LOGISTIC REGRESSION CENTRAL TEST

March 18, 2024

Ordinary Logistic Regrassion

[12]:
```
#import necessary libraries
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
```

[13]:
```
#load data set
iris_data = pd.read_csv("C:\\Users\\DELL\\Desktop\\iris.csv")
iris_data
```

[13]:

	x0	x1	x2	x3	x4	type
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	1	4.9	3.0	1.4	0.2	Iris-setosa
2	1	4.7	3.2	1.3	0.2	Iris-setosa
3	1	4.6	3.1	1.5	0.2	Iris-setosa
4	1	5.0	3.6	1.4	0.2	Iris-setosa
..	..	…	…	…	…	…
145	1	6.7	3.0	5.2	2.3	Iris-virginica
146	1	6.3	2.5	5.0	1.9	Iris-virginica
147	1	6.5	3.0	5.2	2.0	Iris-virginica
148	1	6.2	3.4	5.4	2.3	Iris-virginica
149	1	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 6 columns]

[14]:
```
iris_x=iris_data.drop(["type"],axis=1)
iris_x
```

[14]:

	x0	x1	x2	x3	x4
0	1	5.1	3.5	1.4	0.2
1	1	4.9	3.0	1.4	0.2
2	1	4.7	3.2	1.3	0.2
3	1	4.6	3.1	1.5	0.2
4	1	5.0	3.6	1.4	0.2
..	..	…	…	…	…
145	1	6.7	3.0	5.2	2.3

```
146   1  6.3  2.5  5.0  1.9
147   1  6.5  3.0  5.2  2.0
148   1  6.2  3.4  5.4  2.3
149   1  5.9  3.0  5.1  1.8

[150 rows x 5 columns]
```

[15]:
```
iris_y=iris_data["type"]
iris_y
```

[15]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             …
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: type, Length: 150, dtype: object
```

[16]:
```
#Training and building the model
from sklearn.model_selection import train_test_split
iris_x_train,iris_x_test,iris_y_train,iris_y_test=train_test_split(iris_x,iris_y,test_size=0.
 ↪2,random_state=42)
iris_x_train.shape
```

[16]:
```
(120, 5)
```

[17]:
```
import warnings
warnings.filterwarnings('ignore')
iris_model=LogisticRegression()
iris_model.fit(iris_x_train,iris_y_train)
```

[17]:
```
LogisticRegression()
```

[18]:
```
#Making predictions
iris_y_pred=iris_model.predict(iris_x_test)
iris_y_pred
```

[18]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
```

```
        'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
        'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
        'Iris-virginica', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

[19]:
```
#Evaluation
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
# Calculate evaluation metrics for iris dataset
iris_accuracy = accuracy_score(iris_y_test, iris_y_pred)
iris_precision = precision_score(iris_y_test, iris_y_pred, average='weighted',␣
 ↪labels=np.unique(iris_y_pred))
iris_recall = recall_score(iris_y_test, iris_y_pred, average='weighted',␣
 ↪labels=np.unique(iris_y_pred))
iris_f1 = f1_score(iris_y_test, iris_y_pred, average='weighted', labels=np.
 ↪unique(iris_y_pred))

print("Metrics for Iris dataset:")
print("Accuracy:", iris_accuracy)
print("Precision:", iris_precision)
print("Recall:", iris_recall)
print("F1 Score:", iris_f1)
```

```
Metrics for Iris dataset:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

[20]:
```
#model evaluation
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
```

[21]:
```
#Defining the model
iris_model=LogisticRegression()
iris_model
```

[21]:
```
LogisticRegression()
```

[22]:
```
#defining the params
param_grid={
     "penalty":['l1','l2'],
    'C':[0.001,0.01,0.1,1,10,100],
    'fit_intercept':[True,False],
    'solver':['libliinear','saga']
}
param_grid
```

[22]:
```
{'penalty': ['l1', 'l2'],
 'C': [0.001, 0.01, 0.1, 1, 10, 100],
 'fit_intercept': [True, False],
 'solver': ['libliinear', 'saga']}
```

[23]:
```
#initializing grid search
grid_search=GridSearchCV(iris_model,param_grid,cv=5)
grid_search
```

[23]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
                         'fit_intercept': [True, False],
                         'penalty': ['l1', 'l2'],
                         'solver': ['libliinear', 'saga']})
```

[24]:
```
#Fitting the model with the train values
grid_search.fit(iris_x_train,iris_y_train)
```

[24]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
                         'fit_intercept': [True, False],
                         'penalty': ['l1', 'l2'],
                         'solver': ['libliinear', 'saga']})
```

[25]:
```
#defining best parameters
best_params=grid_search.best_params_
best_params
```

[25]:
```
{'C': 1, 'fit_intercept': True, 'penalty': 'l1', 'solver': 'saga'}
```

[26]:
```
 #Initializing LogisticRegression using best_params
best_iris_model=LogisticRegression(**best_params)
best_iris_model
```

[26]:
```
LogisticRegression(C=1, penalty='l1', solver='saga')
```

[27]:
```
#Fitting the best_iris_model with train vvalues
best_iris_model.fit(iris_x_train,iris_y_train)
```

[27]:
```
LogisticRegression(C=1, penalty='l1', solver='saga')
```

[28]:
```
#making predictions
iris_y_pred=best_iris_model.predict(iris_x_test)
iris_y_pred
```

[28]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
```

```
        'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
        'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
        'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
        'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
        'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
        'Iris-virginica', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

[29]:
```
#Evaluating the model
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
# Calculate evaluation metrics for iris dataset
iris_accuracy = accuracy_score(iris_y_test, iris_y_pred)
iris_precision = precision_score(iris_y_test, iris_y_pred, average='weighted',␣
 ↪labels=np.unique(iris_y_pred))
iris_recall = recall_score(iris_y_test, iris_y_pred, average='weighted',␣
 ↪labels=np.unique(iris_y_pred))
iris_f1 = f1_score(iris_y_test, iris_y_pred, average='weighted', labels=np.
 ↪unique(iris_y_pred))

print("Metrics for Iris dataset:")
print("Accuracy:", iris_accuracy)
print("Precision:", iris_precision)
print("Recall:", iris_recall)
print("F1 Score:", iris_f1)
```

```
Metrics for Iris dataset:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

[ ]: