Machine Learning Final Project

July 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. 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.

First I identified three outliers in the dataset with salary value > 1 million and bonus > 5 million dollars: ['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 replaced all 'NaN' values with '0' in order to clean up data and avoid any problem on future calculations.

Since email features did not returned a precise identifier for POI during course material, I decided to focus more on financial features and make 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 earning called “ave_earnings” which is mean value of 'salary', 'bonus', 'deferral_payments', and 'total_payments' selected from financial features set for each person. Then I tried three different approaches to create classifier which are specified in ML_project_varity_of_classifiers.ipnyb script. The first approach was using different classifiers like GaussianNB and SVC. The second approach was using pipeline of three steps: scaling, SelectKBest, and different classifiers like NaiveBayes. The last approach was using PCA in order to find the best combination of the features that can provide us the identifier with highest precision.

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

Following three approaches were taken in my analysis:

a) I was first wanted to see how the selected number of features in Features List = ['poi', 'salary', 'bonus', 'total_payments'] will perform with GaussianNB classifier. It was returning precision above .33 without any processes. Same process with SVC didn’t’ pass the tester.py, but by manual accuracy calculation it returned accuracy of 0.79 and precision of .62, But since there are many ‘NaN’ values for features and not all features considered, I was wondering if we can get better identifier by selecting 18 relative features including new feature “ave_earnings” and replacing “NaN” to 0 for numerical features. Surprisingly the result was not better for GaussianNB classifier (Precision: 0.28768), but for SVC we got better manual results (accuracy: 0.89, Precision: 0.79). The tester.py didn’t processed the SVC classifiers as before.

b) By putting all new and selected features in one pool for SelectKBest process, I tried different combinations of the number of best features and classifiers. For that purpose I used pipeline of three steps: scaling, SelectKBest, and different classifiers. The starter pipeline with steps: scaler, SelectKBest, and GaussianNB (), we got the best Precision of 0.43492 for the K=5 after testing K for values 3, 4, 5, and 6.
Best result for Pipeline method with Scaler, SelectKBest, and SVC () is Precision: 0.84444

c) The last approach was using PCA in order to find the best combination of the features that can provide us the identifier with highest precision. I used pipeline to transfer the features list to PCA in order to select most efficient features and then apply them in different classifiers like GaussianNB(), and SVC.
The Best result is for Pipeline with PCA (n_components=2) and GaussianNB() is Precision: 0.61854
Pipeline with PCA and SVC, returns no precision result due to following issue: Precision or recall may be undefined due to a lack of true positive predictions.

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”]

   I ended up using Pipeline method with Scaler, SelectKBest with k=2, and SVC (kernel='linear', C=1000, gamma=.001), which returned the Precision: 0.63060

   I tried GaussianNB, SVC classifiers, pipelines with scaling, SelectKBest, and different classifiers, and also pipeline with PCA and different classifiers as described on previous answer.

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”]

   In many cases the precision result was low and for some classifiers like SVC, 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.*
   I also got the precision=1.000 for pipeline with scaler, SelectKBest, and MultinomialNB classifier when k=3, 4, or 5 which was too good to be true.
   In order to tune the parameters, I used GridSearchCV method that I described more on the answer of question 6.

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”]

   For above example when the result for multiple variables are the same, in this case the K value, and it returns highest precision it would not look a valid result. So I tried other combination of steps and different variables.

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”]

   On the Pipeline method with Scaler, SelectKBest, and SVC () method, I made changes on k, C, gamma, and kernel values. I ran GridSearchCV for different combinations of those values and found the best combination with highest score for development set as follows:
   {'svc__gamma': 0.001, 'svc__kernel': 'linear', 'svc__C': 1000, 'skb__k': 2}
   The output of the tester.py for precision was 0.63060.

The most important parameters were C and gamma on the SVC. For lower C values (below 100) the tester was returning warning message: *Precision or recall may be undefined due to a lack of true positive predictions*
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.

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_s
election.SelectKBest