pca-ipython-single-file

August 21, 2024

[21]:
```
#To modify the code to use PCA instead of `SelectKBest`, you'll need to replace␣
 ↪the `selectkbest` function with a function that performs PCA, and then use␣
 ↪the PCA-transformed features for training and testing the classifiers.

#python
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt


# Replace SelectKBest with PCA
def apply_pca(indep_X, n_components):
    pca = PCA(n_components=n_components)
    pca_features = pca.fit_transform(indep_X)
    return pca_features

def split_scalar(indep_X, dep_Y):
    X_train, X_test, y_train, y_test = train_test_split(indep_X, dep_Y,␣
 ↪test_size=0.25, random_state=0)
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    return X_train, X_test, y_train, y_test

def cm_prediction(classifier, X_test, y_test):
    y_pred = classifier.predict(X_test)

    from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
    cm = confusion_matrix(y_test, y_pred)
    Accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    return classifier, Accuracy, report, X_test, y_test, cm

def logistic(X_train, y_train, X_test, y_test):
    classifier = LogisticRegression(random_state=0)
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm = cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm

def svm_linear(X_train, y_train, X_test, y_test):
    from sklearn.svm import SVC
    classifier = SVC(kernel='linear', random_state=0)
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm = cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm

def svm_nl(X_train, y_train, X_test, y_test):
    from sklearn.svm import SVC
    classifier = SVC(kernel='rbf', random_state=0)
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm = cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm

def naive(X_train, y_train, X_test, y_test):
    from sklearn.naive_bayes import GaussianNB
    classifier = GaussianNB()
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm = cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm

def knn(X_train, y_train, X_test, y_test):
    from sklearn.neighbors import KNeighborsClassifier
    classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm = cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm

def decision(X_train, y_train, X_test, y_test):
    from sklearn.tree import DecisionTreeClassifier
    classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm = cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm

def random(X_train, y_train, X_test, y_test):
    from sklearn.ensemble import RandomForestClassifier
    classifier = RandomForestClassifier(n_estimators=10, criterion='entropy',␣
 ↪random_state=0)
    classifier.fit(X_train, y_train)
    classifier, Accuracy, report, X_test, y_test, cm =␣
 ↪cm_prediction(classifier, X_test, y_test)
    return classifier, Accuracy, report, X_test, y_test, cm


def pca_classification(acclog, accsvml, accsvmnl, accknn, accnav, accdes,␣
 ↪accrf):
    dataframe = pd.DataFrame(index=['PCA'], columns=['Logistic', 'SVMl',␣
 ↪'SVMnl', 'KNN', 'Naive', 'Decision', 'Random'])
    for number, idex in enumerate(dataframe.index):
        dataframe['Logistic'][idex] = acclog[number]
        dataframe['SVMl'][idex] = accsvml[number]
        dataframe['SVMnl'][idex] = accsvmnl[number]
        dataframe['KNN'][idex] = accknn[number]
        dataframe['Naive'][idex] = accnav[number]
        dataframe['Decision'][idex] = accdes[number]
        dataframe['Random'][idex] = accrf[number]
    return dataframe
```

[25]:

```
# Load dataset
dataset1 = pd.read_csv("prep.csv", index_col=None)
df2 = pd.get_dummies(dataset1, drop_first=True)

indep_X = df2.drop('classification_yes', axis=1)
dep_Y = df2['classification_yes']
```

[54]:

```
# Apply PCA
pca_features = apply_pca(indep_X, n_components=10)

# Initialize lists to store accuracies
acclog = []
accsvml = []
accsvmnl = []
accknn = []
accnav = []
accdes = []
accrf = []
```

[56]:

```
# Split and scale data
X_train, X_test, y_train, y_test = split_scalar(pca_features, dep_Y)

# Logistic Regression
classifier, Accuracy, report, X_test, y_test, cm = logistic(X_train, y_train,␣
 ↪X_test, y_test)
acclog.append(Accuracy)

# SVM Linear
classifier, Accuracy, report, X_test, y_test, cm = svm_linear(X_train, y_train,␣
 ↪X_test, y_test)
accsvml.append(Accuracy)

# SVM Non-Linear (RBF)
classifier, Accuracy, report, X_test, y_test, cm = svm_nl(X_train, y_train,␣
 ↪X_test, y_test)
accsvmnl.append(Accuracy)

# KNN
classifier, Accuracy, report, X_test, y_test, cm = knn(X_train, y_train,␣
 ↪X_test, y_test)
accknn.append(Accuracy)

# Naive Bayes
classifier, Accuracy, report, X_test, y_test, cm = naive(X_train, y_train,␣
 ↪X_test, y_test)
accnav.append(Accuracy)

# Decision Tree
classifier, Accuracy, report, X_test, y_test, cm = decision(X_train, y_train,␣
 ↪X_test, y_test)
accdes.append(Accuracy)

# Random Forest
classifier, Accuracy, report, X_test, y_test, cm = random(X_train, y_train,␣
 ↪X_test, y_test)
accrf.append(Accuracy)

# Tabulate results
result = pca_classification(acclog, accsvml, accsvmnl, accknn, accnav, accdes,␣
 ↪accrf)

print(result)

### Key Changes:
#1. **PCA Replacement:**
```

```
   #- The `selectkbest` function has been replaced with an `apply_pca` function␣
 ↪that applies PCA to the dataset and returns the transformed features.

#2. **Function Name Adjustments:**
   #- The function `selectk_Classification` is renamed to `pca_classification`␣
 ↪to reflect the use of PCA.

#3. **Data Handling:**
   #- The rest of the workflow (splitting, scaling, and classifier training)␣
 ↪remains largely the same, but it now operates on the PCA-transformed data.

### Execution:
#- This code performs PCA on the input features, reduces them to a specified␣
 ↪number of components (`n_components=5`), and then uses these components to␣
 ↪train various classifiers.
#- Finally, it tabulates the accuracy results of each classifier and prints␣
 ↪them in a pandas DataFrame.

#You can modify `n_components` in `apply_pca` to change the number of principal␣
 ↪components used.
```

```
    Logistic  SVMl SVMnl   KNN Naive Decision Random
PCA     0.96  0.96  0.97  0.95  0.85     0.88   0.95
```

```
C:\Users\Kathirvel\Anaconda3\envs\aiml\lib\site-
packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Kathirvel\Anaconda3\envs\aiml\lib\site-
packages\sklearn\neighbors\base.py:441: DeprecationWarning: distutils Version
classes are deprecated. Use packaging.version instead.
  old_joblib = LooseVersion(joblib_version) < LooseVersion('0.12')
C:\Users\Kathirvel\Anaconda3\envs\aiml\lib\site-
packages\sklearn\neighbors\base.py:441: DeprecationWarning: distutils Version
classes are deprecated. Use packaging.version instead.
  old_joblib = LooseVersion(joblib_version) < LooseVersion('0.12')
C:\Users\Kathirvel\Anaconda3\envs\aiml\lib\site-
packages\sklearn\utils\fixes.py:230: DeprecationWarning: distutils Version
classes are deprecated. Use packaging.version instead.
  if _joblib.__version__ >= LooseVersion('0.12'):
C:\Users\Kathirvel\Anaconda3\envs\aiml\lib\site-
packages\sklearn\utils\fixes.py:230: DeprecationWarning: distutils Version
classes are deprecated. Use packaging.version instead.
  if _joblib.__version__ >= LooseVersion('0.12'):
```

[28]:

```
result
#5
```

```
[28]:     Logistic  SVMl SVMnl   KNN Naive Decision Random
      PCA     0.83  0.83  0.87  0.87  0.81     0.86   0.86
```

```
[31]: result
      #6
```

```
[31]:     Logistic  SVMl SVMnl   KNN Naive Decision Random
      PCA     0.86  0.87  0.91  0.88  0.83     0.87   0.88
```

```
[34]: result
      #4
```

```
[34]:     Logistic SVMl SVMnl   KNN Naive Decision Random
      PCA     0.83  0.8  0.84  0.81  0.77     0.87   0.85
```

```
[37]: result
      #3
```

```
[37]:     Logistic  SVMl SVMnl  KNN Naive Decision Random
      PCA     0.82  0.83  0.81  0.8  0.77     0.87   0.84
```

```
[41]: result
      #2
```

```
[41]:     Logistic  SVMl SVMnl   KNN Naive Decision Random
      PCA     0.75  0.77  0.76  0.79  0.73     0.76   0.77
```

```
[44]: result
      #7
```

```
[44]:     Logistic  SVMl SVMnl   KNN Naive Decision Random
      PCA     0.94  0.92  0.97  0.91  0.89     0.85   0.92
```

```
[47]: result
      #8
```

```
[47]:     Logistic  SVMl SVMnl   KNN Naive Decision Random
      PCA     0.94  0.94  0.97  0.94  0.86     0.88   0.93
```

```
[50]: result
      #9
```

```
[50]:     Logistic  SVMl SVMnl   KNN Naive Decision Random
      PCA     0.95  0.93  0.94  0.93  0.85     0.88   0.95
```

```
[53]: result
      #10
```

[53]:

```
    Logistic  SVMl SVMnl   KNN Naive Decision Random
PCA     0.96  0.96  0.97  0.95  0.85      0.88   0.95
```

[ ]:

```
#using 10 we can choose as best algorithm. we can take mode and get repeated␣
 ↪value as our final answer.
#we are putting input 5 or 6 into all algorithms and getting predictions. and␣
 ↪choosing which algorithm prediction is good.Finally if we mode we can use␣
 ↪repeated answer.
#fLOW DIAGRAM IS need to PREPARE.
```