Udacity.com /ud120 / Intro to Machine Learning / Pattern Recognition for Fun and Profit

www.udacity.com/course/intro-to-machine-learning--ud120

FINAL PROJECT

**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?**

In 2002, Enron, one of the largest companies in the United States, had collapsed into bankruptcy due to widespread corporate fraud. The dataset was originally made public by the Federal Energy Regulatory Commission during its investigation.

The goal was to perform data analysis on Enron dataset to identify Enron employees, who may have committed fraud. I put my machine learning skills to build, tune, evaluate the best performing algorithm, by trying various combinations of supervised classification algorithms and parameters, engineer the features, draw conclusion to make sense of the analysis process and communicating it to others.

The provided dataset contains data of 146 users, with financial and email information organized into 21 features for each. It is not a large dataset to get very clear analysis for a machine learning. After I explored the combined email and finance dataset for outliers, these data points were removed:

LAY KENNETH L - outlier

TOTAL - not a person

THE TRAVEL AGENCY IN THE PARK' - not a person

**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.**

\*\*\* Tested with hand selected features:

1. I calculated new features: "fraction\_from\_poi" & "fraction\_to\_poi" & removed the features used to calculate these new ones.

2. Removed salary, deferral\_payments, bonus, expenses, loan\_advances, other, deferred\_income, long\_term\_incentive features, since total\_payments are sum of these

3. Removed exercised\_stock\_options, restricted\_stock since the total\_stock\_value is sum of these

\*\*\* The final POI identifier uses the automated feature selection:

1. Scaling features with MinMaxScaler. The estimator would not be able to learn from new features correctly as expected, since these are orders of magnitude smaller that others.

2. Using Principal component analysis PCA. Linear dimensionality reduction and keeping only the most significant singular vectors to project the data to a lower dimensional space.

3. Using SelectKBest. Select features according to the k highest scores.

4. Using FeatureUnion to build estimator from PCA and SelectKBest selection. Concatenates results of multiple transformer objects. This estimator applies a list of transformer objects in parallel to the input data, then concatenates the results. This is useful to combine several feature extraction mechanisms into a single transformer.

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

\*\*\* Algorithm tested:

\* SVC(C=10000, cache\_size=200, class\_weight=None, coef0=0.0, decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf', max\_iter=-1, probability=False, random\_state=None, shrinking=True,tol=0.001, verbose=False)

Metrics received with cross\_validation.KFold:

SVC average accuracy: 0.781

SVC average recall\_score: 0.278

SVC average precision\_score: 0.214

SVC average f1\_score: 0.229

Metrics received with cross\_validation.rain\_test\_split:

SVC accuracy: 0.86

SVC recall\_score: 0.0

SVC precision\_score: 0.0

SVC f1\_score: 0.0

\* DecisionTreeClassifier with GridSearchCV(cv=None, error\_score='raise', estimator = DecisionTreeClassifier(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, presort=False, random\_state=42, splitter='best'),

fit\_params={}, iid=True, n\_jobs=1,

param\_grid={'min\_samples\_split': [2, 3], 'criterion': ['gini', 'entropy']},

pre\_dispatch='2\*n\_jobs', refit=True, scoring=None, verbose=0)

Metrics received with cross\_validation.KFold:

Decision Tree average accuracy: 0.803

Decision Tree average recall\_score: 0.486

Decision Tree average precision\_score: 0.344

Decision Tree average f1\_score: 0.392

Metrics received with cross\_validation.rain\_test\_split:

Decision Tree accuracy: 0.86

Decision Tree recall\_score: 0.571

Decision Tree precision\_score: 0.5

Decision Tree f1\_score: 0.533

**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? (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).**

\* GridSearchCV - Exhaustive search over specified parameter values for an estimator. It implements a “fit”, “score”, “predict” methods. The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

Best parameters using cross\_validation.KFold:

clf.best\_params\_ : {'min\_samples\_split': 3, 'criterion': 'gini'}

Best parameters using cross\_validation.train\_test\_split:

clf.best\_params\_ : {'min\_samples\_split': 2, 'criterion': 'gini'}

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

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data (overfitting). To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set.

\*\*\* Used KFold to divide all the samples in various k groups of samples of equal sizes. The prediction function is learned using k - 1 folds, and the fold left out is used for test.

folds = 6

Decision Tree average accuracy: 0.803

Decision Tree average recall\_score: 0.486

Decision Tree average precision\_score: 0.344

Decision Tree average f1\_score: 0.392

folds = 4

Decision Tree average accuracy: 0.811

Decision Tree average recall\_score: 0.308

Decision Tree average precision\_score: 0.259

Decision Tree average f1\_score: 0.271

\*\*\* Used train\_test\_split to split features into random train and test subsets with different test set size, which gives estimate of performance on an independent dataset and serves as check of overfiting.

test\_size = 0.35:

Decision Tree accuracy: 0.86

Decision Tree recall\_score: 0.571

Decision Tree precision\_score: 0.5

Decision Tree f1\_score: 0.533

test\_size = 0.25:

Decision Tree accuracy: 0.722

Decision Tree recall\_score: 0.4

Decision Tree precision\_score: 0.222

Decision Tree f1\_score: 0.286

**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.**

\* The accuracy\_score function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.

\* The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1.

\* The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1.

\* The F1 score can be interpreted as a weighted average of the precision and recall. Best value is 1. The relative contribution of precision and recall to the F1 score are equal. The formula is: F1 = 2 \* (precision \* recall) / (precision + recall)