Machine Learning Final Project

September 2016

Manouchehr Bagheri

Enron Submission Free-Response 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"]

In this project we were interested in finding a Machine Learning method to identify Persons Of Interest in Enron fraud case. The dataset provided for this project contains features in three major types: Financial, email, and POI label. A statistical overview of the data is as follows (more details can be found in poi-id.py):

- total number of data points after removing outliers: 144
- total number of data points after data cleanups: 122

- allocation across classes (POI/non-POI): 18 POI and 104 Non-POI
- number of features in original feature_list with new feature: 18
- Features with many missing values:
    - restricted_stock_deferred : 128
    - deferral_payments : 107
    - deferred_income : 97
    - long_term_incentive : 80
    - bonus : 64
    - from_this_person_to_poi : 60
    - from_messages : 59
    - shared_receipt_with_poi : 59
    - to_messages : 59
    - from_poi_to_this_person : 58
    - expenses : 51
    - salary : 51
    - exercised_stock_options : 44
    - restricted_stock : 36
    - total_payments : 21
    - total_stock_value : 20

For getting realistic results from analysis, I first identified outliers in the dataset with ‘salary value’ > 1 million dollars and ‘bonus’ > 5 million dollars. The Outliers were: ['LAY KENNETH L', 'SKILLING JEFFREY K', 'TOTAL'] since the first two are POIs, I decided to keep them and only remove the 'TOTAL' from dataset. I also removed 'THE TRAVEL AGENCY IN THE PARK' manually as it was not represent a person. Furthermore, I removed entries with 'ave_earnings'=0 as they have no financial information in the dataset (There were 22 employees with this specification and all of them were NON-POI).

I removed entries with all '0' by utilizing the featurFormat parameters in order to clean up the data and avoid any problem on score calculations. Especially when these data entries would not have any impact on finding POI.

data = featureFormat(my_dataset, features_list, sort_keys = True, remove_NaN=True, remove_all_zeroes=True, remove_any_zeroes=False)

> Convert my_dataset to numpy array of features
> ***remove_NaN = True*** will convert "NaN" string to 0.0
> ***remove_all_zeroes = True*** will omit any data points for which
> all the features you seek are 0.0
> ***remove_any_zeroes = False*** will not omit any data points for which
> any of the features you seek are 0.0
> ***sort_keys = True*** sorts keys by alphabetical order. Setting the value as
> a string opens the corresponding pickle file with a preset key
> order (this is used for Python 3 compatibility, and sort_keys
> should be left as False for the course mini-projects).
> ***NOTE:*** first feature is assumed to be 'poi' and is not checked for
> removal for zero or missing values.

In order to verify the accuracy of the identifier, I tried several classifiers to transform, fit and check the output with real results of the POI labels during precision calculation of each classifier by the tester.py script.

Then I tried three different approaches to create classifiers which are specified in poi-id script. Following 5 classifiers are used in each one of the approaches:

1. Logistic Regression
2. LinearSVC
3. Decision Trees
4. Random Forest
5. Gaussian NB

The first approach was using all five classifiers on the original feature_list including the new feature 'ave_earnings'.

The second approach was making further modifications on the features_list by applying MinMaxScaler and SelectKBest based on Chi-squared scoring function to choose best features among them.

The third approach was using GridSearchCV method with stratified shuffle split cross validation to get the best parameter sets for each classifier after applying it in a pipeline using pipeline of three steps: scaling, SelectKBest, and different classifiers like NaiveBayes.

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 importance 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", "properly scale

Since email features did not returned a precise identifier for POI during course materials, I decided to focus more on financial features and engineered a new feature that can present the effective factor of other financial features from one hand and also plays better factor as an identifier from other hand. This new feature shows average value of total earnings and called "ave_earnings" which is mean value of 'salary', 'bonus', 'deferral_payments', and 'total_payments' selected from financial features set for each person.

By looking at the same results for both cases of using or not using the new feature, 'ave_earnings', it is obvious that the new feature has significant effect on the final algorithm performance (more details can be fund by running poi-id-with-new-feature.py and poi-id-without-new-feature.py).

Three approaches of GaussianNB algorithm With new feature			
Accuracy: 0.71662	Precision: 0.23555	Recall: 0.37500 F1: 0.28935	F2: 0.33530
Accuracy: 0.83046	Precision: 0.40520	Recall: 0.21800 F1: 0.28349	F2: 0.24019
Accuracy: 0.86162	Precision: 0.59263	Recall: 0.32150 F1: 0.41686	F2: 0.35388

Three approaches of GaussianNB algorithm Without new feature			
Accuracy: 0.71667	Precision: 0.23641	Recall: 0.50450 F1: 0.32195	F2: 0.41123
Accuracy: 0.83800	Precision: 0.33385	Recall: 0.21600 F1: 0.26230	F2: 0.23241
Accuracy: 0.87140	Precision: 0.53278	Recall: 0.28850 F1: 0.37431	F2: 0.31763

I applied MinMaxScaler and SelectKBest method based on Chi-squared scoring function. I tried several different combinations of k values and train_test_split parameters on Logistic Regression classifier and ended up using following ten best features with k=10:

This combination was returning best accuracy scores and confusion matrix:

```
labels_pred [ 0.  1.  0.  1.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
model acc: 0.692307692308, F-score: 0.0

 confusion matrix:
[[9 3]
 [1 0]]
logistic regression coefficients:
[['salary' '-4.20025284087e-06']
 ['bonus' '5.58531197946e-07']
 ['total_payments' '4.70830214289e-08']
 ['ave_earnings' '-1.94944325581e-07']
 ['total_stock_value' '1.05219278523e-07']
 ['exercised_stock_options' '1.26656710028e-10']
 ['expenses' '-8.9973816001e-06']
 ['long_term_incentive' '-3.40624414906e-07']
 ['shared_receipt_with_poi' '-0.000275018070908']
 ['from_poi_to_this_person' '-4.98213546238e-05']]
```

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 applying first round of scaling and feature selection with SelectKBest (chi2, 10) and applying it to pipeline with different classifiers, I ended up using Pipeline method with GaussianNB Classifier with precision and recall scores above .3 as follows

```
Pipeline(steps=[('SKB', SelectKBest(k=6, score_func=<function f_classif at 0x13B10630>)), ('PCA',
PCA(copy=True, n_components=2, whiten=True)), ('CLF', GaussianNB())])
    Accuracy: 0.86162    Precision: 0.59263    Recall: 0.32150 F1: 0.41686    F2: 0.35388
    Total predictions: 13000    True positives:  643    False positives:  442   False negatives: 1357   True
negatives: 10558
```

I also tried Logistic Regression, LinearSVC, Decision Trees, and Random Forest classifiers with this model and got better results than previous methods, but none on them returned precision and recall scores above .3 except for Decision Tree Classifier with following result:

```
Pipeline(steps=[('SKB', SelectKBest(k=4, score_func=<function f_classif at 0x13B10630>)), ('PCA', PCA(copy=True,
n_components=3, whiten=False)), ('CLF', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
        max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
        min_samples_split=2, min_weight_fraction_leaf=0.0,
        presort=False, random_state=None, splitter='best'))])
    Accuracy: 0.81938    Precision: 0.41077    Recall: 0.40050 F1: 0.40557    F2: 0.40251
    Total predictions: 13000    True positives:  801    False positives: 1149   False negatives: 1199   True negatives: 9851
```

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? (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 item: "tune the algorithm"]

   Tuning an algorithm or machine learning technique, is a process of optimizing the parameters that impact the model in order to enable the algorithm to perform the best. It of course depends on how and what is defined as "best" for an analysis. In our case "best" is being measured by precision and recall value to be above .3 for the algorithm. If we don't do this well, we not only cannot fulfill the criteria, but also cannot get the best result of the algorithm.

   I used GaussianNB algorithm that do not have parameters that I need to tune, but I have tried several tests with SVM algorithm for which I faced with several issues. In many cases the tester.py was not returning any precision result due to issues indicated as: *"Precision or recall may be undefined due to a lack of true positive predictions."* Or *"Got a divide by zero when trying out:"*
   In order to resolve this issue, I used the data cleaning described on the answer of question 1 and removing entries with zero values on their financial data.
   For the SVC algorithm changing kernel, C and gamma parameters can be considered as parameter tuning.
   So the high C value (1000 in this analysis) performed proper classifying of all training examples by giving the model freedom to select more samples as support vectors. At the other hand lower gamma value (.001 in this analysis) provided proper regularization with C and prevented overfitting. Different values for k or kernel were not making much difference on the result.

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 item: "validation strategy"]

   The classic mistake someone can make is fitting the data on the entire dataset as opposed to splitting it into training and testing data. To avoid that mistake I used test size other than 0 on train_test_split function.

   Another mistake that may happen is missing the utilization of the feature_list changes to dataset by featurFormat and extracting features and labels for next steps of the analysis. For example I made changes to the feature_list three times until I reached to 10 best

features and after that I made another set of SelectKBest() in the pipeline. At each step I applied separate targetFeatureSplit in order to use proper data for each analysis.

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"]

   The evaluation metrics I use in this analysis is Precision and Recall. Precision is the fraction of retrieved instances that are relevant. Recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. High precision means that an algorithm returned substantially more relevant results than irrelevant, while high recall means that an algorithm returned most of the relevant results. I am trying to get scores higher than 0.3 for both of them.

   In a simpler words, the higher precision means better and more precise recognition of the POIs, which means among all persons recognized as POI (or predicted as positive) by our model, we had better prediction as the number of true_positives is higher. Because if precision is closer to 1 it means higher true_positives and lower false_positive. But it does not guaranty that our cluster of predicted POIs contains all POIs in the dataset. We might have been just lucky to get a cluster with all members being POI.
   At the other hand, recall value means how big is our examined cluster and what ratio of total number of POIs are in that cluster. The higher value for recall means the higher number of the POIs are recognized by the model. So we try to have both performance to be high in order to not only get the higher number of POIs but also we recognize them precisely with lower mistakes.

   I did not use Accuracy as an evaluation metrics, because using accuracy is only good for symmetric data sets where the class distribution is 50/50 and the cost of false positives and false negatives are roughly the same. We should not rely on it too much because our data sets are far from symmetric.

Github Repository:

Manonuro/Machine-Learning-Project/final_project/:

https://github.com/Manonuro/Machine-Learning-Project/tree/master/final_project

References:

Scikit Learn (Naive Bayes):

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes

Support Vector Machines:

http://scikit-learn.org/stable/modules/svm.html

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Sklearn neighbors (KNeighborsClassifie):

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.neighbors

Generalized Linear Models:

http://scikit-learn.org/stable/modules/linear_model.html

Visualizing K-Means Clustering:

http://www.naftaliharris.com/blog/visualizing-k-means-clustering

Clustering:

http://scikit-learn.org/stable/modules/clustering.html

Preprocessing data:

http://scikit-learn.org/stable/modules/preprocessing.html

Cross-validation: evaluating estimator performance:

http://scikit-learn.org/stable/modules/cross_validation.html

sklearn.grid_search.GridSearchCV:

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.grid_search

sklearn.metrics.accuracy_score:

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Pipeline and FeatureUnion: combining estimators:

http://scikit-learn.org/stable/modules/pipeline.html

sklearn.cross_validation.StratifiedShuffleSplit:

http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html

sklearn.feature_selection.SelectKBest:

http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest