Intro To Machine Learning Project Report

The Enron Corporation was an American energy company based in Houston, Texas. It employed about 20, 000 people and was one of the world's major electricity, gas and communications companies. Enron is now more famous for what is being called the Enron scandal of 2001 where it was revealed the company commited one of the biggest known cases of corporate fraud. Due to it's high profile nature, otherwise confidential records have been made public which allows us to analyze the data using machine learning.

Questions

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"]

The goal of this project is to write a machine learning algorithm that identifies person's of interest using the available information we have. Supervised Machine learning is especially suited to this task as it allows training our model to detect person's of interest by looking for patterns present in the cases of person's of interest already identified. The dataset includes information for 146 employees with 21 features, 14 financial features, 6 email features and 1 denoting if the person is considered a person of interest or not. A number of outliers were found in the dataset, most notable is the entry for Totals , this obviously is an outlier and would heavily skew the data. Also not every feature for every employee has filled in values, and since these ("NaN") values are eventually being represented with zero, I decided it was best exclude entries with too many "NaN" values. How many is too many was a worry since removing too many would heavily reduce the dataset and not removing enough would leave it with too many outliers. Out of 20 features, I decided any entries with more than 15 is too much. below is a list of entries with more than 15 NaN feaures. This number also maximized the precision, accuracy and recall of the algorithm.

LOWRY CHARLES P CHAN RONNIE WODRASKA JOHN URQUHART JOHN A WHALEY DAVID A MENDELSOHN JOHN CLINE KENNETH W WAKEHAM JOHN WROBEL BRUCE MEYER JEROME J GATHMANN WILLIAM D GILLIS JOHN LOCKHART EUGENE E PEREIRA PAULO V. FERRAZ BLAKE JR. NORMAN P THE TRAVEL AGENCY IN THE PARK

TOTAL CHRISTODOULOU DIOMEDES WINOKUR JR. HERBERT S YEAP SOON FUGH JOHN L SCRIMSHAW MATTHEW SAVAGE FRANK GRAMM WENDY L

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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

I initially had all the features present including the two new features created, the first is a feature measuring person of interest interaction, This measures the ratio of messages sent to and received from person's of interest divided by messages sent and received from everybody else. This is a good metric since it operates on the assumption that person's of interest will interact at a much higher rate with other person's of interest relative to their overall interactions with non poi's. and secondly is a "bonusSalary" feature that adds together both, the reason behind is that while there is a totalpayments feature, it seems to be measuring other things in addition. the new bonusSalary feature measures a kind of yearly funds paid out to the employees.

Yes. The MinMaxScaler was applied, Scaling is very appropriate to this dataset since it includes lots of disproportionate features that can be scaled to between 0 and 1. An example would be the messages features like to_messages and from_messages which at the most would be in the hundreds compared to other message features like from & to_poi which would only be a fraction of that and then compared to the financial features total payments which would be in the thousands. Not scaling these features would result in having the some financial features heavily dominating over other financial features and definitely the message features.

PCA was deployed after scaling and the number of components that maximized all accuracy score, precision recall scores out of the ones attempted is 15. PCA's components_ atribute provides the weight of each original feature towards each principal component.

For example:

```
for x in range(len(pca.components_[0])):
    print "Feature", x+1, pca.components_[0][x], my_features_list[x+1]
```

Feature 1 0.00391656756556 salary

Feature 2 -5.09487062883e-06 to_messages

Feature 3 0.0252702207138 deferral_payments

Feature 4 0.0613433847599 total_payments

Feature 5 0.651084531984 exercised_stock_options

Feature 6 0.0188199325712 bonus

Feature 7 0.0959047647805 restricted_stock

Feature 8 7.01133434292e-07 shared_receipt_with_poi

Feature 9 -0.00853779925267 restricted_stock_deferred

Feature 10 0.747412673771 total_stock_value

Feature 11 0.000842259670724 expenses

Feature 12 0.00793032843274 loan_advances

Feature 13 -2.23126715979e-06 from_messages

Feature 14 0.0411033182358 other

Feature 15 -1.85758488662e-07 from_this_person_to_poi

Feature 16 -0.000210753361855 director_fees

Feature 17 -0.0391246521614 deferred_income

Feature 18 0.0117311536064 long_term_incentive

Feature 19 8.64624630599e-07 from_poi_to_this_person

Feature 20 -2.77668440137e-10 messageratio

Feature 21 0.0 bonussalary

This provides the weight of each original feature to the first principal component in order, the last feature in the list corresponds to the last feature added which was the bonussalary feature I created. Also the feature contributing the most to the first pca is feature 10 which corresponds to the total stock value feature.

Feature Selection

```
print pca.explained_variance_ratio_
```

```
[    7.42188132e-01      1.55054359e-01      4.98348977e-02      2.04359341e-02
     1.53369088e-02      8.75579903e-03      5.64495207e-03      2.34763887e-03
     2.71516493e-04      7.21985694e-05      4.42909480e-05      1.10195880e-05
     2.12737891e-06      1.68540141e-07      5.24856675e-08]
```

Feature selection was also deployed using SelectKBest with k = 3, 3 was chosen since it maximizes all three metric scores. The explained_variance_ratio attribute shows how much proportion of variance explained by each of the components in order of strength.

A pipeline is used to go from one stage to the other, First a MinMaxScaler is applied because the next

stage is uses PCA to reduce the number of components to 15, PCA is great for getting the latent features in a dataset and the explained_variance_ratio gives me all 15 features with how much variation they account for. Seeing this, I decided to use SelectKBest to select the first three since they represent the stongest features.

Preprocessing.Normalize and StandardScaler were attempted with both producing much lower scores

Normalizer

Accuracy: 0.79554 Precision: 0.07603 Recall: 0.02950

StandardScaler

Accuracy: 0.81854 Precision: 0.35395 Recall: 0.21750

What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]

```
Using adaboost = AdaBoostClassifier(n_estimators=70)
```

Accuracy: 0.80138 Precision: 0.30231 Recall: 0.22250 F1: 0.25634

```
Using svc = SVC(C=50, kernel='rbf', gamma=2.0)
```

Accuracy: 0.83423 Precision: 0.36332 Recall: 0.10300 F1: 0.16050

I tried a few of the ensemble classifiers as well as the support vector machine, the Gaussian Naive-Bayes however always produced the best results. One possible reason for this is possibly not very large training points made even smaller by my huge reduction of it in the outlier section. Naive bayes is considered a better algorithm when dealing with small training points.

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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Improperly tuned algorithms can result in over/underfitting. Overfitting can result in a

situation where the due to overfitting, the algorithm does well with the training data but can not generalize to new data, likewise underfitting is related to when the algorithm does not fit the data well and thus is biased against the data. Fine-tuning an algorithm can help balance these two by optimizing the algorithm relative to the particular dataset and the questions we're trying to answer.

I tried using the decisiontreeclassifier and attempted using the gridsearch to find the best parameters. The idea was to try both the gini and entropy options for the criterion parameter mixed in with the an increasing min_samples_split from 1 to 100. The minimum samples split specifies the minimum value the tree shouldn't split after. This helps to reduce overfittting. The best tuning found using the best_estimator attribute is

{'min_samples_split': 100, 'criterion': 'gini'}

which gives a score of

Accuracy: 0.82685 Precision: 0.31517 Recall: 0.10700

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"]

Validation is the means of checking that an algorithm works well, the idea behind validation is to assess how an algorithm generalizes to new data since that tells us how well a machine learning algorithm will perform in practice. It is a bad idea to train on an entire dataset since that model will overfit to the data, any predictions made on the same data will produce high accuracy scores and will result in the model not generalizing well to new data.

I used the cross validation train_test split to partition the data in half before using some of the classifiers, a concern with this type of splitting is that using too small a portion for training could also result in overfitting, I selected 50% as the test set size as it gave the highest accuracy score. It should however be noted that the train test split produced much lower accuracy, precision and recall when compared to the scores produced by the stratifiedshufflesplit that comes with the helpers.py code, the method of splitting works by splitting the dataset into k folds, using one fold as a test set and the remaining as a training set and then repeating the process selecting a different fold and then averaging the result, this is good since it uses all the data as both a test and training set.

Below are some scores for both the train test split and stratifiedshuffle using the same classifiers.

Gaussian Naive Bayes

train_test_split

Accuracy: 0.786885245902 Precision: 0.2 Recall: 0.285714285714

```
StratifiedShuffleSplit
```

Accuracy: 0.82777 Precision: 0.42664 Recall: 0.34750

AdaBoost Classifier

train_test_split

Accuracy: 0.770491803279 Precision: 0.25 Recall: 0.2

`StratifiedShuffleSplit`

Accuracy: 0.80177 Precision: 0.30226 Recall: 0.22050

The stratified shuffle split consistently produces better scores for all three scores, the downside is that it takes much longer.

While accuracy is a good way of checking whether an algorithm makes good predictions, it is not a good metric for checking when the classification isn't even for example, cancer detection(most people don't have cancer) or Person of interest detection (Most people are not poi's). As demonstrated in the class, an algorithm that for examples predicts everyone as non-poi will have a very high accuracy since most people are generally not person's of interest, However, this by no means indicate that the algorithm is performing well since all it(accuracy) measures is the quotient of how many it predicted correctly over how many overall. As a result, it's possible to have a high accuracy and an underperforming algorithm especially when there's over-fitting happening.

The analysis is validated using Accuracy, Precision and Recall. There is also the F1 score which is an average of both Precision and Recall. The supervised classifier chosen as well as the number of components selected for pca and number of features chosen on selectKbest were all chosen because they maximized these three metrics.

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"]

Metrics are recall and precision, Using a decision tree classifier gives: accuracy: 0.758432432432 precision: 0.208126673327 recall: 0.205380050505

and using the Naive-Bayes classifier:Accuracy: 0.82777 Precision: 0.42664 Recall: 0.34750 F1: 0.38303

Recall is the probability of an algorithm to predict an event correctly, it's the ratio of the number of times an event is predicted correctly to the number of times the event occurs. Precision however, is the measure of how correct a prediction that has already been made is. So while recall measures the fraction of relevant instances of a result, precision measures how relevant those instances are.

The above results for the decision tree classifier show a 20.5% recall, this means that the algorithm correctly identifies person's of interest 20.5% of the time while the Naive-Bayes classifier is able to do so 34.7% of the time. The precision on the other hand for the Decision Tree is 20.8% indicates that given that the algorithm has

labeled someone as a person of interest, the probably of the person really being a person of interest which in this case the Naive-Bayes does better in as it has a 42.6% precision.