Gabriel Leupin

Introduction

This project applies supervised learning techniques to classify data from a famous case of fraud at Enron in 2003. The set includes 19 categories (p=19) of data, such as a given person's salary or bonus, for 146 people (n=146), as well as a label indicating whether the person is 'of interest' ('poi') or not 'of interest' ('non-poi'), i.e., whether or not a person was suspected of fraudulent activity during the case's investigation. Four machine learning classifiers, including a gaussian naive Bayes classifier, a decision tree, a random forest, and a support vector classifier are trained on shuffled samples of the data, and used to predict whether a person was 'of interest' or not 'of interest' on samples of data whose labels are unknown. The models' predictions are then tested against against the entire data set and evaluated by their respective F1 scores.

Data Exploration

Exploration and subsequent cleaning of the data were undertaken to remove outliers and nonsensical data points. Three entries were removed, one of which included the 'TOTAL' sum of the values in each data category, 'LOCKHART EUGENE E' , which contained no information, and a final entry which contained an irregular name, 'THE TRAVEL AGENCY IN THE PARK'.

Two variables, composed of variables in the original set, were added to the data. One variable, ['from_poi_message_ratio'], represented the proportion of total e-mail correspondence sent to pois, and the other, ['to_poi_message_ratio'], represented the proportion of e-mails received from pois. These were added because, given that those emails more likely contained content indicating or suggesting criminal activity, a higher proportion suggested more criminal activity. While the variables allow for further exploration of the data, they were added without an a priori understanding of the information they might yield. It was therefore possible that, despite their inclusion in the set, no further exploration of that particular data would take place.

There were a number of entries for which no data were recorded. The categories below are followed by the number of entries for which no data are recorded--
poi 0
salary 49

deferral_payments 105
total_payments 20
loan_advances 140
bonus 62
restricted_stock_deferred 126
deferred_income 95
total_stock_value 18
expenses 49
exercised_stock_options 42
other 52
long_term_incentive 78
restricted_stock 34
director_fees 127
to_messages 57
from_poi_to_this_person 57
from_messages 57
from_this_person_to_poi 57
shared_receipt_with_poi 57

Each variable's descriptive statistics, including the means, maxes, and standard deviations, were included during the exploration phase, to verify that none of the remaining data were irregular. No data was removed after the outliers were removed.

Pipeline and GridSearchCV

Hyper-parameters are constraints under which models operate. They indicate "parameter(s) of a prior distribution," are not learned by the model, and describe the operative boundaries of learning models. Hyper-parameter tuning methods are available in sklearn, including 'GridSearchCV'(gridsearchcv), and 'Pipeline'(pipeline).

Pipeline is a method available in the 'sklearn' package. It tests a model using a number of different hyper-parameters. According to the sklearn documentation, pipeline

"can be used to chain multiple estimators into one. This is useful as there is often a fixed sequence of steps in processing the data, for example feature selection, normalization and classification. (Pipeline) serves two purposes here:

Convenience: You only have to call `fit` and `predict` once on your data to fit a whole sequence of estimators.

Joint parameter selection: You can grid search over parameters of all estimators in the pipeline at once."

Gridsearchcv is another method in sklearn that often works with pipeline. Gridsearchcv tests and scores various model estimations which operate on a user-specified preset grid of parameters. It has the effect of being a flexible means of testing many variations of the same model at once. According to the sklearn documentation,

"A (grid)search consists of:

- an estimator (regressor or classifier such as `sklearn.svm.SVC()`);
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme; and
- a score function."

Further explanation of the elements within a grid search follows.

Cross-Validation

Cross-validation partitions complementary subsets of the data into a training set, a *known* set on which a given model is trained, and a testing set of *unknown* data, on which the same model is validated. According to wikipedia, it "performs multiple rounds...using different partitions. The

validation results are averaged over the rounds." All else equal, a model verified by cross-validation will perform better on an independent data set than will one which hasn't been cross-validated. Stratified shuffle split, a cross-validation method available in the 'sci-kit learn' (sklearn) package in the Python programming language, draws samples from random partitions of shuffled sets of data, and can be used as a robust means of evaluating model performance. As the Enron data set was small and imbalanced, i.e. out of 143 total people, only 18 were people 'of interest', the use of stratified shuffle split with a large number of folds reduced both the likelihood of any particular model's results being attributable to random chance and each model's sensitivity to small changes in the data. As explained in the Udacity discussion forum, in conjunction with GridSearchCv(explained below), stratified shuffle split gives the program 'a fair shot at developing an optimal model.'

Scoring

As accuracy scores for classification models can be misleading, choosing an appropriate scoring function is important for the researcher. In the case of imbalanced binary classes, where one category dominates the other, the accuracy score can be uninformative, as models will achieve a relatively high accuracy score by consistently labelling all of the data points as belonging to the dominant class (e.g. accuracy= 0.874125874 for guessing that every person is not of interest when there are 125 people who are not of interest vs. 18 people who are of interest in the original data set). In this case, it is more likely that a model which guesses that every instance belongs to the dominant class exhibits random performance which cannot be generalized to an independent data set.

Grid-searching over the hyper-parameters that maximize the F1 score, as opposed to the accuracy score, was therefore more suited both to the task of model performance optimization and to the project's aims of achieving a threshold precision and recall score of 0.3. The F1 is the harmonic mean of the precision score, which measures the "fraction of retrieved instances that are relevant," and the recall score, which is "the fraction of relevant instances that are retrieved." More specifically, precision is the proportion of positively identified labels that are actually positive, and recall is the proportion of the members of the positive class which were correctly identified. Therefore, precision measures the fraction of people that the model

identifies as people of interest (poi) who are actually people of interest, and recall measures the fraction of people of interest who are correctly identified as people of interest by the model. Broadly speaking, F1 score judges the model's relative ability to identify instances of a class without misidentifying others. Given an imbalanced dichotomous classification problem, it is less likely that an F1-optimized model produces random results, i.e., that the model's performance will degrade on an independent data set. Maximizing the F1 score, as opposed to the accuracy score, was therefore suited both to the task of model performance optimization and to the project's aims of achieving a threshold precision and recall score of 0.3.

Naive Bayes

This project examined four different machine learning classifiers. The first was a 'naive Bayes' classifier. Naive Bayes are "simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features." Naive Bayes classifiers are given a set of data features and labels during training. For held-out data points whose class is unknown, naive Bayes 'guesses' the point's label. Once the point's label is revealed to the model, it 'updates,' through application of Bayesian inference, its understanding of the associations between the data's heretofore-known features, and labels, which are 'unknown' until they are tested. It incorporates this understanding into its future guesses on data points that it tries to classify. In effect, the model incorporates increasingly more information about the data with each data point, and its concomitant knowledge of the associations between the features and labels grows incrementally more robust to noise in the data. Naive Bayes models tend to generalize well across data types and dimensions, though they are especially popular in text learning problems. Naive Bayes classifiers are often used in machine learning because of their generalizability, and were a suitable starting point for this project.

The following hyper-parameters were piped into the naive Bayes classifier: principal component analysis (PCA), "a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components", a 'feature scaler' which scales each variable according to its unit variance, and a 'selectkbest' algorithm which chooses the most informative variables.

In brief, principal component analysis provides lower-dimensional representations of data sets by describing linear spaces along which the data exhibit the most variance. These orthogonally-derived spaces are considered as being the spaces where the features are most descriptive of the targets. More discussion of PCA can be found here-- http://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues . Applied to this data set, PCA identified six axes of maximal variance from the available range of 5-10, which were ranked thus--

[4.69360412 1.55860227 0.90968005 0.82521592 0.67129716 0.46186862]

Feature scaling was applied for a few reasons. Data which has a broad range and many outliers can be less tractable for classifiers which aim to classify points based on the Euclidean distance between their features because of the difficulties associated with comparing points across broad ranges. As mentioned in the sklearn documentation, motivations for scaling variables "include robustness to very small standard deviations of features and preserving zero entries in sparse data." Many of the Enron data were unevenly distributed and admitted of orders of magnitude differences in range. The naive Bayes algorithm featured a standard scaler, by which data points are arranged according to their unit variance.

Select k best chooses the most informative variables according to their ANOVA F-values, a measure of between-variable variance. The F-value's numerator is the variance of the means between samples, and gets larger the more the variances diverge. The denominator is the average of sample variances in each group within the data set, and is a proxy estimate of the overall population variance. The more the means converge, the closer the ratio will be to one. One the other hand, should the means diverge, the greater the F-value will be. This is then compared with a critical statistic from an F-table, from which a p-value will be given. ANOVA F-value was used in each use of SelectKbest.

With the above hyper-parameters, namely PCA, SelectKBest, and feature scaling, the model obtained the following results.

Feature Ranking:

feature no. 1: salary (True)

feature no. 2: total_stock_value (True)

feature no. 3: shared_receipt_with_poi (True)

The naive bayes' selected features are:

['salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'shared_receipt_with_poi', 'to_poi_message_ratio']

Explained variance along x number of axes:

[4.69360412 1.55860227 0.90968005 0.82521592 0.67129716 0.46186862]

GNB report: Pipeline(steps=[('feature_scaling', StandardScaler(copy=True, with_mean=True, with_std=True)), ('skb', SelectKBest(k=10, score_func=<function f_classif at 0x1097f6c08>)), ('reduce_dim', PCA(copy=True, n_components=6, whiten=False)), ('GNB', GaussianNB())])

Accuracy: 0.83053 Precision: 0.34961 Recall: 0.31500 F1: 0.33140 F2: 0.32136

Total predictions: 15000 True positives: 630 False positives: 1172 False negatives: 1370 True negatives: 11828

While these met the passing project specifications, other algorithms were nevertheless explored, and are explained below.

Decision Tree

A decision tree was then fit on the data. A common method in statistical learning, decision tree learning is a schematic representation of feature variables onto a target variable. Within the context of classification problems, a decision tree's leaves represent class labels, and the branches which map onto the leaf represent the features of which the leaf, or class label, is comprised. Each node in a decision tree can be thought of as representing a "'test' on an attribute (e.g. whether a coin flip comes up heads or tails)." The branches stemming from the

node represent the computation and ‘decision’ taken by the ‘test.’ The paths from root to leaf represents classification rules.” Some advantages of decision trees include their relative applicability to many types of data. Indeed, decision tree learning has a reputation as a basic, “off-the-shelf procedure for data mining.” Disadvantages may include their potential sensitivity to minor changes in the data. Examples exist of relatively minor perturbations resulting in far-reaching changes in the decision trees’ structures. Additionally, decision trees have a relative tendency to overfit. Simon Bryne’s explanation on StackOverflow.com of decision trees’ relative shortcomings yields more insight into these particular problems.

“Some of these are related to the problem of multicollinearity: when two variables both explain the same thing, a decision tree will greedily choose the best one, whereas many other methods will use them both. Ensemble methods such as random forests can negate this to a certain extent, but you lose the ease of understanding.

However the biggest problem, from my point of view at least, is the lack of a principled probabilistic framework. Many other methods have things like confidence intervals, posterior distributions etc., which give us some idea of how good a model is. A decision tree is ultimately an ad hoc heuristic, which can still be very useful (they are excellent for finding the sources of bugs in data processing), but there is the danger of people treating the output as "the" correct model (from my experience, this happens a lot in marketing).”

As during Naive Bayes classifier computation, a scaling pipe was chained into GridSearchCV for the decision tree. However, the variables were scaled on a [0,1] minimum, maximum scale. This owes to PCA’s preference for standardized scaling to minimum, maximum scaling, as it is “interested in the components that maximize the variance.” Because decision tree did not include a principal component analysis, unlike the Naive Bayes, the variables were scaled on a [0,1] minimum, maximum scale instead of a unit variance measure.

Finally, Select K best was piped into the decision tree. The following results were found.

The decision tree's selected features are:

['bonus', 'total_stock_value', 'exercised_stock_options']

dtc report: Pipeline(steps=[('scaling', MinMaxScaler(copy=True, feature_range=(0, 1))), ('kbest', SelectKBest(k=3, score_func=<function f_classif at 0x1097f6c08>)), ('DTC', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,

max_features=None, max_leaf_nodes=None, min_samples_leaf=1,

min_samples_split=2, min_weight_fraction_leaf=0.0,

presort=False, random_state=None, splitter='best'))])

Accuracy: 0.80387 Precision: 0.28355 Recall: 0.30850 F1: 0.29550 F2: 0.30316

Total predictions: 15000 True positives: 617 False positives: 1559 False negatives: 1383 True negatives: 11441

The decision tree performed well, reaching the cusp of the required precision threshold and narrowly exceeding that of recall, but not as well as the naive Bayes model.

Random Forest

A random forest was then trained on the data. In classification tasks, random forests construct a number of decision trees and, during training, gather information on the categories to which each leaf belongs. Random forests are 'random' because trees within the forest are comprised of random subsets of the parameters and data available. These are generated by bootstrap aggregation, or bagging. Discussion on bagging is included in the references. Random forests category search over these subsets within trees and, in classification tasks, return information about the modal category for each split.

Random forests, as an aggregation of decision trees, exhibit similar behavior to other tree methods. However, they retain some of decision trees' strengths while mitigating other of their limitations. Random forests, like decision trees, are 'invariant under scaling and various other transformations of feature values, robust to inclusion of irrelevant features, and produce

inspectable models.’ Random forests, however, do not share decision trees’ tendencies to overfit. They avoid this by aggregating the results of each tree within the forest, which has the effect of reducing variance by averaging the errors across a number of different trees. As with all models with bias and variance components, there is an increased bias tradeoff to random forests, as opposed to decision trees, but this is slight, and random forests generally outperform decision trees.

For our example, select k best was piped into the random forest, with the following results obtained.

The random forest' selected features are:

['salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'shared_receipt_with_poi', 'to_poi_message_ratio']

feature importances [0.08263507 0.10763161 0.12553877 0.0664408 0.08813773 0.14647207

0.05811712 0.08806999 0.10434386 0.13261297]

Feature Ranking:

feature no. 1: restricted_stock_deferred (0.146472068135)

feature no. 2: exercised_stock_options (0.132612974171)

feature no. 3: total_payments (0.125538771499)

rf report: Pipeline(steps=[('skb', SelectKBest(k=10, score_func=<function f_classif at 0x1097f6c08>)), ('rf', RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',

max_depth=None, max_features='auto', max_leaf_nodes=None,

min_samples_leaf=1, min_samples_split=2,

min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,

oob_score=False, random_state=None, verbose=0,

warm_start=False))])

Accuracy: 0.86433 Precision: 0.47504 Recall: 0.16650 F1: 0.24658 F2: 0.19136

Total predictions: 15000 True positives: 333 False positives: 368 False negatives: 1667 True negatives: 12632

The Random forest achieved a high precision score, exceeding the required 0.3, but achieved a recall of 0.16650.

Support Vector Machine

The final trained classifier was a support vector machine. Support vector machines ('svm') are a class of algorithms which, as the name implies, treat data points as 'support vectors.' In the context of supervised learning problems, that is, in a classification task where the labels in the training data are known, support vectors act as points in a space around which a hyperplane is constructed. During training, vectorized transformations of the points serve to delineate boundaries in a dimensional search space. Once training is finished, a hyperplane is constructed which separates the data points into the categories specified in training. In low-dimensional spaces, one can visualize this by a line or plane with support vectors on either side. The hyperplane and hyperspace are numerically and symbolically represented in higher-dimensional spaces, but can be harder to visualize than in spaces with fewer dimensions. Additionally, svm has a cost parameter which governs a support vector's influence on the plane, and kerneling, a function which affects the transformation of the data points into support vectors.

The support vector classifier's selected features are:

['salary', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'restricted_stock', 'shared_receipt_with_poi', 'to_poi_message_ratio']

Pipeline(steps=[('skb', SelectKBest(k=10, score_func=<function f_classif at 0x1097f6c08>)), ('feature_scaling', MinMaxScaler(copy=True, feature_range=(0, 1))), ('svm', SVC(C=10000.0, cache_size=2500, class_weight=None, coef0=0.0,

decision_function_shape=None, degree=3, gamma=0.001, kernel='linear',

max_iter=-1, probability=False, random_state=None, shrinking=True,

tol=0.001, verbose=False))])

Accuracy: 0.87333 Precision: 0.57886 Recall: 0.18350 F1: 0.27866 F2: 0.21253

Total predictions: 15000 True positives: 367 False positives: 267 False negatives: 1633 True negatives: 12733

It performed admirably, reaching a high precision, but did not pass, owing to a low recall score.

Conclusion

Machine learning uses statistical methods and operates on the large computing power available in modern computers to derive insight from data. Its utility resides in its scale, adaptability, and precision. Machine learning models are suited to problems demanding a large number of rules, or when such rules which govern the solving of these problems overlap. Their ability to apply a multitude of calculations and rules allows them to outperform simpler statistical or human ability in understanding complex data.

For this project, four classifiers were trained on data from the Enron fraud case. The naive Bayes' algorithm returned the best score on the cross-validated data set. The features which seemed to matter most across all of the classifiers included the 'salary' information, as well as stock information.

Portions of the the project are not exhaustive of all data exploration possibilities. This report tested and cross-validated the data, but more data exploration may be merited. More parameter tuning may have given the models a broader scope of possibilities to explore, though with more computational expense. More kerneling methods could have been explored in the support vector classifier, however these are computationally expensive as well. Another scoring function to replace the 'F1' score may offer another appraisal for which the models perform better. While this project made use of four machine learning models, others, such as artificial neural networks, could have been tested.

References

https://github.com/ryancheunggit/Udacity/tree/master/P5

https://github.com/bhurn/enron-poi-identifier

https://github.com/saraheshuang/Enron_email_Detection

https://github.com/jihotahk/udacity-enron

ftp://131.252.97.79/Transfer/Treg/WFRE_Articles/Liaw_02_Classification%20and%20regression%20by%20randomForest.pdf

http://www.eecs.wsu.edu/~holder/courses/CptS570/fall07/papers/Dietterich00.pdf

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

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://en.wikipedia.org/wiki/Hyperparameter

http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

http://scikit-learn.org/stable/modules/grid_search.html

http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

http://stats.stackexchange.com/questions/12398/how-to-interpret-f-and-p-value-in-anova

http://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

https://en.wikipedia.org/wiki/Decision_tree_learning

http://stats.stackexchange.com/questions/1292/what-is-the-weak-side-of-decision-trees

http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-min-max-scaling

https://en.wikipedia.org/wiki/Random_forest#Tree_bagging

https://en.wikipedia.org/wiki/Support_vector_machine