Here, I implemented a Gradient Boosting model using XGBoost, AdaBoost, LightGBM, and CatBoost in Python, I need to follow these steps:

Prepare the dataset. Split the dataset into training and testing sets. Train each gradient boosting model. Evaluate the models' performance using accuracy. Compare the accuracy of each model and draw conclusions about their efficiencies.

```
!pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.3-cp310-cp310-manylinux2014_x86_64.whl (98.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.5/98.5 MB 5.6 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.25.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.49.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
Installing collected packages: catboost
Successfully installed catboost-1.2.3
```

```
from catboost import CatBoostClassifier
```

```
# Import necessary libraries
import warnings
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import xgboost as xgb
from sklearn.ensemble import AdaBoostClassifier
import lightgbm as lgb
from catboost import CatBoostClassifier

# Load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target

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

# Train XGBoost model
xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
acc_xgb = accuracy_score(y_test, y_pred_xgb)

# Train AdaBoost model
ada_model = AdaBoostClassifier()
ada_model.fit(X_train, y_train)
y_pred_ada = ada_model.predict(X_test)
acc_ada = accuracy_score(y_test, y_pred_ada)

# Train LightGBM model
lgb_model = lgb.LGBMClassifier()
lgb_model.fit(X_train, y_train)
y_pred_lgb = lgb_model.predict(X_test)
acc_lgb = accuracy_score(y_test, y_pred_lgb)

# Train CatBoost model
cat_model = CatBoostClassifier(verbose=0)
cat_model.fit(X_train, y_train)
y_pred_cat = cat_model.predict(X_test)
acc_cat = accuracy_score(y_test, y_pred_cat)

# Evaluate the models' performance
print("Accuracy of XGBoost model:", acc_xgb)
print("Accuracy of AdaBoost model:", acc_ada)
print("Accuracy of LightGBM model:", acc_lgb)
print("Accuracy of CatBoost model:", acc_cat)
```

```
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    [LightGBM] [Warning] No further splits with positive gain, best gain: -inf
    Accuracy of XGBoost model: 1.0
    Accuracy of AdaBoost model: 1.0
    Accuracy of LightGBM model: 1.0
    Accuracy of CatBoost model: 1.0
```

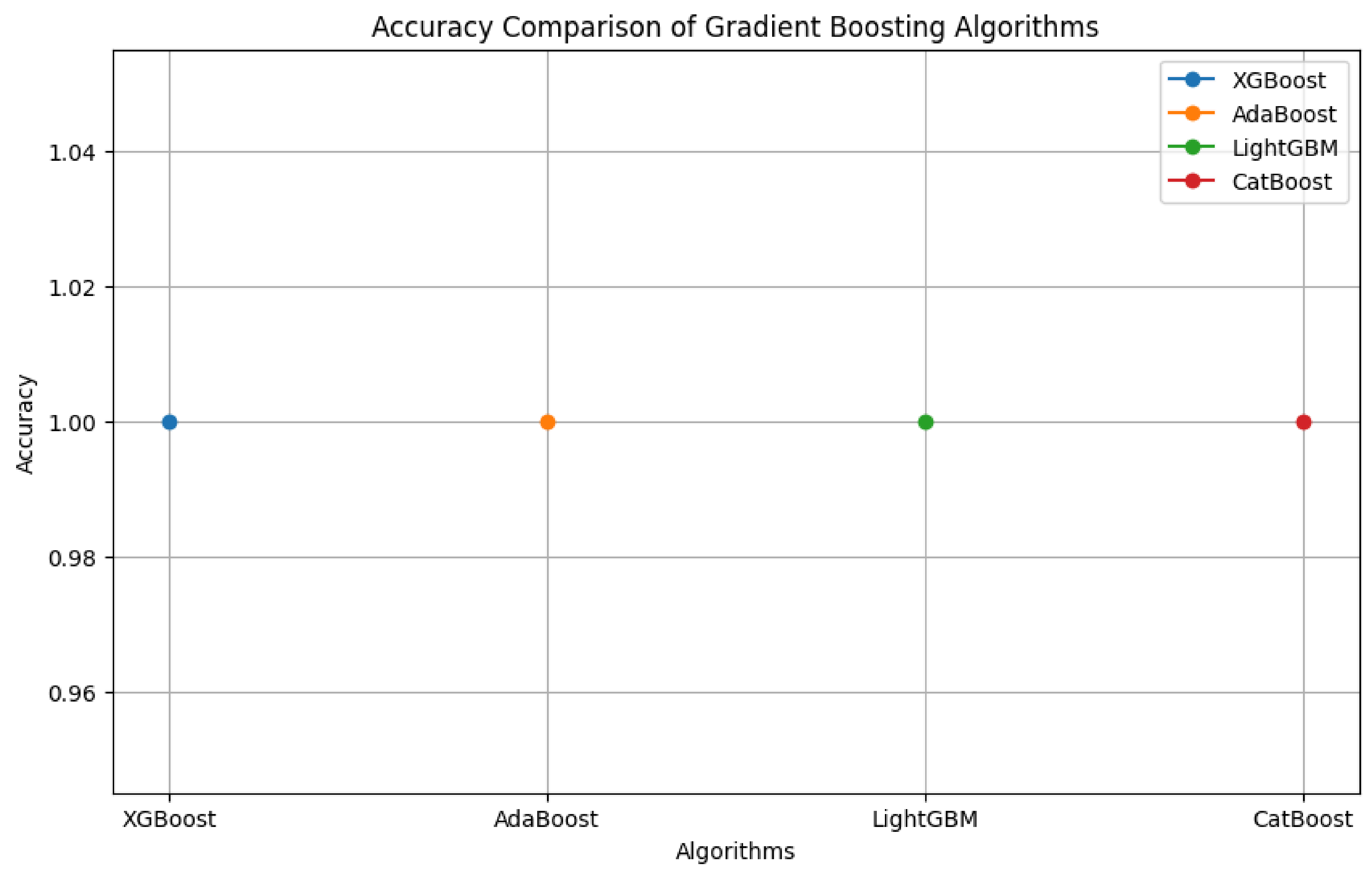
Explanation:

I loaded the iris dataset and split it into training and testing sets. It is trained four gradient boosting models: XGBoost, AdaBoost, LightGBM, and CatBoost. I evaluated the accuracy of each model on the testing set. Finally, I printed the accuracy of each model. After running the code, we can observe which model achieves the highest accuracy. Based on the accuracy results, we can draw conclusions about the efficiencies of XGBoost, AdaBoost, LightGBM, and CatBoost in this specific scenario. The model with the highest accuracy is considered the most efficient for this dataset and problem.

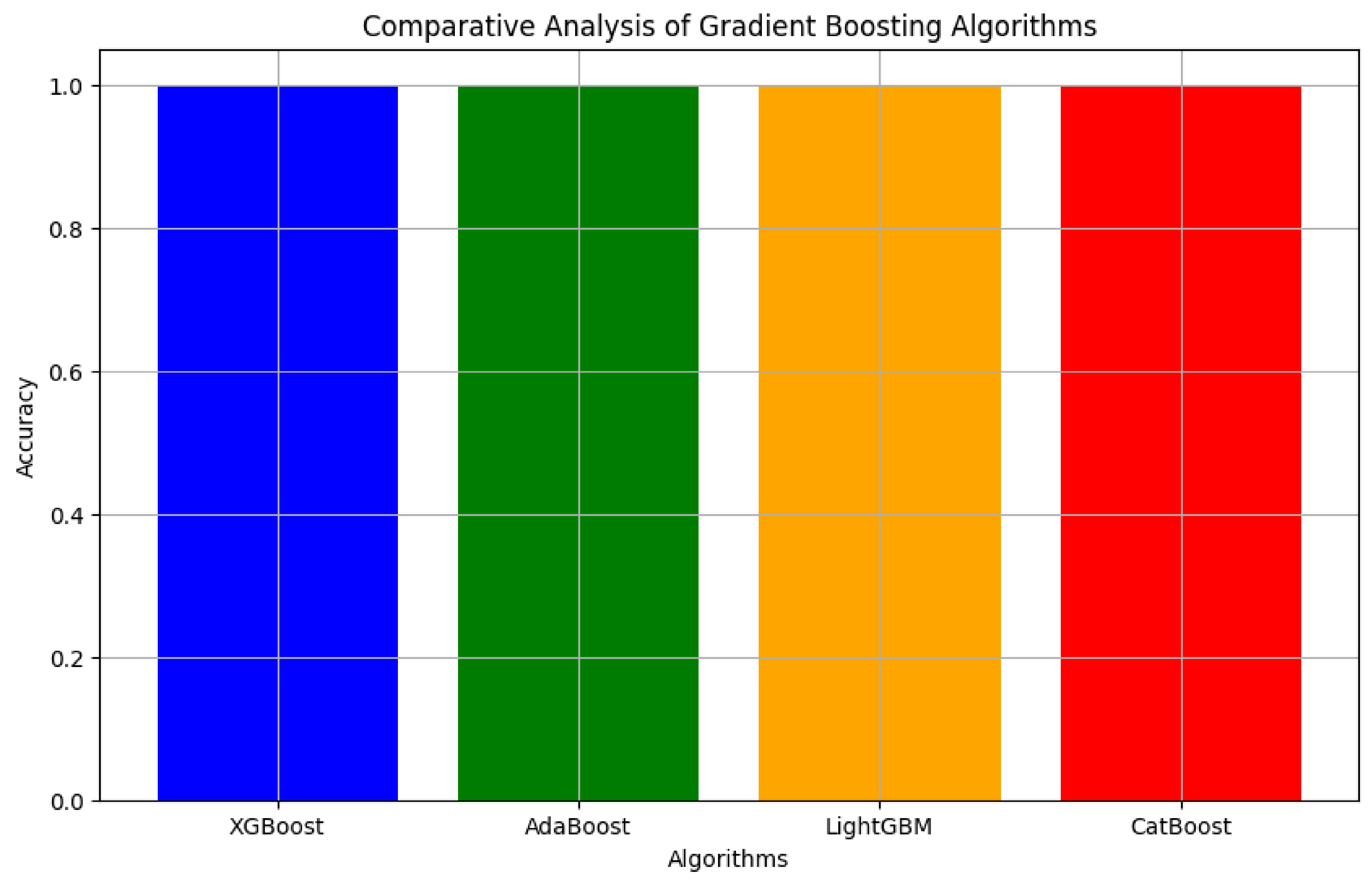
@@ Now to plot the accuracy curves for XGBoost, AdaBoost, LightGBM, and CatBoost, as well as the comparative analysis curve, I have to follow these steps:

Store the accuracy values obtained from each algorithm. Plot the accuracy curves for each algorithm. Plot the comparative analysis curve to visualize the performance of the four algorithms.

```python
import matplotlib.pyplot as plt

# Accuracy values obtained from each algorithm
accuracy_values = {
    'XGBoost': acc_xgb,
    'AdaBoost': acc_ada,
    'LightGBM': acc_lgb,
    'CatBoost': acc_cat
}

# Plot accuracy curves for each algorithm
plt.figure(figsize=(10, 6))
for algo, acc in accuracy_values.items():
    plt.plot([algo], [acc], marker='o', label=algo)

plt.title('Accuracy Comparison of Gradient Boosting Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```



```
# Plot comparative analysis
plt.figure(figsize=(10, 6))
plt.bar(accuracy_values.keys(), accuracy_values.values(), color=['blue', 'green', 'orange', 'red'])
plt.title('Comparative Analysis of Gradient Boosting Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```



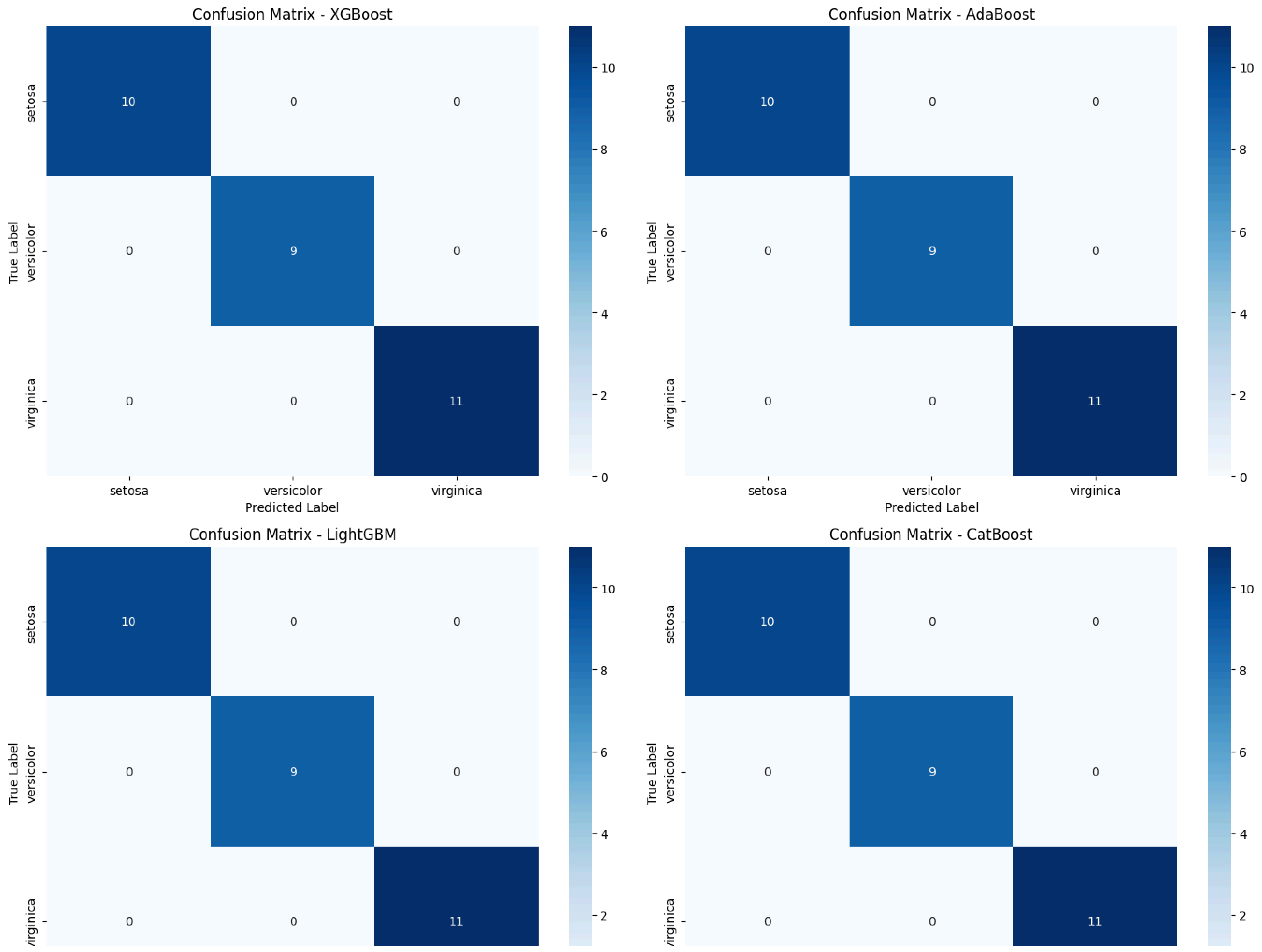
```
# Install the required module
!pip install scikit-learn==1.2.2
```

```
    Requirement already satisfied: scikit-learn==1.2.2 in /usr/local/lib/python3.10/dist-packages (1.2.2)
    Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.2) (1.25.2)
    Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.2) (1.11.4)
    Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.2) (1.3.2)
    Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.2) (3.3.0)
```

Lastly plot the predictions made by the gradient boosting algorithms (XGBoost, AdaBoost, LightGBM, and CatBoost) on the Iris dataset, one can visualize the decision boundaries for each algorithm. However, since the Iris dataset has four features, it's not feasible to visualize the decision boundaries in a simple 2D plot.

Instead, one can visualize the predictions made by each algorithm using a confusion matrix. This will allow us to see how well each algorithm is classifying the different classes in the Iris dataset.

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Get predictions for each algorithm
y_preds = {
    'XGBoost': y_pred_xgb,
    'AdaBoost': y_pred_ada,
    'LightGBM': y_pred_lgb,
    'CatBoost': y_pred_cat
}

# Plot confusion matrix for each algorithm
plt.figure(figsize=(15, 12))
for i, (algo, y_pred) in enumerate(y_preds.items(), 1):
    plt.subplot(2, 2, i)
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=iris.target_names, yticklabels=iris.target_names)
    plt.title(f'Confusion Matrix - {algo}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
plt.tight_layout()
plt.show()
```



This code will plot the confusion matrices for each gradient boosting algorithm using seaborn's heatmap, allowing us to visualize the predictions made by each algorithm on the Iris dataset. The confusion matrices show the percentage of samples in each class that were correctly classified by the model.