Silesian
University
of Technology

Report

Information leakage avoidance

Section

Ankit Rathi

Maurice Ngabonziza

Jean Remy

Leakage of information results in an optimistic bias during testing (or validation) of the classification system and as a result wrong classifier evaluation. It appears when the patterns from the test set are used in any way when teaching the classifier.

In the task we have three different figure given –

First one was An example of a validation scenario in which an information leakage is evident is a resubstitution method in which the same data set is used both to teach the classifier and to test it and the second was where a separate test set is not used to teach the classifier itself, but the information contained therein is used in the earlier stages of data processing, in particular during the selection of features, where as the third one was The correct validation scheme for the multistage classification model, including the stages of feature selection and classification.

The aim of the laboratory was to investigate the phenomenon of information leakage in various validation scenarios.

First we applied one vs one decomposition to obtain two class classification.

We worked on a .csv file provided in the platform we split the data into test and training set. I have used python to solve this task I used sklearn.linear model to import Logistic regression and SGDCclassifier , I have used sklearn.feature_selection to import RFE and sklear.model_selection to import train_test_split.

Than I defined scenarios where first stage was data prepration stage followed by feature selection state and than Trainset Prepartion stage Data Paritioning, followed by learnig stage and evaluation stage, and I have defined Three scenarios named as scenario one , two and three.

We have got the following result for it:

A vs B

Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51

Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.51

Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.50

A vs C

Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51

Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.50

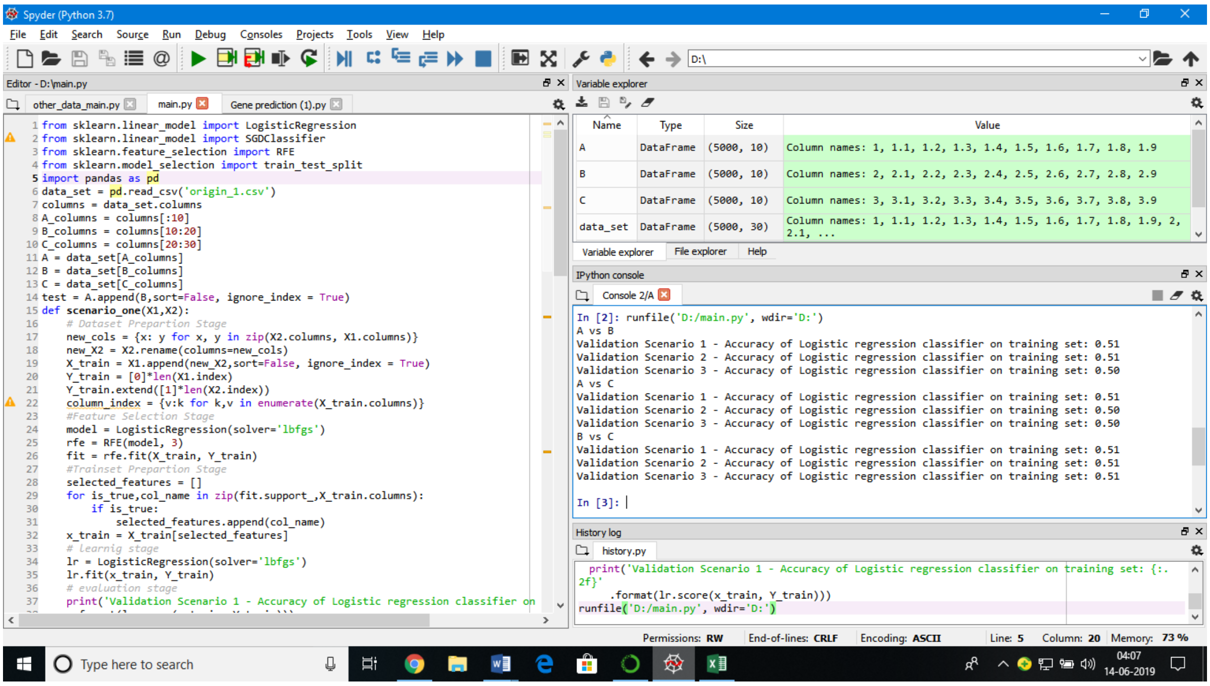
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.50

B vs C

Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51

Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.51

Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.51



```python
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
import pandas as pd
data_set = pd.read_csv('origin_1.csv')
columns = data_set.columns
A_columns = columns[:10]
B_columns = columns[10:20]
C_columns = columns[20:30]
A = data_set[A_columns]
B = data_set[B_columns]
C = data_set[C_columns]
test = A.append(B,sort=False, ignore_index = True)
def scenario_one(X1,X2):
    # Dataset Prepartion Stage
    new_cols = {x: y for x, y in zip(X2.columns, X1.columns)}
    new_X2 = X2.rename(columns=new_cols)
    X_train = X1.append(new_X2,sort=False, ignore_index = True)
    Y_train = [0]*len(X1.index)
    Y_train.extend([1]*len(X2.index))
    column_index = {v:k for k,v in enumerate(X_train.columns)}
    #Feature Selection Stage
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model, 3)
    fit = rfe.fit(X_train, Y_train)
    #Trainset Prepartion Stage
    selected_features = []
    for is_true,col_name in zip(fit.support_,X_train.columns):
        if is_true:
            selected_features.append(col_name)
    x_train = X_train[selected_features]
    # learnig stage
    lr = LogisticRegression(solver='lbfgs')
    lr.fit(x_train, Y_train)
    # evaluation stage
    print('Validation Scenario 1 - Accuracy of Logistic regression classifier on
```

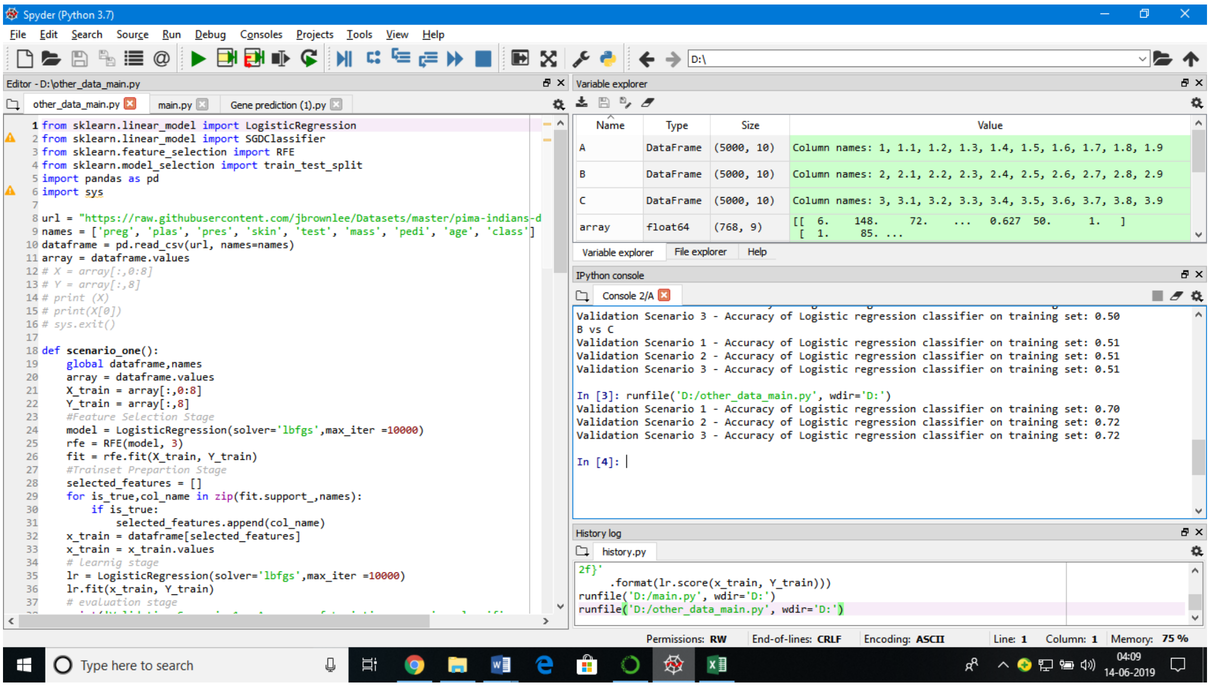
Name	Type	Size	Value
A	DataFrame	(5000, 10)	Column names: 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9
B	DataFrame	(5000, 10)	Column names: 2, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9
C	DataFrame	(5000, 10)	Column names: 3, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9
data_set	DataFrame	(5000, 30)	Column names: 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, ...

```
In [2]: runfile('D:/main.py', wdir='D:')
A vs B
Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.50
A vs C
Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.50
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.50
B vs C
Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.51

In [3]: |
```

```
  print('Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: {:.
2f}'
      .format(lr.score(x_train, Y_train)))
runfile('D:/main.py', wdir='D:')
```

Than we performed the same action for the arbitrary chosen dataset



```python
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
import pandas as pd
import sys

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-d
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pd.read_csv(url, names=names)
array = dataframe.values
# X = array[:,0:8]
# Y = array[:,8]
# print (X)
# print(X[0])
# sys.exit()

def scenario_one():
    global dataframe,names
    array = dataframe.values
    X_train = array[:,0:8]
    Y_train = array[:,8]
    #Feature Selection Stage
    model = LogisticRegression(solver='lbfgs',max_iter =10000)
    rfe = RFE(model, 3)
    fit = rfe.fit(X_train, Y_train)
    #Trainset Prepartion Stage
    selected_features = []
    for is_true,col_name in zip(fit.support_,names):
        if is_true:
            selected_features.append(col_name)
    x_train = dataframe[selected_features]
    x_train = x_train.values
    # learnig stage
    lr = LogisticRegression(solver='lbfgs',max_iter =10000)
    lr.fit(x_train, Y_train)
    # evaluation stage
```

Name	Type	Size	Value
A	DataFrame	(5000, 10)	Column names: 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9
B	DataFrame	(5000, 10)	Column names: 2, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9
C	DataFrame	(5000, 10)	Column names: 3, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9
array	float64	(768, 9)	[[6. 148. 72. ... 0.627 50. 1.] [1. 85. ...

```
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.50
B vs C
Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.51
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.51

In [3]: runfile('D:/other_data_main.py', wdir='D:')
Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.70
Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.72
Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.72

In [4]: |
```

```
2f}'
      .format(lr.score(x_train, Y_train)))
runfile('D:/main.py', wdir='D:')
runfile('D:/other_data_main.py', wdir='D:')
```

Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: 0.70

Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: 0.72

Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: 0.72

#code :

(for question 1)

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
import pandas as pd
data_set = pd.read_csv('origin_1.csv')
columns = data_set.columns
A_columns = columns[:10]
B_columns = columns[10:20]
C_columns = columns[20:30]
A = data_set[A_columns]
B = data_set[B_columns]
C = data_set[C_columns]
test = A.append(B,sort=False, ignore_index = True)
def scenario_one(X1,X2):
    # Dataset Prepartion Stage
    new_cols = {x: y for x, y in zip(X2.columns, X1.columns)}
    new_X2 = X2.rename(columns=new_cols)
    X_train = X1.append(new_X2,sort=False, ignore_index = True)
    Y_train = [0]*len(X1.index)
    Y_train.extend([1]*len(X2.index))
    column_index = {v:k for k,v in enumerate(X_train.columns)}
    #Feature Selection Stage
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model, 3)
    fit = rfe.fit(X_train, Y_train)
    #Trainset Prepartion Stage
    selected_features = []
    for is_true,col_name in zip(fit.support_,X_train.columns):
        if is_true:
            selected_features.append(col_name)
    x_train = X_train[selected_features]
    # learnig stage
    lr = LogisticRegression(solver='lbfgs')
    lr.fit(x_train, Y_train)
    # evaluation stage
    print('Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: {:.2f}'
  .format(lr.score(x_train, Y_train)))


def scenario_two(X1,X2):
    # Dataset Prepartion Stage
    new_cols = {x: y for x, y in zip(X2.columns, X1.columns)}
    new_X2 = X2.rename(columns=new_cols)
    X_train = X1.append(new_X2,sort=False, ignore_index = True)
    Y_train = [0]*len(X1.index)
    Y_train.extend([1]*len(X2.index))
    column_index = {v:k for k,v in enumerate(X_train.columns)}

    #Feature Selection Stage
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model, 3)
    fit = rfe.fit(X_train, Y_train)

    selected_features = []
    for is_true,col_name in zip(fit.support_,X_train.columns):
        if is_true:
            selected_features.append(col_name)

    #Trainset Prepartion stage  Data Paritioning
    x_train = X_train[selected_features]
    X_train, X_test, y_train, y_test = train_test_split(x_train, Y_train, random_state=0)
    # learnig stage
    lr = LogisticRegression(solver='lbfgs')
    lr.fit(X_train, y_train)

    # evaluation stage
    print('Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: {:.2f}'
 .format(lr.score(X_test, y_test)))



def scenario_three(X1,X2):
    # Dataset Prepartion Stage
    new_cols = {x: y for x, y in zip(X2.columns, X1.columns)}
    new_X2 = X2.rename(columns=new_cols)
    X_train = X1.append(new_X2,sort=False, ignore_index = True)
    Y_train = [0]*len(X1.index)
    Y_train.extend([1]*len(X2.index))
    column_index = {v:k for k,v in enumerate(X_train.columns)}

    # Data Partioning
    X_train, X_test, y_train, y_test = train_test_split(X_train, Y_train, random_state=0)

    #Feature Selection Stage
    model = LogisticRegression(solver='lbfgs')
    rfe = RFE(model, 3)
    fit = rfe.fit(X_train, y_train)

    selected_features = []
    for is_true,col_name in zip(fit.support_,X_train.columns):
        if is_true:
            selected_features.append(col_name)
            #Trainset Prepartion stage  Data Paritioning
            X_test = X_test[selected_features]
            X_train = X_train[selected_features]

            # learnig stage
            lr = LogisticRegression(solver='lbfgs')
            lr.fit(X_train, y_train)

            # evaluation stage
            print('Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: {:.2f}'
        .format(lr.score(X_test, y_test)))
print("A vs B")
scenario_one(A,B)
scenario_two(A,B)
scenario_three(A,B)
print("A vs C")
scenario_one(A,C)
scenario_two(A,C)
scenario_three(A,C)
print("B vs C")
scenario_one(B,C)
scenario_two(B,C)
scenario_three(B,C)

#For (task 2 )
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
import pandas as pd
import sys


url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

dataframe = pd.read_csv(url, names=names)

array = dataframe.values

# X = array[:,0:8]

# Y = array[:,8]

# print (X)

# print(X[0])

# sys.exit()


def scenario_one():

    global dataframe,names

    array = dataframe.values

    X_train = array[:,0:8]

    Y_train = array[:,8]

    #Feature Selection Stage

    model = LogisticRegression(solver='lbfgs',max_iter =10000)

    rfe = RFE(model, 3)

    fit = rfe.fit(X_train, Y_train)

    #Trainset Prepartion Stage

    selected_features = []

    for is_true,col_name in zip(fit.support_,names):

        if is_true:

            selected_features.append(col_name)

    x_train = dataframe[selected_features]

    x_train = x_train.values

    # learnig stage

    lr = LogisticRegression(solver='lbfgs',max_iter =10000)
```

```
        lr.fit(x_train, Y_train)
        # evaluation stage
        print('Validation Scenario 1 - Accuracy of Logistic regression classifier on training set: {:.2f}'
    .format(lr.score(x_train, Y_train)))



def scenario_two():
        global dataframe,names
        array = dataframe.values
        X_train = array[:,0:8]
        Y_train = array[:,8]

        #Feature Selection Stage
        model = LogisticRegression(solver='lbfgs',max_iter =10000)
        rfe = RFE(model, 3)
        fit = rfe.fit(X_train, Y_train)

        selected_features = []
        for is_true,col_name in zip(fit.support_,names):
                if is_true:
                        selected_features.append(col_name)

        #Trainset Prepartion stage  Data Paritioning
        x_train = dataframe[selected_features].values
        X_train, X_test, y_train, y_test = train_test_split(x_train, Y_train, random_state=0)

        # learnig stage
        lr = LogisticRegression(solver='lbfgs')
        lr.fit(X_train, y_train)

        # evaluation stage
        print('Validation Scenario 2 - Accuracy of Logistic regression classifier on training set: {:.2f}'
    .format(lr.score(X_test, y_test)))



def scenario_three():
        # Dataset Prepartion Stage
        global dataframe,names
        array = dataframe.values
        X_train = array[:,0:8]
        Y_train = array[:,8]

        # Data Partioning
        X_train, X_test, y_train, y_test = train_test_split(X_train, Y_train, random_state=0)

        #Feature Selection Stage
        model = LogisticRegression(solver='lbfgs',max_iter=10000)
        rfe = RFE(model, 3)
        fit = rfe.fit(X_train, y_train)

        selected_features = []
        for index,is_true in enumerate(fit.support_):
                if is_true:
                        selected_features.append(index)

        #Trainset Prepartion stage  Data Paritioning
        temp = []
        for item in X_train:
                new_arr = []
                for feature in selected_features:
                        new_arr.append(item[feature])
                temp.append(new_arr)
        X_train = temp

        temp = []
        for item in X_test:
            new_arr = []
            for feature in selected_features:
                new_arr.append(item[feature])
            temp.append(new_arr)
        X_test = temp

        # X_test = dataframe[selected_features].values
        # X_train = dataframe[selected_features].values

        # learnig stage
        lr = LogisticRegression(solver='lbfgs')
        lr.fit(X_train, y_train)

        # evaluation stage
        print('Validation Scenario 3 - Accuracy of Logistic regression classifier on training set: {:.2f}'
  .format(lr.score(X_test, y_test)))
scenario_one()
scenario_two()
scenario_three()
```