PRACTICAL 10

AIM: Implementation of various boosting algorithms and ensemble learning techniques using the Letter Recognition dataset from the UCI Repository. The dataset, available at https://archive.ics.uci.edu/ml/machinelearningdatabases/letterrecognition/letterrecognition.data, will be used to train and evaluate the following models:
Boosting algorithms: a) AdaBoost; b) Gradient Boosting; c) XGBoost Ensemble learning variants: a) Stacking; b) Voting Classifier
Compare the performance of these algorithms in recognizing uppercase letters based on various statistical features. Analyze the effectiveness of each method in improving classification accuracy and generalization.

Code:

Import Libraries:

```
import pandas as pd from sklearn.ensemble import
AdaBoostClassifier from sklearn.tree import DecisionTreeClassifier from
sklearn.model_selection import train_test_split from sklearn.metrics
import accuracy_score
from sklearn.ensemble import GradientBoostingClassifier  from
sklearn.preprocessing import LabelEncoder from
xgboost import XGBClassifier from sklearn.ensemble import
StackingClassifier from sklearn.linear_model import
LogisticRegression from sklearn.ensemble import
VotingClassifier from sklearn.metrics import accuracy_score
```

Load the dataset:

```
column_names = ['Letter', 'x-box', 'y-box', 'width', 'height', 'onpix', 'x-bar', 'y-bar', 'x2bar', 'y2bar',
 'xybar', 'x2ybr', 'xy2br', 'x-edge', 'xegvy', 'y-edge', 'yegvx']
data = pd.read_csv('/content/drive/MyDrive/GUNI CLASS MATERIAL/SEM 5/MACHINE
LEARNING (ML)/Dataset/letter-recognition.data', names=column_names)
data.sample(5)
```

	Letter	x-box	y-box	width	height	onpix	x-bar	y-bar	x2bar	y2bar	xybar	x2ybr	xy2br	x-edge	xegvy	y-edge	yegvx
6009	P	5	11	6	8	3	3	14	8	1	11	7	3	1	10	4	8
3934	I	5	10	4	6	2	9	7	3	6	13	4	5	2	9	4	10
9906	M	4	8	6	6	7	7	6	5	5	7	7	10	10	5	2	8
6186	Y	9	15	8	8	5	8	6	4	5	9	8	5	4	10	4	4
13647	X	4	8	6	6	3	7	8	0	7	9	7	8	2	8	3	7

Name: DHRUV SHERE
Enrollment No:23012022021
Batch: 5IT-B-2

#Features and target
X = data.iloc[:, 1:].values
y = data.iloc[:, 0].values

a) AdaBoost:

 # Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # AdaBoost model ada = AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1), n_estimators=50, random_state=42) ada.fit(X_train, y_train) y_pred = ada.predict(X_test)

 # Accuracy print("AdaBoost Accuracy:", accuracy_score(y_test, y_pred))

```
AdaBoost Accuracy: 0.2485
```

b) Gradient Boosting:

 # Gradient Boosting model gbc = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42) gbc.fit(X_train, y_train) y_pred = gbc.predict(X_test)

 # Accuracy print("Gradient Boosting Accuracy:", accuracy_score(y_test, y_pred))

```
Gradient Boosting Accuracy: 0.9205
```

c) XGBoost:

 # Encode the target labels label_encoder = LabelEncoder() y_train_encoded = label_encoder.fit_transform(y_train) y_test_encoded = label_encoder.transform(y_test)

 # XGBoost model xgb = XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42) xgb.fit(X_train, y_train_encoded) y_pred_encoded = xgb.predict(X_test)

```
# Decode the predicted labels back to original y_pred = label_encoder.inverse_transform(y_pred_encoded) # Accuracy print("XGBoost Accuracy:", accuracy_score(y_test, y_pred))
```

```
XGBoost Accuracy: 0.8855
```

d) Stacking:

```
# Base learners estimators = [
    ('AdaBoost', AdaBoostClassifier(n_estimators=50, random_state=42)),
    ('GradientBoosting', GradientBoostingClassifier(n_estimators=100, random_state=42)),
('XGBoost', XGBClassifier(n_estimators=100, random_state=42))
]
# Stacking model stack = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression()) stack.fit(X_train, y_train)
y_pred = stack.predict(X_test)

# Accuracy print("Stacking Accuracy:", accuracy_score(y_test, y_pred))
```

```
Stacking Accuracy: 0.96125
```

e) Voting Classifier:

```
# Define models for voting voting_clf = VotingClassifier(estimators=estimators, voting='soft') voting_clf.fit(X_train, y_train)
y_pred = voting_clf.predict(X_test)

# Accuracy print("Voting Classifier Accuracy:", accuracy_score(y_test, y_pred))
```

```
Voting Classifier Accuracy: 0.95525
```