Enron decision-making

Question 1:

Description:
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 the project is to create a learning model in which to identify persons of interest from the Enron employees. A person of interest in this context indicates a person who could have been of interest in the Enron financial fraud case.

The data used in this task consisted of financial information and information derived from the email messages of a sample of Enron employees. In the dataset, each row consisted of an employee of Enron. The employees had the following fields:

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
from_poi_to_this_person

The dataset is very tiny, with only 145 rows, 18 of which are POIs. The lack of data and heavy bias towards the output variables being non-POIs are both important to know when creating a learning model based on this data.

This data was analysed to explore the data and to see if there are any outliers or anomalies. From the data, the "employees" TOTAL and THE TRAVEL AGENCY IN THE PARK were omitted as they were not considered Enron employees (and TOTAL was deemed to be a heavy outlier).

Additionally, a significant amount of null-values were discovered in the data. This is understandable as the data is mostly financial data of the employees, therefore many of the fields contained financial values not relevant for many employees.

Question 2:

Description:
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"]

For the dataset, I created one additional feature: email_ratio_with_poi. This feature was basically the ratio of email communication made with a poi divided by all email communication. It followed this equation:

$$X_{email\_from\_poi\_ratio} = \frac{X_{from\_poi\_to\_this\_person} + X_{to\_poi\_from\_this\_person}}{X_{from\_messages} + X_{to\_messages}}$$

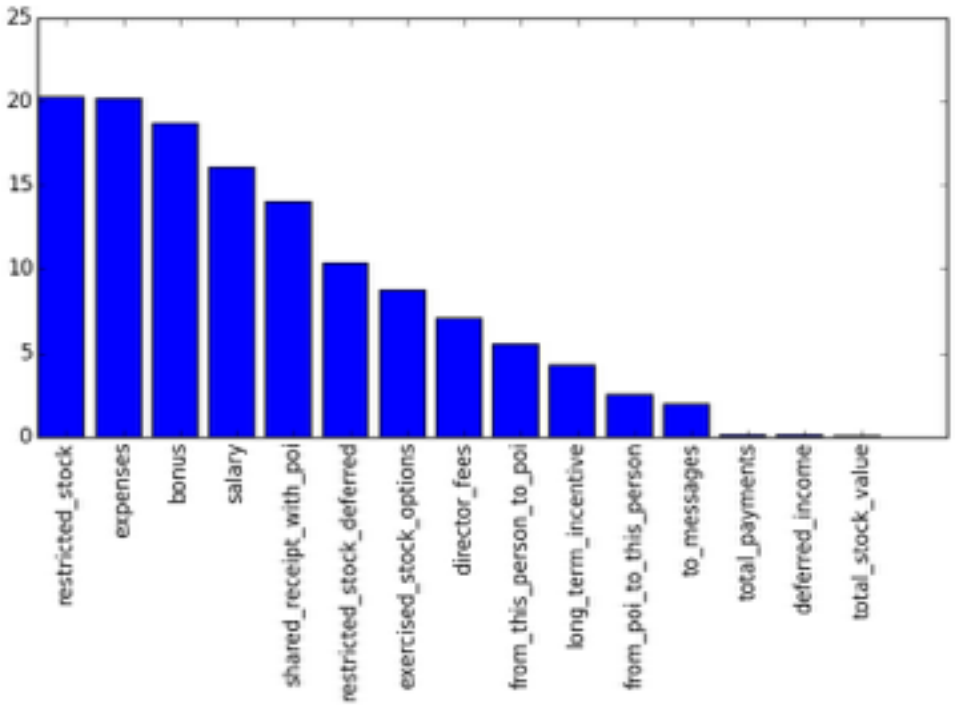
Therefore the total number of features in the dataset was 21. Feature scaling was used because it made data visualisations easier (no huge changes between the values of different features) and because some of the learning algorithms (such as SVM) required for the data to normalised. And also in algorithms such as PCA, the feature scaling makes the algorithms perform significantly better. Min-max normalisation was used with the following formula:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

In choosing the final features, the following strategy was used:
- Start with all features
- Run SelectKBest for all features, plot results
- Pick the optimal set of features (**k** best ones).

I also tried to discover meaningful features by using PCA using the following strategy:
- Run PCA with the whole feature-set with different amount of components (**n**).
- Pick the optimal **n**.

The optimal feature/PCA setup was chosen by running the whole setup using a hand-selected set of algorithms (more on those in Question 3) and picking the setup that gave the best values for the selected test statistics (more on that in Question 5).



What I found out was that the optimal setup was using k best features, when

$$k = 8$$

The feature scores from SelectKFeatures can be seen in the figure below.

Using PCA with the original dataset (without the new feature) when

$$n = 2$$

The principal components explained [0.8237812 0.11490845] of the output variable variance in the training set. Therefore the features could be reduced into a quite a low-dimensional format.

Funnily enough, the test statistic (F2 was used) was exactly the same for both cases. It was 0.58752.

Unfortunately, the additional hand-selected feature was not a part of that optimal setup and hence was not suitable for use.

Question 3:

Description:
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 tried the model using three different algorithms:
- Naive Bayes
- SVM (Support Vector Machines)
- Random Forests with n = 10 and n = 20, where n is the amount of decision trees.

During the analysis, Random Forests were quickly dropped as the testing of the model (using the testing function in tester.py) was significantly slower than SVM or NB and consistently produced significantly worse test statistic-values than the previous two.

After running multiple tests using both different feature-setups and different amounts of PCA components, SVM seemed to always come ahead of NB. Therefore SVM was decided for the final model. The tests can be seen in folder **performance-tests/**.

Question 4:

Description:
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"]

As mentioned in Question 3, the algorithms used were Naive Bayes, SVM, and Random Forests. These were the validation methods used for the algorithms:

For Naive Bayes, no different parameters were used so no validation for the hyperparameters was necessary.

For SVM and RF, the optimal hyperparameters were selected using cross validation, namely GridSearch.

For SVM, the optimal kernel, C, and gamma parameters were selected. The following were considered optimal in the final model:

Kernel	Gamma	C
rbf	1	1

For RF, the optimal criterion, min_sample_split, and max_features were considered. The following were considered optimal:

Criterion	min_sample_split	max_features	Number of trees
rbf	1	1	20

The number of trees was selected by intuition as the amount of trees needed not be too high for a dataset this simple. Probably even using 20 trees was significantly overdoing it.

Question 5:

Description:
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 can be said to involve the method of testing how the learning model handles data it has not previously seen. This is an effective way to create a setting in which the way a model can handle real, unknown data can be quantified using a properly chosen validation metric. To this end, the dataset is usually divided into training,test and validation sets. The training set is used to actually train the model. The validation set is used to determine the hyper parameters the model uses. And finally, the test set is used to test how well the model functions when its given data it has not seen before.

A classic mistake in validating a learning model can be to use a same subset of the data to train, validate, and test the model. This way the model will be severely prone to overfitting on that data subset. Another classic mistake is to use either training or testing data to validate hyper parameters of the model. This is a very common mistake in machine learning tasks. Actually, in this project I took the freedom to follow this anti-pattern and used the training set to validate my hyper parameters, but I only did that because during the project the dataset seemed far too small to divide it into three different subsets.

In this project, two main evaluation metrics were used to evaluate the learning model: F1 and F2. Evaluation metrics based on Precision and Recall was chosen because the datasets dependant classification variable was highly skewed towards negative classifications. Therefore, if using simple squared error-based metrics, simply creating a classifier that classifies each data row as negative would have produced good results.

Therefore, to emphasize positive results, Precision and Recall-based methods were chosen as evaluation metrics. Since both Precision and Recall are important in this learning task, F1 was chosen because it combines both metrics into a single metric. The formula for the F1-metric was the following:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

F2 was chosen, and in fact used as the main metric because it emphasises the Recall-value of the model. Recall was emphasized because I argue that it is far more severe to classify false negatives than it is to classify false positives. This is important in order to minimize the chance that anyone gets away with committing corporate fraud. The formula for F2 was the following:

$$F2 = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) * recall}$$

Where β = 2.

Question 6:

Description:
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 model gave the following evaluation metrics:

Optimize	Algorithm	Features	Precision	Recall	F1	F2
F1	SVM	director_fees, salary, from_poi_to_this_person, expenses, to_messages shared_receipt_with_poi	0.32339	0.69400	0.44120	0.56459
F2	SVM	PCA, n = 2 no new feature	0.28564	0.79850	0.42076	0.58752

Table described the setups and the metrics for models were either F1 or F2. These two metrics were documented for the following reasons:

- Optimal F2-setup was documented because I felt it was the metric that serves the original research question the best (as argued in Question 5). Unfortunately, with the F2-optimized setup, the Precision-value falls below 0.3, which brings me to my reason of including F1.
- F1 was used because it didn't emphasise Recall in a way that it caused the Precision-value to be above 0.3. Precision over 0.3 was deemed valuable as it was determined in the original project specifications.

Despite the fact that the F1-optimized model was included, I still regard the F2-optimized as the model-to-beat. In this model, at least according to the test data, almost 8 out of 10 people who commits corporate fraud will be classified as POIs. Naturally this means that there will be some more false positives (people who are mislabeled as POIs), but I feel that's an acceptable tradeoff to make sure as few false negatives as possible (actual positives will then be classified correctly as POIs).

But to make sure that the final model passes the specs required, the F1-optimized model was used in poi_id.py.

Resources and References

https://www.udacity.com/course/viewer#!/c-ud120-nd/l-2410328539
http://scikit-learn.org/stable/documentation.html