**Code/Classifier: Decision Tree**

* [**https://github.com/sammyrod/Enron\_ML**](https://github.com/sammyrod/Enron_ML)

**Written Documentation**

**Enron Submission Free-Response Questions**

1. **Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?  [relevant rubric items: “data exploration”, “outlier investigation”]**

The goal of the project is to statistically determine who is a point of interest in the Enron scandal that shocked the nation. It was one of the largest companies in the United States in 2000, and by 2002 it collapsed into bankruptcy. The dataset included are data from the Federal Bureau of Investigation made public. It includes features such as bonuses, salaries, emails, etc. of top executives. We are training data using a machine learning model to predict a point of interest given certain selected predictor features on test data.

There were several outliers in the dataset. The total of quantitative data was included, which obviously was detected as an outlier. The removal of this outlier is vital for this analysis as it does not provide any benefit. Moreover, there were other outliers. Some of them presented along with the total and more presented after the removal of the total. However, these remaining outliers are potential points of interest in the case and therefore beneficial for the analysis.

Before removing the outlier, there were 146 data points in the set. However, 1 point was just the grand total. After the removal of it, there are 145 people in the dataset. Only 18 out of 145 people are actual points of interest, which is a 12.41%. There are 145 data points per feature although many with missing values. The feature with the most missing values is loan advances with 97.93% of values missing. The feature with the lowest values missing (disregarding poi) is total stock value with 13.79% of values missing. Point of interest (poi) have not missing values.

1. **What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.  [relevant rubric items: “create new features”, “intelligently select features”, “properly scale features”]**

**Feature Selection:**

The final features were selected algorithmically only if the appended feature added value to the model. The initial test started with poi and salary. Then one feature was appended at a time from a list and its evaluation metrics were compared with the max evaluation metrics. If the feature added ended up maximizing the evaluation metrics, it was kept providing new max evaluation metrics. The process continued until the last feature in the list only appending those which maximized the evaluation metrics. This process was repeated 5 times storing each test in a dictionary, which included the test number, the accuracy, the precision, the recall, and a list of best features. At the end the best features end up being those that returned the maximum accuracy while keeping the precision and recall above .30.

This is an extensive process that managed to increase all evaluation metrics in balance. The optimal features mostly returned are the following:

['poi', 'salary', 'shared\_receipt\_with\_poi', 'director\_fees', 'exercised\_stock\_options']

However, sometimes given the randomized processing of train and test data the optimal features returned are the following:

['poi', 'salary', 'to\_messages\_impact', 'expenses', 'total\_stock\_value', 'from\_this\_person\_to\_poi', 'exercised\_stock\_options']

**New Features:**

I engineered the to\_message\_impact, the from\_message\_impact, and the total\_compensation\_abs. The to\_message\_impact measures the proportion of messages that came from point of interest over all the messages sent to a specific person. The from\_message\_impact measures the proportion of messages that went to a point of interest over all the messages sent by a specific person. The total\_compensation\_abs measures the total compensation including all financial features including deferred payments as positive numbers since that was money owed to the person. The person could potentially receive the differed sum or not.

Adding the new feature from\_message\_impact decreased accuracy, precision, and recall altogether. Adding the other new feature to\_message\_impact also all evaluation metrics. Finally, adding total\_compensation\_abs also decreased performance.

1. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?  [relevant rubric item: “pick an algorithm”]**

After initially selecting support vector machine (SVC), I ended using decision trees because support vector machine was only focusing on getting a high accuracy score. After running the automatic feature selection algorithm, SVC ends up with an accuracy of 95%. However, the precision and the recall was undefined by a divide by zero error. The classifier was only classifying non-point of interest, making it an invalid test. I also tried Naïve Bayes and Random Forest but they were not best at getting high precision and recall scores. Random Forest was a bit slow as well. The Decision Tree classifier ended up providing out of 30 tests an average accuracy of .83, precision of .37, and recall of .36. However, sometimes it provided results out of 30 test as higher as the average accuracy of .85, precision of .47, and recall of .47.

1. **What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well?  How did you tune the parameters of your particular algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).  [relevant rubric items: “discuss parameter tuning”, “tune the algorithm”]**

Tuning parameters for example in Decision Trees could mean to provide the best information gain or best impurity score. By default, the min\_samples\_split parameter is set to 2 splits, which means that the minimum size of the sample to calculate either impurity or information gain is 2. The splitting sample can be a factor in maximizing these values. However, increasing the size too much could be counterintuitive possibly overfitting the test and worsening performance.

I tuned out my algorithm using GridSearchCV. However, by doing so the algorithm managed to only increase accuracy and precision at the cost of recall. To me, this is not valid. I created an auto\_feature algorithm to choose features to help to improve those in conjunction. The algorithm adds the feature that maximizes all evaluation metrics or otherwise it is removed if max\_eval\_foc is “both”. If max\_eval\_foc is “reca,” it adds the feature that maximizes both accuracy and recall while maintaining precision greater than .30. Using the best estimators chosen by GridSearchCV in auto\_feature did not manage to reach .30 on recall. Therefore, I ended up using the default parameters, which were the ones maximizing all evaluation metrics in balance.

1. **What is validation, and what’s a classic mistake you can make if you do it wrong? How did you validate your analysis?  [relevant rubric items: “discuss validation”, “validation strategy”]**

Validation is used to avoid overfitting. Testing the same data can be a problem. The classifier would go over the same data classifying what it already had classified before. This can lead a data scientist to believe that the model is optimal and that it will behave the same way on untested data. On the contrary, by doing a cross-validation such as the train\_test\_split, the data are fitted using the train data and tries to predict on the test data never seen before by the model. This allows the model to truly try to predict and have the chance to make mistakes.

I validated my analysis using StratifiedShuffleSplit, which provides a randomized train and test indexes for each fold, which allows appending random feature values in the train and test data. This allows for true randomization and prevents divided by zero errors as the trains and test are randomized and folded iteratively.

1. **Give at least 2 evaluation metrics and your average performance for each of them.  Explain an interpretation of your metrics that says something human-understandable about your algorithm’s performance. [relevant rubric item: “usage of evaluation metrics”]**

My evaluation metrics are accuracy, precision, and recall. Accuracy is the proportion of correct predictions out of the total predictions. In this case, this is the rate in which the model classified a person as a point of interest when the person indeed was a point of interest or vice versa over all the predictions.

Precision represents how many times the model classifies a point of interest correctly over the times where the model classify a point of interest correctly and incorrectly. False positives should be true negatives. In other words, points of interest classified incorrectly are not actual points of interest. Thus, the increase in wrongly classifying point of interest decreases the ratio and makes it less precise at identifying points of interest.

Recall represents how many times the model classifies a point of interest correctly over the times where the model classify a point of interest correctly and classify a non-point of interest incorrectly. The more it classifies a non-point of interest incorrectly, the less is the ratio. The false negatives should have been true positives. In other words, non-points of interest classified incorrectly are actually points of interest. So the less false negatives, the better is the model at missing points of interest, and the ratio increases.

**Resources:**

[www.scikit-learn.org](http://www.scikit-learn.org)

<https://medium.com/machine-learning-101/chapter-3-decision-tree-classifier-coding-ae7df4284e99>

<https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

<https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/>