

DOCUMENTATION

1. The main goal of this project is to create a person of interest identifier (person of interest is an individual who was indicted, reached a settlement or plea deal with the government, or testifying in exchange for prosecution immunity) based on data made public as a result of the ENRON scandal after collapsed into bankruptcy due to widespread corporate fraud. The dataset of this project is a mix of financial and email features with numerical data along with an extra boolean feature that indicates if a person is a POI(person of interest) or not extracted from an investigation of a report made of findlaw.com. The features in the data fall into three major types, namely financial features, email features and POI labels.

financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (all units are in US dollars)

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

POI label: ['poi'] (boolean, represented as integer)

These features are about to be analyzed and fitted into some algorithms in order to predict other POIs. Some facts about the dataset are the following:

The total number of observations and total Persons of Interest in the dataset: (146, 18)

Missing values for 'salary' feature and percentage to total: (51, 0.3493150684931507)

Missing values for 'bonus' feature and percentage to total: (64, 0.4383561643835616)

Missing values for 'exercised_stock_options' feature and percentage to total: (44, 0.3013698630136986)

Missing values for 'total_payments' feature and percentage to total: (21, 0.14383561643835616)

After plotting some financial data, I saw that there was a max value in 'bonus' and 'salary' features much higher than the rest of the values in the dataset. I append each value of those features into two separate lists and sorted them in descending order. Finally I saw that there was a name called 'TOTAL' in the dataset that represented the sum of all values. This item had been removed as it represented an outlier.

2. The features I end up using in the POI identifier were 'poi', 'bonus', 'exercised_stock_options', 'salary' and 'total_stock_value'. These were extracts of the top 4 best rating features sourced from K-best algorithm plus the 'poi' feature which is needed for the identifier to work properly. The K-best algorithm was fed with all the features of the dataset plus four new features created by me. The rational behind the creation of these new features was to have as many high rating features as possible for the k-best algorithm.

The first two new features were 'rescaled_to_messages' and 'rescaled_from_messages' which derived from implementing MinMaxScaler algorithm to 'to_messages' and 'from_messages' features accordingly, in order to have values from 0 to 1. The other two new features were 'fraction-to-poi' and 'fraction-from-poi' which derived from dividing 'from_this_person_to_poi' with 'from_messages' and 'from_poi_to_this_person' with 'to_messages' features accordingly, in order to find the percentage of emails received from poi and sent to poi for each person.

k_best.scores:

[('exercised_stock_options', 25.097541528735491), ('total_stock_value', 24.467654047526398), ('bonus', 21.060001707536571), ('salary', 18.575703268041785), ('fraction_to_poi', 16.641707070469), ('deferred_income',

11.595547659730601), ('long_term_incentive', 10.072454529369441), ('restricted_stock', 9.3467007910514877), ('total_payments', 8.8667215371077752), ('shared_receipt_with_poi', 8.7464855321290802), ('loan_advances', 7.2427303965360181), ('expenses', 6.2342011405067401), ('from_poi_to_this_person', 5.3449415231473374), ('other', 4.204970858301416), ('fraction_from_poi', 3.2107619169667738), ('from_this_person_to_poi', 2.4265081272428781), ('director_fees', 2.1076559432760908), ('to_messages', 1.6988243485808501), ('rescaled_to_messages', 1.6631007544030192), ('deferral_payments', 0.2170589303395084), ('rescaled_from_messages', 0.169973410804556), ('from_messages', 0.16416449823428736), ('restricted_stock_deferred', 0.06498431172371151)]

The classifier was tested with top 3, 4, 5 and 6 features and it came out with best results when using 4 features. That's why I end up using only 4.

MinMaxScaler algorithm was implemented and tested in all features using a pipeline. But it didn't bring better results than KNeighborsClassifier, thus I chose to keep the latter.

3. The algorithm that I end up using was KNeighborsClassifier as it was the one with the best results. Other algorithms used were SVM, GaussianNB and PCA. The performance was measured by inspecting accuracy score, precision and recall.

Outcome of algorithms used:

Precision and Recall for KNN along with GridSearch:

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='chebyshev', metric_params=None, n_neighbors=4, p=2, weights='distance')

Accuracy: 0.88369 Precision: 0.74498 Recall: 0.37100 F1: 0.49533 F2: 0.4124

Total predictions: 13000 True positives: 742 False positives: 254 False negatives: 1258

True negatives: 10746

Precision and Recall for Pipeline along with GridSearch:

Pipeline(steps=[('reduce_dim', PCA(copy=True, n_components=3, whiten=False)), ('scale_features', MinMaxScaler(copy=True, feature_range=(0, 1))), ('svc', SVC(C=10, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0, kernel='linear', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False))])

Accuracy: 0.85577 Precision: 0.76596 Recall: 0.09000 F1: 0.16107 F2: 0.10929

Total predictions: 13000 True positives: 180 False positives: 55 False negatives: 1820

True negatives: 10945

4. In order for an algorithm to perform well and provide the best possible results, one should consider tuning the parameters of the algorithm. In the KNeighborsClassifier I used GridSearchCV and provide the classifier with a range of possible parameters for the GridSearchCV to find the best combination of parameters that should be used in order to score higher in the final results. In addition I tested various numbers for K value as mentioned in part 2. After using GridCV I used the best estimator as a parameter pack to be implemented to KNeighborsClassifier.
5. Validation is a method that gives an estimate of performance on an independent dataset and serves as check on overfitting. The way to validate a dataset is to split it in training and testing data. Training data is the data that will fit in an algorithm and create a model, and testing is the data that the algorithm will be

tested in order to see how well the algorithm performed.

In order to validate if an algorithm is doing well and has a maximum performance, one should consider evaluation metrics. Evaluation metrics are how you can tell if your machine learning algorithm is getting better and how well you're doing overall. Although accuracy is a perfectly fine metric and a good place to start, it does have some shortcomings. In our case, although there were thousands of people who worked at Enron, there were only a few dozen who ended up really being pursued by the law for their involvement in, in the fraud. So accuracy might not be an ideal metric in that particular case where you have very few examples of one of the classes. Another possibility is that maybe, depending on exactly what you mean by identifying a person of interest, you might want to err on the side of guessing that someone is innocent when you're in doubt. As a result of this, maybe our machine learning algorithm is identifying people who should be thrown in jail or who are guilty and should be claimed as innocent. That's why one should consider other metrics as well. Precision for example is a metric that measures the chances that a person is a POI if the algorithm observes that it is. And recall is a metric that measures the probability of the algorithm to correctly identify a POI provided that the person actually is a POI.

I validated the analysis by implementing GridSearchCV which is an algorithm that splits training and testing data by randomizing folds and choosing the best to split the data in training and testing. The results of the algorithm I used are Precision: 0.74498 and Recall: 0.37100.