Milan Patel

P5: Searching for Fraud in Enron emails

Final project

Overview of data:

Total number of initial features: 19

Total number of used features in final model: 17

Length of data (number of total persons): 146

Length of data used in analysis: 143

Number of POIs: 19

1. The goal of this project is to identify persons of interest for potentially having committed fraud. Machine learning, particularly supervised learning techniques, is useful in this effort because it can effectively explore patterns within the data and attempt to correlate these patterns with expected outcomes.

   Outliers were managed by first exploring the data, by creating a dictionary that showed only the key names and number of missing values. Two key names stood out as not being valid: "TOTAL", and "THE TRAVEL AGENCY IN THE PARK"; additionally, one key stood out as having no values for any of the features, so that was removed as well.

2. I used all features from the two feature groups, except for salary. I did use feature scaling on all features in the model, because KMeansClustering requires use of feature scaling.

   For feature creation, I created two new features: *from_poi_ratio*, which equals *from_poi_to_this_person / from_messages*; and *to_poi_ratio*, which equals *from_this_person_to_poi* / *to_messages.*

   These ratios are intended to measure the proportion of messages sent to and from persons of interest—the idea being that those persons with a high proportion of messages to and from persons of interest have a lot of contact with POIs, and therefore, may possibly be person of interest themselves.

   The results of introducing these ratios into optimized KNearestNeighbors algorithms are as follows:

	Accuracy	**Precision**	**Recall**
no additional fields	0.889	0.62007	0.4325

from_poi_ratio	0.889	0.62007	0.4325
to_poi_ratio	0.8944	0.65805	0.433
both fields	0.89433	0.65708	0.434
best case scenario (with to_poi_ratio, without from_poi_to_this_person and to_messages)	0.9014	0.71547	0.4325

As we can see, the results did not change with the addition of *from_poi_ratio*; however, field *to_poi_ratio* increased the accuracy and precision. I was able to achieve best results by using *to_poi_ratio* and removing *from_poi_to_this_person* and *to_messages* fields.

	Accuracy	**Precision**	**Recall**
KNearestNeighbor (unoptimised)	0.8736	0.57602	0.197
KNearestNeighbor (with PCA)	0.79353	0.22256	0.22
KNearestNeighbor (final)	0.90140	0.71547	0.4325

I used the SelectKBest function in two ways to narrow the list of features from an original list of all features. First, I used the SelectKBest as the first function in a Pipeline before I ran the selected algorithm, and then I used the GridSearchCV function to optimize the K parameter. The GridSearchCV resulted in an optimal parameter of 19 (meaning that all features is the optimal number). After optimal parameters were found for all functions in the pipeline, I used the SelectKBest function again, but this time running a fit_transform to change the features. K values of 18 and 19 both had the same results that optimized the model; as this meant that the extra feature ("salary") was not required, I eliminated from the final model.

Here are the feature scores for all features:

Feature	**Feature Score**
salary	**18.28968**
deferral_payments	**0.224611**
total_payments	**8.772778**
loan_advances	**7.184056**
bonus	**20.79225**
restricted_stock_deferred	**0.0655**
deferred_income	**11.45848**
total_stock_value	**24.1829**
expenses	**6.094173**
exercised_stock_options	**24.81508**
other	**4.187478**

long_term_incentive	**9.922186**
restricted_stock	**9.212811**
director_fees	**2.126328**
to_messages	**1.646341**
from_poi_to_this_person	**5.24345**
from_messages	**0.169701**
from_this_person_to_poi	**2.382612**
shared_receipt_with_poi	**8.589421**
to_poi_ratio	**4.094653**

3. After attempting a number of algorithms including Decision tree and Naïve Bayes, I ended up using the K Nearest Neighbor algorithm. Results as follows:

	Accuracy	**Precision**	**Recall**
Decision Tree	0.7888	0.20445	0.202
Naïve Bayes	0.3362	0.1472	0.83
KNearestNeighbor (unoptimised)	0.8736	0.57602	0.197
KNearestNeighbor (with PCA)	0.79353	0.22256	0.22
KNearestNeighbor (final)	0.90140	0.71547	0.4325

4. Algorithms may have many potential settings that may yield a wide variety of results. Tuning the algorithm means to adjust the parameters of the algorithm to achieve the best desired result—typically highest level of accuracy. I used GridSearchCV to tune my algorithm, setting the parameters and possible values in a dictionary, and feeding those parameters into the algorithms to achieve the highest possible target value (which itself can be set as a parameter; in this case, I aimed for a high recall value because this value felt particularly difficult to get above the desire .3 value.
5. Validation is testing the model against a set of data that was not used in the development of the model, to ensure that the model is generalizable and can be used in circumstances beyond model configuration. One classic mistake is overfitting the data, meaning that the model is created so specifically to the model training data, that the performance of the model against another similar set of data would be significantly lower.

   I validated the data using the pre-built classifier test script (which uses stratified shuffle split cross validation). This method combines the StratifiedKFold and ShuffleSplit methods. Like StratifiedKFold, the data is split into training and test sets based on a fixed percentage, and then the experiment is run k times, after which the results are averaged. Like the ShuffleSplit, the split between training and test sets for each experiment is based on random allocation into each set for each experiment.

6. For the final model, the precision was .7154; this value means that out of all of the instances that were identified as positive, about 72% were correct (“true positives”).
   The recall for the final model was .4325; this value means that out of all true positive items, 43% were identified correctly.