# Enron POI Project

**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 is project is to identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset.

Machine learning will enable us in the identification process based on an algorithm for the given dataset.

The dataset consists of data for 146 people and each having data for 26 features consisting of financial, email and POI (Person Of Interest) information. There are 18 persons who are identified as persons on interest (12 % of the people).

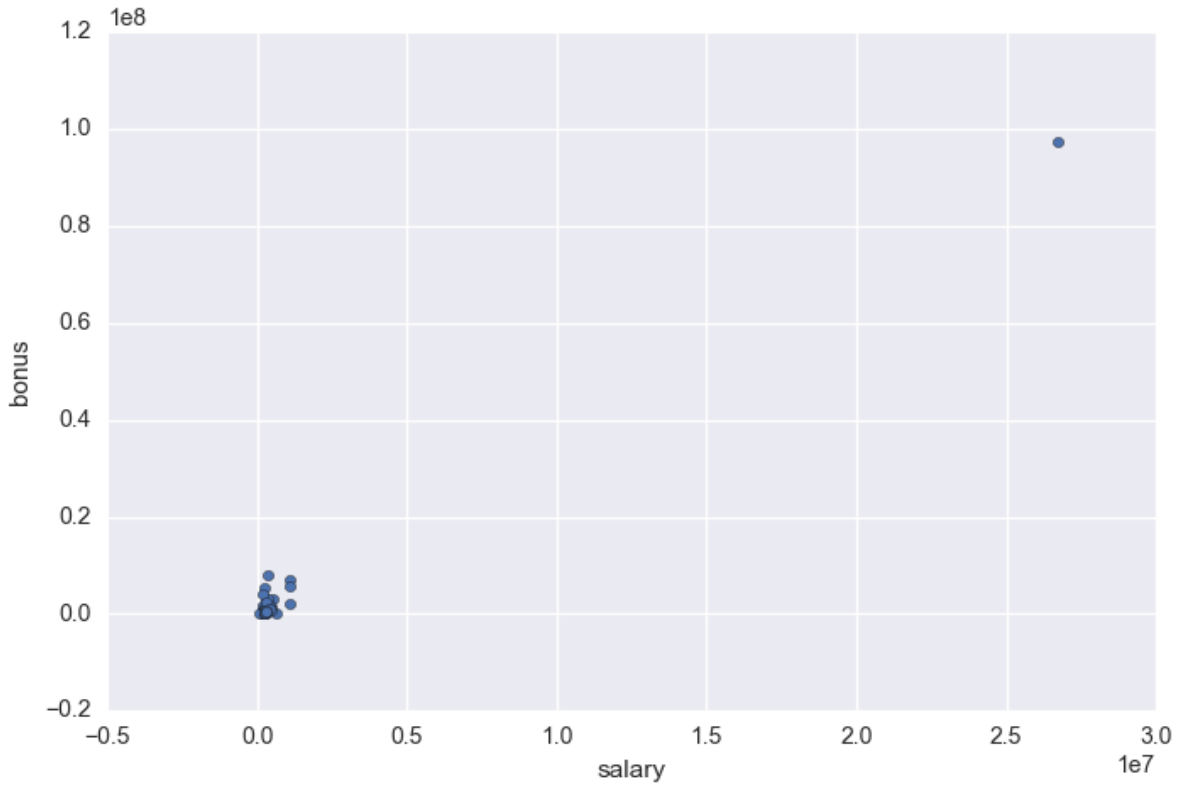
There are lot of values which have NaN's for both financial and email features. For financial data I replaced NaN with 0 but for email data it is not as straight forward. It might be an actual case of missing data. I decided to fill in the missing data using the median values of the columns grouped by POI class.

Count of Missing Values

```
salary                      51
deferral_payments          107
total_payments              21
loan_advances              142
bonus                       64
restricted_stock_deferred  128
deferred_income             97
total_stock_value           20
expenses                    51
exercised_stock_options     44
other                       53
long_term_incentive         80
restricted_stock            36
director_fees              129
to_messages                 60
from_poi_to_this_person     60
from_messages               60
from_this_person_to_poi     60
shared_receipt_with_poi     60
```

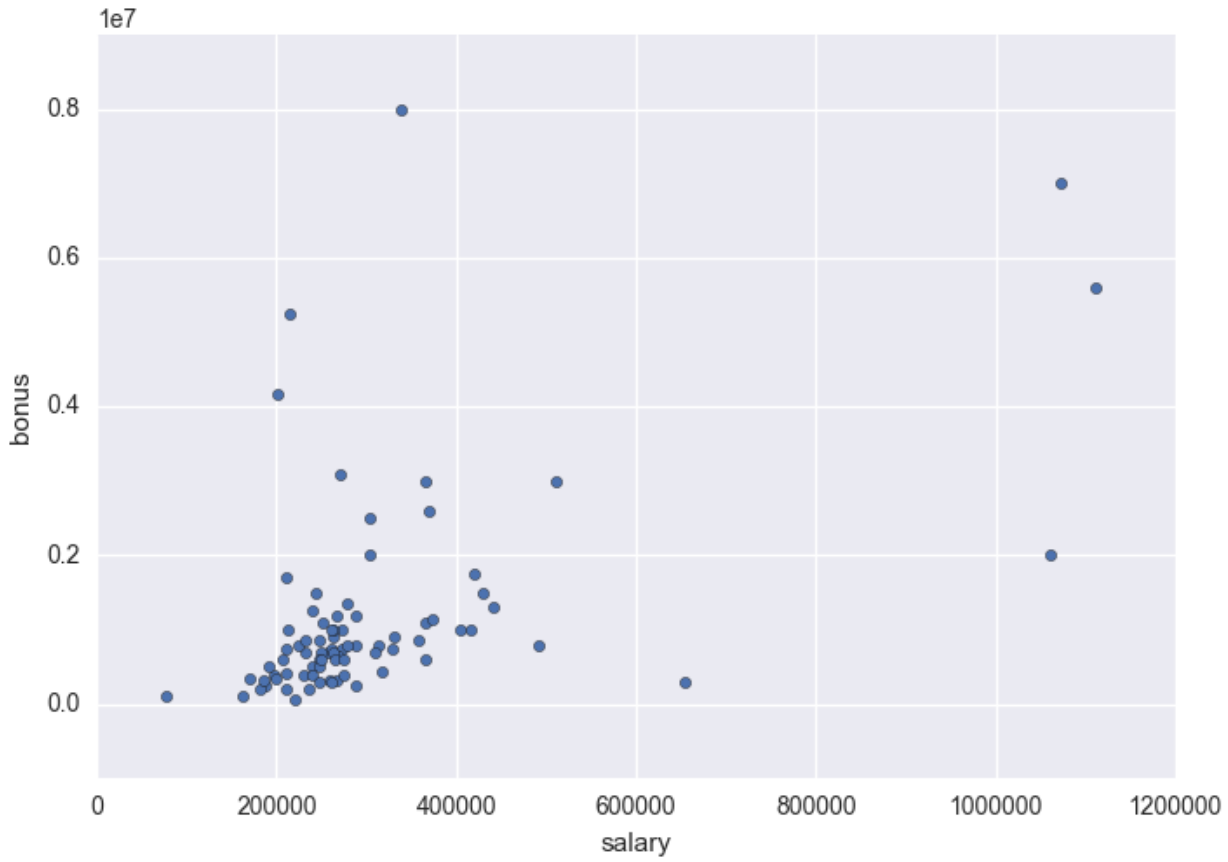
Outliers

I used a scatter plot to plot salary vs bonus to check to outliers -




This plot show shows one obvious outlier, after further investigating it was found to be an error in the dataset as that record ('TOTAL') is the total row at the end of the financial data pdf.

After removing the 'TOTAL' record and plotting again -




The plot looks a little better now but still has some points in the far upper corners. This could be that some people may have far higher salaries and bonuses that others, possibly involved in fraud.

There is one other record 'THE TRAVEL AGENCY IN THE PARK' which according to the pdf is not a person but a company. I decided to remove it as well.

**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"]**

I ended up using the following **19** features (displayed with the SelectKBest scores) –

| **Feature** | **Score** |
|---|---|
| fraction_to_poi | 25.7935 |
| exercised_stock_options | 25.0975 |
| total_stock_value | 24.4677 |
| bonus | 21.0600 |

| | |
|---|---|
| bonus_to_total | 20.9888 |
| salary | 18.5757 |
| shared_receipt_with_poi | 14.6130 |
| deferred_income | 11.5955 |
| bonus_to_salary | 10.9556 |
| long_term_incentive | 10.0725 |
| restricted_stock | 9.3467 |
| total_payments | 8.8738 |
| loan_advances | 7.2427 |
| from_poi_to_this_person | 6.8708 |
| expenses | 6.2342 |
| other | 4.2462 |
| from_this_person_to_poi | 2.6171 |
| fraction_from_poi | 2.1493 |
| director_fees | 2.1077 |

Initially I tried to manually select features which have a score greater than 5 which returned 15 features.

Then I used GridSearchCV with SelectKBest to automatically find the optimal number of features for the DecisionTree (using the code below) classifier to arrive at the feature count of 19.

```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline

n_features = np.arange(1, len(feature_list))
sss=StratifiedShuffleSplit(labels,n_iter=100,test_size=0.3)

pipe = Pipeline([
    ('kbest', SelectKBest()),
    ('clf', DecisionTreeClassifier())
])

param_grid = [
    {
        'kbest__k': n_features
    }
]

gs=GridSearchCV(pipe, param_grid=param_grid, scoring='f1', cv = sss)
gs.fit(features, labels);
gs.best_params_
```

Eventually I decide to use **k=19** as it provided better performance.

I decided to create 2 new email features,

fraction_from_poi – fraction of all messages sent from this person to the messages sent to POI

fraction_to_poi - fraction of all messages sent to this person to the message received from POI

Rationale: As email is main source of communication in corporate world and persons who communication more with a POI can themselves be a POI. A relative number makes more sense in this case than an absolute number.

I also added 2 new financial features,

bonus_to_salary - ratio of bonus to salary

bonus_to_total – ratio bonus to total payments

Rationale: I think people who receive high bonuses than their salary or total payments could most possibly be involved in fraudulent activity as illegal funds can be passed off as bonus. Also, from observing the scatter plot(previous section) this seem liked a good feature to have.

Scaling

Scaling was not used as it didn't have any effect on the performance of the classifier I choose, DecisionTreeClassifier.

**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"]**

The algorithm I ended up using is **DecisionTreeClassifier**.

I tried 3 other algorithms –

- Gaussian Naive Bayes (GaussianNB)
- Linear Support Vector Classifier (LinearSVC)
- KNeighbors Classifier (KNeighborsClassifier)

Initially I ran all the algorithms with the default parameters and using all the features (after data cleansing). I used the following metrics to evaluate the performance of the models,

- Accuracy
- Precision
- Recall
- F1 score

1)Here are the results for the first run,

| Algorithm | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| KNeighborsClassifier | 0.86 | 0.09 | 0.04 | 0.06 |
| GaussianNB | 0.43 | 0.17 | 0.78 | 0.27 |
| LinearSVC | 0.87 | 0.18 | 0.11 | 0.14 |
| DecisionTreeClassifier | 0.83 | 0.32 | 0.36 | 0.33 |

From the first run the decision tree performed well.

2)For the second run I add the new features - fraction_to_poi, fraction_from_poi, bonus_to_salary & bonus_to_total. Here are the results,

| Algorithm | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| KNeighborsClassifier | 0.88 | 0.30 | 0.20 | 0.23 |
| GaussianNB | 0.76 | 0.29 | 0.39 | 0.31 |
| LinearSVC | 0.75 | 0.18 | 0.31 | 0.19 |
| DecisionTreeClassifier | 0.90 | 0.61 | 0.58 | 0.57 |

From the second run the decision tree continued to perform very well with good improvement for naïve bayes classifier.

At this point I decided to go with DecisionTreeClassifier and drop the other three classifiers.

3)For the third run I selected the 19 features from the result of the SelectKBest output,

| Algorithm | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| DecisionTreeClassifier | 0.91 | 0.67 | 0.59 | 0.60 |

The performance of the DecisionTreeClassifier with k=19 improved with F1 score of **0.60**

**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"]**

The tuning of parameters of an algorithm is process of changing the settings that the impact model to allow the algorithm to perform optimally for a given dataset.

I used GridSearchCV to automatically select the optimal values from a grid of parameters related to the DecisionTree classifier. By reviewing the documentation for the classifier on the sci-kit learn website I found that there were numerous parameters than can used. I decided to use the following,

- criterion
- max_depth
- min_samples_split
- max_features

Based on the values identified by GridSearchCV I ended up using the following values for these parameter,

- criterion = 'entropy'
- max_depth = 5
- min_samples_split = None

- max_features =20

Here is the code using GridSearchCV to obtain the best parameters,

```
tree_pipe = Pipeline([
    ('kbest', SelectKBest(k=19)),
    ('clf', DecisionTreeClassifier()),
])

# configuration of parameters for Decision Tree Classifier
param_grid = dict(clf__criterion = ['gini', 'entropy'] ,
                  clf__min_samples_split = [2, 4, 6, 8, 10, 20],
                  clf__max_depth = [None, 5, 10, 15, 20],
                  clf__max_features = [None, 'sqrt', 'log2', 'auto'])

tree_gs = GridSearchCV(tree_pipe, param_grid = param_grid, scoring='f1',
cv=sss)
tree_gs.fit(features, labels)
```

The tuning of the parameters improved the performance of the algorithm, the F1 score increased to **0.73**

**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 is a process of evaluating the generalization ability of a trained model. This is done by applying the model to a separate dataset and measuring the outcome.

If validation is not done correctly we run into a situation called **overfitting.** This can lead higher scores on training data and poor results on unseen data.

Given the small size of the dataset I used stratified shuffle split cross-validation to avoid this situation. This allows data to be randomly split into a pair of training and testing data sets while still preserving the same percentage for the class labels as in the original set. Throughout my analysis I used a split size of 100 to evaluate the performance of the algorithms.

I used the *test_classifier* in *tester.py ,*which uses the StratifiedShuffleSplit cross validation method, to assess the performance of the algorithms.

**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"]**

For the DecisionTreeClassifier the final results were,

- Accuracy: 0.92
- Precision: 0.69
- Recall: 0.78

The accuracy metric tells us that the model classified about 92% of the samples correctly.

The precision metric tells us that of all the POI's classified by model about 69% of them were indeed POIs.

The recall metric tells us that model identified 78% of the POIs in the entire dataset.