Assignment-2_Machine_Learning_Valapadasu_UdayBhaskar

July 1, 2024

[142]:
```
#Assignment 2: Machine Learning
#Name: Uday Bhaskar Valapadasu
#ID: 11696364
```

[143]:
```
#Import Statements

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import warnings
# Suppress the FutureWarning
warnings.simplefilter(action='ignore', category=FutureWarning)
```

[144]:
```
# Using the diabetes_df.csv created from assignment - 1 & Created a Pandas␣
↪dataframe from diabetes_df.csv and named it assignment2_df

assignment2_df = pd.read_csv("diabetes_df.csv")
assignment2_df
```

[144]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35      150  33.6
1              1       85             66             29      150  26.6
2              8      183             64              0      150  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …        …              …             …       …    …
763           10      101             76             48      180  32.9
764            2      122             70             27      150  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0      150  30.1
767            1       93             70             31      150  30.4

     DiabetesPedigreeFunction  Age  Target
0                       0.627   50       1
1                       0.351   31       0
2                       0.672   32       1
3                       0.167   21       0
```

```
4                        2.288   33       1
..                         ...   ...    ...
763                      0.171   63       0
764                      0.340   27       0
765                      0.245   30       0
766                      0.349   47       1
767                      0.315   23       0

[768 rows x 9 columns]
```

[145]:
```
# Setup the Machine Learning Model:
#Dividing the data into features (X) array and target (y) array.

#features array
X = assignment2_df.drop(['Target'], axis=1)
#target array
y = assignment2_df['Target']
```

[146]:
```
# Splitting the dataset into 80-20, 70-30, and 60-40 ratios. (Example: 80-20␣
↪means, 80% training data, 20% testing data, and so on.)

#Split-1 into 80-20
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

#Split-2 into 70-30
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

#Split-3 into 60-40
X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.4,␣
↪random_state=42)
```

[147]:
```
# For each data split, apply logistic regression machine learning model to␣
↪build confusion matrix and accuracy estimates.
# So, importing the necessary accordinhly.

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
```

[148]:
```
# Performing Logistic Regression & Building Confusion Matrix and Accuracy for␣
↪Split-1 Training:Test(80:20) Ratio
lr_split_1 = LogisticRegression(random_state=42, max_iter=1000)
lr_split_1.fit(X_train1, y_train1)
y_pred1 = lr_split_1.predict(X_test1)
y_pred_proba1 = lr_split_1.predict_proba(X_test1)[:, 1]
cm_split_1 = confusion_matrix(y_test1, y_pred1)
```

```
accuracy_split_1 = accuracy_score(y_test1, y_pred1)
auc_split_1 = roc_auc_score(y_test1, y_pred_proba1)

# Displaying the Confusion Matrix, Accuracy, and AUC
print("Confusion Matrix and Accuracy for Split-1 Training:Test(80:20)")
print("Confusion Matrix:")
print(cm_split_1)
print("Accuracy:", accuracy_split_1)
print("AUC:", auc_split_1)
```

```
Confusion Matrix and Accuracy for Split-1 Training:Test(80:20)
Confusion Matrix:
[[80 19]
 [19 36]]
Accuracy: 0.7532467532467533
AUC: 0.8165289256198347
```

[149]:

```
# Performing Logistic Regression & Building Confusion Matrix, Accuracy, and AUC␣
 ↪for Split-2 Training:Test(70:30) Ratio
lr_split_2 = LogisticRegression(solver='lbfgs', random_state=42, max_iter=1000)
lr_split_2.fit(X_train2, y_train2)
y_pred2 = lr_split_2.predict(X_test2)
y_pred_proba2 = lr_split_2.predict_proba(X_test2)[:, 1]
cm_split_2 = confusion_matrix(y_test2, y_pred2)
accuracy_split_2 = accuracy_score(y_test2, y_pred2)
auc_split_2 = roc_auc_score(y_test2, y_pred_proba2)

# Displaying the Confusion Matrix, Accuracy, and AUC
print("Confusion Matrix, Accuracy, and AUC for Split-2 Training:Test(70:30)")
print("Confusion Matrix:")
print(cm_split_2)
print("Accuracy:", accuracy_split_2)
print("AUC:", auc_split_2)
```

```
Confusion Matrix, Accuracy, and AUC for Split-2 Training:Test(70:30)
Confusion Matrix:
[[121  30]
 [ 30  50]]
Accuracy: 0.7402597402597403
AUC: 0.7964403973509934
```

[150]:

```
# Performing Logistic Regression & Building Confusion Matrix, Accuracy, and AUC␣
 ↪for Split-3 Training:Test(60:40) Ratio
lr_split_3 = LogisticRegression(random_state=42, max_iter=1000)
lr_split_3.fit(X_train3, y_train3)
y_pred3 = lr_split_3.predict(X_test3)
y_pred_proba3 = lr_split_3.predict_proba(X_test3)[:, 1]
cm_split_3 = confusion_matrix(y_test3, y_pred3)
```

```
accuracy_split_3 = accuracy_score(y_test3, y_pred3)
auc_split_3 = roc_auc_score(y_test3, y_pred_proba3)

# Displaying the Confusion Matrix, Accuracy, and AUC
print("Confusion Matrix, Accuracy, and AUC for Split-3 Training:Test(60:40)␣
↪Ratio")
print("Confusion Matrix:")
print(cm_split_3)
print("Accuracy:", accuracy_split_3)
print("AUC:", auc_split_3)
```

```
Confusion Matrix, Accuracy, and AUC for Split-3 Training:Test(60:40) Ratio
Confusion Matrix:
[[166  40]
 [ 36  66]]
Accuracy: 0.7532467532467533
AUC: 0.8213401865600609
```

[151]:
```
#Which data split is providing you the best accuracy?

#Ans: As, we observe split_1(80-20), split_3(60-40) ratio are on a tie giving␣
↪the best accuracy of 75% when compared to data split_2(70:30)
# Here, I am selecting split_3 (60:40) for the further analysis, because AUC␣
↪score for Split-3 is higher than i.e 82% than other splits.
```

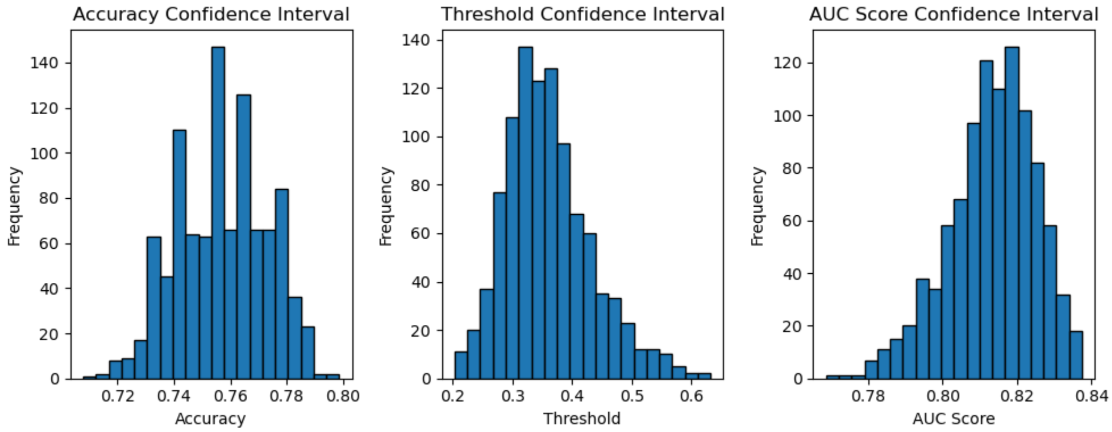
[152]:
```
# Importing all the necessary libraries
from sklearn.utils import resample
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
```

[153]:
```
#Performing bootstrap analysis on the selected data split to calculate␣
↪accuracy, p-value, confidence intervals, and
#ROC threshold, and generate a histogram of the confidence intervals

# No of bootstrap iterations
noofitr = 1000

# Initialized bootstrap results storing arrays
bs_accuracylist = []
bs_thresholdslist = []
bs_auc_scoreslist = []

# Bootstrap iterations are performed below
for _ in range(noofitr):
    # Using the resampled data to train the LR model
```

```python
    X_train_resampled, y_train_resampled = resample(X_train3, y_train3,␣
↪replace=True, n_samples=len(X_train3))

    # Training the LR model on the resampled data
    lr_bootstrap = LogisticRegression(random_state=42, max_iter=1000)
    lr_bootstrap.fit(X_train_resampled, y_train_resampled)

    # Predict using the test set.
    y_pred_bootstrap = lr_bootstrap.predict(X_test3)
    y_pred_proba_bootstrap = lr_bootstrap.predict_proba(X_test3)[:, 1]

    # Determine the current bootstrap iteration's accuracy.
    accuracy_bootstrap = accuracy_score(y_test3, y_pred_bootstrap)
    bs_accuracylist.append(accuracy_bootstrap)

    # Determine the current bootstrap iteration's threshold and ROC curve.
    fpr, tpr, thresholds = roc_curve(y_test3, y_pred_proba_bootstrap)
    optimal_idx = np.argmax(tpr - fpr)
    optimal_threshold = thresholds[optimal_idx]
    bs_thresholdslist.append(optimal_threshold)

    # Determine the current bootstrap iteration's AUC score.
    auc_score = roc_auc_score(y_test3, y_pred_proba_bootstrap)
    bs_auc_scoreslist.append(auc_score)

# Determine the confidence intervals and the p-value.
p_value = np.mean(np.array(bs_accuracylist) > accuracy_split_3)
accuracy_ci = np.percentile(bs_accuracylist, [2.5, 97.5])
threshold_ci = np.percentile(bs_thresholdslist, [2.5, 97.5])
auc_ci = np.percentile(bs_auc_scoreslist, [2.5, 97.5])

# Produce the report.
report = f"""
Logistic Regression Report (Split-3: 60:40 ratio)

Accuracy: {accuracy_split_3:.4f}
P-value: {p_value:.4f}
Accuracy 95% Confidence Interval: [{accuracy_ci[0]:.4f}, {accuracy_ci[1]:.4f}]

Threshold (ROC): {np.mean(bs_thresholdslist):.4f}
Threshold 95% Confidence Interval: [{threshold_ci[0]:.4f}, {threshold_ci[1]:.
↪4f}]

AUC Score: {np.mean(bs_auc_scoreslist):.4f}
AUC Score 95% Confidence Interval: [{auc_ci[0]:.4f}, {auc_ci[1]:.4f}]
"""
```

```python
print(report)

# Plot the confidence interval histogram.
plt.figure(figsize=(10, 4))
plt.subplot(1, 3, 1)
plt.hist(bs_accuracylist, bins=20, edgecolor='black')
plt.xlabel('Accuracy')
plt.ylabel('Frequency')
plt.title('Accuracy Confidence Interval')

plt.subplot(1, 3, 2)
plt.hist(bs_thresholdslist, bins=20, edgecolor='black')
plt.xlabel('Threshold')
plt.ylabel('Frequency')
plt.title('Threshold Confidence Interval')

plt.subplot(1, 3, 3)
plt.hist(bs_auc_scoreslist, bins=20, edgecolor='black')
plt.xlabel('AUC Score')
plt.ylabel('Frequency')
plt.title('AUC Score Confidence Interval')

plt.tight_layout()
plt.show()
```

```
Logistic Regression Report (Split-3: 60:40 ratio)

Accuracy: 0.7532
P-value: 0.5360
Accuracy 95% Confidence Interval: [0.7273, 0.7857]

Threshold (ROC): 0.3601
Threshold 95% Confidence Interval: [0.2415, 0.5325]

AUC Score: 0.8132
AUC Score 95% Confidence Interval: [0.7866, 0.8334]
```


Accuracy Confidence Interval
Frequency
Accuracy
Threshold Confidence Interval
Frequency
Threshold
AUC Score Confidence Interval
Frequency
AUC Score