# Udacity Machine Learning Engineer Nanodegree program
# Final Project – Predicting heart disease

**Estevam Ribeiro do Valle Donnabella Santos**

**August 2019**

## I Definition

### Project overview

Heart diseases are a huge problem in our modern society. It accounts for one third of the total of death cause in the U.S., yet Google and media coverage is about only 2-3%. In order to make society more careful about this matter, we could use machine learning to predict whether people are prone to develop heart disease in the future and alert them in order to prevent deaths. In the last 30 years, a 303-sample dataset about heart disease has been used in many academic studies and it is available on Kaggle since September 2018.

The machine learning society is using this dataset to train their models and it was used used in the article "Heart Disease Prediction Using Machine learning and Data Mining Technique". The article discusses the use of decision trees in the matter using WEKA, a Java library for machine learning. Since the result was not satisfactory with an accuracy of 56.76%, the goal of this project is to reach a more stable result of at least 70% using machine learning.

### Problem statement

The goal is to create a prediction model that predicts with a minimum of 70% of accuracy whether a given person with determined attributes will develop a heart disease or not. The 14 attributes contained in the dataset are well defined and all measureable with a good degree of precision. Since the target variable is known, we will use supervised machine learning techniques to solve the problem. Since it is a categorical prediction problem, the supervised algorithms chose are the following:

1. Gradient Boosting Classifier
2. Random Forest Classifier
3. Logistic Regression
4. SVC

### Evaluation metrics

The target feature - 0 = no presence of heart disease; 1 = presence of heart disease - is very well balanced (138 for “no presence of heart disease“ and 165 for “presence of heart disease“), hence the evaluation metric is the accuracy score. As discussed in the section *II Analysis – data exploration*, the other features are not well balanced, so just in case, we will also explore the confusion matrix results.

## II Analysis

### Data and visualization exploration

In 1988 four institutes collected data from 303 people and made a dataset of 76 attributes about heart disease. The institutes are listed below

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

Historically, from the 76 attributes, only a subset of 14 are been used and considered relevant for hear disease prediction studies, thus this capstone project will use the 14 features subset as the dataset. The target attribute refers to either 0 (no heart disease presence) or 1 (heart disease presence). This dataset was cited in this article "Heart Disease Prediction Using Machine learning and Data Mining Technique" but was collected from Kaggle.

All the features are shown below

1. **Age (age):** numerical
2. **Sex (sex):** categorical – 0 = male and 1 = female
3. **Chest pain type (cp):** categorical – from 1 = low pain to 4 = very painful
4. **Resting blood pressure in mmHg (trestbps):** numerical
5. **Serum cholestoral in mg/dl (chol):** numerical
6. **Fasting blood sugar > 120 mg/dl (fbs):** categorical – 0 = False and 1 = True
7. **Resting electrocardiographic results (restecg):** categorical values 0, 1 and 2
8. **Maximum heart rate achieved (thalach):** numerical
9. **Exercise induced angina (exang):** categorical – 0 = no and 1 = yes
10. **ST depression induced by exercise relative to rest (Oldpeak):** numerical
11. **The slope of the peak exercise ST segment (slope):** categorical values 0, 1 and 2
12. **Number of major vessels colored by flourosopy (ca):** categorical values 0, 1, 2, 3 and 4
13. **Thal:** categorical 1 = normal, 2 = fixed defect and 3 = reversible defect

14. **Target:** categorical 0 = no presence of heart disease and 1 = presence of heart disease

The table 1 shown below is just a glimpse on the first 10 entries of the dataset.

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |
| **5** | 57 | 1 | 0 | 140 | 192 | 0 | 1 | 148 | 0 | 0.4 | 1 | 0 | 1 | 1 |
| **6** | 56 | 0 | 1 | 140 | 294 | 0 | 0 | 153 | 0 | 1.3 | 1 | 0 | 2 | 1 |
| **7** | 44 | 1 | 1 | 120 | 263 | 0 | 1 | 173 | 0 | 0.0 | 2 | 0 | 3 | 1 |
| **8** | 52 | 1 | 2 | 172 | 199 | 1 | 1 | 162 | 0 | 0.5 | 2 | 0 | 3 | 1 |
| **9** | 57 | 1 | 2 | 150 | 168 | 0 | 1 | 174 | 0 | 1.6 | 2 | 0 | 2 | 1 |

Table 1 – A glimpse in the dataset

There are no missing values in the entire dataset. There are some outliers according to the 1.5*IQR rule and this will be addressed in the methodology section.

The image below shows the distribution of the numerical features and the scatter plot.

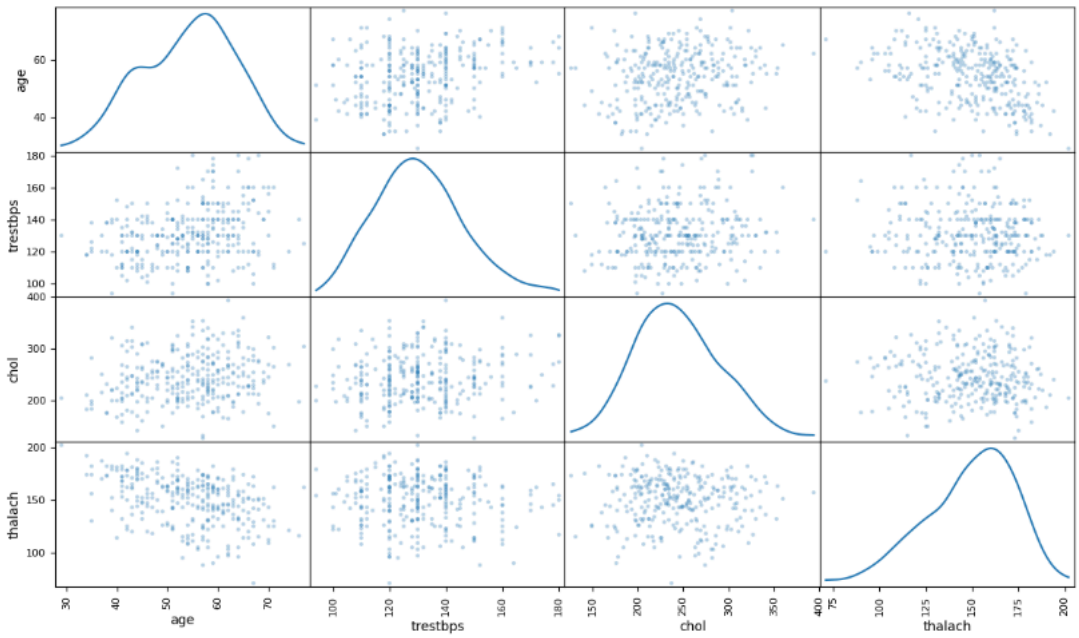



Figure 1 – Distribution and scatter plot of numerical features

It is possible to see that the distributions are skewed either to the left or to the right. Also, although not very clear, it is possible to see a correlation – or at least a trend -

between “age” and “chol” and between “age” and “thalach” looking at the scatter plot. For the nine others categorical data, the distribution is also provided.

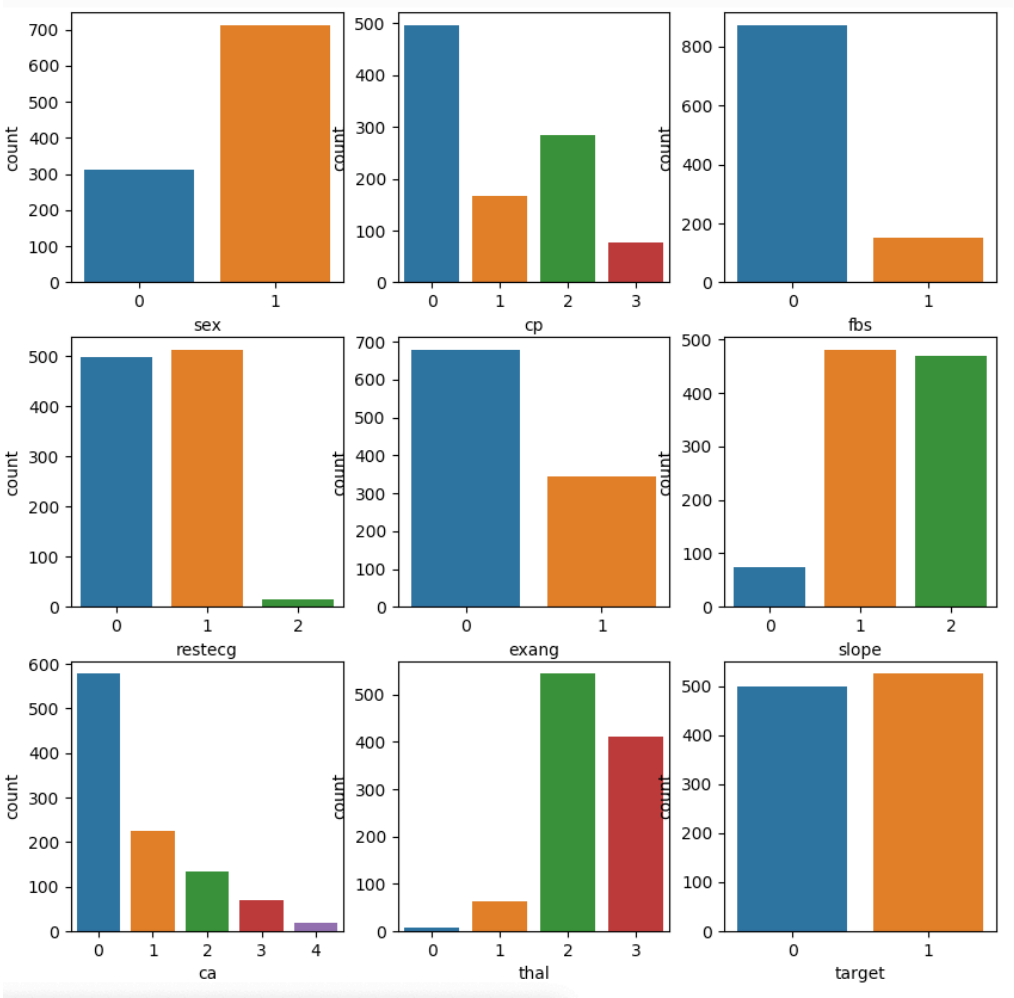



Figure 2 – Distribution of categorical features

As stated before, the target variable is well balanced. Even though the target variable will be not a problem, it would be nice to switch from accuracy to confusion matrix to see the performance of the model in others aspects. Since the goal is to determine if someone is prone to have a heart disease, it is more important for our model to be more accurate about “having the disease” than to “be health”.

This dataset is being used for the last 30 years in studies and although the features are imbalanced, it may represent well the population in a statistic point of view. That said, the model may not present problems like the Amazon’s known case, where their model to chose resumes was flaw because the features about people was imbalanced – and not statistically representative, preventing the model to suggest a “black man” or a “white woman” as a good candidate and only suggesting “white man” as good candidate.

## Algorithms and techniques

The dataset has a target variable, so the solution is to use different supervised machine learning techniques and compare them to see which one would be more accurate to solve the problem. After determining the fine-tuned supervised method, the trained model will be run against the test dataset for validation.

Since it is a categorical prediction problem, the chose supervised algorithms and their short description are:

1. **Gradient Boosting Classifier:** The word Gradient comes from the "gradient descent", a technique that looks for a minimum of a function. The word Boosting is a technique to produce prediction models using an ensemble of weak prediction models. Therefore, this algorithm uses a mix of techniques. The overall goal is to minimize the mean squared error.
2. **Random Forest Classifier:** It combines the result of many individual decision trees and retrieve the mean or mode.
3. **Logistic Regression:** It uses a logistic function to create a filter and segregate values of a dataset into 0s and 1s, accordingly to the weights that the features of the dataset represent to the logistic function.
4. **SVC:** It is a Support-Vector Machine algorithm used for classification and regression problems. The use of the *kernel trick* makes possible the usage of a linear regression method to learn a nonlinear function or decision boundary.

All these algorithms are contained in the Sklearn Python library as well as the metrics validation as accuracy, f1 score and confusion matrix.

In order to fine-tune the algorithms parameters, we will use a technique called Grid search, where some parameter values are passed in and it iterates these values using the algorithm. In other words, it would be like running the same algorithm over and over again, but in each time changing just one parameter to see how it changes the performance. This is a good way to find the optimal parameters for an algorithm given a specific model.

### Benchmark model

The article "Heart Disease Prediction Using Machine learning and Data Mining Technique" reach a best result of 56.76% accuracy using Decision Tree with algorithm J48 with Reduced Error Pruning. This model is going to be used as the Benchmark model, since it is a categorical prediction model using the same dataset.

## III Methodology

### Data preprocessing

Processing the data does not mean that the end result of the machine learning will be better. That said, although some processing techniques will be applied, the machine learning processes will also handle the original dataset for comparison purposes.

The first step is to look for missing values. Fortunately, there is none in this dataset nor there is any mismatching type or similar problem.

The second step is to look for outliers. Applying the standard 1.5*IQR rule, we find out that 4.95% of the value fit this rule. This number is not to ignore. Let us tighten this rule making the threshold 2.0*IQR. Now the real outliers represent only 1.98% of the whole dataset, reducing the total number from 303 to 297 samples.

In figure 1 of the *II Analysis* section we saw that the distributions were either right or left skewed. Applying the natural log transformation, we get a slightly better curve as shown in the figure 3 below.

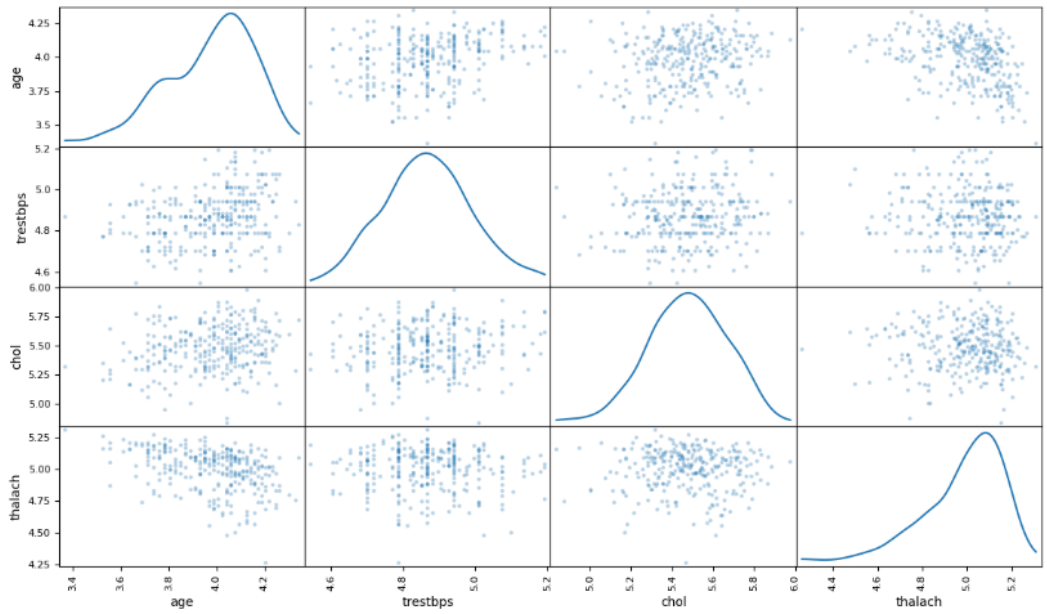



Figure 3 – Distribution and scatter plot of numerical features after log transformation

## Implementation

In order to reproduce this study, some code snippet will be provided. We used Python 3.7 during all this work.

First, the imports needed:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble.gradient_boosting import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
```

The first step is to split the dataset in two parts, one for training the model and other to validate it. For that, the *train_test_split* function of Sklearn Python library will be used. In order to others be able to reproduce the results here presented, this *train_test_split* function will be used with a *random_state = 42* parameter, guaranteeing that everyone running the same algorithm with the same parameters will obtain the same results. The rate used here is 70% of the sample for model training purpose and 30% of the sample for testing purpose.

```
y = dataset['target'].values
X = dataset.drop(['target'], axis=1).values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

These training and test datasets will be running against 4 supervised machine learning algorithms using the Search Grid function. This methodology will be applied within 3 different datasets in that order:

- Original dataset
- Dateset with outliers removed (using the 2.0*IQR rule)
- Dataset with outliers removed and log transformation applied

The table below summarizes all the parameters and parameters values of grid search used in this work. The same algorithms, the same parameters and same parameters values are going to be used against all the 3 datasets listed above.

| Algorithm | Parameter | Value |
|---|---|---|
| Gradient Boosting Classifier | min_samples_leaf | 1 |
| | | 5 |
| | | 10 |
| | | 20 |
| | n_estimators | 20 |
| | | 50 |
| | | 100 |
| | | 150 |
| Random Forest Classifier | min_samples_leaf | 1 |
| | | 5 |
| | | 10 |
| | | 20 |
| | n_estimators | 20 |
| | | 50 |
| | | 100 |
| | | 150 |
| Logistic Regression | penalty | L1 |
| | | L2 |
| | C | 0.1 |
| | | 0.5 |
| | | 1.0 |
| | | 5.0 |
| SVC | kernel | Linear |
| | | Rbf |
| | | Sigmoid |

| | | |
|---|---|---|
| | C | 0.5 |
| | | 1.0 |

Table 2 – Algorithms, parameters and parameters values to be used in the analysis

Please refer to the code snippet below to see how grid search and training the model works in Python:

```
parameters = {'min_samples_leaf':(1, 5, 10, 20), 'n_estimators':[20, 50, 100, 150]}
model = GradientBoostingClassifier()
GBC_clf = GridSearchCV(model, parameters, cv=5)
GBC_clf.fit(X_train, y_train)
pred = GBC_clf.predict(X_test)
GBC_accuracy = np.mean(pred == y_test)
GBC_f1 = f1_score(y_test, pred)

print("Accuracy: {:0.2f}".format(GBC_accuracy))
print(classification_report(y_test, pred))
```

Taking the Gradient Boosting Classifier as an example, first we chose all the parameters that will be tested as stated in the table 2. Then we define the model as being the GradientBoostingClassifier() imported from sklearn previously. Then we apply the GridSearchCV() method passing our model and our parameters. In the next step we train (fit) the model passing the features of the dataset (X_train) and the target variables (y_train). Finally, we test our model using the X_test samples previously segregated from the original dataset to get our labeled prediction array. The final step is to compare what we get to what we expect (y_test) using the desired score, in our case the accuracy and the confusion matrix.

In order to run the others algorithms, please follow the same steps changing the model variable to the other classifiers methods as: RandomForestClassifier(), LogisticRegression() and SVC().

# IV Results

## Model evaluation e validation

The table 3, 4 and 5 below summarizes the best result of each algorithm, which parameters led to that result and which dataset it came from.

| Dataset: Original | | | | | | |
|---|---|---|---|---|---|---|
| **Algorithm** | **Parameters** | **Final result** | | | | |
| | | **Target Value** | **Precision** | **Recall** | **F1 score** | **Accuracy** |
| Gradient Boosting Classifier | min_samples_leaf = 1<br>n_estimators = 100 | 0 | 0.84 | 0.78 | 0.81 | 0.84 |
| | | 1 | 0.83 | 0.88 | 0.85 | |
| Random Forest Classifier | min_samples_leaf = 1<br>n_estimators = 10 | 0 | 0.84 | 0.78 | 0.81 | 0.84 |
| | | 1 | 0.83 | 0.88 | 0.85 | |
| Logistic Regression | Penalty = L2<br>C = 1.0 | 0 | 0.80 | 0.78 | 0.79 | 0.81 |
| | | 1 | 0.82 | 0.84 | 0.83 | |
| SVC | Kernel = Rbf<br>C = 1.0 | 0 | 0.80 | 0.80 | 0.80 | 0.82 |
| | | 1 | 0.84 | 0.84 | 0.84 | |

Table 3 – Best results per algorithm for Original dataset

| Dataset: Outliers removed | | | | | | |
|---|---|---|---|---|---|---|
| **Algorithm** | **Parameters** | **Final result** | | | | |
| | | **Target Value** | **Precision** | **Recall** | **F1 score** | **Accuracy** |
| Gradient Boosting Classifier | min_samples_leaf = 1<br>n_estimators = 100 | 0 | 0.76 | 0.74 | 0.75 | 0.79 |
| | | 1 | 0.81 | 0.83 | 0.82 | |
| Random Forest Classifier | min_samples_leaf = 1<br>n_estimators = 10 | 0 | 0.79 | 0.71 | 0.75 | 0.80 |
| | | 1 | 0.80 | 0.87 | 0.83 | |
| Logistic Regression | Penalty = L2<br>C = 1.0 | 0 | 0.87 | 0.71 | 0.78 | 0.83 |
| | | 1 | 0.81 | 0.92 | 0.86 | |
| SVC | Kernel = Rbf<br>C = 1.0 | 0 | 0.84 | 0.71 | 0.77 | 0.82 |
| | | 1 | 0.81 | 0.90 | 0.85 | |

Table 4 – Best results per algorithm for dataset with outliers removed

| Dataset: Outliers removed and log transformation applied | | | | | | |
|---|---|---|---|---|---|---|
| **Algorithm** | **Parameters** | **Final result** | | | | |
| | | **Target Value** | **Precision** | **Recall** | **F1 score** | **Accuracy** |
| Gradient Boosting Classifier | min_samples_leaf = 1<br>n_estimators = 100 | 0 | 0.78 | 0.74 | 0.76 | 0.80 |
| | | 1 | 0.81 | 0.85 | 0.83 | |
| Random Forest Classifier | min_samples_leaf = 1<br>n_estimators = 10 | 0 | 0.73 | 0.84 | 0.78 | 0.80 |
| | | 1 | 0.87 | 0.77 | 0.82 | |
| Logistic Regression | Penalty = L2<br>C = 1.0 | 0 | 0.81 | 0.68 | 0.74 | 0.80 |
| | | 1 | 0.79 | 0.88 | 0.84 | |
| SVC | Kernel = Rbf<br>C = 1.0 | 0 | 0.88 | 0.74 | 0.80 | 0.84 |
| | | 1 | 0.83 | 0.92 | 0.87 | |

Table 5 – Best results per algorithm for dataset with outliers removed and log transformation applied

For the original dataset, the best result was obtained using the either the Gradient Boosting Classifier or the Random Forest Classifier, both resulting in a F1 score of 0.81 for predicting health people and 0.85 for predicting people with heart disease and an overall accuracy of 0.84.

Considering the dataset with the outliers removed, the best algorithm is the Logistic Regression resulting in a F1 score of 0.78 for predicting health people and 0.86 for predicting people with heart disease and an overall accuracy of 0.83.

Considering the dataset with the outliers removed and the log transformation applied, the best algorithm is the SVC resulting in a F1 score of 0.80 for predicting health people and 0.87 for predicting people with heart disease and an overall accuracy of 0.84.

That said, the best alternative is considering the outlier removal, log transformation and applying the SVC algorithm which give us the 87% accuracy in predicting that someone has heart disease.

It is also possible to see that all the results in all three conditions lad to good results. This shows that changes in the dataset does not seem to greatly change the result.

### Justification

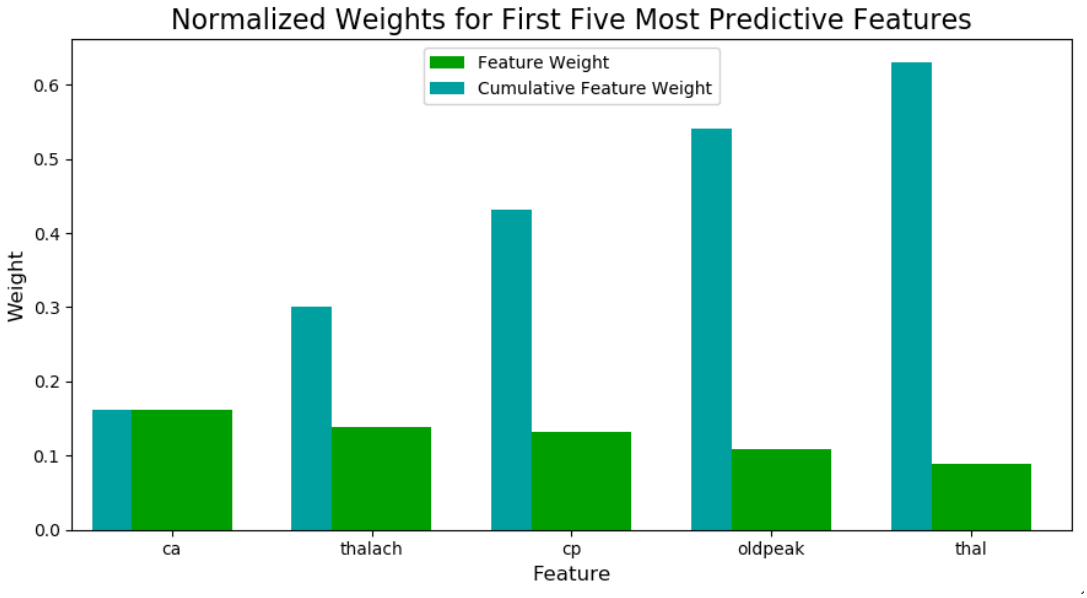
Comparing to the benchmarking model, our model outstands. Even our Random Forest algorithm did way better than the benchmarking model, which is a Decision Tree with algorithm J48 with Reduced Error Pruning resulting in 56.76% accuracy.

The authors of the benchmarking paper didn't preprocessing the data, but the interesting point is that even our untreated data, i.e. the original dataset, performed way better with a accuracy of 81% to 84%.

## V Conclusion

### Visualization

Although the model that best performed was the SVC with Rbf kernel, we will analyze a feature importance using the second best case, which is the Random Forest Classifier considering the original dataset. That is because the SVC with a non-linear kernel transforms the data to another space, which are not really related to the original input.




It is interesting to see that there is not a major feature much more important than the others. Actually, ca, thalach and cp are very close to each other in importance. At least it is possible to see that the most five predictive features sums up to more than 60% of the total weight, being the other 8 features less than 40% of the importance.

### Reflection

Let us recap the step by step of this work in order to highlight the main points learned during the process of the project. First of all, some research in Kaggle.com

for a good dataset was needed. Although doing a deep learning project was temping, this health segment dataset looked very attractive due to the importance of the matter.

The first step after choosing the dataset was to do a few analysis to be sure that the dataset could really be used. This is really a real-life situation, since usually in a machine-learning course all dataset provided is surely workable on. Finding out how clean and complete was the dataset and how serious were the sources was a challenge.

The second step was to dive into more details in order to write the capstone proposal. Defining the metrics score, defining the goal, finding a good benchmark study and so on. It was a big deal to agree to make a work that results in a machine-learning model better than a formal academic study. Maybe this was the most difficult part for me.

Therefore, I decided to use at least two algorithms surely more modern than the simple decision tree used in the benchmark case: Gradient Boost Classifier and Random Forest Classifier.

I was more relieved after seen a better result using the algorithms in the dataset before preprocessing. I was convinced that after the preprocessing part at least a slightly better result would be achieved.

The second most difficult part was trying to make sense of the highly imbalanced features of the dataset. After some research and some time to think better on that matter, I concluded that it was not a problem in this case. That is because as stated before, the imbalanced features really represent imbalanced characteristics of the population. That said, our model was trained considering, for example, *just some people with severe chest pain (cp = 4)* and in real-life just a few people really have severe chest pain. Again, that is a different scenario from the Amazon case previously mentioned in this work, where “white man“ was highly present in the training dataset as a good candidate, while in the real population, half of the people are women and a considerable amount of this same population is black.

After that, preprocessing the data, training the model and analyzing the results were not a big task since the foundations were well built during the nanodegree program.

## Improvement

Further studies can be done with unsupervised learning clusterization algorithms to try to determine groups with similar characteristics and split the original dataset in smaller datasets and then training supervised models against them. This can be good to improve the accuracy of the F1 score and can define a real life tool where the subject is first filtered accordingly to the cluster group and then analyzed by the supervised model to get a more trustworthy result.

It would be also interesting to use this dataset in neural networks algorithms to compare performance.