Introduction to Machine learning

Shmuel Naaman, Jun 2015.

1. Some background on the dataset and how it can be used. Handling the outliers in the data

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into public record, including tens of thousands of emails and detailed financial data for top executives. The data set includes:

Financial features: 'salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'

Email features: 'to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', 'shared_receipt_with_poi'

POI label: 'poi'

The goal of this project, I will build using a machine learning algorithm, a person of interest ("poi") identifier based on the Enron public financial and email data. As a starting point I have a list of Enron employs and their associate emails and detailed financial data. For each employ, the list indicates weather he is "poi" or not. The algorithm should decide based on the list of features, with a reasonable amount of certainty which of the employees is "poi" and which is not.

Machine learning evolved from the study of pattern recognition and computational learning in artificial intelligent. Machine learning explores the construction of algorithm that learns from example and then makes data-driven predictions or decisions on data, rather than following strictly static program instructions.

Outliers: TOTAL: the data set include a summary of all rows that is label as TOTAL. Since TOTAL is not relevant to the question we ask, it was removed before any analysis took place.

"NaN": values in the data set were replaced by 0. This values were not considered when we scaled the features.

Data statistics,	Total numbers of data points	2880	Total numbers of NaN	1023
	Total numbers of features	21	Total numbers employees	144
	Total numbers poi	18		

List of NaN's

'loan_advances',	142	'director_fees',	129
'restricted_stock_deferred',	128	'deferral_payments',	107
'deferred_income',	97	'long_term_incentive',	80
'bonus',	64	'other',	53
'expenses',	51	'salary',	51
'exercised_stock_options',	44	'restricted_stock',	36
'total_payments',	21	'total_stock_value',	20

2. The features did I used in the POI identifier, and the selection process to pick them. Scaling, Feature engineering.

Scaling: All the financial features were normalized as fallow: Xn = (X-Xmin)/(Xmax-Xmin). Where X is the feature, Xn is the normalized feature. Xmin and Xmax are the corresponding min and max of each feature not including zeros. I did not include the zeros because many of the zeros are 'NaN' value that being replaced by zeros. Therefore, including the zeros will artificially affect the scaling. The reason to scale each feature is that we are more interested in the differences between 'poi' and non 'poi' and not in the real value of each feature. Another important issue is that some algorithems required scalling, and at this point Im not sure yet which algorithm I will use.

Engineering: Another important source of information might be in the Emails between employees. It might be the case that 'poi' communicates between them more than the average. This can be detected by calculating the percentage of email to or from 'poi' relative to each person total number of Emails. For that I calculate 3 new features.

A_F_P =	from_poi_to_this_person	A_T_P =	from_this_person_to_poi
A_F =	from_messages	A_T =	to_messages

Norm_Email = (A_F_P + A_T_P)/(A_F + A_T) - percentage of email exchange with 'poi' with respect to total email exchange.

Norm_Email_T = (A_F_P + A_T_P) / (A_T) - percentage of email exchange with 'poi' with respect to total email send.

Norm_Email_F = (A_F_P + A_T_P) / (A_F) - percentage of email exchange with 'poi' with respect to total email received.

After a first look at the data using a scatter plot (not included in the final version of the code), I decided that I cannot really make any smart decision just by looking at the data, or considering the name or content of each feature. However I can reduce redundancy by omitting features with similar information (avoid multicolinarity), toward that I calculate correlation values between each pair of features as depict on figure 1.

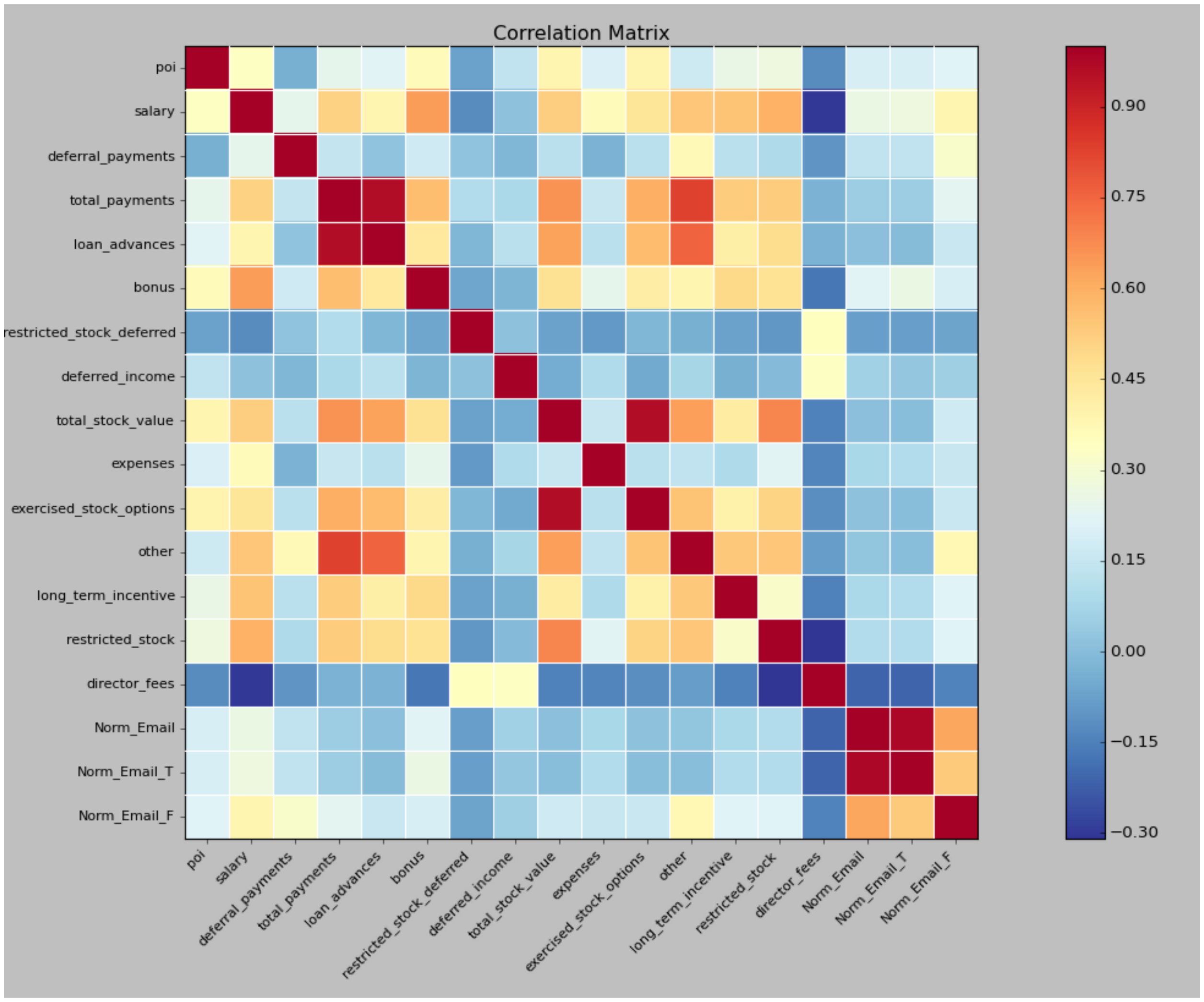

Figure 1: correlation values between each pair of features. Feature names indicate on the x and y axis, colors represents correlation values (see color bar).

I can omit one feature from each pair that shows high correlation value (r > 0.8). This is possible because high correlation indicates about a redundancy:

- 'total_payments' correlates with 'loan_advances' (r= 0.9636), I choose to keep the, 'loan_advances' because it is more specific, and the information included in the 'total_payments' is just the sum of all other payments.
- 'total_payments' correlates with 'other' (r= 0.829), so I left "others".

- 'total_stock_value' correlates with 'exercised_stock_options' (r= 0.9638), in this case I left the 'exercised_stock_options', because I believe that ‘poi’ might know in advance that something going to blow up soon.
- 'Norm_Email' correlates with 'Norm_Email_T' (r= 0.977), so again I left the more specific feature 'Norm_Email_T'.

Another important consideration is the Importance of each feature to the model:

'other',	0.23333333	'expenses',	0.2
'salary',	0.16666667	'exercised_stock_options',	0.16666667
'deferred_income',	0.1	'bonus',	0.06666667
'restricted_stock',	0.03333333	'Norm_Email_T',	0.03333333
'deferral_payments',	0	'loan_advances',	0
'restricted_stock_deferred',	0	'long_term_incentive',	0
'director_fees',	0	'Norm_Email_F'	0

After removing the highly correlated features the model prediction change

From:

Accuracy: 0.84620, Precision: 0.39508, Recall: 0.28900, F1: 0.33381, F2: 0.30540

To:

Accuracy: 0.85320, Precision: 0.43293, Recall: 0.32600, F1: 0.37193, F2: 0.34294

After removing the features with zero importance the model prediction change to:

n_estimators = 30

Accuracy: 0.85807, Precision: 0.45639, Recall: 0.33750, F1: 0.38804, F2: 0.35605

Apparently, features good selections increase the model prediction power.

n_estimators = 20

Accuracy: 0.85687, Precision: 0.44769, Recall: 0.31450, F1: 0.36946, F2: 0.33440

n_estimators = 60

Accuracy: 0.86027, Precision: 0.46624, Recall: 0.33150, F1: 0.38749, F2: 0.35184

When using the GaussianNB for the final same features

Accuracy: 0.84727, Precision: 0.37885, Recall: 0.22750, F1: 0.28429, F2: 0.24726

When using the GradientBoostingClassifier for the final same features

Accuracy: 0.84240, Precision: 0.38481, Recall: 0.30400, F1: 0.33966, F2: 0.31733

3. The algorithm that I end up using and other that I tried.

Eventually I choose the “AdaBoostClassifier”

{ clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=30, learning_rate=1)}

I did try the: “GaussianNB()”, “GradientBoostingClassifier()“.

4. Tune the parameters of an algorithm,

After selecting the algorithm that fits with the characteristics of the features in the dataset, it is important to tune the model parameters. Every model has some parameters that can be adjusted, for example in a “decisiontreeclassifier” you can adjust among other parameters the “**min_samples_split** “. That means that you set the minimum number of samples required to be at a leaf node. This process is important because we wants to optimize the model we used to the specific data set, problem, requirements.

For the analysis in this report: first I tested the performance of few algorithms to find the model that suite best. I then adjusted or tune the parameters (n_estimators and learning_rate) by trial and error. My goal was to achieve at least the minimum requirements indicated in the report instructions. After achieving that, I attempt to fine tune the parameters programmatically using the GridSearchCV function. I choose not to use the programmatic option eventually because it took very long time and because the optimization did not consider the requirements of the report instructions (so for example I got high precision but low recall).

5. Validation

Validation is a process of deciding whether the result or the accuracy of the model is good enough to describe the data. A classic mistake will be to use similar dataset for validation and training. In many cases this will give you a wrong impression about the real performance of the model. You will get high performances but when you try to implement the result to real data the performance will be much low. This is because of over fitting.

To validate the model, I separate the data in to test (10% of the data) and train (90% of the data) subsets. Then calculated the confusion matrix and extract the precision and recall of the model. This procedure was repeated twice each time with different subset.

6. Evaluation metrics, and average performance for each of them

True Positive: 1, True Negative: 11, False Positive: 2, False Negative: 2

True Positive: 1, True Negative: 12, False Positive: 1, False Negative: 1

The average performance:

The precision of this classifier: 0.4 The recall of this classifier: 0.5

The interpretation of this result: Precision measures the proportion between numbers of true positive to the number of positive calls. In other words the ratio between the real poi detected to the total poi detected (true and false). This measure as its name hint indicates about the precision of the model. In this case we got a precision of 0.4, which means that out of all poi detected by the model 40% are real poi's.

Recall measure the proportion between the numbers of true positive to the total number of positive. In other words the ratio between the real poi detected to the number of poi in the group. This parameter measure the ability of the model to correctly detect poi's. In this case the model detects just a little more than 50% of the poi's in the group.

Of course the algorithm is far from perfect. The main problem is the small amount of data, especially the small numbers of poi in the data set.

References

www.wikipedia.org

www.scikit-learn.org

www.stackoverflow.com