Linear Model 2 - Logistic regression

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.genmod.generalized_linear_model import GLM
from sklearn.feature_selection import RFECV
from sklearn.model_selection import train_test_split
from statsmodels.genmod.families.family import Binomial
from statsmodels.tools.tools import add_constant
from sklearn.metrics import confusion_matrix, classification_report
, accuracy_score
from auxiliars import *
import pickle
```

In [3]:
```
np.random.seed(543)
```

Data

Standarized data loading:

In [4]:
```
data = pd.read_csv("./data/stdHTRU_2.csv")
```

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

In [5]:
```
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)
print(data[data.columns[0:8]])
```

```
       Profile_mean  Profile_stdev  Profile_skewness  Profile_kurt
osis  \
0          1.149317       1.334832         -0.669570         -0.40
0459
1         -0.334168       1.802265         -0.011785         -0.37
0535
2         -0.314372      -1.053322         -0.145233         -0.11
6593
3          1.000694       1.553254         -0.513409         -0.39
0178
4         -0.871402      -0.858879          0.115609         -0.10
4866
...             ...            ...               ...
...
17893      0.988208       1.943284         -0.625655         -0.40
6697
17894      0.447319       0.429062         -0.328831         -0.23
4643
17895      0.321842       1.956220         -0.299334         -0.40
7492
17896      0.133628       1.074510         -0.260050         -0.29
1041
17897     -2.105762       5.735470          0.872674         -0.27
2508

        DM_mean  DM_stdev  DM_skewness  DM_kurtosis
0     -0.319440 -0.370625    -0.072798    -0.287438
1     -0.371102 -0.588924     0.504427     0.211581
2     -0.322107 -0.235328    -0.125996    -0.391373
3     -0.304404 -0.275666    -0.312265    -0.481300
4     -0.388010 -0.763111     1.324026     1.386794
...         ...       ...          ...          ...
17893 -0.384010 -0.727295     1.586054     1.700034
17894  0.128776  0.939926    -1.189159    -0.906574
17895  0.299137  1.671568    -1.288079    -0.941330
17896 -0.361967 -0.664857     0.378257     0.275850
17897  5.961291  1.971546    -2.197327    -0.971052

[17898 rows x 8 columns]
```

I order to improve the performance of logistic regression, we will also analyze the performance of the method with no-correlated standarized data:

In [8]:
```
noCorrData = pd.read_csv("./data/noCorrStdHTRU_2.csv")
```

In [9]:
```
X_train_NC, X_test_NC, y_train_NC, y_test_NC = train_test_split(noCorrData[noCorrData.columns[0:6]],
                                                                noCorrData['class'],
                                                                test_size = 0.2)

print(noCorrData[noCorrData.columns[0:6]])
```

```
       Profile_mean  Profile_stdev  Profile_skewness   DM_mean  DM_stdev  \
0          1.149317       1.334832         -0.669570 -0.319440 -0.370625
1         -0.334168       1.802265         -0.011785 -0.371102 -0.588924
2         -0.314372      -1.053322         -0.145233 -0.322107 -0.235328
3          1.000694       1.553254         -0.513409 -0.304404 -0.275666
4         -0.871402      -0.858879          0.115609 -0.388010 -0.763111
...             ...            ...               ...       ...       ...
17893      0.988208       1.943284         -0.625655 -0.384010 -0.727295
17894      0.447319       0.429062         -0.328831  0.128776  0.939926
17895      0.321842       1.956220         -0.299334  0.299137  1.671568
17896      0.133628       1.074510         -0.260050 -0.361967 -0.664857
17897     -2.105762       5.735470          0.872674  5.961291  1.971546

       DM_skewness
0        -0.072798
1         0.504427
2        -0.125996
3        -0.312265
4         1.324026
...            ...
17893     1.586054
17894    -1.189159
17895    -1.288079
17896     0.378257
17897    -2.197327

[17898 rows x 6 columns]
```

Model Training

Scikit-learn library offersa method for Logistic Regression classification.

In [10]:
```
from sklearn.linear_model import LogisticRegression
```

In [11]:
```
LR = LogisticRegression(n_jobs = -1)
```

LogisticRegression allow us to hypertuning the following parameters:

- Penalty: Used to specify the norm used in the penalization.
  - L1: Lasso regression.
  - L2: Ridge regression.
- C: Inverse of regularization strength
- Algorithm to use in the optimization problem:
  - liblinear: for small datasets.
  - saga: for larger datasets.

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

In [12]:
```
from sklearn.model_selection import GridSearchCV
```

Values of the 10-Fold CV Grid to test:

In [13]:
```
grid = {'penalty' : ['l1','l2'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['liblinear', 'saga']}
```

In [14]:
```
grid
```

Out[14]:
```
{'penalty': ['l1', 'l2'],
 'C': array([1.00000000e-04, 2.63665090e-04, 6.95192796e-04, 1.832
98071e-03,
        4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790
e-02,
        2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240
e+00,
        1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808
e+02,
        5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000
e+04]),
 'solver': ['liblinear', 'saga']}
```

Grid Search 10-Fold CV:

In [15]:
```
gs10cv = GridSearchCV(LR, param_grid = grid, cv = 10, n_jobs = -1)
```

Normal Data Training

In [16]:
```
gs10cv.fit(X_train, y_train)
```

```
/home/ferja/.local/lib/python3.7/site-packages/sklearn/linear_mode
l/_logistic.py:1539: UserWarning: 'n_jobs' > 1 does not have any e
ffect when 'solver' is set to 'liblinear'. Got 'n_jobs' = 4.
  " = {}.".format(effective_n_jobs(self.n_jobs)))
```

Out[16]:
```
GridSearchCV(cv=10, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None
, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_
ratio=None,
                                          max_iter=100, multi_clas
s='auto',
                                          n_jobs=-1, penalty='l2',
                                          random_state=None, solve
r='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'C': array([1.00000000e-04, 2.6366...
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e
-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e
+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e
+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e
+04]),
                         'penalty': ['l1', 'l2'],
                         'solver': ['liblinear', 'saga']},
             pre_dispatch='2*n_jobs', refit=True, return_train_sco
re=False,
             scoring=None, verbose=0)
```

In [17]:
```
gs10cv.best_params_
```

Out[17]:
```
{'C': 4.281332398719396, 'penalty': 'l1', 'solver': 'liblinear'}
```

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

Out[18]:
```
mean_fit_time                                                 0.16
2662
std_fit_time                                                  0.05
0302
mean_score_time                                             0.0019
2318
std_score_time                                              0.0002
4696
param_C                                                        4.2
8133
param_penalty
l1
param_solver                                                libli
near
params               {'C': 4.281332398719396, 'penalty': 'l1', 'so
l...
split0_test_score                                             0.9
7905
split1_test_score                                             0.9
7905
split2_test_score                                            0.97
6257
split3_test_score                                            0.97
9749
split4_test_score                                            0.97
5559
split5_test_score                                            0.98
1145
split6_test_score                                             0.9
8324
split7_test_score                                             0.9
7905
split8_test_score                                            0.98
1132
split9_test_score                                            0.97
9734
mean_test_score                                              0.97
9397
std_test_score                                              0.0021
4541
rank_test_score
1
Name: 44, dtype: object
```

In [19]:
```
# Save model
LRFile = open('./models/LR_BestCV_STDData_pickle_file', 'wb')
pickle.dump(gs10cv, LRFile)
```

No-correlated Data Training

Grid Search 10-Fold CV:

In [20]:
```
gs10cv_nc = GridSearchCV(LR, param_grid = grid, cv = 10, n_jobs = -
1)
```

Training:

In [21]:
```
gs10cv_nc.fit(X_train_NC, y_train_NC)
```

```
/home/ferja/.local/lib/python3.7/site-packages/sklearn/linear_mode
l/_logistic.py:1539: UserWarning: 'n_jobs' > 1 does not have any e
ffect when 'solver' is set to 'liblinear'. Got 'n_jobs' = 4.
  " = {}.".format(effective_n_jobs(self.n_jobs)))
```

Out[21]:
```
GridSearchCV(cv=10, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None
, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_
ratio=None,
                                          max_iter=100, multi_clas
s='auto',
                                          n_jobs=-1, penalty='l2',
                                          random_state=None, solve
r='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'C': array([1.00000000e-04, 2.6366...
       4.83293024e-03, 1.27427499e-02, 3.35981829e-02, 8.85866790e
-02,
       2.33572147e-01, 6.15848211e-01, 1.62377674e+00, 4.28133240e
+00,
       1.12883789e+01, 2.97635144e+01, 7.84759970e+01, 2.06913808e
+02,
       5.45559478e+02, 1.43844989e+03, 3.79269019e+03, 1.00000000e
+04]),
                         'penalty': ['l1', 'l2'],
                         'solver': ['liblinear', 'saga']},
             pre_dispatch='2*n_jobs', refit=True, return_train_sco
re=False,
             scoring=None, verbose=0)
```

In [22]:
```
pd.DataFrame(gs10cv_nc.cv_results_).iloc[gs10cv_nc.best_index_]
```

Out[22]:
```
mean_fit_time                                                  0.045
2667
std_fit_time                                                 0.0041
9163
mean_score_time                                              0.0016
0186
std_score_time                                              0.00021
9846
param_C                                                        4.2
8133
param_penalty
l1
param_solver                                                 libli
near
params               {'C': 4.281332398719396, 'penalty': 'l1', 'so
l...
split0_test_score                                             0.97
9749
split1_test_score                                             0.97
2067
split2_test_score                                              0.9
8324
split3_test_score                                             0.98
0447
split4_test_score                                             0.97
8352
split5_test_score                                             0.97
6955
split6_test_score                                             0.98
2542
split7_test_score                                             0.97
3464
split8_test_score                                             0.98
0433
split9_test_score                                             0.98
0433
mean_test_score                                               0.97
8768
std_test_score                                               0.0034
6507
rank_test_score
1
Name: 44, dtype: object
```

In [23]:
```
# Save model
LRFileNC = open('./models/LR_BestCV_NCorrSTDData_pickle_file', 'wb'
)
pickle.dump(gs10cv_nc, LRFile)
```

Testing

Normal Data Model Testing

In [48]:
```
y_pred = gs10cv.predict(X_test)
print("Confusion Matrix:")
confusionMatrix(y_test, y_pred, classes = [0,1])
```

```
Confusion Matrix:
```

Out[48]:

Predicted	0	1
Real		
0	3229	20
1	60	271

In [47]:
```
print(classification_report(y_test, y_pred, target_names=['no', 'ye
s']))
```

```
              precision    recall  f1-score   support

          no       0.98      0.99      0.99      3249
         yes       0.93      0.82      0.87       331

    accuracy                           0.98      3580
   macro avg       0.96      0.91      0.93      3580
weighted avg       0.98      0.98      0.98      3580
```

In [44]:
```
print("Test Error:")
(1-accuracy_score(y_test, y_pred))*100
```

```
Test Error:
```

Out[44]:
```
2.2346368715083775
```

No-correlated Data Model Testing

In [26]:
```
y_pred_NC = gs10cv_nc.predict(X_test_NC)
print("Confusion Matrix:")
confusionMatrix(y_test_NC, y_pred_NC, classes = [0,1])
```

```
[0 0 0 ... 0 0 0]
Confusion Matrix:
```

Out[26]:

Predicted	0	1
Real		
0	3228	15
1	77	260

In [50]:
```
print(classification_report(y_test_NC, y_pred_NC, target_names=['no
', 'yes']))
```

```
              precision    recall  f1-score   support

          no       0.98      1.00      0.99      3243
         yes       0.95      0.77      0.85       337

    accuracy                           0.97      3580
   macro avg       0.96      0.88      0.92      3580
weighted avg       0.97      0.97      0.97      3580
```

In [27]:
```
print("Test Error:")
(1-accuracy_score(y_test_NC, gs10cv_nc.predict(X_test_NC)))*100
```

```
Test Error:
```

Out[27]:
```
2.5698324022346397
```