ENRON_ML_PROJECT

ENRON CASE - PERSON OF INTEREST CLASSIFIER - INTRO TO MACHINE LEARNING FINAL PROJECT

Introduction

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. In this project, we will play detective, and build a person of interest classifier based on financial and email data made public as a result of the Enron scandal.

Data Exploration

General Observations

Our dataset is a dictionary containing financial and email data about 146 persons. For each person, we have 12 financial features including: - 9 payment features describing payments those people got from Enron before the collapse. - 3 stock value features describing different stock value owned by those people before the collapse.

As well, the dataset provides email data including: - number of emails sent by a person and number of emails sent to a person of interest. - number of emails received by a person and number of emails received from a person of interest.

Among the 146 persons included into the dataset, 18 of them are labelled as person of interest.

As the number of observations is low and the number of poi is only a small number of the total number of observations, **we will need to use a strong cross validation method like Stratified Shuffle Split.** This will ensure that we use efficient training and testing samples. Also, because the data is unbalanced, we will use precision and recall as evaluation metrics.

Features Exploration

We noticed two interesting things: - some data are either missing or non-existent and carry the value 'NaN'. - two columns 'Total Payments' and 'Total Stock Value' are the sum of other columns.

Dealing with missing or non-existent data

For each feature, we will look at the distribution of missing data and evaluate their potential impact on our future algorithm.

SALARY

```
poi    salary
False  Value     78
       NoValue    50
True   Value     17
       NoValue     1
```

Almost all persons of interest (poi) has an attributed salary whereas only 60% of non-poi has an attributed salary. We can easily make the assumption that missing data are actually missing because most of the employee in our dataset should have a salary. Replacing 'NaN' with 0 would completely biased our algorithm.

We won't include salary into our list of selected features.

BONUS

```
poi    bonus
False  Value     66
       NoValue    62
True   Value     16
       NoValue     2
```

Almost all poi has a bonus informed in the dataset while only 50ish% of non-poi has an informed bonus. It makes sense that bonus is something only granted to specific individuals and the trend could be useful for our algorithm.

We will replace NaN values by 0 and pre-select bonus as a potential feature.

LONG TERM INCENTIVE

```
poi    long term incentive
False  Value     74
       NoValue    54
True   Value     12
       NoValue     6
```

Long term incentive is quite evenly distributed between poi and non-poi. As it makes sense that some individuals don't have long term incentives, we can make the assumption that missing data are zeros.

We will replace NaN values by 0 and pre-select long term incentive as a potential feature.

DEFERRAL PAYMENTS AND DEFERRED INCOME

```
poi    deferral payments
False  Value     34
       NoValue    94
True   Value      5
       NoValue    13

poi    deferred income
False  Value     38
       NoValue    90
True   Value     11
       NoValue     7
```

As Long term incentive, we can make the assumption that missing data are zeros. It seems safe to infer that most of the individuals were not concerned with such things. The large ratio of poi concerned by deferred income is also interesting and could be useful for our algorithm.

We will replace NaN values by 0 and pre-select deferral payments and deferred income as potential features.

DIRECTOR FEES

```
poi    director fees
False  Value     17
       NoValue   111
True   Value      0
       NoValue    18
```

Director fees is an interesting feature. 0 poi are directors as individuals involved in the fraud were people managing the company on the day to day. It should be a powerful feature to help our algorithm exclude directors as potential poi.

We will replace NaN values by 0 and pre-select director fees as a potential feature.

LOAN ADVANCES

```
poi    loan advances
False  Value      3
       NoValue   125
True   Value      1
       NoValue    17
```

Only a few employees got loan advances and the only poi who got one (Kenneth Lay) got a 81M loan. As it seems to be a exceptional event, it won't help our algorithm identify poi.

We won't include loan advances into our list of selected features.

EXPENSES

```
poi    loan advances
False  Value     77
       NoValue    51
True   Value     18
       NoValue     0
```

All poi got some expenses! It seems to be a powerful features we want to test for our algorithm.

We will replace NaN values by 0 and pre-select expenses as a potential feature.

STOCK OPTIONS FEATURES

```
poi    exerciced stock options
False  Value     90
       NoValue    38
True   Value     12
       NoValue     6

poi    restricted stock
False  Value     90
       NoValue    38
True   Value     12
```

```
       NoValue     6

poi    restricted stock deferred
False  Value     90
       NoValue    38
True   Value     12
       NoValue     6
```

All stock values features will be pre-selected as features as they might be good clue to identify poi.

EMAILS AND SHARED RECEIPTS

```
poi             from messages  to messages
False  Value     72             72
       NoValue    50             50
True   Value     14             14
       NoValue     4              4

poi             from this person to poi  from poi to this person
False  Value     72                       72
       NoValue    50                       50
True   Value     14                       14
       NoValue     4                        4

poi             shared receipt with poi
False  Value     72
       NoValue    50
True   Value     14
       NoValue     4
```

By looking at the distribution of emails and shared receipts features, we noticed that 60 individuals in our dataset don't have any of those data informed. We built the following script which returns 60 to confirm that hypothesis. `python count_email_data_missing = 0 features_to_check = ['shared_receipt_with_poi','to_messages','from_messages','from_poi_to_this_person','from_this_person_to_poi'] for key, value in data_dict.iteritems():`

```
    count_per_key = 0
    for features, values in value.iteritems():
        if features in features_to_check:
            if values == 'NaN':
                count_per_key += 1
    if count_per_key == 5:
        count_email_data_missing += 1
print count_email_data_missing
```

Obviously, we can't assume those people never received or sent any emails, so those data are missing.

This discovery is important because these features are interesting but we will need to remove 60 individuals from our dataset to use them properly.

Conclusion of missing values exploration As a conclusion of our missing values exploration, we can say that we already exclude some features we think won't help us to identify poi. Also, we acknowledged a major aspect of our dataset by noticing that 60 of our 146 individuals don't have any emails data.

Outliers exploration

'TOTAL' OBSERVATION

When plotting exploring and plotting data, we noticed that an observation was always way higher than the other observations. `python`

```
for key, value in data_dict.iteritems():
    if value['expenses'] > 5000000 and value['expenses'] != 'NaN':
        print key
        print value
```

By using the simple script we found out that the TOTAL observation was part of our dataset. We removed this observation from our dataset.

USING TOTAL PAYMENTS AND TOTAL STOCK VALUES TO IDENTIFY DISCREPANCIES

By using two existing features of our dataset, total payments and total stock values, we were able to identify potential other discrepancies in our dataset. We compared these two features to the sum of the other features they relate to.

We identified the two following issues:

- The amount of deferral payments attributed to BELFER ROBERT has been wrongly entered as it should be a positive number and should be equal to the total payments.
- The amount of restricted stock deferred attributed to BELFER ROBERT has been

wrongly entered as it should be a negative number and should be equal to the total stocks.

These two items have been corrected.

CHECK OF NAMES IN THE DATASET

We will end up our outlier investigation by looking at the list of names of individuals to ensure we did not miss something.

By doing this we identified the following observation 'THE TRAVEL AGENCY IN THE PARK' which can be considered as an individual and so can't be considered as a potential person of interest.

We removed this observation from the dataset.

Features Creation

During our feature exploration, we noticed 4 interesting features tracking the volume of emails an individual would exchange with others and with persons of interests. But the number of email can be influenced by the position and the activities of an individual. However, we could test two new features which would be less biased by the position or normal activities of an individual. By creating a ratio of email sent to or received from persons of interest, we could track the relative importance of conversations with persons of interest among all conversations of an individual.

We created the two following new features: - ratio_of_emails_sent_to_poi - ratio_of_emails_received_from_poi

These two features will be tested later on.

Data Preparation

We will create two different dictionaries to test two different set of features. As we noticed before, email data are not complete for every individuals. We'll then test: - financial features with the relevant observations - financial + email features with the relevant observations

The two dictionaries are called: - data_financial - data_email_financial

Classifier testing

Validation phase

We tested the following combinations of data sets and features: - dataset: data_financial, features: financial features selected during our features exploration. - Only 2 outliers have been removed from this dataset. - we applied some automatic features selection method. - dataset: data_email_financial, features: financial features selected during our features exploration + email features. - 2 outliers and 60 observations have been removed from this dataset. This dataset might be more prone to overfitting. - we applied some automatic features selection method.

In order to test how well our models have been trained and to estimate model properties, we used GridSearchCV to validate them. Because our data set is really small, we decided to apply a Stratified Shuffle Split which ensure the ratio of POI and non-POI is the same during training and testing. To maximize this, we decided to use a 1000 folds stratified shuffle split validation.

We applied feature scaling when the scale of a feature was irrelevant or misleading. Because Kneighbors considers Euclidean distance to be meaningful, If a feature has a big scale compared to another, but the first feature truly represents greater diversity, then clustering in that dimension should be penalized. For these reasons we normalized the features before applying Kneighbors model using the MinMaxScaler algorithm which normalize each feature on 0 to 1 scale.

We applied univariate feature selection to identify the useful features and exclude the unnecessary ones (for both computing time and overfitting consideration). Using a pipeline, we applied SelectKBest which selects the k best features for the model and we set the range to [3,5,10] so that the performance of the model is tested with 3 best features, 5 best features and 10 best features.

Our goal is also to set parameters to optimal values that enable us to complete this learning task in the best way possible. Thus, we tuned to optimize the parameters that impact them in order to enable the algorithm to perform the best. We used f1 (an harmonic mean between precision and recall) to evaluate the performance of our algorithm.

Please find in the function below the different pipeline and parameters we tried:

```
sss = StratifiedShuffleSplit(
    n_splits = 1000,
    test_size=0.1,
    train_size=None,
    random_state=42)

def grid_search(features, labels):
    pipeline1 = Pipeline((
    ('scale',MinMaxScaler()),
    ('kbest', SelectKBest()),
    ('kneighbors', KNeighborsClassifier()),
    ))

    pipeline2 = Pipeline((
    ('tree', DecisionTreeClassifier()),
    ))

    pipeline3 = Pipeline((
    ('scale',MinMaxScaler()),
    ('kbest', SelectKBest()),
    ('svc', SVC()),
    ))

    parameters1 = {
    'kneighbors__n_neighbors': [3, 7, 10],
    'kneighbors__weights': ['uniform', 'distance'],
    'kbest__k': [3,5,10]
    }

    parameters2 = {
    'tree__criterion': ('gini','entropy'),
    'tree__splitter':('best','random'),
    'tree__min_samples_split':[2, 10, 20],
    'tree__max_depth':[10,15,20,25,30],
    'tree__max_leaf_nodes':[5,10,30],
    'tree__max_features': [3,5,10]
    }

    parameters3 = {
    'svc__C': [0.01, 0.1, 1.0],
```

```
    'svc__kernel': ['rbf', 'poly'],
    'svc__gamma': [0.01, 0.1, 1.0],
    'kbest__k': [3,5,10]
    }

    pars = [parameters1, parameters2, parameters3]
    pips = [pipeline1, pipeline2, pipeline3]

    print "starting Gridsearch"
    for i in range(len(pars)):
        gs = GridSearchCV(pips[i], pars[i], scoring = 'f1', cv=
sss, n_jobs=-1)
        gs = gs.fit(features, labels)
        print "finished pipeline"+ str(i+1) + " Gridsearch"
        print("The best parameters are %s with a score of %0.2f" %
(gs.best_params_, gs.best_score_))
```

Evaluation

As explained before, we used f1 to evaluate the efficiency of our models.

We got the following results: - With financial features and full dataset: - Best KNeighbors Classifier had the following parameters {'kbest__k': 3, 'kneighbors__n_neighbors': 3, 'kneighbors__weights': 'uniform'} and got a score of 0.38. - Best DecisionTree Classifier had the following parameters {'tree__criterion': 'gini', 'tree__max_depth': 20, 'tree__min_samples_split': 2, 'tree__splitter': 'random', 'tree__max_leaf_nodes': 30} and got a score of 0.34. - Best SVM Classifier had the following parameters {'svc__gamma': 1.0, 'kbest__k': 10, 'svc__kernel': 'poly', 'svc__C': 1.0} and got a score of 0.08. - With financial and email data and a reduced dataset: - Best KNeighbors Classifier had the following parameters {'kbest__k': 3, 'kneighbors__n_neighbors': 3, 'kneighbors__weights': 'uniform'} and got a score of 0.26. - Best DecisionTree Classifier had the following parameters {'tree__criterion': 'entropy', 'tree__max_depth': 20, 'tree__min_samples_split': 2, 'tree__splitter': 'random', 'tree__max_leaf_nodes': 30} and got a score of 0.32. - Best SVM Classifier had the following parameters {'svc__gamma': 1.0, 'kbest__k': 10, 'svc__kernel': 'poly', 'svc__C': 0.01} and got a score of 0.07.

We noticed that adding the two created features about email interactions did not improve the performance of our classifiers BUT we had to remove more than 30% of our dataset to use them. So, the decrease in efficiency from 0.38 to 0.34 with the

KNeighbours Classifier is hard to assess as we removed a lot of data. Those features could have improved our classifier if we had data for all the observations.

Conclusion

In conclusion, our best classifier is the KNeighbours Classifier with 3 features. It got a f1 score of 0.38. According to tester.py, the precision is about 0.58 and the recall is about 0.34 which means: - when the model predicts that someone is a person of interest, then there is around 58% chance that it's right - when someone is a person of interest, then the model has 34% of identifying him.

This results are satisfactory as in legal cases it's better to have a strong precision to preserve the presumption of innocence.