PROJECT SPECIFICATION

Identify Fraud from Enron Email

Quality of Code

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
Functionality	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.	The code performs the functions that is documented in this report. It will also clearly specify the analysis strategy used to achieve the intended performance i.e. **precision>0.3** and **recall >0.3**.
Usability	*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*.	*tester.py* has been imported to *poi_id.py* using the import function to allow the former script to be called every time the latter script is called. This eases the verification process to see if I have achieved the intended performance.

Understanding the Dataset and Question

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
Data Exploration	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.	Data exploration is used to understand the available features. I have chosen the ones I deem most relevant in our investigation. In any fraud cases, it involves money. I am concerned the means money can be siphoned out, usually in more than 1 method either in direct liquid form or otherwise i.e. s***alary*** in the direct, liquid form on monthly basis; ***bonus*** in the direct, liquid form on yearly basis; ***total stock value*** in the indirect, non-liquid form where the stocks have to be sold in order to turn it liquid. • total number of data points =146 • allocation across classes (POI/non-POI) =POI:18, Non-POI:128 • number of features used =4[***poi***, ***salary***, ***bonus***, ***total_stock_value]*** • are there features with many missing values? Out of the 4 used features, ***bonus*** has the highest missing values (64/146) followed by ***salary*** (51), ***total_stock_value*** (20) and ***poi*** (0). In the total 21 available features, ***loan_advances*** has the highest missing values (142/146) followed by ***director_fees*** (129), ***restricted_stock_deferred*** (128) and ***deferral_payments*** (107).
Outlier Investigation	Student response identifies outlier(s) in the financial data, and explains how they are removed or otherwise handled.	Outlier Investigation is important especially when it coming from different sources for different classes as it can easily lead to bias/ mistakes. Outlier can be detected using several methods via: a) exploratory analysis. Using this method, I have removed **TOTAL** from ***data_dict***, as salary $26,704,229 is way far from average; b) manual checking. Using this method, I have removed i) **THE TRAVEL AGENCY IN THE PARK** from ***data_dict***, as it is not a name of a person ii) **LOCKHART EUGENE E** from ***data_dict***, as no value is available for its 20 features except for ***poi***. Now, total number of data points =143.

Optimize Feature Selection/Engineering

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
Create new features	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	Added two new features: a) ***fraction_from_poi***, which is defined as the fraction of ***from_poi_to_this_person*** and ***to_messages***; b) ***fraction_to_poi***, which is defined as the fraction of ***from_this_person_to_poi*** and ***from_messages***. This two new features were added because of the hunch that POI would write and

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
	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.	receive much more emails than non-POI. Keen to study if this feature could assist to identify more new POI. Now, number of features used=6.
Intelligently select features	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.	Feature selection a) Manual selection by hand Difficult to address bias ~ variance dilemma if done manually. Different combinations of features are attempted. There is no intelligence in selecting the feature, the process: human intuition -> code up the feature -> visualise -> repeat. The performance is documented for each one. Using clf = DecisionTreeClassifier() i) Features List= ['poi', 'salary', 'bonus', 'total_stock_value'] Features Importance= [0.21513238, 0.47434166, 0.31052596] ii) Features List= ['poi', 'salary', 'bonus', 'total_stock_value', 'fraction_from_poi', 'fraction_to_poi'] Features Importance= [0.11965812, 0.15306268, 0.49722959, 0.06410256, 0.16594705] b) Univariate (SelectPercentile(), SelectKBest()) i) selector = SelectPercentile(f_classif, percentile=10) Features List= ['poi', 'salary', 'bonus', 'total_stock_value'] Selector Scores= [11.18466395, 15.04137673, 16.4183221] selector = SelectKBest(f_classif, k=2) Features List= ['poi', 'salary', 'bonus', 'total_stock_value'] Selector Scores= [11.18466395, 15.04137673, 16.4183221] ii) selector = SelectPercentile(f_classif, percentile=10) Features List= ['poi', 'salary', 'bonus', 'total_stock_value', 'fraction_from_poi', 'fraction_to_poi'] Selector Scores= [11.1719400, 10.1229852, 19.5434702, 0.0120766361, 4.09931158] selector = SelectKBest(f_classif, k=2) Features List= ['poi', 'salary', 'bonus', 'total_stock_value', 'fraction_from_poi', 'fraction_to_poi'] Selector Scores= [11.1719400, 10.1229852, 19.5434702, 0.0120766361, 4.09931158] c) Recursive i) selector = RFE(clf, 2, step=1) Features List= ['poi', 'salary', 'bonus', 'total_stock_value'] Selector Scores n_features_= 2 Selector Scores support_= [True True False] Selector Scores ranking_= [1 1 2] ii) selector = RFE(clf, 2, step=1) Features List= ['poi', 'salary', 'bonus', 'total_stock_value', 'fraction_from_poi', 'fraction_to_poi'] Selector Scores n_features_= 2 Selector Scores support_= [False False True False True] Selector Scores ranking_= [3 2 1 4 1]
Properly scale features	If algorithm calls for scaled features, feature scaling is deployed.	The algorithms that are affected by feature rescaling are Support Vector Machine with RBF (supervised learning) and k-means

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
		clustering (unsupervised learning). As we used Support Vector Machine algorithm, we will rescale it using MinMax rescaler. Found no difference in Precision and Accuracy. data = featureFormat(my_dataset, features_list, sort_keys = True) from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler() data = scaler.fit_transform(data) labels, features = targetFeatureSplit(data) clf = SVC(kernel='rbf', C=10.0) Before MinMax rescaling: Accuracy: 0.85757 Precision: 0.54286 Recall: 0.01900 F1: 0.03671 F2: 0.02354 Total predictions: 14000 True positives: 38 False positives: 32 False negatives: 1962 True negatives: 11968 After MinMax rescaling: Accuracy: 0.85757 Precision: 0.54286 Recall: 0.01900 F1: 0.03671 F2: 0.02354 Total predictions: 14000 True positives: 38 False positives: 32 False negatives: 1962 True negatives: 11968

Pick and Tune an Algorithm

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
Pick an algorithm	At least two different algorithms are attempted and their performance is compared, with the best performing one used in the final analysis.	The problem involves supervised learning with non-ordered, labelled data. Hence chosen to evaluate with : NaiveBayes, Support Vector Machine, Decision Tree, K-Nearest Neighbour, AdaBoost, and Random Forest algorithms. The performance in compared using Precision, Recall, Accuracy, and F1 Score metrics.
Discuss parameter tuning and its importance	Response addresses what it means to perform parameter tuning and why it is important.	The goal of machine learning is to enable a system that can build models automatically from data without requiring much human intervention. However, one of the difficulties of learning algorithms, such as Support Vector Machine, Decision Trees and K Nearest Neighbour, require one to set its parameters before using it. Choosing the right parameters is important as it leads to an optimal model balancing bias-variance dilemma. In an initial learning phase (a.k.a. initial parameter tuning phase), training data are used to adjust the classification parameters where the model is tuned to achieve high quality results. Hence, parameter tuning for an algorithm or machine learning technique can be defined as an optimisation process leading to the algorithm to perform the best.
Tune the algorithm	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	In Support Vector Machine two parameters are investigated with at least 4 settings: kernel (linear, rbf, poly, sigmoid), C (1, 10, 100, 1000, 10000, 100000, 1000000). In Decision Tree and Random Forest, two parameters are investigated: criterion (gini, entropy), min_samples_split (2, 10, 20, 25, 30, 40, 50). In K-Nearest Neighbour three parameters are investigated with at least 2 settings: n_neighbors (2, 3, 4, 5, 6, 7, 8), algorithm (auto, brute, ball_tree, kd_tree), weights (uniform, distance). In AdaBoost three parameters are investigated with at least 2 settings: n_estimators (20, 30, 40, 50, 60, 70, 80), algorithm ('SAMME.R', 'SAMME'), learning_rate (0.80, 0.85, 0.90, 0.95, 1.0).

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
	algorithm-tune combination selected for final analysis).	Implemented NaiveBayes, Support Vector Machine, Decision Tree, K-Nearest Neighbour, AdaBoost, and Random Forest algorithms. The best algorithm-tune combination will be selected for final analysis. Did not implement GridSearchCV which generates candidates from a grid of parameter values as we have used list with multiple options for the same purpose.

Validate and Evaluate

CRITERIA	MEETS SPECIFICATIONS	RESPONSES FROM SARAVANAN KANDAMY
Usage of Evaluation Metrics	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.	Accuracy, Precision, Recall, and F1 are used as performance evaluation metrics for the evaluated machine learning algorithms. In the context of the project task, Accuracy is the ratio of correctly predicted observation to the total observations, mathematically defined as the ratio of sum of true positives and true negatives, to total predictions. Precision also known as positive predictive value is the ratio of correctly predicted positive observations to the total predicted positive observations, mathematically defined as the ratio of true positives to, the sum of true positives and false positives. Recall is also known as sensitivity, mathematically defined as the ratio of true positives with, the sum of true positives and false negatives. F1 is the weighted average of Precision and Recall, mathematically defined as twice the ratio of precision times recall with, sum of Precision and Recall.
Discuss validation and its importance	Response addresses what validation is and why it is important.	Validation is an important process where data is separated for training and testing purposes. In training phase, training dataset are used to learn the best approach for prediction and in test phase, the performance of the model(s) can be calculated using the test dataset. K-Fold cross validation is an example of validation algorithm. Validation allow us to avoid overfitting, where the training can be stopped when the error in a test dataset starts growing (before convergence).Plus in the validation process, we can estimate the performance of independent test data before using it for actual implementation.
Validation Strategy	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.	In this project, the train_test_split is used as a strategy to split the data in 70:30 ratio. The identified 30% test dataset is used later to predict using the model obtained from 70% training dataset.
Algorithm Performance	When tester.py is used to evaluate performance, precision and recall are both at least 0.3.	*tester.py* is used to evaluate the NaiveBayes, Support Vector Machine, Decision Tree, K-Nearest Neighbour, AdaBoost, and Random Forest algorithms with various combination of parameter values. Apart from Precision and Recall, Accuracy and F1 are evaluated. The results are shown in decreasing order below, where Precision and Recall are both above 0.3, #SVC(kernel='poly', C=1000000.0) Accuracy: 0.84279, Precision: 0.44942, Recall: 0.44650, F1: 0.44796 #SVC(kernel='rbf', C=1000000.0) Accuracy: 0.83293, Precision: 0.41823, Recall: 0.43350, F1: 0.42573 #SVC(kernel='rbf', C=100000.0) Accuracy: 0.82800, Precision: 0.38356, Recall: 0.33600, F1: 0.35821 #KNeighborsClassifier(n_neighbors=2, algorithm='kd_tree', weights ='distance')

Accuracy: 0.81800, Precision: 0.35173, Recall: 0.32500, F1: 0.33784
#KNeighborsClassifier(n_neighbors=2, algorithm='brute', weights ='distance')
Accuracy: 0.81800, Precision: 0.35173, Recall: 0.32500, F1: 0.33784
#KNeighborsClassifier(n_neighbors=2, algorithm='auto', weights ='distance')
Accuracy: 0.81800, Precision: 0.35173, Recall: 0.32500, F1: 0.33784
#DecisionTreeClassifier(criterion='entropy', min_samples_split=2)
Accuracy: 0.80814, Precision: 0.33268, Recall: 0.34100, F1: 0.33679

Hence, **Support Vector Machine using poly kernel with C=1,000,000** is the best machine learning algorithm for our data in this project. It achieves Accuracy of 84.3%, Precision of 44.9%, Recall of 44.7% and F1 score of 0.44796.