P5: Intro to Machine Learning Report

Shawar Nawaz

About the dataset

Length of the dataset = 146

Number of POIs = 18

Number of non-POIs = 128

Number of features(excluding email-address) = 20

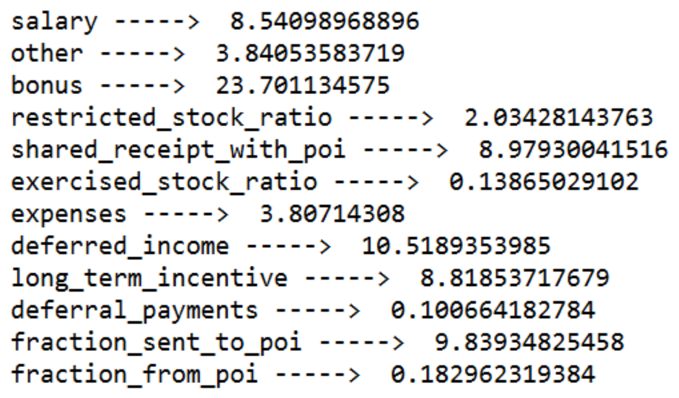
Almost all features have missing values.

Q1. 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?

- In 2001, Enron was one the largest corporations in the world. But it was revealed that they were involved in a lot of malpractices by using accounting loopholes, special purpose entities and poor financial reporting and were able to hide billions of dollars in debt from failed deals and projects.
- Eventually this scandal became the source of a massive dataset after the release of most of the higher level executives' emails and financial reports. In our dataset, we have the financial list along with whether this was a suspicious 'person of interest' during the trial. We can use machine learning algorithms to study various features of their financial info in a training set and then test it on another portion of the dataset to check various features of our accuracy. If we get good results (provided the training and test sets are massive), then we can use this algorithm in future cases to find other potential criminals in executive positions.
- In terms of outliers, there was one person whose exercised stock options were way above the rest of them. On closer investigation, I saw that this data point was actually a sum of all the others. So I went ahead and removed it. Another anomaly I noticed was in the deferred income section where one of the values were negative and all the rest were positive. On looking at the official financial info, I noticed that this negative value was actually a typo, so I used the absolute value in the dataset. Another discrepancy I found was that there was a name by 'The travel agency in the park' which was clearly not a person, so I removed it. Finally there was a person in the dataset who had missing values in all the features so I removed him too.

Q2. 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.

- The features that were up for consideration were -> ['poi', 'salary','other', 'bonus', 'restricted_stock_ratio', 'shared_receipt_with_poi', 'exercised_stock_ratio', 'expenses', 'deferred_income', 'long_term_incentive', 'deferral_payments', 'fraction_sent_to_poi', fraction_from_poi']. Of these restricted stock ratio, exercised stock ratio, fraction_sent_to_poi and fraction_from_poi were features I created because there were a couple of features that were the sum of other features. So, instead of using all of them(which would create problems since they were related), I decided to use them as fractions of the total and eliminated the rest.
- To check the difference in results by using the created features vs the original features, I used the same algorithms twice, once with the new features and once with the original features. In the case of the algorithm with the original features, I got a precision score of 0 and a recall score of 0 as well. Compare this with the precision score of 0.167 and the recall score of 0.33 I got when using the algorithm with the new feature list.
- I used the SelectKBest(f_classif) which is a form of Univariate feature selection method that works by selecting the best features based on univariate statistical tests. K denotes the number of best features you want it to select and f_classif computes the ANOVA F-value for the provided sample. I tried getting the best features in two ways using the SelectKBest method. The first was a step by step process where I chose the number of features I wanted based on the p-value. If the p-value was < 0.1, that feature was chosen. In a later step, I tried using a pipeline along with the GridSearchCV method that gives us the best values for the parameters.
- While running the data step by step I got 0.33 for both precision and recall.
- I did not require any scaling as the classifiers I used (Decision tree and Naïve Bayes) did not depend on Euclidian distance of the points.
- The scores of the various features are given below:-



```
salary ----->  8.54098968896
other ----->  3.84053583719
bonus ----->  23.701134575
restricted_stock_ratio ----->  2.03428143763
shared_receipt_with_poi ----->  8.97930041516
exercised_stock_ratio ----->  0.13865029102
expenses ----->  3.80714308
deferred_income ----->  10.5189353985
long_term_incentive ----->  8.81853717679
deferral_payments ----->  0.100664182784
fraction_sent_to_poi ----->  9.83934825458
fraction_from_poi ----->  0.182962319384
```

Features chosen – ‘salary’, ‘other’, ‘bonus’, ‘shared_receipt_with_poi’, ‘expenses’, ‘deferred_income’, ‘long_term_incentive’, ‘fraction_sent_to_poi’ as they seem to have a strong relationship with the poi factor.

Q3. & Q4. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

- The Decision tree classifier algorithm I have used in this project has various parameter options and it is very important to use optimum values for these parameters to get best results. Thus, tuning is the process by which we obtain the best values for to obtain the best efficiency of our algorithm.
- You have to be very careful when tuning your algorithm as extreme values for parameters may result in overfitting/underfitting the data. For example, I used min_sample_split as a parameter to tune my decision tree classifier algorithm, where, if I had used a very high value could have lead to under fitting and thus making bad predictions on the test set.
- If we aim for high prediction scores on our training data, we run into the issue of overfitting over that data and not being able to make good predictions over our test data.
- The two algorithms I ended up using were the Decision Tree classifier and the Naïve Bayes classifier. I used GridSearchCV, which iterates over all the combinations of various parameters fed into it and returns the best combination for our classifier. Not using this step will result in usage of a classifier not up to its optimum efficiency.
- The parameters I tuned for my decision tree classifier algorithm were ‘splitter’, ‘max_depth’ and ‘min_samples_split’. The optimum values returned were ‘random’, ‘None’ and ‘10’ respectively. I also tried tuning the number of features to select which returned 8.
- For the naïve bayes algorithm, I tuned the number of features we use to get the best results possible.
- The DecisionTree Classifier gave me a precision score of 0.167 and a recall score of 0.33. But the Naïve Bayes Classifier gave me a precision of 0.5 and a recall of 0.33. Thus I chose the Naïve Bayes classifier as the better algorithm for this case.

Q5. & Q6. What is validation, and what’s a classic mistake you can make if you do it wrong? How did you validate your analysis?

- Cross validation is the method by which we randomly split the data into training and testing sets. Then we train the model over the various training sets and test it over the test sets, the result of which is taken as the average of all. This gives you a more optimum result instead of randomly splitting the dataset over a pre defined value

- I validated my model by using the stratified shuffle split method using 1000 random splits of the dataset and returning the average of the predictions.
- I chose this method as the dataset was too small with only 18 pois, therefore, if I randomly split the dataset over a pre defined value say 20% as test data, I would probably get only 3 pois in the test set.
- The best result I obtained with tester.py was by using Naïve Bayes after splitting the data by the stratified shuffle split and using the parameters with a p_value < 0.1.
- The final result I obtained was 0.418 for precision and 0.303 for the recall.