Battle of The Cars¶

By: Kenneth Vargas, Baheem Ferrell, Sofia Rossi

Abstract:

We were looking at data related to aspects of cars and trying to compare any data we find using machine learning methods including *Linear Regression*, *Decision Tree* and *k-NN* (*k-nearest neighbors*). We will answer any questions that we come up with from the results we find from each model presented. We will also display some graphs using matplotlib and seaborn, along with basic info provided. We will use the *automobile* dataset from a website of machine learning repository https://archive.ics.uci.edu/ml/datasets/automobile .

Introduction:

For our dataset, we decided to look at the patterns in prices and insurance ratings for various cars, as well as to compare a variety of attributes from the vehicles including makes and horsepowers. We started off by running some basic statistical analysis and modeling on the dataset, which gave us more data-driven information. The *automobile* dataset that we chose has 26 columns and 205 rows. In total, there are 16 numeric columns and 10 categorical columns, which are converted into numeric values to be evaluated in various circumstances. To answer two of our questions, we determined that there were 4 target labels: *price*, *symboling*, *horsepower* and *make*.

We preprocessed the dataset so that it would make it easier for the data analysis. First of all we found a few missing data points that were subject to correction. Afterwards, we implemented various models including *linear regression*, *decision tree* and *k-NN* models using *sklearn*, *matplotlib*, and *seaborn* to answer the questions provided below. We then used *seaborn* to plot the results we obtained from the models to better illustrate the different relationships observed.

Finally, we employed these results to answer the following 4 questions.

1. Does a vehicle's engine size and curb-weight affect the overall cost of the vehicle?
2. Does a vehicle's height, width and curb-weight affect the car's horsepower?
3. Would a model be able to detect what the make of the car is based on various features given by the dataset?
4. Do faster and/or bigger cars affect the insurance rating of a vehicle?

The results from the research and the learnings from various resources led us to believe the aforementioned models we selected fit best to the dataset. By utilizing the various models, we were able to provide answers to all of the above questions, which we'll discuss in detail in the following sections.

Related Work:

Can machine learning techniques predict customer dissatisfaction? A feasibility study for the automotive industry

https://www5.informatik.uni-erlangen.de/Forschung/Publikationen/2017/Meinzer17-CML.pdf

This paper takes a look at the automotive industry as a whole to find a better way of predicting customer satisfaction without having to use questionnaires. They took a look at some fundamental business questions such as "Can dissatisfied customers be classified based on data that is produced during every service visit?" and "Can the dissatisfaction indicators be derived from service process data?". They compared 5 machine learning classifiers in analyzing 19,008 data records. They suggested that their model was able to classify customer dissatisfaction only by looking into the objective data that every record contains.

Predicting the Price for Used Cars

https://towardsdatascience.com/building-a-linear-regression-model-for-predicting-the-price-of-a-used-car-now-with-logs-54a478438d1

This paper investigates if it is possible to accurately predict a used car's price based on dependent variables such as *city_mpg*, *highway_mpg*, *mileage*, *yr_old*, and *luxury*. The data was gathered from *cars.com* and was cleaned up. A linear regression model was used along with natural logs to find any different coefficient between the x values and the price. It was concluded that using the obtained formula the price could be predicted fairly accurately given other features of vehicles.

Statistical Analysis/Experiment:

- Seaborn Graphs to Display Models
- Heat Map
- Bar Graphs
- Histograms
- Density Plots

In the beginning of the Jupyter notebook, we performed some statistical analysis on our dataset to best understand the underlying relationships between various features.

	symboling	wheel-base	length	width	height	\
count	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	0.834146	98.756585	174.049268	65.907805	53.724878	
std	1.245307	6.021776	12.337289	2.145204	2.443522	
min	-2.000000	86.600000	141.100000	60.300000	47.800000	
25%	0.000000	94.500000	166.300000	64.100000	52.000000	
50%	1.000000	97.000000	173.200000	65.500000	54.100000	
75%	2.000000	102.400000	183.100000	66.900000	55.500000	
max	3.000000	120.900000	208.100000	72.300000	59.800000	

	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000
mean	2555.565854	126.907317	10.142537	25.219512	30.751220
std	520.680204	41.642693	3.972040	6.542142	6.886443
min	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2935.000000	141.000000	9.400000	30.000000	34.000000
max	4066.000000	326.000000	23.000000	49.000000	54.000000

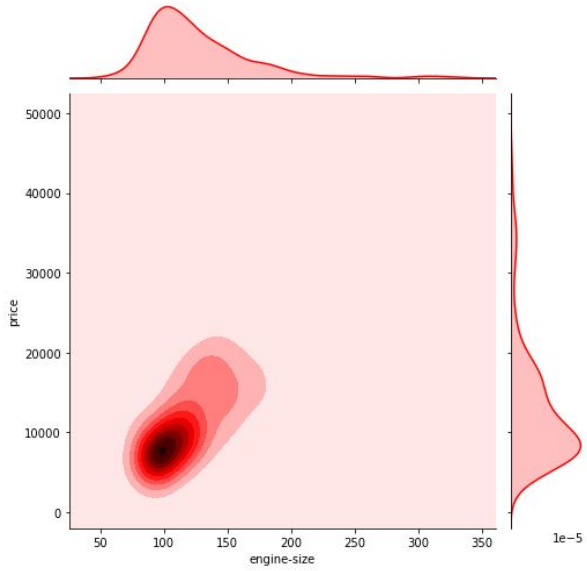
```
There are 205 rows in the dataframe
There are 26 columns in the dataframe
The target label for the data set is make
The dataset is mostly numeric with 16 columns and 10 categorical
column
```

This was done by printing out the first 5 rows of the dataset and calculating basic statistical values such as the mean and standard deviation of each feature.

A joint plot and a pair plot were then drawn between each pair of variables to understand the relationship between them. Density plots between *engine-size* and *price*, and between *curb-weight* and *price* are illustrated below.

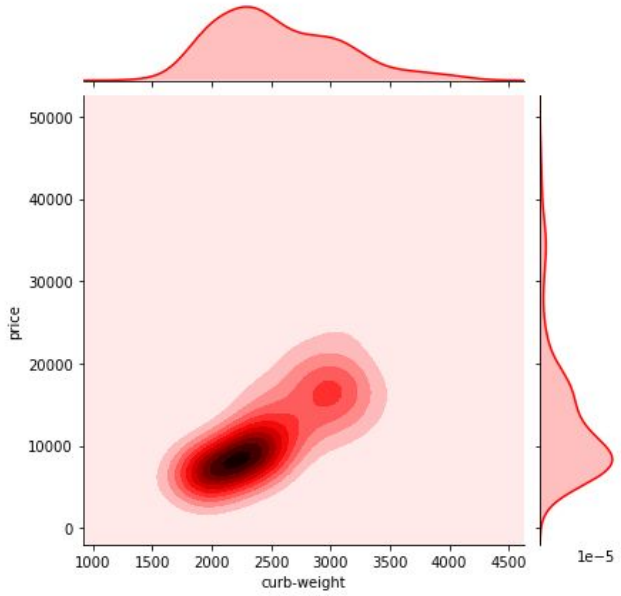
```
#density plot showing price and engine-size
sns.jointplot(x="engine-size", y="price", data=car.replace('?',np.NAN), height=7, kind="kde", color="r")
```

```
<seaborn.axisgrid.JointGrid at 0x20d63780700>
```

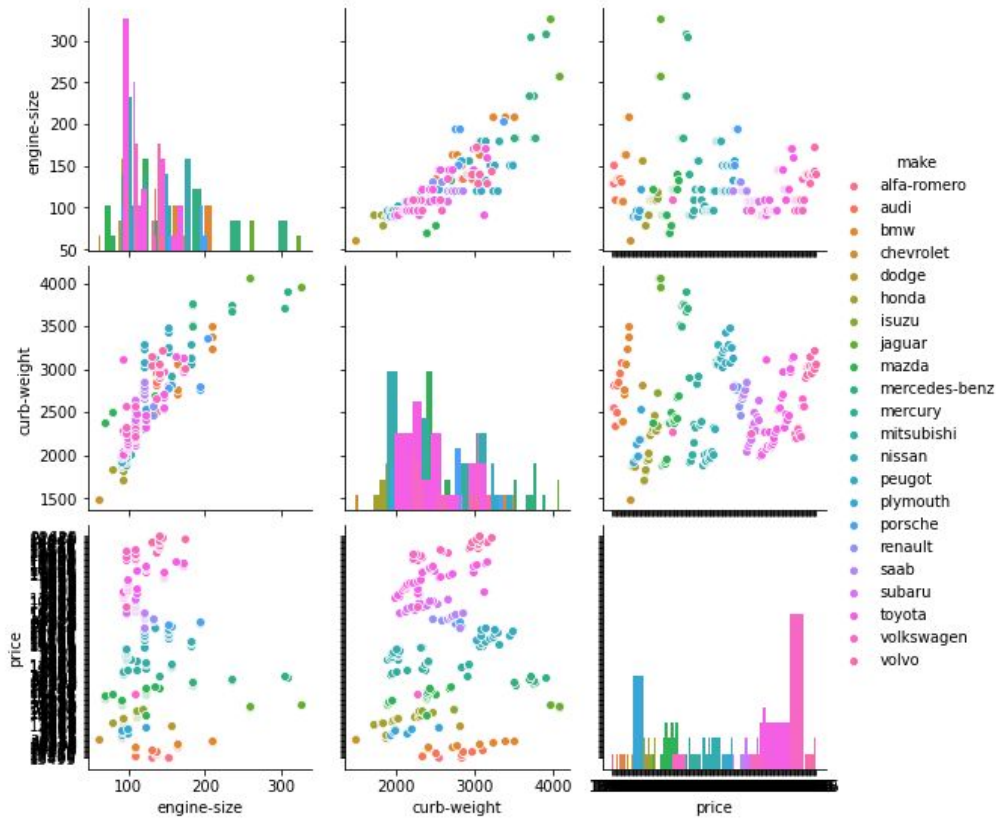


```
#density plot showing price and curb-weight
sns.jointplot(x="curb-weight", y="price", data=car, height=6, kind="kde", color="r")
```

```
<seaborn.axisgrid.JointGrid at 0x20d638a26d0>
```




```
# scatterpots that compair vairables engine-size, curb-weight, and price to find any visual relations
g = sns.pairplot(car[["engine-size", "curb-weight","price", "make"]], hue="make", diag_kind="hist")
```



As can be seen from the above graphs, *engine-size* and *curb-weight* are in positive correlation. Similar graphs for *price* and *engine-type* did not illustrate the same correlation.

```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(df_automobile['curb-weight'], df_automobile['price'])
print( "The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P = ", p_value)
```

```
The Pearson Correlation Coefficient is 0.8169803056556605  with a P-value of P =  2.0043475228093137e-50
```

+ Code + Text

Since the p-value is < 0.001, the correlation between curb-weight and price is statistically significant, and the linear relationship is quite strong (~0.834).

```
[ ] pearson_coef, p_value = stats.pearsonr(df_automobile['engine-size'], df_automobile['price'])
    print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.8509113328793512  with a P-value of P = 1.1514343900966746e-58
```

Since the p-value is < 0.001, the correlation between engine-size and price is statistically significant, and the linear relationship is very strong (~0.872).

```
[ ] pearson_coef, p_value = stats.pearsonr(df_automobile['engine-type'], df_automobile['price'])
    print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

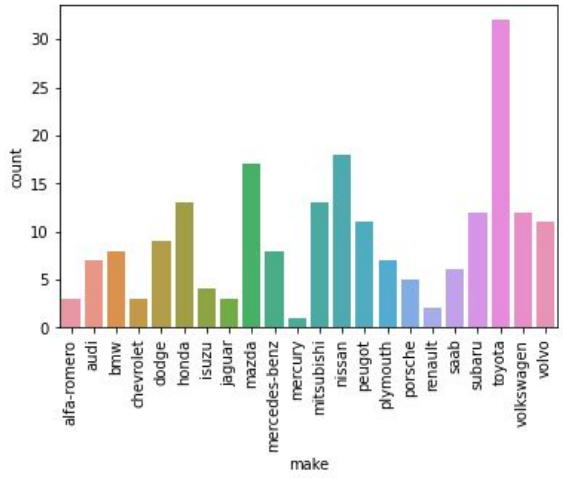
```
The Pearson Correlation Coefficient is 0.08253271381569338  with a P-value of P = 0.2394066564640234
```

Since the p-value is > 0.001, the correlation between engine-size and price is statistically significant, and the linear relationship is 0.07

We then calculated correlations (Pearson coefficient) between *engine-size*, *type* and *curb-weight* with *price* respectively. When this coefficient is below <0.001, it means there is a significant correlation between these variables. Otherwise, it means those variables are not related, i.e there is no correlation. The results are displayed above.

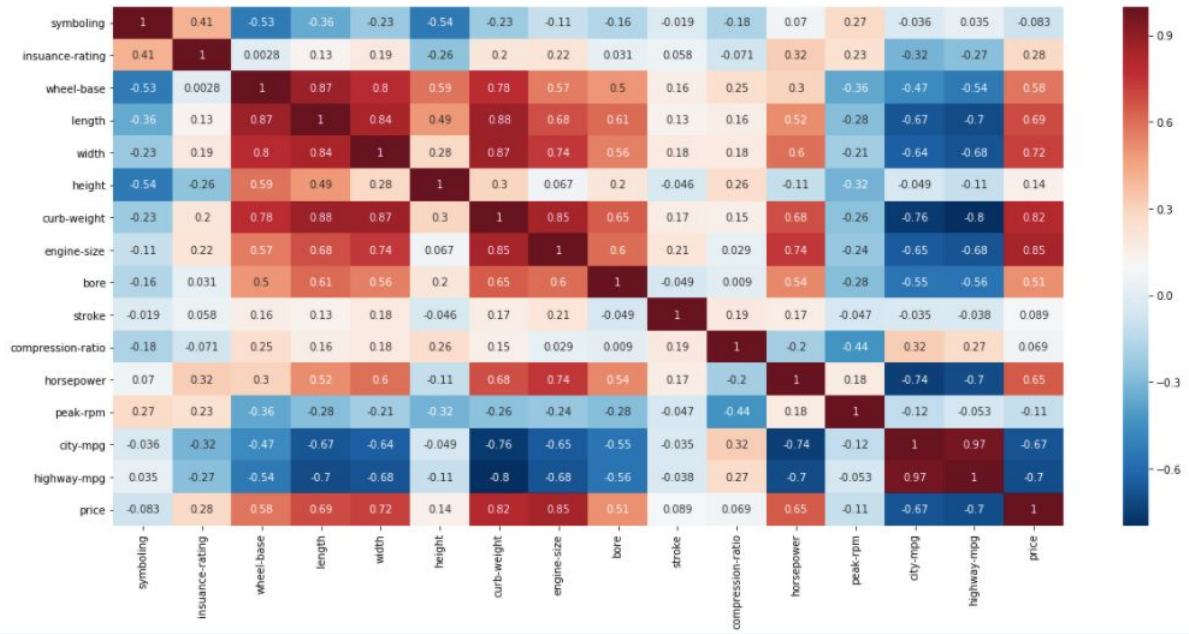
We then ran a bar graph that showed the number of cars in each category.

```
#counts the number of make cars in the Dataset
make = sns.countplot(car['make'])
make.set_xticklabels(make.get_xticklabels(), rotation = 90);
```



Linear Regression

```
#see hat varriabe goes best with the dependent variables of price and horsepower
correlation = car.corr()
plt.figure(figsize=(20, 9))
heatmap = sns.heatmap(correlation, annot=True, linewidths=0, cmap="RdBu_r")
```



A heat map was used to find any correlations between features in the dataset. It was found that *curb-weight* and *engine-size* had relatively high correlations around 0.82 and 0.85 respectively with *price*. It was also found that *height*, *width* and *curb-weight* had correlations of -0.11, 0.6 and 0.68 respectively with *horsepower*. It was therefore

determined prior to any model training that the *horsepower* model won't be as accurate as the *price* model.

Methods:

- *Linear Regression*
- *k-NN*
- *Decision Tree*

The first method we used was the *Linear Regression Model* which worked well in predicting y value based on other features of vehicles. The variable we want to predict is called the dependent variable or the y value. The variable(s) we are using to predict the dependent variable is(are) called the independent variable(s) or the x value(s). What we will be predicting is the overall cost of the vehicle and the car's horsepower using the independent variables including *engine-size*, *curb-weight*, *height*, and *width*.

Model 1 Engine Size, Curb-Weight vs. Price

```
#drop nans
car1 = car.dropna();

#get our x and y variables ready
x = car1.iloc[:, [13,16]].values
y = car1['price'].values

#Fit linear regression model and get coefficient, intercept and slop along with
#Predicted Response
model = LinearRegression().fit(x, y)
r_sq = model.score(x, y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
y_pred = model.intercept_ + model.coef_ * x
print('predicted response:', y_pred, sep='\n')
```

```
coefficient of determination: 0.8092652000281897
intercept: -14552.15911513411
slope: [ 8.45753484 43.46960542]
predicted response:
[[  5213.09979552  -9813.97212446]
 [  9331.91926043  -8640.29277815]
 [  9501.06995714  -8640.29277815]
```

Model 1 uses engine-size and curb-weight as x values to predict the y value (price) by fitting the dataset into a linear regression model. From the obtained model R squared value is calculated as the score of our model. We then take the intercept of the model and the slope using intercept_ and coef_ respectively. Using these values, y value is predicted by multiplying the x values of each variable with the corresponding slope and

adding them together and finally added together with the intercept. The obainted accuracy was 80.9%. This answered our first question.

Model 2 Height, Width, and Curb-Weight vs. Horsepower

```
#get our x and y variables ready
x = car1.iloc[:, [11,12,13]].values
y = car1['horsepower'].values

#Fit linear regression model and get coefficient, intercept and slop along with
#Predicted Response
model = LinearRegression().fit(x, y)
r_sq = model.score(x, y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
y_pred = model.intercept_ + model.coef_ * x
print('predicted response:', y_pred, sep='\n')
```

```
coefficient of determination: 0.7006095550973628
intercept: 223.53565706683588
slope: [-0.90963798 -4.02525424  0.06051615]
predicted response:
[[ 1.63317622e+02  4.96435197e+00  3.64961891e+02]
 [ 1.63135695e+02  4.96435197e+00  3.94433255e+02]
 [ 1.58587505e+02 -6.71003962e-01  3.95643578e+02]
```

We go through the same steps to produce Model 2 except that we substitute the x values with new independent variables (*height*, *width* and *curb-weight*) and y values with the new dependent variable (*horsepower*). The obtained accuracy was 70.0%. This provided the answer to our second question.

Secondly the k-NN algorithm was most useful for answering our third question: Would a model be able to detect what the make of the car is based on various features given by the dataset? Our goal here is to determine if any features of a vehicle can be used to predict the target feature -- in this case, the make of the car.

The first feature we decided to test was the size of the engine. With just the engine size, we're able to get an accuracy of 65.4% when k = 6, which isn't particularly accurate.

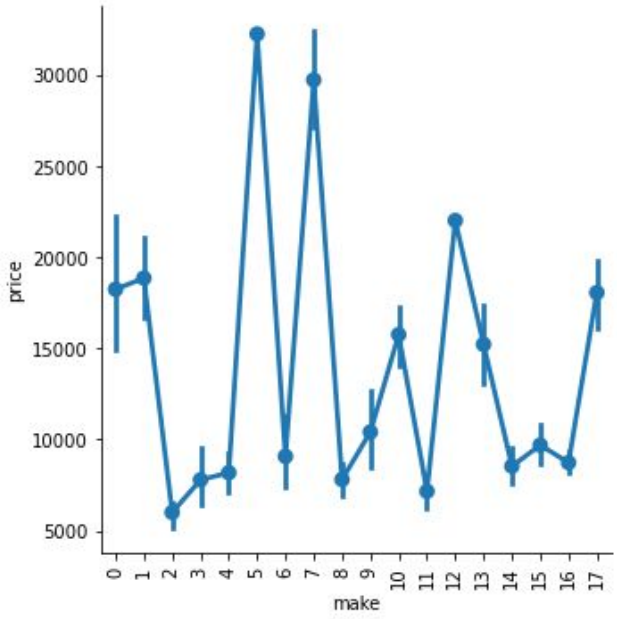
The next feature we decided to test was the wheelbase, which is the distance between the centers of the front and rear wheels of the vehicle. With just the wheelbase, we're able to get an accuracy of 81.7 when k = 1, which is definitely an improvement from just having the engine size.

Finally, we decided to combine those two features, to see if having both of those features together would improve the accuracy at all. As it turns out, it did improve the accuracy when trying to predict the make using the K-NN algorithm. We now ended up

with an accuracy of 94.9% when k = 1. So, by just utilizing these two features (out of a possible 20+ features), we're able to accurately predict the make of the vehicle we are looking at.

```
le = preprocessing.LabelEncoder()
car2 = car.dropna().copy(True)
car2["make"] = le.fit(car2["make"]).transform(car2["make"])
df_temp = car2[car2['price']!='?']
normalised_mean = car2['price'].astype(int).mean()
car2['price'] = car2['price'].replace('?',normalised_mean).astype(int)
g = sns.catplot(data=car2, x="make", y="price", kind="point")
g.set_xticklabels(rotation=90)
```

```
<seaborn.axisgrid.FacetGrid at 0x20d677d8d60>
```



In addition to these features, different models appear to have different price ranges according to the above catplot. However, there still exists price overlaps between different models. Therefore, price was also used to predict the make with *k-NN* which resulted in 93.7% accuracy. This combined with the above results answered our third question. (5-fold cross-validation is used to suppress overfitting. Note that there is a warning of a class containing only 1 member. This is due to the fact that the dataset is small even for this simple model.)

```
# KNN
X = car2["price"]
X = np.array(X).reshape(-1,1)
Y = car2["make"]
model = KNeighborsClassifier()
# grid search
param_grid = {'n_neighbors': list(range(1,11))}
grid = GridSearchCV(model, param_grid, cv=5)
grid.fit(X, Y)
print("Grid Search: best parameters: {}".format(grid.best_params_))
# accuracy of best model with confidence interval
best_model_knn = grid.best_estimator_
Y_predict = best_model_knn.predict(X)
acc = accuracy_score(Y, Y_predict)
print("Accuracy: {:3.6f}".format(acc))
```

```
Grid Search: best parameters: {'n_neighbors': 1}
Accuracy: 0.937107
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:670: UserWarning: The least populated class in
y has only 1 members, which is less than n_splits=5.
  warnings.warn(("The least populated class in y has only %d"
```

Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making). Aim of the decision tree model is to predict the car brand based on its *engine-size* and *wheel-base*. In the code firstly we have initialized DecisionTreeClassifier. Once this is initialised, X and Y (where X is *engine-size* and *wheel-base*, and Y is *make*) is fed to the model to be trained. Once the model is trained, we put the X again to predict the values. In this step the model predicts the car make and this is stored in *y_pred*.

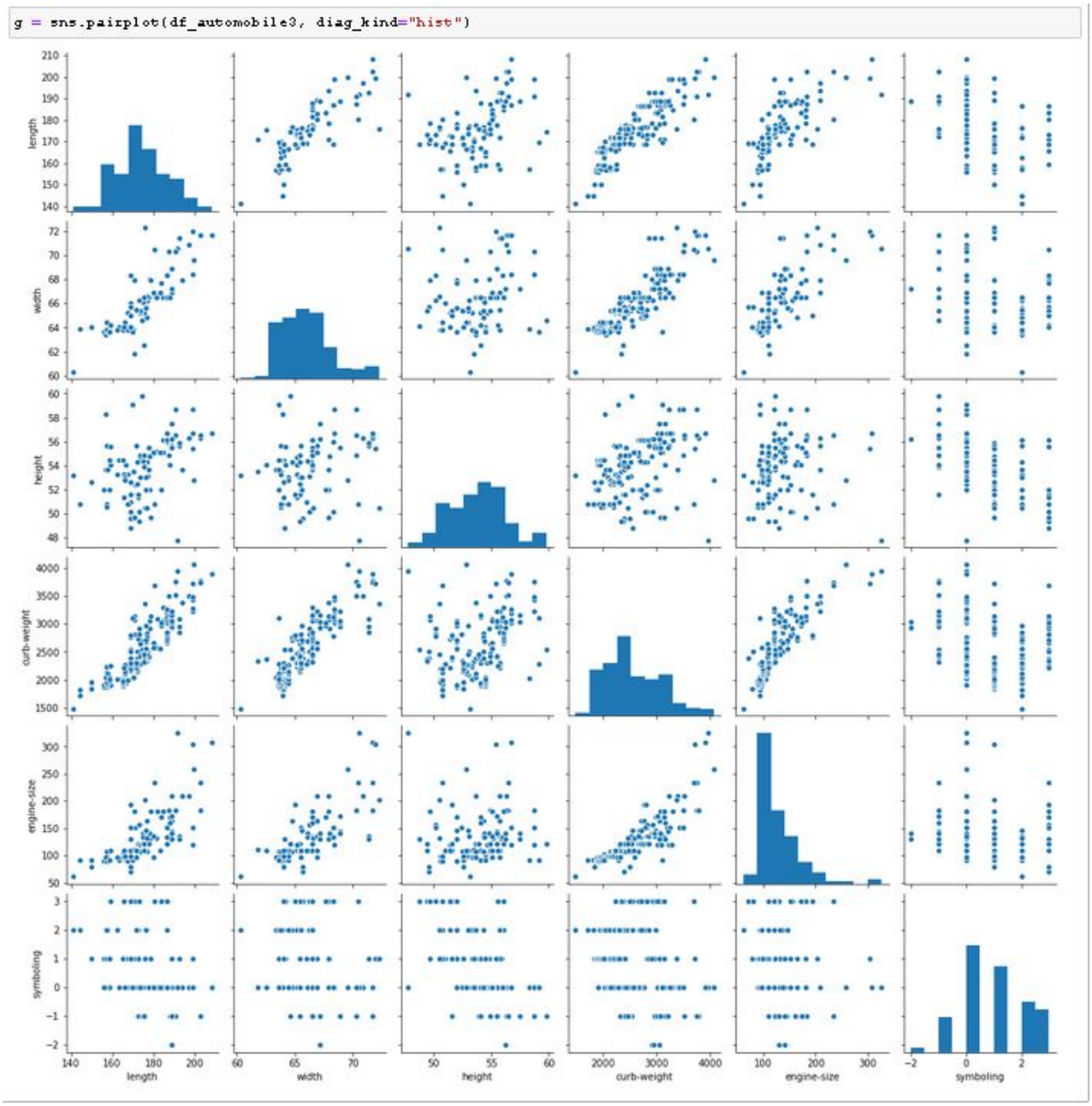
DECISION TREE

```
X = car2[["engine-size", "wheel-base"]]
X = np.array(X).reshape(-1,2)
Y = car2["make"]
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, Y)
y_pred = clf.predict(X)
acc = accuracy_score(Y, y_pred)
print("Accuracy: {:3.6f}".format(acc))
```

```
Accuracy: 0.955975
```

We compare the predicted make with the labeled values. This way accuracy is calculated to be 95%.

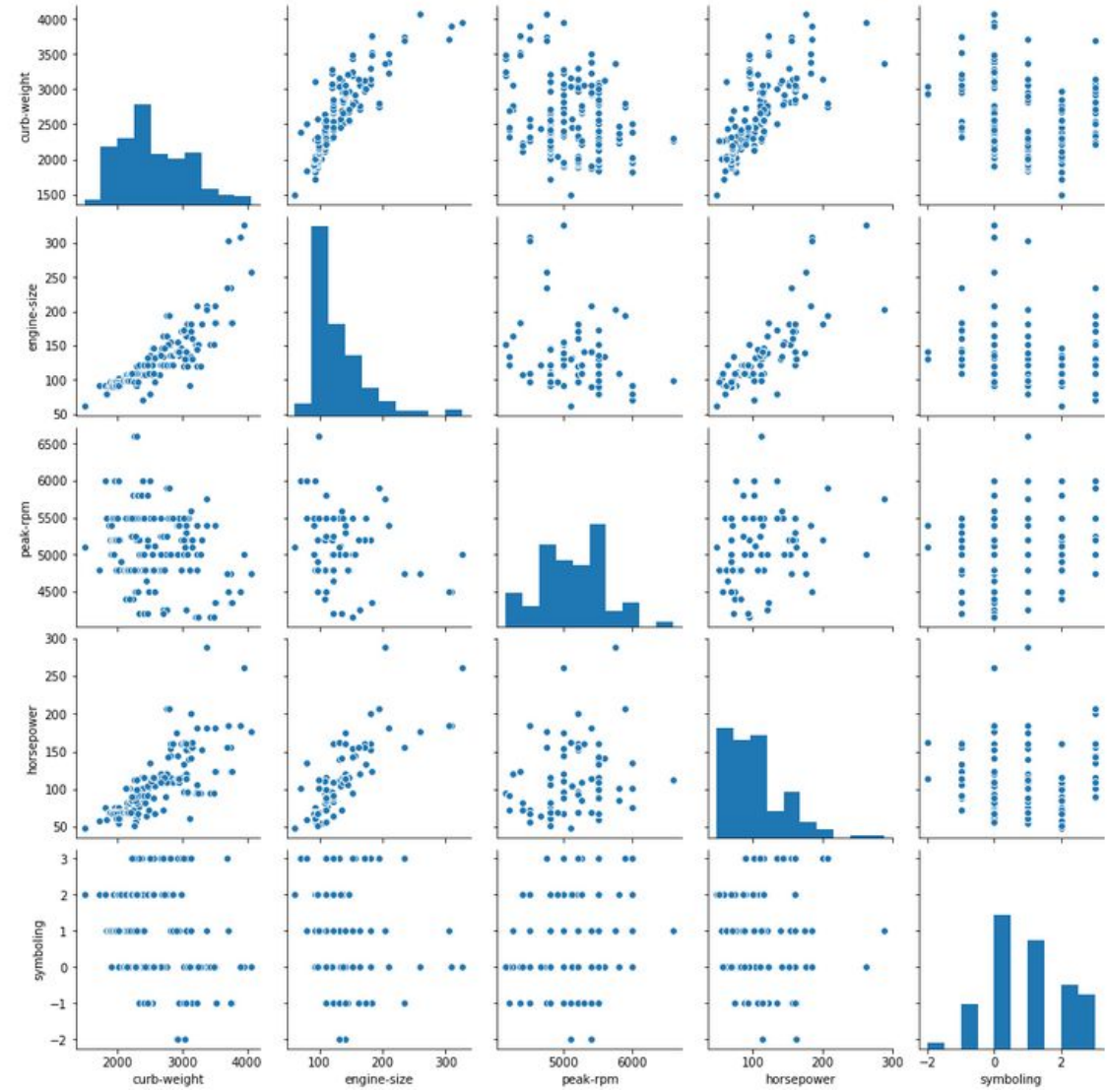
In order to answer our fourth question, pairplot between various features and *symboling* are drawn.



According to the above table and plots, size-related features have more or less negative correlations with *symboling* meaning that the bigger the car, the lower the *symboling* and therefore the safer.

Speed-related features do not appear to have similar correlations with *symboling*. Some features have positive correlations while others have negative correlations in addition to *horsepower* having almost no correlation with *symboling*.

```
g = sns.pairplot(df_automobile4, diag_kind="hist")
```



Conclusion:

In conclusion, *engine-size* and *curb-weight* are positively correlated with price. *height* has no effect on horsepower whereas *width* and *curb-weight* have positive effects on horsepower. When k=1, car models can be detected based on prices with 95% accuracy. Hence, a better model will be able to detect what kind of car brand it is based on values provided that there is enough data. Size-related features have more or less negative correlations with *symboling* meaning that the bigger the car, the lower the symboling and therefore the safer. In conclusion, bigger cars tend to have better insurance ratings than smaller cars. Speed-related features do not appear to have

similar correlations with symboling. Some features have positive correlations while others have negative correlations. In the end, speed is not a major factor in deciding insurance ratings of vehicles.