lab08

April 29, 2024

[ ]:
```
import pandas as pd
f = pd.read_csv("diabetes.csv")
```

[ ]:
```
f.head()
```

[ ]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

Types of Logistic Regression * **Binary Logistic Regression:** The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer. * **Multinomial Logistic Regression:** The target variable has three or more nominal categories such as predicting the type of Wine. * **Ordinal Logistic Regression:** the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

[ ]:
```
#split dataset in features and target variable
feature_cols = ['Pregnancies', 'Insulin', 'BMI',␣
 ↪'Age','Glucose','BloodPressure','DiabetesPedigreeFunction']
X = f[feature_cols] # Features
y = f.Outcome # Target variable
```

[ ]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=16)
```

[ ]:
```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16)

# fit the model with data
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
# import the necessary libraries
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True)

# split the train and test dataset
X_train, X_test,\
        y_train, y_test = train_test_split(X, y,
                                                                    test_size=0.
 ↪20,
                                                                    random_state=23)
# LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)

# Prediction
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

```
Logistic Regression model accuracy (in %): 95.6140350877193

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
from sklearn.model_selection import train_test_split

f1 = f.loc[:, f.columns != "Outcome"]

x_train, x_test,\
    y_train, y_test = train_test_split(f1, f.Outcome ,test_size=0.
 ↪20,random_state=0)
```

[ ]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True)

# split the train and test dataset
X_train, X_test,\
        y_train, y_test = train_test_split(X, y,
                                                                test_size=0.
 ↪20,
                                                                random_state=23)
# LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)

# Prediction
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

```
Logistic Regression model accuracy (in %): 95.6140350877193

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

clf = LogisticRegression(random_state=0)
clf.fit(x_train, y_train)

# Prediction
y_pred = clf.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

```
Logistic Regression model accuracy (in %): 82.46753246753246

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[ ]:
```
f.describe()
```

[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	
std	3.369578	31.972618	19.355807	15.952218	115.244002	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000

```
mean    31.992578                 0.471876    33.240885    0.348958
std      7.884160                 0.331329    11.760232    0.476951
min      0.000000                 0.078000    21.000000    0.000000
25%     27.300000                 0.243750    24.000000    0.000000
50%     32.000000                 0.372500    29.000000    0.000000
75%     36.600000                 0.626250    41.000000    1.000000
max     67.100000                 2.420000    81.000000    1.000000
```

[ ]:

[ ]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
model_score= model.score (X_test, y_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-93-2c6c475383bd> in <cell line: 2>()
      1 model = LogisticRegression()
----> 2 model.fit(X_train, y_train)
      3 y_predict = model.predict(X_test)
      4 model_score= model.score (X_test, y_test)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py in␣
 ↪fit(self, X, y, sample_weight)
   1194             _dtype = [np.float64, np.float32]
   1195
-> 1196         X, y = self._validate_data(
   1197             X,
   1198             y,

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(self,
 ↪X, y, reset, validate_separately, **check_params)
    582                 y = check_array(y, input_name="y", **check_y_params)
    583             else:
--> 584                 X, y = check_X_y(X, y, **check_params)
    585             out = X, y
    586

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py 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)
   1122     y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric,␣
 ↪estimator=estimator)
   1123
-> 1124     check_consistent_length(X, y)
```

```
   1125
   1126     return X, y

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
 ↪check_consistent_length(*arrays)
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:
--> 397         raise ValueError(
    398             "Found input variables with inconsistent numbers of samples␣
 ↪%r"
    399             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [455,␣
 ↪614]
```

[ ]:
```
from sklearn.model_selection import train_test_split
X= f.drop("Outcome", axis=1)
y= f[["Outcome"]]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30,␣
 ↪random_state=7)
```

[ ]:
```
from sklearn import metrics
```

[ ]:
```
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  warnings.warn(
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-84-3c219ab0becc> in <cell line: 1>()
----> 1 y_pred_proba = logreg.predict_proba(X_test)[::,1]
      2 fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
      3 auc = metrics.roc_auc_score(y_test, y_pred_proba)
      4 plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
      5 plt.legend(loc=4)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py in␣
 ↪predict_proba(self, X)
   1370         )
```

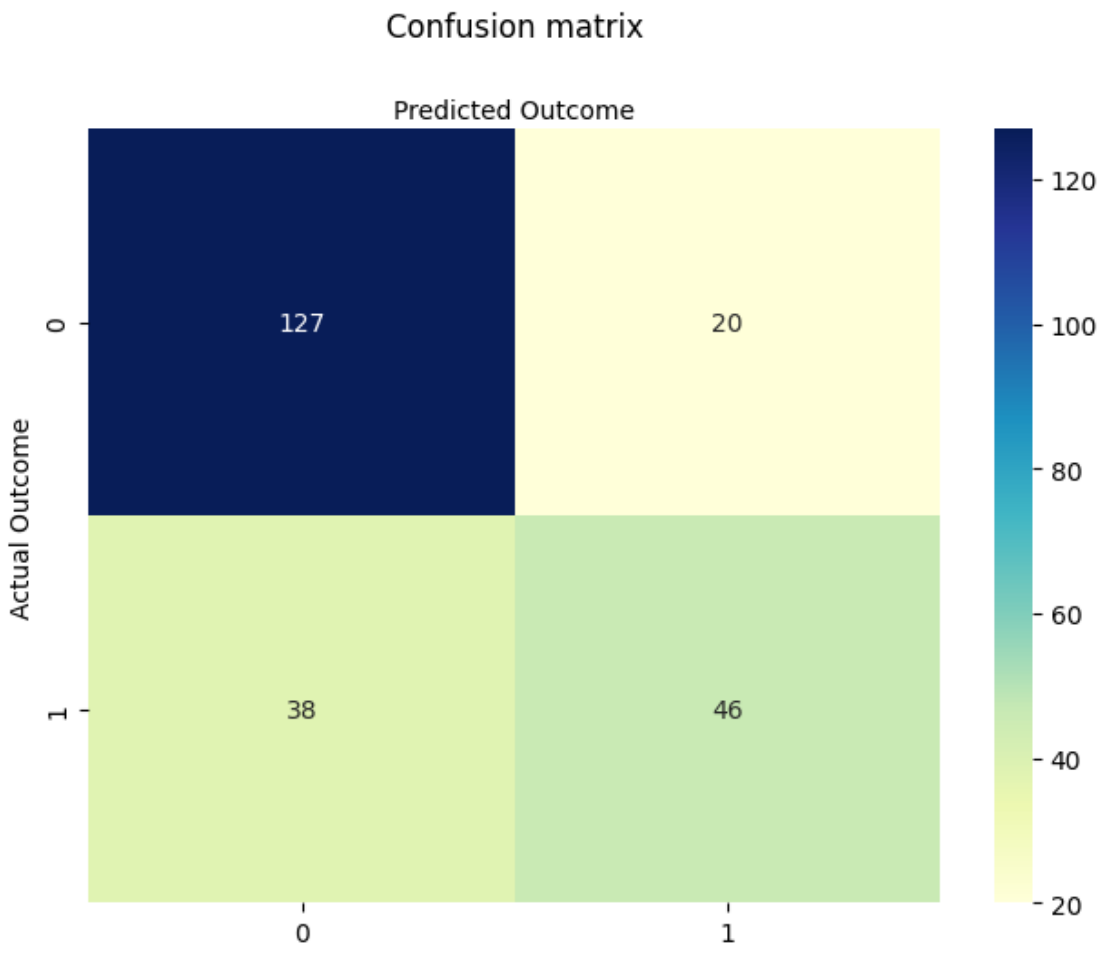
```
   1371         if ovr:
-> 1372             return super()._predict_proba_lr(X)
   1373         else:
   1374             decision = self.decision_function(X)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_base.py in␣
 ↪_predict_proba_lr(self, X)
    432         multiclass is handled by normalizing that over all classes.
    433         """
--> 434         prob = self.decision_function(X)
    435         expit(prob, out=prob)
    436         if prob.ndim == 1:

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_base.py in␣
 ↪decision_function(self, X)
    398         xp, _ = get_namespace(X)
    399
--> 400         X = self._validate_data(X, accept_sparse="csr", reset=False)
    401         scores = safe_sparse_dot(X, self.coef_.T, dense_output=True) +␣
 ↪self.intercept_
    402         return xp.reshape(scores, -1) if scores.shape[1] == 1 else scor s

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(self,
 ↪X, y, reset, validate_separately, **check_params)
    586
    587         if not no_val_X and check_params.get("ensure_2d", True):
--> 588             self._check_n_features(X, reset=reset)
    589
    590         return out

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in␣
 ↪_check_n_features(self, X, reset)
    387
    388         if n_features != self.n_features_in_:
--> 389             raise ValueError(
    390                 f"X has {n_features} features, but {self.__class__.
 ↪__name__} "
    391                 f"is expecting {self.n_features_in_} features as input.

ValueError: X has 30 features, but LogisticRegression is expecting 7 features a␣
 ↪input.
```

[ ]:
```
# print(model_score)
cnf_matrix = metrics.confusion_matrix(y_test, y_predict)
print(cnf_matrix)
```

[ ]:
```
import numpy as np
```

[ ]:
```
class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual Outcome')
plt.xlabel('Predicted Outcome')
```

[ ]:
```
Text(0.5, 427.9555555555555, 'Predicted Outcome')
```



[ ]:
```
true_neg, false_pos, false_neg, true_pos = cnf_matrix.ravel()
true_neg, false_pos, false_neg, true_pos
total = true_neg + false_pos + false_neg + true_pos

accuracy = (true_pos + true_neg)/total
print(accuracy)

precision = true_pos/(true_pos + false_pos)
print(precision)

recall = true_pos/(true_pos + false_neg)
print(recall)

f1_score = (2*precision*recall)/(precision+recall)
print(f1_score)
```

```
0.7489177489177489
0.696969696969697
0.5476190476190477
0.6133333333333334
```