

[Add topics](#)

📄 17 commits

🌿 1 branch

📦 0 releases

👤 1 contributor

Branch: master ▾


New pull request

Create new file

Upload files

Find file

Clone or download ▾

 tonybastin	Updated for Rev. comments 1	Latest commit 767e0d0 18 hours ago
📁 final_project	Updated for Rev. comments 1	18 hours ago
📁 tools	Added classifier and validation function	2 days ago
📄 .gitignore	Initial commit	6 days ago
📄 README.md	Updated for Rev. comments 1	18 hours ago

📖 README.md

Identifying fraud from Enron Email

by Tony Bastin

Questions

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”]

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.

The goal of this project is to make machine learning model that can identify a person of interest (POI) using Enron financial and email data. The given Enron dataset contained details about 146 persons, out of which 18 were persons of interest (POI). For each given person three major types of features were available as given below :

- 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'] (boolean, represented as integer)

The email feature 'email_address' was removed from further analysis as it won't convey any meaningful information to the machine learning model.

The outliers found in the dataset were the following :

1. 'TOTAL'
2. 'THE TRAVEL AGENCY IN THE PARK'
3. 'LOCKHART EUGENE E'

I used the following techniques to identify the outliers:

1. Data exploration (detected - 'TOTAL', see plots)

2. Reading through the names of the person after printing them out(detected - 'TOTAL' & 'THE TRAVEL AGENCY IN THE PARK')
3. Wrote a function to detect people with less than 10% features(detected - 'LOCKHART EUGENE E')

(Details of the functions made can be seen in [data_exploration.py](#) and the plots here (1,2))

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"]

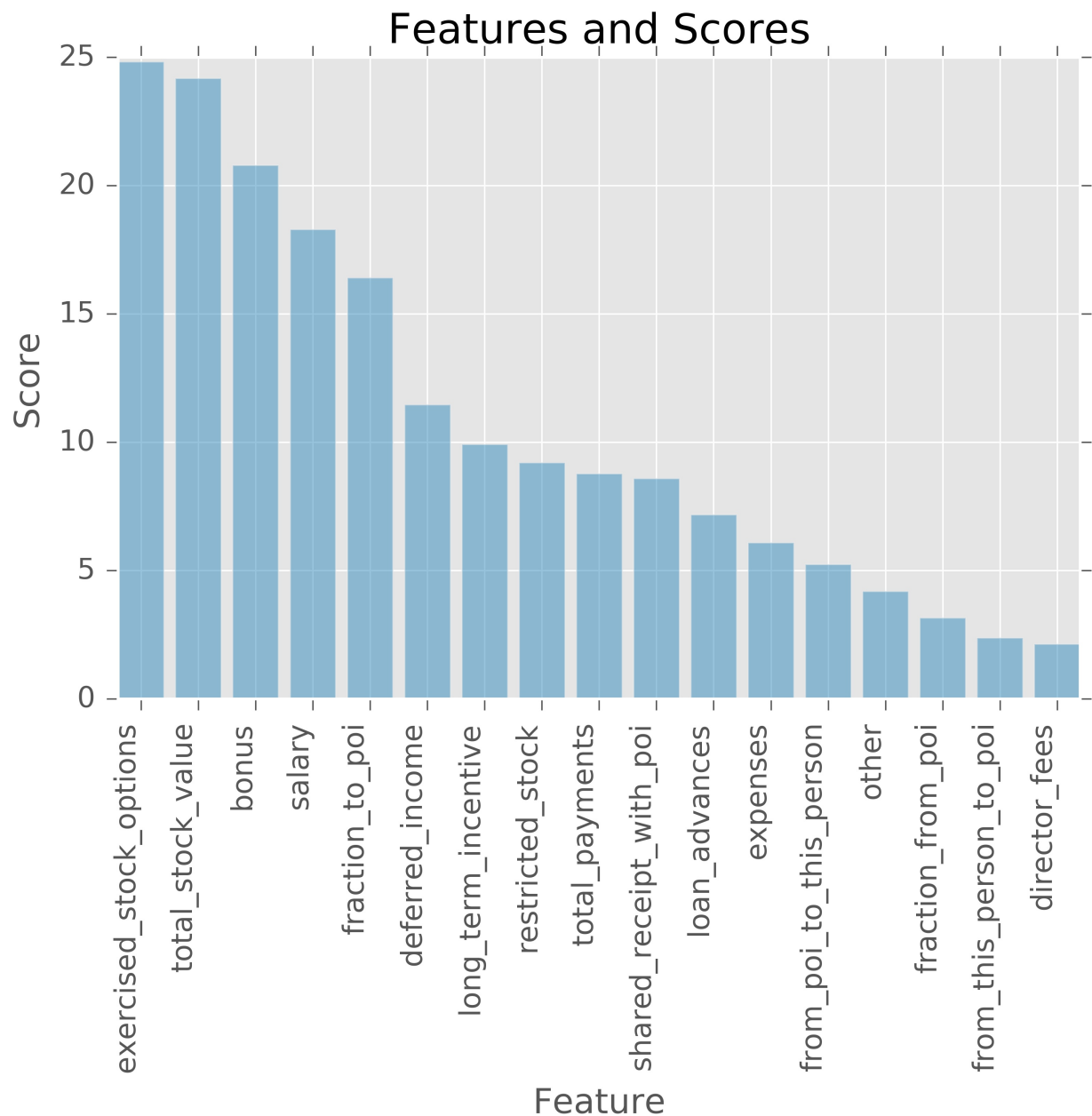
Two new features were introduced into the dataset : 1) "fraction_to_poi": ratio of total emails to a POI to total emails 2) "fraction_from_poi": ratio of total emails from a POI to total emails. The rationale for addition of these features is that persons who interacted with POI's at higher rate may be colluding with those POIs to commit fraud and therefore be POIs themselves.

After that I used the SelectKBest to select 15 best features based on their scores as given below :

No.	Feature	Score
1)	exercised_stock_options	24.82
2)	total_stock_value	24.18
3)	bonus	20.79
4)	salary	18.29
5)	fraction_to_poi	16.41
6)	deferred_income	11.46
7)	long_term_incentive	9.92
8)	restricted_stock	9.21
9)	total_payments	8.77
10)	shared_receipt_with_poi	8.59
11)	loan_advances	7.18
12)	expenses	6.09
13)	from_poi_to_this_person	5.24
14)	other	4.19
15)	fraction_from_poi	3.15
16)	from_this_person_to_poi	2.38
17)	director_fees	2.13

After looking in the above table, it is seen that the newly introduced feature "fraction_to_poi" has obtained the fifth position in the table with a score of 16.41 and "fraction_from_poi" has 15th position with a score of 3.15. Thus it shows that "fraction_to_poi" will have a significant effect on the prediction model when compared to the other features which was already there in the dataset.

A bar chart of the above table is given below:



After looking through the above bar chart, there was a big dip in the feature score after the fifth feature showing the importance of these 5 features and hence the first 5 features with best scores were chosen for building the machine learning model.

After selecting the 5 best features, the features were scaled using the `MinMaxScaler()` before selecting and tuning the machine learning algorithms.

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 the following algorithms - Naive Bayes classifier, Random Forest classifier, Decision Tree classifier, Support Vector classifier and Logistic Regression. However ended up tuning further Decision Tree classifier and Logistic Regression as they were yielding better results ("Precision" & "Recall"). The final model I choose between the two was Logistic Regression.

Model	Precision	Recall
Logistic Regression	0.352	0.669
Decision Tree	0.335	0.343

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"]

Parameter Tuning refers to varying the parameters of a machine learning algorithm either manually or automatically to find out the set of parameters which yield the best performance for an algorithm. The parameters of an algorithm used on a particular dataset may not yield good performance on another similar dataset and we will have to do the tuning again if the dataset has changed even slightly.

In this case, I used GridSearchCV for tuning the parameters for the chosen algorithms.

The parameters tuned are as listed below:

1. Logistic Regression ('C', 'tol')
2. Decision Tree ('criterion', 'min_samples_split')

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"]

In machine learning, model validation is referred to as the process where a trained model is evaluated with a testing data set. The testing data set is a separate portion of the same data set from which the training set is derived. The common evaluation parameters are Precision and Recall.

The classic mistake is overfitting where you train the algorithm on all available data instead of splitting it into training and testing data. Overfitting causing the model to merely memorize classification and not 'learn' to generalize and apply this information to new datasets.

The final model was validated using the `estimator_evaluator()`. It uses a loop to iterate `num_iter` times (used `num_iter = 1000`) splitting the dataset into training and test set in the ratio 0.7:0.3 using `train_test_split`, and making predictions the accuracy, precision & recall using `sklearn.metrics`. The average values are reported here.

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"]

The evaluation matrix for the 2 best algorithms are as listed below :

Model	Precision	Recall
Logistic Regression	0.352	0.669
Decision Tree	0.335	0.343

For the logistic regression prediction is 32% that is out of all items that are predicted by the model as correct, 35% are correct. A high precision means POIs identified by an algorithm tended to be correct. Recall of 67% means that out of all items predicted correctly, 67% are truly correct. In this case, a high recall means if there are POIs in the dataset, an algorithm has good chance to identify them.

References:

<https://matplotlib.org/devdocs/index.html> <https://www.tutorialspoint.com/python/>
<https://stackoverflow.com/questions/41885743/skip-to-next-item-in-dictionary> <http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html> <https://classroom.udacity.com/>