Identify Fraud from Enron Email

by Karina Lund

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"]*

Background on the dataset

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

Machine learning can be useful within dataset to find out if there are any patterns within the emails of people who were persons of interest in the fraud case.

The goal of this project is to build a predictive model that can identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset, using machine learning.

The dataset used in project contains financial data, email metadata for 146 Enron employees. Just 18 of them are marked as persons of interest [1](POI). This imbalance between total number of a class of data and total number of another class data is a common problem in machine learning called "class imbalance problem". Therefore, to compare algorithms performance I should use alternative metrics (True Positive, True Negative, False Positive, False Negative) instead of using just accuracy.

Dataset is stored in a dictionary, where each key in the dictionary is a person's name and the value is a dictionary containing all the features of that person.

Dataset has 21 features divided into 3 types:

financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

POI label: ['poi']

Missing values in dataset by feature:

feature	count	POI	Non-POI
total_stock_value	20	0.0 %	100.0 %
total_payments	21	0.0 %	100.0 %
restricted_stock	36	2.78 %	97.22 %
exercised_stock_options	44	13.64 %	86.36 %
salary	51	1.96 %	98.04 %
expenses	51	0.0 %	100.0 %

[1] individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity

other	53	0.0 %	100.0 %
to_messages	60	6.67 %	93.33 %
from_poi_to_this_person	60	6.67 %	93.33 %
from_messages	60	6.67 %	93.33 %
from_this_person_to_poi	60	6.67 %	93.33 %
shared_receipt_with_poi	60	6.67 %	93.33 %
bonus	64	3.13 %	96.87 %
long_term_incentive	80	7.5 %	92.5 %
deferred_income	97	7.22 %	92.78 %
deferral_payments	107	12.15 %	87.85 %
restricted_stock_deferred	128	14.06 %	85.94 %
director_fees	129	13.95 %	86.05 %
loan_advances	142	11.97 %	88.03 %

Last 2 columns show what ratio of missing values belongs to POI and non-POI persons. Missing values will be replaced with 0.

Outliers

Visualization is one of the most powerful tools for finding outliers.

After plotting salary and bonus on chart there were one outlier that couldn't be ignored. This outlier had a dictionary key «Total» which is just a total row. This outlier should by cleaned away.

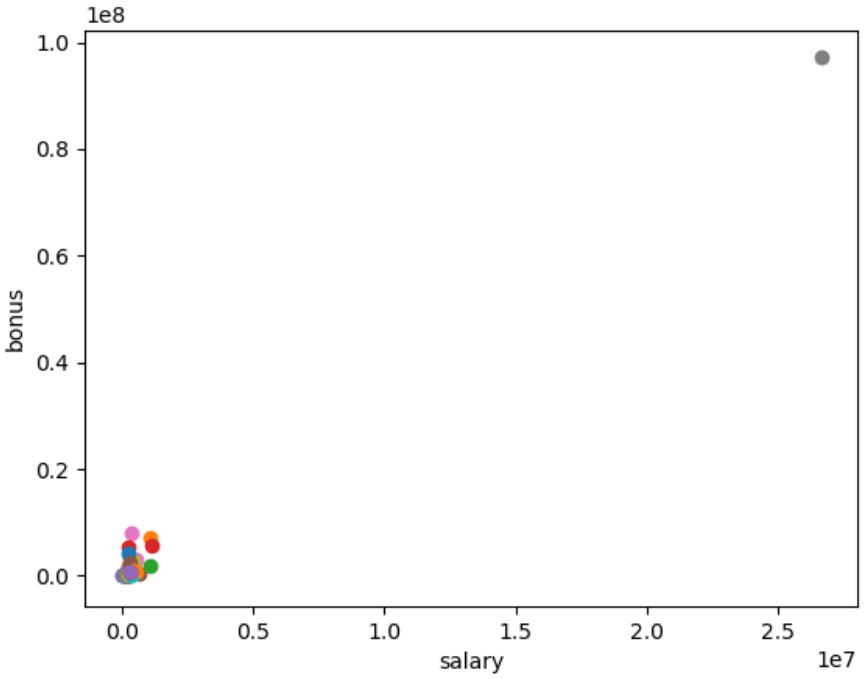


Figure 1 Scatterplot salary vs bonus

After that I made a scatterplot for "total_payments" and "total_stock_value". It was a data point which looked like an outlier, but it belonged to Kenneth Lay. It is a valid data point and it shouldn't be removed.

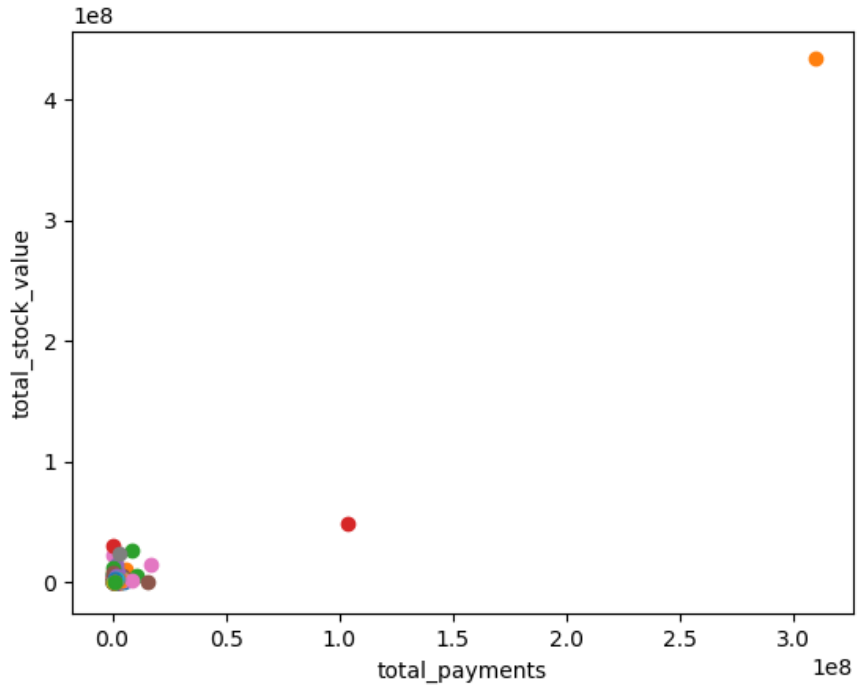


Figure 2 Scatterplot total payments vs total stock value

There are 2 other outliers that were found in dataset and cleaned away:

- "The travel agency in the park". It's not really a person, it's unnecessary to have it further in project.
- Lockhart Eugene. Data for him contains only NaNs.

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. (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"]*

Features needs to really capture the trends and patterns in data. Machine learning algorithm is only going to be as good as the features that are put into it. I should select the best features that are available to me.

I've created two new features: "fraction_to_poi" and "fraction_from_poi" that presents the fraction of messages to this person that come from persons of interest and from this person to persons of interest.

I used SelectKBest algorithm to find out what K features are most influential. The function returned these scores for all available and two new the features:

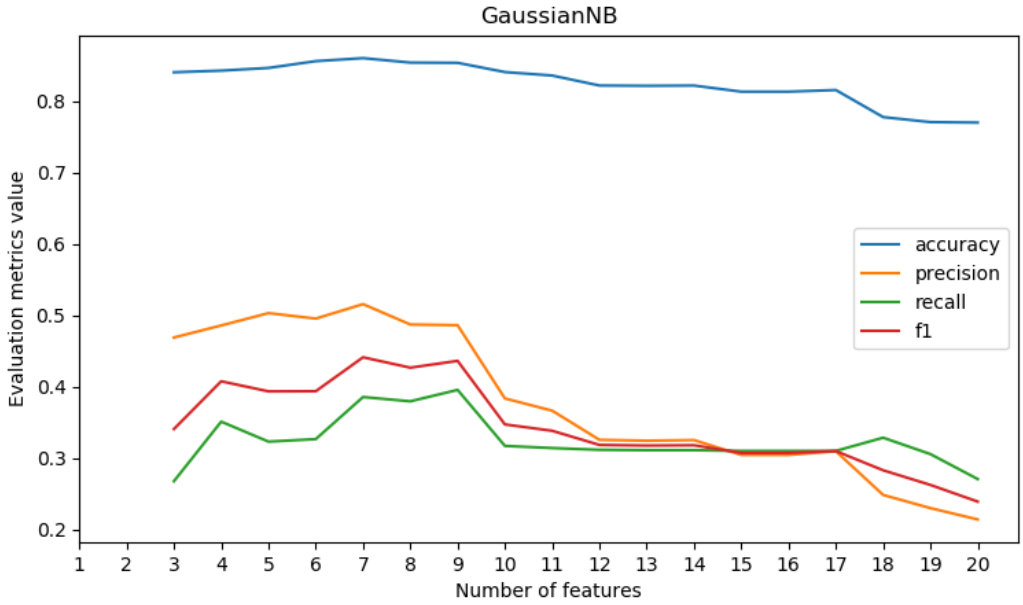
nr	feature	score
1	exercised_stock_options	25.097541528735491
2	total_stock_value	24.467654047526398
3	bonus	21.060001707536571
4	salary	18.575703268041785
5	fraction_to_poi	16.641707070468989
6	deferred_income	11.595547659730601
7	long_term_incentive	10.072454529369441
8	restricted_stock	9.346700791514877
9	total_payments	8.8667215371077717
10	shared_receipt_with_poi	8.7464855321290802
11	loan_advances	7.2427303965360181
12	expenses	6.2342011405067401
13	from_poi_to_this_person	5.3449415231473374
14	other	4.204970858301416
15	fraction_from_poi	3.2107619169667441
16	from_this_person_to_poi	2.4265081272428781
17	director_fees	2.1076559432760908
18	to_messages	1.6988243485808501
19	deferral_payments	0.2170589303395084
20	from_messages	0.16416449823428736
21	restricted_stock_deferred	0.06498431172371151

One of newly added features “fraction_to_poi” showed a good score!

When deciding what number of features to use, I should keep in mind that with low number of features, model can be underfitted and with high number – overfitted.

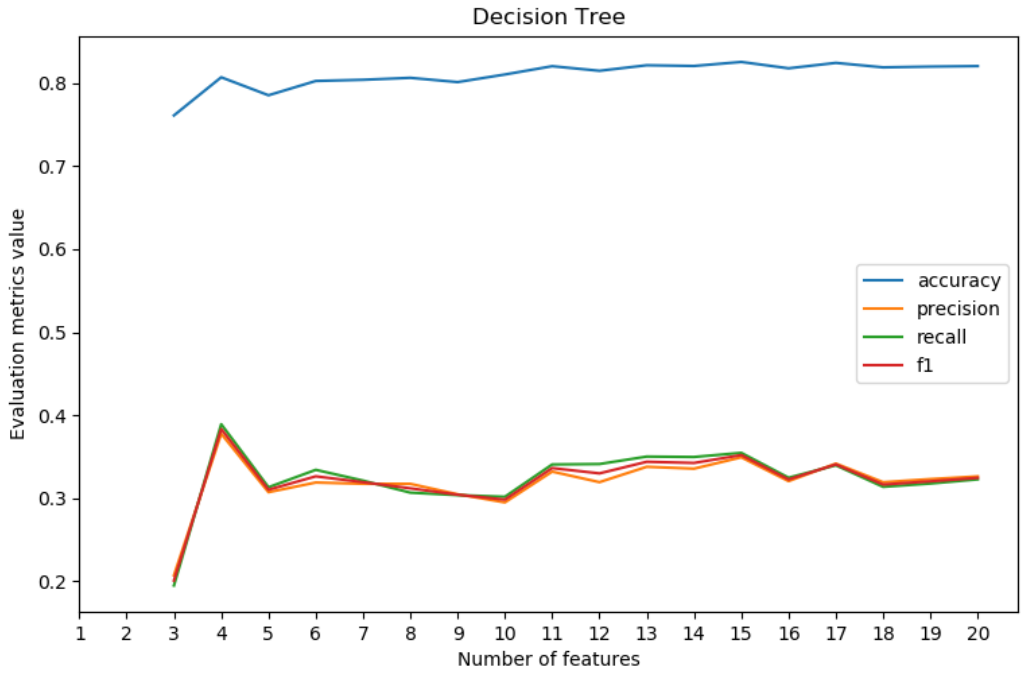
I plotted what values for evaluation metrics I’ve got for every algorithm I choosed and here are my results:

- Gaussian NB



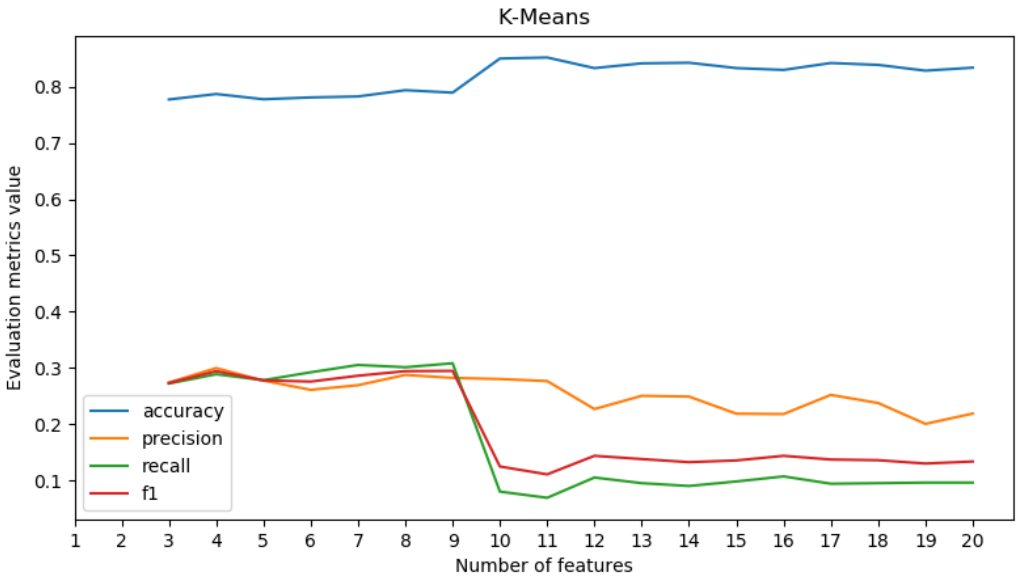
Best combination of accuracy, precision, recall and f1 is for 7 best features. (include 'poi' feature)

- Decision Tree



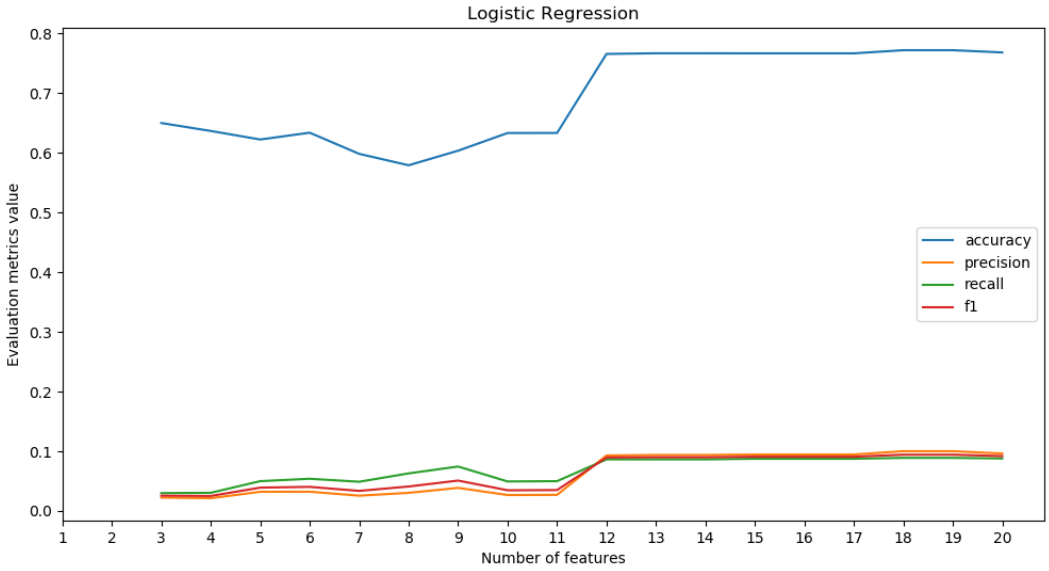
4 features will give highest precision and recall

- K-means with 2 clusters



It looks like the best combination can be achieved with 4 best parameters, but precision and recall lying on edge of .3

- Logistic Regression



Plot shows that the best evaluation metrics values can be achieved by using more than 11 top features. I will use 18, because it seems it is a peak.

Further I will use mentioned number of features for algorithms.

Feature Scaling

Before I started to train Machine Learning algorithm classifiers, I scaled all feaures with sklearn's MinMaxScaler(). MinMaxScaler scales all selected features to a given range (between 0 and 1). It was important since features have different units and range. For example, salary's range is between thousands and million dollars, but email features are up to thousands.

3. *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 4 different algorithms and ended up using Gaussian Naive Bayes because it scored the highest evaluation metrics. Other algorithms that I tried were:

- Decision Tree
- K-means
- Logistic Regression

It took longer time (*12 times than others) to run K-means algorithm.

Logistic Regression have been choosed over Linear Regression since output (POI or non-POI) are discrete values.

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? 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 adjusting the parameters in a certain way to achieve optimal algorithm performance. Algorithm performance can be measured in different ways, like accuracy, precision, recall, f1 score. Adjusting the parameters can be done manually (trying different parameters) or automatically using functions (like GridSearchCV). If algorithm is not tuned well the consequence could be a poor performance: algorithm won't be learned well and it will not succeed in predictions of new data.

All algorithms have its own set of parameters and default values for them. It's not always necessary to tune, but if task is to achieve a best performance, in most cases tuning of algorithm parameters will help. So, using different set of parameters gives different result, different score.

Sklearn has GridSearchCV function that working though combinations of parameters tunes and determine which tune gives best performance. I get optimal parameter combination by using "best_params_" to GridSearchCV.

algorithm	**Parameters for tune**	**best parameters by GridSearchCV**
Gaussian NB	-	-
Decision Tree	{'splitter':('best','random'), 'criterion':('gini', 'entropy')}	{'splitter': 'random', 'criterion': 'entropy'}
K-means	-	-
Logistic Regression	{'C': [0.1, 0.01, 0.001, 0.0001], 'tol': [1, 0.1, 0.01, 0.001, 0.0001]}	{'C': 0.1, 'tol': 1}

Gaussian Naive Bayes don't have parameters for tune, and for K-means algorithm I used just 2 clusters, because I have 2 categories that should be classified.

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 items: "discuss validation", "validation strategy"]*

Validation is the process where a trained model is evaluated with a testing dataset. The classic mistake in validation process is build the model that can predict a training data very well, but having poorly performance on new/unseen data. It calls overfitting. Goal of validation is to avoid this situation. It can be accomplished though cross-validation process.

Cross-validation is a process that randomly splits data into training and testing set. So, model will be train on training dataset and will be validated on testing data set.

Since "class imbalance problem" have a place here, I used the StratifiedShuffleSplit function to split the data into training and tests.

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"]*

To validate algorithms I used precision, recall and accuracy scores.

- Accuracy is the most common measure of an algorithm's performance. It shows the number of predicted labels it got correct over all predicted labels. However, it's not right to rely only on accuracy as evaluation metric in this project. Since there is a large imbalance in the data labels (more Non-POI than POI labels), high accuracy can be achieved by predicting all labels to be Non-POI. That's why it's better to use precision and recall
- Precision shows the fraction of persons of interest predicted by the algorithm that are truly persons of interest. Other words is number of correct positive classifications divided by the total number of positive labels assigned
- Recall shows the fraction of the total number of persons of interest in the data that the classifier identifies. Other words it is the number of correct positive classifications divided by the number of positive instances that should have been identified

Table below shows validation scores for algorithms for 7 best features:

Algorithm	**Precision**	**Recall**	**Accuracy**
Gaussian NB	0.51572	0.38550	0.86050
Decision Tree	0.37537	0.37650	0.80769
K-means	0.27256	0.29600	0.77015
Logistic Regression	0.10023	0.08900	0.77200

For this project, I should tune classifier to achieve better than .3 precision and recall using our testing script (tester.py).

Final results:

Algorithm: Gaussian NB
Accuracy: 0.86050
Precision: 0.51572
Recall: 0.38550
F1: 0.44120

F2: 0.40600
Total predictions: 14000
True positives: 771
False positives: 724
False negatives: 1229
True negatives: 11276

7. Reflection

In this project, I needed to build a predictive model that can identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset, using machine learning. To accomplish that I went through multiply stages.

First, I explored the dataset, looking for missing values and used visualization as a tool to find outliers. I found out that dataset is small and imbalanced, so I should use special methods to deal with it, like using a shuffle split. Using SelectKBest function from sklearn to look into most influential features from dataset. 'fraction_to_poi' was one of new features I added to dataset and it was great that this feature made sense and were very influent. Since features in dataset had different units and range I rescaled them with MinMaxScaler. Then I tried 4 different algorithms (Gaussian Naive Bayes, Decision Tree, K-means and Logistic Regression), tuned their parameters with GridSearchCV and evaluated with 3 metrics: precision, recall and accuracy.

Algorithm I ended up with is Gaussian Naive Bayes. It showed the highest precision and recall and both were better than .3.