Experiment No. 6
Apply XGBOOST for credit fraud detection
Date of Performance:
Date of Submission:

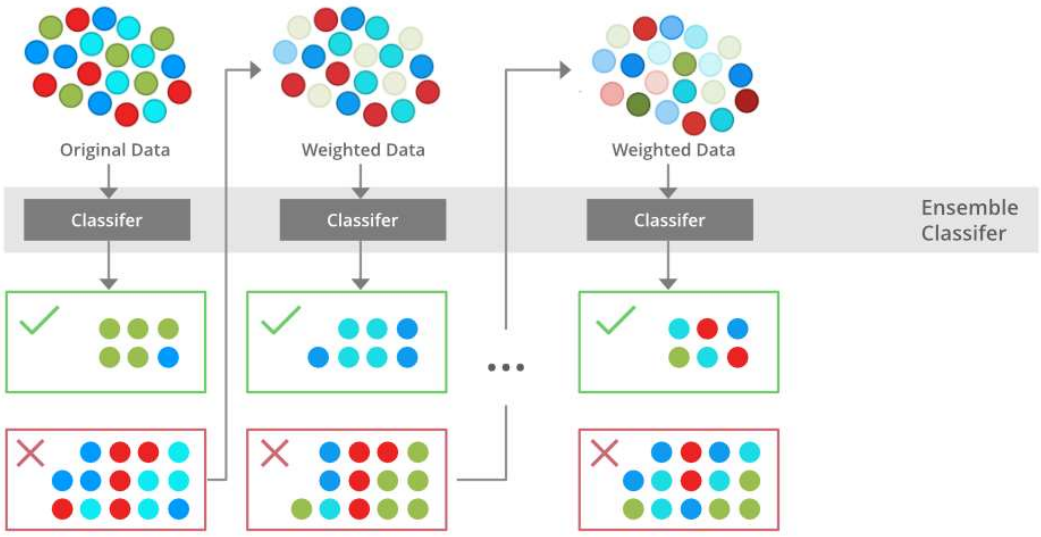
Aim: Apply XGBOOST for credit fraud detection.

Objective: Ability to implement ensemble learning algorithms.

Theory:

Boosting:

Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



XGBoost:

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for "Extreme Gradient Boosting" and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

Code:

```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Create synthetic dataset
X, y = make_classification(n_samples=10000, n_features=20, n_classes=2, weights=[0.99, 0.01], random_state=42)

# Convert to DataFrame
dataset = pd.DataFrame(X, columns=[f'feature_{i}' for i in range(X.shape[1])])
dataset['Class'] = y

# Split the dataset into features and labels
X = dataset.drop('Class', axis=1)  # Features
y = dataset['Class']  # Labels

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the XGBoost model
model = XGBClassifier()

# Train the model
model.fit(X_train, y_train)

# Predictions on the test set
y_pred = model.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(conf_matrix)
```

Output:

```
Accuracy: 0.9885
Precision: 0.6666666666666666
Recall: 0.23076923076923078
F1 Score: 0.3428571428571429
Confusion Matrix:
[[1971    3]
 [  20    6]]
```

Conclusion:

XGBoost is known for its high performance in classification tasks and is often used in fraud detection due to its ability to handle imbalanced datasets and capture complex patterns. Results may vary depending on the quality of the dataset, feature engineering, hyperparameter tuning, and the specific characteristics of the fraud patterns. It's important to continuously monitor and update the model to adapt to new fraud patterns and ensure its effectiveness over time.