UDACITY

Data Analyst Nanodegree
Project P5 – Machine Learning
Identify Fraud from Enron Email

Submitted by Narendran Santhanam

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"]

Introduction

The Enron Corporation, an energy trading company based in Houston, Texas, faced a huge scandal in 2001 followed by bankruptcy, as a result of unethical business practices that included using accounting limitations to misrepresent earnings and modify the balance sheet to indicate favorable performance. The magnitude of the scandal meant that there were multiple people in the company involved. This project aims to identify such "persons of interest" (POI) based on financial and other information that are publicly available.

Data – overview and exploration

The data available for this analysis includes a lot of financial and email information about a list of Enron employees. Here are some characteristics about the dataset:

- No. of records in the dataset : 146
- No. of POIs in the dataset : 18
- Proportion of POIs in the dataset : 12.33%
- No. of features in the dataset : 21 (including the POI feature)
- No. of financial features : 14
- No. of email features : 6
- Proportion of missing data : 44.29%

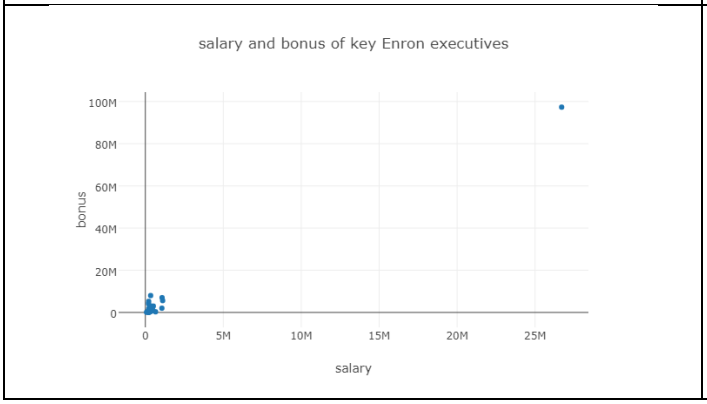
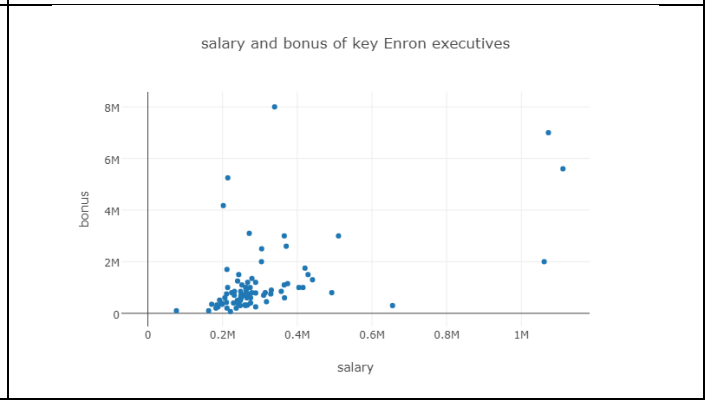
There are a couple of key observations from exploring the dataset:

1) The dataset is highly unbalanced, i.e., the number of POIs is very less, at 12.33% of the total number of records.
2) The dataset is highly sparse, i.e., there are a lot of missing data.

Outlier identification and removal

Exploring the dataset revealed that there are certain data points that need to be removed before analysis.

1) 'TOTAL' was an outlier data point that was the sum of all financial information of the executives in the dataset. We can see how dramatically the plot changes before and after removing this data point.

Before outlier removal	After outlier removal
	

Although the second plot seems to have some outliers, these data may be useful for informing our algorithm (given the sparse nature of the dataset) and it's better to retain them.

2) One of the executives, Eugene E Lockhart, did not have any information in the dataset (other than the POI column). This record will not be of any use in our analysis and can be removed.
3) Examining the names of the executives, one of the records was named 'THE TRAVEL AGENCY IN THE PARK' – this is clearly an erroneous record and can be removed.

---

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 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"]

Working with features

Feature Engineering

There were three features added to the existing ones:

1) Examining the financial features, the features bonus and salary stand out as the major ones. Bonuses are normally linked to the financial performance of the company, and a high bonus-to-salary ratio may serve as a reasonable indicator of involvement in financial manipulation.
2) Examining the email features, the number of emails sent to and received from POIs for each executive is available. Using these, we can find if a majority of one's sent and received emails are to and from POIs. This gives us two features – ratio of emails sent to POIs and ratio of emails received from POIs.

Feature Scaling

Looking at the units in which the features are expressed, the financial features are in dollar units, and range in the millions, whereas the email features are in no. of emails units and range in the hundreds and thousands. There is a big difference in the scale of these two features. In order to overcome this, we can use the MinMaxScaler since it will preserve the zero values that exist due to the sparseness of the data.

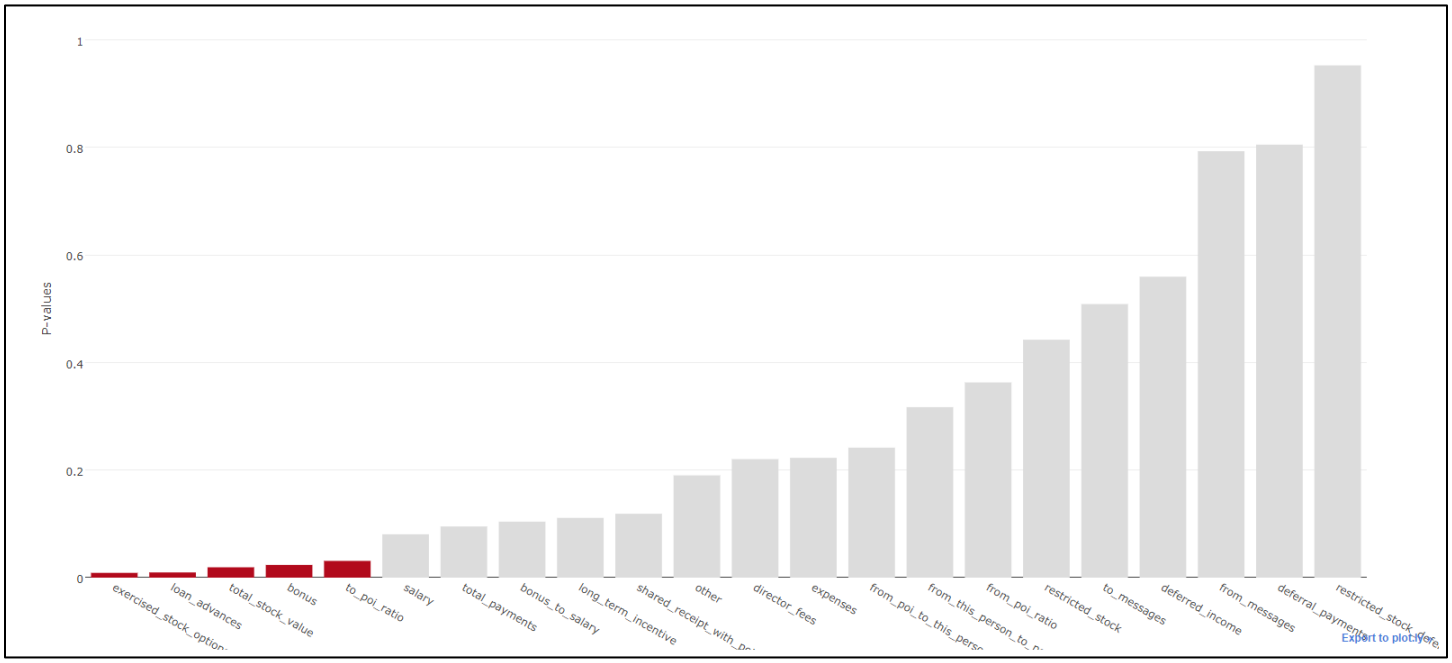
Feature Selection

I ran an exhaustive grid search using multiple classifiers, and used a SelectKBest feature selection step in the pipeline. The best classifier was Logistic Regression, with 5 selected features. The selected features and their corresponding p-values are shown below:

```
Feature: exercised_stock_options
P-value: 0.00888643603705
chi2 score: 6.84550933503
-----------------
Feature: loan_advances
P-value: 0.00970214838479
chi2 score: 6.68878173834
-----------------
Feature: total_stock_value
P-value: 0.0192725990709
chi2 score: 5.47661009929
```

```
-----------------
Feature: bonus
P-value: 0.0236413404294
chi2 score: 5.12075413709
-----------------
Feature: to_poi_ratio
P-value: 0.0311519043525
chi2 score: 4.64457200336
-----------------
```

These were the features with the lowest p-values in the dataset, as illustrated below:



---

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"]

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? 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"]

Model Selection

I considered a number of algorithms to test on the dataset. The list of algorithms that I ran on the dataset (and the parameters that I used) is given below:

- Support Vector Machine (C)
- Decision Trees (min_samples_split)
- Random Forests (min_samples_split, n_estimators)
- Naïve Bayes (None)
- Logistic Regression (class_weight)
- K Nearest Neighbors (n_neighbors)

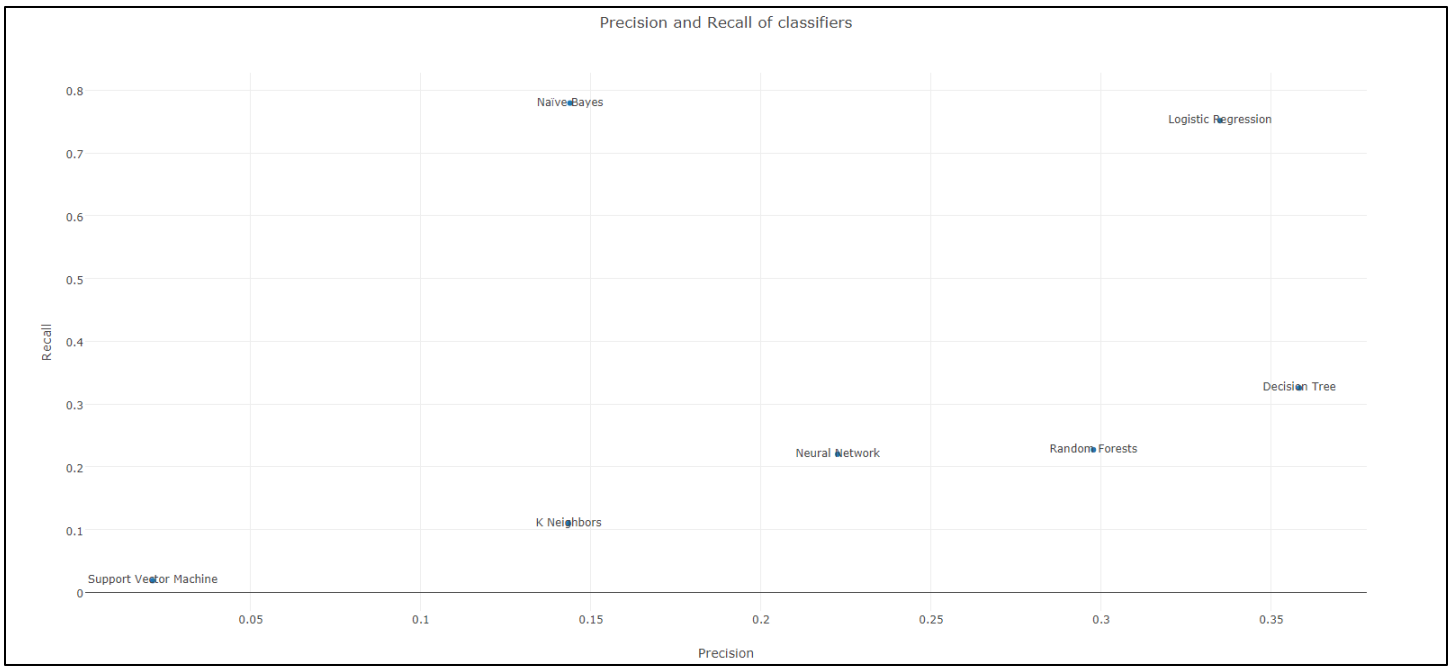
- Neural Network (Multi-Layer Perceptron) (hidden_layer_sizes, solver)

Each of these algorithms has a list of parameters that can be passed, and the performance may vary based on how we set these parameters. The performance of an algorithm for a certain set of parameters may also depend on the nature of data in the dataset. Unless we run multiple models with different parameters and gauge the performances against one another, we would not be able to achieve an optimal performance. Thankfully, sklearn offers a model selection framework called Grid Search, that enables one to search among a grid of parameters that can be as exhaustive as one wants it to be, and returns the combination that works best. I used precision, recall and F1 scores in the grid search, with F1 as the estimator for best performance.

I opted for selecting the model through a grid search cross validation by passing a wide range of parameters for each algorithm. The grid search took more than 14 hours to complete! The results of the grid search are given below:

```
Classifier: Support Vector Machine
Best score: 0.0191666666667
Best precision: 0.0211666666667
Best recall: 0.0185
Best parameters: {'dim_reduction__k': 6, 'clf__C': 100}

Classifier: Decision Tree
Best score: 0.32040952381
Best precision: 0.35825
Best recall: 0.3255
Best parameters: {'dim_reduction__k': 9, 'clf__min_samples_split': 2}

Classifier: Random Forest
Best score: 0.24425952381
Best precision: 0.297716666667
Best recall: 0.227
Best parameters: {'dim_reduction__k': 5, 'clf__n_estimators': 10, 'clf__min_samples_s
plit': 5}

Classifier: Naive Bayes
Best score: 0.231481653477
Best precision: 0.143841061716
Best recall: 0.7795
Best parameters: {'dim_reduction__k': 14}

Classifier: Logistic Regression
Best score: 0.448442685093
Best precision: 0.334929834055
Best recall: 0.7515
Best parameters: {'dim_reduction__k': 5, 'clf__class_weight': 'balanced'}

Classifier: K Nearest Neighbors
Best score: 0.116926984127
Best precision: 0.143442857143
Best recall: 0.1095
Best parameters: {'dim_reduction__k': 7, 'clf__n_neighbors': 3}

Classifier: Neural Network
Best score: 0.206667460317
Best precision: 0.222538095238
Best recall: 0.22
Best parameters: {'clf__hidden_layer_sizes': (10, 5), 'clf__solver': 'lbfgs', 'dim_re
duction__k': 15}
```

In the end, logistic regression seemed to work best, with 5 selected features. The performance plot of these algorithms are shown below:



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 items: "discuss validation", "validation strategy"]

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"]

Validation

One of the classic mistakes in machine learning is running an algorithm against a set of data for training purposes, and testing the performance against the same set of data. A machine learning algorithm is said to perform well only if it gives reasonable good results on data that it has not seen before. In order to simulate this, it is common practice to split the dataset into a training set and a testing set.

But, if the training and testing sets are the same for multiple runs of the algorithm, it will get tuned to perform well just for that testing set and not any new data. This is called overfitting. In order to overcome this, we do what is referred to as "K-fold cross validation" that splits the dataset into multiple "folds" such that each record will be a training data point and a testing data point at least once.

In order to perform K-fold cross validation, I used a stratified shuffle split method, since the dataset is highly imbalanced. The grid search model selection method comes with a built in cross validation. I used a 1000-fold cross validation to make sure the performance was averaged well over multiple combinations of training and test data points.

Given the highly imbalanced nature of the dataset, accuracy would not be an appropriate metric to assess the performance of the algorithms. Instead, I used precision, recall and the F1 score (the harmonic mean of precision and recall) as the scoring parameters in the grid search.

Precision indicates the proportion of the records identified by the algorithm as POIs, who are actually POIs, whereas recall indicates the proportion of the actual POIs identified by the algorithm as POIs. Recall is more important in such a dataset, where the emphasis is to identify as many potential POIs as possible.