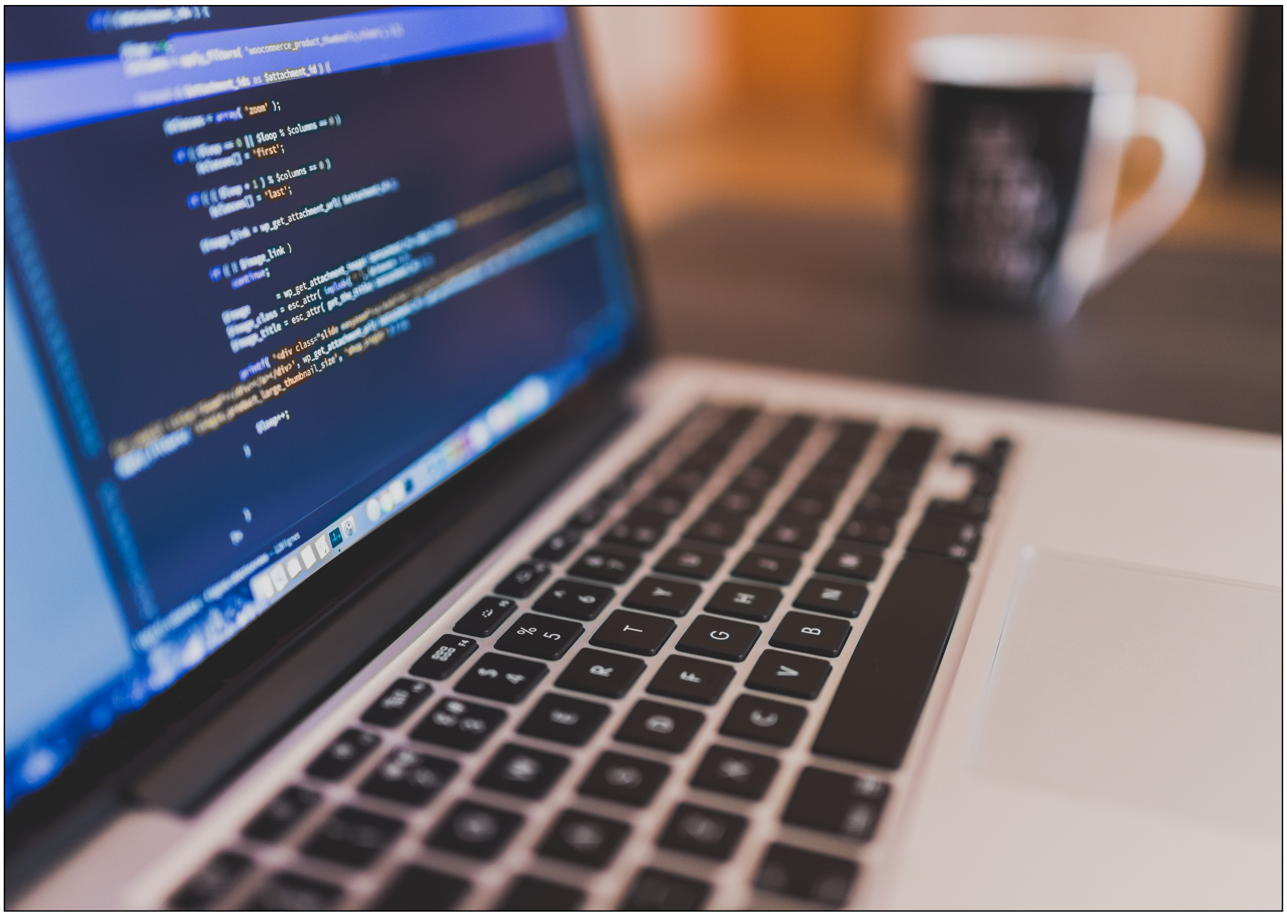
Capstone Project - Predicting Job Chances after finishing a Coding Bootcamp

Machine Learning Engineer Nanodegree

Janna Brettingen - August 1st 2016



I. Definition

Project Overview

Technology is disrupting the way people live, communicate and drive. But it doesn't stop there. Schools and Universities still try to teach the traditional way, but quite some companies started teach coding online. A main difference to the traditional approach is, that it is less passive listening and more active doing. It is possible to learn online for free or for a fraction of the usual costs.

This project was inspired by the "2016 New Coder Survey" dataset from FreeCodeCamp and CodeNewbie.org published on kaggle.com. In this survey, more than 15,000 people actively learning to code answered a variety of questions including if they participated in a bootcamp and if they found a developer job afterwards.

Problem Statement

The goal is to predict for a bootcamp participant if he is more likely to get a job after finishing the bootcamp or not. This could enable bootcamp providers to help participants with their future job success early on.

The tasks involved are the following:

- Preprocess "2016 New Coder Survey" dataset
- Implement Training and Testing Data Split
- Train a classifier to predict a bootcamp participants job chances
- Tweak parameters of the classifier

Metrics

Predicting job chances for bootcamp participants is a binary classification problem. To measure the performance of my classifier, I use the F_1 score. It takes precision and recall into account and is calculated as follows:

$$F_1 = 2 \cdot \text{precision} \cdot \text{recall} \ / \ (\text{precision} + \text{recall})$$

Precision is the number of correct positive results divided by the number of all positive results and recall is the number of correct positive results decided by the number of positive results that should have been returned.

II. Analysis

Data Exploration

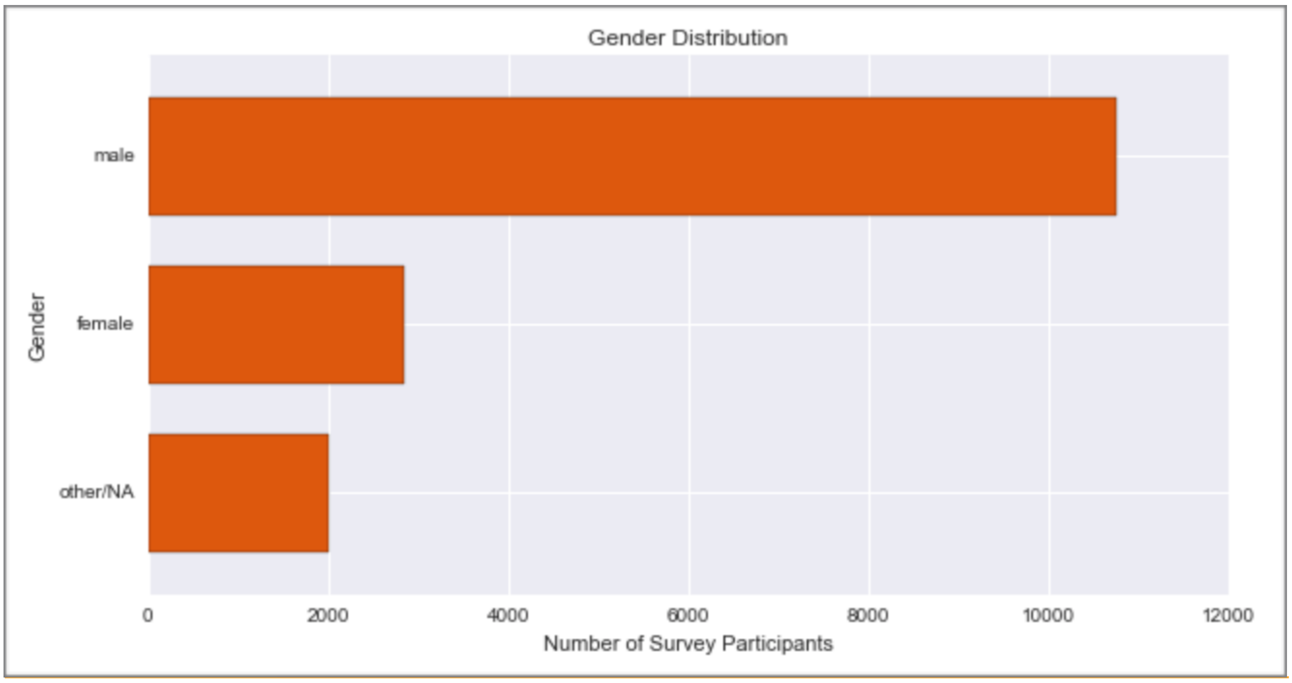
The “2016 New Coder Survey” dataset includes 15,620 data points with 113 features. To train a classifier predicting job chances for bootcamp participants only a fraction of the data points is needed. Looking only at people who finished a bootcamp reduces the data points to 635. Taking the answer to the question if a participant got a job after finishing the bootcamp as my target column, I reduced the features to 112. This is 112 questions each participant answered. These questions range from age, gender, if the person has children over resources they used like a variety of podcasts, resources like Coursera or Udacity and StackOverflow, if and which coding events they participated in to if they are employed, have student debt and the hours they spent learning. The whole list of features can be found in the Jupiter Notebook new_coder_survey under the heading “Preparing the Data”.

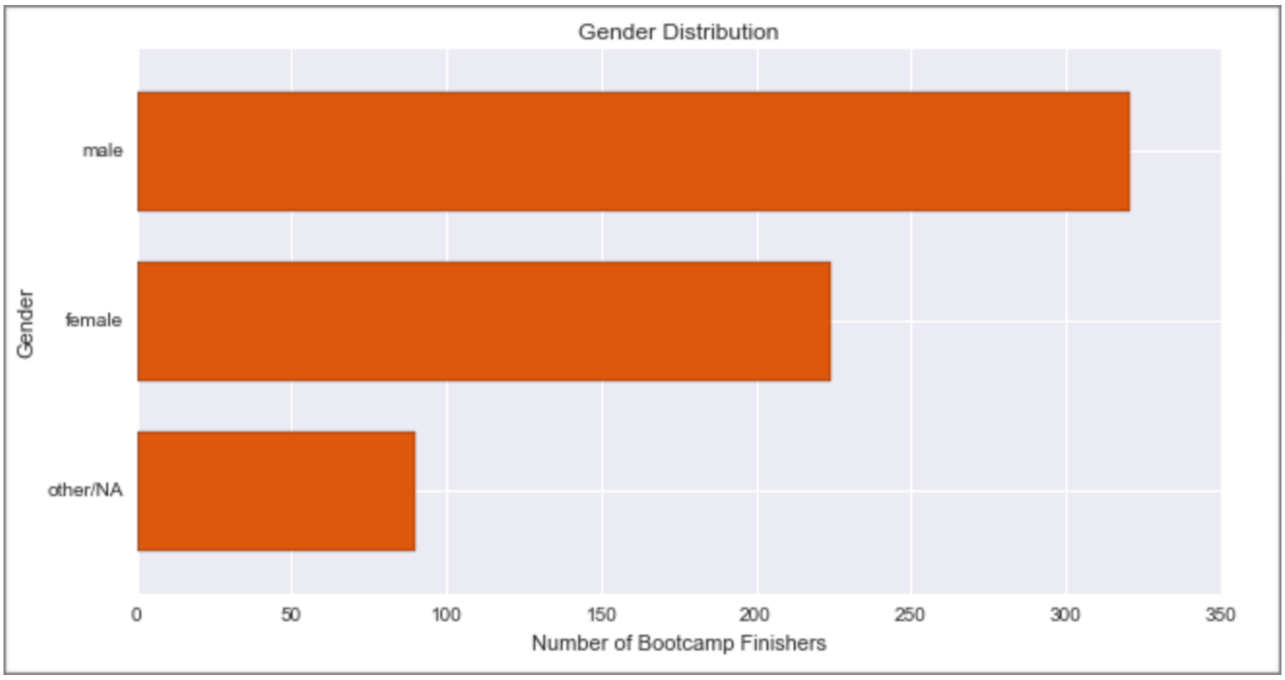
Of these 635 participants, 371 managed to get a developer job afterwards and 264 did not. This leads to a job success rate of 58.43 %.
The mean age of bootcamp participants who finished a bootcamp is 27 years. 34.4 % paid a bootcamp loan.

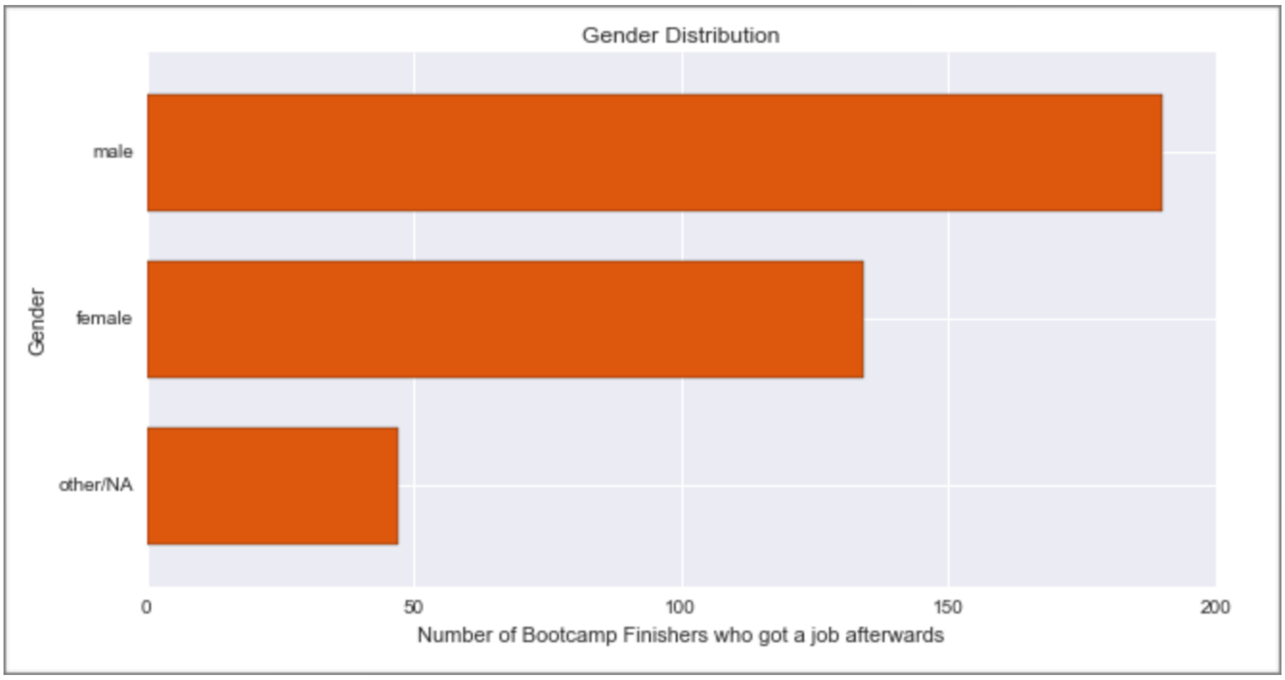
60.47 % of bootcamp finishers are already working in Software Development. This will probably have a great influence on job chances.

Exploratory Visualizations

Let us take a look at the gender distribution of all survey participants,, bootcamp finishers and those, who managed to get a job after finishing the bootcamp.







Interestingly, the majority of survey participants is male, but if we look at bootcamp finishers, there is a higher percentage of women. The gender distribution of bootcamp finishers and those who get a job after finishing a bootcamp is almost similar.

Algorithms and Techniques

The algorithm I chose is a Random Forest Classifier. This is an ensemble learning method for classification and does this by constructing a multitude of Decision Trees. Decision Trees are easy to use and easy to understand graphically. They are prone to overfitting, so that fine tuning the parameters is very important. They are relatively fast with large datasets.

I will use Grid Search to fine tune the parameter `min_samples_split` after using the default value 2.

Benchmark

I will use the F_1 score to evaluate the predictions. My goal is to be able to predict if a bootcamp participant is likely to get a job after the bootcamp with an F_1 score of 0.85 or above.

It is probably not possible, to get a F_1 score much higher than 0.9, because there are several factors in the process of getting a job like how a cover letter is written, how the person presents itself, how well he or she is connected and more which will have an influence on whether or whether not a candidate will get a job.

A F_1 score of 0.85 or above would be a nice start to estimate job chances.

III.Methodology

Data Preprocessing

At first, I reduced the dataset to only coding learners who finished a bootcamp. This reduced the size from 15,620 to 635. After this step, it was possible to remove the columns:

- AttendedBootcamp
- BootcampFinish

In the next step, I had to remove all columns related to the end of the bootcamp, because the classifier should be able to predict if a student will be successful in finding a new job during a bootcamp. The information how many month ago the bootcamp was finished does not necessarily give information about job chances, but is only available for those who finished the bootcamp already. I removed the following columns:

- BootcampMonthsAgo
- BootcampPostSalary
- BootcampRecommend

I also removed the following columns, because their values are unique to each individual person and are not needed to make predictions:

- ID.x
- ID.y
- NetworkID
- Part1EndTime
- Part1StartTime
- Part2EndTime
- Part2StartTime

The next step in cleaning the data was to make every column numerical. This is important for most Machine Learning Algorithms. There are several kinds of non-numerical values.

In the present dataset were the following:

- heavy use of `NaN` in yes/no kind of features
- use of 'NaN' in otherwise numerical kind of features
- use of 'NaN' and strings in non-numerical kind of features

I tested some numerical features like `HasChildren` and found, that the majority of ‘NaN’ should be replaced with 0. Then I replaced all occurrences of ‘NaN’ with zeros.

This made yes/no kind of features binary features with the values 1 and 0. It also solved the issues with the second category, the otherwise numerical features.

For the third category, the strings and non-numerical kind of features I used a common technique. These features are called categorical variables. The creation of as many columns as possible values and filling these with binary values was the approach I chose.

Implementation

After preprocessing the dataset, the data got split into training and testing data. I used Cross Validation and kept 25% of the data as testing data. This made a total of 476 training samples and 159 testing samples.

By using 25 % in a separate testing dataset without training the classifier using the data, I ensured that the classifier does not overfit too much.

I used a Random Forest Classifier trained with the preprocessed data to make predictions for the testing samples and calculated the F_1 score.

These are the 10 most important features used by the Random Forest Classifier in the format (importance, feature name) in order:

1. (0.1156, ‘IsSoftwareDev’)
2. (0.0985, ‘JobPref_0')
3. (0.0843, ‘JobWherePref_0')
4. (0.0716, ‘ExpectedEarning')
5. (0.0658, 'JobWherePref_in an office with other developers’)
6. (0.0637, 'EmploymentField_software development and IT’)
7. (0.0446, 'EmploymentStatus_Employed for wages’)
8. (0.0316, ‘CommuteTime')
9. (0.0287, "JobApplyWhen_I'm already applying”)
10. (0.0226, 'JobWherePref_no preference’)

The zeros in this list are an equivalent to the answer “None”. The most important feature the Random Forest Classifier identified is if a person is already working as a Software Developer. I’ll talk about this point in the “Refinement” section.

Refinement

In my initial solution, I didn't fine-tune any parameters. The F_1 score for the training data was 0.9910 and for the testing data 0.8947.

Looking at the mean value of the feature `IsSoftwareDev` of every person who got a job after finishing a bootcamp, the result was 0.92. This is way too high and it seems that the feature `IsSoftwareDev` does mean the person is currently working as software developer. Therefore, I had to drop this feature.

This lead to a F_1 score for the training data of 0.9873 and for the testing data 0.8808.

Then, I fine-tuned the parameter `min_samples_split` using Grid Search. Using the best estimator for this parameter, I got a F_1 score for the training data of 0.9018 and for the testing data of 0.8832.

These are relatively high F_1 scores and I didn't use any other fine-tuning techniques.

The final F_1 score is actually lower than my initial F_1 score. This is due to the fact, that I suspect the feature `IsSoftwareDev` reveals if someone got a job as a software developer. Naturally, removing this feature lowers the F_1 score.

The new 10 most important features used by the Random Forest Classifier in the format (importance, feature name) in order are:

1. (0.1462, 'JobRoleInterest_0')
2. (0.0891, 'JobPref_0')
3. (0.086, 'ExpectedEarning')
4. (0.0725, 'CommuteTime')
5. (0.0559, "JobApplyWhen_I'm already applying")
6. (0.0554, 'JobWherePref_0')
7. (0.0523, 'HoursLearning')
8. (0.0518, 'EmploymentField_software development and IT')
9. (0.0508, 'EmploymentStatus_Not working but looking for work')
10. (0.0448, 'JobPref_work for a medium-sized company')

The zeros in this list are an equivalent to the answer "None". The most important feature the Random Forest Classifier identified is if a person answered the question "Job Role Interest" with "None".

IV.Results

Model Evaluation and Validation

During the whole process of choosing a Machine Learning Algorithm to fine-tuning it, I used the F_1 score on the training dataset and a separate testing dataset including 25 % of the original data to be able to validate the results.

The Random Forest Classifier gets a higher F_1 score on the training dataset naturally, because as it is a combination of Decision Trees, it's prone to overfitting. Using GridSearch to fine-tune parameters helped increasing the F_1 score on the testing dataset and decreasing it on the training dataset. This made it possible to get a F_1 score on the testing dataset of 0.8832 in the end.

Justification

The benchmark set for this problem was a F_1 score of 0.85 or above. This criteria is met with the final F_1 score of 0.8832.

The model was able to detect the most important features for predicting if a bootcamp participant is likely to get a job afterwards or not.

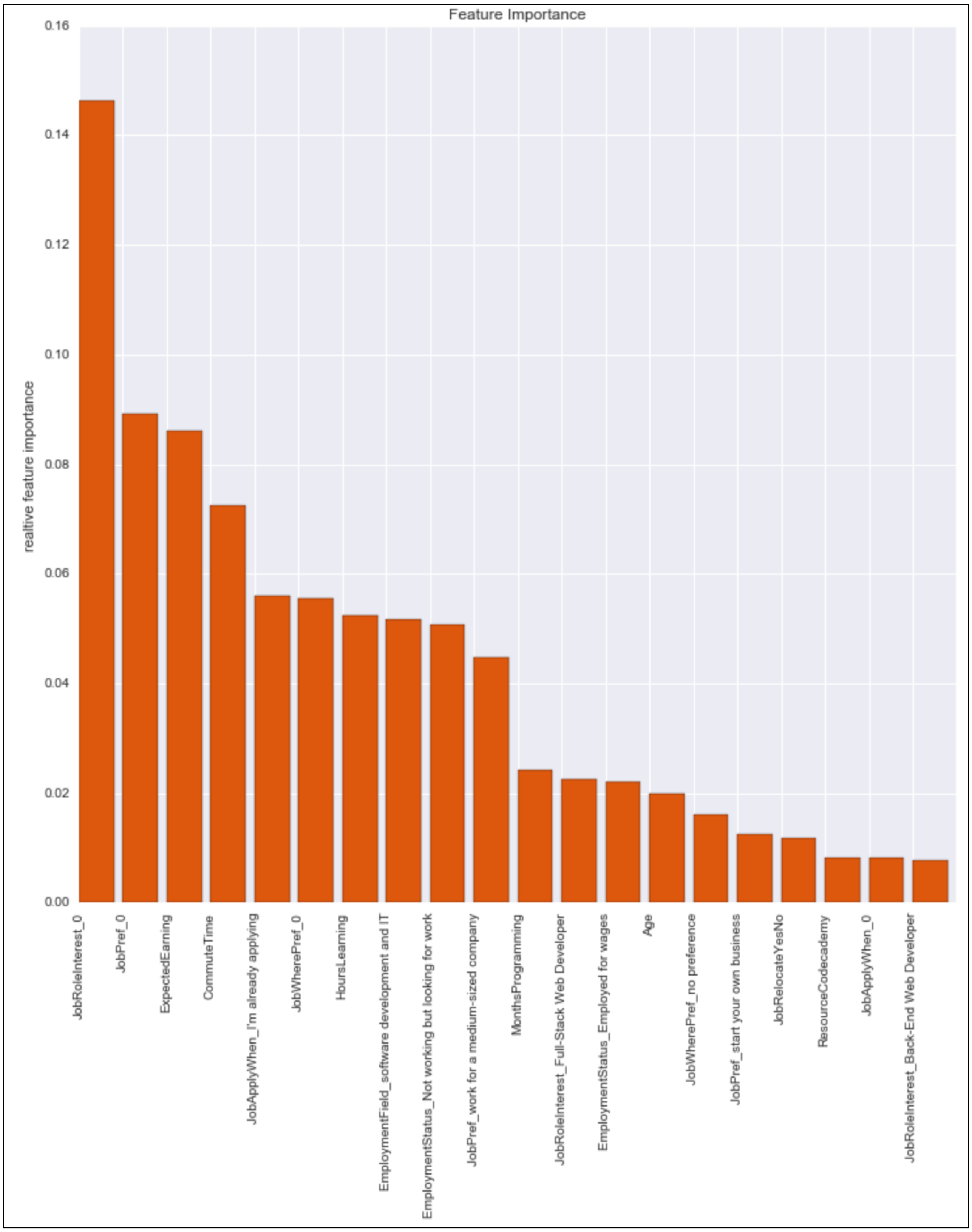
In the "Improvements" section, I discussed some ideas how to further improve the model and make use of it. This might lead to getting rid of a lot of features and change the predictions. For the question I tried to answer with this project, these features do not have to be removed.

The final solution is significant enough to make predictions on job success after finishing a bootcamp. But the journey doesn't end here.

V. Conclusion

Free-Form Visualization

This is a visualization of the most important features the Random Forest Classifier found.



Reflection

The process used for this project can be summarized using the following steps:

- Found an interesting dataset and developed a question
- Downloaded and preprocessed data
- Created a Benchmark for the classifier
- Split the data into training and testing datasets
- Trained a Decision Tree Classifier using the training dataset
- Used F_1 score to evaluate predictions
- Changed the algorithm to use a Random Forest Classifier instead
- Used F_1 score to evaluate predictions
- Fine-tuned 'min_samples_split' parameter
- Split the dataset into participants already working as software developers and those who aren't
- Repeated training and evaluating Random Forest Classifier on both datasets

Improvement

There are several ways to improve the predictions. Here are some ideas:

- larger dataset, more surveys
- quantify participation in events
- reduce features

These ideas could help in making predictions more general.

One issue is, that the most important feature the Random Forest Classifier identified first is whether someone is already working as a software developer or not. I am not completely sure if that refers to the job at the time of the survey or before taking the bootcamp. If it is at the time of the survey, the feature should be dropped. That's what I did, because the mean value of people who got a job after the bootcamp for this feature was 0.92. If it refers to before taking the bootcamp, it might be relevant. It could be interesting to only look at bootcamp participants who weren't working in software development before, but if the feature refers to the job at the time of the survey, it would not be possible to drop every person who's `isSoftwareDev` feature is true.

Another idea is to try to find a number of features like for example 10 which predict the chances of getting a job after a bootcamp best. These features should be features a participant has control over. Using those 10 features like for example 'HoursLearning' or 'ResourceStackOverflow' might help participants to increase their odds to get a job.