Identifying Fraud(POI) using Enron financial and email dataset

Introduction

Enron Corporation was one of the world's major electricity, natural gas, communications and pulp and paper companies with approximately 20,000 staff before its bankruptcy at the end of 2001. Accounting fraud perpetrated by top executives resulted in one of the largest bankruptcies in U.S. History.

In this project, predictive models were built using scikit learn and numpy modules in Python. The target of the classifiers was persons-of-interest (POI's) who were 'individuals who were indicted, reached a settlement, or plea deal with the government, or testified in exchange for prosecution immunity.' Financial data and email statistics from the Enron dataset were used as features for prediction.

Dataset specifications

The motto of this project is to predict POIs' of the dataset. The dataset has 146 records out of which POIs' are 18. 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', '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)

Handling outliers

There were few outliers found in non-numerical values for few financial. Below are the details:

- **TOTAL:** This has been removed as it was just a sum of all respective column values.
- **THE TRAVEL AGENCY IN THE PARK:** This has been removed as it got only two entries for others and Total payments.

NOTE: 'Other' feature is not being used for analysis.
After removing these two records, there were 144 data points left in the dataset.

Additional features creation

The log values of 'total_payments' and 'total_stock_value' and ratio of email messages sepeartely for from and to messages were added to the dataset. For log values, if 'NaN' then '0' and if less than '0' then retain the same value. So only positive values are converted to log values.

Over sampling dataset

As already mentioned, dataset is not huge and the numbers data points which belongs to 'POI' as true are very less compared to NON-POI. This is an imbalanced dataset which is quite often. This issue was resolved using *over sampling* which is add copies of instances from the under-represented class. So, the data points which belongs to POI have been copied under the same name [user name = user name+'dup' in the data_dict dictionary to avoid duplicate key error] with their respective values. This step was taken because while using random sampling as evaluation method, for many trials there were no enough labelled data related to POI which resulted in unclear prediction. The total data points used in the analysis is 162.

Feature Selection

Below are the features used for the final model,

- poi = ['poi']
- email_features = ["from_poi_to_this_person", "from_this_person_to_poi", "shared_receipt_with_poi"]
- financial_features = ["bonus","deferral_payments","deferred_income", "exercised_stock_options", "expenses", "long_term_incentive", "restricted_stock", "restricted_stock_deferred", "salary"]
- new_features = ['ratio_of_from_messages', 'ratio_of_to_messages', 'log_totalPayments', 'log_total_stock_value']

For the final algorithms tried with all the features and different combination of features as well, but as the dataset is small more number of features affected the performance.

SelectKBest and *Principal Components Analysis (PCA)* dimension reduction were then used as part of the *GridSearchCV* pipeline when searching for the best estimator parameter Used above feature selection and dimensionality reduction independently with each classification algorithms. The K-best features were selected using the *f_regression*. For each list of parameters were given and finally the best parameters were fed to the classification algorithm.

Algorithms and evaluation methods

Classification algorithms were tested since this is a classic binary classification task. Algorithms may perform differently using different parameters depending on the structure of the data. Algorithms were tuned using an exhaustive grid search over any major tune-able parameter. The algorithms tried and tested are *Decision Tree, Random Tree, Gaussian Naïve Bayes and KNN*. Amongst all these Random Forest and KNN performed well and Gaussian Naïve Bayes gave the worst performance. All the above

four classifiers were trained and tested under two evaluation methods that are Train_Test_Split CV and k-fold CV.

Validation

Validation is the processed of checking to see how your model performs on unseen data. A classic mistake would be tuning your model can predict your training data very well, but then having it perform poorly on unseen out-of-sample testing data. One of the major goals in validation is to avoid overfitting

Train_Test_Split crossvalidation

In random sampling, each data point is chosen randomly and entirely by chance, such that each data point has the same probability of being chosen at any stage during the sampling process, and each subset of k individuals has the same probability of being chosen for the sample as any other subset of k data points. While using random sampling the best parameters obtained are chosen over different trials. Algorithms may perform differently using different parameters depending on the structure of your data and sample data choose parameters by training the models repeatedly. In the train_test_split function, test_size chosen was 0.3.

Best Parameters obtained using GridSearch	*Decision Tree*	*Random Forest*	*KNN*	*Gaussian*
pca	7	5	4	5
min_samples_split	5	-	-	-
criterion	gini	entropy	-	-
max_features	auto	-	-	-
weights	-	-	distance	-
n_estimators	-	30	-	-
n_neighbors	-	-	7	-

Table 1 Best parameters for classifiers using dimensionality reduction(PCA)

Best Parameters obtained using GridSearch	*Decision Tree*	*Random Forest*	*KNN*	*Gaussian*
Kbest	5	8	15	5
min_samples_split	5	-	-	-
criterion	entropy	entropy	-	-
max_features	auto	-	-	-
weights	-	-	distance	-
n_estimators	-	6	-	-
n_neighbors	-	-	7	-

Table 2 Best parameters for classifiers using Select K best feature selection

K-Fold Cross-Validation

Cross-validation is the process of randomly splitting the data into training and testing data. Then the model can train on the training data, and be validated on the testing data. In the basic approach, called k-fold CV, the training set is split into k smaller sets. The following procedure is followed for each of the k "folds":

- A model is trained using $k - 1$ of the folds as training data;
- the resulting model is validated on the remaining part of the data.

The model selection process was validated using 1000 randomized Stratified ShuffleSplit cross-validator and selecting the parameters which performed best on average over the 1000 splits. This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.

Parameters Provided for GridSearch	*Decision Tree*	*Random Forest*	*KNN*	*Gaussian*
pca	3, 5, 7	4, 5	4, 5, 10	3,5,7,10,15
min_samples_split	3, 5, 10, 15, 25, 40	-	-	-
criterion	'gini', 'entropy'	entropy	-	-
max_features	'auto', 'sqrt'	-	-	-
weights	-	-	'uniform', 'distance'	-
n_estimators	-	10, 20, 30	-	-
n_neighbors	-	-	3, 4, 5, 6, 7	-

Table 3 Best parameters for classifiers using dimensionality reduction(PCA)

Best Parameters obtained using GridSearch	*Decision Tree*	*Random Forest*	*KNN*	*Gaussian*
pca	3, 5, 7	4, 5	4, 5, 10	3,5,7,10,15
min_samples_split	3, 5, 10, 15, 25, 40	-	-	-
criterion	'gini', 'entropy'	entropy	-	-
max_features	'auto', 'sqrt'	-	-	-
weights	-	-	'uniform', 'distance'	-
n_estimators	-	10, 20, 30	-	-

n_neighbors	-	-	3, 4, 5, 6, 7	-

Table 4 Best parameters for classifiers using Select K best feature selection

Classifiers Performance Analysis

The performance of classifiers can be analyzed over multiple attributes which are accuracy, precision and recall. Accuracy is the proximity of measurement results to the true value. Sometimes accuracy can be misleading. Sometimes it may be desirable to select a model with a lower accuracy because it has a greater predictive power on the problem. For example, in a problem where there is a large class imbalance, a model can predict the value of the majority class for all predictions and achieve a high classification accuracy, the problem is that this model is not useful in the problem domain. This is called the Accuracy Paradox. For problems like, this additional measures are required to evaluate a classifier.

Precision

Precision is the number of True Positives divided by the number of True Positives and False Positives. Put another way, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV). Precision can be thought of as a measure of a classifiers exactness. A low precision can also indicate many False Positives.

In this project, if we were using the model to judge whether to investigate someone as a possible person of interest, it would be how often the people we chose to investigate turned out to be persons of interest.

Recall

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity

or the True Positive Rate. Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

For this project, Finding the fraud individual is more important, recall is the most important criteria of the two. We would like to find all people who were involved, even if it means we must investigate and clear more innocent people. We can't afford losing POIs' in this case.

Below are the statistics obtained under to evaluation methods with dimensionality reduction and K best feature selection.

	Decision Tree	*Random Forest*	*KNN*	*Gaussian*
Accuracy (%)	85	89	87	79
Recall	0.75	0.75	0.75	0.25
Precision	0.54	0.66	0.6	0.33
F1	0.63	0.70	0.66	0.28

Table 5 Performance metrics with PCA using train_test_split

	Decision Tree	*Random Forest*	*KNN*	*Gaussian*
Accuracy (%)	89	97	89	83
Recall	0.5	1.0	0.75	0.375
Precision	0.8	0.88	0.67	0.5
F1	0.61	0.94	0.70	0.42

Table 6 Performance metrics with SelectKbest feature selection using train_test_split

	Decision Tree	*Random Forest*	*KNN*	*Gaussian*
Accuracy (%)	86	91	92	76
Recall	0.76	0.88	0.95	0.47
Precision	0.67	0.79	0.78	0.5
F1	0.72	0.83	0.86	0.49
F2	0.74	0.86	0.91	0.48

Table 7 Performance metrics with PCA using K fold CV

	Decision Tree	*Random Forest*	*KNN*	*Gaussian*
Accuracy (%)	87	92.5	92	79
Recall	0.78	0.87	0.95	0.4
Precision	0.71	0.82	0.79	0.59

F1	0.74	0.84	0.86	0.47
F2	0.76	0.86	0.91	0.42

Table 8 Performance metrics with SelectKbest feature selection using K fold CV

Observations and learnings

I have a learnt many classifiers and importance of their respective parameters. One major task during the project was tuning the parameters to obtain required recall, precision and F1 score.

Though the dataset was small, it was a quite a challenge to work with that as it was imbalanced. It's highly difficult to obtain good results on imbalanced dataset even though we try to tune the parameters, so in our case I used over sampling technique to overcome imbalance.

Amongst all the tried classifiers, Random forest took more training time compared to others while Gaussian Naïve Bayes took the less time.

I would like to have also found a more intelligent way to add new features, and prune them back than the univariate K-best selection process. Perhaps removing correlated features, based on correlations with each other as well as all other features in the dataset. I would say that selection of features is a vital task.

The one more important learning during the project was accuracy paradox, though in the beginning I thought accuracy the major factor in gauging the classifier performance but the by the end of it I understood it's not only accuracy but other factors like precision, recall also can be considered.

Reference:

1. http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
2. http://scikit-learn.org/stable/modules/decomposition.html#decompositions
3. http://scikit-learn.org/stable/model_selection.html#model-selection
4. http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing
5. http://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/
6. http://fch808.github.io/Identifying_Fraud_at_Enron.html