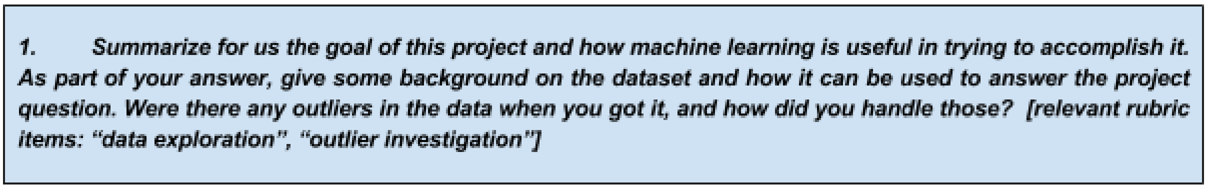
Enron Submission Questions

Student: Peter Carsten Petersen

P5 - Identify Fraud from Enron Email



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 build a predictive model, which from a dataset of "Enron" employee's salary items and Email habits can aid in identifying who was a person of interest "poi" in the criminal investigations which followed on the 2002 bankruptcy of the company.

The dataset provided includes 146 total records(Named Employees) with 21 features per record. There are 18 records labeled as POI's equal to 12,3% of total.

The total number of data points in the dataset which are labeled as "NaN" is 1,323. Given that all but 1 of the features are numerical data, it is unclear whether or not the "NaN" is actual missing data or should be interpreted as 0. For this exercise I have treated and transformed into 0's, but as a general rule further diligence should be applied to investigating cause and type of "missing" Data, as a weak dataset could potentially lead to predictions with little or wrong results.

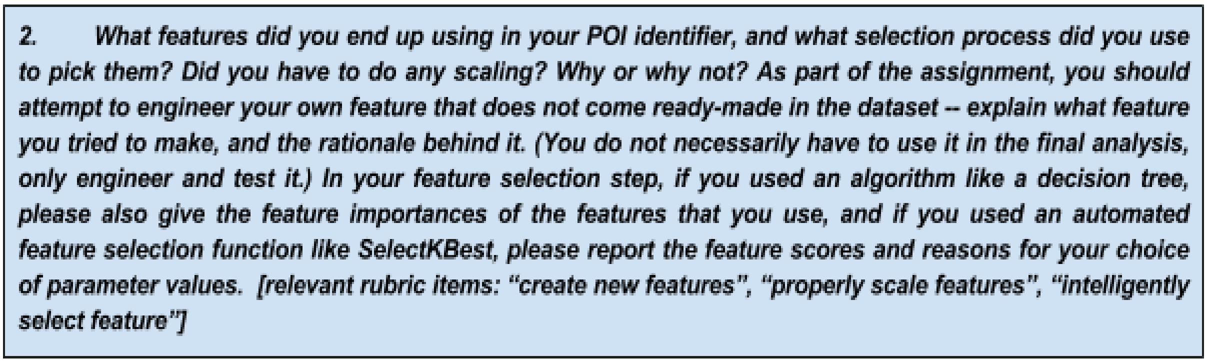
Armed only with this dataset, the task essentially boils down to a highly complex statistical exercise, aimed at identifying possible patterns between the many combinations of features arriving at a choice of algorithm for solving the problem. This could theoretically be done without the assistance of machines but will be a cumbersome and error prone task, and this is where Machine Learning proves to be a powerful tool.

Reaching the project goal employing Machine Learning tools has the following benefits:

1. Calculation time will be barely noticeable and error free compared to a manual process.
2. There are numerous tools available which can aid us in choosing features, scaling data, identifying outliers, splitting datasets for training/testing and reviewing the validity and strength of our result.
3. Armed with this computational power and available tools, we can easily test many possible algorithms, and run many simulations in order to ultimately end at the best possible model for prediction.

Included in the provided dataset is the information if the person is a “poi” or not, i.e. we are already provided with the label of interest. The Project goal is therefore to be reached via a supervised learning algorithm, this will affect the choice of models to be tested.

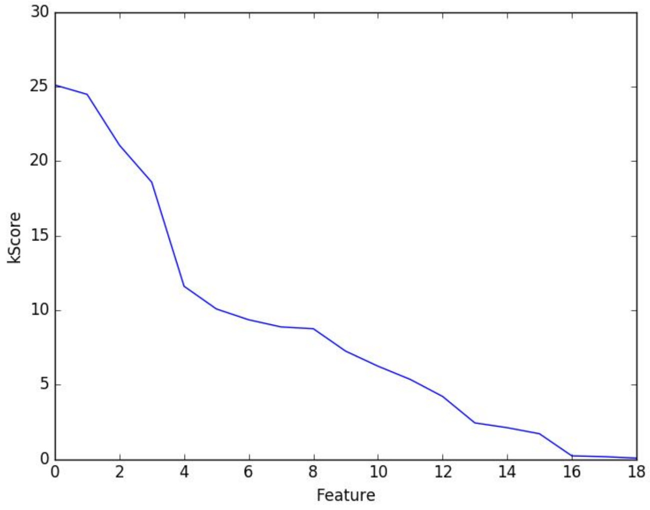
From visual inspection of the dataset as well as from the exercise in “outliers”, we can easily identify and remove 2 particular entries, which are definitely not persons, “Total” which is a summation of all persons data, and “The Travel Agency in the Park”. All remaining “outliers” high and low values I have chosen to keep in the dataset. “High” entries since these are most likely to point towards a “POI”, and “Low” are equally important to keep as removing them will create a dataset more biased towards only possible “POI’s”



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”]

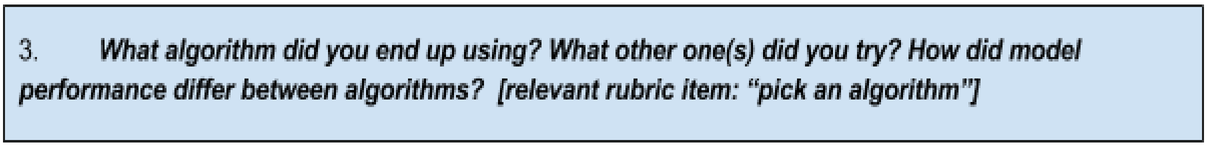
Given that the task is to identify “Persons of Interest”, I created and included in the Features_List a calculated value “Bonus Salary Multiple”, i.e. how many times bigger was respective individuals Bonus over Salary, which I suspect would be a strong indicator for “POI” identification, in other words if you were involved in the “Fraud” you would expect to be paid handsomely via excessive Bonus.

From this extended Features_List I have calculated the respective feature scores using SelectKbest, result in below illustration.



Investigating the respective kScores, there are a number of logical choices for choosing number of features to be included in final model. 2, 4 and 8 seem to show the most obvious drop-off points, in the end the 6 highest scoring features where choosen for the final featurelist as this maximized the accuracy of the final model.

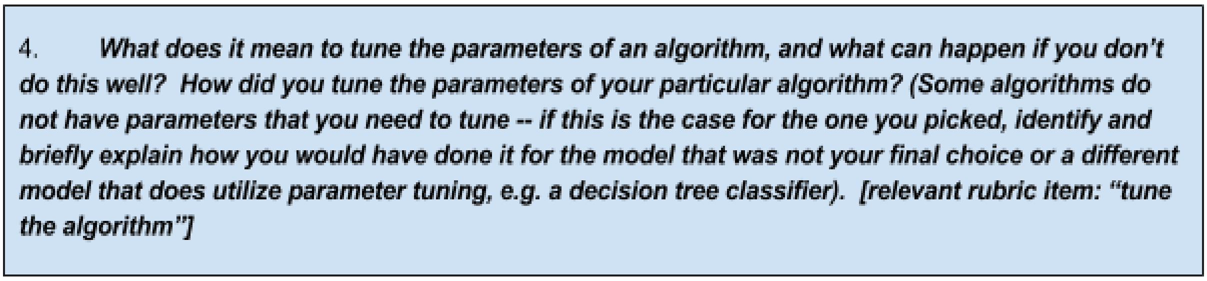
Feature	**Score**
Exercised stock options	25,10
Total Stock Value	24,47
Bonus	21,06
Salary	18,58
Deferred Income	11,60
Bonus Salary Multiple **(new!)**	10,96
Long term incentive	10,07
Restricted Stock	9,35
Total Payments	8,87
Shared receipt with POI	8,75



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"]***

Algorithm	**Pre Tuning**			**Post Tuning and Stratified ShuffleSplit**		
	Accuracy Score	Precision Score	Recall Score	Accuracy Score	Precision Score	Recall Score
GaussianNB	0.926	0.500	0.667	0.871	0.500	0.444
Support Vector Classifier (SVC)	0.929	0.000	0.000	0.900	1.000	0.222
RandomForrest	0.905	0.429	1.000	0.877	0.800	0.444

3 different Algorithms were chosen and while the "SVC" and "RandomForrest" performed better on the dataset post tuning, the GuassianNB was the only model which provided the required scoring on tester.py data, and therefore is the chosen clf for this project.



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"]***

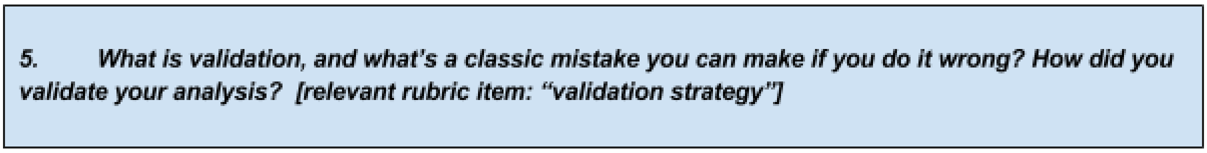
Parameter Tuning is a method of increasing the accuracy/strength of the Algorithm, if done correctly the ML model will deliver a strong validated product

If done incorrectly it can seriously decrease the speed of using the model and/or can lead to a model with low performance.

For all chosen Algorithms I have applied a GridSearch process, where a range of possible parameters have been tested for all possible combinations in order to find the optimal parameters.

Specifically for the GaussianNB which is the chosen "clf", there are no parameters to tune, instead the strategy here was to do a Gridsearch over a range of PCA "Principal Component Analysis" components. A step before PCA was to employ scaling on the chosen features, since the data includes very large variances between lowest and highest values. When this is the case PCA will perform better after data has been scaled. This also evident in tester.py score which was higher after data was scaled.

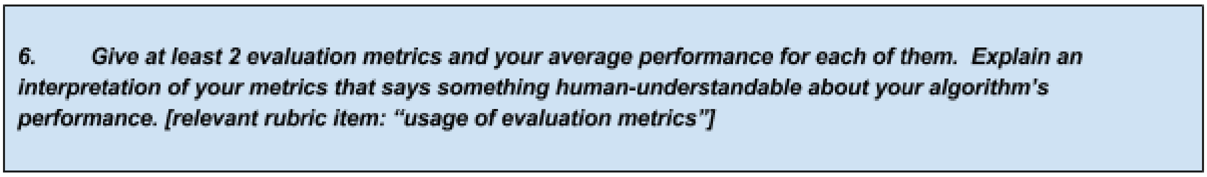
The chosen 'cfl' with scaling, PCA and GridSearch ended being:
Pipeline(steps=[('scaling', MinMaxScaler(copy=True, feature_range=(0, 1))), ('reduce_dim', PCA(copy=True, n_components=3, whiten=False)), ('NaiveBayes', GaussianNB())])



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 is the process of assessing the strength of the Machine Learning Model. This is done via splitting the data into training sets and test sets, so that the Model can be "validated" on the test set after being fitted on the training set. Among classic mistakes can be testing on same data as model has been trained, allowing too many features/classifications of data "Overfitting" the model, or only validating on the "Accuracy" score.

Ideally validation should always be done via splitting data in training/test sets, this done practically by allocating unique portions of data to training and test sets respectively, or running many fitting/validating exercises on Model via randomly splitting data many times in training/test sets. The latter option has been used for this exercise via a "StratifiedShuffleSplit" when running chosen ML algorithms with parameter tuning.



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"]

Algorithm	**tuned Models in tester.py**		
	Accuracy Score	Precision Score	Recall Score
GaussianNB	**0.864**	**0.538**	**0.338**
Support Vector Classifier (SVC)	0.849	0.398	0.111
RandomForrest	0.863	0.578	0.158

Precision Score: Tells us how many of the "POI's" the ML model has predicted in the test set, which were actually classified as "POI's". Score is between 0 (no "POI's" successfully classified") to 1 (all "POI's" predicted were actual "POI's"). This score does not tell us how many of the "POI's" were found. Above score therefore means that of the predicted "POI's" from the ML model 0.394 (39,4%) were also actually classified as "POI's"

Recall Score: Tells us how many of the "POI's" the ML model has predicted in the test set, vs. the Actual amount of "POI's" in the test set. Score is between 0 (no "POI's" in the test test successfully classified) to 1 (all "POI's" in the test set successfully classified). This score does not tell us if all predicted "POI's" were actual "POI's". Above score therefore means that the ML model was able to successfully identify 0.334 (33,4%) of the actually classified "POI's".

So in order to conclude if ML Model is strong it takes a combination of scores close to 1 for both Precision and Recall. I.e. amount of "POI's" predicted had a high Precision and amount of Actual "POI's" predicted was a larger portion of total - Recall.

References:

SKLearn documentation
Udacity Forum discussions
Github repositories on Machine Learning
Wikipedia