Michael C. Villadelrey

Project 5

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”]

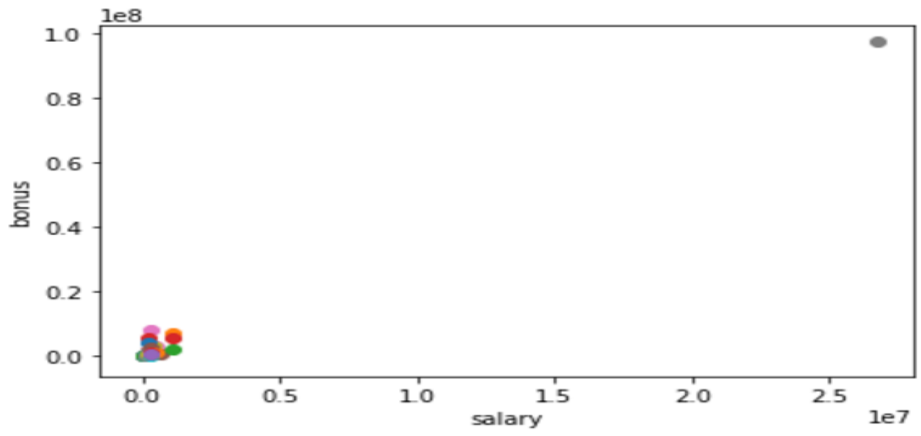
ANS:

The goal of the project is to investigate the ENRON fraud case and identify other persons of interest (POI) that we can put into jail or subject to further investigation, using our current list of persons of interest (POI). Machine learning is useful in this case because of its powerful and systematic algorithm of classifying the POIs from non-POIs. The data available for this investigation is the financial details of the employees of ENRON (salary, deferral payments, total payments, loan advances, bonus, restricted stock deferred, deferred income, total stock value, expenses, exercised stock options, long term incentive, restricted stock, director fees - all units are in US dollars). We also have email echanges between employes.

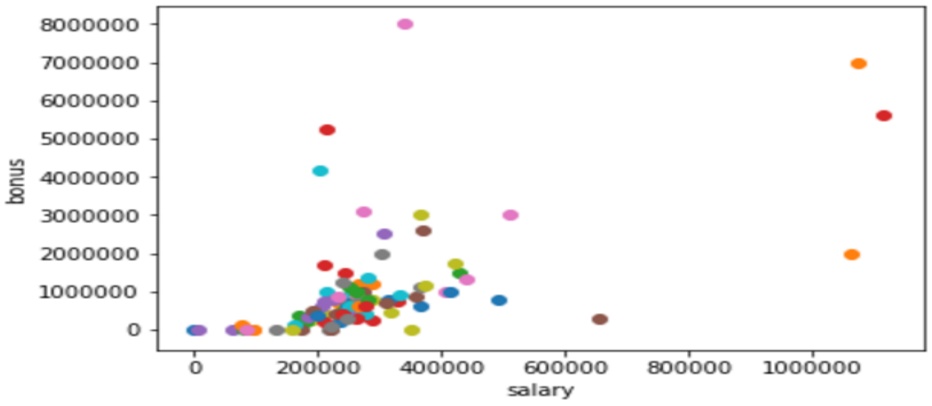
We have 146 observations, with 21 variables in the dataset. There are 18 POIs and 128 non POIs in the dataset. There are also features with a lot of missing values.

```
Index: 146 entries, ALLEN PHILLIP K to YEAP SOON
Data columns (total 21 columns):
salary                         95  non-null   float64
to_messages                    86  non-null   float64
deferral_payments              39  non-null   float64
total_payments                125  non-null   float64
exercised_stock_options       102  non-null   float64
bonus                          82  non-null   float64
restricted_stock              110  non-null   float64
shared_receipt_with_poi        86  non-null   float64
restricted_stock_deferred      18  non-null   float64
total_stock_value             126  non-null   float64
expenses                       95  non-null   float64
loan_advances                   4  non-null   float64
from_messages                  86  non-null   float64
other                          93  non-null   float64
from_this_person_to_poi        86  non-null   float64
poi                           146  non-null   float64
director_fees                  17  non-null   float64
deferred_income                49  non-null   float64
long_term_incentive            66  non-null   float64
email_address                 146  non-null   object
from_poi_to_this_person        86  non-null   float64
```

By plotting our initial features (‘salary’, ‘bonus’), I was able to identify a clear outlier in the dataset:



With simpy checking the enron61702insiderpay.pdf file, I found out that this value refers to the row 'TOTAL'. This is not a valid data point so I removed it immediately from dataset. After removing the entry of ‘TOTAL’, i got this updated plot:



I initially thought that I might have here 4 more potential outliers. Based from the plot, there are two persons who made bonuses of at least 5M dollars, and salary of over 1M dollars. So I deeply checked the dataset and found out that these numbers are from the big bosses of ENRON, who are definitely POIs.

	salary	**to_messages**	**deferral_payments**	**total_payments**	**exercised_stock_options**	**bonus**
SKILLING JEFFREY K	1111258.0	3627.0	NaN	8682716.0	19250000.0	5600000.0
LAY KENNETH L	1072321.0	4273.0	202911.0	103559793.0	34348384.0	7000000.0

These are valid data points so I decided to leave them in as part of the dataset.

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"]

ANS:
I ended with the following features for the model:
features_list = ["poi", "shared_receipt_with_poi", "fraction_from_poi", "fraction_to_poi"]

I have not done any scaling since the features i used are more meaningful in its original scale.
I have created two new features, namely "fraction_from_poi" and "fraction_to_poi". These new features was obtained from the number of emails from POI to the person, and number of emails from this person to POIs. Obviously, if there are persons in the company that sends or receives emails in general, then there's a high chance that these employees will have high quantity of emails received or sent from POIs. So it better to create new features that will give the fraction of messages to/from that person that are from/to a POI.

I also used a decision tree to get the importances of the features:
Feature Ranking:

1	feature	salary	0.211707
2	feature	bonus	0.14623
3	feature	fraction_from_poi	0.120902
4	feature	fraction_to_poi	0.118337
5	feature	deferral_payments	0.08798
6	feature	total_payments	0.074783
7	feature	loan_advances	0.074533
8	feature	restricted_stock_deferred	0.074401
9	feature	deferred_income	0.053416
10	feature	total_stock_value	0.037712
11	feature	expenses	0
12	feature	exercised_stock_options	0
13	feature	long_term_incentive	0
14	feature	shared_receipt_with_poi	0
15	feature	restricted_stock	0
16	feature	director_fees	0

I tried using the top 10 features based on importances but i ended with not so good evaluation metric values. So I tested some combinations of the features and got a fairly decent evaluation metric values using features_list = ["poi", "shared_receipt_with_poi", "fraction_from_poi", "fraction_to_poi"].

After tuning up the Decision tree model with the use of GridsearchCV, i tried checking the performance of the model **without the “new features**”:

features_list = ["poi", "salary", "bonus", 'deferral_payments', 'total_payments', 'loan_advances', 'restricted_stock_deferred',
'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options',
'long_term_incentive', 'shared_receipt_with_poi', 'restricted_stock', 'director_fees']

Output:

```
training time: 0.138 s
prediction time: 0.001 s
Accuracy:  0.733
precision =  0.0
recall =  0.0

Removing those with 0 importances:
```

features_list = ["poi", "salary", "bonus", 'deferral_payments', 'total_payments', 'loan_advances', 'restricted_stock_deferred',
'deferred_income']

Output:

```
training time: 0.137 s
prediction time: 0.001 s
Accuracy:  0.923
precision =  0.0
recall =  0.0
```

The model gave **good accuracy value but very poor precision and recall.**

By simply **adding the new features**, the model gave a pretty **decent precison and recall** values:

features_list = ["poi", "salary", "bonus", "fraction_from_poi", "fraction_to_poi", 'deferral_payments', 'total_payments', 'loan_advances', 'restricted_stock_deferred',
'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options',
'long_term_incentive', 'shared_receipt_with_poi', 'restricted_stock', 'director_fees']

Output:

```
training time: 0.143 s
prediction time: 0.0 s
Accuracy:  0.8
precision =  0.667
recall =  0.5
```

And this one gave the best result:

features_list = ["poi", "shared_receipt_with_poi", "fraction_from_poi", "fraction_to_poi"]

Output:

```
Accuracy:  0.893
precision =  0.5
recall =  0.667
```

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”]

ANS:

My ended up using a decision tree classifier. I have also tried Gaussian Naive Bayes and SVC but these models resulted to a good accuracy but very low precision and recall.

```
clf = GaussianNB()
Result:
training time: 0.001 s
prediction time: 0.001 s
Accuracy:  0.879
precision =  0.0
recall =  0.0
```

```
clf = SVC()
training time: 0.002 s
prediction time: 0.001 s
Accuracy:  0.909
precision =  0.0
recall =  0.0
```

```
clf = tree.DecisionTreeClassifier()
training time: 0.004 s
prediction time: 0.0 s
Accuracy:  0.818
precision =  0.2
recall =  0.2
```

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”]

ANS:

Tuning the parameters means finding the value of the parameters of the model that will give you the best evaluation metrics. If we don’t do this well, we might end up with a suboptimal model for our dataset.

In my case, I used GridsearchCV to tune up the criterion and min_samples_split of the decision tree model.

```
param_grid = {
        'criterion': ['gini', 'entropy'],
        'min_samples_split': [2, 5, 10, 15, 20, 25],
        }
clf = GridSearchCV(DecisionTreeClassifier(), param_grid)
```

Min_samples_split has been proven to heavily affects the performance of a decision tree model. I also wanted to test the model’s criterion, whether the “Gini” impurity or the “entropy” information gain best fit the dataset.

Checking the best estimator:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy',
            max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=10, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

And with this, i obtained a more decent evalution metric values:

Accuracy: 0.893
precision = 0.5
recall = 0.667

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”]

ANS:

Validation is basically measuring the performance of your machine learning model. Depending on your goal/objective, you might want to set a different evaluation metrics that will suit best on your problem.

If we are not aware of the weaknesses or drawbacks of a metric, we might ended up using a model that is not good on a specific problem. A classic mistake of this is using accuracy metric on a classification problem that has huge difference in the quantity of the classes/groups. This is the case of ENRON dataset where we have a very few POIs compared to the total number of employees in our dataset. Using accuracy alone will not make sense in this dataset. That is why we also included recall and precision as part of our evaluation metrics.

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”]

ANS:

In this project, i also used recall and precision as part of the evaluation metrics. My final model obtained

Accuracy: 0.893
precision = 0.5
recall = 0.667

Accuracy just tells you the % of your correct prediction out of the total number of predictions. Like I said high accuracy alone in this dataset is not enough since the number of POIs is significantly small compared to the number of employees available in the dataset. So in my result, 89% of the time, my prediction is correct.

Precision tells us how likely someone is guilty given that we flagged him as a POI. In my result, given that a person is flagged as a POI, there is a 50% chance that he really is a POI.

Recall on the other hand is the chance of our model, flagging the guilty people as POI. In my result, we have 67% chance flagging those involved in the fraud as POI.