Practical 9 Implementation of Bagging Algorithm:

a.Random Forest

```
import numpy as np
import pandas as pd
```

Importing Dataset

```
dataset = pd.read_csv('weatherAUS.csv')
X = dataset.iloc[:,[1,2,3,4,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]].values
Y = dataset.iloc[:,-1].values
```

```
print(X)
```

```
[['Albury' 13.4 22.9 ... 16.9 21.8 'No']
 ['Albury' 7.4 25.1 ... 17.2 24.3 'No']
 ['Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 ['Sydney' 21.3 36.5 ... 29.1 32.4 nan]
 ['Sydney' 13.7 23.5 ... 16.8 21.8 'Yes']
 ['Sydney' 13.8 21.5 ... nan nan nan]]
```

```
print(Y)
```

```
['No' 'No' 'No' ... 'Yes' 'No' nan]
```

```
Y = Y.reshape(-1,1)
```

Dealing with invalid Data

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan,strategy='most_frequent')
X = imputer.fit_transform(X)
Y = imputer.fit_transform(Y)
```

Encoding Dataset

```
from sklearn.preprocessing import LabelEncoder
le1 = LabelEncoder()
X[:,0] = le1.fit_transform(X[:,0])
le2 = LabelEncoder()
X[:,4] = le2.fit_transform(X[:,4])
le3 = LabelEncoder()
X[:,6] = le3.fit_transform(X[:,6])
le4 = LabelEncoder()
X[:,7] = le4.fit_transform(X[:,7])
le5 = LabelEncoder()
X[:,-1] = le5.fit_transform(X[:,-1])
le6 = LabelEncoder()
Y[:,-1] = le6.fit_transform(Y[:,-1])
```

```
print(X)
```

```
[[0 13.4 22.9 ... 16.9 21.8 0]
 [0 7.4 25.1 ... 17.2 24.3 0]
 [0 12.9 25.7 ... 21.0 23.2 0]
 ...
 [10 21.3 36.5 ... 29.1 32.4 0]
 [10 13.7 23.5 ... 16.8 21.8 1]
 [10 13.8 21.5 ... 19.0 18.4 0]]
```

```
print(Y)
```

```
[[0]
 [0]
 [0]
 ...
 [1]
 [0]
 [0]]
```

```
Y = np.array(Y,dtype=float)
print(Y)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [1.]
 [0.]
 [0.]]
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

```
print(X)
```

```
[[-1.57320195  0.02304777 -0.18794702 ... -0.1670575  -0.10503399
  -0.5268001 ]
 [-1.57320195 -1.01411762  0.17979137 ... -0.11067369  0.32519811
  -0.5268001 ]
 [-1.57320195 -0.06338268  0.28008366 ...  0.60352124  0.13589599
  -0.5268001 ]
 ...
 [ 1.63669994  1.38864885  2.08534483 ...  2.12588412  1.71915013
  -0.5268001 ]
 [ 1.63669994  0.07490603 -0.08765473 ... -0.1858521  -0.10503399
   1.89825325]
 [ 1.63669994  0.09219212 -0.42196235 ...  0.22762918 -0.69014965
  -0.5268001 ]]
```

Splitting Dataset into Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100,random_state=0)
classifier.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:1389: DataConversionWarning: A column-vector y was passed when a 1d array w
  return fit_method(estimator, *args, **kwargs)
▾    RandomForestClassifier
RandomForestClassifier(random_state=0)
```

```
classifier.score(X_train,Y_train)
```

```
0.9996544309629857
```

```
y_pred = le6.inverse_transform(np.array(classifier.predict(X_test),dtype=int))
Y_test = le6.inverse_transform(np.array(Y_test,dtype=int))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_label.py:151: DataConversionWarning: A column-vector y was passed wh
  y = column_or_1d(y, warn=True)
```

```
y_pred = y_pred.reshape(-1,1)
Y_test = Y_test.reshape(-1,1)
```

```
df = np.concatenate((Y_test,y_pred),axis=1)
dataframe = pd.DataFrame(df,columns=['Rain on Tommorrow Actual Data','Prediction of Rain Predicted from Model '])
```

```
print(dataframe)
```

```
     Rain on Tommorrow Actual Data Prediction of Rain Predicted from Model
0                              No                                       No
1                              No                                       No
2                              No                                       No
3                              No                                       No
4                             Yes                                       No
...                           ...                                      ...
6506                           No                                       No
6507                           No                                       No
6508                          Yes                                       No
6509                          Yes                                      Yes
6510                          Yes                                      Yes

[6511 rows x 2 columns]
```

Calculating Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,y_pred)
```

```
0.8504070035324834
```