Enron Machine Learning Project 2
Bryony Miles

Question: Which Enron Employees may have committed fraud based on the public Enron dataset.

1. Data Exploration

First I listed the 21 available features:

Email related	Financial	POI
1. email_address 2. to_messages 3. from_messages 4. shared_receipt_with_poi 5. from_this_person_to_poi 6. from_poi_to_this_person	1. deferred_income 2. salary 3. director_fees 4. total_payments 5. long_term_incentive 6. expenses 7. exercised_stock_options 8. restricted_stock_deferred 9. restricted_stock 10. loan_advances 11. bonus 12. total_stock_value 13. deferral_payments 14. other	true(1) or false(0)

I then used *explore_enron_data.py* as a starting point to confirm what I knew already:

```
number of players: 146
total POIs: 18
```

I then had a look at the null values, for everyone and just for POI's. Stats gleaned as follows:

Field	No.of Nulls	POI null values
email address	35	
email address but **no messages**	60	`4: ['FASTOW ANDREW S', 'KOPPER MICHAEL J', 'HIRKO JOSEPH', 'YEAGER F SCOTT']`
salary	51	`1: ['HIRKO JOSEPH']`
restricted stock	36	`1: ['HIRKO JOSEPH']`
bonus	64	`2: ['YEAGER F SCOTT', 'HIRKO JOSEPH']`
long term incentive	80	`6: ['RIEKER PAULA H', 'YEAGER F SCOTT', 'BELDEN TIMOTHY N', 'COLWELL WESLEY', 'SHELBY REX', 'HIRKO JOSEPH']`
director fees	129	`no POIs have values`
restricted_stock_deferred	128	`no POIs with values`
exercised stock options	44	`6: ['KOPPER MICHAEL J', 'COLWELL WESLEY', 'FASTOW ANDREW S', 'BOWEN JR RAYMOND M', 'CALGER CHRISTOPHER F', 'CAUSEY RICHARD A']`
3 finance fields: director_fees, restricted_stock_deferred, exercised_stock_options,	29	`6: see above`

deferred income	97	`7: ['SKILLING JEFFREY K', 'YEAGER F SCOTT', 'GLISAN JR BEN F', 'HIRKO JOSEPH', 'DELAINEY DAVID W', 'KOPPER MICHAEL J', 'KOENIG MARK E’]`
deferral payments	107	`All null except 5: [‘RIEKER PAULA H’],[‘LAY KENNETH L’],[‘BELDEN TIMOTHY N’],[‘COLWELL WESLEY’],[‘HIRKO JOSEPH’]`
loan advances	142	`All null except 1: [‘LAY KENNETH L’]`
other	53	`all POIs have values`
expenses	51	`all POIs have values`
total stock value	20	`all POIs have values`
total payments	21	`all POIs have values`

Thoughts and Questions at this stage:

- All POI’s have expenses, total stock options, total payments and exercised stock options
- There are no POI’s with director fees
- Only 4 people, including one POI had a loan advance
- Deferral payments (39/146), Restricted Stock Deferred (18/146) and Director Fees (17/146) are rare.
- All POIs have email addresses but 4 have no messages
- Why no salary? Freelance?

Joe Hirko and Scott Yeager have a lot of null values. Are they important?

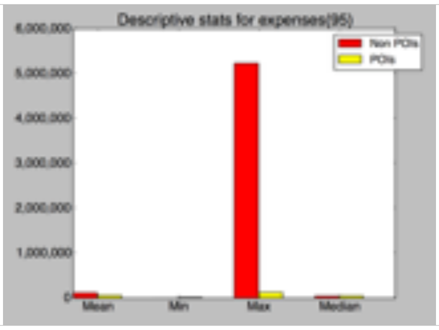
```
{‘email_address’: 'joe.hirko@enron.com',
   'deferral_payments': 10259,
   'expenses': 77978,
   'exercised_stock_options': 30766064,
   'total_stock_value': 30766064,
   'poi': True}

{'email_address': 'scott.yeager@enron.com',
   'other': 147950,
   'salary': 158403,
   'total_payments': 360300,
   'expenses': 53947,
   'restricted_stock': 3576206,
   'exercised_stock_options': 8308552,
   'total_stock_value': 11884758,
   'poi': True}
```

The data they do have shows they could definitely be POIs!

Decision: convert all null values to 0 for the moment.

2. Outliers

To look at potential outlers, I drew descriptive stats graphs for each field. All the fields were behaving really oddly with very high values for non POIs such as expenses below.



I searched for expenses over 1000000 and returned the following:

```
TOTAL : expenses are  5235198
```

That’s the first outlier to remove!

I deleted TOTAL and ran the graphs again and noticed some non-POIs with high values (see below). For this project I'm to assume that means they were exonerated. They might be useful later though:

```
HORTON STANLEY C : exercised_stock_options are  5210569
DERRICK JR. JAMES V : exercised_stock_options are  8831913
CHRISTODOULOU DIOMEDES : exercised_stock_options are  5127155
THORN TERENCE H : exercised_stock_options are  4452476
FREVERT MARK A : salary are  1060932, deferral_payments are  6426990, deferred_income are  -3367011,
exercised_stock_options are  10433518, other are  7427621
DIMICHELE RICHARD G : exercised_stock_options are  8191755
MARTIN AMANDA K : long_term_incentive are  5145434
WALLS JR ROBERT H : exercised_stock_options are  4346544
MCCLELLAN GEORGE : expenses are  228763
OVERDYKE JR JERE C : exercised_stock_options are  5266578
REDMOND BRIAN L : exercised_stock_options are  7509039
BAXTER JOHN C : exercised_stock_options are  6680544
ELLIOTT STEVEN : exercised_stock_options are  4890344
REYNOLDS LAWRENCE : exercised_stock_options are  4160672
URQUHART JOHN A : expenses are  228656
BANNANTINE JAMES M : exercised_stock_options are  4046157
ALLEN PHILLIP K : deferred_income are  -3081055
LAVORATO JOHN J : bonus are  8000000, exercised_stock_options are  4158995
PAI LOU L : exercised_stock_options are  15364167, total_stock_value are  23817930
WHITE JR THOMAS E : restricted_stock are  13847074
```

En route I also noticed the name "THE TRAVEL AGENCY IN THE PARK". Searching for this I noticed there are only other payments. This is outlier number 2, remove it. I checked all the other names manually and they all look like genuine people.


```
{'restricted_stock_deferred': 0, 'from_poi_to_this_person': 0, 'from_this_person_to_poi': 0,
'exercised_stock_options': 0, 'total_payments': 362096, 'long_term_incentive': 0,
'restricted_stock': 0, 'deferral_payments': 0, 'other': 362096, 'to_messages': 0,
'total_stock_value': 0, 'salary': 0, 'email_address': 0, 'loan_advances': 0, 'expenses': 0,
'shared_receipt_with_poi': 0, 'poi': False, 'from_messages': 0, 'bonus': 0, 'director_fees': 0,
'deferred_income': 0}
```

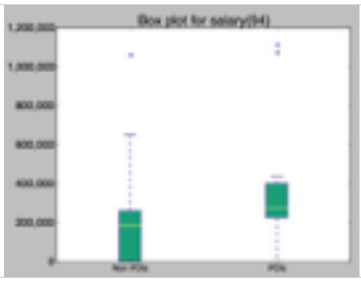
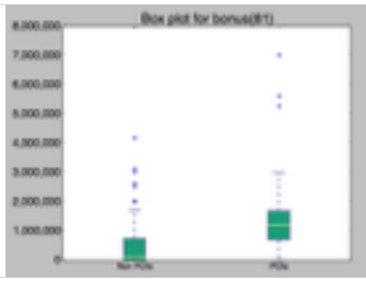
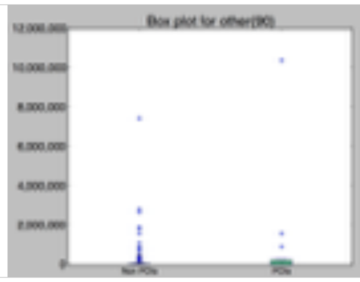
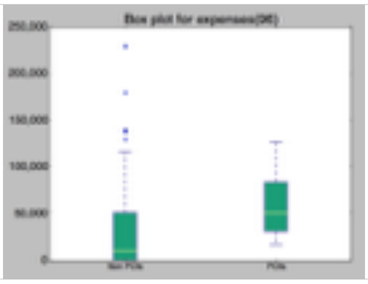
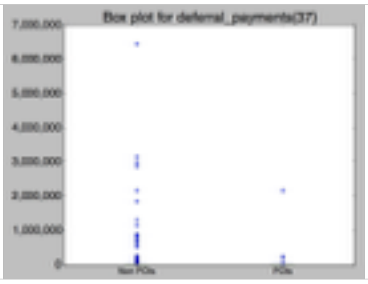
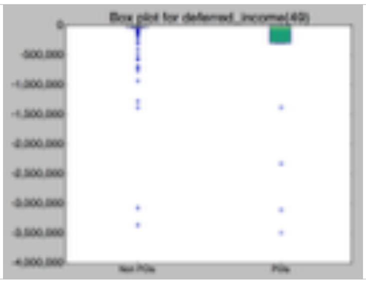
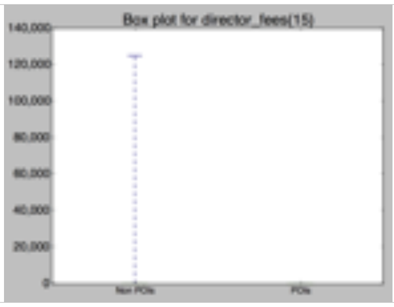
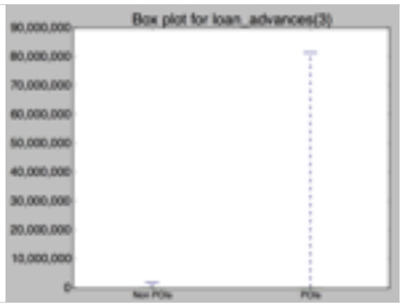
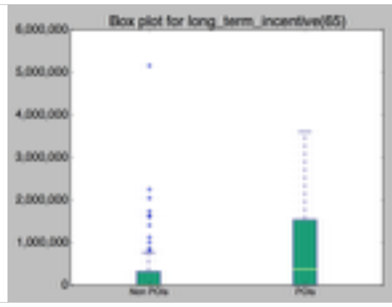

I then decided to look at the total payments field. After a bit of exploration, I discovered it was a sum of the certain financial fields. However there were two that didn't tally:

BHATNAGAR SANJAY **total_payments : 15456290 (137864)** deferred_income : 0 salary : 0 director_fees : 137864 **(0)** long_term_incentive : 0 expenses : 0 **(137864)** exercised_stock_options : 2604490 **(15456290)** restricted_stock_deferred : 15456290 **(-2604490)** restricted_stock : -2604490 **(2604490)** loan_advances : 0 bonus : 0 total_stock_value : 0 **(15456290)** deferral_payments : 0 other : 137864 **(0)**	BELFER ROBERT **total_payments : (3285)** deferred_income : 0 **(-102500)** salary : 0 director_fees : **(102500)** long_term_incentive : 0 expenses : 0 **(3285)** exercised_stock_options : 3285 **(0)** restricted_stock_deferred : 44093 **(-44093)** restricted_stock : 0 **(44093)** loan_advances : 0 bonus : 0 total_stock_value : -44093 **(0)** deferral_payments : -102500 **(0)** other : 0

I went through various options and then discovered the pdf *"enron61702insiderpay.pdf".* There seemed to be a data entry problem. The actual values are in brackets in purple. I therefore updated them manually in the code.

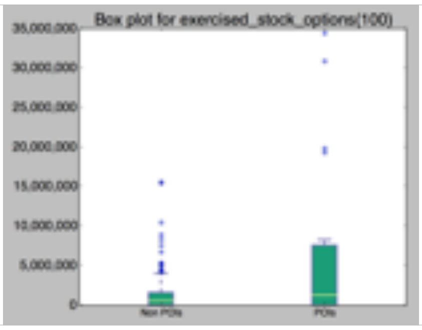
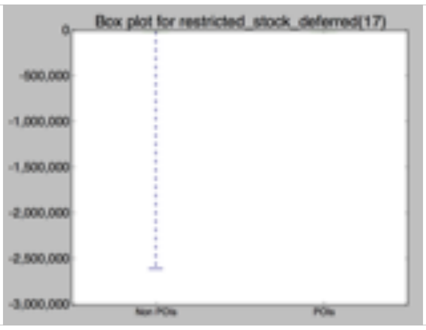
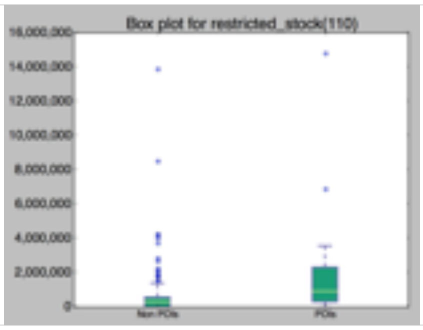
3. Create new features

I used graphs to help me decide on new features. After my first submission the reviewer correctly suggested that box plots would be more informative.

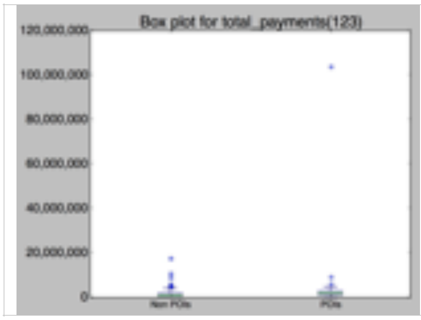
Salary: clearly a key feature	**Bonus:** check relationship with salary.	**Other:** disproportionately high values.
		
Expenses: not significant amounts. **(don't use this feature at this stage)**	**Deferral Payments:** high values again.	**Deferred Income:** compare with payments?
		
Director Fees: Relates to non-employee directors. low amounts and no POIs. **(don't use this feature at this stage)**	**Loan advances:** something dodgy here!	**Long term incentive:** Relates to employee retention which I would argue could put this in the same category as stocks..
		

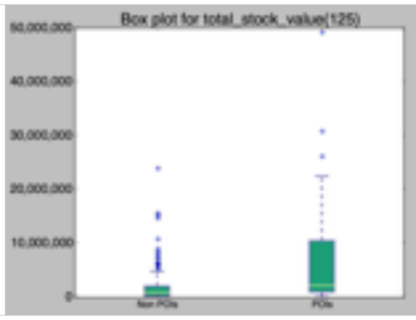
New payment related features:

- *key_payments:* salary + bonus + other
- *deferral_balance:* deferral_payments + deferred_ income

Exercised SO: amount paid out	**Restricted_stock** and **Restricted_stock_deferred:** Only non POIs have deferred. The amounts concerned for deferred are also low compared to restricted stock (which includes deferred). However, as it applies to non-POIs I would like to include it as a total variable.	
		

No new variables here.



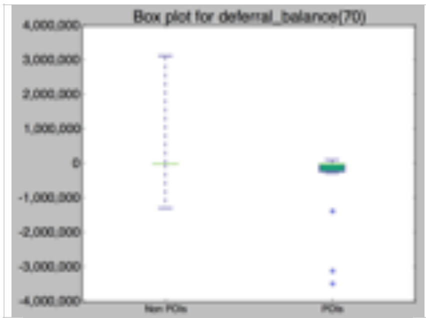


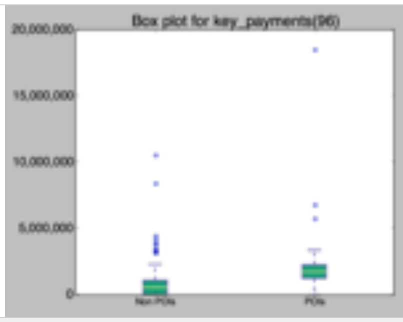
Finally two new features linked to totals:

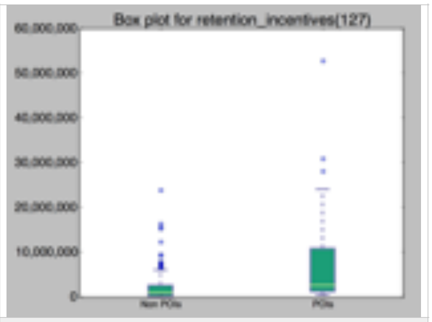
retention incentives: long_term_incentive + total_stock_value

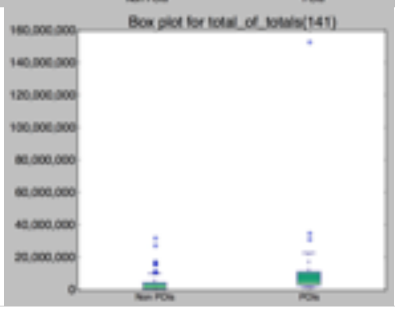
total_of_totals: total_payments + total_stock_value

I then looked at the new features graphically:





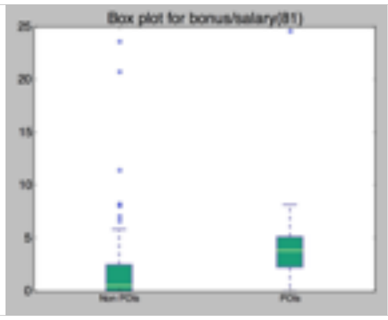


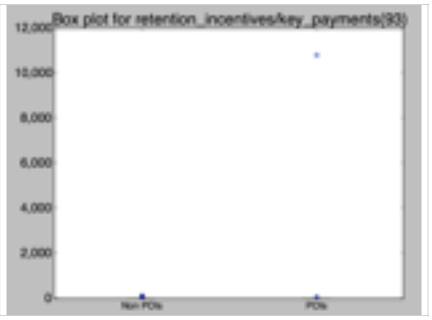


Almost all POIs have a negative deferral balance.

If anything key payments shows less of POI/non-POI split than

I also decided to look at some ratios: *bonus/salary*, *exercised_stock_options/salary*, *retention_incentives/key_payments.*





I used the graphs to compile a new odd_payments features which encompassed the POI only extreme values. This started with a large combination of features but it turned out that *retention_incentives > 28,000,000* and *deferral_balance < -1,500,000* nets the same 5 POIs.

4. Intelligently select features

Firstly, I decided to miss out two features based on the descriptive stats graphs:

- *expenses* - the amounts involved (240,000 or less) were insignificant compared to the other figures.
- *director_fees* - no POI's involved so potentially less relevant.

and some of my new features which didn't seem to go anywhere:

- *bonus/salary*
- *exercised_stock_options/salary*

I then ran SelectKBest** on the remaining fields and got the following:

```
bonus/salary KBEST 10.7835847082
exercised_stock_options/salary KBEST 0.119303870047
poi_emailratio KBEST 5.39937028809
exercised_stock_options/salary KBEST 0.119303870047
odd_payments KBEST 47.4045185584
key_payments KBEST 17.4322739258
deferral_balance KBEST 13.5354825655
retention_incentives KBEST 23.8497535003
total_of_totals KBEST 16.9893364218
salary KBEST 18.2896840434
bonus KBEST 20.7922520472
other KBEST 4.20243630027
deferral_payments KBEST 0.228859619021
deferred_income KBEST 11.4248914854
loan_advances KBEST 7.18405565829
long_term_incentive KBEST 9.92218601319
exercised_stock_options KBEST 22.3489754073
restricted_stock_deferred KBEST 0.768146344787
restricted_stock KBEST 8.82544221992
```

The top 7 features look pretty good.

I played around with GridSearchCV and Kbest options for my chosen algorithms but in the end I concluded that this overkill. KBest scores look good for the first seven which is reasoned enough.

*** First time round I went down the wrong path and decided to choose the features intuitively as KBest seemed to be throwing out inconsistent results. I was clearly not reading the results properly… All part of the learning curve.*

5. Properly scale features

About the features

- they all the same type - integer
- they could all be related
- there are a lot of outliers which are potentially key
- the numbers vary considerably between features

Since they are all integers, I went for a MinMaxScaler.

6. Algorithms and Parameter Tuning

Now for the fun bit! Finding an algorithm and playing around with the features.

For the evaluation metrics I decided to use *precision* and *recall*.

Precision = likelihood that an identified POI is actually a POI. % false alarms.
Recall = likelihood of identifying a POI in the dataset if there was one.

As the rubric asks for above 0.3 in both I'm assuming they are of equal importance.

I needed to split the data. Initially I used train_test_split but then decided to take advantage of the validator in *tester.py* which uses Stratified Shuffle Split to create the train and test data and has a very detailed report.

I tested out five algorithms using the MinMaxScaler, KBest k=7. The GaussianNB result was pretty good: *Precision* 0.48193, *Recall* 0.28. There are no further parameters to tune but it's an indication I'm on the right track.

I started out using GridSearchCV to tune the parameters but the search time was excruciatingly slow and I decided it would be a better learning experience if I iterated through the parameters myself and got a better idea of their properties. Using the Sklearn website, I worked systematically on the remaining four algorithms.

Decision Tree			Logistic Regression			LinearSVC			Random Forest		
	p	r		p	r		p	r		p	r
initial result	0.40367	0.39600		0.80662	0.15850		0.74294	0.25000		0.51138	0.23600
criterion = entropy	0.39067	0.35600	c = 1000	0.51330	0.26050	c=1000	0.29516	0.29900	n_estimators	improves balance not speed.	
splitter = random	0.41445	0.4820	As C increases, precision decreases and recall increases			loss='hinge'	1	0.27550	criterion = 'entropy'	0.52625	0.22050
max features	best at 7 (as you'd expect)		class weight = balanced	**0.33021**	**0.45850**	fit intercept = false	0.31234	0.24300	bootstrap ='false'	0.51796	0.32450
min_samples_split = 5	0.46245	0.39100	parameters with no impact	fit intercept = false, intercept scaling verbose, njobs		class weight = balanced	0.30977	0.48850	oob score n_estimators = 100	0.59936	0.27900
class weight = balanced	0.31463	0.27750				parameters with no impact	verbose, max_iter,dual		class weight = balanced	0.49195	0.16800
random state = '42'	0.39731	0.38400				I then looked at the various combinations and optimum result was.			As you can't have out of bag estimation if bootstrap=False and that had the best result I went for		
min_samples_split = 5 *and* splitter = random	**0.50879**	**0.41950**				C=10, class *and* weight = balanced	**0.31288**	**0.56600**	n_estimators = 100 *and* bootstrap = False	**0.51608**	**0.33700**

Finally I tested my best algorithm (Decision Tree, min_sample_splits = 5, splitter = random) with different KBest values and it turned out that k=6 was marginally better. Here are the final full results:

```
Pipeline(steps=[('Scaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('SKB', SelectKBest(k=6,
score_func=<function f_classif at 0x107b47e18>)), ('Decision Tree',
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=5, min_weight_fraction_leaf=0.0,
            presort=False, random_state=42, splitter='random'))])
      Accuracy: 0.88127 Precision: 0.56616     Recall: 0.46850   F1: 0.51272      F2: 0.48524
      Total predictions: 15000     True positives:  937    False positives:  718   False negatives:
1063  True negatives: 12282
```

Note that I've changed random state to 42 to maintain consistent results.

Conclusion

So the final results means that 57% of POIs identified are definitely POIs and 47% of POIs in the dataset are being identified.

I'm sure there are other things I could do to improve this score. I didn't perform PCA and I did not use a Parameter Tuning algorithm. However, the major drawback is the restriction of a very small dataset (144 after outlier removal) with a small % of POIs - 12.5%. Ideally more data would be released for the 18 POIs identified who had no financial records in the dataset. This however is unlikely after so many years.

The logical next step therefore would be to look at the actual emails and tracking links between POIs and other email addresses and identifying key words which are more common when POIs are involved.

This has been a fabulous project. It has been a learning curve twice over and I would like to come back to the data again and have a dig into the email content linked to POIs.

The code is found in four python files:

- *explore_data.py*
- *feature_selection.py (commented out in this case)*
- *algorithms.py*
- *poi_id.py*