Enron Submission Free-Response Questions

A critical part of machine learning is making sense of your analysis process and communicating it to others. The questions below will help us understand your decision-making process and allow us to give feedback on your project. Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response!

When your evaluator looks at your responses, he or she will use a specific list of rubric items to assess your answers. Here is the link to that rubric: [**Link**] Each question has one or more specific rubric items associated with it, so before you submit an answer, take a look at that part of the rubric. If your response does not meet expectations for all rubric points, you will be asked to revise and resubmit your project. Make sure that your responses are detailed enough that the evaluator will be able to understand the steps you took and your thought processes as you went through the data analysis.

Once you've submitted your responses, your coach will take a look and may ask a few more focused follow-up questions on one or more of your answers.
We can't wait to see what you've put together for this project!

Notes on the resubmission:

Two outliers have been removed from the data set. One was the person TOTAL, which was not a person at all. The other was a person who had a null set of data.

In order to determine why my results were not anywhere near the same as the results of the official tester, I had to rewrite a lot of my code. First, I abandoned the idea of temporarily using a data frame and rewrote all of the code, including in the replace_nan_with_mean and feature_nulls_analyze functions to use the original dicitonary. Then, I took out any previous scaling I had done and the label encoder. Next, I introduced pipelines to use with scaling. After this, I rewrote the param_grids for all three algorithms and then tested each of them with and without scaling.

In this next version of the project, SelectKBest is implemented into the pipelines as well as the grid searches.

Parameter grids have been updated once again in order to consider website examples of parameter grids, which have been included in the sources.

Research shows that feature scaling is not needed for RandomForest or AdaBoost, but is needed for SVC(). I included scaling in the pipeline this time for SVC() only.

In addition, I tried using a stratified shuffle split on the data, hopefully mimicking the tester.py function.

Finally, in this most recent rendition of the project, I divided the features into financial and behavioral features, and also used PCA to reduce the financial features to one feature.

It turns out that none of my algorithms met the requirements, so I added in Naïve Bayes.

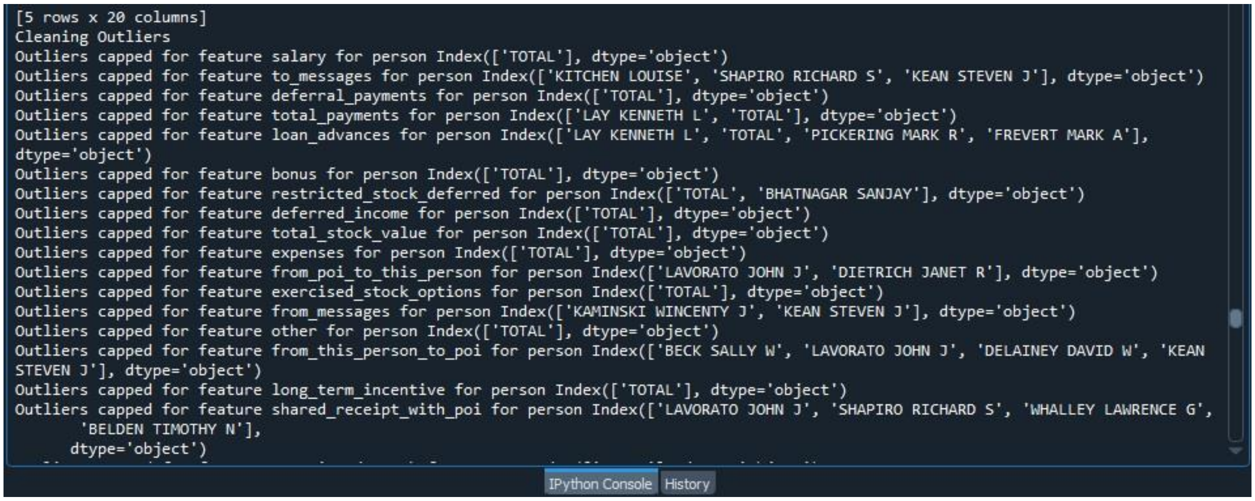
1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the

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”]

This data set includes a list of Enron employees. The ‘poi’ column is a binary column that states whether they were a person of interest in the Enron case. The rest of the columns are various statistics about each person, and we will be using those statistics to run machine learning algorithms and see if we can get one that can make decent predictions about who is a poi or not.

There are 146 people in the dataset with 20 numerical features and 1 label, the ‘poi’ column.

There were some outliers in the data, and we used an outlier cleaner at first to cap the top and bottom of the data using a z-score method. Anything above a z-score of 3 was capped at a z-score of 3, and those data points are printed to the console. However – as outliers may indicate fraud, we are going to put the automatic outlier cleaner on hold.



```
[5 rows x 20 columns]
Cleaning Outliers
Outliers capped for feature salary for person Index(['TOTAL'], dtype='object')
Outliers capped for feature to_messages for person Index(['KITCHEN LOUISE', 'SHAPIRO RICHARD S', 'KEAN STEVEN J'], dtype='object')
Outliers capped for feature deferral_payments for person Index(['TOTAL'], dtype='object')
Outliers capped for feature total_payments for person Index(['LAY KENNETH L', 'TOTAL'], dtype='object')
Outliers capped for feature loan_advances for person Index(['LAY KENNETH L', 'TOTAL', 'PICKERING MARK R', 'FREVERT MARK A'],
dtype='object')
Outliers capped for feature bonus for person Index(['TOTAL'], dtype='object')
Outliers capped for feature restricted_stock_deferred for person Index(['TOTAL', 'BHATNAGAR SANJAY'], dtype='object')
Outliers capped for feature deferred_income for person Index(['TOTAL'], dtype='object')
Outliers capped for feature total_stock_value for person Index(['TOTAL'], dtype='object')
Outliers capped for feature expenses for person Index(['TOTAL'], dtype='object')
Outliers capped for feature from_poi_to_this_person for person Index(['LAVORATO JOHN J', 'DIETRICH JANET R'], dtype='object')
Outliers capped for feature exercised_stock_options for person Index(['TOTAL'], dtype='object')
Outliers capped for feature from_messages for person Index(['KAMINSKI WINCENTY J', 'KEAN STEVEN J'], dtype='object')
Outliers capped for feature other for person Index(['TOTAL'], dtype='object')
Outliers capped for feature from_this_person_to_poi for person Index(['BECK SALLY W', 'LAVORATO JOHN J', 'DELAINEY DAVID W', 'KEAN
STEVEN J'], dtype='object')
Outliers capped for feature long_term_incentive for person Index(['TOTAL'], dtype='object')
Outliers capped for feature shared_receipt_with_poi for person Index(['LAVORATO JOHN J', 'SHAPIRO RICHARD S', 'WHALLEY LAWRENCE G',
       'BELDEN TIMOTHY N'],
      dtype='object')
```

Instead, I wrote functions to look at the number of nulls per feature and per person in the dictionary.

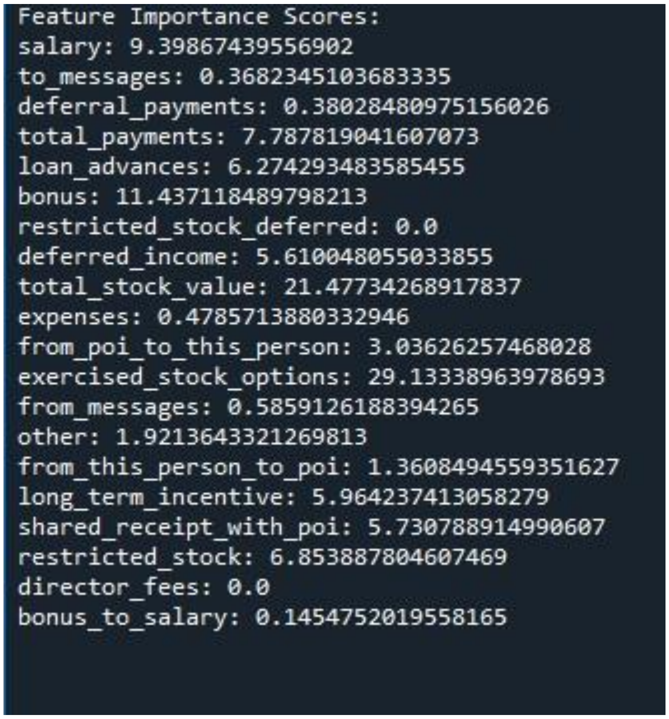
I found that this person: LOCKHART EUGENE E had a null dataset entirely, so I removed him from the dictionary.

I found that the person TOTAL was not a person at all, and removed them.

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”]

At first, I decided to use all the numeric features in the POI identifier so that I could max out the classifiers with as much data as possible. However, I decided use SelectKBest to determine the “best” features, and here is the result of the test, including the new feature:



```
Feature Importance Scores:
salary: 9.39867439556902
to_messages: 0.3682345103683335
deferral_payments: 0.38028480975156026
total_payments: 7.787819041607073
loan_advances: 6.274293483585455
bonus: 11.437118489798213
restricted_stock_deferred: 0.0
deferred_income: 5.610048055033855
total_stock_value: 21.47734268917837
expenses: 0.4785713880332946
from_poi_to_this_person: 3.03626257468028
exercised_stock_options: 29.13338963978693
from_messages: 0.5859126188394265
other: 1.9213643321269813
from_this_person_to_poi: 1.3608494559351627
long_term_incentive: 5.964237413058279
shared_receipt_with_poi: 5.730788914990607
restricted_stock: 6.853887804607469
director_fees: 0.0
bonus_to_salary: 0.1454752019558165
```

I did create a new feature which I included in the tests – this was bonus / salary. I wanted a number that would express any bonus they made relative to their salary, as people with higher salaries might get higher bonuses.

However, this new feature was not of high relevance according to selectKBest. **I decided to use SelectKBest in the pipelines with a combination of grid search and trial-and-error to determine the number of features to keep for each classifier.** Bonus_to_salary was not included due to its SelectKBest score.

I also used PCA to reduce the financial features to one feature after separating those from the behavioral features. Then, I made my features_list = behavioral_features + financial_features.

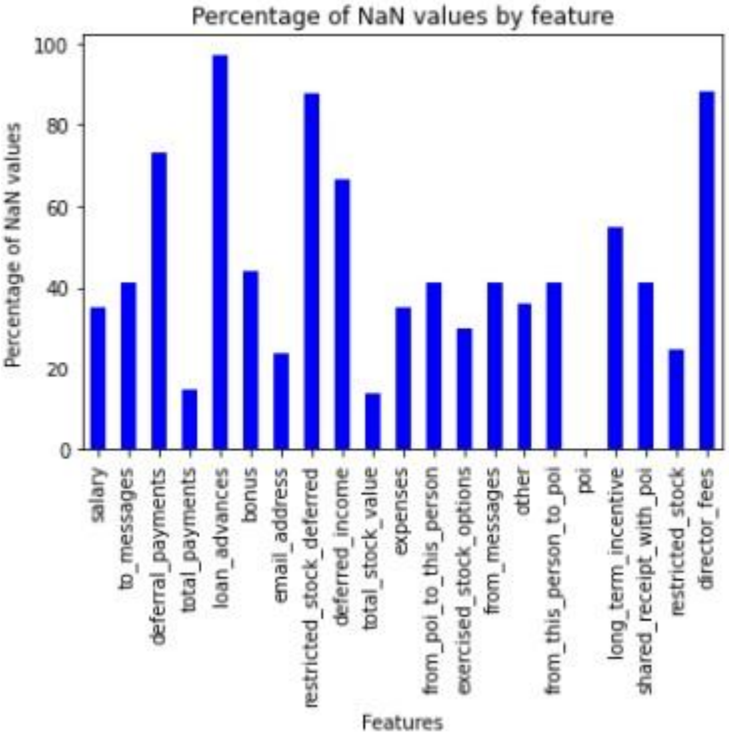
There are 146 people in the original dataset with 20 numerical features and 1 label, the ‘poi’ column.

NAN VALUES were present in the following columns and were replaced by the column mean. As you can see, the label “poi” had no NaN values. After getting a chart like this in one of my critiques of the project, I was able to reproduce the code to make a similar bar graph. The function is called feature_nulls_analyze() and is located in outlier_cleaner.py, which is where I added a few more functions for the project.

When analyzing the null values, I learned that they were actually “NaN” strings – therefore, I had to convert them to np.nan using the following line of code:

```
data_frame.replace("NaN", np.nan, inplace=True)
```

Then I ran the replace_nan_with_mean() function I wrote to replace NaN values with the mean of each feature.

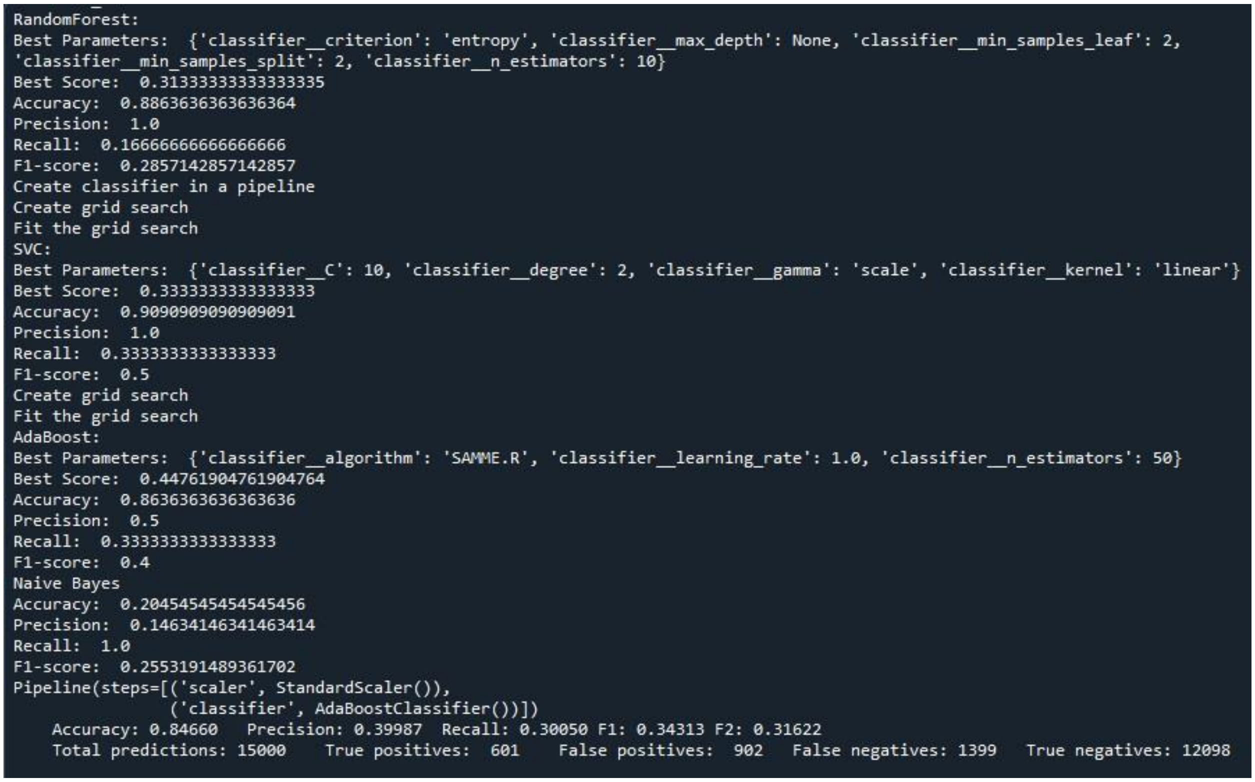


Scaling has been done using pipelines for all three chosen algorithms.

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 RandomForest, SVC and AdaBoost because those seemed like good choices to use when we have a binary label. I also used Naïve Bayes as a control.

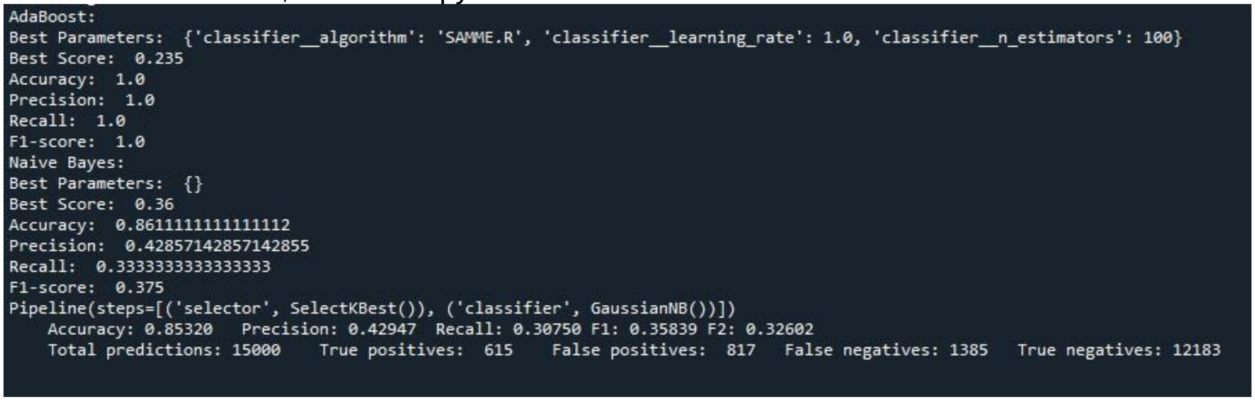
Here are the evaluation metrics for them all and best grid search parameters, before all the changes:



```
RandomForest:
Best Parameters:  {'classifier__criterion': 'entropy', 'classifier__max_depth': None, 'classifier__min_samples_leaf': 2,
'classifier__min_samples_split': 2, 'classifier__n_estimators': 10}
Best Score:  0.31333333333333335
Accuracy:  0.8863636363636364
Precision:  1.0
Recall:  0.16666666666666666
F1-score:  0.2857142857142857
Create classifier in a pipeline
Create grid search
Fit the grid search
SVC:
Best Parameters:  {'classifier__C': 10, 'classifier__degree': 2, 'classifier__gamma': 'scale', 'classifier__kernel': 'linear'}
Best Score:  0.3333333333333333
Accuracy:  0.9090909090909091
Precision:  1.0
Recall:  0.3333333333333333
F1-score:  0.5
Create grid search
Fit the grid search
AdaBoost:
Best Parameters:  {'classifier__algorithm': 'SAMME.R', 'classifier__learning_rate': 1.0, 'classifier__n_estimators': 50}
Best Score:  0.44761904761904764
Accuracy:  0.8636363636363636
Precision:  0.5
Recall:  0.3333333333333333
F1-score:  0.4
Naive Bayes
Accuracy:  0.20454545454545456
Precision:  0.14634146341463414
Recall:  1.0
F1-score:  0.2553191489361702
Pipeline(steps=[('scaler', StandardScaler()),
                ('classifier', AdaBoostClassifier())])
    Accuracy: 0.84660   Precision: 0.39987  Recall: 0.30050 F1: 0.34313 F2: 0.31622
    Total predictions: 15000    True positives:  601    False positives:  902   False negatives: 1399   True negatives: 12098
```

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”]

Here are the results of my evaluation metrics again, after changes are made. Notice that the tester.py is giving vastly different results than my code. It turns out Naive Bayes works while none of the others do, with tester.py.



```
AdaBoost:
Best Parameters:  {'classifier__algorithm': 'SAMME.R', 'classifier__learning_rate': 1.0, 'classifier__n_estimators': 100}
Best Score:  0.235
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
F1-score:  1.0
Naive Bayes:
Best Parameters:  {}
Best Score:  0.36
Accuracy:  0.8611111111111112
Precision:  0.42857142857142855
Recall:  0.3333333333333333
F1-score:  0.375
Pipeline(steps=[('selector', SelectKBest()), ('classifier', GaussianNB())])
	Accuracy: 0.85320	Precision: 0.42947	Recall: 0.30750 F1: 0.35839 F2: 0.32602
	Total predictions: 15000	True positives:  615	False positives:  817	False negatives: 1385	True negatives: 12183
```

The parameters used when running an algorithm can create vastly different results depending on what is chosen. This means that the algorithm can perform poorly if it is not optimized. Therefore, I used a grid search on all three algorithms in order to determine the optimal parameters.

```
### Paramgrid for RandomForest

param_grid = {'classifier__n_estimators': [2, 5, 10, 50, 100, 200],

              'classifier__max_depth': [None, 5, 10, 20, 40, 50],

              'classifier__min_samples_split': [2, 5, 10],

              'classifier__min_samples_leaf': [1, 2, 4],

              'classifier__criterion': ['gini', 'entropy'],

              'classifier__max_features': ['auto', 'sqrt']}
}




# Define the parameter grid for the SVC

param_grid = {'classifier__C': [0.1, 1, 10],

              'classifier__kernel': ['linear', 'rbf', 'poly'],

              'classifier__degree': [2,3,4],

              'classifier__gamma': ['scale', 'auto']}
```

```
### Define the parameter grid for the AdaBoostClassifier

param_grid = {'classifier__n_estimators': [10, 50, 100, 200,500],

          'classifier__learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.5, 1.0],

          'classifier__algorithm': ['SAMME', 'SAMME.R']}
```

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”]

I validated all three chosen algorithms by dividing the data into test and training data, and then using the test results to determine various evaluation metrics for each algorithm, including accuracy, recall, precision and f1 score. Since the algorithms were trained on the training data, we can test them on the test data and validate that way.

If you do not use this method, you have not tested the algorithm on data it has not seen yet, only on the data it was trained on. This can make the algorithm over-perform.

If you don’t look at precision or recall, you might have a high accuracy that is irrelevant. In the context of this project, for example, if you said no one was a poi, you would have a high accuracy just because there are so many non-pois.

The test_classifier() function in tester.py uses a stratified shuffle split in order to cross-validate the data. I used this as well in my algorithms in combination with grid search.

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 best performance was with AdaBoost using optimal parameters and it was the following:

I will be using accuracy, precision, recall and f1 scores as evaluation metrics.

AdaBoost:

Best Parameters: {'classifier__algorithm': 'SAMME.R', 'classifier__learning_rate': 1.0, 'classifier__n_estimators': 100}

Best Score: 0.235

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

F1-score: 1.0

Even with perfect accuracy, precision and recall tester.py rejected AdaBoost in favor of Naïve Bayes.

The accuracy of 1 means that the algorithm was able to make the correct determination 100% of the time.

The precision of 1 means that 100% of the time the algorithm chose someone as a person of interest, they were one.

The recall of 1 means that 100% of the time, the people of interest were discovered by the algorithm.

The f1 score is the harmonic mean of recall and precision. This is used to balance the trade-off between recall and precision by using one of their means.

Notes on the code, before latest updates:

The first thing that I do is transform the dictionary into a dataframe, which I found easier to use when writing outlierCleaner and replace_nan_with_mean, both of which are included in the outlier_cleaner.py file.

outlierCleaner uses a z-score method to cap any outliers above a z-score of 3 to a value at a z-score of 3.

Replace_nan_with_mean calculates the mean of each column and replaces any nan values with that.

Next, selectKBest is used in order to do feature analysis and those scores are printed out.

After this, I scale the features_test and features_train variables

Next, I try three different classifiers – RandomForest, adaboost and SVC. I used search grids on each one to find the best values to use for the parameters.

Updates:

All of the code was rewritten to work with dictionaries and I kept the data_dict as a dictionary the whole time. Pipelines have been added to the classifiers for use with scaling, and the original scaling has been deactivated. A different outlier cleaner method was deployed looking for persons or features with null data sets.