Project: Identifying Fraud From Enron Emails

Machine Learning Project by VIDYA KESAVAN

TABLE OF CONTENTS

1. Introduction

In 2000 Enron was one of the largest companies in the United States. By 2002, the company went bankrupt due to widespread corporate fraud. The Federal Energy Regulatory Commission, made public and posted to the web, data from about 150 users, mostly senior management of Enron.

2. Goal of the project

The goal of this project is to build an algorithm using Machine Learning to identify Enron employees (known as Person of Interest, POI), who may have committed fraud based on the publicly available email and financial records.

The project dataset already contains a hand-generated list of persons of interest in the fraud case. These are individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Machine learning is highly useful in working with large datasets to construct algorithms that can learn from historical data and make predictions on new input data. In this project, machine learning is used to learn the features of what makes an employee a POI and use it to predict POIs in a similar fraud.

3. Data Exploration

A quick look at the dataset reveals the most important characteristics.

Length of the dataset is: 146
Count of POIs in the dataset is: 18
Count of Non-POIs in the dataset is: 128
Number of features in the dataset is: 21

Number of NaN's per feature is:
{'salary': **51**, 'to_messages': **60**, 'deferral_payments': **107**, 'total_payments': **21**, 'long_term_incentive': **80**, 'loan_advances': **142**, 'bonus': **64**, 'restricted_stock': **36**, 'restricted_stock_deferred': **128**, 'total_stock_value': **20**, 'shared_receipt_with_poi': **60**, 'from_poi_to_this_person': **60**, 'exercised_stock_options': **44**, 'from_messages': **60**, 'other': **53**, 'from_this_person_to_poi': **60**, 'deferred_income': **97**, 'expenses': **51**, 'email_address': **35**, 'director_fees': **129**}

The next step is to identify how many of these 21 features are missing some values. In this project, the missing values will simple be converted to zero.

3.1. Outliers

Outlier detection and removal:

In order to build an effective model, it is important to detect and remove outliers (where necessary).

Since this is a small dataset of 146 observations, it is easy to find some outliers just by looking at the data.

1. I can see a record for 'TOTAL' which in fact is just the sum of all observations. This is clearly one outlier that can be removed.
2. 'THE TRAVEL AGENCY IN THE PARK' is not an Enron employee and appears to be an external agency.This is one more outlier that I can safely remove.
3. The employee 'LOCKHART EUGENE E' does not seem to have data for any feature. Since all values are NaN for this employee, I can remove this record also as an outlier.

4. Feature Selection

Feature Selection in machine learning is the process of selecting relevant features for use in model construction.

4.1 Manual feature selection

deferral_payments, loan_advances, restricted_stock_deferred, director_fees have over a 100 missing values. I removed these features as they may not be very helpful in building an identifier.

Salary and Bonus are highly correlated. Employees with higher salaries are likely to get a higher bonus too. I removed these 2 variables and instead introduced a bonus to salary ratio to represent this relationship.

4.2 New Features

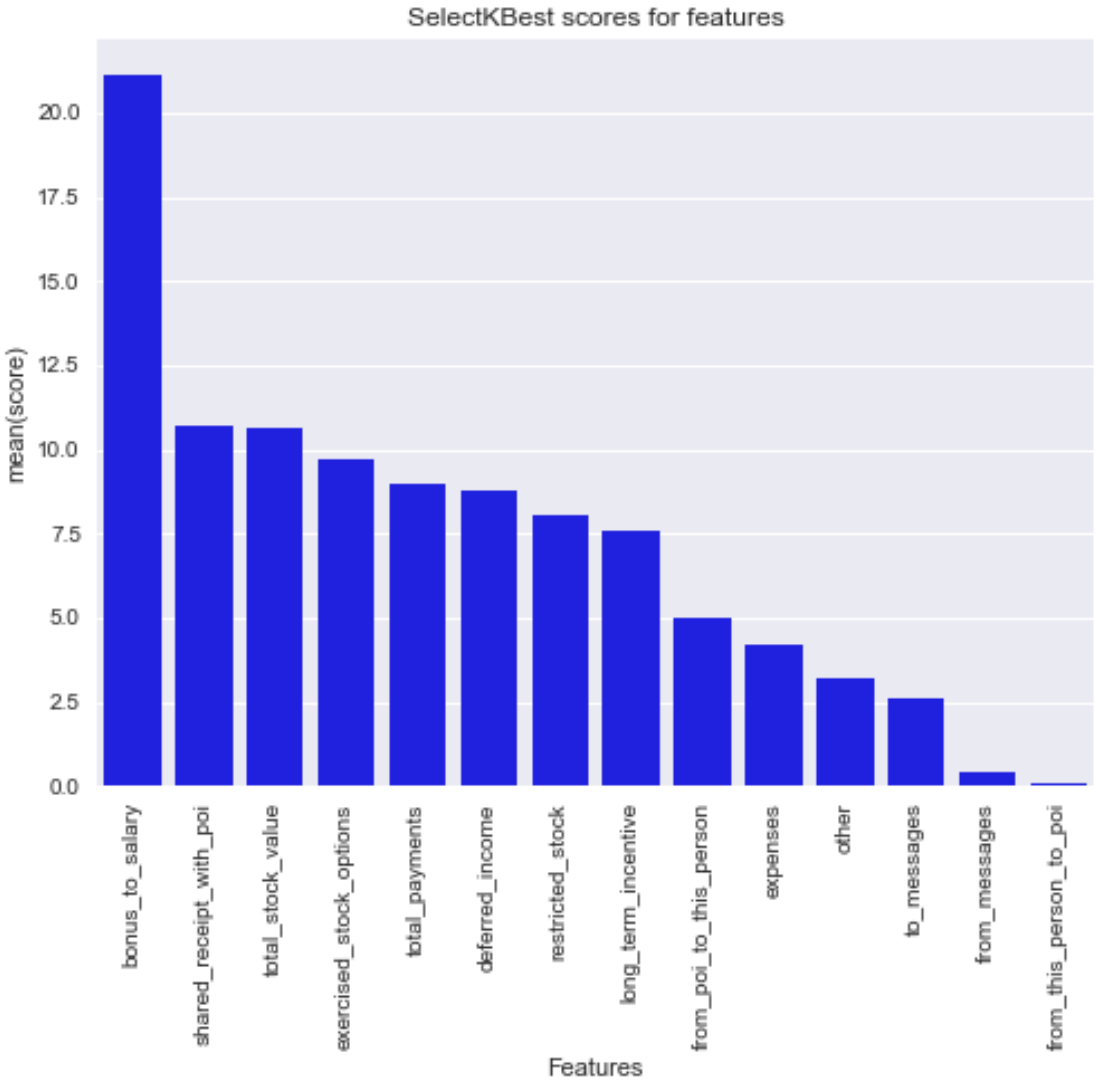
I created a bonus_to_salary feature to represent the ratio between the two variables.

bonus_to_salary_ratio = bonus/salary

4.3 Univariate Feature Selection using SelectKBest

Plotting the SelectKBest scores for all the features, I can see that bonus_to_salary has the highest score. This plot gives me an approximate value of 8 for K, after which the score decreases faster.

Bonus_to_Salary feature has the best K-Score. This is expected since both bonus and salary are important variables in the dataset, hence the ratio seems to have a huge impact on the model.



4.4 Feature Importances using Decision Tree

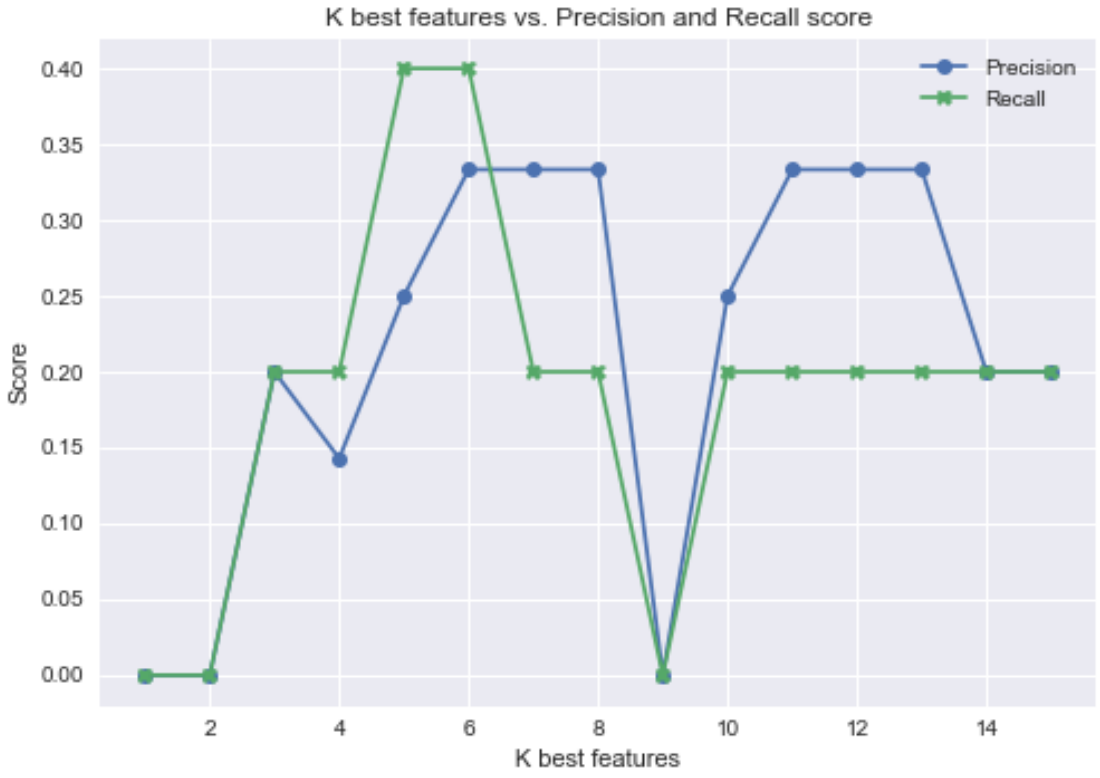
The decision tree feature_importances, also suggests the bonus_to_salary as the most important feature. But there are some very different results with the other features.

1 feature bonus_to_salary (0.217015423912)
2 feature restricted_stock (0.183260981809)
3 feature deferred_income (0.141467727675)
4 feature total_payments (0.137046861185)
5 feature other (0.132625994695)
6 feature expenses (0.101329066846)
7 feature from_messages (0.0757862826828)
8 feature from_this_person_to_poi (0.0114676611954)
9 feature to_messages (0.0)
10 feature from_poi_to_this_person (0.0)
11 feature total_stock_value (0.0)
12 feature shared_receipt_with_poi (0.0)
13 feature long_term_incentive (0.0)
14 feature exercised_stock_options (0.0)
15 feature email_address (0.0)

4.5 Determining KBest features

I computed the Precision and Recall values for each k value, to determine the best k value.

The Precision and Recall both peak at k=6, and higher k values have a lower precision and recall. So I decided to go with k=6.



5. Algorithm

I tried Naive Bayes, Decision Tree and Support Vector Machines on the data. These are the model performances.

Model Name	Precision	Recall	F1
Naive Bayes	0.34989	0.39100	0.36930
Decision Tree	0.32586	0.34900	0.33704
Support Vector Machines	0.17567	0.14800	0.16065

I picked Decision Tree Classifier as my algorithm for further tuning in the next section.

6. Tuning the algorithm

Parameter Tuning

Learning algorithms (like Decision trees) require us to set some parameters before we can use the models. These parameters whose values are set in the beginning of the algorithm are called hyperparameters. The process of choosing the optimal set of hyperparameters for a learning algorithm is called ***parameter tuning***. A metric is identified to measure the performance of the machine learning algorithm and hyperparameters that optimise this metric are chosen.

One of the traditional ways of tuning parameters in machine learning is to use GridSearch. GridSearch performs an exhaustive sweep through a subset of parameters specified and identifies the optimal parameter values.

The performance of the grid search algorithm can be measured using validation on a reserved validation set or using cross validation on the training set. Cross validation is commonly used when there is not enough data available to partition it into separate training and test sets without losing significant modelling capability.

This project uses GridSearch with Cross Validation for parameter tuning.

If parameter tuning is not done correctly, the algorithm may either be too generic or too specific. These two common problems are called under-fitting and over-fitting.

Over-fitting happens when the algorithm is trained too specific to the training data. The algorithm learns the noise and random fluctuations in the training data as concepts. These may not apply to the testing data and hence the algorithm could show poor performance.

Under-fitting happens when the algorithm is not trained well enough to the training data. In this case, the algorithm cannot perform well on a future data set because it has not learnt the concepts well.

The parameters I tuned for decision tree are:

Parameter	Values
Criterion	Gini, Entropy
max_leaf_nodes	None,5,10,20
max_depth	4,5,6,7
min_samples_leaf	1,5,10
min_samples_split	2,4,6

I used GridSearchCV to systematically tune the parameters and pick the best ones.

The selected KBestFeatures are:
'deferred_income', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'total_stock_value', ‘bonus_to_salary'

7. Validation

Model validation is the process where a trained model is evaluated against a test dataset. A common mistake in validation strategy is not using a test dataset to evaluate the model and instead evaluating the model on the training dataset. Using a test dataset evaluates the performance on an independent dataset and serves as a check on overfitting.

In this case, since the dataset is sparse, it makes sense to use StratifiedShuffleSplit to split the data into training and testing sets.

In normal test/train split, the algorithm is fitted using one training subset and validated using one test subset. Since there are very few data points available and the number of POIs in the sample is very small compared to number of non-POIs, a simple test/train split might give an unbalanced sample (All POIs could be in test test, for example).

Stratified Shuffle Split returns test/train splits while preserving the percentage of samples for each class. This way we can fit and validate the algorithm on a better balanced sample.

8. Evaluation Metrics

Given that the data is highly skewed with more non-POIs than POIs, accuracy is not a good metric to evaluate the performance of the model as we could get a lot of non-POI predictions right and none of the POI predictions correct and still get a good accuracy score.

Precision and Recall are the best metrics to evaluate the performance of the model on the given dataset.

Precision is a metric that answers the question, out of all the employees that were identified as POI, how many were actually a POI.

Recall is a metric that answers the question, out of all the actual POI employees, how many did the model classify correctly as POI.

9. References

https://discussions.udacity.com/t/sklearn-selectkbest-how-to-create-a-list-of-feature1-score-feature2-score/214617

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

https://discussions.udacity.com/t/plotting-precision-vs-recall-using-pandas-plot-how-to-facet-by-clf/182117/3

https://www.cs.cmu.edu/~enron/

https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)

https://stackoverflow.com/questions/22903267/what-is-tuning-in-machine-learning

https://en.wikipedia.org/wiki/Cross-validation_(statistics)