Identify Fraud from Enron Email

Final Report

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”]

Answer:

* Goals:
  + - * Examine the dataset. Understand the story behind it.
      * Data Analysis - Use numpy and pandas to explore the dataset
      * Outlier removal - identify and remove outliers
      * Feature selection and scaling
      * Classifier selection
      * Classifier tuning
      * Performance and evaluation metric to identify best algorithm

Machine learning helps model the data and determine new patterns and relationships. In this project, we were able to determine additional features which could point us to people who participated in enron fraud. In essence, scikit-learn was used to utilize several machine learning algorithms to find ‘person of interest’ for the fraud.

The data comprised of 146 records with 14 financial, 6 email and 1 labeled feature(poi). 18 of these records were labeled as ‘poi’.

Two outliers were present in the dataset -

1. ‘TOTAL’ - extreme outlier due to spreadsheet,
2. ‘THE TRAVEL AGENCY IN THE PARK’ - does not represent an individual
3. These outliers were removed by popping out the keys from the dictionary which held them.
4. 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”]

* Features Used:
  + - * features\_list = ['poi','exercised\_stock\_options', 'other', 'expenses', 'emails\_to\_poi\_fraction', 'shared\_receipt\_with\_poi', 'total\_stock\_value' ]
* Feature Selection:
  + - * In order to obtain the most important features from the features\_list, I utilized scikit-learn’s SelectKBest module to get the most influential features.
      * I also used feature\_importances\_ attribute of classifier to see the important features.
* Feature Engineering:
  + - * I create 2 new email based features due to the lack of the same.
      * The 2 added features are:
        + “emails\_from\_poi\_fraction” - fraction of all emails to a person that were sent from a poit
        + “emails\_to\_poi\_fraction” - fraction of all emails to a person that were sent from a poi
        + Both features have numerical value
      * The ideology behind creating these are to have a measure of the strength of communication between people at Enron
* Feature Scaling:
  + - * The range of all features was standardized between (0,1) using MinMaxScaler().
      * By doing so, features like salary and bonus did not dominate or hold undue importance over other features.

1. 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”]

Answer: I used multiple algorithms and evaluated performance metrics for each to reach the optimal algorithm.

Algorithms Tried:

* Naive Bayes
* Nearest Neighbors
* Lienar SVM
* RBF SVM
* Decision Tree
* Random Forest
* Adaboost
* Extra Trees

For each algorithm mentioned above, I calculated:

* + Accuracy
  + Precision
  + Recall
  + ROC curve

After observing the above mentioned metrics for each algorithm, I finally decided to use Decision Tree Classifier.

The results for each algorithm are :

--------------------------------EVALUATING CLASSIFIERS---------------------------------

-> Classifier: Naive Bayes

GaussianNB(priors=None)

precision: 0.373142929293

recall: 0.241118686869

Accuracy: 0.93 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: Nearest Neighbors

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=3, p=2,

weights='uniform')

precision: 0.411849603175

recall: 0.218414069264

Accuracy: 0.86 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: Linear SVM

SVC(C=0.025, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma='auto', kernel='linear',

max\_iter=-1, probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

precision: 0.0

recall: 0.0

Accuracy: 0.88 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: RBF SVM

SVC(C=1, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma=2, kernel='rbf',

max\_iter=-1, probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

precision: 0.151833333333

recall: 0.0338388888889

Accuracy: 0.88 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: Decision Tree

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=5,

max\_features=4, max\_leaf\_nodes=None, min\_impurity\_split=1e-07,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,

splitter='best')

precision: 0.373359665335

recall: 0.352594588745

Accuracy: 0.88 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: Random Forest

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=5, max\_features=4, max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=10, n\_jobs=1, oob\_score=False, random\_state=42,

verbose=0, warm\_start=False)

precision: 0.404909126984

recall: 0.227013636364

Accuracy: 0.90 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: AdaBoost

AdaBoostClassifier(algorithm='SAMME.R', base\_estimator=None,

learning\_rate=1.0, n\_estimators=50, random\_state=None)

precision: 0.457395260295

recall: 0.366957503608

Accuracy: 0.88 (+/- 0.00)

x------------------------------------------------------------------------------x

-> Classifier: Extra Trees

ExtraTreesClassifier(bootstrap=False, class\_weight=None, criterion='gini',

max\_depth=5, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=10, n\_jobs=1, oob\_score=False, random\_state=None,

verbose=0, warm\_start=False)

precision: 0.488873809524

recall: 0.158714321789

Accuracy: 0.88 (+/- 0.00)

x------------------------------------------------------------------------------x

\* Using Precision,recall, accuracy and roc\_curve; Decision Tree Classifier turns out to be the best classifier.

The ROC curve analysis can be seen in the IPython notebook.

1. 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”]

Answer:

All machine learning algorithms take some input parameters through the constructor which can be tweaked in order to obtain optimal performance.

I utilized GridSearch to select the hyperparameter values to obtain the best model.

After running Grid Search, Decision Tree Classifier was chosen to be the best method.

Parameters passed:

parameters = {

'max\_depth': [1,2,3,4,5,6,8,9,10],

'min\_samples\_split':[2,3,4,5],

'min\_samples\_leaf':[1,2,3,4,5,6,7,8],

'criterion':('gini', 'entropy')

}

Grid Search ran all the possible permutations and returned:

Fitting 3 folds for each of 576 candidates, totalling 1728 fits

[Parallel(n\_jobs=-1)]: Done 34 tasks | elapsed: 9.4s

[Parallel(n\_jobs=-1)]: Done 303 tasks | elapsed: 11.6s

Best Score: 0.893

Best parameters set:

criterion: 'gini'

max\_depth: 4

min\_samples\_leaf: 4

min\_samples\_split: 5

Accuracy: 0.928571428571

Precision: 0.75

Recall: 0.6

[Parallel(n\_jobs=-1)]: Done 1713 out of 1728 | elapsed: 15.8s remaining: 0.0s

[Parallel(n\_jobs=-1)]: Done 1728 out of 1728 | elapsed: 15.8s finished

Taking into account the result of grid search, decision tree classifier was made again.

1. 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”]

Answer:

Validation is a critical component of machine learning algorithms. It is performed to ensure that the algorithm generalizes well.

A classic mistake, made too often is over-fitting. Here, the model performs really well on the training data but performs rather poorly on test data. It performs poorly of the cross-validation and test datasets.

Validation was done with the help of evaluate.py which provides custom validation metrics. It took average precision and recall over 1000 randomized trials with data divided into training and test(3:1).

1. 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”]

**2 Classifiers and their performance metrics:**

* Final Decision Tree Classifier:
  + - * Precision: 0.56967
      * Recall: 0.39250
      * Accuracy: 0.9285
      * AUC: 0.98
* Extra trees classifier
  + - * Precision: 0.4888
      * Recall: 0.1587
      * Accuracy: 0.88
      * AUC: 0.88
* Interpreting Metrics

The main evaluation metrics used were ROC curve, precision and recall.

**Precision:** is the fraction of the true positive over the sum of true positives and false positives.

**Recall:** is the fraction of the true positives of over the sum of true positives and false negatives i.e. the fraction of the truly positive instances that the classifier recognizes.

**ROC Curve:**

- Has better statistical foundations than most other measures.

- Visualizes a classifiers performance. Unlike accuracy, ROC curve is insensitive to data sets with unbalanced class proportions.

- Unlike precision and recall, ROC curve illustrates the classifiers performance for all values of the discrimination threshold.

- ROC curve plots the classifiers recall against its fall-out/false positive rate.

- Each point on ROC represents different tradeoff(cost ratio) between false positives and false negatives.

**- ROC curve metric is slowly become ore popular in Machine Learning**