Answers to Machine Learning Project 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"]*

   The Enron case is the largest story of corporate fraud that led to many people losing their job and the company going bankrupt. The data set contains data of individuals who were employed to Enron or connected to the company. Its contains financial data, email data from 146 individuals, and identifies whether a certain individual is a POI of interest. The dataset contains 21 features that we can use to help us train a machine learning algorithm to identify POIs which is the goal of this project. I have identified 3 outliers in the data set, "TOTAL"," THE TRAVEL AGENCY IN THE PARK, and "LOCKHART EUGENE E" Once these were identified, the outliers were removed and I continued with the project using 143 data points.

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"]*

   For my feature engineering, I created two new features:
   - fraction_from_poi
   - fraction_to_poi

   I created these new features to get a sense of the fraction of emails that a person receives from a POI. I'm hoping this gives me some insight when trying to identify POIs.

   To start off my feature selection process, I decided to separate the financial data from the email data and find the top features from each group. In order to find the top features from each group, I used SelectKBest as my method to score each feature. I used the 'all' parameter for K to ensure I get a score for every feature. I picked the top 4 features from each group and combined them into another list. I executed SelectKBest again using k='all' to get new scores for the new feature list to explore the importance of each feature from the combined group. After going through this process, the features I used are below:

Feature	Score
exercised_stock_options	21.71552656

total_stock_value	21.05899501
bonus	17.8573624
salary	15.14904119

Just to check for consistency, I decided to run SelectKBest on the entire list of features to confirm if these 4 features were the top 4 from the entire list. I also ran SelectPercentile as well. After running SelectKBest and SelectPercentile on the entire feature list, I confirmed my initial selections were the top four. The entire feature scores can be seen below.

Feature	Score
'exercised_stock_options'	24.81507973
'total_stock_value'	24.18289868
'bonus'	20.79225205
'salary'	18.28968404
'fraction_to_poi'	16.40971255
'deferred_income'	11.45847658
'long_term_incentive'	9.92218601
'restricted_stock'	9.21281062
'total_payments'	8.77277773
'shared_receipt_with_poi'	8.58942073
'loan_advances'	7.18405566
'expenses'	6.09417331
'from_poi_to_this_person'	5.24344971
'other'	4.18747751
'fraction_from_poi'	3.12809175
'from_this_person_to_poi'	2.38261211
'director_fees'	2.1263278
'to_messages'	1.64634113
'deferral_payments'	0.22461127
'from_messages'	0.16970095
'restricted_stock_deferred'	0.06549965

To further validate my feature selections, I created a scatter plot matrix of the top 4 features from the email and financial lists to visualize and pick out any patterns in the data that could be useful. I also tried different combinations of these features with the classifiers and compared the recall and precision from each combination. I left the different combination commented out in the code so you can see the different combinations I tried. Please the performance metrics from each combination I attempted below. Please note that Feature List 1 is the feature list that contains the top 4 features from the entire feature list based of

the scores from SelectKBest. Since the classifiers I used are not affected by feature scaling, I did not perform any.

Classifier	Precision	Recall	True positives	True negatives	Feature List	Parameter Tuned?
Naïve Bayes	0.503	0.323	646	10362	1	No
Decision Tree	0.324	.338	675	9596	1	No
Random Forest	.497	.240	480	10514	1	No
K nearest Neighbor	0.66	0.269	537	10723	1	No
Decision Tree	0.307	.307	615	9610	1	Yes
Random Forest	0.389	.266	532	10163	1	Yes
K nearest Neighbor	0.49	0.382	763	10205	1	Yes
Naïve Bayes	0.25	0.112	224	9328	2	No
Decision Tree	0.4	0.31	613	9065	2	No
Random Forest	0.377	0.18	352	9394	2	No
K nearest Neighbor	0.001	0.001	2	9729	2	No
Decision Tree	0.4	0.309	618	9071	2	Yes
Random Forest	0.35	0.19	378	9310	2	Yes
K nearest Neighbor	0.246	0.226	452	8613	2	Yes
Naïve Bayes	0.142	0.092	92	7444	3	No
Decision Tree	0.353	0.415	415	7241	3	No
Random Forest	0.332	0.253	253	7491	3	No
K nearest Neighbor	0.06	0.05	49	7167	3	No
Decision Tree	0.328	0.406	406	7169	3	Yes
Random Forest	0.339	0.343	343	7331	3	Yes
K nearest	0.233	0.307	307	6987	3	Yes

Neighbor						
Naïve Bayes	0.273	0.11	220	9415	4	No
Decision Tree	0.435	0.359	718	9067	4	No
Random Forest	0.35	0.154	308	9429	4	No
K nearest Neighbor	0.007	0.001	2	9729	4	No
Decision Tree	0.365	0.3	600	8958	4	Yes
Random Forest	0.38	0.229	457	9255	4	Yes
K nearest Neighbor	0.246	0.226	452	8613	4	Yes
Naïve Bayes	0.39	0.109	218	8659	5	No
Decision Tree	0.343	0.296	591	7870	5	No
Random Forest	0.333	0.144	288	8424	5	No
K nearest Neighbor	0.007	0.001	2	8735	5	No
Decision Tree	0.332	0.308	616	7763	5	Yes
Random Forest	0.335	0.138	276	8453	5	Yes
K nearest Neighbor	0.277	0.239	478	7753	5	Yes
K nearest Neighbor	0.007	0.001	2	9729	6	No
Naïve Bayes	0.25	0.112	224	9328	6	No
Decision Tree	0.265	0.228	456	8732	6	No
Random Forest	0.16	0.056	111	9413	6	No
Decision Tree	0.247	0.203	406	8761	6	Yes
Random Forest	0.181	0.096	192	9132	6	Yes
K nearest Neighbor	0.246	0.226	452	8613	6	Yes
Naïve Bayes	0.467	0.324	647	10263	7	No
Decision Tree	0.313	0.303	606	9669	7	No
Random Forest	0.469	0.219	438	10505	7	No

K nearest Neighbor	0.647	0.258	516	10718	7	No
Decision Tree	0.377	0.367	734	9788	7	Yes
Random Forest	0.488	0.237	473	10504	7	Yes
K nearest Neighbor	0.623	0.292	583	10645	7	Yes

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"]

Since I didn't have a particular algorithm to try in mind, I chose to iterate through several classifiers to evaluate different metrics without making any parameter tunes to get a baseline of how each classifier performs. This will help me choose which classifier to focus tuning parameters on. When iterating, I captured the accuracy of a feature train/test split as well as a mean accuracy when running Stratisfied Shuffle split validation test to compare the two scores. I felt it was necessary to perform Stratisfied Shuffle split cross validation because of the imbalance in POIs and non POIs and to ensure all data is included in a test and a training procedure. Additionally, I dumped out the classifier, dataset, and feature list in each iteration so I can call tester.py. I recorded all the scores the tester file provides scores of each classifier. After iterating through each classier, I observed the following metrics:

Classifier	Precision	Recall	Mean SSS CV Recall	F1 Score	True positives	False positives	False negatives	True negatives
Decision Tree	0.321	0.334	0.2	0.327	667	1408	1333	9592
Naïve Bayes	0.503	0.323	0.25	0.393	646	638	1354	10362
K nearest Neighbor	0.66	0.267	0.3	0.382	537	277	1463	10723
Random Forest	0.498	0.232	0.1	0.316	463	466	1537	10534

Due to the drawbacks accuracy can have when evaluating a classifier, I am choosing to focus on Precision and Recall. The classifiers that have the highest precision are:

- Naive Bayes
- K Nearest Neighbor

The classifiers that have the highest recall are:

- Decision Tree
- Naive Bayes

After making parameter tunes on Decision Tree, Random Forest, and K Nearest Neighbor, I observed the following metrics:

Classifier	Precision	Recall	F1 Score	True positives	False positives	False negatives	True negatives
Decision Tree	.342	.348	0.345	695	1336	1305	9664
Random Forest	.418	.279	0.334	557	774	1443	10226
K nearest Neighbor	0.49	.382	0.429	763	795	1237	10205

Cleary, the algorithm the benefited the most from parameter tuning was K Nearest Neighbor. It achieved the highest recall from all the classifiers I explored with and without parameter tunes. This was the classifier I chose for my final analysis. After looking at the post here regarding feature scaling for knn, I decided to evaluate how feature scaling affected the performance.

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"]*

   Parameters can make a huge difference in coming up with decision boundaries in classifiers. Tuning parameters means that you can set certain parameters in a given classifier to help control overfitting and help improve model performance. If this is not done correctly, your models can end up overfitting the data and can lead to inaccurate predictions and degraded performance. The algorithms and parameters I tuned can be found below:

   - Decision Tree
     - criterion(gini,entropy)
     - splitter (best,random)
     - min_samples_split:[2,50,100,1000]

    - max_features:[1,2,3,4]
- Random Forest
    - min_samples_split:[2,50,100]
    - n_estimators:[5,10,50,100]
- K Nearest Neighbors
    - n_neighbors[1-10]

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 a technique used in machine learning to make sure an algorithm is performing the way you want it to perform against certain metrics you chose. We perform validation by splitting our data into training and testing data. This gives estimates of performance on an independent dataset and serves as a check on overfitting. A mistake you can do in validation is call a fit function on a test data set. We want to look for patterns in training sets and then use the test sets to validate steps we take when fitting data.

   I used a combination of Train/Test split and Stratisfied Shufflesplit cross validation to validate my analysis. I used a combination of both so I could compare how each validation performed against each other.

6. *Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something humanunderstandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]*

   After exploring several different classifiers with parameter tunes, I decided to use K Nearest Neighbor as my classifier. The 2 evaluation metrics I used for this classifier are below:

   - Recall
       - Average recall = .3
       - Having good recall means that nearly every time a POI shows up in my test set, I am able to identify him or her. The cost of this is that some non-POIs get flagged
       - It seems that feature scaling didn't an effect on the outcome.
   - Precision
       - Average Precision = .35
       - Having good precision means that whenever a POI gets flagged, I know with confidence that it's very likely to be a POI. The cost of this is that I sometimes miss real POIs.
       - It seems that feature scaling didn't an effect on the outcome