REPORT FROM FINAL PROJECT – INTRODUCTION TO MACHINE LEARNING

COURSE: DATA SCIENCE FOUNDATIONS II

STUDENT: LEANDRO BARUCH

Identify Fraud from Enron Dataset

*"**Enron Corporation** was an American energy, commodities, and services company based in Houston, Texas. It was founded in 1985 as a merger between Houston Natural Gas and InterNorth, both relatively small regional companies. Before its bankruptcy on December 3, 2001, Enron employed approximately 29,000 staff and was a major electricity, natural gas, communications and pulp and paper company, with claimed revenues of nearly $101 billion during 2000. Fortune named Enron "America's Most Innovative Company" for six consecutive years.*

At the end of 2001, it was revealed that Enron's reported financial condition was sustained by institutionalized, systematic, and creatively planned accounting fraud, known since as the Enron scandal. Enron has since become a well-known example of willful corporate fraud and corruption. The scandal also brought into question the accounting practices and activities of many corporations in the United States and was a factor in the enactment of the Sarbanes–Oxley Act of 2002. The scandal also affected the greater business world by causing the dissolution of the Arthur Andersen accounting firm, which had been Enron's main auditor for years.

*Enron filed for bankruptcy in the Southern District of New York in late 2001 and selected Weil, Gotshal & Manges as its bankruptcy counsel. It ended its bankruptcy during November 2004, pursuant to a court-approved plan of reorganization. A new board of directors changed the name of Enron to **Enron Creditors Recovery Corp**., and emphasized reorganizing and liquidating certain operations and assets of the pre-bankruptcy Enron. On September 7, 2006, Enron sold Prisma Energy International Inc., its last remaining business, to Ashmore Energy International Ltd. (now AEI)." – From Wikipedia, the free encyclopedia*

The objective of this project, is use both financial and email data from 146 executives, and use those data to identify Persons of Interest (POIs). For that, I will analyse the dataset, all the data features, understand their correlations and see which are the best for this purpose. After that I will create a Machine Learning Model to indentify the POIs based on those features and check their efficience.

Summary

Exploring the Dataset 2

Checking the Features 4

Checking Classifiers with all features 5

Checking Classifiers with 5 top features 7

Checking Classifiers with new feature 8

Checking Classifiers with top 5 features from SelectKBest 9

Checking RandomForest and DecisionTree with adjusted parameters using GridSearchCV .. 10

Conclusion 12

References 12

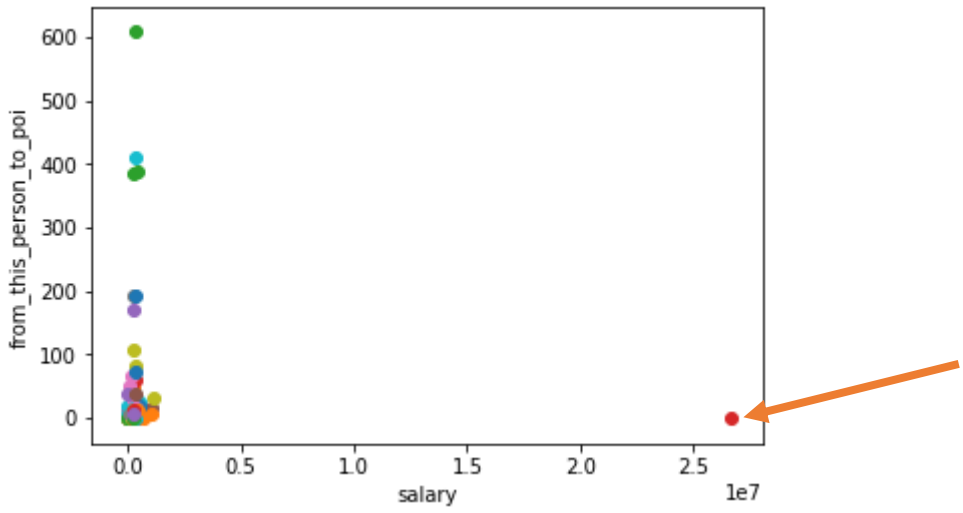
Exploring the Dataset

During dataset exploration, I have found some interesting information:

- We have 146 registers in our dataset, one of them was the 'TOTAL', so it was removed and now we have 145 registers;
- From 145 people, 18 of them are flagged as POI, they are:
    - Hannon Kevin P, Colwell Wesley, Rieker Paula H, Kopper Michael J, Shelby Rex, Delainey David W, Lay Kenneth L, Bowen Jr Raymond, Belden Timothy N, Fastow Andrew S, Calger Christopher F, Rice Kenneth D, Skilling Jeffrey K, Yeager F Scott, Hirko Joseph, Koenig Mark E, Causey Richard A and Glisan JR Ben F;
- The features we have at our disposal are:
    - salary, to_messages, deferral_payments, total_payments, exercised_stock_options, bonus, restricted_stock, shared_receipt_with_poi, restricted_stock_deferred, total_stock_value, expenses, loan_advances, from_messages, other, from_this_person_to_poi, poi, director_fees, deferred_income, long_term_incentive, email_address, from_poi_to_this_person

Checking those features, of course those which more call attention are Salary, Bonus, Total Stock Value. These features can tell us which person are receiving more money from Enron, and outliers may be guilty in this case.

Check #1

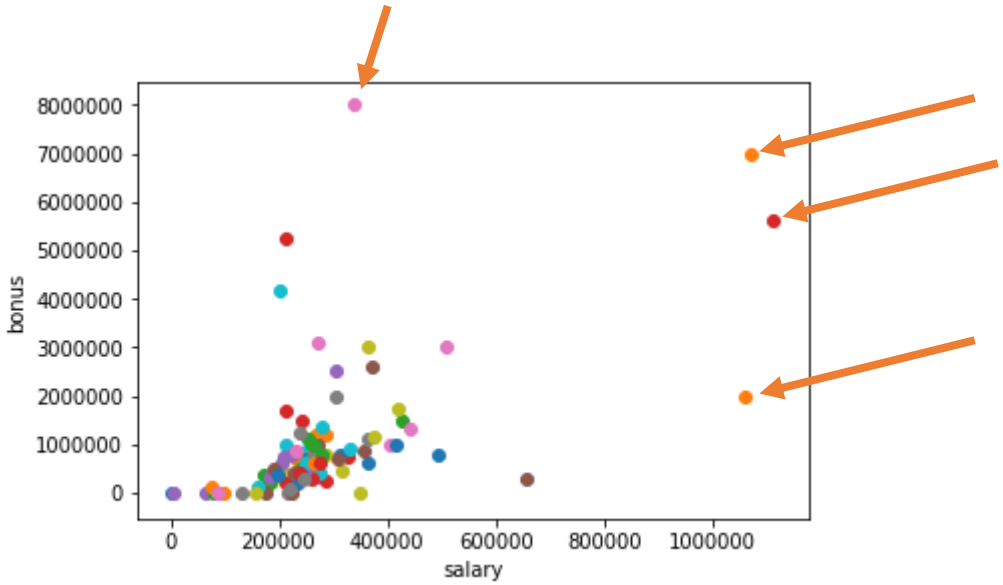


Salary x Messages to POI

We can see a big outlier.

After checking it, its name is ‘TOTAL’. Probably when the dataset was created someone forgot to delete the Total Row. I just deleted it.

Check #2



Salary x Bonus

Now we can see 4 more outliers. Three of them with the higher salaries in company, and one with the highest bonus.

The person with the highest bonus, is Lavorato John J, head of operations with a bonus of $8.000.000. And he is not flagged as POI. Suspicious…

The three top salaries of Enron are:

- Skilling Jeffrey K, CEO
  - Salary: $1.111.258
- Lay Kenneth L, founder and CEO
  - Salary: $1.072.321
- Frevert Mark A, CEO
  - Salary: $1.060.932

Those three occupied the highest role in the company, but it means that big difference in salary, since we are talking only about the high level of executives of Enron? Also, Frevert Mark A was not flagged as POI.

Seeing those differences, make me wonder how much is the average salary and bonus for high executive level in Enron. And the result is:

- Average salary: $ 284.088
- Average bonus: $ 1.201.773

It means the exemples above are way higher the average.

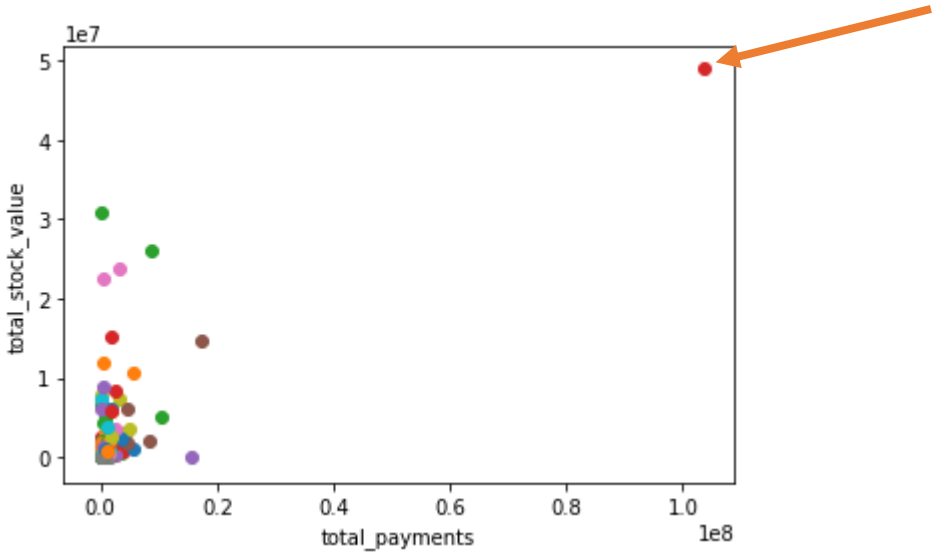
Also, to be honest, this average salary looks weird. The Total was around $26kk and dividing it per number of people, I expected another value.

So, I have checked for 'NaN' values and I have found this:

- Amount of NaN salary: 51
- Amount of NaN bonus: 64

Looks like the data is far from complete and it may interfere the capability of our model detect new POIs. We will have to use the data from e-mails to be more precise.

Check #3



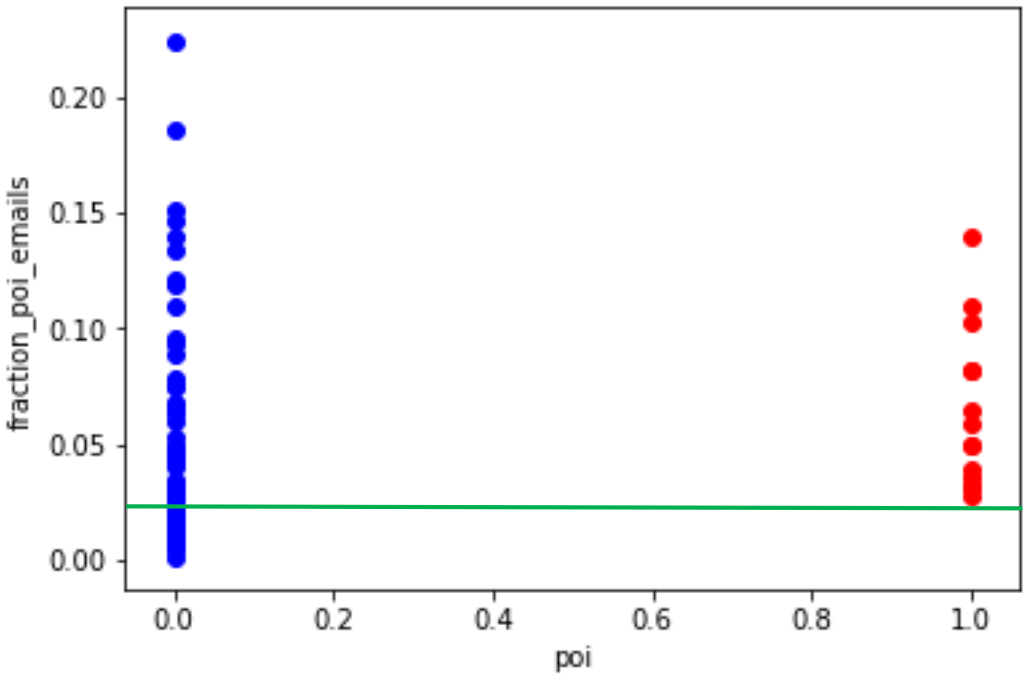
Total Payments x Total Stock Value

Now, considering all payments and stock value, a big outlier appears. And it is his second time, Lay Kenneth L with the big value of $103.559.793 in total payments and $49.110.078 of total stock value.

Checking the Features

Now we have removed a outlier and detected some behaviors from dataset, we need to check the features, and see what feature is more important in detecting a POI.

As I said before, use the amount of e-mails sent and received from POIs may be a good indicative to find a POI. Also, we have a few examples that received a big amount of cash but it's not a POI.



As we can see with the green line, POIs tend to exchange e-mail among them at least 3% of their total e-mails while not POIs sometimes have 0%.

Also we have a considerable difference between their averages:

POIs uses to have an average of 5% of e-mails exchanged among them, while Not POIs uses to have an average of 2,5% of e-mails exchanged with POIs.

It's worth to use this features in our classifier to see if improves it.

Checking Classifiers with all features

First, I believe we have too much features for the training points we have.

Let's check:

```
Number of training points:  100
```

```
Number of features:  20
```

I believe 5 features would be enough, but lets try with all.

SVM LinearSVC

```
training_time: 0.003 s
prediction time: 0.002 s
Accuracy with SVM * feat:  0.5227272727272727
Precision with SVM * feat:  0.16666666666666666
Recall with SVM * feat:  0.8
F1 Score with SVM * feat:  0.27586206896551724
```

Naive Bayes

```
training_time: 0.001 s
prediction time: 0.004 s

Accuracy with Naive Bayes:  0.8863636363636364
Precision with Naive Bayes:  0.5
Recall with Naive Bayes:  0.4
F1 Score with Naive Bayes:  0.444444444444445
```

Decision Tree

```
training_time: 0.001 s
prediction time: 0.002 s
Accuracy with Decision Tree * feat:  0.8409090909090909
Precision with Decision Tree * feat:  0.25
Recall with Decision Tree * feat:  0.2
F1 Score with Decision Tree * feat:  0.22222222222222224
Feature Ranking:
1 feature salary (0.220426513942)
2 feature to_messages (0.168173897756)
3 feature deferral_payments (0.136188597727)
4 feature total_payments (0.132625994695)
5 feature exercised_stock_options (0.105863661155)
6 feature bonus (0.0757862826828)
7 feature restricted_stock (0.0757862826828)
```

Random Forest

```
f1_score(labels_test, pred)
training time: 0.016 s
predicting time: 0.002 s
Accuracy with RandomForest * feat:  0.9090909090909091
Precision with RandomForest * feat:  1.0
Recall with RandomForest * feat:  0.2
F1 Score with RandomForest * feat:  0.33333333333333337
```

So, our F1 Score is too low as expected. Lets use the 5 more importants features from Decision Tree and check again.

Checking Classifiers with 5 top features

The five features that we will use in next test are: salary, to messages, deferral payments, total payments and exercised stock options.

SVM LinearSVC - Improved Accuracy and F1 Score

```
training_time: 0.003 s
prediction time: 0.002 s
Accuracy with SVM top5 feat:  0.7045454545454546
Precision with SVM top 5 feat:  0.21428571428571427
Recall with SVM top 5 feat:  0.6
F1 Score with SVM top 5 feat:  0.3157894736842105
```

Naive Bayes – same result

```
training_time: 0.001 s
prediction time: 0.001 s
Accuracy with Naive Bayes top 5 feat:  0.8863636363636364
Precision with Naive Bayes top 5 feat:  0.5
Recall with Naive Bayes top 5 feat:  0.4
F1 Score with Naive Bayes top 5 feat:  0.4444444444444445
```

Decision Tree – slightly improvement overall

```
training_time: 0.001 s
prediction time: 0.001 s
Accuracy with Decision Tree top 5 feat:  0.8636363636363636
Precision with Decision Tree top 5 feat:  0.3333333333333333
Recall with Decision Tree top 5 feat:  0.2
F1 Score with Decision Tree top 5 feat:  0.25
```

Random Forest – considerable improvement in F1 score (precision and recall) and better accuracy

```
training time: 0.016 s
predicting time: 0.003 s
Accuracy with RandomForest top 5 feat:  0.9090909090909091
Precision with RandomForest top 5 feat:  0.6666666666666666
Recall with RandomForest top 5 feat:  0.4
F1 Score with RandomForest top 5 feat:  0.5
```

Now lets try our classifiers with the feature we have created.

Checking Classifiers with new feature

In this scenario, we will use poi plus our top 1 feature and our new feature. So we will have: poi, fraction poi emails and salary.

SVM LinearSVC – decreased F1 Score

```
training_time: 0.003 s
prediction time: 0.001 s
Accuracy with SVM new feat:  0.7045454545454546
Precision with SVM new feat:  0.16666666666666666
Recall with SVM new feat:  0.4
F1 Score with SVM new feat:  0.23529411764705882
```

Naive Bayes – same result

```
training_time: 0.0 s
prediction time: 0.001 s
Accuracy with Naive Bayes new feat:  0.8863636363636364
Precision with Naive Bayes new feat:  0.5
Recall with Naive Bayes new feat:  0.4
F1 Score with Naive Bayes new feat:  0.4444444444444445
```

Decision Tree – same result

```
training_time: 0.0 s
prediction time: 0.0 s
Accuracy with Decision Tree new feat:  0.8636363636363636
Precision with Decision Tree new feat:  0.3333333333333333
Recall with Decision Tree new feat:  0.2
F1 Score with Decision Tree new feat:  0.25
```

Random Forest – decreased result

```
training time: 0.017 s
predicting time: 0.003 s
Accuracy with RandomForest new feat:  0.8863636363636364
Precision with RandomForest new feat:  0.5
Recall with RandomForest new feat:  0.2
F1 Score with RandomForest new feat:  0.28571428571428575
```

Checking Classifiers with top 5 features from SelectKBest

As we saw, the new feature didn't improve our results, what did was using the top 5 features. We had a good results with Random Forest, but I believe we can get even better, maybe using SelectKBest.

After user SelectKBest, I get to another top 5 of features: poi, total_payments, exercised_stock_options, bonus, restricted_stock_deferred.

Lets try again and see if we get some improvement.

SVM LinearSVC – better accuracy but lower F1 Score

```
training_time: 0.002 s
prediction time: 0.003 s
Accuracy with SVM with SelectKBest:  0.75
Precision with SVM with SelectKBest:  0.2
Recall with SVM with SelectKBest:  0.4
F1 Score with SVM with SelectKBest:  0.26666666666666666
```

Naive Bayes – same result

```
training_time: 0.001 s
prediction time: 0.003 s
Accuracy with Naive Bayes with SelectKBest:  0.8863636363636364
Precision with Naive Bayes with SelectKBest:  0.5
Recall with Naive Bayes with SelectKBest:  0.4
F1 Score with Naive Bayes with SelectKBest:  0.4444444444444445
```

Decision Tree – same result

```
training_time: 0.001 s
prediction time: 0.001 s
Accuracy with Decision Tree with SelectKBest:  0.8409090909090909
Precision with Decision Tree with SelectKBest:  0.25
Recall with Decision Tree with SelectKBest:  0.2
F1 Score with Decision Tree with SelectKBest:  0.22222222222222224
```

Random Forest – still good accuracy, but f1 score decreased considerable

```
f1_score(labels_test, pred)
training time: 0.016 s
predicting time: 0.003 s
Accuracy with RandomForest with SelectKBest:  0.8863636363636364
Precision with RandomForest with SelectKBest:  0.5
Recall with RandomForest with SelectKBest:  0.2
F1 Score with RandomForest with SelectKBest:  0.28571428571428575
```

Considering all tests made, Random Forest had the best result, using the top 5 features from *DecisionTree.features_importance_*. After it, Naive Bayes came with an average result but it

didn’t change in any test. For this reason, I will pick DecisionTree as second choice and I will try to improve it with GridSeachCV, to find the best parameters.

It is important to observe that our goal in this project is to achieve at least 0.3 in precision and recall. We already had better results with RandomForest, and DecisionTree needs a improvement in the recall. With that said, I believe that if we adjust the parameters we can have this result with the both classifiers.

Checking RandomForest and DecisionTree with adjusted parameters using GridSearchCV

RandomForest final test

The first mix of parameters I got was those below:

```
rfc = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=80, max_features=2, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=3, min_samples_split=12,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

The result was really bad as you can see in the next image:

```
training time: 0.133 s
predicting time: 0.007 s
Accuracy with RandomForest with SelectKBest:  0.8863636363636364
Precision with RandomForest with SelectKBest:  0.0
Recall with RandomForest with SelectKBest:  0.0
F1 Score with RandomForest with SelectKBest:  0.0
```

I didn’t change the string, that’s why it says ‘with SelectKbest’, but actually was using top 5 features and the parameters above.

Maybe its better to simplify a little, I will user GridSearchCV looking for only 3 parameters.

The suggestion of GridSearchCV was:

```
rfc = RandomForestClassifier(n_estimators = 10, criterion = 'gini', max_features = 3)
```

And the result:

```
training time: 0.016 s
predicting time: 0.003 s
Accuracy with RandomForest with adjusted parameters:
0.8863636363636364
Precision with RandomForest with adjusted parameters:  0.5
Recall with RandomForest with adjusted parameters:  0.2
F1 Score with RandomForest with adjusted parameters:
0.28571428571428575
```

RandomForest Conclusion

Our best result was using Top 5 features from *DecisionTree.importance_features_*, and default parameters as you can see below:

```
training time: 0.016 s
predicting time: 0.003 s
Accuracy with RandomForest top 5 feat:  0.9090909090909091
Precision with RandomForest top 5 feat:  0.6666666666666666
Recall with RandomForest top 5 feat:  0.4
F1 Score with RandomForest top 5 feat:  0.5
```

DecisionTree final test

Parameters suggested by GridSearchCV:

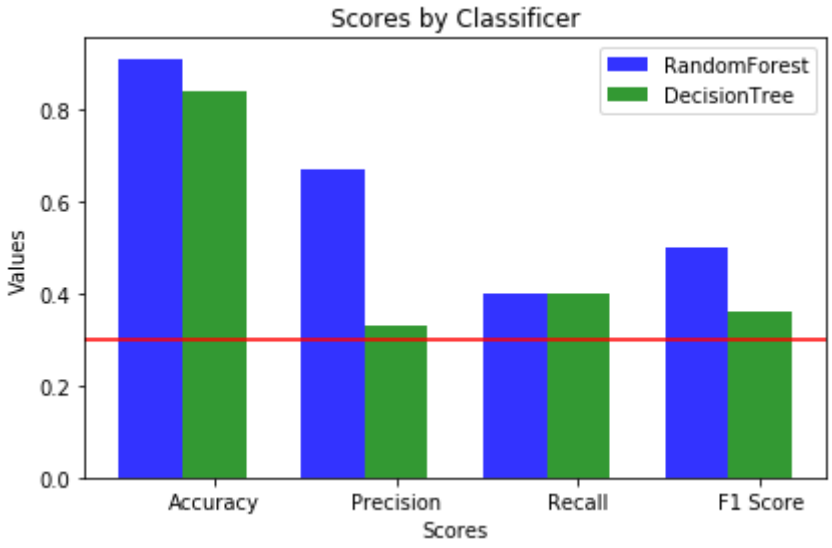
```
dtc = DecisionTreeClassifier(min_samples_split = 2, max_depth = 1, max_features = 1)
```

We could manage to find an appropriate parameter in order to achieve our project goal:

```
training_time: 0.001 s
prediction time: 0.002 s
Accuracy with Decision Tree with adjusted parameters:
0.8409090909090909
Precision with Decision Tree with adjusted parameters:
0.3333333333333333
Recall with Decision Tree with adjusted parameters:  0.4
F1 Score with Decision Tree with adjusted parameters:
0.3636363636363636
```

Precision and recall higher than 0.3, but still are low, but was our best DecisionTree result.

Below a plot comparing best result from RandomForest and DecisionTree



Conclusion

Best result so far was using RandomForest with those data below:

```
#Preparing the chosen classifier...
features_list = ['salary', 'to_messages', 'deferral_payments', 'total_payments', 'exercised_stock_options']
data = featureFormat(my_dataset, features_list, sort_keys = True)
labels, features = targetFeatureSplit(data)

clf = RandomForestClassifier()
t0 = time()
clf.fit(features_train, labels_train)
print "training time:", round(time()-t0, 3), "s"
t0 = time()
pred = clf.predict(features_test)
print "predicting time:", round(time()-t0, 3), "s"
print "Accuracy with RandomForest top 5 feat: ", accuracy_score(labels_test, pred)
print "Precision with RandomForest top 5 feat: ", precision_score(labels_test, pred)
print "Recall with RandomForest top 5 feat: ", recall_score(labels_test, pred)
print "F1 Score with RandomForest top 5 feat: ", f1_score(labels_test, pred)

#training time: 0.016 s
#predicting time: 0.003 s
#Accuracy with RandomForest top 5 feat:  0.9090909090909091
#Precision with RandomForest top 5 feat:  0.6666666666666666
#Recall with RandomForest top 5 feat:  0.4
#F1 Score with RandomForest top 5 feat:  0.5
```

References

I used the site https://scikit-learn.org to check parameters from classifiers and my own mini projects from this course.

I used the Git Hub from user zelite to find a function to calculate my new feature: https://github.com/zelite

I used ProgramCreek website to get examples of how to use SelectKBest: https://www.programcreek.com/python/example/93974/sklearn.feature_selection.SelectKBest