Identify Fraud from Enron Email - *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? [relevant rubric items: "data exploration", "outlier investigation"]**

The goal of this project is to handle a real-word dataset to create, by using the tools learned across the course, an algorithm to make predictions. In concrete, we used a dataset based on Enron data, to create an algorithm for detecting POIs (Persons of Interest), which are the employees suspected of committing fraud.

The dataset consist in 146 data points with financial and email features each data point corresponds to an Enron Employee. The dataset also contains a POI/NO POI feature, which is break apart to use as a label to corroborate the predictions and allow us to calculate metrics. It is also important to note that the number of POI is significantly lower than NO POI. There are 18 POI, which represent around 12% of original dataset and the 13.6% of the outliers free dataset (this scarce POI number had a direct influence in the metric chosen and in the validation process).

For some features I expected missing values (like Deferred Payments), but not for some others like Salary. Anyway, after noting that almost the third part of the employees hadn't the Salary populated (probably due to a hiring mode) I decided to keep these data points, and I took the same decision for the other missing values which are present in all the features across the data points (but in different proportions).

I found 14 outliers. In order to find them, and taking into account that the dataset is relatively small, I first visualized the data and I found three of them

- **TOTAL** (it is not actually an employee, is a line of total amount)
- **THE TRAVEL AGENCY IN THE PARK** (it is clearly not an employee either)
- **LOCKHART EUGENE E** (all features are empty for this employee)

In order to find other outliers, I looked for data points which extreme feature values. POIs where discarded from this search, because precisely the unusual features values which can be useful for the algorithm to detect them. To accomplish this in a precise and trustable way, I used a function which returns, for each feature, the data point which exceeded two standard deviations (considering a normal distribution). By doing this, I found 11 extra outliers.

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? 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. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

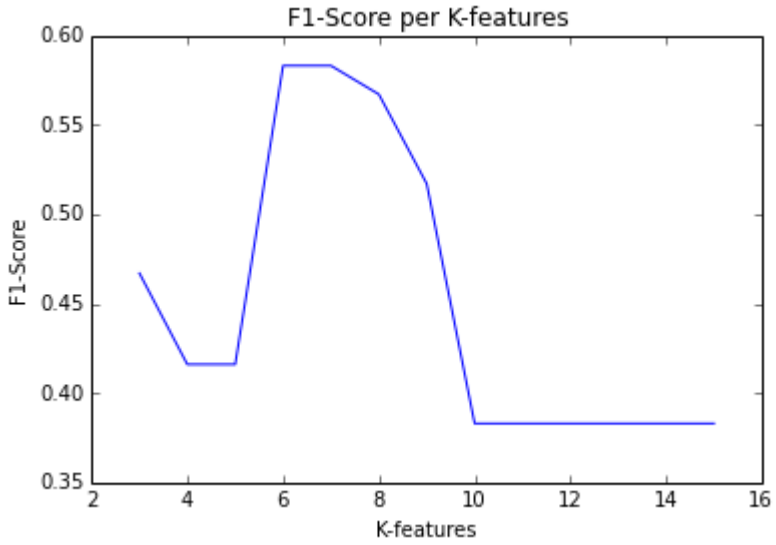
I ended up using the following features:

- Salary
- Long Term Incentive
- Bonus
- Deferred Income
- Total Stock Value
- Exercised Stock Options

I used two processes to select these features:

1) **A manual process** which consisted in discards the features which has not a clear correlation with the POI (like email_address).
2) **An automatic process** which consisted in using SelectKBest with the remaining features from previous step and the two new features.

In order to select the most suitable SelectKBest I just picked up the one which gave me the biggest F1-Score (K=6):



As I used PCA, I used feature scaling to improve its performance. But even if I would have not used PCA, I would end up using feature scaling to improve the SVM predictions. The number of components of PCA was chosen in combination to SelectKBest, to increase de F1-score, and it turned up to be 1.

As regards the new features, I opted for using 'fraction_from_poi' and 'fraction_to_poi' which are mentioned in “Featured Selection” lesson. Each of these two features combines two features from the original dataset in a more meaningful relation. For example, fraction_from_poi, express, as it name indicates, a fraction between from_messages and from_poi_to_this_person which is independent of the actual number of emails, but which proportion of those were from a poi.

I obtained the following features scores by using SelectKBest (in other words the six features most correlated with POI):

Feature	Score
Salary	25.2
Long Term Incentive	23.85
Bonus	36.1
Deferred Income	25.23
Total Stock Value	35.1
Exercised Stock Options	32.1

On the other hand, I took the six features less correlated with POI (not from all the features, just six taken from the subset of 16 which were selected in the manual process):

Feature	Score
Deferral Payments	0.02
Restricted Stock Deferred	1.37
Director Fees	2.18
Fraction From POI	3.48
Expenses	5.84
Load Advances	6.59

3) **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]**

I ended up using SVM, and apart from this one I also tried Decision Tree.

In order to choose between these two models, I evaluated the F1-Score metric which gave me the following result (using cross validation with KFold = 10):

Algorithm	F1-Score
SVM	0.58
Decision Tree	0.42

4) **What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your particular algorithm?** (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: "tune the algorithm"]

It is the process of finding the most suitable arguments in order to make an algorithm to achieve the best predictions for an unknown dataset. If this task is not done properly, the result would be an algorithm which scores poorly.

I've used GridSearchCV, for both my final choice (DecisionTree) and my intial choice (SVM). In both cases I created an array for the most relevant parameters and got the values for the best estimators.

5) What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

Validation consists in separating the data set in training and test data, with the first one we fit the algorithm (in other words, it learns from many examples), and with the test data we check that the algorithm is capable of predict new scenarios (we verify that it has learned properly from the test data).

A classic mistake is to use the test data (in addition to the training data) to fit the algorithm, and as result we obtain better metrics that we should, which finally will contrast with the results predicted in the real world.

In order to validate my algorithm I used K-Fold cross validation with 10 folds (specifically the cross_val_score algorithm imported from sklearn.cross_validation)

6) Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]

In the following table I detail the metrics I used and the results I got for each one:

Metric	Average Perf.
Precision	0.75
Recall	0.5
F1-score	0.58

0.75 of precision means that if 100 POI predictions are randomly taken, 75 of them will be actually POI (true positives), and the remaining 25 will be NO-POI (false positives).

0.5 of recall means that if 100 POI (positive cases) are randomly taken, and the algorithm is applied to them, only 50 cases will be actually predicted correctly (as POI).

F1-score shows a relation between precision and recall, and is a way to combine both metrics in a single one because it expresses a balance between both metrics. For instance, if precision is close to one and recall is close to zero F1-score will be close to zero also, and the same happens in the opposite case. In this particular case (0.58) F1-score shows a reasonable balance between both metrics.