ML - EXP - 7

Atharva Prashant Pawar (9427) - [Batch - D]

Dataset

↳ *1 cell hidden*

Code

Method - 1 : Bagging ensemble method.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv("heart.csv")
```

```
data.shape
```

```
    (303, 14)
```

```
data.head(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

```
data.tail(5)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	th
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.6
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.9
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.5
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.0
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.0
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
# Split the data into features (X) and target (y)
X = data.drop("target", axis=1)
y = data["target"]
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)
```

```
▾        RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Print the evaluation results
print("Accuracy:", round(accuracy * 100, 2) , "\n\n")
print("Confusion Matrix:\n", conf_matrix, "\n\n")
print("Classification Report:\n", class_report, "\n\n")
```

```
Accuracy: 83.61 


Confusion Matrix:
 [[24  5]
 [ 5 27]] 


Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.83      0.83        29
           1       0.84      0.84      0.84        32

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61
```

Save the Trained Model

```
import joblib

# Save the trained Random Forest classifier model to a file
model_filename = "random_forest_model.joblib"
joblib.dump(rf_classifier, model_filename)
```

```
print(f"Model saved as '{model_filename}'")
```

```
     Model saved as 'random_forest_model.joblib'
```

Load the Save Model

[ ] ↳ *1 cell hidden*

Inference

Class 0:

Usually represents the absence of a heart disease condition. In other words, it indicates that the individual does not have heart disease or is classified as "healthy" or "no heart disease."

Class 1:

Typically represents the presence of a heart disease condition. This indicates that the individual is diagnosed with heart disease or is classified as "having heart disease."

```
import joblib
import pandas as pd

# Load the saved model from the file
model_filename = "random_forest_model.joblib"
loaded_model = joblib.load(model_filename)

# Load new data for inference (replace this with your own data)
new_data = pd.DataFrame({
    'age': [55],
    'sex': [1],
    'cp': [2],
    'trestbps': [130],
    'chol': [250],
    'fbs': [0],
    'restecg': [1],
    'thalach': [155],
    'exang': [0],
    'oldpeak': [1.2],
    'slope': [2],
    'ca': [1],
    'thal': [3]
})

# Use the loaded model to make predictions on new data
new_predictions = loaded_model.predict(new_data)

# Display the predictions
print("Predicted Class for New Data:", new_predictions[0])
```

```
     Predicted Class for New Data: 1
```

```
if new_predictions[0] == 0:
  print("Heart Disease : Absence ")
else:
  print("Heart Disease : Present ")
```

Method - 2 : Boosting ensemble method

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier  # Import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv("heart.csv")
```

```
# Split the data into features (X) and target (y)
X = data.drop("target", axis=1)
y = data["target"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an AdaBoost classifier with a base estimator (e.g., Decision Tree)
base_estimator = DecisionTreeClassifier(max_depth=1)  # You can choose a different base estimator if needed
adaboost_classifier = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)

# Train the AdaBoost classifier on the training data
adaboost_classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_
  warnings.warn(
▸          AdaBoostClassifier
▸ base_estimator: DecisionTreeClassifier
        ▸ DecisionTreeClassifier
```

```
# Make predictions on the test data
y_pred = adaboost_classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Print the evaluation results
print("Accuracy:", round(accuracy * 100, 2), "\n\n")
print("Confusion Matrix:\n", conf_matrix, "\n\n")
print("Classification Report:\n", class_report, "\n\n")
```

```
Accuracy: 80.33


Confusion Matrix:
 [[25  4]
 [ 8 24]]


Classification Report:
               precision    recall  f1-score   support

           0       0.76      0.86      0.81        29
           1       0.86      0.75      0.80        32

    accuracy                           0.80        61
   macro avg       0.81      0.81      0.80        61
weighted avg       0.81      0.80      0.80        61
```

Save the Trained Model

```
import joblib

# Save the trained Random Forest classifier model to a file
model_filename = "adaboost_classifier_model.joblib"
joblib.dump(adaboost_classifier, model_filename)

print(f"Model saved as '{model_filename}'")
```

```
Model saved as 'adaboost_classifier_model.joblib'
```

Inference

Class 0:

Usually represents the absence of a heart disease condition. In other words, it indicates that the individual does not have heart disease or is classified as "healthy" or "no heart disease."

Class 1:

Typically represents the presence of a heart disease condition. This indicates that the individual is diagnosed with heart disease or is classified as "having heart disease."

```
# Load the saved AdaBoost classifier model from a file
model_filename = "adaboost_classifier_model.joblib"
loaded_adaboost_model = joblib.load(model_filename)

# Load new data for inference (replace this with your own data)
new_data = pd.DataFrame({
    'age': [55],
    'sex': [1],
    'cp': [2],
    'trestbps': [130],
    'chol': [250],
    'fbs': [0],
    'restecg': [1],
    'thalach': [155],
    'exang': [0],
    'oldpeak': [1.2],
    'slope': [2],
    'ca': [1],
    'thal': [3]
})

# Use the loaded AdaBoost model to make predictions on new data
new_predictions = loaded_adaboost_model.predict(new_data)

# Display the predictions
print("Predicted Class for New Data:", new_predictions[0])
```

```
     Predicted Class for New Data: 0
```

```
if new_predictions[0] == 0:
  print("Heart Disease : Absence ")
else:
  print("Heart Disease : Present ")
```

```
     Heart Disease : Absence
```