IMPORT NEEDED LIBRARIES

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import precision_recall_fscore_support
import time
```

BINARY DATASET

IMPORT AND READ BINARY DATASET

```
names =
['pH','TDS','Turbidity','Phospate','Nitrate','Iron','COD(mg/L)','Chlor
ine','Sodium','Class']
bdataframe = pandas.read_csv("binary.csv", names=names)
array = bdataframe.values
Xb = array[:,0:9]
Yb = array[:,9]
```

SPLITTING TRAIN,TEST AND VALIDATION DATA

```
Xb_train, Xb_test, Yb_train, Yb_test = train_test_split(Xb, Yb,
test_size=0.3)
Xb_test, Xb_val, Yb_test, Yb_val = train_test_split(Xb_test, Yb_test,
test_size=0.4)
```

SPECIFYING BASE CLASSIFIER

```
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed,
shuffle=True)
cart = DecisionTreeClassifier()
num_trees = 100
```

CREATING AND FITTING BAGGING CLASSIFIER TO MODEL

```
bstart = time.time()
bagging =
BaggingClassifier(base_estimator=cart,n_estimators=num_trees,
random_state=seed)
b_bagging_model = bagging.fit(Xb_train,Yb_train)
bend = time.time()
# total time taken
print(f"Runtime of the Bagging is {bend - bstart}")
```

```
Runtime of the Bagging is 0.9419000148773193
```

ACCURACY

```
Yb_pred = b_bagging_model.predict(Xb_test)
print("Accuracy on test data:", metrics.accuracy_score(Yb_test,
Yb_pred))
Yb_val_pred = b_bagging_model.predict(Xb_val)
print("Accuracy on validation data:", metrics.accuracy_score(Yb_val,
Yb_val_pred))
```

```
Accuracy on test data: 0.9988888888888889
Accuracy on validation data: 0.9983333333333333
```

PRECISION-RECALL-F1 SCORE

```
print("Precision-Recall-Fscore[binary] on test data:",
precision_recall_fscore_support(Yb_test, Yb_pred, average='binary'))
print("Precision-Recall-Fscore[binary] on validation data:",
precision_recall_fscore_support(Yb_val, Yb_val_pred,
average='binary'))
```

```
Precision-Recall-Fscore[binary] on test data: (1.0,
0.9915966386554622, 0.9957805907172996, None)
Precision-Recall-Fscore[binary] on validation data: (1.0,
0.9859154929577465, 0.9929078014184397, None)
```

CROSS VALIDATION SCORE

```
results = model_selection.cross_val_score(bagging, Xb, Yb, cv=kfold)
print(results.mean())
```

```
0.9996
0.9996
```

MULTICLASS DATASET

IMPORT AND READ MULTICLASS DATASET

```
dataframe = pandas.read_csv("multi.csv", names=names)
array = dataframe.values
Xm = array[:,0:9]
Ym = array[:,9]
```

SPLITTING TRAIN, TEST AND VALIDATION DATA

```
Xm_train, Xm_test, Ym_train, Ym_test = train_test_split(Xm, Ym,
test_size=0.3)
Xm_test, Xm_val, Ym_test, Ym_val = train_test_split(Xm_test, Ym_test,
test_size=0.4)
```

CREATING AND FITTING BAGGING CLASSIFIER TO MODEL

```
start = time.time()
m_bagging_model = bagging.fit(Xm_train,Ym_train)
end = time.time()
```

```
# total time taken
print(f"Runtime of the Bagging is {end - start}")
```

```
Runtime of the Bagging is 0.7528047561645508
```

ACCURACY

```
Ym_pred = m_bagging_model.predict(Xm_test)
print("Accuracy on test data:", metrics.accuracy_score(Ym_test,
Ym_pred))
Ym_val_pred = m_bagging_model.predict(Xm_val)
print("Accuracy on validation data:", metrics.accuracy_score(Ym_val,
Ym_val_pred))
```

```
Accuracy on test data: 1.0
Accuracy on validation data: 0.9991783073130649
```

PRECISION-RECALL-F1 SCORE

```
print("Precision-Recall-Fscore[weighted] on test data:",
precision_recall_fscore_support(Ym_test, Ym_pred, average='weighted'))
print("Precision-Recall-Fscore[weighted] on validation data:",
precision_recall_fscore_support(Ym_val, Ym_val_pred,
average='weighted'))
```

```
Precision-Recall-Fscore[weighted] on test data: (1.0, 1.0, 1.0, None)
Precision-Recall-Fscore[weighted] on validation data:
(0.9991848808545605, 0.9991783073130649, 0.9991787934277186, None)
```

CROSS VALIDATION SCORE

```
results = model_selection.cross_val_score(bagging, Xm, Ym, cv=kfold)
print(results.mean())
```

```
1.0
```