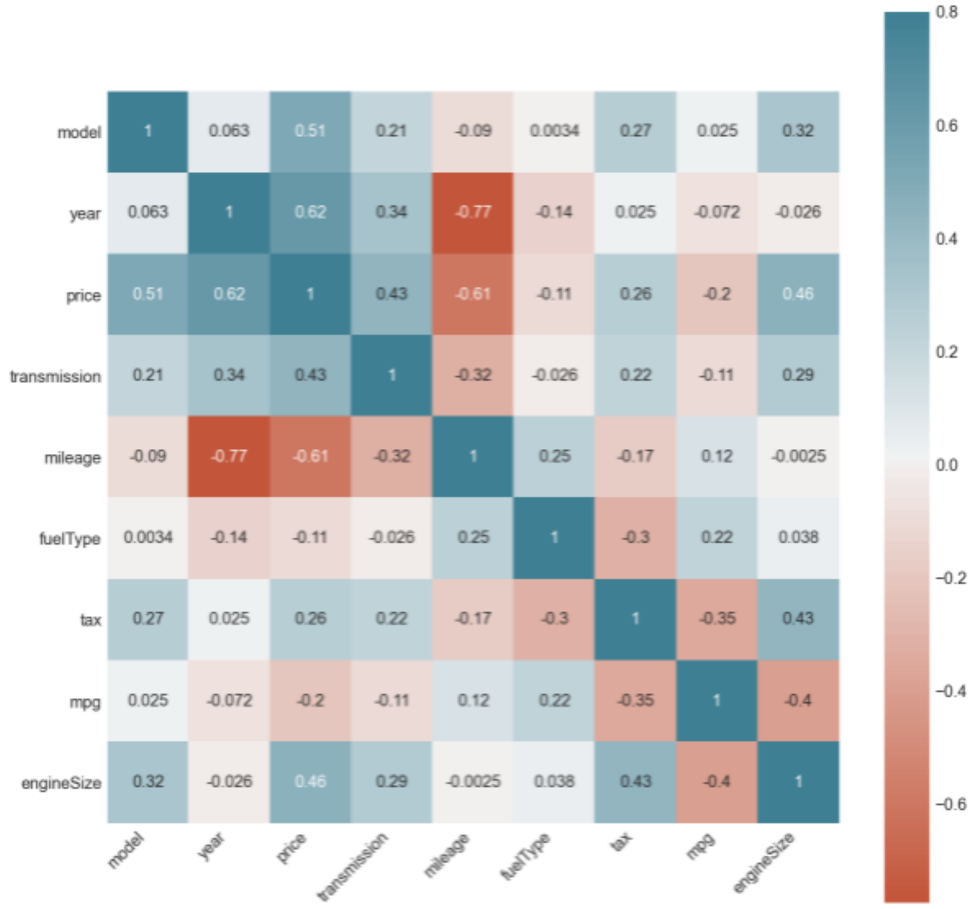


Figure 5-5: The correlation heat map shows that 'price' correlates highest with 'year' (0.62), ‘model’ (0.51), 'engineSize' (0.46), and 'transmission' (0.43). Correlations of 0.46 and 0.43 are not very high, but they still show that there is a relationship with 'price'. There's a negative correlation between 'price' and 'mileage' (-0.61) that is notable. This makes sense on an intuitive level.

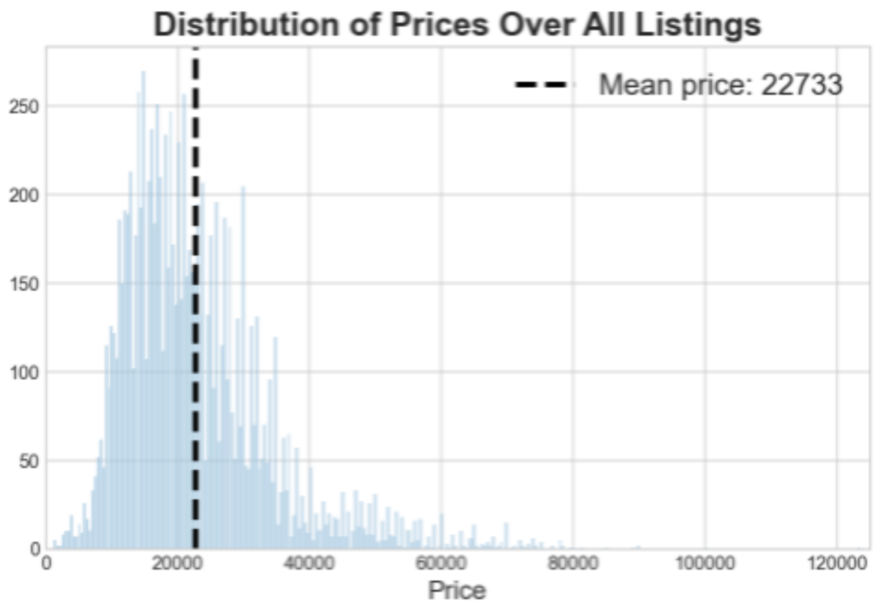


Figure 5-6: The mean price in the dataset is $22,733 and the price distribution skews right.

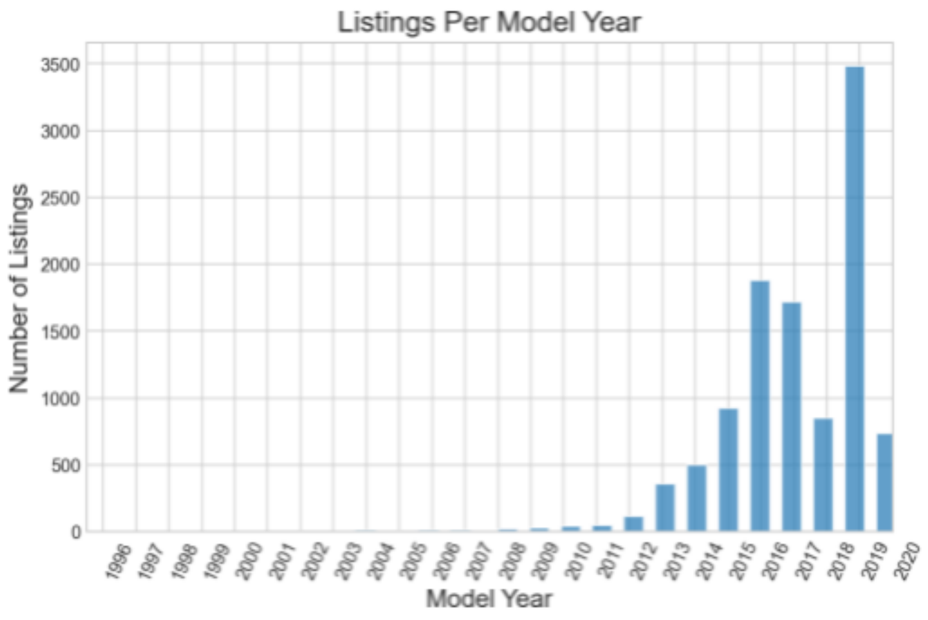


Figure 5-7: The histogram shows that there are more newer cars in this dataset. Year is highly correlated with price, so it is expected that this will play a role in the analysis.

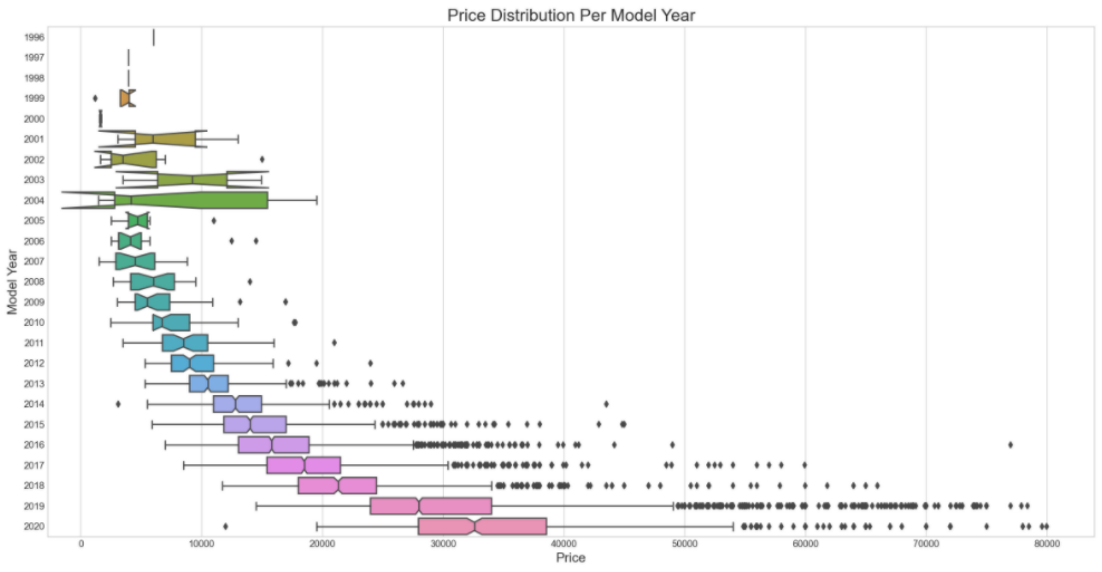


Figure 5-8: Newer cars tend to have a wider price range. This may be due to a difference in mileage within each year. Older cars may not be as appealing as newer cars if they have high mileage. This may be the reason they are not represented as much in this dataset.

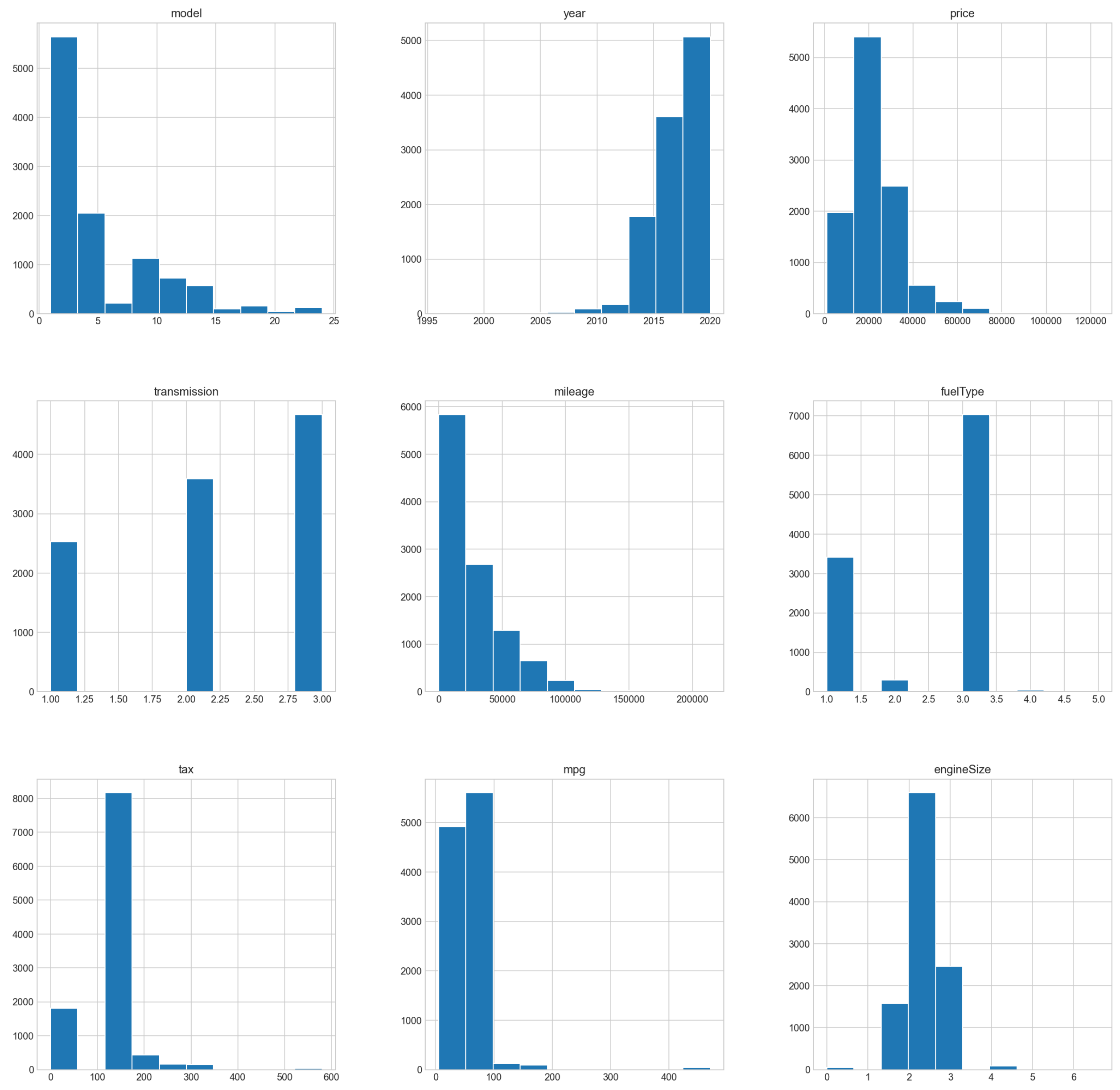


Figure 5-9: An overview of the dataset. There are more newer cars that are mostly priced around the $20,000 range with lower mileage.

6 Exploratory Data Analysis Summary

The dataset contains 10,781 entries and has a total of eight attributes. There are no missing values, but there are three categorical features that needed to be encoded. Because it is a small dataset, outliers will not be removed.

The average price of a used BMW vehicle in the dataset is $22,733 with a standard deviation of $11,415. The lowest priced vehicle is $1,200 and the highest priced is $123,456. The average mileage is 10,781 with a standard deviation of 25,143. The highest mileage on a vehicle is 214,000 and the lowest is 1. The oldest vehicle is from 1996 and the newest is from 2020.

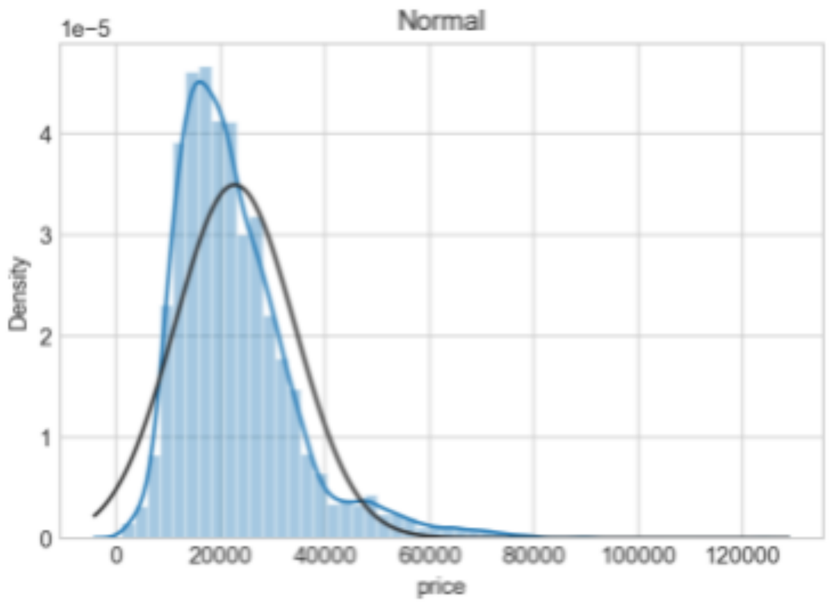
The attributes that show the highest correlation are *year*, *mileage*, *model*, *engine size*, and *transmission*. A closer look at the distribution of price per year shows that there is a general trend of prices increasing as the vehicle age decreases and the spread of the price range decreases as the age of the vehicle increases.

7 Model Development

Regression models would be appropriate to answer the questions being posed because they take a target and use a set of features, or predictors, to predict a target value. Regression is a supervised learning task. A linear regression model can make a prediction by computing a weighted sum of the features (1). A random forest regression model could be used in the same manner. It is highly accurate because it takes multiple regression decision trees running in parallel and merges the output of multiple decision trees to generate an output (2). Each tree predicts an outcome and averages them for a final estimate.

Linear Regression

Linear regression assumes a linear relationship between the target and inputs and assumes a normal distribution. The histogram for the target variable shows that the data are not normally distributed and skew right. In addition, the probability plot shows that the data points don't follow the fitted distribution line closely at the extremes. A log transformation was performed prior to performing the linear regression. Log transformation can smooth out the data, remove the skew, and make the distribution more 'normal', which will allow for the statistical analysis results from these data be more valid (3).



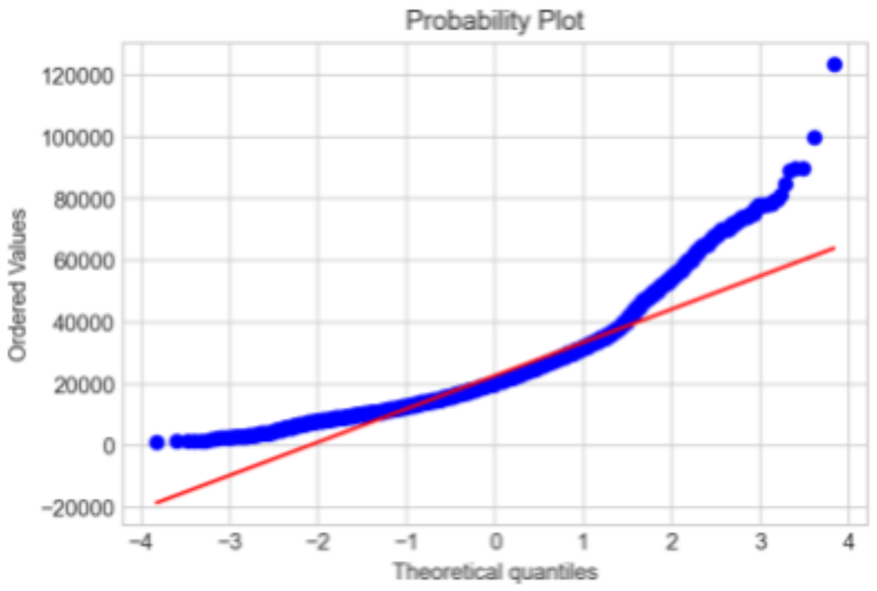
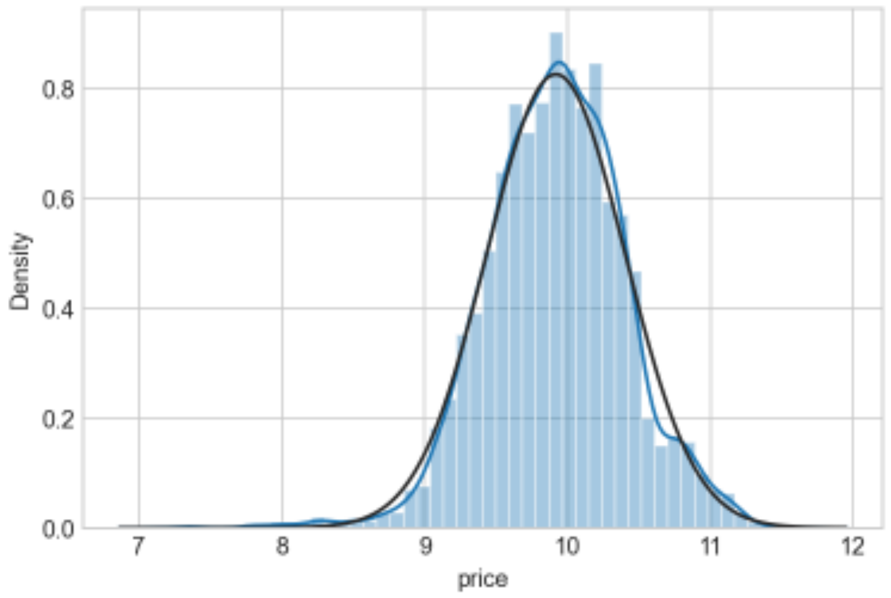


Figure 7-1: The histogram and probability plot for the target value shows that the data are not normally distributed.



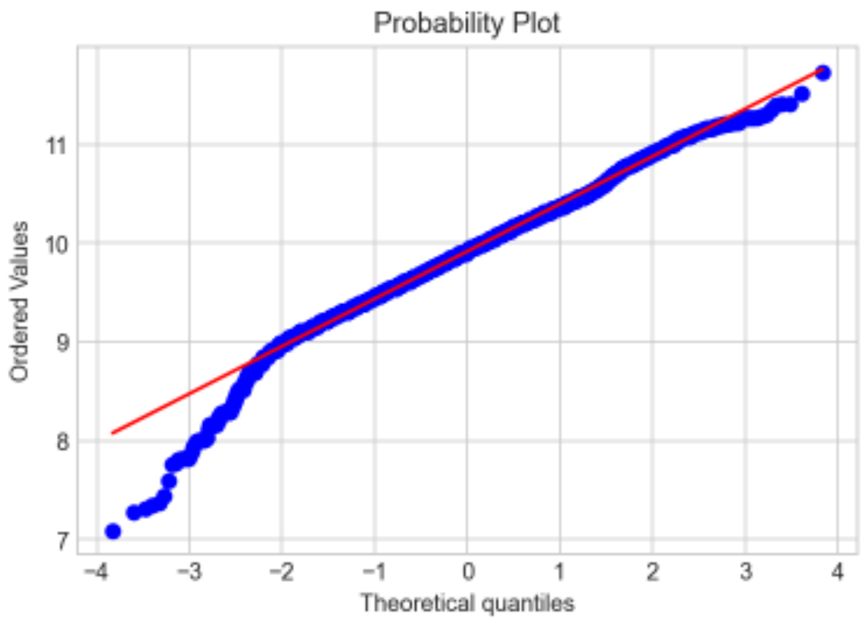
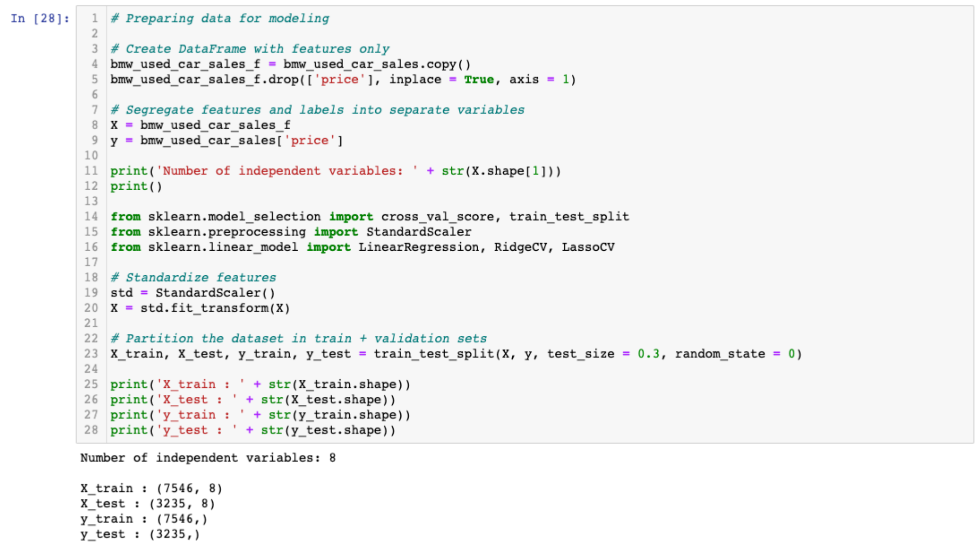
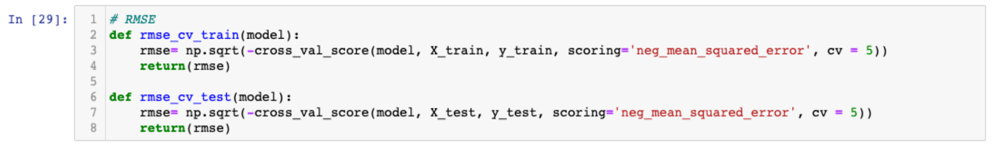


Figure 7-2: The histogram and probability plot show improvement of the distribution after log transformation.

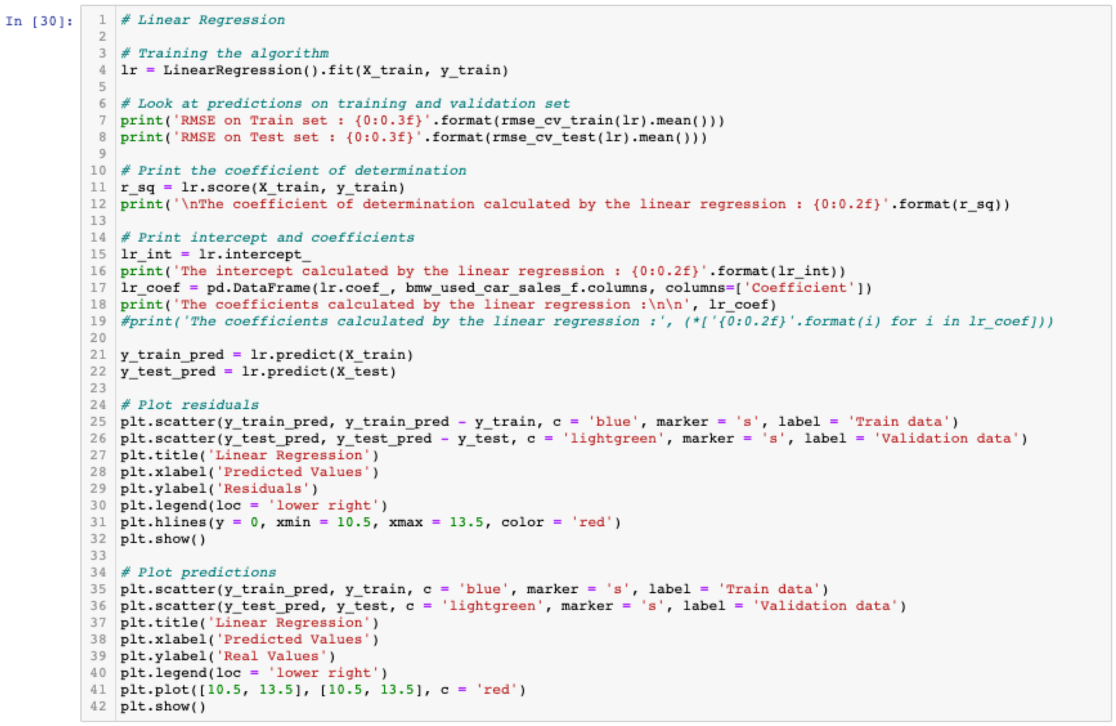


```
In [28]:
# Preparing data for modeling

# Create DataFrame with features only
bmw_used_car_sales_f = bmw_used_car_sales.copy()
bmw_used_car_sales_f.drop(['price'], inplace = True, axis = 1)

# Segregate features and labels into separate variables
X = bmw_used_car_sales_f
y = bmw_used_car_sales['price']

print('Number of independent variables: ' + str(X.shape[1]))
print()

from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV

# Standardize features
std = StandardScaler()
X = std.fit_transform(X)

# Partition the dataset in train + validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

print('X_train : ' + str(X_train.shape))
print('X_test : ' + str(X_test.shape))
print('y_train : ' + str(y_train.shape))
print('y_test : ' + str(y_test.shape))
```

```
Number of independent variables: 8

X_train : (7546, 8)
X_test : (3235, 8)
y_train : (7546,)
y_test : (3235,)
```

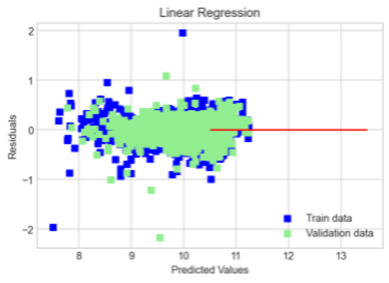


```
In [29]:
# RMSE
def rmse_cv_train(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, y_train, scoring='neg_mean_squared_error', cv = 5))
    return(rmse)

def rmse_cv_test(model):
    rmse= np.sqrt(-cross_val_score(model, X_test, y_test, scoring='neg_mean_squared_error', cv = 5))
    return(rmse)
```



```
In [30]:
# Linear Regression

# Training the algorithm
lr = LinearRegression().fit(X_train, y_train)

# Look at predictions on training and validation set
print('RMSE on Train set : {0:0.3f}'.format(rmse_cv_train(lr).mean()))
print('RMSE on Test set : {0:0.3f}'.format(rmse_cv_test(lr).mean()))

# Print the coefficient of determination
r_sq = lr.score(X_train, y_train)
print('\nThe coefficient of determination calculated by the linear regression : {0:0.2f}'.format(r_sq))

# Print intercept and coefficients
lr_int = lr.intercept_
print('The intercept calculated by the linear regression : {0:0.2f}'.format(lr_int))
lr_coef = pd.DataFrame(lr.coef_, bmw_used_car_sales_f.columns, columns=['Coefficient'])
print('The coefficients calculated by the linear regression :\n\n', lr_coef)
#print('The coefficients calculated by the linear regression :', (*['{0:0.2f}'.format(i) for i in lr_coef]))

y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

# Plot residuals
plt.scatter(y_train_pred, y_train_pred - y_train, c = 'blue', marker = 's', label = 'Train data')
plt.scatter(y_test_pred, y_test_pred - y_test, c = 'lightgreen', marker = 's', label = 'Validation data')
plt.title('Linear Regression')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.legend(loc = 'lower right')
plt.hlines(y = 0, xmin = 10.5, xmax = 13.5, color = 'red')
plt.show()

# Plot predictions
plt.scatter(y_train_pred, y_train, c = 'blue', marker = 's', label = 'Train data')
plt.scatter(y_test_pred, y_test, c = 'lightgreen', marker = 's', label = 'Validation data')
plt.title('Linear Regression')
plt.xlabel('Predicted Values')
plt.ylabel('Real Values')
plt.legend(loc = 'lower right')
plt.plot([10.5, 13.5], [10.5, 13.5], c = 'red')
plt.show()
```

```
RMSE on Train set : 0.158
RMSE on Test set : 0.165

The coefficient of determination calculated by the linear regression : 0.89
The intercept calculated by the linear regression : 9.92
The coefficients calculated by the linear regression :

               Coefficient
model                 0.13
year                  0.24
transmission          0.04
mileage              -0.14
fuelType              0.01
tax                   0.01
mpg                   0.01
engineSize            0.14
```



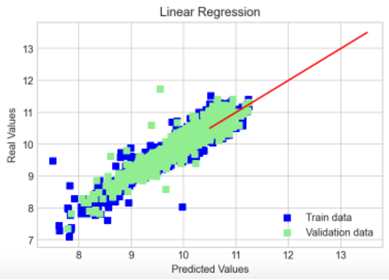
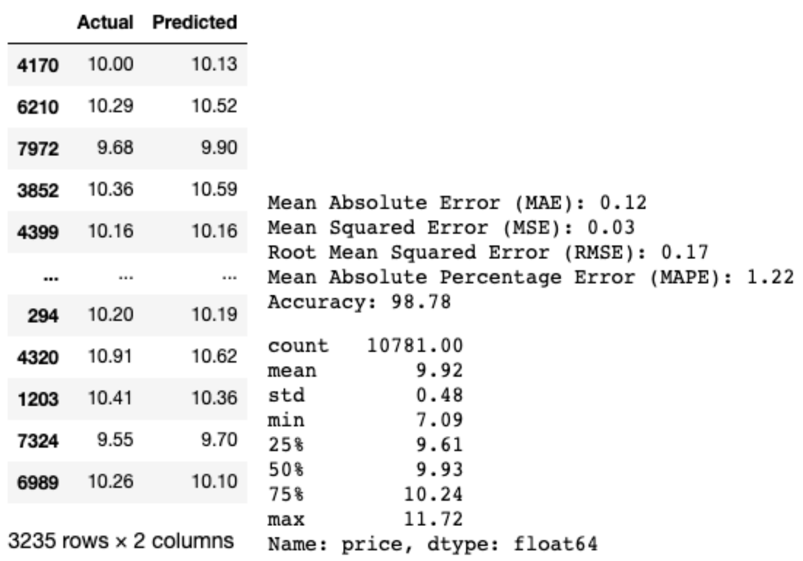


Figure 7-3: Residuals are the difference between the observed value of the target value and the predicted value (the error of prediction). In the residual plot, the predicted values are on the x-axis and the error on the y-axis. The residuals are

randomly and uniformly disbursed around the zero line (meaning a linear regression model is appropriate for the data) and cluster around it, so the model is performing well.

The coefficient of determination is large (close to 1), which could indicate the model is performing well. A large coefficient of determination could also be a sign of overfitting. The RMSE scores for both the train and test data are small, which is a good sign, and the two scores are very close to each other showing that the model is not over- or under-fitting. The linear regression model assigned highest importance to the attributes, *year*, *engine size*, *mileage*, and *model*.

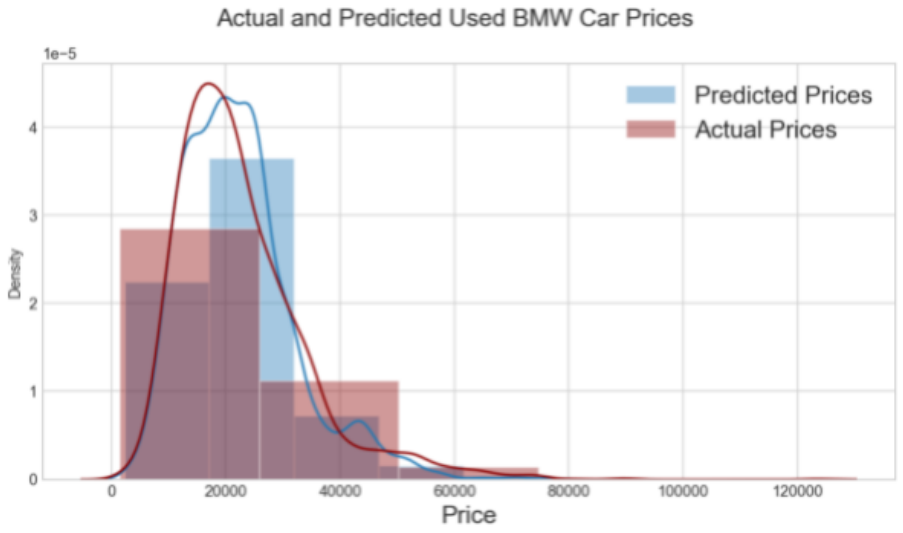


	Actual	Predicted
4170	10.00	10.13
6210	10.29	10.52
7972	9.68	9.90
3852	10.36	10.59
4399	10.16	10.16
...	...	...
294	10.20	10.19
4320	10.91	10.62
1203	10.41	10.36
7324	9.55	9.70
6989	10.26	10.10

3235 rows × 2 columns

```
Mean Absolute Error (MAE): 0.12
Mean Squared Error (MSE): 0.03
Root Mean Squared Error (RMSE): 0.17
Mean Absolute Percentage Error (MAPE): 1.22
Accuracy: 98.78

count    10781.00
mean         9.92
std          0.48
min          7.09
25%          9.61
50%          9.93
75%         10.24
max         11.72
Name: price, dtype: float64
```

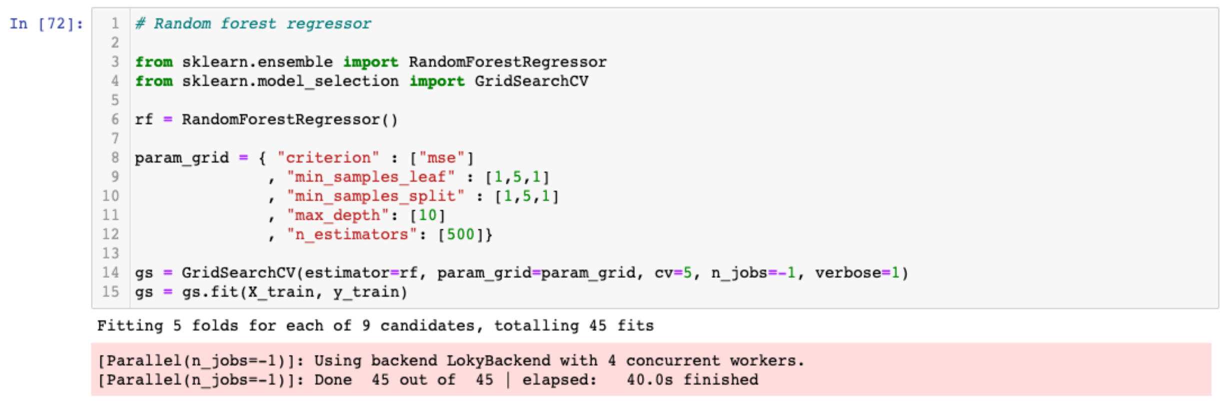
The model was used to predict new values to compare to actual values. The value of the RMSE is 0.17, less than 10% of the mean value of the price, 9.92 (this is the log transformed value), which shows that the model is performing well. The MAE, MSE, and MAPE are also low. The model accuracy is 98.78, which is great!



A close look at the actual versus predicted prices further demonstrates that the model is performing well.

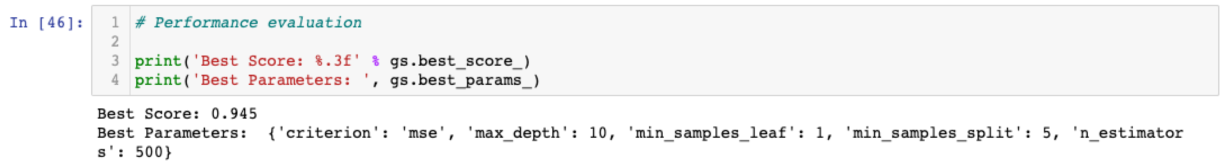
Random Forest Regressor

A random forest regressor model will also be useful for this analysis. It is efficient and highly accurate.



```
In [72]:
# Random forest regressor

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

rf = RandomForestRegressor()

param_grid = { "criterion" : ["mse"]
              , "min_samples_leaf" : [1,5,1]
              , "min_samples_split" : [1,5,1]
              , "max_depth": [10]
              , "n_estimators": [500]}

gs = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=1)
gs = gs.fit(X_train, y_train)
```

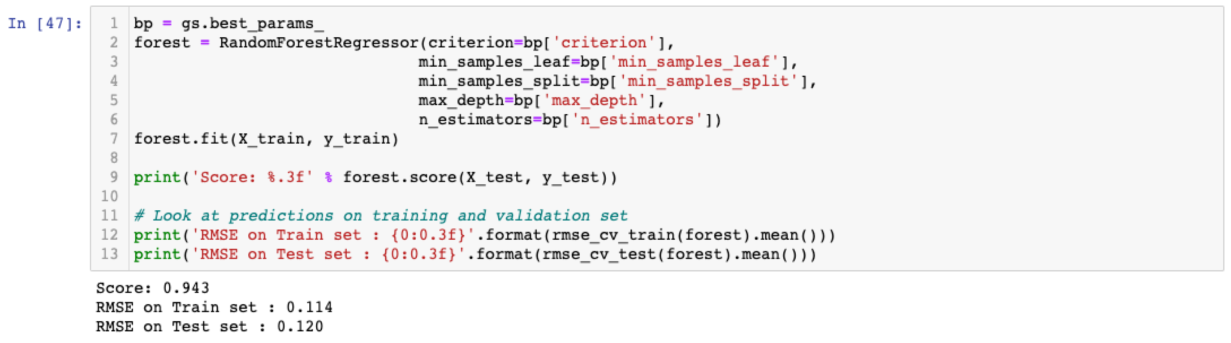
```
Fitting 5 folds for each of 9 candidates, totalling 45 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  45 out of  45 | elapsed:   40.0s finished
```



```
In [46]:
# Performance evaluation

print('Best Score: %.3f' % gs.best_score_)
print('Best Parameters: ', gs.best_params_)
```

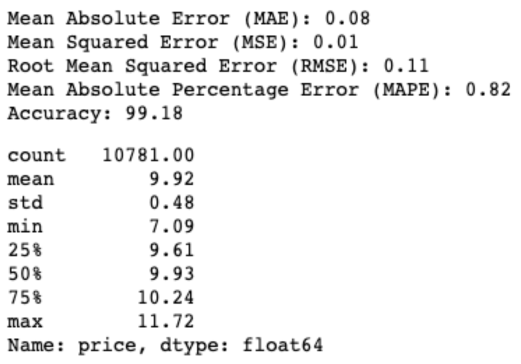
```
Best Score: 0.945
Best Parameters:  {'criterion': 'mse', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimator
s': 500}
```

Figure 7-4: A grid search was used to determine the best score and parameters for the model.

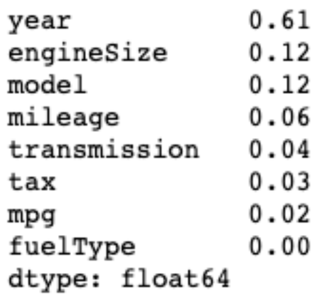


```
In [47]:
bp = gs.best_params_
forest = RandomForestRegressor(criterion=bp['criterion'],
                               min_samples_leaf=bp['min_samples_leaf'],
                               min_samples_split=bp['min_samples_split'],
                               max_depth=bp['max_depth'],
                               n_estimators=bp['n_estimators'])
forest.fit(X_train, y_train)

print('Score: %.3f' % forest.score(X_test, y_test))

# Look at predictions on training and validation set
print('RMSE on Train set : {0:0.3f}'.format(rmse_cv_train(forest).mean()))
print('RMSE on Test set : {0:0.3f}'.format(rmse_cv_test(forest).mean()))
```

```
Score: 0.943
RMSE on Train set : 0.114
RMSE on Test set : 0.120
```

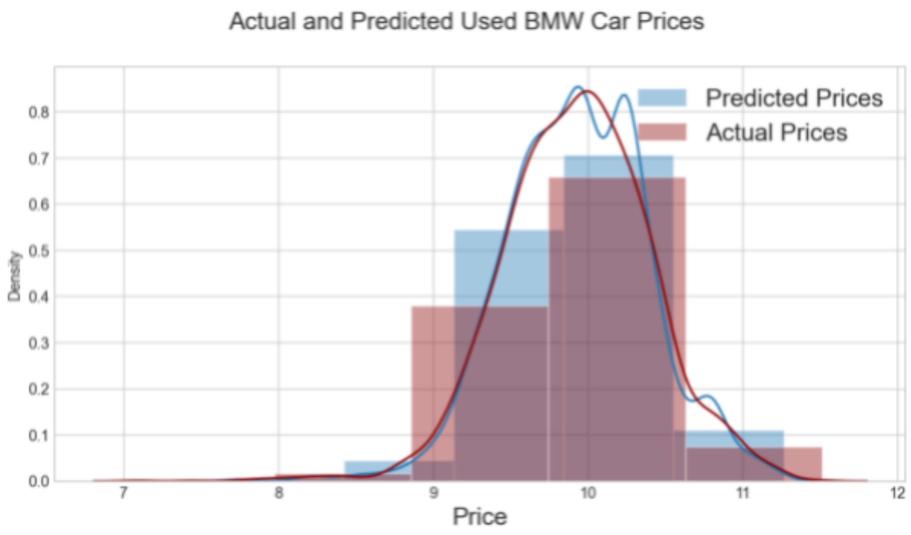


```
Mean Absolute Error (MAE): 0.08
Mean Squared Error (MSE): 0.01
Root Mean Squared Error (RMSE): 0.11
Mean Absolute Percentage Error (MAPE): 0.82
Accuracy: 99.18

count    10781.00
mean         9.92
std          0.48
min          7.09
25%          9.61
50%          9.93
75%         10.24
max         11.72
Name: price, dtype: float64
```



```
year            0.61
engineSize      0.12
model           0.12
mileage         0.06
transmission    0.04
tax             0.03
mpg             0.02
fuelType        0.00
dtype: float64
```

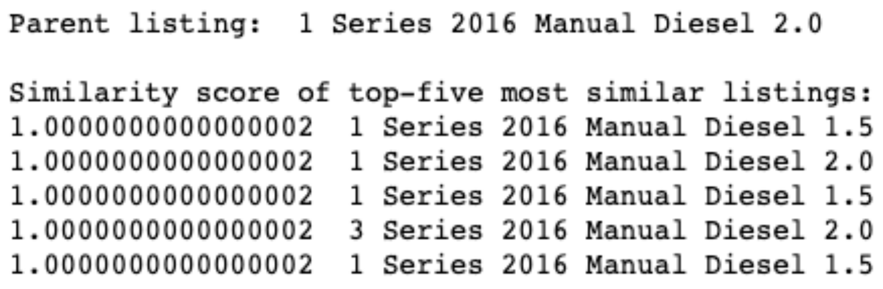
The random forest regressor model shows some improvement from the linear regression model when the MAE, MSE, RMSE, MAPE, and accuracy score are compared. The accuracy of this model is 99.18, which is very good! The random forest regressor model determined that the *year*, *engine size*, *model*, and *mileage* attributes are most important, which is consistent with the linear regression model, as well as what was discovered during exploratory data analysis.



A close look at the actual versus predicted price shows that the model is performing well.

Recommendation Engine

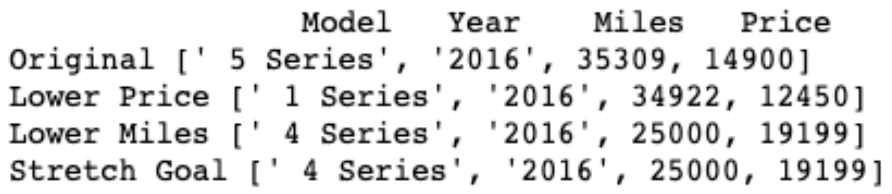
A recommendation engine was created to generate recommendations that had a 15% lower price, 10% fewer miles, and 1-30% higher price than the target car. The engine is trained on the model, year, and miles features. A sentence describing each vehicle using these features will be created and a proportion of the appearance of each word will be determined. The similarity ranking of each listing is then calculated against the rest of the listings.



```
Parent listing:  1 Series 2016 Manual Diesel 2.0

Similarity score of top-five most similar listings:
1.0000000000000002  1 Series 2016 Manual Diesel 1.5
1.0000000000000002  1 Series 2016 Manual Diesel 2.0
1.0000000000000002  1 Series 2016 Manual Diesel 1.5
1.0000000000000002  3 Series 2016 Manual Diesel 2.0
1.0000000000000002  1 Series 2016 Manual Diesel 1.5
```

The recommendation engine consists of the following functions, which will be used to return listings based on the similarity scores with these criteria:

1. Return info about any listing
2. Return a similar model with a lower price
3. Return a similar model with lower miles
4. Return a similar model with a higher price



```
                Model    Year    Miles   Price
Original [' 5 Series', '2016', 35309, 14900]
Lower Price [' 1 Series', '2016', 34922, 12450]
Lower Miles [' 4 Series', '2016', 25000, 19199]
Stretch Goal [' 4 Series', '2016', 25000, 19199]
```

A 5 Series 2016 model with 35,309 miles and priced at $14,900 returns a 1 Series 2016 model with 34,922 miles and priced at $12,450, and a 4 Series 2016 model with 25,00 miles and priced at $19,199.

8 Conclusion and Recommendations

Linear Regression

The linear regression model performed well. The coefficient of determination was large (close to 1), which could indicate the model is performing well. However, a large coefficient of determination could also be a sign of overfitting. The RMSE scores for both the train and test data were small, which is a good sign, and the two scores were very close to each other showing that the model is not over- or under-fitting.

The predicted values compared well to the actual values. The value of the RMSE for the predicted values was 0.17, less than 10% of the mean value of the actual prices, 9.92, which shows that the model performed well. With an accuracy score of 98.78, the model could be trusted to predict reliable used BMW car prices.

Random Forest Regressor

The random forest regressor model achieved an accuracy score of 99.18, which is very good. The RMSE on the train and test set showed that the model was not overfitting and the performance metrics of the model showed it was performing well and was reliable.

Recommendation Engine

The recommendation engine performed well, recommending listings of a similar year and mileage or of a similar model.

Final Thoughts

This analysis could have been more sophisticated with more car details, such as number of seats, door count, exterior and interior color, mpg, among others. This project could be expanded upon by creating a comparison between other region's and dealer's prices. In addition, data could be collected over time to predict price trends, such as how fast a model depreciates and what factors contribute to the rate of depreciation.

9 References

- (1) Géron, Aurélien (2019): Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow
- (2) M. K., Gurucharan (2020): Machine Learning Basics: Random Forest Regression, https://towardsdatascience.com/machine-learning-basics-random-forest-regression-be3e1e3bb91a
- (3) Hair, Joseph F., et. al. (2013): Multivariate Data Analysis