Machine Learning Project: ENRON

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"]**

   Answer

   The objective of this this project is to create a machine learning based model which can identify potential Persons Of Interest (POI) involved in the Enron scandal. It would do so by analysing the email and financial data which was released as a result of the fraud investigation.

   Machine learning can be utilized to achieve this objective as it can recognize trends hidden in large volumes of data. It can then "learn" these trends, and make valuable predictions based on new data from a similar dataset.

   a. Dataset Background

   Enron was one of the largest companies in the United States in 2000. By 2002, it collapsed into bankruptcy due to corporate fraud. During a federal investigation, this dataset comprising of emails and financial information was released to the public. This has been further processed by Udacity instructors for use in this exercise.

   b. Exploration

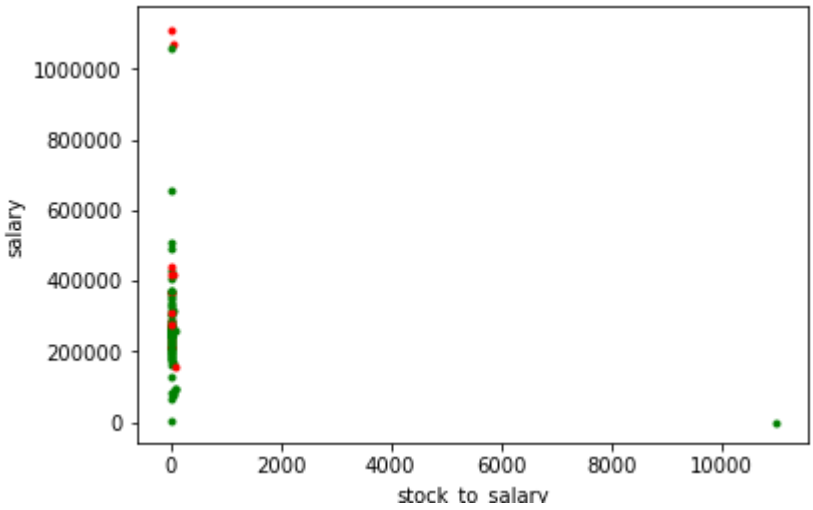
   As a first step, data exploration was carried out to get an overview. There were some anomalies. Outliers were found and treated, which is explained in the section below. Some more outliers were found in later iterations when new features were created. Missing or NaN values were replaced by 0 where appropriate to felicitate numerical operations. Key data characteristics are listed below

   - Number of datapoints: 146
   - Number of features per datapoint: 21
   - Number of poi in datapoints: 18
   - Number of non-poi datapoints: 128
   - Number of people with missing Total Payments Info: 21

c. **Outliers**

While exploring the data, some outlying values ('outliers') were seen. In fraud detection analysis, outliers are not always bad, i.e. they may often provide key features helpful for useful predictions.

- Given that data was structured by names, I separated out all data values whose "names" did not have either 2 or 3 parts. (e.g. First, middle & last). This upon inspecting this list, I found 2 outliers, "TOTAL" with extremely high values. This could possibly be a result of importing a summation line from a spreadsheet. The other was "THE TRAVEL AGENCY IN THE PARK", which is probably a business partner company. Both these data points were deleted
- Another characteristic was the high frequency of "NaN" values. This can sometimes cause errors with mathematical operators, and hence all of these (with the exception of emails) were changed to '0'
- One more data point which I removed in a later iteration was BANNANTINE JAMES M. I was creating a new feature Total Stock value/Salary. For James, this value was significantly higher than the group (>10,000). To be able to effectively use this feature, this data point was also removed. The graph bellows shows the distribution before any James' data was 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"]

Answer

I did not use any manual filtering on the features. Hence I used all available features, plus the 5 new engineered features from below. These features were first scaled, then the SelectKBest() algorithm was used to select the best features.

a. **Feature Engineering**: 5 new features were engineered and added to the database. They are

    1. **[to_poi_fraction]: % of mails sent to a POI / total mails sent**
       A person who mainly communicates with POI's may have a high likelihood of being a POI himself/herself
       Note: This new feature was selected by SelectKBest as one of the features finally used in the classifier

    2. **[from_poi_fraction]: % of mails received from a POI / total mails received**
       A person who mainly communicates with POI's may have a high likelihood of being a POI himself/herself
       Note: This new feature was selected by SelectKBest as one of the features finally used in the classifier

    3. **[shared_poi_rct]: % of mails with shared POI receipt / total mails received**
       A person who was often copied on mails with POI's may have a high likelihood of being a POI himself/herself
       Note: This new feature was selected by SelectKBest as one of the features finally used in the classifier

    4. **[salary_fraction] % of salary to total payment**
       A POI may have a higher likelihood of benefitting financially. As a significantly higher salary compared to peers may be difficult to justify, other means could have been used. (bonus etc). Hence I added a feature which compares salary to total payments

5. **[stock_to_salary] % of Total Stock Value / Salary**
   A POI may hold a large percentage of stocks V/S his/her salary. This feature tries to capture this information. An outlier, BANNANTINE JAMES M was found, who was removed from the dataset
   Note: This new feature was selected by SelectKBest as one of the features finally used in the classifier

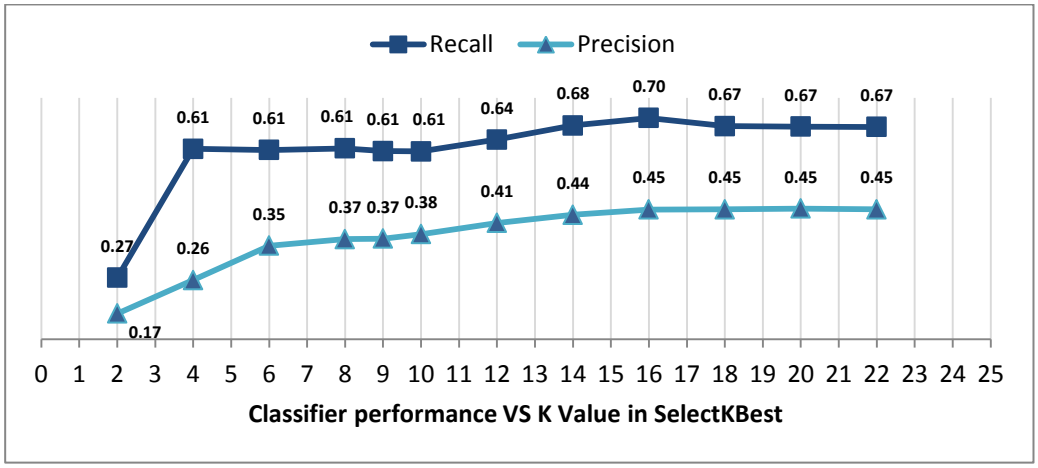
b. Feature Scaling & Selection

1. Scaling: Due to data points lying over a wide range of values, feature scaling was used in the form of the MinMaxScaler()

2. Selection: SelectKBest was used to perform automated feature selection, both before and after algorithm selection & tuning. The results of varying the k value in the final algorithm are as below

K Value / Performance	2	4	6	8	9	10	12	14	16	18	20	22
Recall	0.27	0.61	0.61	0.61	0.61	0.61	0.64	0.68	0.70	0.67	0.67	0.67
Precision	0.17	0.26	0.35	0.37	0.37	0.38	0.41	0.44	0.45	0.45	0.45	0.45

These values have been plotted on the graph below



3. Increasing the K value increases the precision, until K=16 (Precision = 0.45). On the other hand, increasing the K value increases the recall until k=16, after which it starts declining. Given this performance, K=16 was finally chosen as it provided the best performance for both Precision & Recall.

   The feature scores are as below
   1. ('loan_advances', '6.72')
   2. ('exercised_stock_options', '5.09')
   3. ('total_stock_value', '4.22')
   4. ('to_poi_fraction', '3.81') **(New feature made in last step)**
   5. ('bonus', '3.11')
   6. ('total_payments', '2.68')
   7. ('salary', '2.64')
   8. ('shared_poi_rct', '2.39') **(New feature made in last step)**
   9. ('long_term_incentive', '1.89')
   10. ('expenses', '1.44')
   11. ('shared_receipt_with_poi', '1.37')
   12. ('from_poi_to_this_person', '0.98')
   13. ('director_fees', '0.93')
   14. ('from_this_person_to_poi', '0.80')
   15. ('from_poi_fraction', '0.74') **(New feature made in last step)**
   16. ('stock_to_salary', '0.62') **(New feature made in last step)**

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"]

Answer

GridSearchCV was used to test out a number of algorithms, as listed below,

1. GaussianNB
2. DecisionTree
3. AdaBoost
4. ExtraTrees

Below are the results from different algorithms before tuning

Alogorithms/ Performance	GaussianNB	DecisionTree	AdaBoost	Extra Trees
Recall	0.23	**0.65**	0.36	0.19
Precision	0.30	**0.35**	0.31	0.29
F1	0.25	**0.43**	0.31	0.22

Given the results, the DecisionTree algorithm was chosen for further tuning.

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"]**

Answer

Tuning implies varying & testing the parameters of an algorithm to produce the desired results in terms of quality and computational time.

Not tuning or using inappropriate parameters could not only produce poor results, but also feed on computational power. It may also produce deceptive results which may seem OK but would not scale well to untested data.

I used GridSearchCV to fine tune certain parameters of my chosen algorithm from the last step.

The parameters I tuned were

DecisionTreeClassifier			
Sn.	**Parameters**	**Options**	**Final Choice**
1	min_samples_leaf	[1,2,3,4,5]	1
2	min_samples_split	[2,3,4,5]	3
3	max_depth	[1,2,3]	2
4	class_weight:	['balanced',None]	balanced
5	criterion	['entropy','gini']	gini
6	splitter	['best','random']	best

The best performance was achieved with the above mentioned parameters which take into account the output from the GridSearch above, along with manual adjustments.

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"]**

   Answer

   Validation involves letting the algorithm make predictions for "untrained" data and measuring the performance of these prediction. It's an essential process to measure the quality of a machine learning model

   A classic mistake can be made if the Machine Learning model is trained on the entire available data. This can cause "Overfitting", where the algorithm works very well but for only this specific data. It is not scalable and cannot generalize its predictions.

   I used Stratified Shuffle Split (like in tester.py) as the number of data points in this dataset was small. Also, Stratified Shuffle Split counters the class imbalance problem in the dataset, as the number of POI's is significantly less than the number of non-POI's. The following parameters were used

   sss=StratifiedShuffleSplit(labels, 1000, random_state=42)

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"]**

   Answer

   Since there were so few POIs, accuracy was not a very good metric. Hence I used Precision & Recall.

   The performance on the metrics is as below

   1. **Recall** tells us the proportion of the true POIs correctly identified by the algorithm. In this case it is .70 or 70%

2. **Precision** tells us how many of the algorithm identified positives were truly positives. In this case it is .45 or 45 %

In this algorithm, the recall value is .70. This means 70% of true positives were correctly identified. That is, if a person is a POI, there is a 70% chance that the algorithm will catch him/her.

The other metric, precision, has a value of .45. This means that if the algorithm predicts that a person is a POI, then there is a 45% chance that the person is truly a POI.

I chose the algorithm and its parameters to maximize "Recall" & optimize "Precision". As this is probably the first step in the investigation this strategy should provide the first list of potential POI's which would then be further investigated. Hence, it is better to correctly identify a high % of 'actual POI's' in this step, while maintaining an acceptable level of precision.

Statement: I hereby confirm that this submission is my work. I have cited below the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.

References

1. **https://www.udacity.com/**
   (Machine Learning Project: Data input, Validation & Evaluation code)
2. **http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html**
3. **http://scikit-learn.org/stable/auto_examples/plot_compare_reduction.html#sphx-glr-auto-examples-plot-compare-reduction-py**
4. **http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html**
5. **http://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html#sphx-glr-auto-examples-model-selection-plot-multi-metric-evaluation-py**