PA1 Report

Taewoo Kim
2021 3873
tkimae@connect.ust.hk

Computing Environment

Computer model: MacBook Pro(13-inch, 2017)
OS: macOS Mojave ver 10.14
Processor: 2.3 GHz Intel Core i5
Memory: 8GB 2133 MHz LPDDR3
Graphics: Intel Iris Plus Graphics 640 1536 MB

Time required by each task(in seconds)

Linear regression:

```
Overall running clock (process time) : 2.45
Overall running time (wall time) : 2.493
```

Logistic regression:

```
Overall running clock (process time) : 0.675
Overall running time (wall time) : 0.447
```

Neural network:

```
Overall running clock (process time) : 282.929
Overall running time (wall time) : 444.898
```

Logistic regression turned out to be the fastest learning algorithm, while neural network algorithm to be the slowest. Reason why linear regression algorithm took quite long time to run is that calculating squared error and plotting them took some time. And for neural network, cross validation procedure and drawing learning graph of each model took long time to execute.

Linear Regression

Linear regression is machine learning algorithm based on supervised learning, and it performs regression task.
With given training data, it estimates the parameter values of the linear equation, and enables us to predict the output on unlabeled data set.

$y=a_0+a_1x_1+a_2x_2+\cdots$

Compute the R^2 score of the linear regression model on both the training and test sets

Formula for R^2 score is as follows:

$$R^2 \text{ score (Variance score): } R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$$

In this formula y stands for expected output and $\hat{y}$ stands for predicted output and $\bar{y}$ stands for mean.
I have calculated variance score of data set using r2_score() function from sklearn.metrics library

Variance score for three data sets: fifa, finance, orbit

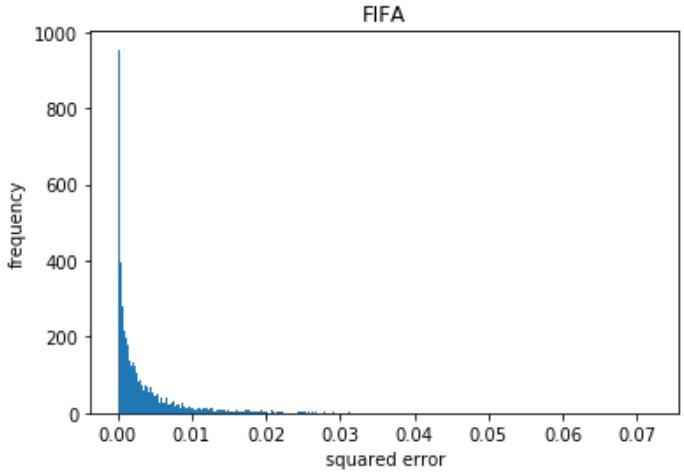
```
Variance score for fifa test set: 0.84
Variance score for fifa train set: 0.84
Variance score for finance test set: 1.00
Variance score for finance train set: 1.00
Variance score for orbits test set: 0.70
Variance score for orbits train set: 0.69
```

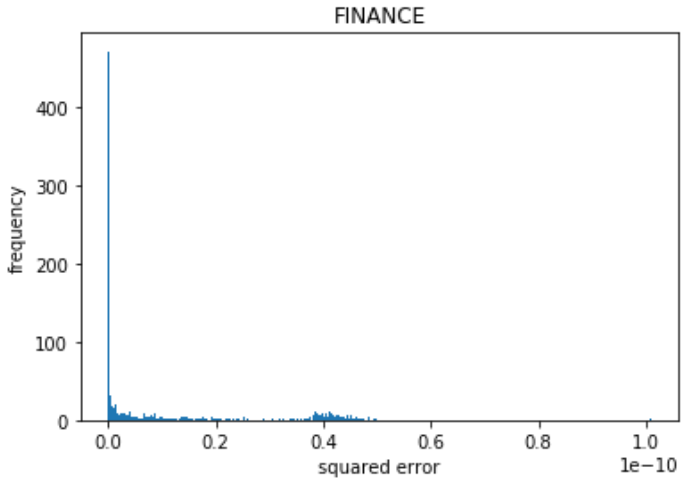
From these variance score recorded in 2 significant figure, I can rank the model in the following way from best to worst:

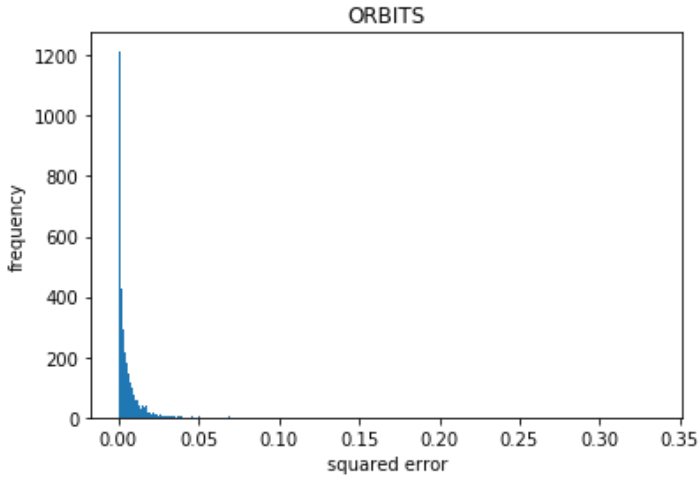
1. Finance
2. Fifa
3. Orbits

Finance set shows the variance score of 1.00 on both training and test set, which implies that the predicted data points are not far from expected output. Fifa data set shows variance score of 0.84, from which I can tell the predicted data points are not very far from the expected output, but some distance from expected output is expected. Moreover, third one which is orbits data set shows variance score of 0.70 on test data set 0.69 on train data set. This implies that predicted outputs are comparatively far from expected output. From this, I can conclude that ranking of the reliability of data set is as stated above

Depict a histogram of the squared errors of the data points in the test set of the linear regression model







Squared error shows how a predicted value is far from the actual value. Through the visualization by graph, we can see squared error on each test data set of finance is very small, compared to the other two datasets, which shows that this model performs the best among 3. Second best model is model on fifa dataset since frequency of squared error is more concentrated on smaller squared error value compared to the model on orbits dataset. Third best performed model will be orbits.

Logistic regression

Logistic regression solves the classification problems

Build the logistic regression model by adopting the gradient descent optimization algorithm, and present the model settings

I have built the logistic regression model using SGDClassifier() function from sklearn.linear_model library. I chose this function since it employs the stochastic gradient descent (SGD) learning. There are various hyperparameters to be set, and I mostly used default setting. Model setting that I have used is as follows:

```
Model Setting on fifa data set:
{'alpha': 0.0001, 'average': False, 'class_weight': None,
'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.06, 'fit_intercept':
True, 'l1_ratio': 0.15, 'learning_rate': 'optimal', 'loss': 'log',
'max_iter': 50, 'n_iter': None, 'n_iter_no_change': 5, 'n_jobs': None,
'penalty': 'l2', 'power_t': 0.5, 'random_state': None, 'shuffle': True,
'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 1, 'warm_start':
False}

Model Setting on finance data set:
{'alpha': 0.0001, 'average': False, 'class_weight': None,
'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.06, 'fit_intercept':
True, 'l1_ratio': 0.15, 'learning_rate': 'optimal', 'loss': 'log',
'max_iter': 50, 'n_iter': None, 'n_iter_no_change': 5, 'n_jobs': None,
'penalty': 'l2', 'power_t': 0.5, 'random_state': None, 'shuffle': True,
'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 1, 'warm_start':
False}

Model Setting on orbits data set:
{'alpha': 0.0001, 'average': False, 'class_weight': None,
'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.06, 'fit_intercept':
True, 'l1_ratio': 0.15, 'learning_rate': 'optimal', 'loss': 'log',
'max_iter': 50, 'n_iter': None, 'n_iter_no_change': 5, 'n_jobs': None,
'penalty': 'l2', 'power_t': 0.5, 'random_state': None, 'shuffle': True,
'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 1, 'warm_start':
False}
```

What I have changed from default setting is ‘eta0’ which implies the learning rate of the model. I have tried a few learning rate smaller than 1, and I turned out that 0.06 gave optimal result with highest mean accuracy score on models. Moreover, for loss parameter, I have used log since the ‘log’ loss gives logistic regression, a probabilistic classifier. Moreover, max_iter has been set to 50 to prevent iterating many times, and verbose was defined as 1 to enable verbosity.

These models were trained using fit(train_X_data, train_y_data) method provided in SGDClassifier. From this model, coefficients and intercept was obtained using coef_ and Intercept_ attributes on fitted SGDClassifier model.

Compute the accuracy of the logistic regression model on both the training and test sets

I have computed the accuracy of the logistic regression model on both training and test sets using score() method which is also provided in SGDClassifier. This method returns mean accuracy on given data and labels. Basically, it measures how the model performs on new data (test set)

accuracy(fraction of correct predictions) = correct predictions / total number of data points

Output on my program:

```
Mean Accuracy of fifa training set is 0.855204305966189 and test set is
0.8521719354105072
Mean Accuracy of finance training set is 0.8159041394335512 and test
set is 0.7973856209150327
Mean Accuracy of orbits training set is 0.9633893383115536 and test set
is 0.9595645412130638
```

I printed mean accuracy on both training and test set in order to validate whether there is overfitting happening, which can be seen if there is a quite large difference between training accuracy score and test accuracy score.

This results led me to rank model in this way:

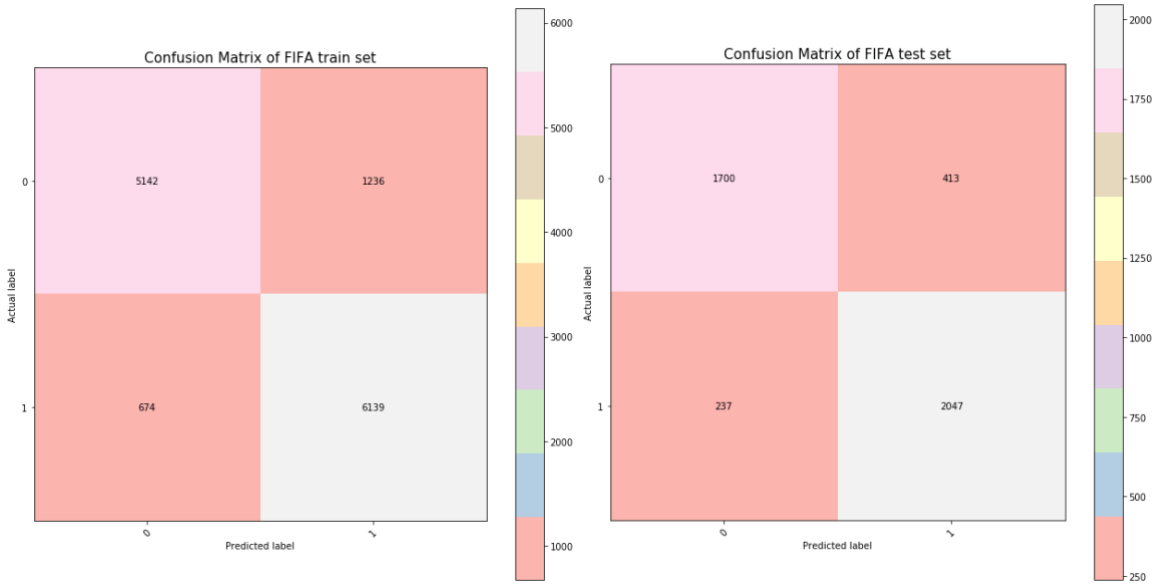
1. orbits
2. finance
3. fifa

For the model on orbits dataset, mean accuracy is 0.959 which shows the highest score among 3 of them and which is quite close to perfect score 1. We can deduce this model shows high precision. Secondly, model on fifa dataset, mean accuracy is 0.852. This is also not really far from the perfect score 1, but I can say it has relatively low precision compared to the model on orbits. Third one was model built with finance dataset. This showed 0.797 and we can conclude this model has the lowest precision among three.

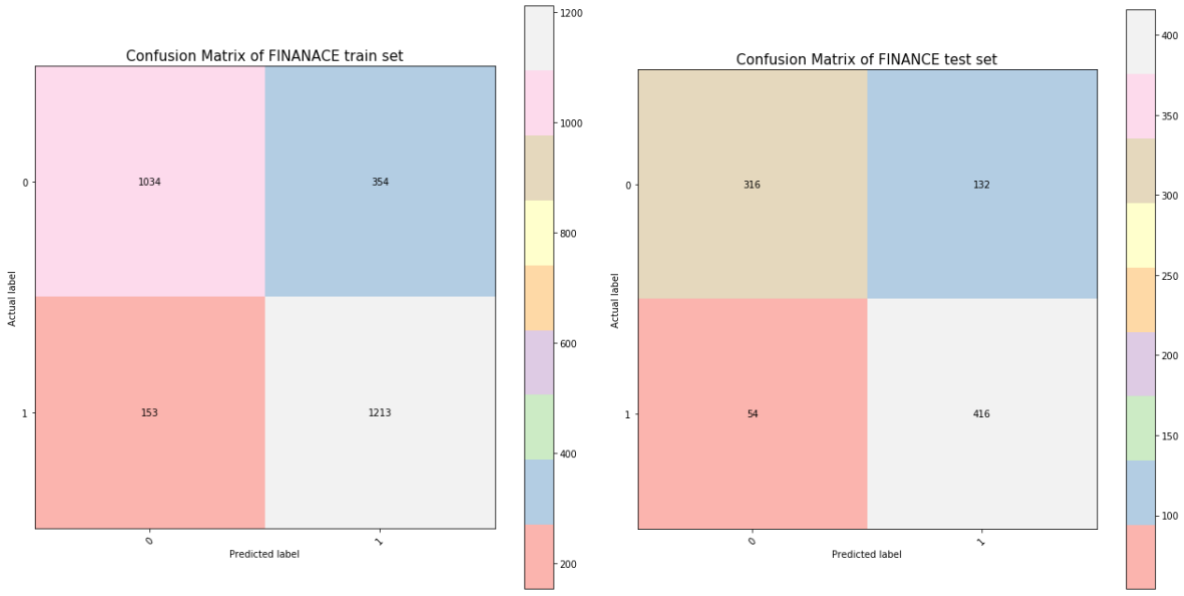
Record and visualize the experiment results of the logistic regression model on both the training and test sets

The result of experiment has been visualized using confusion_matrix from sklearn.metrics library. This is to visualize the experiment results and evaluate the accuracy of logistic regression model.

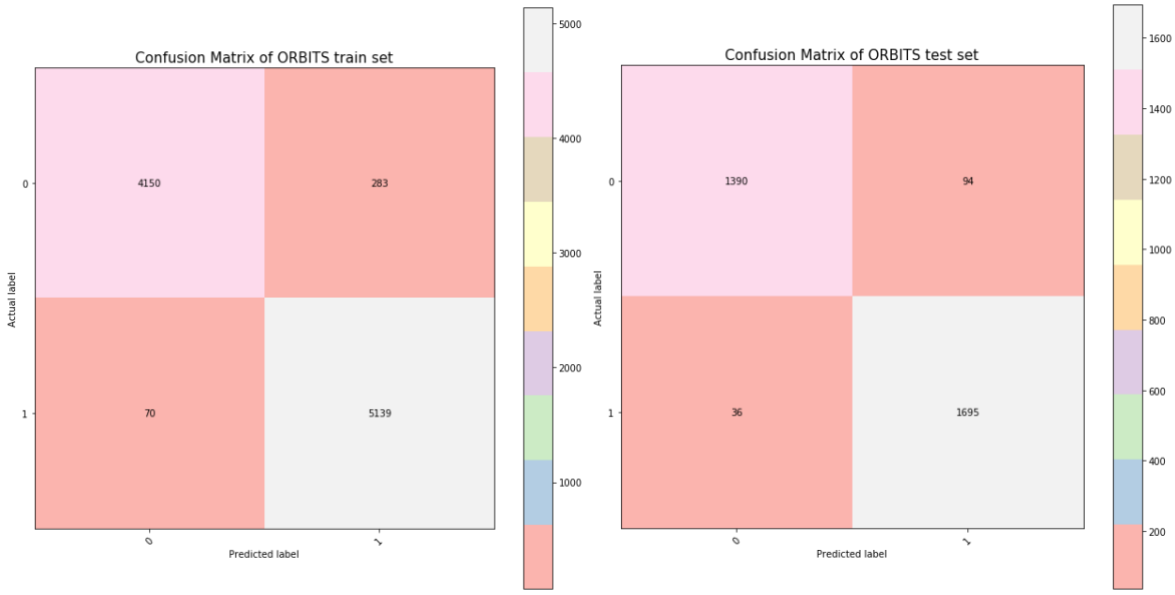
This is confusion matrix of fifa training and test set showed on output screen



This is confusion matrix of finance training and test set showed on output screen



This is confusion matrix of orbits training and test set showed on output screen



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 0.
Like this confusion matrix help us to visualize the result of classification.

Neural Network

Build the neural network model by adopting the gradient descent optimization algorithm, and present the model settings

Fifa:

```
Parameter setting for MLPClassifierfifa

{'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1':
0.9, 'beta_2': 0.999, 'early_stopping': True, 'epsilon': 1e-08,
'hidden_layer_sizes': (2,), 'learning_rate': 'constant',
'learning_rate_init': 0.001, 'max_iter': 300, 'momentum': 0.9,
'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5,
'random_state': 1, 'shuffle': True, 'solver': 'sgd', 'tol': 0.0001,
'validation_fraction': 0.1, 'verbose': True, 'warm_start': False}
```

Highlighted parameters are the parameters that I have changed from the default setting. Early stopping has been set to True to terminate training when validation score is not improving and prevent overfitting of the model. Hidden layer size is set to (2,0) which was proven to be the best during cross validation procedure. Max_iter has been set to 300 to limit the number of epoch, and random_state has been set to 1 to be used as a seed for random number generator. I have chosen sgd as a solver which employs stochastic gradient descent algorithm and verbose has been set to true to enable verbosity.

Finance:

```
Parameter setting for MLPClassifierfinance

{'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1':
0.9, 'beta_2': 0.999, 'early_stopping': True, 'epsilon': 1e-08,
'hidden_layer_sizes': (10,), 'learning_rate': 'constant',
'learning_rate_init': 0.001, 'max_iter': 300, 'momentum': 0.9,
'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5,
'random_state': 1, 'shuffle': True, 'solver': 'sgd', 'tol': 0.0001,
'validation_fraction': 0.1, 'verbose': True, 'warm_start': False}
```

Highlighted parameters are the parameters that I have changed from the default setting. Early stopping has been set to True to terminate training when validation score is not improving and prevent overfitting of the model. Hidden layer size is set to (10,0) which was proven to be the best during cross validation procedure. Max_iter has been set to 300 to limit the number of epoch, and random_state has been set to 1 to be used as a seed for random number generator. I have chosen sgd as a solver which employs stochastic gradient descent algorithm and verbose has been set to true to enable verbosity.

Orbits:

```
Parameter setting for MLPClassifierorbits

{'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1':
0.9, 'beta_2': 0.999, 'early_stopping': True, 'epsilon': 1e-08,
'hidden_layer_sizes': (3,), 'learning_rate': 'constant',
'learning_rate_init': 0.001, 'max_iter': 300, 'momentum': 0.9,
'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5,
'random_state': 1, 'shuffle': True, 'solver': 'sgd', 'tol': 0.0001,
'validation_fraction': 0.1, 'verbose': True, 'warm_start': False}
```

Highlighted parameters are the parameters that I have changed from the default setting. Early stopping has been set to True to terminate training when validation score is not improving and prevent overfitting of the model. Hidden layer size is set to (3,0) which was proven to be the best during cross validation procedure. Max_iter has been set to 300 to limit the number of epoch, and random_state has been set to 1 to be used as a seed for random number generator. I have chosen sgd as a solver which employs stochastic gradient descent algorithm and verbose has been set to true to enable verbosity.

Report the parameter tuning results of the neural network model using cross validation

On each dataset, cross validation was performed through GridSearchCV() from the library sklearn.model_selection. Cross validation is done by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%. Five such random data splits are performed and the average over these five trials is used to estimate the generalization performance. This was done by taking cv parameter as 5 on GridSearchCV().

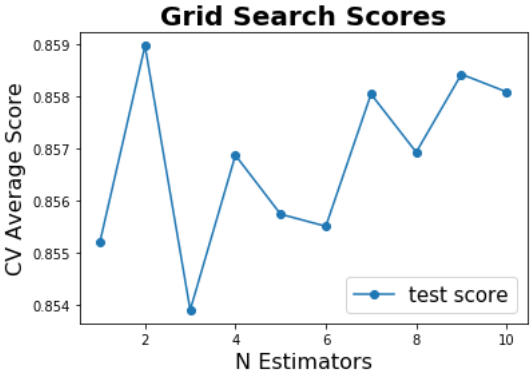
Fifa:

```
Best parameters set found on development set:

{'hidden_layer_sizes': (2,), 'learning_rate': 'constant',
'learning_rate_init': 0.01}

Grid scores on development set:

0.855 (+/-0.011) for {'hidden_layer_sizes': (1,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.859 (+/-0.009) for {'hidden_layer_sizes': (2,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.854 (+/-0.007) for {'hidden_layer_sizes': (3,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.857 (+/-0.018) for {'hidden_layer_sizes': (4,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.856 (+/-0.006) for {'hidden_layer_sizes': (5,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.856 (+/-0.009) for {'hidden_layer_sizes': (6,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.858 (+/-0.012) for {'hidden_layer_sizes': (7,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.857 (+/-0.009) for {'hidden_layer_sizes': (8,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.858 (+/-0.015) for {'hidden_layer_sizes': (9,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.858 (+/-0.014) for {'hidden_layer_sizes': (10,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
```



From this result on the output screen, it is possible to deduce that when hidden layer size was (2,), CV Average score was highest and shows the best performance. This leads us to pick the third parameter among them which

is {'hidden_layer_sizes': (2,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}

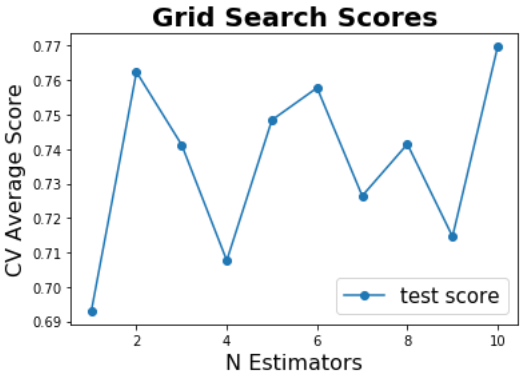
Finance

```
Best parameters set found on development set:

{'hidden_layer_sizes': (10,), 'learning_rate': 'constant',
'learning_rate_init': 0.01}

Grid scores on development set:

0.693 (+/-0.198) for {'hidden_layer_sizes': (1,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.762 (+/-0.067) for {'hidden_layer_sizes': (2,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.741 (+/-0.105) for {'hidden_layer_sizes': (3,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.708 (+/-0.035) for {'hidden_layer_sizes': (4,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.748 (+/-0.100) for {'hidden_layer_sizes': (5,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.758 (+/-0.068) for {'hidden_layer_sizes': (6,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.726 (+/-0.064) for {'hidden_layer_sizes': (7,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.741 (+/-0.042) for {'hidden_layer_sizes': (8,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.715 (+/-0.052) for {'hidden_layer_sizes': (9,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.770 (+/-0.047) for {'hidden_layer_sizes': (10,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
```



From this result on the output screen, it is possible to deduce that when hidden layer size was (10,), CV Average score was highest and shows the best performance. This leads us to pick the third parameter among them which is {'hidden_layer_sizes': (10,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}

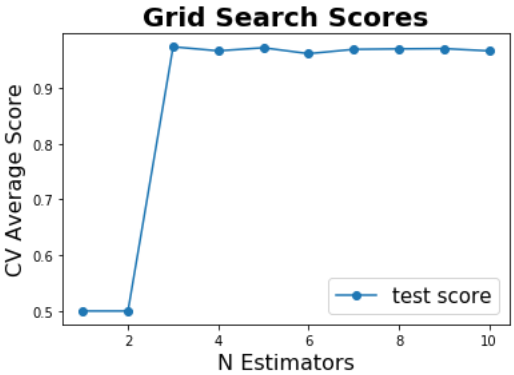
Orbits

```
Best parameters set found on development set:

{'hidden_layer_sizes': (3,), 'learning_rate': 'constant',
'learning_rate_init': 0.01}

Grid scores on development set:

0.500 (+/-0.000) for {'hidden_layer_sizes': (1,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.500 (+/-0.000) for {'hidden_layer_sizes': (2,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.974 (+/-0.013) for {'hidden_layer_sizes': (3,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.966 (+/-0.018) for {'hidden_layer_sizes': (4,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.972 (+/-0.017) for {'hidden_layer_sizes': (5,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.961 (+/-0.006) for {'hidden_layer_sizes': (6,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.969 (+/-0.021) for {'hidden_layer_sizes': (7,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.970 (+/-0.020) for {'hidden_layer_sizes': (8,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.970 (+/-0.012) for {'hidden_layer_sizes': (9,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
0.966 (+/-0.016) for {'hidden_layer_sizes': (10,), 'learning_rate':
'constant', 'learning_rate_init': 0.01}
```



From this result on the output screen, it is possible to deduce that when hidden layer size was (3,), CV Average score was highest and shows the best performance. This leads us to pick the third parameter among them which is `{'hidden_layer_sizes': (3,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}`

Compute the best (i.e., lowest) loss of the neural network model on both the training and test sets before the model is overfitted

Fifa:

```
Best loss before overfitting for train set is 0.3734014627634255
Best loss before overfitting for test set is 0.5226291333216415
```

Finance:

```
Best loss before overfitting for train set is 0.6815258689760293
Best loss before overfitting for test set is 0.6963021581617177
```

Orbits:

```
Best loss before overfitting for train set is 0.2585003390815626
Best loss before overfitting for test set is 0.512900968186387
```

If you see the scores, you can see loss score on finance dataset is highest, which imply it will perform the worst among three datasets. Next one is model on fifa set, and the model with the best performance will be model on orbits dataset.

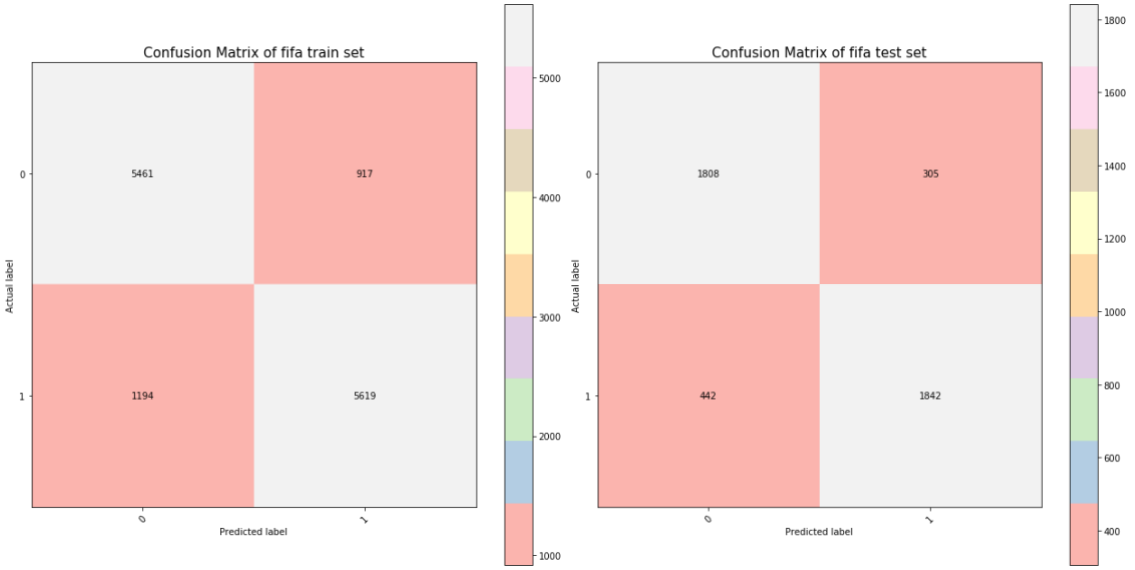
Record and visualize the experiment results of the neural network model, including performance change over time

Fifa:

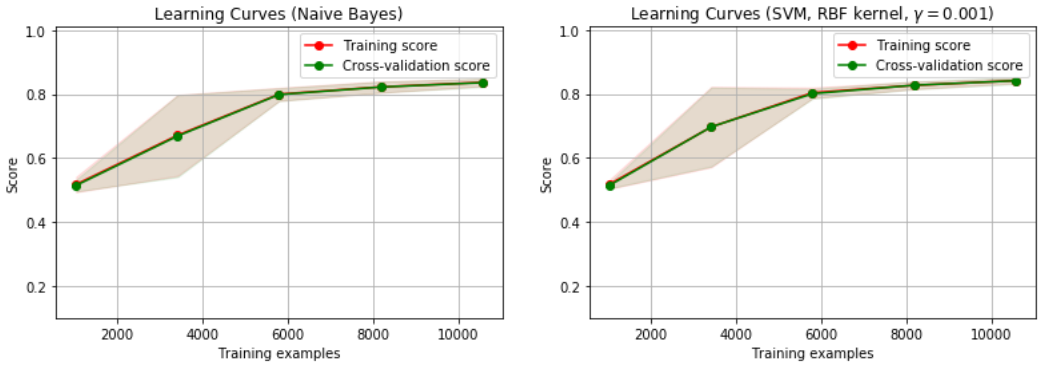
```
Mean accuracy score of fifa data training set is 0.8399666439238875 and
test set is 0.8301114396179213
```

From these output, we can say that neural network model on fifa data performs the second best.

Confusion matrix:



Learning curve:



With the data provided, mean accuracy score is second highest among three neural network models. Score is quite close to one from which I can say performance is descent. I have visualized the performance through showing the classification performed on training and test set with confusion matrix. Number written on the top left square denotes the number of correctly classified data with label 0 and number written on the bottom right square shows the number of correctly classified data with label 1.

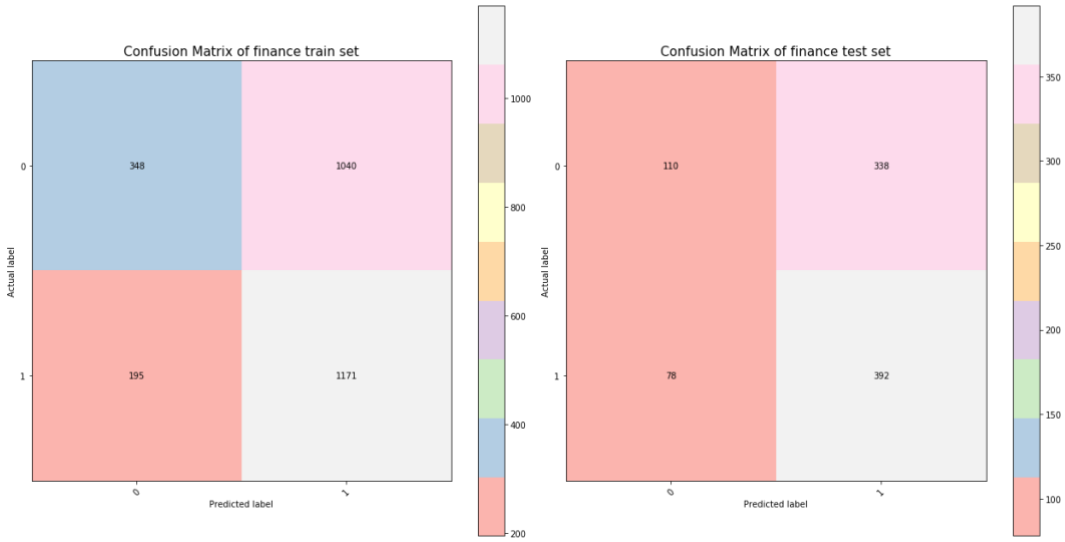
Learning curve was drawn in order to visualize the performance change over time. As the training goes on, score is gradually increasing and at the end, we can see it reaches above the score of 0.8. hence from this learning curve we can deduce score is improving as the time passes(number of training example trained increases) but the rate of increase gets low as it reaches the end.

Finance:

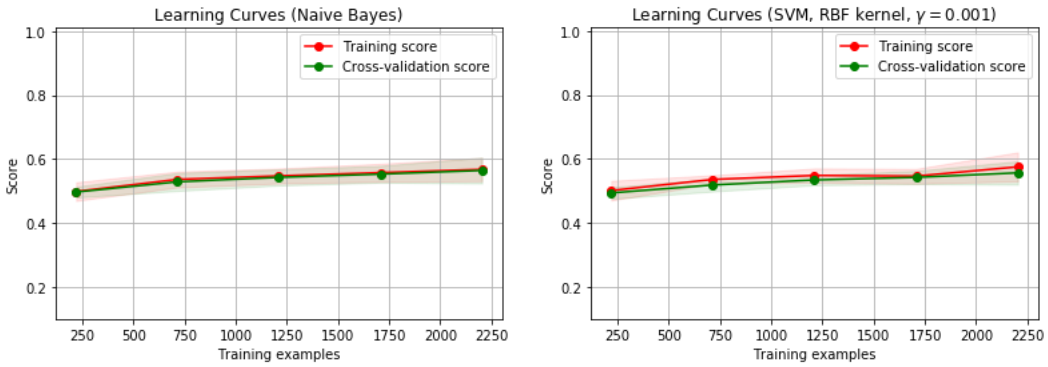
```
Mean accuracy score of finance data training set is 0.5515613652868555
and test set is 0.5468409586056645
```

From these output, we can say that neural network model on orbits data performs the worst.

Confusion matrix:



Learning curve:



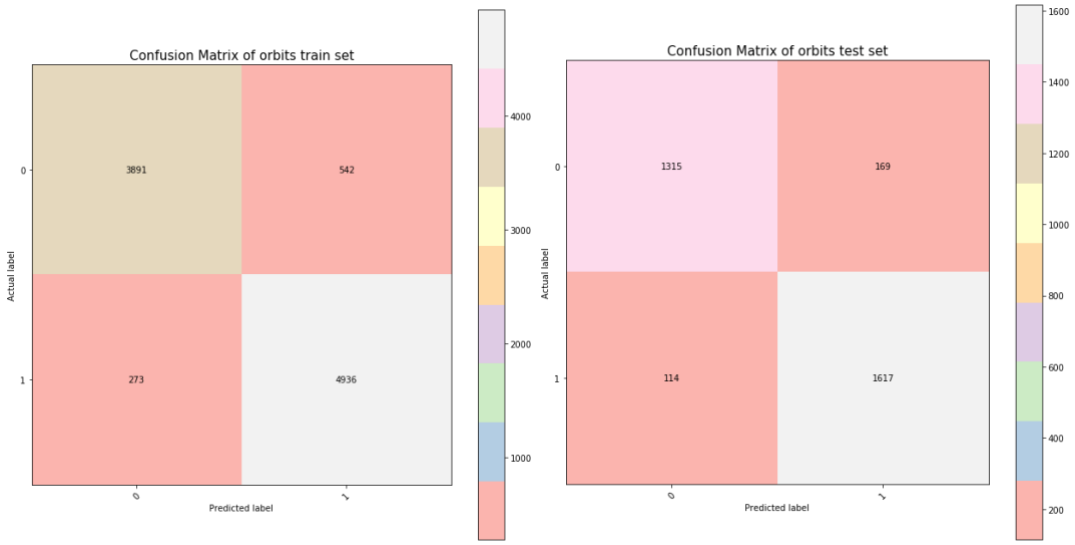
With the data provided, mean accuracy score is worst highest among three neural network models. Score is quite far from one, from which I can say performance is bad. I have visualized the performance through showing the classification performed on training and test set with confusion matrix. Number written on the top left square denotes the number of correctly classified data with label 0 and number written on the bottom right square shows the number of correctly classified data with label 1.

Learning curve was drawn in order to visualize the performance change over time. As the training goes on, score seems to increase but in very slow rate, and at the end, we can see it reaches above the score of 0.5. Hence from this learning curve we can deduce score is improving as the time passes(number of training example trained increases) but the rate is very slow.

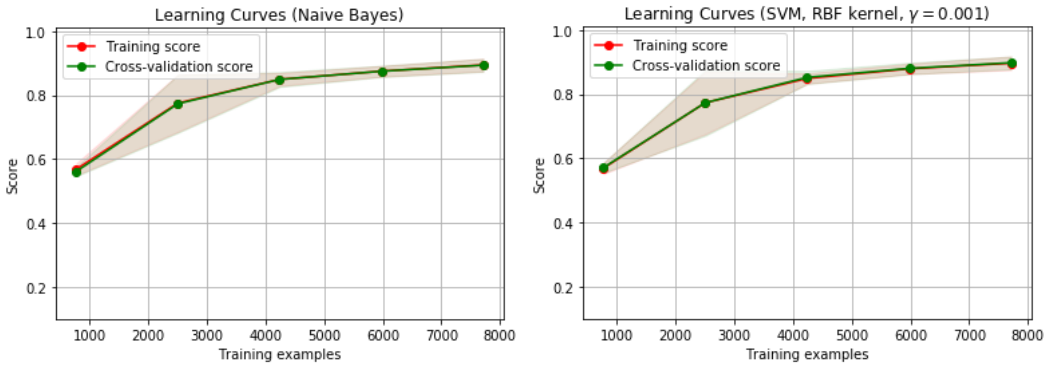
Orbits:

```
Mean accuracy score of orbits data training set is 0.9154739680564198
and test set is 0.9119751166407465
```

Confusion matrix:



Learning curve:



With the data provided, mean accuracy score is highest among three neural network models. Score is quite close to one from which I can say performance is good. I have visualized the performance through showing the classification performed on training and test set with confusion matrix. Number written on the top left square denotes the number of correctly classified data with label 0 and number written on the bottom right square shows the number of correctly classified data with label 1.

Learning curve was drawn in order to visualize the performance change over time. As the training goes on, score is gradually increasing and at the end, we can see it reaches above the score of 0.9. Hence from this learning curve we can deduce score is improving as the time passes(number of training example trained increases) but the rate gets slow as it reaches the end.

Compare and analyze the performance of all the regression and classification methods involved

To compare the performance of linear regression, logistic regression, and neural network performed on each dataset, I need to consider time taken, and the accuracy on the test set. In order to compare the performance and time, I will recall the values used to measure the accuracy, and the time taken.

Time required by each task(in seconds)

Linear regression:

```
Overall running clock (process time) : 2.45
Overall running time (wall time) : 2.493
```

Logistic regression:

```
Overall running clock (process time) : 0.675
Overall running time (wall time) : 0.447
```

Neural network:

```
Overall running clock (process time) : 282.929
Overall running time (wall time) : 444.898
```

Scores used to measure performance

Linear regression:

```
Variance score for fifa test set: 0.84
Variance score for fifa train set: 0.84
Variance score for finance test set: 1.00
Variance score for finance train set: 1.00
Variance score for orbits test set: 0.70
Variance score for orbits train set: 0.69
```

Logistic regression:

```
Mean Accuracy of fifa training set is 0.855204305966189 and test set is
0.8521719354105072
Mean Accuracy of finance training set is 0.8159041394335512 and test
set is 0.7973856209150327
Mean Accuracy of orbits training set is 0.9633893383115536 and test set
is 0.9595645412130638
```

Neural network:

```
Mean accuracy score of fifa data training set is 0.8399666439238875 and
test set is 0.8301114396179213
```

```
Mean accuracy score of finance data training set is 0.5515613652868555
and test set is 0.5468409586056645

Mean accuracy score of orbits data training set is 0.9154739680564198
and test set is 0.9119751166407465
```

On fifa datasets, linear regression, logistic regression, and neural network model perform almost equally likely. Hence, for fifa dataset, considering the time taken, logistic regression will be the best model to be used. On finance dataset, mean accuracy score is highest on logistic regression, and lowest on neural network. Moreover, logistic regression's execution time is fastest. So I can say on finance dataset, logistic regression will be the best method to be used. In addition, for orbits dataset, mean accuracy score is highest on logistic regression method and it is the fastest method. Hence even on orbits dataset, logistic regression will be the best method to be used. Overall, I can conclude that logistic regression performs the best among three methods, considering both mean accuracy score and time taken. Comparing linear regression method and neural network method, linear regression method performs way better on finance dataset, while neural network method performs better on orbits dataset. However considering the time taken, linear regression is a better method. Hence I would rank three methods as follows:

1. logistic regression
2. linear regression
3. neural network