In [5]:
```
import pandas as pd

df = pd.read_csv('tahkeer_data_cleaned.csv')
```

In [6]:
```
columns = df.columns.tolist()
columns.remove("smoking")
features_x = df[columns]
class_y = df["smoking"]
```

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

xtrain, xtest, ytrain, ytest = train_test_split(features_x, class_y, test_size=0.30, shuffle=False, tra
in_size=0.70)
```

In [8]:
```
import numpy as np
from ensemble_methods import BaggingClassifier, RandomForestClassifier, AdaBoost
```

In [30]:
```
bagging_model = BaggingClassifier(n_estimators=100, max_depth=60)
bagging_model.fit(xtrain, ytrain)

bagging_predictions = bagging_model.predict(xtest)
bagging_accuracy = accuracy_score(ytest, bagging_predictions)
print(f"Bagging Accuracy: {bagging_accuracy}")
```

```
Bagging Accuracy: 0.7316913595880292
```

In [31]:
```
adaboost = AdaBoost(n_estimators=300)
adaboost.fit(xtrain, ytrain)

predictions = adaboost.predict(xtest)

accuracy = accuracy_score(ytest, predictions)
print(f'Boosting Accuracy: {accuracy}')
```

```
Boosting Accuracy: 0.7187117470769491
```

In [32]:
```
random_forest = RandomForestClassifier(n_estimators=150, max_depth=3, min_samples_split=2, min_samples_
leaf=1)
random_forest.fit(xtrain, ytrain)

predictions = random_forest.predict(xtest)
accuracy = accuracy_score(ytest, predictions)
print(f'Random Forest Accuracy: {accuracy}')
```

```
Random Forest Accuracy: 0.7062597610907095
```

Hyperparameter Tuning

We will use grid search and randomized search methods to choose better hyperparameters

In [22]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from skopt import BayesSearchCV
import multiprocessing

n_jobs = multiprocessing.cpu_count()-1

np.int = int

def print_results(search_results):
    best_params = search_results.best_params_
    best_score = search_results.best_score_

    print("Best Parameters:", best_params)
    print("Best Score:", best_score)
```

Tuning Bagging model

In [23]:
```
params = {
    'n_estimators': [100, 200, 250, 300],
    'max_depth': [50, 60, 70, 90],
    'threshold': [0.3, 0.5, 0.7],
}

bagging = BaggingClassifier()
grid_search = GridSearchCV(estimator=bagging, param_grid=params, scoring='accuracy', n_jobs=n_jobs)
grid_search.fit(xtrain, ytrain)
print("Grid search - Bagging: ")
print_results(grid_search)
```

```
Grid search - Bagging:
Best Parameters: {'max_depth': 50, 'n_estimators': 100, 'threshold': 0.5}
Best Score: 0.7288794679763679
```

In [24]:
```
random_search = RandomizedSearchCV(estimator=bagging, param_distributions=params, n_iter=6, scoring='ac
curacy', random_state=42, n_jobs=n_jobs)

random_search.fit(xtrain, ytrain)
print("Random search - Bagging: ")
print_results(random_search)
```

```
Random search - Bagging:
Best Parameters: {'threshold': 0.5, 'n_estimators': 100, 'max_depth': 90}
Best Score: 0.7281558532037508
```

In [ ]:
```
bayes_optimization = BayesSearchCV(bagging, params, n_iter=6, scoring='accuracy', random_state=42, n_jo
bs=n_jobs)

bayes_optimization.fit(xtrain, ytrain)
print("Bayesian Optimization - Bagging: ")
print_results(bayes_optimization)
```

Tuning AdaBoost model

In [15]:
```
params = {
    'n_estimators': [100, 200, 250, 300, 500, 1000],
}

adaboost = AdaBoost()
grid_search = GridSearchCV(estimator=adaboost, param_grid=params, scoring='accuracy', n_jobs=n_jobs)
grid_search.fit(xtrain, ytrain)
print("Grid search - AdaBoost: ")
print_results(grid_search)
```

```
Grid search - AdaBoost:
Best Parameters: {'n_estimators': 100}
Best Score: 0.714262437542956
```

In [37]:
```
random_search = RandomizedSearchCV(estimator=adaboost, param_distributions=params, n_iter=6, scoring='a
ccuracy', random_state=42, n_jobs=n_jobs)

random_search.fit(xtrain, ytrain)
print("Random search - AdaBoost: ")
print_results(random_search)
```

```
Random search - AdaBoost:
Best Parameters: {'n_estimators': 100}
Best Score: 0.714262437542956
```

In [18]:
```
bayes_optimization = BayesSearchCV(adaboost, params, n_iter=6, scoring='accuracy', random_state=42, n_j
obs=n_jobs)

bayes_optimization.fit(xtrain, ytrain)
print("Bayesian Optimization - AdaBoost: ")
print_results(bayes_optimization)
```

```
Bayesian Optimization - AdaBoost:
Best Parameters: OrderedDict([('n_estimators', 250)])
Best Score: 0.714262437542956
```

Tuning Random Forest model

In [38]:
```
params = {
    'n_estimators': [100, 200, 250],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 3],
    'min_samples_leaf': [1, 2, 4],
}

rf_model = RandomForestClassifier()
grid_search = GridSearchCV(estimator=rf_model, param_grid=params, scoring='accuracy', n_jobs=n_jobs)
grid_search.fit(xtrain, ytrain)

print("Grid search - Random Forest: ")
print_results(grid_search)
```

```
Grid search - Random Forest:
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 25
0}
Best Score: 0.7214714514997054
```

In [40]:
```
random_search = RandomizedSearchCV(estimator=rf_model, param_distributions=params, n_iter=6, scoring='a
ccuracy', random_state=42, n_jobs=n_jobs)
random_search.fit(xtrain, ytrain)

print("Random Search - Random Forest: ")
print_results(random_search)
```

```
Random Search - Random Forest:
Best Parameters: {'n_estimators': 250, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_depth': 1
0}
Best Score: 0.7211367772692194
```

In [20]:
```
bayes_optimization = BayesSearchCV(rf_model, params, n_iter=6, scoring='accuracy', random_state=42, n_j
obs=n_jobs)

bayes_optimization.fit(xtrain, ytrain)
print("Bayesian Optimization - Random Forest: ")
print_results(bayes_optimization)
```

```
Bayesian Optimization - Random Forest:
Best Parameters: OrderedDict([('max_depth', 10), ('min_samples_leaf', 2), ('min_samples_split', 3),
('n_estimators', 200)])
Best Score: 0.7210644310501265
```