Machine Learning Project

Predicting people of interest for the Enron case
by Ignacio Ferreras Astorqui

This project is composed by 145 employees from Enron multinational. In all these employees there are only 18 of them marked as POI. This leaves a greatly uneven distribution of 127 non-POI against 18 POIs. I started using all the possible features, 21, but my algorithm establishes which ones are more relevant to the classification. There are plenty of features with NaN as their values but change it to 0 might input wrong values into our calculations. However, in order to do some calculations I had to do that transformation, I will get into that in the following 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?*

The goal of the project is to create a machine that should be capable of predicting possible persons of interest for the Enron case, based on the available data. This is the perfect case for using machine learning given that with it we can not only classify the possible persons of interest but also predict if a newly found employee is a person of interest or not. Given information like their salary or the number of times they exchanged messages with known POIs (Persons of Interest). This information could give us some insight about possible POIs if some information about them stands out from the rest and not from the POIs.
For the outliers I just removed the one we found during the investigation of the dataset, that was a mistake in the dataset. However I decided not to remove any other possible outliers given that they are all employees and the might provide useful information or be POIs themselves.

2. *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?*

For the project I added a new feature called total_income which was composed of all the different ways of income of the employee. The addition of this feature affected greatly to the project, as we can see in the following result:
Results of the original dataset:
Accuracy: 0.853933333333 Precision: 0.449336870027 Recall: 0.4235
Results of the edited dataset (new feature included):
Accuracy: 0.8548 Precision: 0.453157894737 Recall: 0.4305
And the weight the tree estimator assigned it was: 0.12523754135219584

We started using all the possible features. Bearing in mind that we were going to use PCA or decision trees for the project. The number was reduced once we applied the previous algorithms. I am going to focus on the results provided by the decision tree due to the fact that it is the algorithm I have chosen to work with.
The cleaning of the features was made by removing the features which the decision tree classifier deemed useless by establishing the feature importances to 0.0. In every iteration there were more features set to 0.0 until it got reduced to 7 features.

N. of features	Accuracy	Precision	Recall
22	0.8262	0.336	0.311
8	0.850	0.44	0.41
7	0.854	0.452	0.4295

g? What other one(s) did you try? How did
ms?

I used the Decision Tree Algorithm. I tried the linear regression, logistic regression, GaussianNB and finally MLPClasifier. I used the model performance given by the GridSearchCV and once I got the best combination of params I used the tester provided to find the best between them.
Surprisingly the MLPClassifier performed the best in the GridSearchCV, however it was the worst in the given test. This was probably due to the small dataset. There are some graphs that will be discussed later on the document.

f an algorithm, and what can
the parameters of your particular
rs that you need to tune -- if this
y explain how you would have
a different model that does
).

4.

Tuning parameters in an algorithm makes it perform better. However tuning it wrong can make it become. For decision Tree Classifier there are a great number of parameters to tune. From the maximum number of features to consider when looking for the best split to presort the data or not.
The way I tune my algorithms was by trying every possible combination of the parameters just like using GridSearchCV but testing the results with the given test. This way at the end of the execution I have the best combination of parameters for the Decision Tree Classifier for this data. Which was:

{“presort": true,
"criterion": "gini",

"max_features": "auto",
"min_impurity_split": 1e-07,
"class_weight": "balanced"}

ou do it

5.

[relevant rubric item: “validation strategy”]

Validation consist in checking the efficiency of our classifier by testing it against part of the data and see how close are the p
redicted results to the real ones. For the validation of my classifier I first used the cross validation that comes with the GridSearchCV. Once I decided which classifier I was going to use I started using the test given for the project. The test calculates the precision and recall that gives a great understanding of how well the classifier is doing. It also uses the StratifiedShuffleSplit in order to preserve the percentage of samples for each class. This helps to have test and training data with an even number of POIs in them, given that there are so few of them (18 out of 146)

6.

[relevant rubric item: “usage of evaluation metrics”]

Precision is the fraction of retrieved documents that are relevant to the query:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

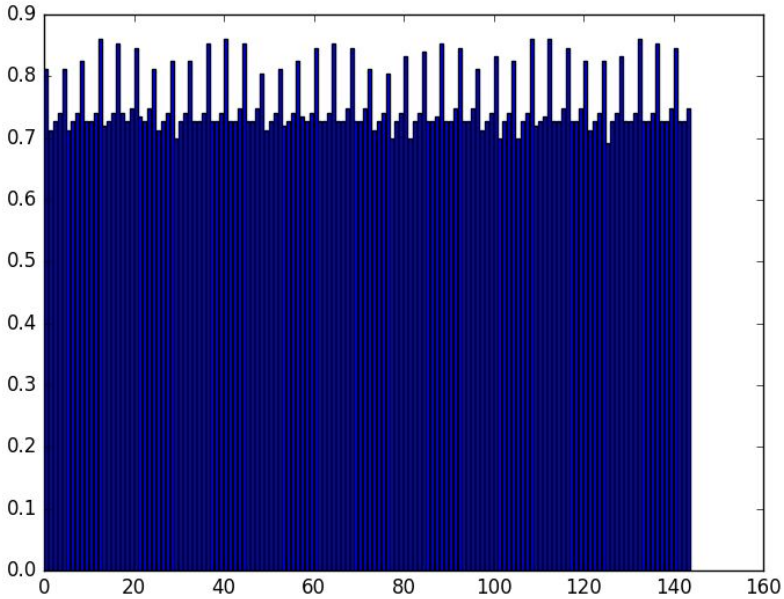
The precision compares the number of positive predictions that are right to the sum of all (true positives, false positives). However the recall compares the positive predictions that are right to the sum of the number of right positive predictions and wrong negative predictions.

This way the precision studies how the positive prediction is working. The recall studies the predictions against the real positive results.

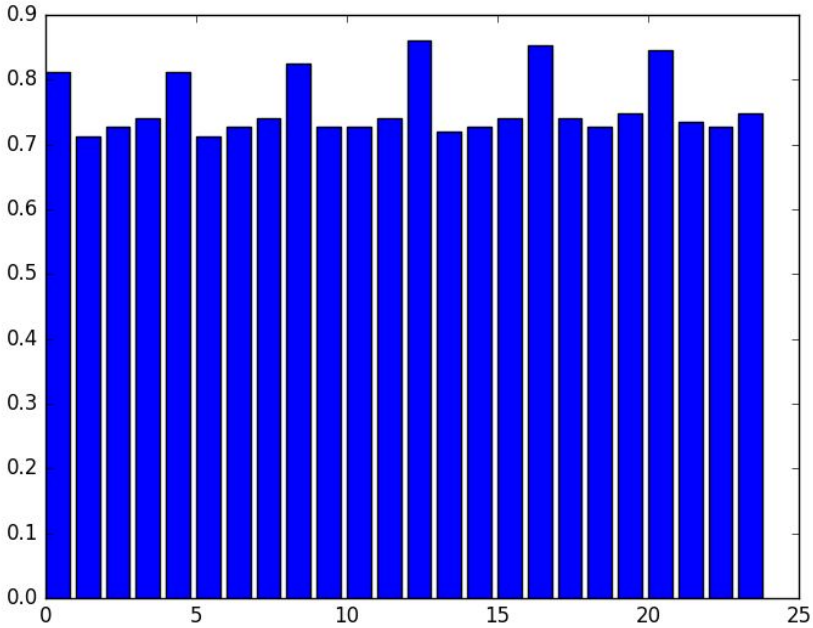
For my classifier my best results were:

Precision: 0.457519788918 Recall: 0.4335

Understanding the results of the investigation

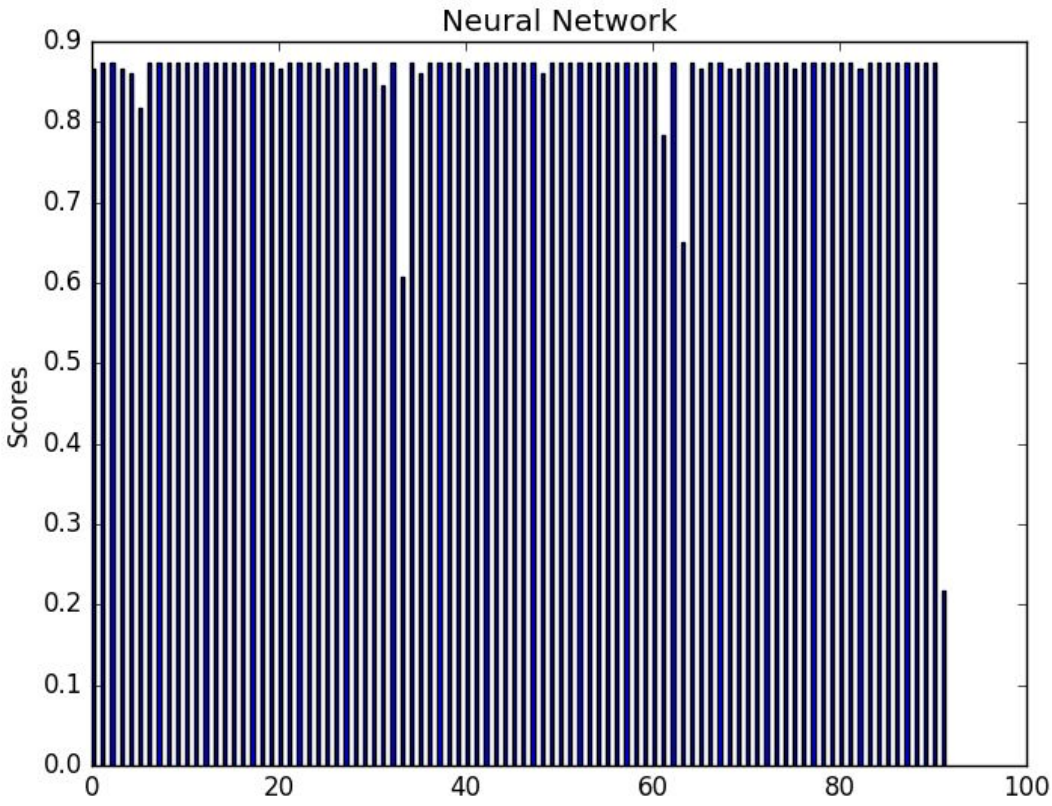


For example here we can see the results of the logistic regression obtained thanks to the GridSearchCV. It is curious to see the pattern in the results. It has to be related to the inputted parameters in our classifier. In order to see which one of the parameters is causing this effect we should zoom in the data.

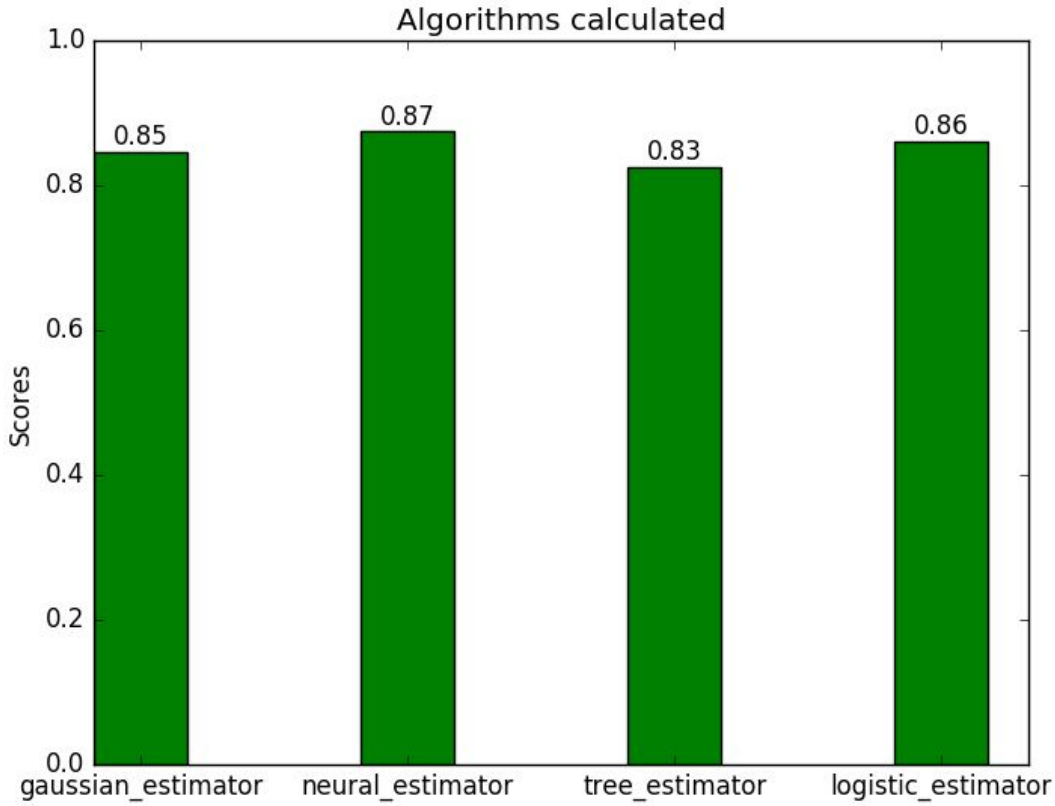


Here we can see a zoom in in the first group of data. Once we know in which execution the highest result is given we can investigate in the saved data for the parameter selection. The parameter that seem to make the greatest effect in the results was the solver, and the best solver was sag.

Now we are going to take a look at the neural networks in the MLPClassifier. During the making of the plots I found a huge drop between executions. The drop of accuracy was about a 0.6. After a hard look at the results I found out that the number of iterations greatly affected performance. Being 150 iterations the best result. This could be due to the small dataset. In my experience the neural networks work great for bigger datasets. However it seems that it works really bad for small datasets. Because even though the results below look really promising once we calculate the recall or the precision we see the worst scores of all the classifiers I have tested


Neural Network
Scores
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100

Finally we are going to take a look at all the results obtained during the testing and investigation of all our classifiers.



As we can see the best result is wielded by the neural estimator. However once we performed the evaluation metrics we saw that the Tree estimator came first by a long distance to the second.
This gives a great incentive to not only make cross validation but to go further into validation by using the evaluation metrics, which are the ones that really give some in depth insight to our classifier and its predictions.