chap10 - machineLearning2

June 25, 2022

[2]:
```
# import libraries
import pandas as pd

# read the dataset
diabetes = pd.read_csv("diabetes.csv")

# show top 5-records
diabetes.head()
```

[2]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[3]:
```
# split dataset in two parts: feature set and target set
feature_set = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigree']
features = diabetes[feature_set]
target = diabetes.label

# partition data into training and testing set
from sklearn.model_selection import train_test_split

feature_train, feature_test, target_train, target_test = \
train_test_split(features, target, test_size=0.3, random_state=1)
```

[10]:
```
## import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
model = GaussianNB()

# train the model using the training sets
model.fit(feature_train, target_train)

# forecast the target variable for given test dataset
```

```
predictions = model.predict(feature_test)
```

[12]:
```
# import metrics module for performance evaluation
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

# calculate model accuracy
print("Accuracy: ", accuracy_score(target_test, predictions))

# calculate model precision
print("Precision: ", precision_score(target_test, predictions))

# calculate model recall
print("Recall: ", recall_score(target_test, predictions))

# calculate model f1 score
print("F1-Score: ", f1_score(target_test, predictions))
```

```
Accuracy:  0.7748917748917749
Precision:  0.7391304347826086
Recall:  0.6
F1-Score:  0.6623376623376623
```

[15]:
```
## import Decision Tree model
from sklearn.tree import DecisionTreeClassifier

# create a Decision Tree classifier object
clf = DecisionTreeClassifier()

# train the model using training dataset
clf = clf.fit(feature_train, target_train)

# predict the response for test dataset
predictions = clf.predict(feature_test)
```

[16]:
```
# calculate model accuracy
print("Accuracy: ", accuracy_score(target_test, predictions))

# calculate model prediction
print("Prediction: ", precision_score(target_test, predictions))

# calculate model recall
print("Recall: ", recall_score(target_test, predictions))

# calculate model f1 score
```

```
print("F1-Score: ", f1_score(target_test, predictions))
```

```
Accuracy:  0.683982683982684
Prediction:  0.5769230769230769
Recall:  0.5294117647058824
F1-Score:  0.5521472392638036
```

[4]:
```
## import KNN model
from sklearn.neighbors import KNeighborsClassifier

# create a KNN classifier object
model = KNeighborsClassifier(n_neighbors=3)

# train the model using the training dataset
model.fit(feature_train, target_train)

# predict the target variable for test dataset
predictions = model.predict(feature_test)
```

[5]:
```
# import metrics module for performance evaluation
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

# calculate model accuracy
print("Accuracy: ", accuracy_score(target_test, predictions))

# calculate model precision
print("Precision: ", precision_score(target_test, predictions))

# calculate model recall
print("Recall: ", recall_score(target_test, predictions))

# calculate model f1 score
print("F1-Score: ", f1_score(target_test, predictions))
```

```
Accuracy:  0.7532467532467533
Precision:  0.7058823529411765
Recall:  0.5647058823529412
F1-Score:  0.6274509803921569
```

[7]:
```
## import SVM classification

# import SVM model
from sklearn import svm
```

```
# create a SVM classifier object
clf = svm.SVC(kernel='linear')

# train the model using the training sets
clf.fit(feature_train, target_train)

# predict the target variable for test dataset
predictions = clf.predict(feature_test)
```

[8]:
```
# calculate model accuracy
print("Accuracy: ", accuracy_score(target_test, predictions))

# calculate model precision
print("Precision: ", precision_score(target_test, predictions))

# calculate model recall
print("Recall: ", recall_score(target_test, predictions))

# calculate model f1 score
print("F1-Score: ", f1_score(target_test, predictions))
```

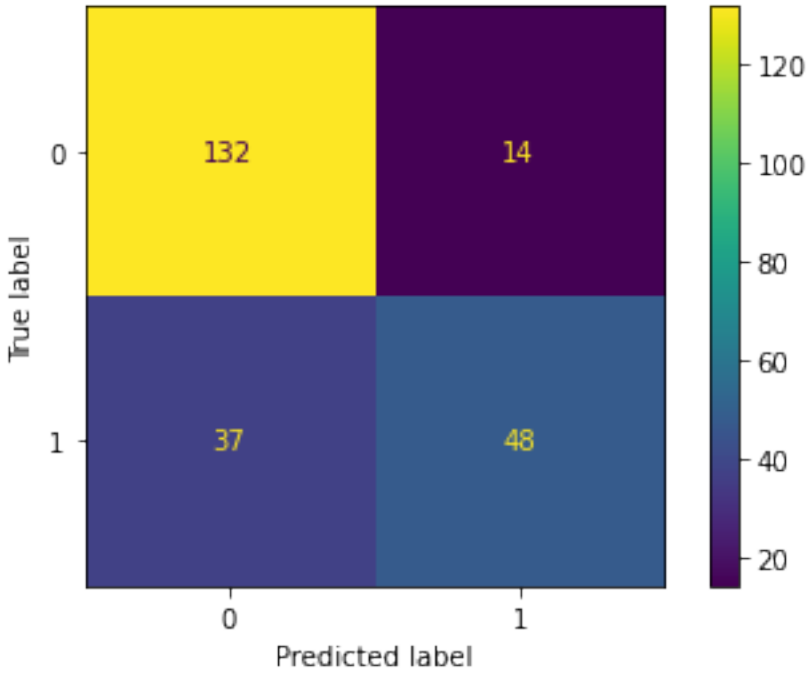
```
Accuracy:  0.7835497835497836
Precision:  0.7868852459016393
Recall:  0.5647058823529412
F1-Score:  0.6575342465753424
```

[9]:
```
# import libs
import pandas as pd

# read the dataset
diabets = pd.read_csv("diabetes.csv")

# split dataset in two parts: feature set and target label
feature_set = ['pregnant', 'bmi', 'age', 'glucose', 'bp', 'pedigree']
features = diabets[feature_set]

target = diabetes.label

# partition data into training and testing set
from sklearn.model_selection import train_test_split
feature_train, feature_test, target_train, target_test = \
train_test_split(features, target, test_size=0.3, random_state=1)

# import logistic regression scikit-learn model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score # from performance evaluation
```

```
# instantiate the model
logreg = LogisticRegression(solver='lbfgs')

# fit the model with data
logreg.fit(feature_train, target_train)

# forcast the target variable for given test dataset
predictions = logreg.predict(feature_test)

# get prediction probability
predictions_prob = logreg.predict_proba(feature_test)[::,1]

# import the confusion matrix
from sklearn.metrics import plot_confusion_matrix

# plot Confusion matrix
plot_confusion_matrix(logreg, feature_test, target_test, values_format='d')
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

[9]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x294fd917c10>



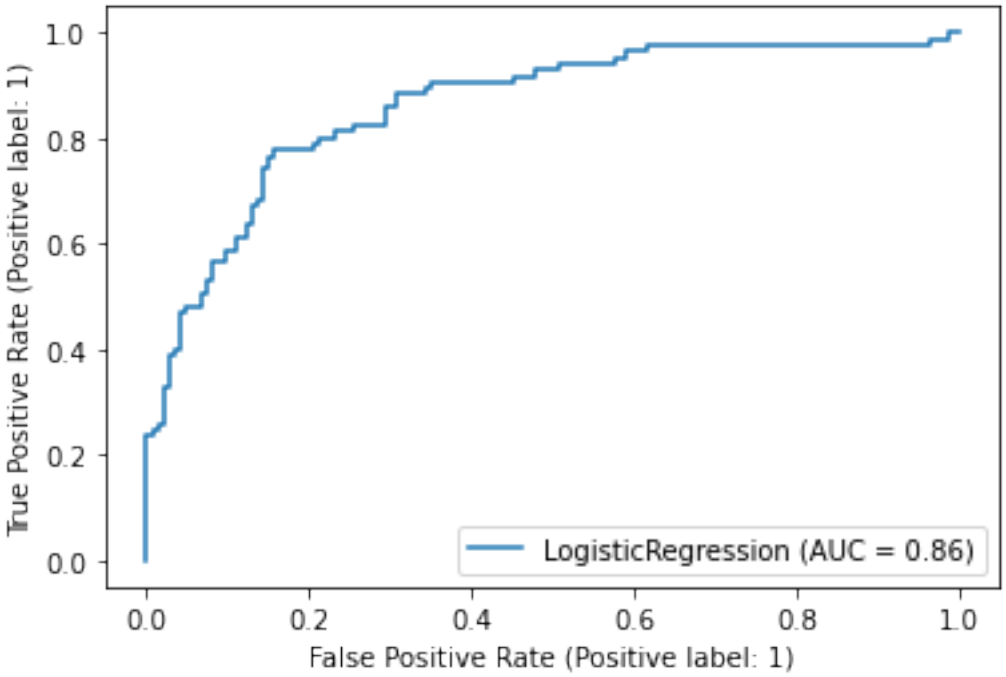
[10]:
```
# import classification report
from sklearn.metrics import classification_report

# create classification report
print(classification_report(target_test, predictions, target_names=['Yes(1)',␣
 ↪'No(0)']))
```

```
              precision    recall  f1-score   support

     Yes(1)       0.78      0.90      0.84       146
      No(0)       0.77      0.56      0.65        85

    accuracy                           0.78       231
   macro avg       0.78      0.73      0.75       231
weighted avg       0.78      0.78      0.77       231
```

[11]:
```
# import plot_roc_curve
from sklearn.metrics import plot_roc_curve

plot_roc_curve(logreg, feature_test, target_test)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87:
FutureWarning: Function plot_roc_curve is deprecated; Function
:func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: :meth:`sklearn.metric.RocCurveDisplay.from_predictions` or
:meth:`sklearn.metric.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

[11]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x29486c48af0>
```



[12]:
```
# import ROC AUC score
from sklearn.metrics import roc_auc_score

# compute the area under ROC curve
auc = roc_auc_score(target_test, predictions_prob)

# print auc value
print("Area Under Curve: ", auc)
```

```
Area Under Curve:  0.8614826752618856
```

[ ]: