Non-linear Model 1 - Multi-layer Perceptron

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from auxiliars import *
import pickle
```

Data ¶

In [2]:
```
data = pd.read_csv("./data/stdHTRU_2.csv")
```

We split a separate test set of relative size 20%:

In [3]:
```
X_train, X_test, y_train, y_test = train_test_split(data[data.colum
ns[0:8]],
                                                    data['class'],
                                                    test_size = 0.2
,
                                                    random_state =
1234)
```

We will analyze the performance of the method with no-correlated standarized data:

In [16]:
```
noCorrData = pd.read_csv("./data/noCorrStdHTRU_2.csv")
```

In [17]:
```
X_train_NC, X_test_NC, y_train_NC, y_test_NC = train_test_split(noC
orrData[noCorrData.columns[0:6]],
                                                    noCorrData['cla
ss'],
                                                    test_size = 0.2
,
                                                    random_state =
1234)
```

Model Training

In [4]:
```
from sklearn.neural_network import MLPClassifier
```

In [5]:
```
MLPC = MLPClassifier(random_state = 1234, solver = 'adam', max_iter
=100)
```

MLPClassifier allow us to hypertuning the following parameters:

- Hidden Layer Sizes
- Activation function
  - Logistic Sigmoid Function (logistic)
  - Hyperbolic tan Function (tanh)
  - Rectified Linear Unit Function (relu)
- Alpha (L2 Regularization)

In order to hypertuning model parameters and get a better idea on how the model performs on unseen data, we will use GridSearchCV.

In [6]:
```
from sklearn.model_selection import GridSearchCV
```

Values of the 10-Fold CV Grid to test:

In [11]:
```
grid = {'hidden_layer_sizes': [(20,), (40,), (50,), (70,), (100,),
(20,20,20), (50,50,50), (20,50,200), (50,100,50)],
        'activation': ['logistic', 'tanh', 'relu'],
        'alpha': 10.0 ** -np.arange(1, 7)}
```

In [12]:
```
grid
```

Out[12]:
```
{'hidden_layer_sizes': [(20,),
  (40,),
  (50,),
  (70,),
  (100,),
  (20, 20, 20),
  (50, 50, 50),
  (20, 50, 200),
  (50, 100, 50)],
 'activation': ['logistic', 'tanh', 'relu'],
 'alpha': array([1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06])}
```

Grid Search 10-Fold CV:

In [13]:
```
gs10cv = GridSearchCV(MLPC, param_grid = grid, cv = 10, n_jobs = -1
)
```

Normal Data Training

In [14]:
```
gs10cv.fit(X_train, y_train)
```

Out[14]:
```
GridSearchCV(cv=10, error_score=nan,
             estimator=MLPClassifier(activation='relu', alpha=0.00
01,
                                     batch_size='auto', beta_1=0.9
,
                                     beta_2=0.999, early_stopping=
False,
                                     epsilon=1e-08, hidden_layer_s
izes=(100,),
                                     learning_rate='constant',
                                     learning_rate_init=0.001, max
_fun=15000,
                                     max_iter=100, momentum=0.9,
                                     n_iter_no_change=10,
                                     nesterovs_momentum=True, powe
r_t=0.5,
                                     random_stat...
                                     validation_fraction=0.1, verb
ose=False,
                                     warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'activation': ['logistic', 'tanh', 'relu'
],
                         'alpha': array([1.e-01, 1.e-02, 1.e-03, 1
.e-04, 1.e-05, 1.e-06]),
                         'hidden_layer_sizes': [(20,), (40,), (50,
), (70,),
                                                (100,), (20, 20, 2
0),
                                                (50, 50, 50), (20,
50, 200),
                                                (50, 100, 50)]},
             pre_dispatch='2*n_jobs', refit=True, return_train_sco
re=False,
             scoring=None, verbose=0)
```

In [15]:
```
gs10cv.best_params_
```

Out[15]:
```
{'activation': 'tanh', 'alpha': 0.1, 'hidden_layer_sizes': (50, 10
0, 50)}
```

In [18]:
```
pd.DataFrame(gs10cv.cv_results_).iloc[gs10cv.best_index_]
```

Out[18]:
```
mean_fit_time
22.0088
std_fit_time
1.79401
mean_score_time
0.0131889
std_score_time
0.00378683
param_activation
tanh
param_alpha
0.1
param_hidden_layer_sizes                                          (5
0, 100, 50)
params                       {'activation': 'tanh', 'alpha': 0.1, '
hidden_l...
split0_test_score
0.97905
split1_test_score
0.98324
split2_test_score
0.979749
split3_test_score
0.980447
split4_test_score
0.977654
split5_test_score
0.981844
split6_test_score
0.98324
split7_test_score
0.981844
split8_test_score
0.983229
split9_test_score
0.981132
mean_test_score
0.981143
std_test_score
0.00181967
rank_test_score
1
Name: 62, dtype: object
```

In [20]:
```
# Save model
MLPClassFile = open('./models/MLPClass_BestCV_STDData_pickle_file',
'wb')
pickle.dump(gs10cv, MLPClassFile)
```

No-correlated Data Training

In [22]:
```
gs10cv_nc = GridSearchCV(MLPC, param_grid = grid, cv = 10, n_jobs =
-1)
```

In [23]:
```
gs10cv_nc.fit(X_train_NC, y_train_NC)
```

Out[23]:
```
GridSearchCV(cv=10, error_score=nan,
             estimator=MLPClassifier(activation='relu', alpha=0.00
01,
                                     batch_size='auto', beta_1=0.9
,
                                     beta_2=0.999, early_stopping=
False,
                                     epsilon=1e-08, hidden_layer_s
izes=(100,),
                                     learning_rate='constant',
                                     learning_rate_init=0.001, max
_fun=15000,
                                     max_iter=100, momentum=0.9,
                                     n_iter_no_change=10,
                                     nesterovs_momentum=True, powe
r_t=0.5,
                                     random_stat...
                                     validation_fraction=0.1, verb
ose=False,
                                     warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'activation': ['logistic', 'tanh', 'relu'
],
                         'alpha': array([1.e-01, 1.e-02, 1.e-03, 1
.e-04, 1.e-05, 1.e-06]),
                         'hidden_layer_sizes': [(20,), (40,), (50,
), (70,),
                                                (100,), (20, 20, 2
0),
                                                (50, 50, 50), (20,
50, 200),
                                                (50, 100, 50)]},
             pre_dispatch='2*n_jobs', refit=True, return_train_sco
re=False,
             scoring=None, verbose=0)
```

In [24]:
```
gs10cv_nc.best_params_
```

Out[24]:
```
{'activation': 'logistic', 'alpha': 0.01, 'hidden_layer_sizes': (2
0, 20, 20)}
```

In [25]:
```
pd.DataFrame(gs10cv_nc.cv_results_).iloc[gs10cv_nc.best_index_]
```

Out[25]:
```
mean_fit_time
17.0445
std_fit_time
2.21789
mean_score_time
0.0158448
std_score_time
0.0152688
param_activation
logistic
param_alpha
0.01
param_hidden_layer_sizes                                          (
20, 20, 20)
params                       {'activation': 'logistic', 'alpha': 0.
01, 'hid...
split0_test_score
0.979749
split1_test_score
0.981844
split2_test_score
0.97905
split3_test_score
0.979749
split4_test_score
0.97905
split5_test_score
0.984637
split6_test_score
0.984637
split7_test_score
0.97905
split8_test_score
0.983229
split9_test_score
0.979734
mean_test_score
0.981073
std_test_score
0.00219582
rank_test_score
1
Name: 14, dtype: object
```

In [26]:
```
# Save model
MLPClassFileNC = open('./models/MLPClass_BestCV_NCorrSTDData_pickle
_file', 'wb')
pickle.dump(gs10cv_nc, MLPClassFileNC)
```

Testing

Normal Data Model Testing

In [27]:
```
y_pred = gs10cv.predict(X_test)
```

In [28]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      3249
           1       0.90      0.84      0.87       331

    accuracy                           0.98      3580
   macro avg       0.94      0.92      0.93      3580
weighted avg       0.98      0.98      0.98      3580
```

In [29]:
```
print ("Confusion Matrix:")
confusionMatrix(y_test, y_pred, classes = [0,1])
```

```
Confusion Matrix:
```

Out[29]:

Predicted	0	1
Real		
0	3218	31
1	52	279

In [30]:
```
print("Test Error:")
(1-accuracy_score(y_test, gs10cv.predict(X_test)))*100
```

```
Test Error:
```

Out[30]:
```
2.3184357541899403
```

No-correlated Data Model Testing

In [31]:
```
y_pred_NC = gs10cv_nc.predict(X_test_NC)
```

In [32]:
```
print(classification_report(y_test_NC, y_pred_NC))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      3249
           1       0.91      0.84      0.87       331

    accuracy                           0.98      3580
   macro avg       0.95      0.91      0.93      3580
weighted avg       0.98      0.98      0.98      3580
```

In [33]:
```
print ("Confusion Matrix:")
confusionMatrix(y_test_NC, y_pred_NC, classes = [0,1])
```

```
Confusion Matrix:
```

Out[33]:

Predicted	0	1
Real		
0	3223	26
1	54	277

In [34]:
```
print("Test Error:")
(1-accuracy_score(y_test_NC, gs10cv_nc.predict(X_test_NC)))*100
```

```
Test Error:
```

Out[34]:
```
2.2346368715083775
```