Beth Morrissette - Introduction to Machine Learning (Analysis)

# Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it.

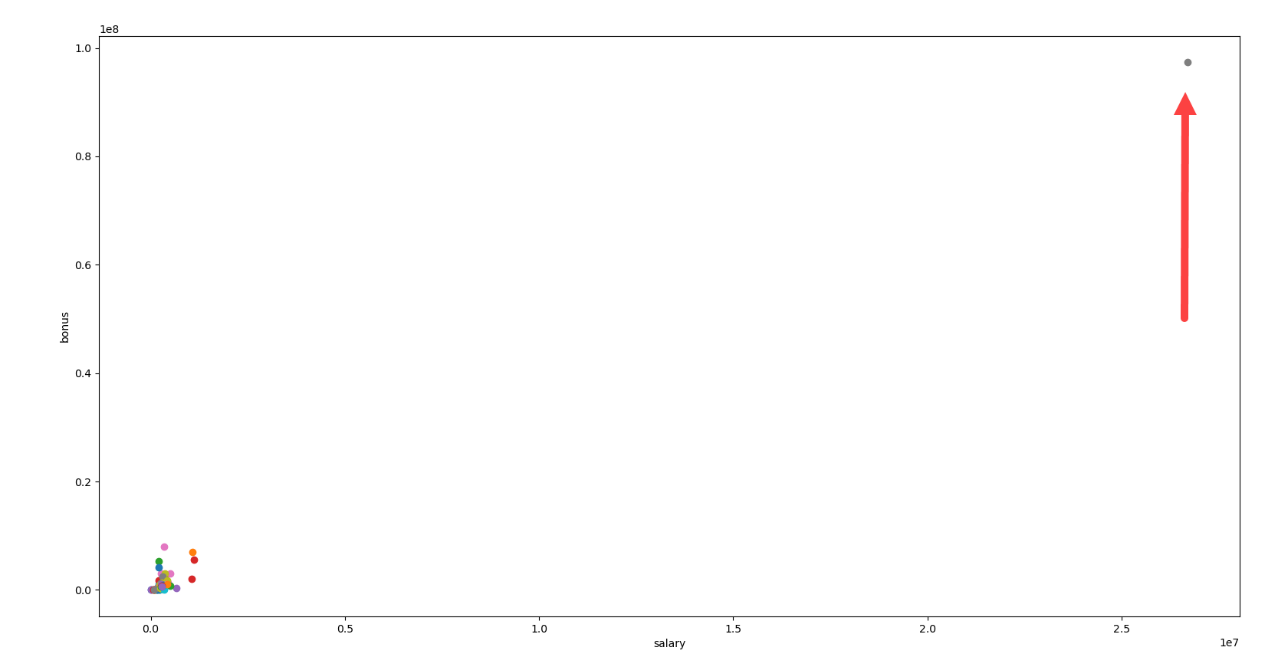
The goal of this project it to review the Enron data set and see if we can identify “persons of interest” based on email communications. “Persons of interest” or “POI” are defined as people who may have had knowledge of and/or participated in the Enron scandal of 2002. Machine learning is uniquely positioned to assist with this work. Consisting of algorithms designed to make predictions based on data previously collected, machine learning can evaluate emails related to both “POI” and “Non-POI” senders. Based on this evaluation, it can predict the status of other, unknown, authors based on emails.

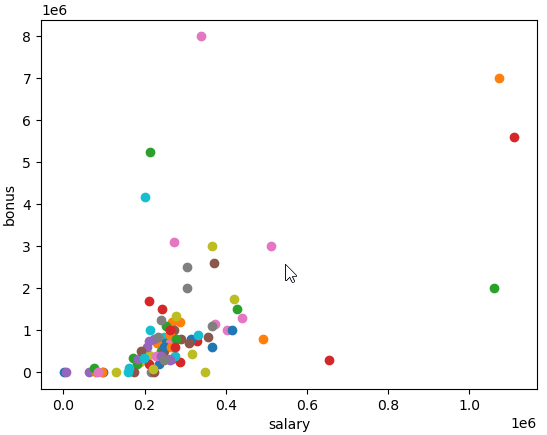
## Data Set Exploration

The data set provided in the “final\_project\_dataset” file consists of 145 people (there are 146 points but one is not a person – more on that later!) and each person has 21 features associated with them. Only 18 of the people in the initial data set are identified as “POI” which is a lower number than we’d like but still workable. Another area of concern is missing values. Checking several of the features, we find multiple “NaN” or null values in the responses. This could impact our algorithm’s ability to learn patterns in the emails.

## Outlier Identification and Removal

A quick review of the data set shows a single outlier when comparing salary and bonus. Further identification showed this was a “Total” row included in the data. This was removed from the dictionary using the dict.pop() method.

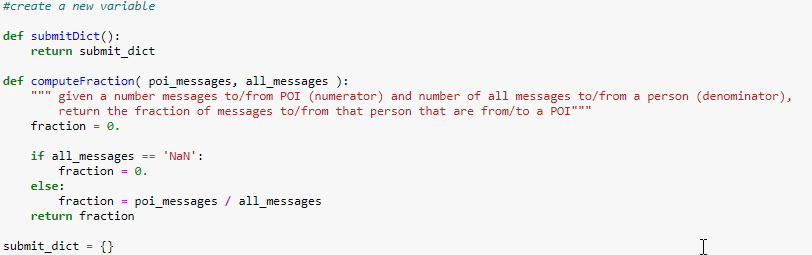
Figure 2 (Scatter Plot with Outlier)

Figure 3 (Scatter Plot with Outlier Removed)

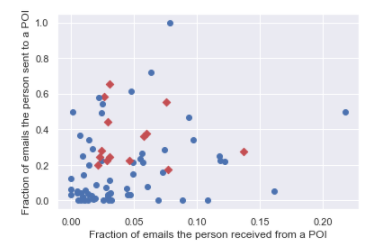
Additionally, in looking at the data set, we see an entry for “Travel Agency in the Park”. In a Time magazine article from 2002, we see that this was a florist business owned by Ken Lay’s sister. Since this is definitely not a POI or likely to give indication of POI features, we will remove it from the dataset.

Engineer new feature(s)

Using intuition, it seems like there should be a relationship between the amount of email exchanges between POI. The raw number in the dataset might not have as much impact as the percent. If a larger percentage of your email is with a POI, intuition tells me you are likely a POI as well. Created 2 new features - % from POI to person and %to POI from person.

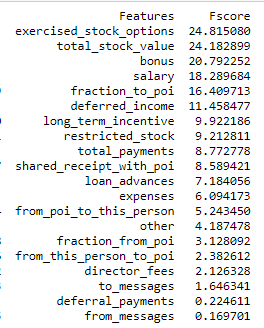


We’re then able to visualize this relationship and see that in general, identified POI were more likely to have a larger percentage of emails with each other than with non-POIs.



What features did you end up using in your POI identifier, and what selection process did you use to pick them?

Initially looked at the features with the SelectKBest function from SKlearn. This returned an array of fscores for the features. I decided to use Kbest because it was fairly simple to set up and evaluate.

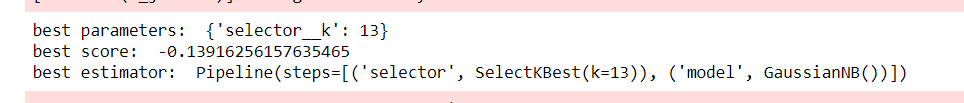
Interestingly, stock options and value had the highest f-scores but as expected, the percentage of emails sent to the POI plays a role in your category.

The new email fraction features are on a far different scale than the other items in the dataset so I wanted to scale the dataset in order to see if that made a difference

I used the SKLearn module MinMaxScaler to automate the scaling of the features.

Scaling the features didn’t have an impact on the KBest algorithm for feature selection but it will have an impact on the models. Models affected by unscaled data include the Support Vector Machine (with RBF kernel) and Kmeans since they compare dimensions against each other and if one is a much larger scale than the other, the comparison will be skewed

To ensure I use the optimal features for analysis, I combined SelectKBest with GridSearchCV. This checked all of the possible K values for Kbest to exhaustively determine which would be the best for the algorithm. Running this returned a K value of 13 and I used the transform function of Select K Best to reshape the features set to this shape.



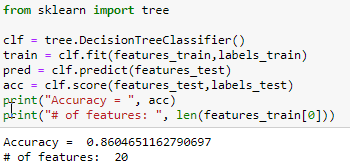
## Test Classifiers

So, let’s try the algorithms with all of the features first and then we can see if using the suggestion from KBest works and the effect of scaling.

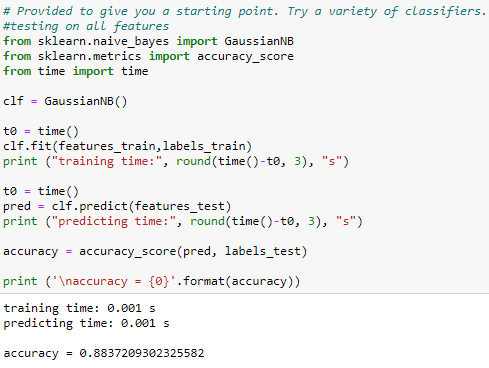
I used SKLearn to create a testing and a training set – standard machine learning recommendation is 30% of the set for testing.

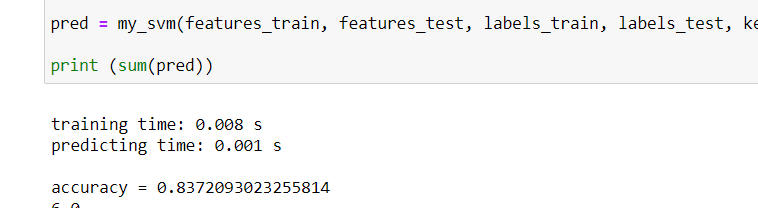


First, a simple decision tree.



86% is not a bad accuracy score but this was with all of the features and the unscaled version of them.

For a second classifier, I tried a Naïve Bayes algorithm. This had better accuracy than the decision tree and shouldn’t be affected by feature scaling.

And finally, let’s see how a support vector machine (SVM) works. We get our lowest accuracy rating with this algorithm yet.

Next, let’s see the impact of scaling data. I ran all 3 algorithms again with the scaled features. There was no real impact on the decision tree or the SVM or decision support tree but we do see a jump above 0.9 for the NB algorithm.

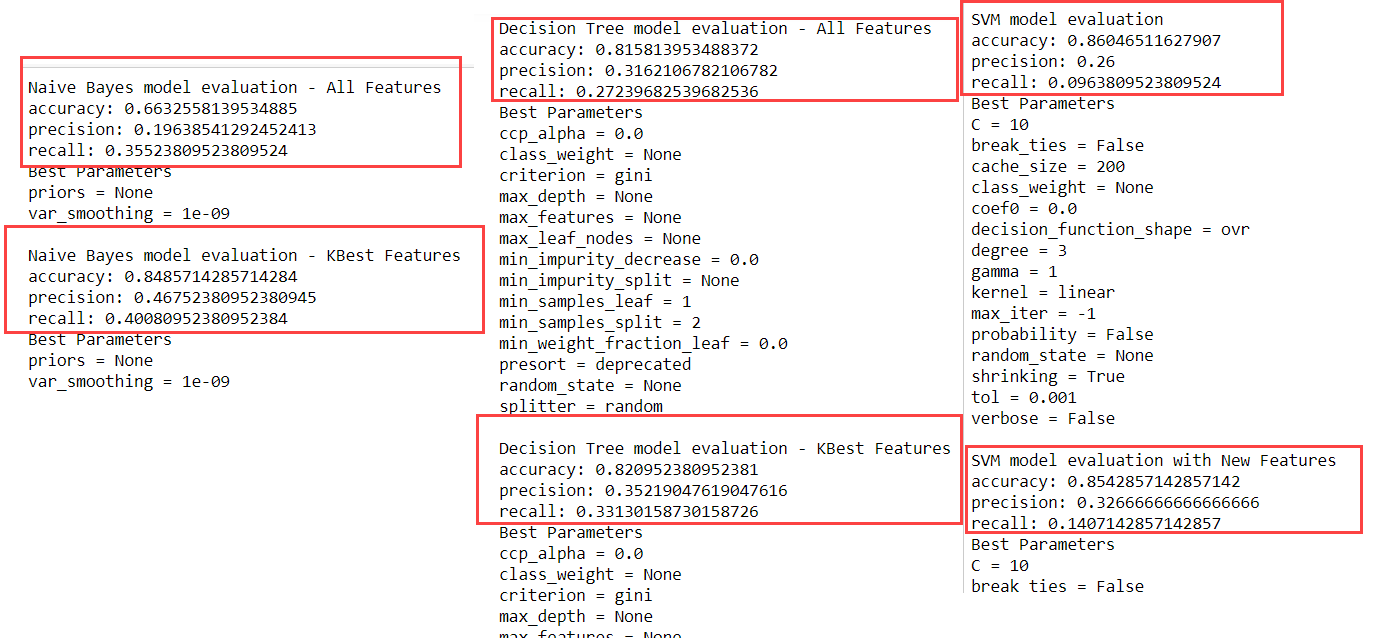
|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Accuracy | | |
| All Features | Unscaled | All Features | Scaled | Top Features | Scaled |
| Naïve Bayes (Gaussian) | 0.88 | 0.91 | 0.86 |
| Decision Support Tree | 0.91 | 0.91 | 0.83 |
| Support Vector Machine | 0.84 | 0.84 | 0.83 |

Finally, let’s see what the impact of using scaled features as well as the top features suggested by the Kbest algorithm. These were: exercised stock options, total stock value, bonus, salary, fraction of email to poi, fraction of email from poi and deferred income. Accuracy scores did go down but this may be reflective of overfitting in the initial models.

To further evaluate and identify the best algorithm for the analysis, we can use SKLearn’s Grid\_Search feature. This feature takes an algorithm (estimator) and checks the cross product of the possible parameters for it to find the best combination – in other words the highest precision/recall/accuracy. In checking this, I scaled all data and checked the parameter combinations using both the full feature set as well as the suggested feature set from KBest.

Additionally, I wanted to validate the test/train split to make sure that wasn’t impacting my scores – a mistake that can happen is a preponderance of one group in the test or train set. By using GridSearch along with 10 random iterations of test/train data creation, we can validate our algorithm. This is important due to our uneven balance of POI vs. Non-POI in the dataset.

This is also known as model tuning or hyperparameter optimization (Anyana, 2016). As stated in this article from busigense.com, “Hyperparameters refer to another kind of parameters that cannot be directly learned from the training process and need to be predefined”. This is important because depending on the size of data and parameters chosen, modeling can use a lot of computing resources and take a long time for results. I saw this even in this small evaluation where the SVM Model took over 15 minutes to complete on my personal machine. For this analysis, my best classifier was the NB Gaussian which didn’t require tuning as it has few parameters. However, the Decision Tree model suggested tuning of several parameters. I had originally used a criterion of ‘entropy’ as discussed in the lectures but GridSearch indicated ‘gini’ was optimal. Additionally, I had used “best” for splitter and GridSearch actually indicated “random” was better.



## Conclusions

In the end, the machine learning algorithm selected was the Naïve Bayes Model evaluation using a scaled feature set recommended by the KBest selector. The final values were accuracy: 0.86, recall: 0.38 and precision: 0.52. I decided to use this because it had good accuracy, precision and recall. Accuracy is a simple measure of the number of items labeled correctly divided by all items. It is not good for imbalanced data which with only 18 of over 100 people in the data set being POI, is an issue here. This is why we are more likely to use recall and precision as validation. Recall is the number of truly positive items in the data set that were correctly classified as positive. Precision looks at the number of items that were labeled as positive, how many truly belong in the positive class. For my identifier, both the false positive and false negative rates are low. This means I can identify POI’s reliably and accurately. If my identifier finds a POI, then it is highly likely the person is a POI.

Files Used

Udacity Machine Learning Classroom Lectures, Exercises, Starter Code

SkLearn Website (multiple modules) <https://scikit-learn.org/stable/index.html>

Stack Overflow Order of Machine Learning <https://stackoverflow.com/questions/46062679/right-order-of-doing-feature-selection-pca-and-normalization>

Comprehensive Guide to Automated Feature Selection <https://www.datagraphi.com/blog/post/2019/9/23/feature-selection-with-sklearn-in-python>

Feature selection via gridsearch

<https://medium.com/data-science-reporter/feature-selection-via-grid-search-in-supervised-models-4dc0c43d7ab1>

Hyperparameter Tuning

<http://busigence.com/blog/hyperparameter-optimization-and-why-is-it-important>