Specification Change Response

Specification 1

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.

Required

The report should include the following key characteristics of the data:

1. The total number of data points
2. The number of POIs

Suggestion

The report could also include:

1. The number of non-POIs
2. The number of features used
3. Comments on any features with many missing values

I've included a more explicit section on important data characteristics.

Data Overview¶

Important Dataset Characteristics

The original data contains 146 observations with 21 features. Of these 146 observations, 128 observations were persons of interests (POI). With no missing value, 18 observations were non-POIs.

For our Machine Learning analyses, we focussed on three features: exercised_stock_options, bonus, and salary. We could have gone with more but we found out that additional features did not add much to the model, in terms of model performance.

"NaN" issue is an issue in this dataset. Only 4 features have values for more than 75% of observations. Six of them have less than 50% missing values, i.e. loan_advances, director_fees, restricted_stock_deferred, deferral_payments, deferred_income, and long_term_incentive. For this project, we assumed these "NaN" values to be zero for simplicity.

Specification 2

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.

Required

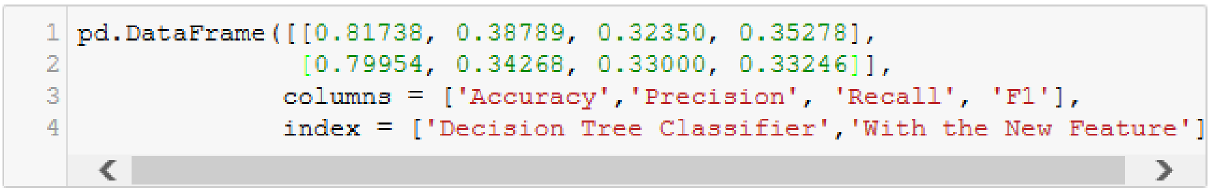
The report should mention at least one new feature that was created. Also, explain the effect of this feature on the final algorithm's performance. This can be done by training a simple classifier with and without the new features. The feature importance scores of all the features (new and existing) could also be provided to show the strength of the new features.

New Feature: Stock-Salary Ratio

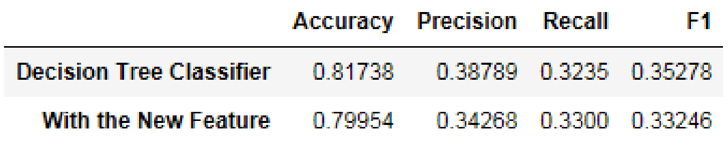
We created a stock-salary ratio with the thought that a high stock-salary ratio would be an indication that a person may know about the wrong-doings of the company that they may buy or sell their stocks at the right time.

However, with a score of `0.117`, this ratio was not one of the features with high SelectKBest scores. In fact, the ratio ranked 19th (best) among all features.

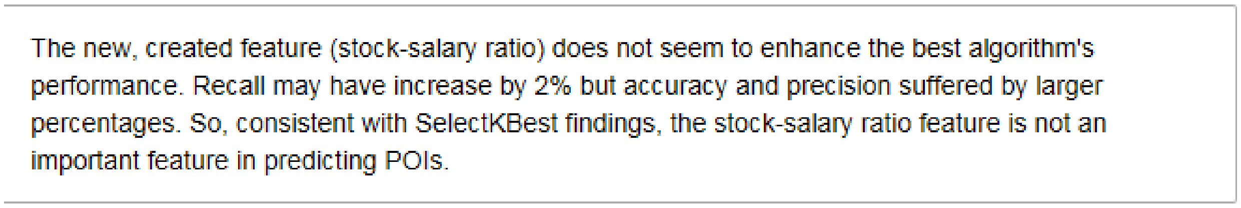
I tested my selected best model with and without the created, new featured. Their algorithm performance, as pasted in below, had mixed results. The new, created feature (stock-salary ratio) does not seem to enhance the best algorithm's performance. Recall may have increase by 2% but accuracy and precision suffered by larger percentages. So, consistent with SelectKBest findings, the stock-salary ratio feature is not an important feature in predicting POIs.



```
pd.DataFrame([[0.81738, 0.38789, 0.32350, 0.35278],
              [0.79954, 0.34268, 0.33000, 0.33246]],
             columns = ['Accuracy','Precision', 'Recall', 'F1'],
             index = ['Decision Tree Classifier','With the New Feature']
```



	Accuracy	Precision	Recall	F1
Decision Tree Classifier	0.81738	0.38789	0.3235	0.35278
With the New Feature	0.79954	0.34268	0.3300	0.33246



The new, created feature (stock-salary ratio) does not seem to enhance the best algorithm's performance. Recall may have increase by 2% but accuracy and precision suffered by larger percentages. So, consistent with SelectKBest findings, the stock-salary ratio feature is not an important feature in predicting POIs.

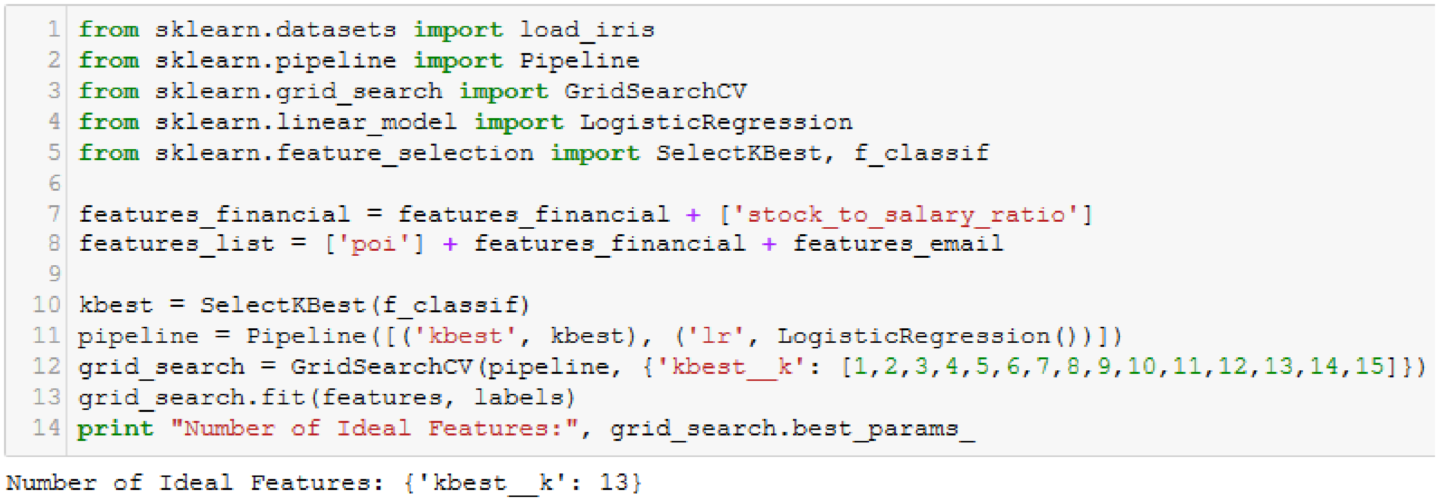
Specification 3

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.

Required

The report mentions that the top 3 features were chosen because their scores were higher than the rest. While this is a good start, remember that the feature selection process should be exhaustive. If the features were manually chosen, provide the performance achieved with several combinations. This way, we can determine the combination that gives the best results. Alternatively, you could use GridSearchCV to tune for **k** from SelectKBest.

Thank you for this suggestion. We employed this SelectKBest plus GridSearchCV approach. The resulting ideal number of features was 13.



```
from sklearn.datasets import load_iris
from sklearn.pipeline import Pipeline
from sklearn.grid_search import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest, f_classif

features_financial = features_financial + ['stock_to_salary_ratio']
features_list = ['poi'] + features_financial + features_email

kbest = SelectKBest(f_classif)
pipeline = Pipeline([('kbest', kbest), ('lr', LogisticRegression())])
grid_search = GridSearchCV(pipeline, {'kbest__k': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]})
grid_search.fit(features, labels)
print "Number of Ideal Features:", grid_search.best_params_
```

```
Number of Ideal Features: {'kbest__k': 13}
```

We decided to stick to the 3 features due to likely correlation (e.g. 'exercised_stock_options' and 'total_stock_value') and an error message, i.e. "The least populated class in y has only 1 member, which is too few. The minimum number of labels for any class cannot be less than 2." If given more time, we would consider having more than 3 features.

Specification 4

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.

Required

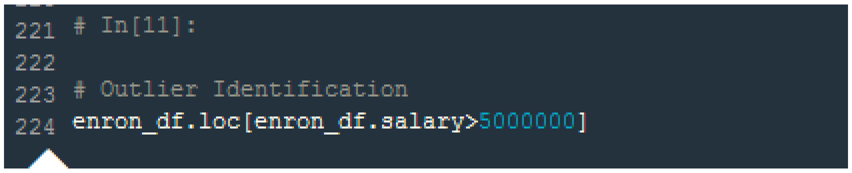
The report mentions that the `test_classifier()` function from the `tester.py` file was used. However, please be specific about the type of validation method used inside this function and explain why it is the preferred validation method for this project.

I elaborated my answer in the short question's response (item 5):

> In the project, we run the tester.py algorithm to validate our findings. The test_classifier function indicates that the performance metrics were calculated based on 1,000 runs with randomized train-test subsamples – all using the same Enron dataset. Based on tabulations of true_negatives, false_negatives, false_positives, and true_positives from these 1,000 runs, the tester.py program calculates accuracy, precision, f1, and f2 performance metrics. This method is preferred because it averages out performance metrics from a huge number of run. A train-test data sampling may be biased that we may end up with a train sample with zero POIs. This bias should be minimized with 1,000 runs with randomized train-test subsample data shuffling.

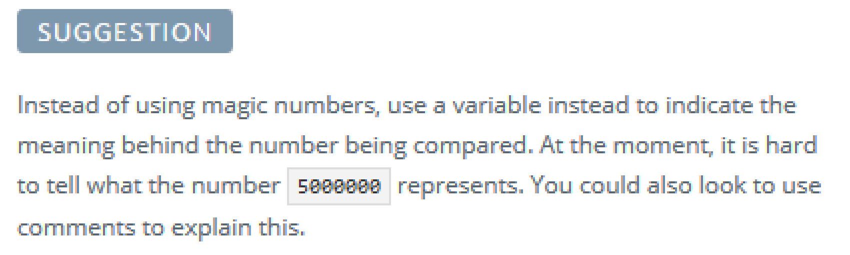
Core Review Response

Suggestion 1: Magic Number



```
# In[11]:

# Outlier Identification
enron_df.loc[enron_df.salary>5000000]
```



SUGGESTION

Instead of using magic numbers, use a variable instead to indicate the meaning behind the number being compared. At the moment, it is hard to tell what the number `5000000` represents. You could also look to use comments to explain this.

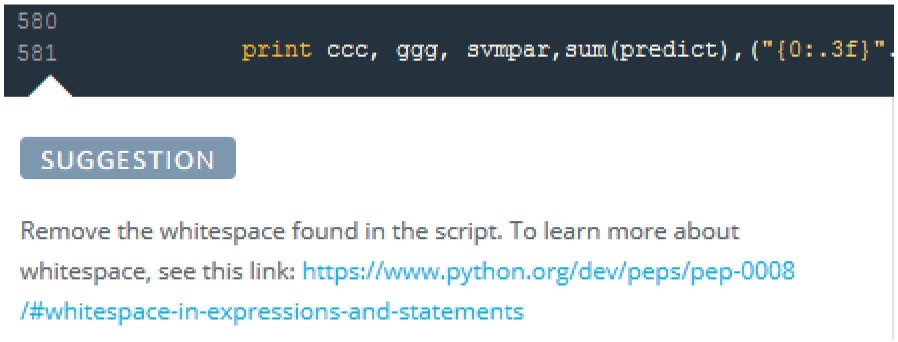
Comment 1 Change: Print observations with values greater than 3 standard deviation value

```
# Outlier Identification
# OLD: enron_df.loc[enron_df.salary>5000000]

enron_df.loc[enron_df.salary > \
            enron_df.salary.mean() + (3 * enron_df.salary.std())]

# enron_df.loc['THE TRAVEL AGENCY IN THE PARK', :]
```

<u>Suggestion 2: Whitespace</u>



```
            print ccc, ggg, svmpar,sum(predict),("{0:.3f}".
```

SUGGESTION

Remove the whitespace found in the script. To learn more about whitespace, see this link: https://www.python.org/dev/peps/pep-0008/#whitespace-in-expressions-and-statements

Thank you for the suggestion. I will keep trying to apply a sound discipline in my programming. For this particular example, I opted to have a line continuation with a backslash “\”.

```
print ccc, ggg, svmpar,sum(predict),("{0:.3f}".format(acc)), \
("{0:.3f}".format(prec)), ("{0:.3f}".format(recall)),"\n"
```