Enron Project

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 Machine Learning Project is all about using the public record available for the Enron corporate fraud, to build a Person of Interest Identifier.

The dataset I got to work with, came as a pickle file, created by a Python module. The dataset is a dictionary saved into a pickle file. The keys of the dictionary are employees of Enron. The values are again dictionaries with key-value pairs giving further information about an employee.

The dataset gives a chance to look at the employees and their financial and email information. It tells whether each employee is a Person of Interest in the fraud or not. Using this information, one can apply Machine Learning tactics to build a Model that can identify a new Person of Interest.

There are 146 data points in the dataset with 18 POIs and 128 Non-POIs. The dataset contains 21 features for each employee. 10 features inform about the payments made to the employees, 4 features about the stock value of each employee, 6 features about their e-mails and 1 feature about their POI status. 46 data points have more than 50% of their features missing values.

In the very beginning stages of data exploration, I plotted the features salary and bonus as a scatterplot. It showed a clear outlier. Looking into the original data source helped in spotting this outlier. It was the data point with dictionary key TOTAL. It was a spreadsheet quirk. After removing it, I could see few more outliers in the scatterplot. But these were valid data points and would certainly help me in identifying Persons of Interest. So I kept them.

Then I looked at the NaN values. Some data points had way more NaN values than others. The data point with most NaN values, in fact had all features except “poi” with missing values. So I removed it. Then looking at the next 3 data points with maximum missing values I saw an entry which did not look like an employee. It was a travel agency. So I removed it.

I did not attempt to do any further Data Cleaning.

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

Enron Project

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”]

After knowing the dataset, I started experimenting with the features without assessing them. I added all the features to the features_list (except email address). I already had a basic skeleton code that took all the features in the features_list, converted the data from dictionary format to python list, and also separated the labels from the features.

To reduce the number of features, I tried the Linear model, Lasso, along with SelectFromModel. I used the features thus selected with many different classifiers, and got good accuracy but very poor precision score and recall score. I realized that I was using a regression model, while I should be using a classification model.

Then I tried selecting features using SelectPercentile and f_classif. Again I got good accuracy, but very poor precision and recall scores.

Next I tried PCA. Principal Component Analysis takes n number of features and transforms them to n number of orthogonal components, with the most prominent component representing the maximum variance in the data, and each consecutive component having less and lesser variance. I chose the 2 most variant components by finding the “explained_variance_ratio_” of the PCA. I tried different numbers of components and checked the evaluation metrics too.

Feature Scaling is an important preprocessing step for PCA. After reading about the different scalers, I tried using RobustScaler because my data had outliers. But I soon realized that I need a scaler that is sensitive to outliers. Hence I tried StandardScalar, but finally chose MinMaxScaler.

I created two new features in my dataset, “poi_fraction_to_messages” and “poi_fraction_from_messages”. I believe that if a person is a POI (Person of Interest) he/she would exchange comparatively more messages with another POI. So, the POI fraction of all the to and from messages seemed to be reasonable features. Since I added these 2 features, I removed from the features_list, the 4 features that I had used to create my new features.

Also, I removed the features, “total_payments” and “total_stock_value” since their values were simply sum of values of some other features.

Finally, when I used GridSearchCV, and it chose the best parameters for me, it chose only 1 component for the pca. This component explains about 33% of the variation. Looking at the coefficients of this Principle Component, the features that are most strongly correlated with this Component are in descending order, “poi_fraction_from_messages”, “salary”, “shared_receipt_with_poi”, “bonus” and so on.

The precision score and recall score are 0.417 and 0.328 when I do not include

my new features in the features_list. And when I replace 4 features with my 2 new features the precision score and recall score are 0.58 and 0.35. So, sure enough, they improve the scores !

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”]

After feature scaling and feature selection, it was time to choose a classifier that will identify POI s with a good precision and recall. Logic told me that I should use LinearSVC from SVM. The scikit learn tutorial gives a nice flowchart to choose an estimator. It told me that if I am predicting a category and if I have labeled data, then LinearSVC should be a good fit. It worked, but did not give me good precision and recall scores. So I tried as many other classifiers like Naïve Bayes, KNeighborsClassifier, RandomForestClassifier, as I could. None of them gave me good results.

Then came Parameter Tuning !

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”]

I went back to LinearSVC and started tuning its parameters. I changed the default values of penalty, dual, C, max_iter. None of that helped. And then, I changed class_weight from None to 'balanced' and got good scores.

I read somewhere :

“Balanced Linear SVM, means that each observation has a weight inverse to its frequency, so that it "oversamples" from the minority class and "undersamples" from the majority class. “

From this I understand that in my dataset, the POIs are oversampled since they are minority class and Non-POIs are undersampled since they are majority class. Thus the classes are balanced.

For rest of the parameter tuning, I used the GridSearchCV. I used it to tune the pca parameters as mentioned earlier. For the GridSearchCV I used recall as the scoring method.

It is important to split the available data into a development set (fed to the GridSearchCV instance) and an evaluation set to compute performance metrics. Because of the small size of the dataset, I used StratifiedShuffleSplit cross validation. It

ensures that an equal ratio of POIs to non-POIs are found in the training and testing sets.

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"]

When we create classifiers, or preprocessing algorithms or feature selection models, we should not test them on the same data that we used to train them. Otherwise it causes overfitting. So it is a common practice to split the available data into training and testing data.

Now, a classic mistake one can make is doing preprocessing or feature selection before splitting the data. Just like estimators, data transformations like standardization, feature selection should also be learnt from training set and applied to testing set for predictions.

A Pipeline makes it easier to avoid the mistake mentioned above. It ensures that the same samples are used to train the transformers and predictors.

I experimented with different evaluation techniques, the simplest being train_test_split. I also tried K-Fold, Repeated K-Fold, StratifiedKFold, ShuffleSplit and StratifiedShuffleSplit. The Tester script for this project uses StratifiedShuffleSplit.

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"]

In my classifier I pipelined a Scaler, a PCA and LinearSVC. I got a precision score : 0.58. The Tester gave a precision score : 0.34. The precision score tells how well the POI Identifier is able to point at the POIs without being wrong about it.

I got a recall score : 0.35. The Tester gave a recall score : 0.775. The recall score tells how well the POI Identifier is able to point at POIs without missing them.

In case of the POI Identifier, we need the recall score to be better than the precision score, because we do not want to miss any POIs.

I also used accuracy score to evaluate my estimators. It is the fraction of correct predictions. At the end I got an accuracy score : 0.79. The Tester code gave an accuracy score : 0.77.

I also calculated the F1 score which is another measure of accuracy. It is the harmonic average of precision and recall. I got an F1 score : 0.436. The Tester gave an F1 score : 0.47.