November 1, 2023

1 Logistic Regression

Exp no.: 8

Aim: Logistic Regression

[1]:
```
#Name:Vedant Wankhade
#Roll no.:74
#Sec:B
#Aim:SVM Classifier
#Year:3rd Year
```

[2]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
os.getcwd()
```

[3]:
```
'C:\\Users\\hp\\Downloads'
```

[4]:
```
os.chdir('C:\\Users\\HP\\Desktop')
```

[5]:
```
df=pd.read_csv('framingham.csv')
```

[6]:
```
df.head()
```

[6]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds  prevalentStroke  \
0     1   39        4.0              0         0.0     0.0                0
1     0   46        2.0              0         0.0     0.0                0
2     1   48        1.0              1        20.0     0.0                0
3     0   61        3.0              1        30.0     0.0                0
4     0   46        3.0              1        23.0     0.0                0
```

```
   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0     77.0
1             0         0    250.0  121.0   81.0  28.73       95.0     76.0
2             0         0    245.0  127.5   80.0  25.34       75.0     70.0
3             1         0    225.0  150.0   95.0  28.58       65.0    103.0
4             0         0    285.0  130.0   84.0  23.10       85.0     85.0

   TenYearCHD
0           0
1           0
2           0
3           1
4           0
```

[7]:
```
df.tail()
```

[7]:
```
      male  age  education  currentSmoker  cigsPerDay  BPMeds  \
4233     1   50        1.0              1         1.0     0.0
4234     1   51        3.0              1        43.0     0.0
4235     0   48        2.0              1        20.0     NaN
4236     0   44        1.0              1        15.0     0.0
4237     0   52        2.0              0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
4233                0             1         0    313.0  179.0   92.0  25.97
4234                0             0         0    207.0  126.5   80.0  19.71
4235                0             0         0    248.0  131.0   72.0  22.00
4236                0             0         0    210.0  126.5   87.0  19.16
4237                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
4233       66.0     86.0           1
4234       65.0     68.0           0
4235       84.0     86.0           0
4236       86.0      NaN           0
4237       80.0    107.0           0
```

[8]:
```
df.info
```

[8]:
```
<bound method DataFrame.info of        male  age  education  currentSmoker
cigsPerDay  BPMeds  \
0         1   39        4.0              0         0.0     0.0
1         0   46        2.0              0         0.0     0.0
2         1   48        1.0              1        20.0     0.0
3         0   61        3.0              1        30.0     0.0
4         0   46        3.0              1        23.0     0.0
…     …  …         …             …         …      …
```

```
4233     1   50        1.0                1         1.0      0.0
4234     1   51        3.0                1        43.0      0.0
4235     0   48        2.0                1        20.0      NaN
4236     0   44        1.0                1        15.0      0.0
4237     0   52        2.0                0         0.0      0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                 …             …        …        …      …      …      …
4233                0             1         0    313.0  179.0   92.0  25.97
4234                0             0         0    207.0  126.5   80.0  19.71
4235                0             0         0    248.0  131.0   72.0  22.00
4236                0             0         0    210.0  126.5   87.0  19.16
4237                0             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose  TenYearCHD
0          80.0     77.0           0
1          95.0     76.0           0
2          75.0     70.0           0
3          65.0    103.0           1
4          85.0     85.0           0
…           …        …           …
4233       66.0     86.0           1
4234       65.0     68.0           0
4235       84.0     86.0           0
4236       86.0      NaN           0
4237       80.0    107.0           0

[4238 rows x 16 columns]>
```

[9]:
```
df.describe()
```

[9]:
```
              male          age    education  currentSmoker   cigsPerDay  \
count  4238.000000  4238.000000  4133.000000    4238.000000  4209.000000
mean      0.429212    49.584946     1.978950       0.494101     9.003089
std       0.495022     8.572160     1.019791       0.500024    11.920094
min       0.000000    32.000000     1.000000       0.000000     0.000000
25%       0.000000    42.000000     1.000000       0.000000     0.000000
50%       0.000000    49.000000     2.000000       0.000000     0.000000
75%       1.000000    56.000000     3.000000       1.000000    20.000000
max       1.000000    70.000000     4.000000       1.000000    70.000000

            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
```

```
count  4185.000000      4238.000000   4238.000000  4238.000000  4188.000000
mean      0.029630         0.005899      0.310524     0.025720   236.721585
std       0.169584         0.076587      0.462763     0.158316    44.590334
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   263.000000
max       1.000000         1.000000      1.000000     1.000000   696.000000

             sysBP        diaBP          BMI    heartRate      glucose  \
count  4238.000000  4238.000000  4219.000000  4237.000000  3850.000000
mean    132.352407    82.893464    25.802008    75.878924    81.966753
std      22.038097    11.910850     4.080111    12.026596    23.959998
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.070000    68.000000    71.000000
50%     128.000000    82.000000    25.400000    75.000000    78.000000
75%     144.000000    89.875000    28.040000    83.000000    87.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

        TenYearCHD
count  4238.000000
mean      0.151958
std       0.359023
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

[10]:
```
df.isna().sum()
```

[10]:
```
male                 0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
```

```
dtype: int64
```

[11]:
```
df['glucose'].fillna(value = df['glucose'].mean(),inplace=True)
```

[12]:
```
df['education'].fillna(value = df['education'].mean(),inplace=True)
```

[13]:
```
df['heartRate'].fillna(value = df['heartRate'].mean(),inplace=True)
```

[14]:
```
df['BMI'].fillna(value = df['BMI'].mean(),inplace=True)
```

df['cigsPerDay'].fillna(value = df['cigsPerDay'].mean(),inplace=True)

[15]:
```
df['totChol'].fillna(value = df['totChol'].mean(),inplace=True)
```

[16]:
```
df['BPMeds'].fillna(value = df['BPMeds'].mean(),inplace=True)
```

[17]:
```
df.isna().sum()
```

[17]:
```
male                0
age                 0
education           0
currentSmoker       0
cigsPerDay         29
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol             0
sysBP               0
diaBP               0
BMI                 0
heartRate           0
glucose             0
TenYearCHD          0
dtype: int64
```

[18]:
```
df.isna().sum()
```

[18]:
```
male                0
age                 0
education           0
currentSmoker       0
cigsPerDay         29
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
```

```
totChol             0
sysBP               0
diaBP               0
BMI                 0
heartRate           0
glucose             0
TenYearCHD          0
dtype: int64
```

[19]:
```
#Splitting the dependent and independent variables.
x = df.drop("TenYearCHD",axis=1)
y = df['TenYearCHD']
```

[20]:
```
x #checking the features
```

[20]:
```
      male  age  education  currentSmoker  cigsPerDay   BPMeds  \
0        1   39        4.0              0         0.0  0.00000
1        0   46        2.0              0         0.0  0.00000
2        1   48        1.0              1        20.0  0.00000
3        0   61        3.0              1        30.0  0.00000
4        0   46        3.0              1        23.0  0.00000
…      …   …          …            …          …      …
4233     1   50        1.0              1         1.0  0.00000
4234     1   51        3.0              1        43.0  0.00000
4235     0   48        2.0              1        20.0  0.02963
4236     0   44        1.0              1        15.0  0.00000
4237     0   52        2.0              0         0.0  0.00000

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                   0             0         0    195.0  106.0   70.0  26.97
1                   0             0         0    250.0  121.0   81.0  28.73
2                   0             0         0    245.0  127.5   80.0  25.34
3                   0             1         0    225.0  150.0   95.0  28.58
4                   0             0         0    285.0  130.0   84.0  23.10
…                …             …        …       …      …     …     …
4233                0             1         0    313.0  179.0   92.0  25.97
4234                0             0         0    207.0  126.5   80.0  19.71
4235                0             0         0    248.0  131.0   72.0  22.00
4236                0             0         0    210.0  126.5   87.0  19.16
4237                0             0         0    269.0  133.5   83.0  21.47

      heartRate     glucose
0          80.0   77.000000
1          95.0   76.000000
2          75.0   70.000000
3          65.0  103.000000
4          85.0   85.000000
```

```
…           …          …
4233       66.0    86.000000
4234       65.0    68.000000
4235       84.0    86.000000
4236       86.0    81.966753
4237       80.0   107.000000

[4238 rows x 15 columns]
```

2 Train Test Split

[21]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
↪2,random_state=42)
```

[22]:
```
y_train
```

[22]:
```
3252    0
3946    0
1261    0
2536    0
4089    0
       ..
3444    0
466     0
3092    0
3772    0
860     0
Name: TenYearCHD, Length: 3390, dtype: int64
```

3 Logistic Regression Algorithm

[23]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(x_train,y_train)
model.score(x_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 2
      1 from sklearn.linear_model import LogisticRegression
----> 2 model = LogisticRegression().fit(x_train,y_train)
      3 model.score(x_train, y_train)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:1151, in _fit_context.
↪<locals>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1144     estimator._validate_params()
```

```
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:1207, in LogisticRegression.fit(self, X, y, sample_weight)
   1204 else:
   1205     _dtype = [np.float64, np.float32]
-> 1207 X, y = self._validate_data(
   1208     X,
   1209     y,
   1210     accept_sparse="csr",
   1211     dtype=_dtype,
   1212     order="C",
   1213     accept_large_sparse=solver not in ["liblinear", "sag", "saga"],
   1214 )
   1215 check_classification_targets(y)
   1216 self.classes_ = np.unique(y)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:621, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_params)
    619         y = check_array(y, input_name="y", **check_y_params)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1147, in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but the target y is None"
   1145     )
-> 1147 X = check_array(
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
```

```
   1155     ensure_2d=ensure_2d,
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=estimator)
   1165 check_consistent_length(X, y)

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:959, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    953         raise ValueError(
    954             "Found array with dim %d. %s expected <= 2."
    955             % (array.ndim, estimator_name)
    956         )
    958     if force_all_finite:
--> 959         _assert_all_finite(
    960             array,
    961             input_name=input_name,
    962             estimator_name=estimator_name,
    963             allow_nan=force_all_finite == "allow-nan",
    964         )
    966 if ensure_min_samples > 0:
    967     n_samples = _num_samples(array)

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:124, in _assert_all_finite(X, allow_nan, msg_dtype, estimator_name, input_name)
    121 if first_pass_isfinite:
    122     return
--> 124 _assert_all_finite_element_wise(
    125     X,
    126     xp=xp,
    127     allow_nan=allow_nan,
    128     msg_dtype=msg_dtype,
    129     estimator_name=estimator_name,
    130     input_name=input_name,
    131 )

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:173, in _assert_all_finite_element_wise(X, xp, allow_nan, msg_dtype, estimator_name, input_name)
    156 if estimator_name and input_name == "X" and has_nan_error:
    157     # Improve the error message on how to handle missing values in
    158     # scikit-learn.
```

```
    159     msg_err += (
    160         f"\n{estimator_name} does not accept missing values"
    161         " encoded as NaN natively. For supervised learning, you might␣
  ↪want"
   (…)
    171         "#estimators-that-handle-nan-values"
    172     )
--> 173 raise ValueError(msg_err)

ValueError: Input X contains NaN.
LogisticRegression does not accept missing values encoded as NaN natively. For␣
  ↪supervised learning, you might want to consider sklearn.ensemble.
  ↪HistGradientBoostingClassifier and Regressor which accept missing values␣
  ↪encoded as NaNs natively. Alternatively, it is possible to preprocess the␣
  ↪data, for instance by using an imputer transformer in a pipeline or drop␣
  ↪samples with missing values. See https://scikit-learn.org/stable/modules/
  ↪impute.html You can find a list of all estimators that handle NaN values at␣
  ↪the following page: https://scikit-learn.org/stable/modules/impute.
  ↪html#estimators-that-handle-nan-values
```