UDACITY

Logo

PROJECT

Identify Fraud from Enron Email

A part of the Data Analyst Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

You have properly implemented all previous critical suggestions. Congratulations, you have successfully passed all specifications in this project.

Quality of Code

✓ **Code reflects the description in the answers to questions in the writeup. i.e. code performs the functions documented in the writeup and the writeup clearly specifies the final analysis strategy.**

✓ **poi_id.py can be run to export the dataset, list of features and algorithm, so that the final algorithm can be checked easily using tester.py.**

`poi_id.py` can be properly run, good job.

Understanding the Dataset and Question

✓ **Student response addresses the most important characteristics of the dataset and uses these characteristics to inform their analysis. Important characteristics include:**

- **total number of data points**
- **allocation across classes (POI/non-POI)**
- **number of features used**
- **are there features with many missing values? etc.**

Good job including most of the important characteristics of the dataset. We suggest including some comments on features containing many missing values to be included in the response to pass this specification. One way this can be presented is by including the number of NaN data points each feature has. This characteristic is important to help you decide how to treat NaN values (e.g. convert them to mean or 0).

✓ **Student response identifies outlier(s) in the financial data, and explains how they are removed or otherwise handled.**

Good job removing the three most prominent outliers in this project.

Optimize Feature Selection/Engineering

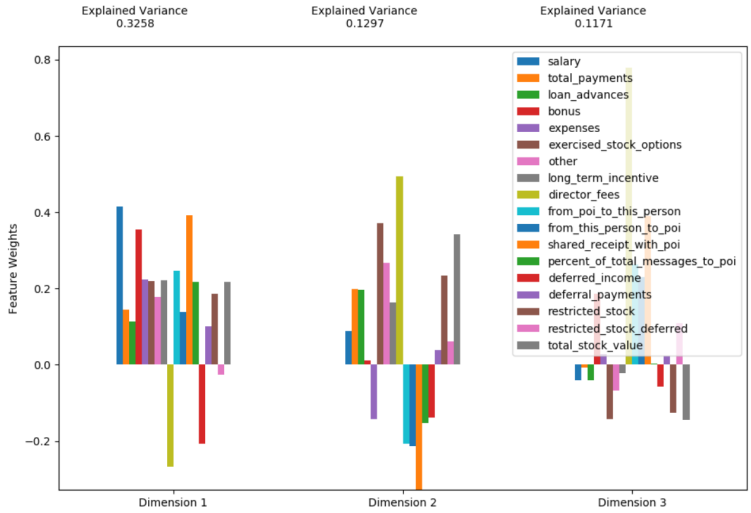
✓ **At least one new feature is implemented. Justification for that feature is provided in the written response. The effect of that feature on final algorithm performance is tested or its strength is compared to other features in feature selection. The student is not required to include their new feature in their final feature set.**

Good job creating new features, justifying their use, and measuring their effect on the final algorithm by testing its performance on the test dataset.

✓ **Univariate or recursive feature selection is deployed, or features are selected by hand (different combinations of features are attempted, and the performance is documented for each one). Features that are selected are reported and the number of features selected is justified. For an algorithm that supports getting the feature importances (e.g. decision tree) or feature scores (e.g. SelectKBest), those are documented as well.**

Your description of the feature selection process indicates that you have a good understanding of the techniques involved. Well done. Since you are using PCA, we recommend that explained variance ratio of PCA components are reported, along with how important the variables that form each component.

(Optional) Better yet, try to create a visualization to display the scores. Here is an idea for one such visualization:



Try to come up with your own visualization so you may learn how to do it for future projects, or otherwise use this function:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

def pca_results(features_list, pca):
    '''
    Create a DataFrame of the PCA results
    Includes dimension feature weights and explained variance
    Visualizes the PCA results
    Usage: `pca_results(features_list[1:], pca_object)`
    '''

    # Dimension indexing
    dimensions = dimensions = ['Dimension {}'.format(i) for i in range(1,
len(pca.components_)+1)]

    # PCA components
    components = pd.DataFrame(np.round(pca.components_, 4), columns = fea
tures_list)
    components.index = dimensions

    # PCA explained variance
    ratios = pca.explained_variance_ratio_.reshape(len(pca.components_),
1)
    variance_ratios = pd.DataFrame(np.round(ratios, 4), columns = ['Expla
ined Variance'])
    variance_ratios.index = dimensions

    # Create a bar plot visualization
```

```
    plt.figure(figsize=(16,8))
    ax = plt.subplot2grid((10,8), (0, 0), colspan=7, rowspan=8)

    # Plot the feature weights as a function of the components
    components.plot(ax = ax, kind = 'bar');
    ax.set_ylabel("Feature Weights")
    ax.set_xticklabels(dimensions, rotation=0)
    plt.legend(bbox_to_anchor=(1.02, 1), loc=2, borderaxespad=0.)


    # Display the explained variance ratios
    for i, ev in enumerate(pca.explained_variance_ratio_):
        ax.text(i-0.40, ax.get_ylim()[1] + 0.05, "Exp. Var.\n%.4f"%(ev))

    plt.show()
    # Return a concatenated DataFrame
    return pd.concat([variance_ratios, components], axis = 1)
```

Note: This visualization code was from another Nanodegree program: "Machine Learning Nanodegree". If you are interested, might be a good idea to look into it after you have graduated from this Data Analysis program.

✓ **If algorithm calls for scaled features, feature scaling is deployed.**

PCA *may* require feature scaling as it utilizes Euclidean distance in its use. I said *may* since there were reports that worse model may be produced instead when all the scaled variables on which PCA is performed are using the same unit (detailed explanation here). The conclusion from that experiment was to not scale features when all variables are using the same unit of measurements. Either way please incorporate this understanding in your project by trying to use feature scaling prior to PCA step, and report its effect to final algorithm's performance.

Pick and Tune an Algorithm

✓ **At least two different algorithms are attempted and their performance is compared, with the best performing one used in the final analysis.**

To use SVM algorithm correctly, one way that I found working was using StandardScaler with Pipeline. Try the following code:

```
svm_clf = Pipeline(steps=[
                    ('scaler', StandardScaler()),
                    ('classifier', SVC())
                    ])
svm_clf.fit(features_train,labels_train)
accuracy = svm_clf.score(features_test,labels_test)
predicted = svm_clf.predict(features_test)
output = test_classifier(svm_clf, my_dataset, features_list)
```

Somehow, by including scaler inside a pipeline (the recommended way with sklearn) SVC was able to perform (F1/precision/recall/accuracy score > 0). Removing that scaler would give "divide by zero" error.

✓ **Response addresses what it means to perform parameter tuning and why it is important.**

✓ **At least one important parameter tuned with at least 3 settings investigated systematically, or any of the following are true:**

- **GridSearchCV used for parameter tuning**
- **Several parameters tuned**
- **Parameter tuning incorporated into algorithm selection (i.e. parameters tuned for more than one algorithm, and best algorithm-tune combination selected for final analysis).**

Good job clarifying which parameters were tuned.

Validate and Evaluate

✓ **At least two appropriate metrics are used to evaluate algorithm performance (e.g. precision and recall), and the student articulates what those metrics measure in context of the project task.**

Response described a correct definition of precision and recall in the context of this project, well done.

✓ **Response addresses what validation is and why it is important.**

✓ **Performance of the final algorithm selected is assessed by splitting the data into training and testing sets or through the use of cross validation, noting the specific type of validation performed.**

✓ **When tester.py is used to evaluate performance, precision and recall are both at least 0.3.**

Running `tester.py` with produced pkl files got me the following result:

```
Pipeline(memory=None, steps=[('feature_transformer', MinMaxScaler(copy=True, feature_range=(0, 1))), ('reduce_dim', PCA(copy=True, iterated_power='auto', n_components=7, random_state=None, svd_solver='auto', tol=0.0, whiten=False)), ('clf', KNeighborsClassifier(algorithm='ball_tree', leaf_size=1, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=1, p=2, weights='uniform'))])
    Accuracy: 0.83547    Precision: 0.39126    Recall: 0.42100    F1: 0.40559    F2: 0.41470
    Total predictions: 15000    True positives:  842    False positives: 1310    False negatives: 1158    True negatives: 11690
```

Good job passing the required score.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ