**Udacity Machine Learning Final Project**

Kishan Patel

Santa Clara University.

Date: 07/31/2017

**Understanding the Dataset and Question**

Data Exploration (related lesson: "Datasets and Questions") :

**Task 1:**

So here we have pre-chosen two features ‘poi’ and ‘salary’. Now as we want to evaluate this problem with using more feature, I think we need to use ‘from\_poi\_to\_person’ and ‘long\_term\_incentives’, As they are the most credible subjects after salary, As the most people who are convicted were communicating frequently and they all have huge long-term profits.

1. Total number of data points

Now as we are using these features the total no. of useful data on those features are as follows,

|  |  |
| --- | --- |
| Feature | No. of non-zero data |
| Poi\_d (the people who are person of interest) | 18 |
| salary\_d (Salary information) | 95 |
| from\_poi\_d (emails from POI people to other) | 86 |
| long\_term\_d (long term Incentives | 66 |

\*The above dictionaries are just for personal reference which allows us to see the distribution of specific features

with the name of the person associated with it.

I have checked several other features like 'bonus', 'total\_payments', 'exercised\_stock\_options’ and plot them on scatter plot, But I did not able to see that much deviation in data, they all falls on same small range.   
But I think these factors will help us in finding the fraud people in the company.

1. Allocation across classes (POI/non-POI):

Here we have 18 POI people and all other 128 people are not POI.

1. Number of features used:

As mentioned in the above table, here we are using total **4 features.**

Here there are some missing values but these is the problem with all the other features also, these number of data are sufficient for our analysis.

Outlier Investigation:

**Task 2:**

There are several outliers in our data which are just typos as they don’t have any significance related fraud like **‘TOTAL’**, which we learn in the outlier class which is purely an error in data base which is outlying in every category and changing the training model. The data in this name are as follows:

{'salary': 26704229, 'to\_messages': 'NaN', 'deferral\_payments': 32083396, 'total\_payments': 309886585, 'exercised\_stock\_options': 311764000, 'bonus': 97343619, 'restricted\_stock': 130322299, 'shared\_receipt\_with\_poi': 'NaN', 'restricted\_stock\_deferred': -7576788, 'total\_stock\_value': 434509511, 'expenses': 5235198, 'loan\_advances': 83925000, 'from\_messages': 'NaN', 'other': 42667589, 'from\_this\_person\_to\_poi': 'NaN', 'poi': False, 'director\_fees': 1398517, 'deferred\_income': -27992891, 'long\_term\_incentive': 48521928, 'email\_address': 'NaN', 'from\_poi\_to\_this\_person': 'NaN'}

The other data same as this is **"THE TRAVEL AGENCY IN THE PARK"** which by name only doesn’t make any sense to include in to the dataset. Now this falls really on other side of previous outliers as the maximum no. of data in this subject are ‘NaN’ or zero which also affect our train model. The data in this name are as follows:

{'salary': 'NaN', 'to\_messages': 'NaN', 'deferral\_payments': 'NaN', 'total\_payments': 362096, 'exercised\_stock\_options': 'NaN', 'bonus': 'NaN', 'restricted\_stock': 'NaN', 'shared\_receipt\_with\_poi': 'NaN', 'restricted\_stock\_deferred': 'NaN', 'total\_stock\_value': 'NaN', 'expenses': 'NaN', 'loan\_advances': 'NaN', 'from\_messages': 'NaN', 'other': 362096, 'from\_this\_person\_to\_poi': 'NaN', 'poi': False, 'director\_fees': 'NaN', 'deferred\_income': 'NaN', 'long\_term\_incentive': 'NaN', 'email\_address': 'NaN', 'from\_poi\_to\_this\_person': 'NaN'}

Now the other outliers are being tested as 'LAVORATO JOHN J', 'DIETRICH JANET R', 'MARTIN AMANDA K' as they are at extremes in ‘from\_poi\_to\_person’ dictionary ‘from\_poi\_d’ and ‘long\_term\_d’. And as they are not POI people they are tried by including and excluding. At the end only 'MARTIN AMANDA K' was removed and all other data kept.

\*This is really subjective, one can also argue that this is important for finding trends, May be but I didn’t find any significance of this data in our training.

Now there are other outliers in salary like the ‘KENNETH L’ but as he has a high position in the organization and that is actual salary it can be prove as good data so that it is being kept for the training.

**Optimize Feature Selection/Engineering:**

Now in this section we basically are creating new feature and selecting the best features form that using SelectKBest feature and then scaling them for proper weightage for training.

Create new features:

**Task 4:**

So, now as we can see in the dataset that there are very little amount of financial feature other than salary and long\_term\_incentive that have deviation and can help us to bring new perspective in training our model.

* So, the following feature like **‘salary to bonus ratio’** which can show us whether the fraud people are

getting high bonus instead of salary or if they have significant deviation from other people.

* The other feature that I think makes sense is **‘from this person to poi email ration’** as from this we can see the priority of emails form the person to POI, as there might be several people who need to email too many peoples, so only ‘from messages’ don’t make sense while this ration shows us that what amount of total email the person is sending to POI people.

So, following two features are created:

1. **'salary\_bonus\_ratio'**
2. **'messages\_to\_poi\_percentage'**

Intelligently select features (related lesson: "Feature Selection"):

Now to select the best features form the list of five features we are using ‘**SelectKBest’** algorithm that looks in to features and labels and gives us no. of most important features for optimizing the time as well as improve accuracy.

Now, I’ve decided to work with best 3 features from our list of five features. Now I also test the accuracy with Two features and I found It was not as better as this. And 3 feature training will be relatively faster and very standard in Machine learning.

So, with the SelectKBest algorithm following are features their scores and their selection.

**Features:**

['salary’, 'from\_poi\_to\_this\_person', 'long\_term\_incentive', 'salary\_bonus\_ratio', 'messages\_to\_poi\_percentage']

**Scores:**

[ 1.38874181 , 0.59466452, 2.74371308, 0.10634337, 2.54830139]

**Selection:**

[ True False True False True]

So, as we can see above the features with highest scores were selected to proceed with. The top 3 features ‘Salary’, ‘long\_term\_incentive’ and ‘messages\_to\_poi\_percentage’.

Properly scale features (related lesson: "Feature Scaling"):

Now as we learned in the class, Scaling of the feature is important to apply equal weightage to every feature, as our features ranges are too high. As the salary is in range of 1e+06 while salary\_bonus\_ration is in range 1e+02, which might make it just bias in the data and which means our training model will completely ignore these features.

So, to scale all the features in the range of 0 to 1, **‘MinMaxScalar’** function is used.

Now, as our all features are in range 0 to 1, we can give equal importance to all the features.

**Pick and Tune an Algorithm:**

Pick an algorithm (related lessons: "Naive Bayes" through "Choose Your Own Algorithm"):

**Task 4:**

Now as we learned there are many classification algorithms, I’ve tested almost every classifier from **Naïve Bayes** to **AdaBoost** and **SVM**. The main parameters the classifier is evaluated are their **accuracy** before tuning and their **precision and recall values**.

Now I end up with using **DecisionTreeClassifier** as there are many other classifiers which initially have high value of accuracy but their precision and recall were very low, so they have been eliminated.

So Following Classifiers are being tested on following criteria:

|  |  |  |
| --- | --- | --- |
| **Classifier** | **Accuracy Score** | **Precision and Recall Score** |
| Naïve Bayes | Low | Low |
| AdaBoost | High | Low |
| Random Forest | High | ILL Defined |
| **Decision Tree** | High | **High** |
| SVM | High | ILL Defined |

So, now from above selection Metrix, we can see that the Decision Tree classifier is the best choice for us. Now you can also see the accuracy scores on the code. There are algorithms like ‘State Vector Machine’ and ‘Random Forest’ which have high accuracy have **‘Ill defined’** accuracy and recall as all the prediction of the test data falls in to the one criteria. So those algorithms are ignored.

Now the best algorithm for this case **Decision Tree Classifier** is selected and tuned for following parameters:

* **Criterion:** The function to measure the quality of a split. Supported criteria are “Gini” for the Gini

impurity and “entropy” for the information gain. I have tried both and for our case ‘entropy’ worked best.

* **max\_depth:** The maximum depth of tree, which means the nodes will expand until this no. of leaves

left. So basically, we expand our tree until we have 6 samples left.

* **min\_samples\_split:** As we learned this in the class which defines minimum number of samples

needed to create node, which means we need no. of elements in other side to

expand in that area. Which reduces the over fitting of the data, which increases

accuracy.

* **Presort:** We have also used Presort function as it will expedite the splitting process as it will first sort

the data which are easy to split and better result can be obtained.

* **max\_features:** Minimum number of feature being consider while looking for split will protect our

training from over fitting.

* **min\_samples\_leaf:** This is also saving us from over fitting our data as it restricts the minimum number

of samples required to be at leaf to create a node.

Tune the algorithm:

The algorithm was tried to be tuned with ‘**GridCVSearch’** but the accuracy was not getting better. You can see that attempt in coding part, but as the accuracy or precision and recall are not improving. With some tuning, I was able to increase the accuracy but the precision and recall were very low than our requirement. So, the above tuning was kept as it was successfully allow us to meet our requirements.

Now there was also option to use PCA to reduce the feature and then apply a classifier, but as we are meeting our requirement with ‘**SelectKBest’** algorithm, this option was ignored.

**Validate and Evaluate:**

Usage of Evaluation Metrics (related lesson: "Evaluation Metrics")

Now for the validation as we have separated 30% of data for the testing purpose, we are using ‘**accuracy\_score’, ‘precision\_score’** and **‘recall\_score’** to evaluate and validate our classification.

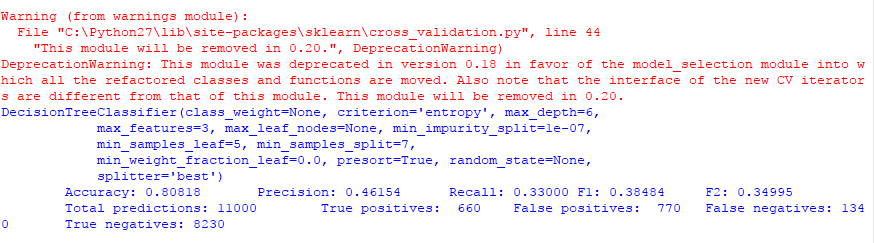
Now as we have 30% of data which was not used for training which means that data is totally unknown to our training model. So, from these results we can justify the accuracy of our model as well as being confident for our model to work in real world to unknown data with reasonable accuracy.

Discuss validation and its importance:

As discussed above the validation is important to check whether our training model is working correctly or not with unknown data. Now there is tradeoff between selection of training data and test data split, as we have chosen the very standard practice used in industry to split data in 70-30 ratio. As above we have used **accuracy\_score** and **precision** and **recall scores**.

Algorithm Performance:

The algorithm performance was evaluated on every step with **accuracy\_score** the final performance using tester.py gives us following result.



So, as we can see our accuracy is 81% and Precision and Recall both are above 0.3 which meets our requirements.