a27-exp05

October 23, 2024

[24]:
```
Name:Viraj Mulik
Class:
```

[24]:
```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import␣
 ↪accuracy_score,confusion_matrix,classification_report,r2_score,precision_score
import matplotlib.pyplot as plt
```

[25]:
```
data= load_breast_cancer()
```

[26]:
```
df=pd.DataFrame(data = data.data, columns = data.feature_names)
target=pd.Series(data=data.target,name='target')
```

[27]:
```
df.head(5)
```

[27]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

	mean fractal dimension	…	worst radius	worst texture	worst perimeter	\
0	0.07871	…	25.38	17.33	184.60	
1	0.05667	…	24.99	23.41	158.80	
2	0.05999	…	23.57	25.53	152.50	
3	0.09744	…	14.91	26.50	98.87	

```
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[28]:
```
target
```

[28]:
```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: target, Length: 569, dtype: int64
```

[29]:
```
#Task 2: Split the Dataset into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(df,target,␣
 ↪random_state=42,test_size=0.3)
```

[30]:
```
#Task 3: Standardize the Features
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
standard_data=scaler.fit_transform(df)
print(standard_data)
```

```
[[ 1.09706398 -2.07333501  1.26993369 …  2.29607613  2.75062224
   1.93701461]
 [ 1.82982061 -0.35363241  1.68595471 …  1.0870843  -0.24388967
   0.28118999]
```

```
 [ 1.57988811  0.45618695  1.56650313 …  1.95500035  1.152255
   0.20139121]
 …
 [ 0.70228425  2.0455738   0.67267578 …  0.41406869 -1.10454895
  -0.31840916]
 [ 1.83834103  2.33645719  1.98252415 …  2.28998549  1.91908301
   2.21963528]
 [-1.80840125  1.22179204 -1.81438851 … -1.74506282 -0.04813821
  -0.75120669]]
```

[44]:
```
#Task 4: Train the KNN Classifier
knn=KNeighborsClassifier(7)
knn.fit(X_train,y_train)
```

[44]:
```
KNeighborsClassifier(n_neighbors=7)
```

[45]:
```
#Task 5: Make Predictions and Evaluate the Model
y_pred=knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9649122807017544
```

[46]:
```
confusion = confusion_matrix(y_test, y_pred)
print(confusion)
```

```
[[ 59   4]
 [  2 106]]
```

[47]:
```
report = classification_report(y_test, y_pred)
print("Report:", report)
```

```
Report:               precision    recall  f1-score   support

           0       0.97      0.94      0.95        63
           1       0.96      0.98      0.97       108

    accuracy                           0.96       171
   macro avg       0.97      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

[ ]:
```

```

```
-0.75120669]]
```

[40]:
```
#Task 4: Train the KNN Classifier
knn=KNeighborsClassifier(3)
knn.fit(X_train,y_train)
```

[40]:
```
KNeighborsClassifier(n_neighbors=3)
```

[41]:
```
#Task 5: Make Predictions and Evaluate the Model
y_pred=knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9415204678362573
```

[42]:
```
confusion = confusion_matrix(y_test, y_pred)
print(confusion)
```

```
[[ 57   6]
 [  4 104]]
```

[43]:
```
report = classification_report(y_test, y_pred)
print("Report:", report)
```

```
Report:               precision    recall  f1-score   support

           0       0.93      0.90      0.92        63
           1       0.95      0.96      0.95       108

    accuracy                           0.94       171
   macro avg       0.94      0.93      0.94       171
weighted avg       0.94      0.94      0.94       171
```

[31]:
```
#Task 4: Train the KNN Classifier
knn=KNeighborsClassifier(5)
knn.fit(X_train,y_train)
```

[31]:
```
KNeighborsClassifier()
```

[32]:
```
#Task 5: Make Predictions and Evaluate the Model
y_pred=knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9590643274853801
```

[38]:
```
confusion = confusion_matrix(y_test, y_pred)
print(confusion)
```

```
[[ 57   6]
 [  1 107]]
```

[39]:
```
report = classification_report(y_test, y_pred)
print("Report:", report)
```

```
Report:               precision    recall  f1-score   support

           0       0.98      0.90      0.94        63
           1       0.95      0.99      0.97       108

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

[ ]:

```
=0.75120669]]
```

[44]:
```
#Task 4: Train the KNN Classifier
knn=KNeighborsClassifier(7)
knn.fit(X_train,y_train)
```

[44]:
```
KNeighborsClassifier(n_neighbors=7)
```

[45]:
```
#Task 5: Make Predictions and Evaluate the Model
y_pred=knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9649122807017544
```

[46]:
```
confusion = confusion_matrix(y_test, y_pred)
print(confusion)
```

```
[[ 59   4]
 [  2 106]]
```

[47]:
```
report = classification_report(y_test, y_pred)
print("Report:", report)
```

```
Report:               precision    recall  f1-score   support

           0       0.97      0.94      0.95        63
           1       0.96      0.98      0.97       108

    accuracy                           0.96       171
   macro avg       0.97      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

[ ]: