Ensemble Methods in Python

This assignment focuses on implementing and understanding ensemble methods in Python. You will work with **Bagging**, **Boosting**, and **Stacking** to solve a classification problem using the synthetic dataset provided in the instructions.

Task 1: Setup the Environment

In [ ]:
```
# pip install xgboost
```

In [23]:
```
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

In [25]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

Task 2: Create a Synthetic Dataset

In [27]:
```
# Generate synthetic dataset
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15,
                           n_redundant=5, random_state=42)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Task 3: Implement Bagging (Random Forest)

In [29]:
```
# Train Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Predictions and evaluation
y_pred_rf = rf.predict(X_test)
print("Bagging (Random Forest) Accuracy:", round(accuracy_score(y_test, y_pred_rf), 3))
```

```
Bagging (Random Forest) Accuracy: 0.887
```

Task 4: Implement Boosting (XGBoost)

In [31]:
```
# Train XGBoost Classifier
xgb = XGBClassifier(n_estimators=100, learning_rate=0.1, random_state=42)
xgb.fit(X_train, y_train)

# Predictions and evaluation
y_pred_xgb = xgb.predict(X_test)
print("Boosting (XGBoost) Accuracy:", round(accuracy_score(y_test, y_pred_xgb), 3))
```

```
Boosting (XGBoost) Accuracy: 0.907
```

Task 5: Implement Stacking

In [33]:
```
# Define base models
base_models = [
    ('svm', SVC(probability=True)),
    ('tree', DecisionTreeClassifier())
]

# Define the meta-model
meta_model = LogisticRegression()

# Create Stacking Classifier
stack = StackingClassifier(estimators=base_models, final_estimator=meta_model, cv=5)
stack.fit(X_train, y_train)
```

```python
# Predictions and evaluation
y_pred_stack = stack.predict(X_test)
print("Stacking Accuracy:", round(accuracy_score(y_test, y_pred_stack), 3))
```

```
Stacking Accuracy: 0.937
```