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 identify Enron employees, who may be ‘persons of interest’ (POI), i.e. persons, who might have committed fraud. These POIs should be identified by investigating a dataset comprising Enron financial and email data using machine learning. This dataset includes 146 Enron employees and 21 features representing financial or email data. One of the features is ‘poi’ which can either 1, when the person is known to have committed fraud in the Enron case, or 0 if not. There are 18 known POIs in this dataset, representing ~12% of the total number of people in the dataset. So the vast majority has not been identified as person of interest. Machine learning can now be used to train a classifier algorithm on a subset of the total dataset and then run this classifier on the rest of the dataset. The classifier is used to predict the ‘poi’ label for every person in the dataset based on data of known POIs. This way new persons of interest might be identified if their financial or email data is similar to known POIs.

I find the question on outliers quite hard to answer in this case. I would argue, that for most of the features, I would be the most interested in outliers. To me extreme values in those features would signal a deviation from the norm and therefore might indicate a person of interest. However, going through the entries of the dataset, I found 2 that I decided would be outliers for me and that I subsequently removed. Those 2 entries were: “TOTAL” (which would mess with the max values of each feature) and “THE TRAVEL AGENCY IN THE PARK” (which seems to be a company). Furthermore, I decided to treat entries with too many “NaNs” as outliers and remove those entries. For each person I noted how many features had a value of “NaN” and divided this number by the total number of features. If 75% or more of the features had a value of “NaN”, then this person was removed. This was the case for 22 people, none of which was a known POI, so that in the end the dataset comprised 122 people.

The featureFormat function provided with the course material replaces “NaN” in the data with 0. Thus there are many features with a value of 0, and one can not be sure whether this is an actual zero or whether this has been a “NaN” before. I had the idea for some features, that seemed relevant to me, to impute to zeros with the mean value of this feature. I chose the mean, because for some feature was also zero. But running the classifiers with this imputed data showed little to no effect.

1. 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”, “intelligently select features”, “properly scale features”]

The features that I used in the end are: 'from\_message\_ratio', 'messages\_total', ‘total\_stock\_value’, 'bonus', 'total\_payments\_log', 'long\_term\_incentive\_log', 'bonus\_total\_ratio', 'exercised\_stock\_options', 'incentive\_total\_ratio'.

Out of these 9 features, 6 are features I newly created.

‘Messages\_total’ – the sum of ‘from\_messages’ and ‘to\_messages’, I wanted to capture what the overall message traffic of either mails from or to a person was. (score: 4.63851426e-03)

‘From\_message\_ratio’ – the ratio, that messages sent from a person make up from the overall messages (‘messages\_total’). I created this feature in order to represent whether a person was rather receiving or sending more messages. (score: 3.86882260e-01)

‘total\_payments\_log’ and ‘long\_term\_incentive\_log’ – is the log10 of ‘total\_payment’, ‘salary’ and ‘long\_term\_incentive’ . I introduced them to reduce the range of absolute values for these features. (score: 3.53423653e+00, 3.61630339e+00)

‘incentive\_total\_ratio’ – Is the ratio that ‘long\_term\_incentive’ make up of ‘total\_payments’. I thought if long term incentives make up a significant portion of all payments a person receives, there might be a good reason in trying to tie this person to the company for the long term. (score: 1.13104491e+01)

'bonus\_total\_ratio' – Is the portion the ‘bonus’ makes up of ‘total\_payments’. Similarly I thought if the bonus makes up a significant proportion of overall payments, it might signal a irregularity. (score: 1.07975382e+01)

(scores: total\_stock\_value – 1.53423319e+01, bonus - 9.73033887e+00, exercised\_stock\_options - 1.60526291e+01)

I selected these features using a mixture of running SelectKBest and hand-picking features I thought make sense intuitively. I also looked at missing values for each feature. In the dataset with 122 entries, the features with the most “NaN” values were ‘loan\_advances’ (97% “NaNs”), ‘director\_fees’ (95% “NaNs”) and ‘restricted\_stock\_deferred’ (89% “NaNs”). The features with the least missing values (except for ‘poi’) were ‘total\_stock\_value’ (5% “NaNs”) and ‘total\_payments’ (9% “NaNs”). If a feature is missing values for most of the people in the dataset, it is probably not suited for generalizing. On the other hand, a feature that is only present in a small subset of people might be qualified to make predictions in skewed datasets. The Enron dataset is such a case, were one is looking for few POIs (outliers). Interestingly I tested replacing the ‘total\_stock\_value’ feature with ‘director\_fees’ in a decision tree classifier and the recall/precision rates were very similar.

I ran SelectKBest using various numbers for K and also compared two scoring functions (f\_classif and mutual\_info\_classif). The corresponding recall and precision scores are summarized in the following tables:

F\_classif

|  |  |  |
| --- | --- | --- |
| **K** | **Precision** | **Recall** |
| 4 | 0.28836 | 0.27250 |
| 6 | 0.31652 | 0.25100 |
| 9 | 0.34008 | 0.26050 |
| **9 hand** | **0.41747** | **0.34650** |
| 12 | 0.41805 | 0.34050 |
| 15 | 0.27184 | 0.20850 |

Mutual\_info\_classif

|  |  |  |
| --- | --- | --- |
| **K** | **Precision** | **Recall** |
| 4 | 0.27405 | 0.24500 |
| **6** | **0.44022** | **0.36450** |
| 9 | 0.41941 | 0.34350 |
| 12 | 0.37662 | 0.32050 |
| 15 | 0.33607 | 0.28700 |

The features marked in the f\_classif table as ‘9 hand’ are the ones described above. From the table one can see that the best 12 features from running SelectKBest achieve a similar performance. These features are: 'deferred\_income', 'shared\_receipt\_with\_poi', 'loan\_advances', 'bonus', 'total\_stock\_value', 'from\_poi\_ratio', 'salary', 'total\_payments', 'exercised\_stock\_options', 'bonus\_salary\_ratio', 'incentive\_total\_ratio', 'bonus\_total\_ratio'. I decided against using these features, since they are mainly of financial nature. There is only 1 non-financial feature among those, so I thought the 9 hand-picked features represent a more balanced approach between financial and message features. One can also see from the mutual\_info\_classif table, that the 6 best features as selected by the mutual\_info\_classif perform even better than the 9 hand-picked features. Those 6 features are: 'deferral\_payments', 'expenses', 'shared\_receipt\_with\_poi', 'other', 'bonus', 'from\_poi\_ratio'. I decided against using this feature set since for all the feature list coming from the mutual\_info\_classif a lot of importance is put onto the ‘other’ feature. This feature to me seems meaningless in identifying a POI, it is not comprehensible why the non-descriptive feature ‘other’ should be specific for a POI.

I mainly used a decision tree classifier and naïve bayes classifier, which are not dependent on feature scaling, since they don´t discriminate by distances.

1. 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 a decision tree algorithm. I also tried a Gaussian Naïve Bayes classifier and PCA in conjunction with either a decision tree or Gaussian Naïve Bayes algorithm. The decision tree I used in the yielded precision of ca. 0.40 and recall around 0.32. Interestingly, when I first tried the naïve Bayes classifier with all of the original features and without outlier removal, it showed a very high recall rate of e.g. 0.85 but a quite low precision of only around 0.1. After addition of new features and so on I ended up with precision around 0.43 and recall around 0.28. Running PCA with 8 components before running a naïve Bayes algorithm slightly improved the results to precision around 0.43 and recall around 0.32. The decision tree classifier in conjunction with PCA (3 components) yielded precision around 0.39 and recall around 0.36.

1. 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? What parameters did you tune? (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 items: “discuss parameter tuning”, “tune the algorithm”]

Tuning the parameters of an algorithm means telling it what to base the decisions on. Not doing this well can result in overfitting, where one adjusts the parameters in such a way that they perform well on a fixed test set, but applying this classifier to new, unknown datasets yields bad performance. Tuning the parameters also determines the balance between the bias and variance of an algorithm. For the decision tree classifier, I ended up tuning just one parameter: ‘min\_samples\_split’. This parameter defines how many samples are required for each node to be split into 2 new nodes. The default value is 2, but I assumed that a higher number might make more sense. My idea behind this was, that I want 1 (or few) outliers or POIs from a higher number of unsuspicious people. If min\_samples\_split is relatively high, I thought that there must be a significant difference between 2 groups in order to trigger a split. If you split just 2 samples, slight differences between those 2 might be enough to justify a split. By just trying out different numbers I ended up at min\_samples\_split = 11. I also tried to determine the optimal number for min\_samples\_split (and max\_features) using GridSearchCV but the numbers that I got from this performed worse in my hands than the number I found by randomly trying (or the default value for in case of max\_features). I also played around with the other parameters for the decision tree classifier, but the default values performed best for me.

Here are the results for varying number of min\_samples\_split:

|  |  |  |
| --- | --- | --- |
| min\_samples\_split | Precision | Recall |
| 2 | 0.32993 | 0.35500 |
| 4 | 0.33708 | 0.34450 |
| 7 | 0.36609 | 0.33900 |
| 9 | 0.38126 | 0.35000 |
| **11** | **0.41234** | **0.33750** |
| 13 | 0.42543 | 0.30950 |
| 15 | 0.38974 | 0.25450 |

1. 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 items: “discuss validation”, “validation strategy”]

Validation = split into test and training set + random cross validation????

Validation means splitting your dataset into two parts. One part is used to train the classifier (training set) and the other part is used to evaluate the performance of the trained classifier on an unknown dataset (test set). A classic mistake would be, evaluating the performance of the classifier using the same dataset it was trained on. In this case, one would get perfect but meaningless performance scores. For the split in training and test set, one has to find balance between the sizes of both sets. The training set needs to be big enough, so that the classifier ‘sees’ a broad spectrum during training. On the other hand, the test set also has to have a certain size in order to be representative for a scope of unknown cases. For my analysis, I used the stratifiedShuffleSplit that is given in the test\_classifier function. I tried different test sizes and different random states to make sure I don´t optimize for one certain test set only. I settled for a test size of 0.15. Recall and precision was slightly better for the default size of 0.1, and decreased for test sizes >= 0.3. The stratifiedShuffleSplit function is a form of cross-validation, where validation is run multiple times (folds = 1000) and for each run a different part is set aside as test set.

1. 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”]

The evaluation metrics I used for the project, were ‘recall’ and ‘precision’ (calculated by the test\_classifier function provide). The decision tree classifier had an average recall rate of 0.32 and a precision of 0.40. The Naïve Bayes classifier in conjunction with PCA showed an average recall rate of 0.31 and precision of 0.43.

Recall is the proportion of true positives that were identified correctly. Precision is the proportion of cases that were identified as positive and turned out to be actually true positives. So in this case it means that around one third of actual POIs were identified and that for persons identified to be a POI there is a ca. 40% chance that they are actually a POI.