ECGR_4105_HW3_Problem_4_TBCPDF

October 17, 2024

[2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[3]:

```
file_url = 'https://raw.githubusercontent.com/lnguye782/ECGR-4105-Intro-to-ML/
↪refs/heads/main/HW3/cancer.csv'
data = pd.read_csv(file_url)

data.head()
```

[3]:

```
   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 texture  worst perimeter  worst area  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625
```

```
   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

[4]:
```
# Separate features and target variable (30 input features / 1 output target)
X = data.drop(columns=['target'])
Y = data['target']
```

[5]:
```
# Split the data set into Training Data (80%) and Test Data (20%)
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)
```

[6]:
```
# Scale the data between 0 and 1 to get better accuracy
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

[29]:
```
# Use PCA feature extraction for training
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

accuracy_scores = []
precision_scores = []
recall_scores = []
f1_scores = []

# Maximum number of features to consider
K_max = 30

for K in range(1, K_max + 1):
    # Apply PCA to reduce the dimensionality to K components
    pca = PCA(n_components=K)
    X_train_pca = pca.fit_transform(X_train)
    X_test_pca = pca.transform(X_test)

    # Train a logistic regression model with the reduced features
```

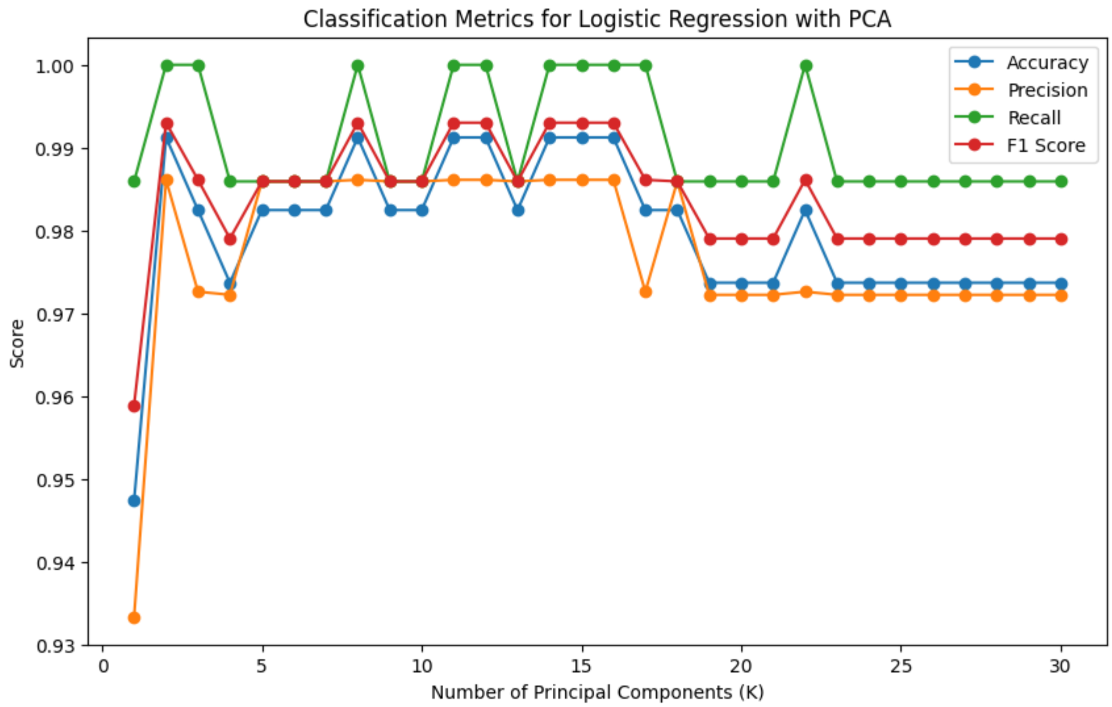
```
    classifier_with_pca = LogisticRegression(max_iter=10000)
    classifier_with_pca.fit(X_train_pca, Y_train)

    # Make predictions on the test set
    Y_pred_pca = classifier_with_pca.predict(X_test_pca)

    # Evaluate the model with PCA using model evaluation metrics: accuracy,␣
  ↪precision, recall, and F1 score
    accuracy_scores.append(metrics.accuracy_score(Y_test, Y_pred_pca))
    precision_scores.append(metrics.precision_score(Y_test, Y_pred_pca))
    recall_scores.append(metrics.recall_score(Y_test, Y_pred_pca))
    f1_scores.append(metrics.f1_score(Y_test, Y_pred_pca))
```

[30]:
```
# Identify the optimal number of principal components (K) for highest accuracy
optimal_K = accuracy_scores.index(max(accuracy_scores)) + 1
optimal_K, max(accuracy_scores)
```

[30]: (2, 0.9912280701754386)

[31]:
```
# Plot your classification accuracy, precision, recall, and F1 score over a␣
  ↪different number of Ks
plt.figure(figsize=(10, 6))
plt.plot(range(1, K_max + 1), accuracy_scores, label='Accuracy', marker='o')
plt.plot(range(1, K_max + 1), precision_scores, label='Precision', marker='o')
plt.plot(range(1, K_max + 1), recall_scores, label='Recall', marker='o')
plt.plot(range(1, K_max + 1), f1_scores, label='F1 Score', marker='o')
plt.title('Classification Metrics for Logistic Regression with PCA')
plt.xlabel('Number of Principal Components (K)')
plt.ylabel('Score')
plt.legend()
```

[31]: <matplotlib.legend.Legend at 0x7fc1f2978c10>


Classification Metrics for Logistic Regression with PCA
Accuracy
Precision
Recall
F1 Score
Score
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
5
10
15
20
25
30
Number of Principal Components (K)