# Project Report

**Note : All the steps described below are shown in more detail in Enron Email POI Classification notebook . The file is named as Enron Email POI Classification.ipynb**

## 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?

**Background to the dataset**
Enron was an American Organization based in Houston, Texas . Founded in 1985 , this ompany employed 20,000 employees towards the end in 2001 . In December , it was revealed that its reported financial condition was sustained by institutionalized, systematic, and creatively planned accounting fraud, known since as the Enron scandal.

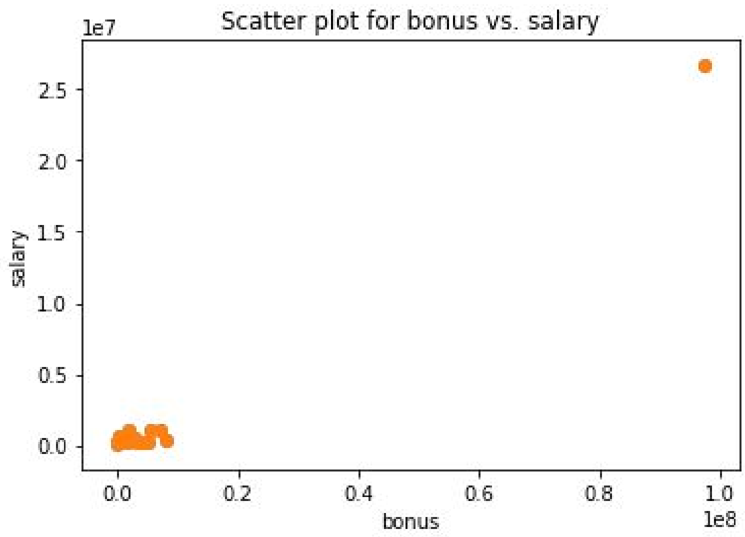
This dataset basically contains information regarding the email sent by it's employees . There are certain people who are marked as Person of Interest (POI) . A person of interest (POI) is someone who was indicted for fraud, settled with the government, or testified in exchange for immunity. This dataset contains over 600,000 emails sent by 158 employees . It was released after the company declared bankruptcy . Detailed financial records of the said employees was also released as part of the dataset .

**Goal of the project and How machine Learning can help**
The goal of the project is to identify POI from the dataset using Predictive Machine Learning Models . Machine Learning allows us to build mathematical models that use the features present in a dataset to make future predictions when it encounter real world data .

The dataset contains financial information and emails of 146 people in our datatset . There are total 35 POI's out of which we have information of about 18 of them . The financial data is missing for the remaining POIs .

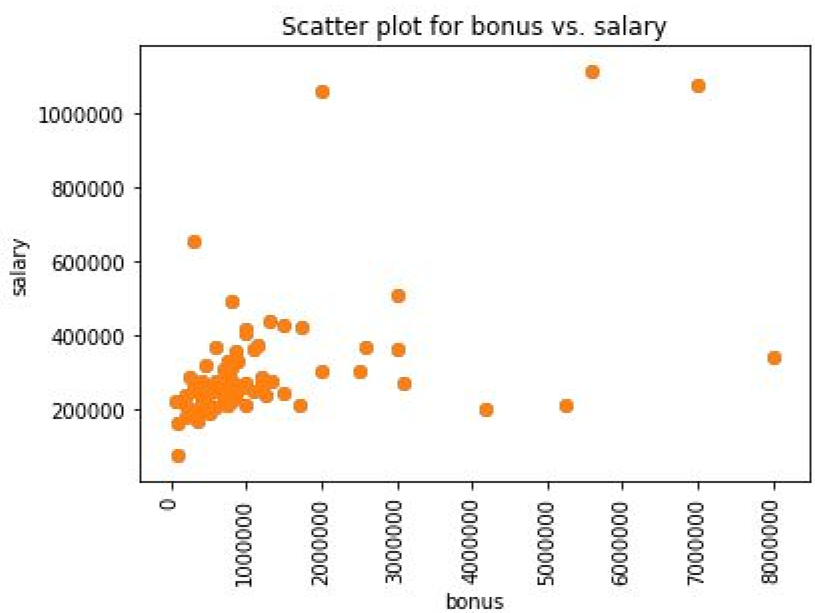
**Outlier Detection**

Below is the scatter plot for salary vs bonus .




The outlier has the name 'TOTAL' , suggesting that it is basically a sum of all the values . Since it does not make any sense for our analysis I have removed it .

After removing the above outlier, we get the following following plot .




After examining the outliers I came to the conclusion that we do not delete the the outliers since these are valid data points .

**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 does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it.**

**Dealing with NaN values .**

In the initial dataset we had the following NaN values in each columns .

```
Count of NaNs in each column after cleaning

bonus                         64
deferral_payments            107
deferred_income               97
director_fees                129
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
poi                            0
restricted_stock              36
restricted_stock_deferred    128
salary                        51
shared_receipt_with_poi       60
to_messages                   60
total_payments                21
total_stock_value             20
dtype: int64
```

Let's get convert the above values into percentage :

```
Percentage of NaNs in each column

bonus                        43.835616
deferral_payments            73.287671
deferred_income              66.438356
director_fees                88.356164
exercised_stock_options      30.136986
expenses                     34.931507
from_messages                41.095890
from_poi_to_this_person      41.095890
from_this_person_to_poi      41.095890
loan_advances                97.260274
long_term_incentive          54.794521
other                        36.301370
poi                           0.000000
restricted_stock             24.657534
restricted_stock_deferred    87.671233
salary                       34.931507
shared_receipt_with_poi      41.095890
to_messages                  41.095890
total_payments               14.383562
total_stock_value            13.698630
dtype: float64
```

One more thing to taken into consideration before deleting any column is that we need to check is that if the count of NaNs is not specific for POI or non-POI . If that is the case the the machine learning algorithm will use NaN to make prediction for POI or non-POI . So I have included the further analysis based on the label associated with each row .
This table shows % of NaNs in each column grouped by label :

| | Not POI % | POI % |
|---|---|---|
| **bonus** | 48.43750 | 11.111111 |
| **deferral_payments** | 73.43750 | 72.222222 |
| **deferred_income** | 70.31250 | 38.888889 |
| **director_fees** | 86.71875 | 100.000000 |
| **exercised_stock_options** | 29.68750 | 33.333333 |
| **expenses** | 39.84375 | 0.000000 |
| **from_messages** | 43.75000 | 22.222222 |
| **from_poi_to_this_person** | 43.75000 | 22.222222 |
| **from_this_person_to_poi** | 43.75000 | 22.222222 |
| **loan_advances** | 97.65625 | 94.444444 |
| **long_term_incentive** | 57.81250 | 33.333333 |
| **other** | 41.40625 | 0.000000 |
| **restricted_stock** | 27.34375 | 5.555556 |
| **restricted_stock_deferred** | 85.93750 | 100.000000 |
| **salary** | 39.06250 | 5.555556 |
| **shared_receipt_with_poi** | 43.75000 | 22.222222 |
| **to_messages** | 43.75000 | 22.222222 |
| **total_payments** | 16.40625 | 0.000000 |
| **total_stock_value** | 15.62500 | 0.000000 |

After observing the above table , I decided to remove columns which had more than 70% NaNs in them .

So I removed the following columns :

- Deferred Payments
- Deferred Income
- Director Fees
- Loan advances
- Restricted Stock Deferred

Next , I removed up those rows which had more than 70% NaN values .

**Note :** To be sure I did not delete any row with POI labels , I double checked among the rows that were to be deleted whether they were POI or not . After inspection I found out that they were all non-POI rows. So I deleted them .

After all this cleaning of NaN values, The % reduction in NaN values is as follows :

```
Reduction in percentage of NaNs in each column after cleaning

bonus                      40.625000
exercised_stock_options    34.090909
expenses                   37.254902
from_messages              43.333333
from_poi_to_this_person    43.333333
from_this_person_to_poi    43.333333
long_term_incentive        32.500000
other                      45.283019
poi                              NaN
restricted_stock           52.777778
salary                     50.980392
shared_receipt_with_poi    43.333333
to_messages                43.333333
total_payments             47.619048
total_stock_value          65.000000
dtype: float64
```

**Feature Engineering**

The following features were created from the existing dataset :

- **Fraction_from_poi** : fraction of from_messages received that were from POI
- **Fraction_to_poi :** fraction of to_messages sent to POI
- **Related to poi :** fraction of all messages (sent+received) that were related to POI
- **Effective salary :** This is basically calculated as salary+bonus+long term incentive - expenses .

**Feature Selection :**
I used sklearn's selectKBest function to select the features with highest ranking . The ranking is as follows :

```
Ranking of features is as follows
1 . Score for  exercised_stock_options  is  23.6138062092
2 . Score for  total_stock_value  is  16.7366284243
3 . Score for  Effective Salary  is  11.5559727014
4 . Score for  bonus  is  9.73938229385
5 . Score for  salary  is  7.99996454308
6 . Score for  related_to_poi  is  7.72312955873
7 . Score for  total_payments  is  5.75730233923
8 . Score for  restricted_stock  is  5.34226430067
9 . Score for  long_term_incentive  is  5.07283618479
10 . Score for  shared_receipt_with_poi  is  4.87403011599
11 . Score for  fraction_from_poi  is  3.11963598574
12 . Score for  from_poi_to_this_person  is  2.58082108539
13 . Score for  fraction_to_poi  is  1.99020062202
14 . Score for  other  is  1.59257036438
15 . Score for  from_this_person_to_poi  is  1.15629869907
16 . Score for  from_messages  is  0.497759385305
17 . Score for  expenses  is  0.496738048357
18 . Score for  to_messages  is  0.312817105062
```

I decided to make 3 sets of features , each containing Top 5 , Top 7 and Top 10 features from the above ranking .

After performing analysis with the above 3 sets using different machine learning algorithms, I decide to use the following 5 features which gave the most optimum results :

- **Exercised Stock Options**
- **Total Stock values**
- **Effective Salary**
- **Related to POI**
- **Total Payments**

**Note :** The above mentioned Analysis is shown in detail in the notebook .

**Feature Scaling**
I used sklearn's Standard Scalar function to scale the feature . This was added as a step in the machine learning pipeline . The main idea behind feature scaling is to ensure that all features are to be treated equally regardless of the values that they contain, i.e. one feature should not affect the other feature just because the former one had larger values , etc. For more on Standard Scalar function [check this link](#) .

**3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?**

I tried various machine learning algortihms and compared their accuracies, recall and precision :

```
Using Top 7 fetaures
```

| Classifier | Accuracy | Precision | Recall |
|---|---|---|---|
| KNN | 0.166666666667 | 0.0666666666667 | 0.861111111111 |
| Decision Tree | 0.419312169312 | 0.466666666667 | 0.805555555556 |
| SVM_rbf | 0.235338345865 | 0.666666666667 | 0.648148148148 |
| Gaussian Naive Bayes | 0.361111111111 | 0.266666666667 | 0.833333333333 |
| SVM_linear | 0.323232323232 | 0.6 | 0.768518518519 |
| Logistic Regrssion | 0.308913308913 | 0.666666666667 | 0.740740740741 |
| SVM_sigmoid | 0.224937343358 | 0.733333333333 | 0.611111111111 |
| Random Forest | 0.0833333333333 | 0.0666666666667 | 0.833333333333 |

Now out of the above algorithms I decided to select SVM with sigmoid kernel , Logistic Regression and Decision trees for parameter tuning .

**Note :** Random Forest was not chosen because grid search over Random Forest was taking a lot of time and I did not have the necessary computing resources to do it quickly .

**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 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)**

Most of the machine learning algorithms have different parameter that control it's performance and are needed to be tuned according the data . This requires running the algorithms multiple times using different sets of parameter and finally selecting the one with the best performance . This also ensures that our algorithm do not overfit on the training data .

I have used sklearn's GridSearchCV to perfrom parameter tuning of the selected machine learning algorithm . Check out [this link] for more info .

I also used startified train/test split to perfrom cross validation over the dataset . Check out [this link] for more info .

The features tuned for Logistic Regression were tol , C and solver .
The features tuned for SVM with sigmoid kernel were tol , C and gamma .
The features tuned for Logistic Regression were min_samples_split, max depth and max_leaf_nodes .

Also note that class-weight was set to 'balanced' to account for class imbalance and random state was set to 0 for all the classifiers .

**Note :** The performance metric chosen to compare algorithms was F1 score with average parameter set to 'weighted' to account for class imbalance **.** This choice will be explained in the next section . You can check about sklearn's implementation of F1 score on this link .

The performance of the three algorithms after parameter tuning is as follows :

| Classifier | F1 score |
|---|---|
| SVM with sigmoid kernel | 0.658785332315 |
| Logistic Regression | 0.769213139801 |
| Decision Tree | 0.534759358289 |

The final algorithm chosen was Logistic Regression with the following configuration :
LogisticRegression(C=0.001, class_weight='balanced', dual=False,
fit_intercept=True, intercept_scaling=1, max_iter=500,
multi_class='ovr', n_jobs=1, penalty='l2', random_state=0,
solver='sag', tol=0.01, verbose=0, warm_start=False))])

**5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?**
Validation is basically a measure to check how well our model performs before using it on real world data . It can also be used to check if our model does not overfit over the training dataset .

Generally , we use accuracy as a measure to check how well our algorithms performs . Since our data suffers from class imbalance problem , i.e. out of the 146 people only 18 are POI . So even if we marked all points as non-POI , we would get an accuracy of 128/146 , i.e. approx 87% . To rectify this error , we use precision , recall , F1 score , F2 score and area under ROC curve which are explained below :

- **Precision :** If our algorithm identifies a person as POI, the probability that the person is actually a POI
- **Recall :** Probability of identifying a POI , given that the person is actually a POI.
- **F1 score :** 2.0 * true_positives/(2*true_positives + false_positives+false_negatives)
- **F2 score :** (1+2.0*2.0) * precision*recall/(4*precision + recall)

The file tester.py is used to analyse the performance of the final model . It uses 1000 randomized stratified splits of our dataset to evaluate our model and returns the accuracy,precision, recall, F1 score and F2 score averaged over the 1000 splits .

**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.**

I have used precision and recall as the final measure to evaluate the performance of the final model .
The final Logistic Regression model reported the following results :

- **Precision of 37 .668% :** If our algorithm predicts that a person is POI, then there is 34.168% chance that he/she is actually a POI .
- **Recall of 38.25% :** There is 49.15% chance of identifying a person as POI, given that that person is actually a POI .

Our final model reported the final results over the different evaluation metrics :

| **Metric** | **Value** |
|---|---|
| Accuracy | 79.158% |
| Precision | 37.668% |
| Recall | 38.25% |
| F1 score | 37.956% |
| F2 score | 38.132% |