**Documentation**

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 use machine learning to detect fraudsters from the Enron dataset. The Enron dataset contains a large quantity of emails between Enron employees along with data about employee financial compensation, e.g. salary, bonuses, etc. Each person of interest is labelled so we have a supervised learning problem. The Enron data set contains 146 persons, 18 of which are poi's. Each person in the dataset has 18 features. Across the total dataset, the number of NaN's for each feature are:

{'bonus': 64,

'deferral\_payments': 107,

'deferred\_income': 97,

'director\_fees': 129,

'email\_address': 35,

'exercised\_stock\_options': 44,

'expenses': 51,

'from\_messages': 60,

'from\_poi\_to\_this\_person': 60,

'from\_this\_person\_to\_poi': 60,

'loan\_advances': 142,

'long\_term\_incentive': 80,

'other': 53,

'restricted\_stock': 36,

'restricted\_stock\_deferred': 128,

'salary': 51,

'shared\_receipt\_with\_poi': 60,

'to\_messages': 60,

'total\_payments': 21,

'total\_stock\_value': 20}

Outlier's were investigated visually using a pairwise scatter plot matrix. The pairwise plot matrix shows a clear outlier in each of the plots. This outlier was explored with a function max\_feature() and max\_person() which each take a feature as an input and return the maximum value of a feature, and the person with the maximum value. Since the outlier was found to return the summary statistic TOTAL, this was removed with the remove() function. A scatter plot of salary and bonus was plotted to explore if additional outliers existed, which they did. The largest remaining outliers in each of the financial\_features list were printed using the max\_person() function. Since these were all people, these outliers were kept in the dataset for further processing.

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 doesn’t 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]

**Create new features:**

A new financial feature called non\_salary was created which involved summing all financial features excluding total\_compensation and salary as well as the poi identifier. The rationale behind creating this feature is that different poi’s may have exploited different non\_salary compensation mechanisms to increase their total compensation. The total compensation feature might not bring this feature out since some non-poi employees with a high salary with limited non\_salary compensation could have similar total compensation as the poi’s with high non\_salary compensation. The evaluation metrics with and without this new features are summarized in the answer to question 6 below.

**Properly scale features:**

Financial features were scaled using the MinMaxScaler function. Feature scaling is used to standardize the range of values in each of the features so that the classification algorithm does not put more weight on financial features that have a large range of values compared to financial features with a smaller range of values. This is because most classification algorithms use the Euclidian distance between data points in their algorithm.

**Intelligently select features:**

Financial features from the dataset were manually selected and stored in the *financial\_features* list. For financial features where the total count of NaNs for a feature was greater than 50% of the total data points, this feature was excluded from the assessment. After scaling the features using the MinMaxScaler(), the final features were selected from the financial features subset using SelectKBest(chi2, k=4).

3. What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: “pick an algorithm”]

KNeighborsClassifier (n\_neighbors=3, weights='uniform', algorithm='auto', leaf\_size=10, p=1, metric='minkowski') was used as the tuned algorithm. Feature scaling was done using MinMaxScaler() and feature selection was done with SelectKBest (k=4 was the tuned k value with the features and their scores highlighted below). GridSearchCV was used for parameter tuning. Results for the evaluation metrics were:

Accuracy: 0.88462 Precision: 0.74272 Recall: 0.38250

Features\_scores using SelectKBest were as follows:

|  |  |
| --- | --- |
| **Feature** | **Score** |
| **bonus** | **5.193349** |
| deferral\_payments | 0.058561 |
| deferred\_income | 0.341726 |
| director\_fees | 1.489217 |
| **exercised\_stock\_options** | **6.927477** |
| expenses | 1.525657 |
| **loan\_advances** | **6.742748** |
| long\_term\_incentive | 2.57969 |
| restricted\_stock | 0.595438 |
| restricted\_stock\_deferred | 0.003456 |
| salary | 3.116644 |
| total\_payments | 2.811509 |
| **total\_stock\_value** | **5.54484** |
| non\_salary | 4.291605 |

Other classifiers explored with parameter tuning included Adaboost, RandomForestClassifier, and LinearSVC. Random forest classifiers and Adaboost were found to be much slower for this classification project. Adaboost and Random forest classifiers were not selected as the final tuned algorithm due to their slower performance. Feature selection was explored for LinearSVM with both SelectKBest and PCA while for other classifiers, features selection was limited to SelectKBest.

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 don’t 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 if you used, say, a decision tree classifier). [relevant rubric item: “tune the algorithm”]

**Tune the algorithm:**

Each classifier explored in the assessment has one or more parameters related to the structure of the algorithm. 'Tuning' these parameters affects the score of our classifier based on our selected evaluation metrics so it is important to explore how changes to the default parameter assumptions influence relevant evaluation metrics. If you don’t tune the parameters, you are going to end up with a model that has poor performance relative to a parameter tuned model.

GridSearchCV with StratifiedShuffleSplit(n\_iters=100) as the cross validation method was used to tune parameters of the algorithms identified above. StratifiedShuffleSplit was found to be valuable as a cross-validation method because it preserves the ratio of targets to features in each cross-validation ‘fold’. This is important when the frequency of a target class (e.g. poi’s) in the dataset is low because having randomly selecting few of a target class of interest (for us poi’s) in a random fold will limit the effectiveness of model training. GridSearchCV takes in as arguments that algorithm, a cross-validation method, a scoring and a dictionary of key:value pairs that include the parameters to be tuned, and an array of values for each of these parameters. For each algorithm, accuracy, precision and recall scores were passed as scoring metrics for GridSearchCV to optimize.

Below are the parameters and scores achieved after parameter tuning with GridSearchCV.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | LinearSVM | LinearSVM | Adaboost | RandomForestClassifier | KNearestNeighbor |
| SelectKBest(k=) | 13 | 2 | 11 | 5 | 4 |
| SelectKBest(score\_func=) | chi2 | f\_classif | f\_classif | f\_classif | chi2 |
| PCA(n\_components=) |  | 1 |  |  |  |
| C | 1000 | 10 |  |  |  |
| n\_estimators |  |  | 100 | 10 |  |
| max\_depth |  |  |  | None |  |
| min\_samples\_split |  |  |  | 5 |  |
| min\_samples\_leaf |  |  |  | 1 |  |
| max\_leaf\_nodes |  |  |  | 7 |  |
| n\_neighbors |  |  |  |  | 3 |
| weights |  |  |  |  | uniform |
| leaf\_size |  |  |  |  | 10 |
| p |  |  |  |  | 1 |
| accuracy | 0.78 | 0.79593 | 0.85067 | 0.86387 | 0.89087 |
| precision | 0.2191 | 0.23328 | 0.41004 | 0.47506 | 0.66667 |
| recall | 0.2535 | 0.232 | 0.2735 | 0.2 | 0.363 |

Note that these scores are derived from passing SelectKBest through the pipeline which implies that different k best parameters are likely to be selected through each fold of the validation procedure. Results presented above for the final algorithm extracted the tuned parameters for SelectKBest and used these to select the best features that were passed to the final algorithm, e.g.:

* tunedParams = clf["precision"].best\_params\_.copy()
* selector = createSelector(tunedParams)
* kBestFeatures = selectFeatures(selector).

createSelector initiates SelectKBest using the tuned parameters returned from GridSearchCV, and selectFeatures returns the features selected from SelectKBest.

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 strategy:**

Validation involves training our model and exploring how good this trained model is, in terms of how effective it is at providing the correct prediction, when we throw new data at it, as judged by specific evaluation metrics. A classic validation mistake is to train a model using all of the data which leads to overfitting. To avoid overfitting, sklearn’s cross validation method StratifiedShuffleSplit was used to split the data into training and testing sets when using GridSearchCV. StratifiedShuffleSplit provides a set of randomized training/testing sets ensuring that the target class ratio (i.e. poi’s to non-poi’s) of the training and testing datasets are the same as in the original dataset. The test\_classifier() function also applies StratefiedShuffleSplit when evaluating the performance of the model. As pointed out above, ensuring the ratio of each target class are equivalent in the training and testing sets are similar to those in the original dataset is important when the number of a particular class is low since this affects the validity of a model. In the extreme case, if no poi’s are in a given fold, we are unable to develop a useful model to predict poi’s from this fold and this will influence the scoring metrics of the final algorithm.

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”]

Three performance metrics were explored. Scores with and without the new non\_salary feature are provided below with the score prior to implementing the non\_salary feature provided in brackets:

\*accuracy = 0.884(0.850)

- the percentage of poi's that were correctly identified (true positives / total people)

\* precision = 0.631 (0.392)

- when our model predicts a poi, how likely is it that this prediciton is correct (true positives/(true positives + false positives))

\* recall = 0.316 (0.222)

- when the true classification is poi, the likelihood that our model correctly predicts poi's (true positives/(true positives + false negatives))

Introducing the non\_salary feature led to a slight increase in the accuracy metric, precision increased by almost 25% and recall increased by almost 10%.