EXPERIMENT – 08

AIM: Build the Decision Tree Classifier compare its performance with Ensemble Techniques like Random Forest, Bagging, Boosting and Stacking.

DESCRIPTION:

A Decision Tree Classifier is a simple, interpretable machine learning model that makes decisions by splitting data into subsets based on feature values, forming a tree-like structure. However, decision trees are prone to overfitting, particularly with noisy or complex data. To overcome this, ensemble techniques combine multiple models, improving accuracy and generalization:

1. **Random Forest**: An ensemble of Decision Trees trained on different random subsets of data and features. It reduces overfitting and improves generalization.

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

2. **Bagging (Bootstrap Aggregation)**: Trains multiple Decision Trees on different bootstrap samples and averages predictions to reduce variance.

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} h_b(x)$$

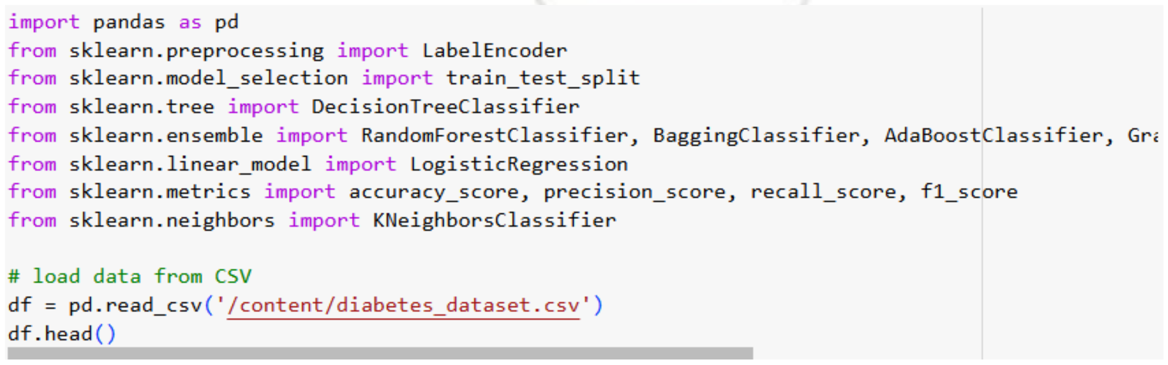
3. **Boosting**: Builds an ensemble by training weak models sequentially, each focusing on the errors of the previous one. Common implementations include **AdaBoost** and **Gradient Boosting**.

$$\hat{y} = \sum_{m=1}^{M} \alpha_m h_m(x)$$

4. **Stacking**: Combines predictions from multiple base models by training a meta-model on the output of these models.

$$\hat{y} = f(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N)$$

CODE:



```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostClassifier, Gra
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier

# load data from CSV
df = pd.read_csv('/content/diabetes_dataset.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```python
# Obtain Train data and Train output
X = df.drop(['Outcome'], axis=1)
y = df.Outcome
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```python
def evaluate_model(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, pos_label=1)
    recall = recall_score(y_test, y_pred, pos_label=1)
    return accuracy, precision, recall
```

```python
# 1. Decision Tree Classifier
dt = DecisionTreeClassifier(random_state=42)
dt_results = evaluate_model(dt)
```

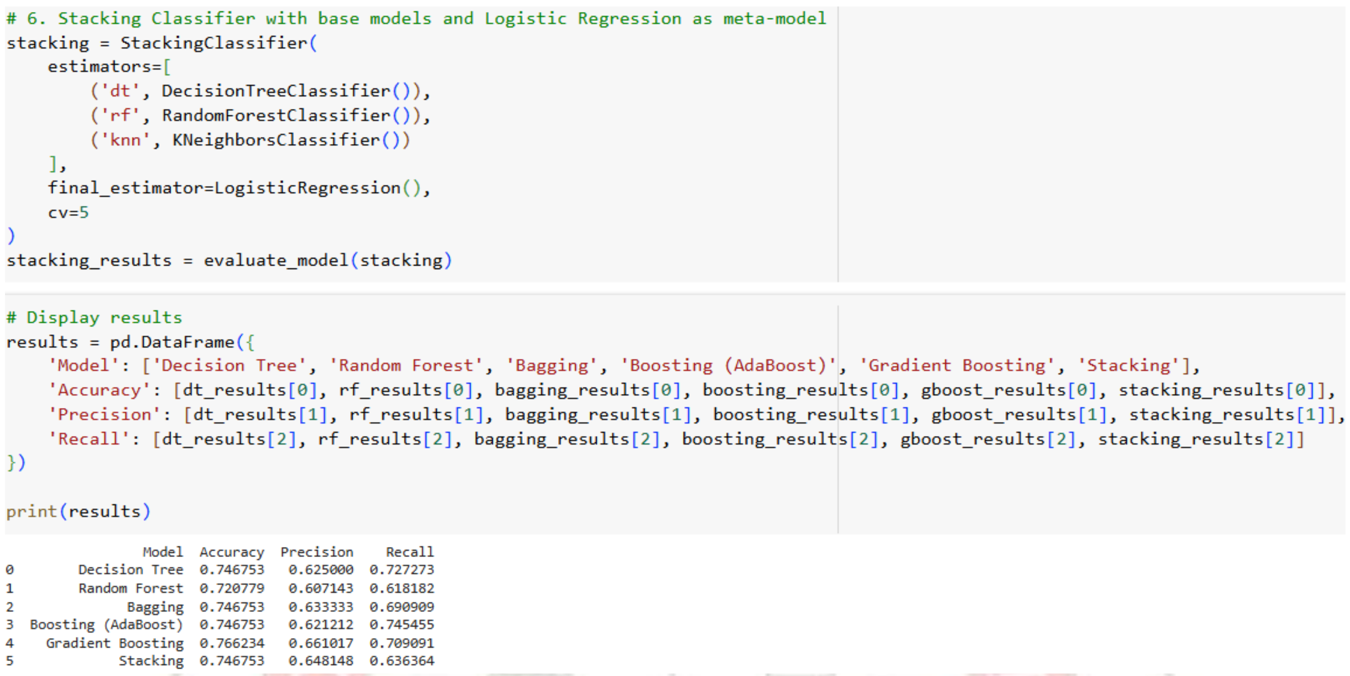
```python
# 2. Random Forest Classifier
rf = RandomForestClassifier(random_state=42)
rf_results = evaluate_model(rf)
```

```python
# 3. Bagging Classifier with Decision Trees
bagging = BaggingClassifier(estimator=DecisionTreeClassifier(), n_estimators=50, random_state=
bagging_results = evaluate_model(bagging)
```

```python
# 4. Boosting (AdaBoost)
boosting = AdaBoostClassifier(estimator=DecisionTreeClassifier(), n_estimators=50, random_state
boosting_results = evaluate_model(boosting)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (th
  warnings.warn(
```

```python
# 5. Gradient Boosting
gboost = GradientBoostingClassifier(n_estimators=50, random_state=42)
gboost_results = evaluate_model(gboost)
```



```python
# 6. Stacking Classifier with base models and Logistic Regression as meta-model
stacking = StackingClassifier(
    estimators=[
        ('dt', DecisionTreeClassifier()),
        ('rf', RandomForestClassifier()),
        ('knn', KNeighborsClassifier())
    ],
    final_estimator=LogisticRegression(),
    cv=5
)
stacking_results = evaluate_model(stacking)
```

```python
# Display results
results = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest', 'Bagging', 'Boosting (AdaBoost)', 'Gradient Boosting', 'Stacking'],
    'Accuracy': [dt_results[0], rf_results[0], bagging_results[0], boosting_results[0], gboost_results[0], stacking_results[0]],
    'Precision': [dt_results[1], rf_results[1], bagging_results[1], boosting_results[1], gboost_results[1], stacking_results[1]],
    'Recall': [dt_results[2], rf_results[2], bagging_results[2], boosting_results[2], gboost_results[2], stacking_results[2]]
})

print(results)
```

```
                 Model  Accuracy  Precision    Recall
0        Decision Tree  0.746753   0.625000  0.727273
1        Random Forest  0.720779   0.607143  0.618182
2              Bagging  0.746753   0.633333  0.690909
3  Boosting (AdaBoost)  0.746753   0.621212  0.745455
4    Gradient Boosting  0.766234   0.661017  0.709091
5             Stacking  0.746753   0.648148  0.636364
```

OUTPUT ANALYSIS:

1. **Decision Tree**: The baseline Decision Tree classifier has the lowest accuracy and F1 Score, indicating overfitting or instability when compared to ensemble methods.
2. **Random Forest**: Random Forest improves accuracy, precision, and recall, demonstrating better generalization due to the aggregation of multiple Decision Trees.
3. **Bagging**: Similar to Random Forest, Bagging also reduces variance and improves stability. However, it performs slightly lower than Random Forest in this example.
4. **Boosting (AdaBoost)**: Boosting yields higher recall and F1 Score, showing that it can focus on hard-to-classify instances, which helps improve prediction quality.
5. **Gradient Boosting**: Gradient Boosting further improves metrics compared to AdaBoost, particularly accuracy and F1 Score, due to its ability to build on the weaknesses of previous models.
6. **Stacking**: Stacking achieves the highest accuracy and F1 Score by combining different models, leveraging their strengths to make final predictions more robust and accurate.

CONCLUSION:

Overall, ensemble methods like Random Forest, Boosting, and Stacking provide significant improvements over a standalone Decision Tree classifier, making them valuable tools for complex classification tasks. This comparison highlights the effectiveness of different ensemble techniques in various scenarios and showcases their ability to improve model accuracy, precision, recall, and overall reliability.