Udacity Enron Machine Learning Project
Arata Tomiyoshi Kagan

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?

The goal of this project is to build a prediction model using the Enron dataset to detect employees who committed fraud in the famous early 21st century Enron scandal. Machine learning is useful for exploring this dataset because the dataset contains both feature data (Enron employees' financial data) and labels (whether the employee committed fraud). In this project, I successfully used a supervised learning algorithm that learns the label POI (person of interest) based on the input feature of the data (financial records). The algorithm can learn from the features data to determine who committed fraud.

The Enron dataset includes 145 data points from Enron employees with 21 features including their salary, total stock values, number of contacts with POIs (person of interest), etc. By selecting features from the dataset, applying different machine learning algorithms and fine tuning their parameters, this algorithm tries to learn from the dataset and predict who is a POI. POIs are Enron employees who were indicted, settled without admitting guilt, or testified in exchange for immunity from the Enron fraud case.

There are 18 POIs and 127 non-POIs in this dataset. There are two features where the majority of the datapoints are missing values: 'loan_advances' with only 3 data points and 'director_fees' with 16 data points. I chose not to discard these features since the dataset in its entirety is relatively small and discarding datapoints may impact the performance of the algorithm. In regard to outliers, there is a datapoint with the key name 'TOTAL.' This datapoint is an outlier because unlike the rest of the datapoints which reflect information from each Enron employee *individually,* this datapoint shows a figure which is an aggregate of *all* Enron employees' financial records. I used the dict.pop() function to extract the values from the key, 'TOTAL' and replace the values with 0.

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. 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.

Using the SelectKBest method, I initially preselected 14 features (see below) and deployed gridsearchCV to pick the best combinations of the feature in order to avoid overfitting the data. In the process of pre-selecting these features, different numbers of features were tested, incrementing from 2 features to 16 features. In addition, I used MinMax feature scaling to equally weigh each value and normalize negative values to compute a chi-squared test for SelectKBest. After reaching 14 features, the evaluation metric score did not improve. Therefore, I selected 14 features to incorporate in my algorithm:

('feature name','feature score','p-value')('exercised_stock_options', '6.85', '0.009'), ('total_stock_value', '5.48', '0.019'),('bonus', '5.12', '0.024'), ('salary', '3.05', '0.081'), ('total_payments', '2.78', '0.095'),('long_term_incentive', '2.54', '0.111'), ('shared_receipt_with_poi', '2.43', '0.119'),('other', '1.72', '0.190'), ('expenses', '1.49', '0.223'), ('from_poi_to_this_person', '1.37', '0.242'), ('from_this_person_to_poi', '1.00', '0.317'), ('restricted_stock', '0.59', '0.443'), ('to_messages', '0.44', '0.509'), ('deferred_income', '0.34', '0.560')

As part of the process of creating new features from existing features, I conducted a feature analysis and examined which features are highly correlated using Pandas corr method. As a result, I discovered that the following features are highly correlated: 'exercised_stock_options' and 'total_stock_value' (0.963921), 'restricted_stock' and 'total_stock_value' (0.780558), and 'salary' and 'bonus' (0.649638).

Based on these results, I created three new features: 1) 'ratio_restricted_total_stock' as of computing the ratio between 'restricted_stock' and 'total_stock_value,' 2) 'ratio_exercised_stock_total_stock' as of computing the ratio between 'restricted_stock' and 'total_stock_value,' and 3) 'salary_times_bonus' by multiplying 'salary' and 'bonus.' In addition to the pre-selected 14 features, I incorporated 'ratio_restricted_total_stock' and 'ratio_exercised_stock_total_stock' into the algorithm because they increased the evaluation scores of precision and recall.

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I tested the Naive Bayes and Decision Trees algorithms and ended up choosing the Naive Bayes algorithm for the final project due to its better evaluation metrics. The following scores were the highest scores obtained from the Naive Bayes algorithm: Precision 0.43134, Recall 0.34550.

For Decision Trees, I obtained the following scores as the highest scores:
Precision 0.32780, Recall 0.30600.

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?

The goal of parameter tuning is to set the best combination of parameters and optimize the learning process of the algorithm as best as possible. For the Decision Trees algorithm, different combinations of parameters decide how to split nodes in the decision trees and the outcome of the different combinations create a big difference in the accuracy score of the algorithm. For example, if you do not set any parameter for the decision tree algorithm (meaning the algorithm uses the default setting of the parameter), the results are: Precision 0.23872, Recall 0.15350. By contrast, if I define the quality of split, minimal number of samples for splitting the nodes and maximum depth of the tree, the scores improve: Precision 0.32780, Recall 0.30600.

I used GridSearchCV to find the best combination of parameters for the decision tree algorithm. For the process of tuning the parameter, first, I produced the outcome without setting any parameter and used the score as a benchmark for further outcomes. For each parameter, I kept adding new parameter values until the outcome score became lower than the previous scores. I tested the following parameters: 'criterion,' 'splitter,' 'min_samples_split,' 'max_depth,' and 'max_leaf_nodes.'

'DecisionTree__criterion':['gini','entropy'],
'DecisionTree__splitter':['best','random'],
'DecisionTree__min_samples_split':[2, 10, 20],
'DecisionTree__max_depth':[10,15,20,25,30],
'DecisionTree__max_leaf_nodes':[5,10,30,40]

What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is the process of dividing a dataset into training and test data and using the test data to check whether the algorithm can accurately predict an unseen dataset (test data). If you do not split the dataset and validate the test data on training data, the model may pay too much attention to the existing data and produce a poor result when trying to predict a future dataset - which is called "over-fitting." For the validation process for this project, I used the train_test_split method from sklearn's cross-validation. I split the training data and test data based on a 7:3 ratio. In order to test the accuracy of the training data, I applied the test data from the 7:3 split to see if the evaluation metrics perform well.

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.

The two main evaluation metrics for this project are “precision scores” and “recall scores.” From the Naive Bayes algorithm, I obtained 0.43134 for the precision score and 0.34550 for the recall score. In this case, we have a higher precision score and a lower recall score. This means that when the machine is presented with an Enron employee, it is more likely to think that he or she is a POI (person of interest) than not. When the employee is actually a POI, the machine has a higher chance to correctly categorize the employee accurately.

References:

https://stackoverflow.com/questions/22903267/what-is-tuning-in-machine-learning
https://www.quora.com/What-is-parameter-tuning-in-machine-learning
https://www.geeksforgeeks.org/confusion-matrix-machine-learning/