Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

# Car Evaluation Project Report

There are more cars than people today. According to A Plus, there are 10 billion cars in the world, yet only a small portion of those are active and on the road. With all these different cars, there are many attributes that constitute to an acceptable car. That is the goal of my project. I'm trying to find the features and attributes a car must have to determine it "acceptable". Also, I'm trying to understand which classifier algorithms pave the way for determining the acceptability or unacceptability of future cars.

I am using the Car Evaluation dataset from UCI. This dataset was derived from a simple hierarchical decision model. The model was developed from the demonstration of M. Bohanec, V. Rajkovic, who were experts in systems for decision making. The model evaluates cars according to different attributes in a specific structure.

The Car Evaluation dataset contains 1,700+ samples with six features: buying, maint, doors, person, lug_boot, safety. The **buying** feature corresponds to the buying price of the car. The **maint** feature is the price of maintenance. The **doors** feature is the number of doors, **persons** is the capacity in terms of persons to carry, **lug_boot** is the size of the luggage boot, and **safety** is the estimated safety of the car.

One problem I encountered with this dataset was with the values. the data was given with categorical labels, that is, the labels were of the string datatype instead of ints.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | vhigh | vhigh | 2 | 2 | small | low | unacc |
| 1 | vhigh | vhigh | 2 | 2 | small | med | unacc |
| 2 | vhigh | vhigh | 2 | 2 | small | high | unacc |
| 3 | vhigh | vhigh | 2 | 2 | med | low | unacc |
| 4 | vhigh | vhigh | 2 | 2 | med | med | unacc |
| 5 | vhigh | vhigh | 2 | 2 | med | high | unacc |
| 6 | vhigh | vhigh | 2 | 2 | big | low | unacc |

String labels are unsuitable for machine learning so I had to preprocess the data to make all the values of the integer datatype. I needed to convert the strings into unique numeric values.

Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

I accomplished this by creating a simple conversion chart in which each of the categorical labels corresponded to a certain number. For example, any label with vhigh, high, med, and low would correspond to 4, 3 , 2, 1 respectively. In the code, this is simply done with the *replace* function.

```
df = df.replace('vhigh', 4)
df = df.replace('high', 3)
df = df.replace('med', 2)
df = df.replace('low', 1)
df = df.replace('5more', 6)
df = df.replace('more', 5)
df = df.replace('small', 1)
df = df.replace('med', 2)
df = df.replace('big', 3)
df = df.replace('unacc', 1)
df = df.replace('acc', 2)
df = df.replace('good', 3)
df = df.replace('vgood', 4)
```

Now the new dataframe is a lot more useful for machine learning.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 4 | 4 | 2 | 2 | 1 | 1 | 1 |
| 1 | 4 | 4 | 2 | 2 | 1 | 2 | 1 |
| 2 | 4 | 4 | 2 | 2 | 1 | 3 | 1 |
| 3 | 4 | 4 | 2 | 2 | 2 | 1 | 1 |
| 4 | 4 | 4 | 2 | 2 | 2 | 2 | 1 |
| 5 | 4 | 4 | 2 | 2 | 2 | 3 | 1 |
| 6 | 4 | 4 | 2 | 2 | 3 | 1 | 1 |
| 7 | 4 | 4 | 2 | 2 | 3 | 2 | 1 |
| 8 | 4 | 4 | 2 | 2 | 3 | 3 | 1 |

Further preprocessing I made was creating redundant features in order to put them in variables, to use them in an integer index to represent a value for an attribute. This is a redundant process however it will be worth it when creating the tree. This will allow the tree to have more choice on how to split the data on each branch.

I decided to initially use a DecisionTreeClassifier. Decision trees are good algorithms for discovering the structure hidden behind data. And that is the essence of what I'm trying to figure out. The inputs of the features in the dataset cover all possible predictions and since two examples with the same input values would be guaranteed to have the same prediction, a decision tree would be ideal.

Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

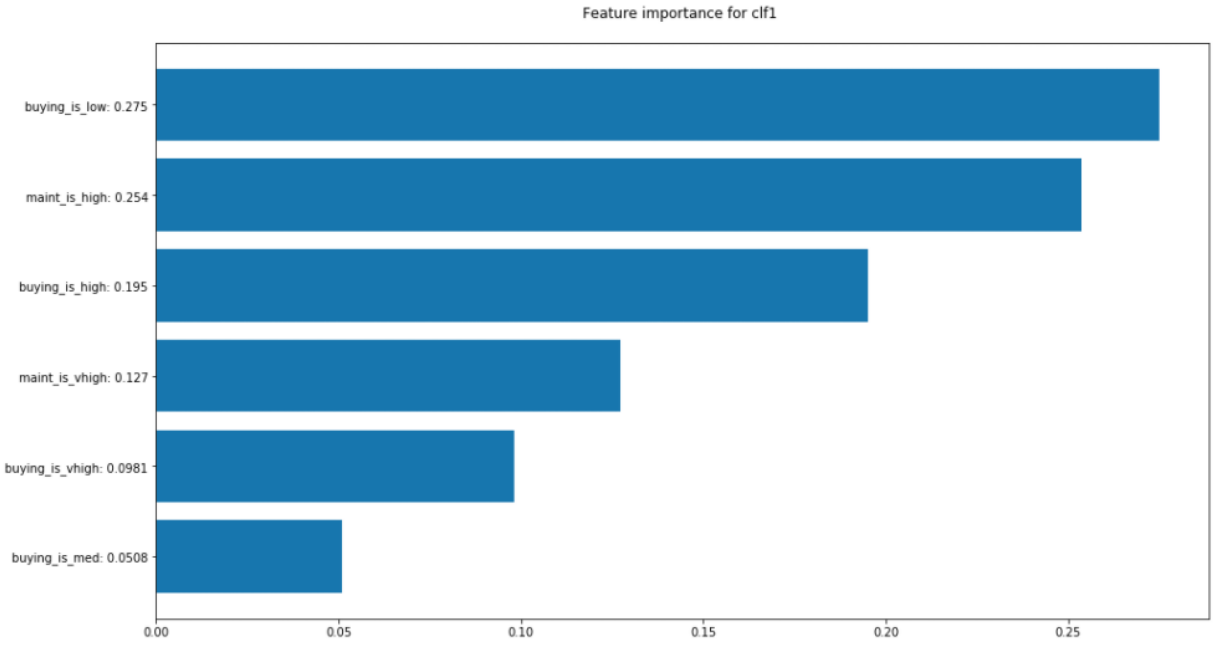
First, I split the data into training and testing data. I initialize a DecisionTreeClassifier and tune one of the hyperparameters, *max_depth*, to six as that is the number of features that go into the 'acceptability' target I am trying to dig deeper into.

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0

from sklearn import tree

max_depth = 6
clf1 = tree.DecisionTreeClassifier(max_depth=max_depth)
clf1 = clf1.fit(X_train, y_train)
print("Decision Tree accuracy: ", clf1.score(X_test, y_test) * 100)
```

```
Decision Tree accuracy:  92.2928709056
```

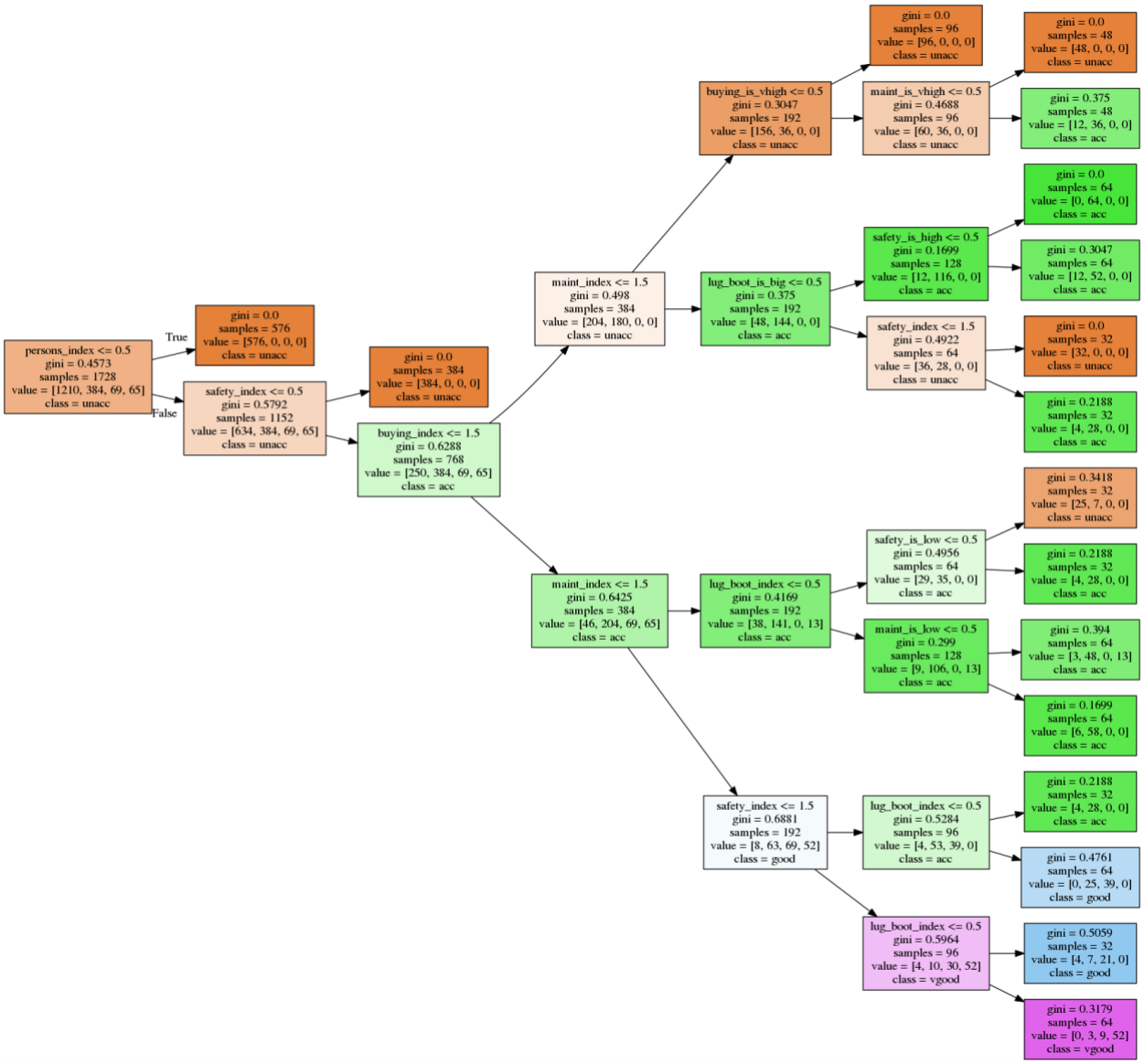
That's not a bad score for a basic DecisionTreeClassifier!

This was the first run of a basic decision tree and it scored a very high score of 97%. The algorithm worked very efficiently. The success of the tree tell me that the data holds some important information as to what drives the acceptability and unacceptability of cars. Here is a graph of the feature importance from the tree. The second classifier will surely have a different feature importance chart than the first one.



Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

And here is a graph of the tree itself.




(code for plotting the feature importance and tree can be found in the jupyter notebook)

Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

The graphs here show that the buying price of the car is the main driving force for the algorithm to predict the acceptability of the car. To get a more accurate score, I tuned the *max_depth* parameter to 'None', for full depth. After running the algorithm, an even higher accuracy was achieved.

```
# Increasing the max_depth

clf1 = tree.DecisionTreeClassifier(max_depth=None)
clf1 = clf1.fit(X_train, y_train)
print("Decision Tree accuracy: ", clf1.score(X_test, y_test) * 100)
```

```
Decision Tree accuracy:  97.6878612717
```

Even though that is a high accuracy, I decided to push a little further using a different classifier. A random forest classifier is still essentially the same thing but on a deeper scale. Decision trees, especially random forests, work exceptionally well on a dataset such as this one because it is very large and has many features that may or may not be significant. Random forests can ignore irrelevant attributes, thus, making the *Gain* = 0.

I initialize a RandomForestClassifier, fit the data, and ran a test and got a score of 97.4%. Unfortunately, it was lower than the basic decision tree score however I ran a few tests to minimize the error. Most machine learning algorithms perform very well when the data is scaled between the maximum and minimum. I scaled the data and got an accuracy higher than all the other previous decision tree classifiers instantiated before.

```
print("Random Forest accuracy: ", clf.score(X_test, y_test)*100)
```

```
Random Forest accuracy:  97.4951830443
```

```
print("Scaled Random Forest accuracy: ", clf.score(X_test_scaled,y_test)*100)
```

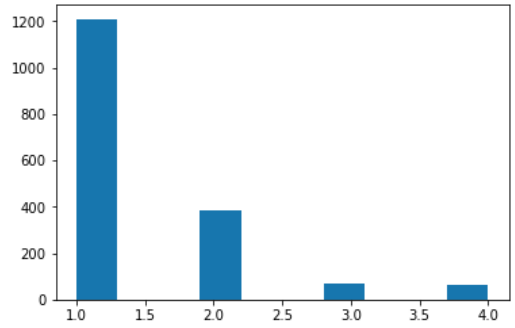
```
C:\Users\moemi\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning: Data with input dtype int32
was converted to float64 by MinMaxScaler.
  warnings.warn(msg, DataConversionWarning)
```

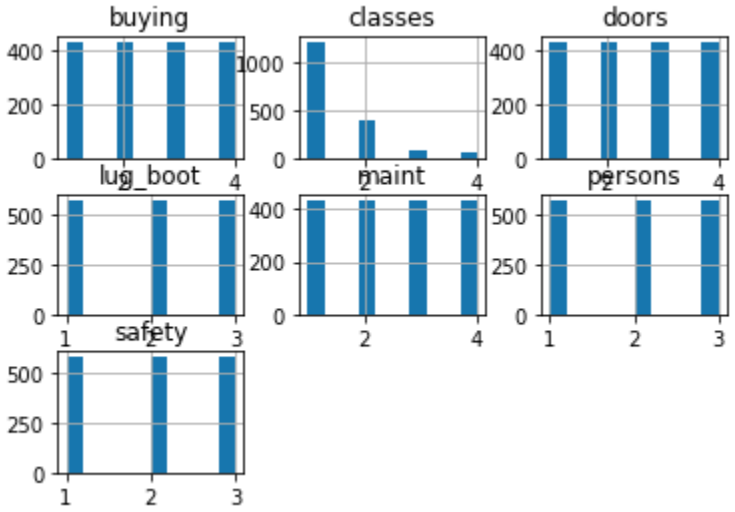
```
Scaled Random Forest accuracy:  97.880539499
```

Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

I managed to achieve good classification results using decision trees and could show those results visually and graphically. In this specific case, where two of the same input values in the training data can result in two different possible predictions, a tree cannot model the data set with 100% accuracy. A regression classifier would not be ideal in this situation since the targets are not classified in a defined binary.

However, even though there aren't only two targets, the inputs that lead to the targets relate to other instances that have the same target value. I wasn't exactly sure how the features of the instances stayed continuous through multiple instances. Here is a histogram of the classes and the features.



Mohamed Abdalla
Machine Learning
1 May 2020
Prof. Geena Kim

The top graph is a graph of the distribution of classes. The graph clearly shows that most of the instances are in the 'unacc'(unacceptable) output class. The graph below represents the distribution of the features to their classes. Each of the features seem to be evenly distributed, this implies that their multivariate interrelation is what causes the distribution of classes that we can see in the first graph.

The consistency of the features allows a classifier such as a KNeighbors classifier to be rather efficient. This is also the reason why I implemented this classifier algorithm into the data as well. I instantiated the class with the *n_neighbors* hyperparameter equal to ten. After fitting the data, the classifier reached a score of 91.9%. This was rather low. I was expecting a higher score and so I assumed that the *n_neighbors* was too high so I set it to 5 (the default value) and the classifier scored an accuracy of 93.6%.

```
score = accuracy_score(y_test, y_pred)
print("KNN accuracy: ", score*100)
```

```
KNN accuracy:  93.6416184971
```

GitHub: https://github.com/MoeMixMC/Car-Evaluation-ML