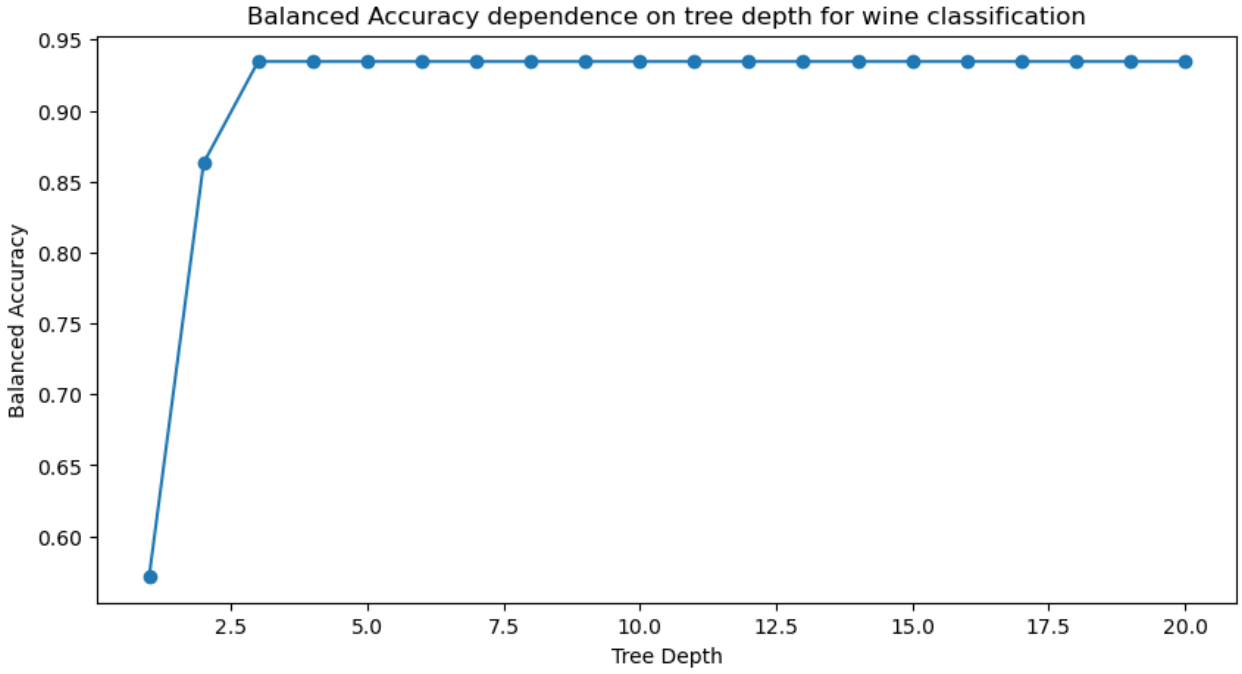
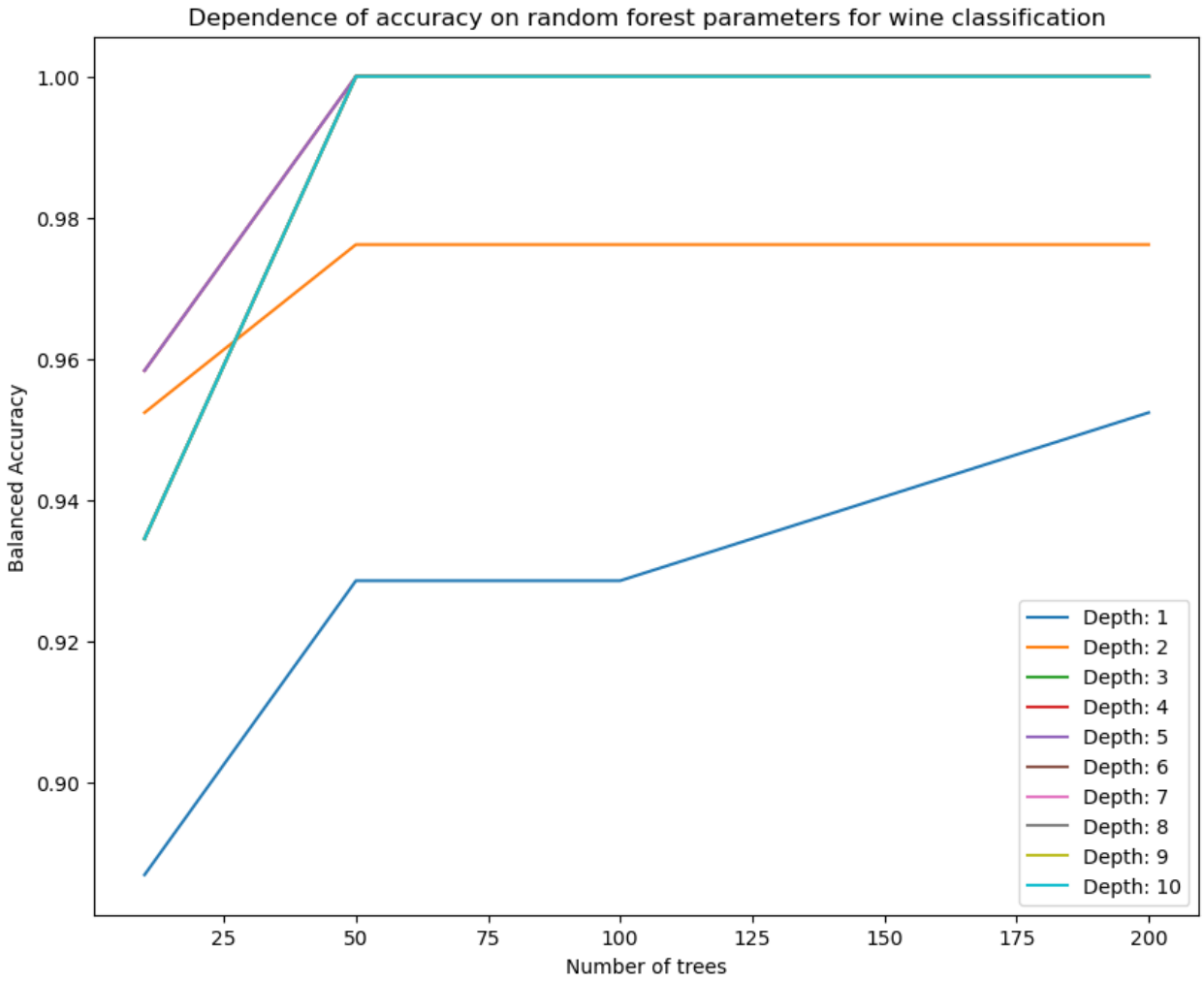
In [1]:

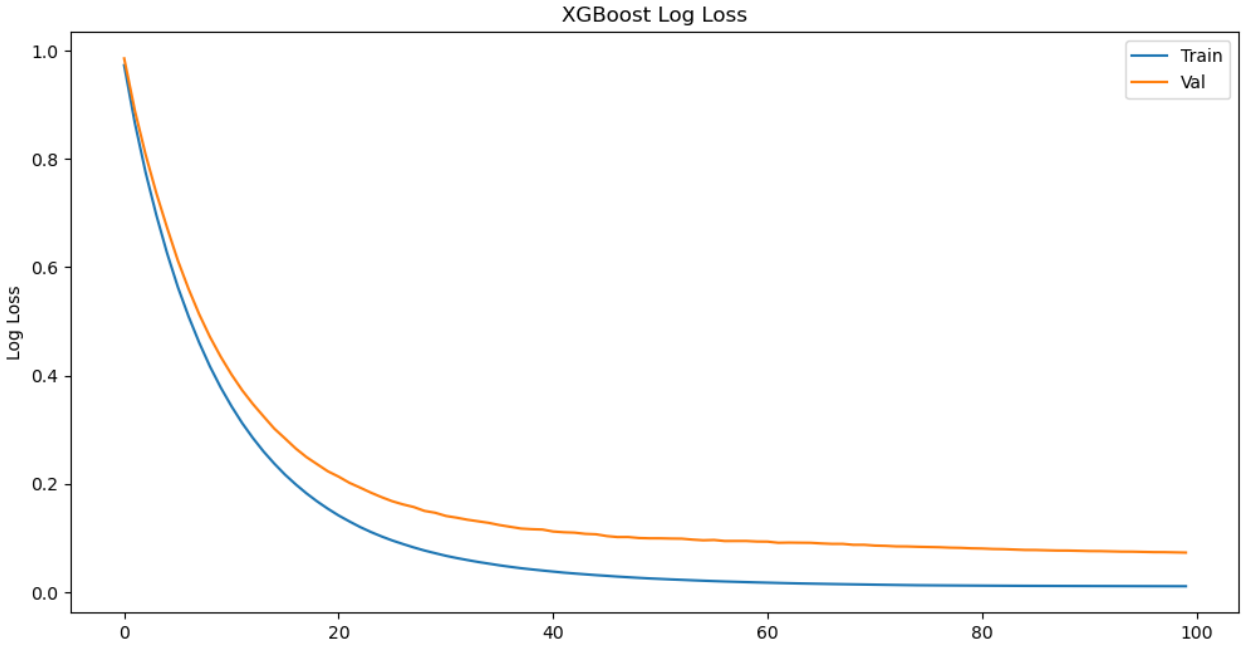
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
import matplotlib.pyplot as plt
from xgboost import XGBClassifier
import numpy as np
import os


# Uploading wine data
columns = ['Class', 'Alcohol', 'Malic Acid', 'Ash', 'Alcalinity of Ash',
           'Total Phenols', 'Flavanoids', 'Nonflavanoid Phenols', 'Proant
           'Color Intensity', 'Hue', 'OD280/OD315 of Diluted Wines', 'Pro
wine_data = pd.read_csv('wine.data', header=None, names=columns)

# Separation of the target variable and features
X = wine_data.drop('Class', axis=1)
y = wine_data['Class'] - 1  # Recoding classes

# Dividing the data into training and validation samples
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, ra

# Selection of the decision tree model
accuracy_scores_tree = {}
for depth in range(1, 21):
    model_tree = DecisionTreeClassifier(max_depth=depth, random_state=42)
    model_tree.fit(X_train, y_train)
    y_pred_tree = model_tree.predict(X_val)
    accuracy_scores_tree[depth] = balanced_accuracy_score(y_val, y_pred_t

# Visualization for the decision tree
plt.figure(figsize=(10, 5))
plt.plot(list(accuracy_scores_tree.keys()), list(accuracy_scores_tree.val
plt.xlabel('Tree Depth')
plt.ylabel('Balanced Accuracy')
plt.title('Balanced Accuracy dependence on tree depth for wine classifica
plt.show()

# Selection of a random forest model
depths = np.arange(1, 11)
n_estimators_range = [10, 50, 100, 200]
scores = np.zeros((len(depths), len(n_estimators_range)))

for i, depth in enumerate(depths):
    for j, n_estimators in enumerate(n_estimators_range):
        model_rf = RandomForestClassifier(max_depth=depth, n_estimators=n
        model_rf.fit(X_train, y_train)
        y_pred_rf = model_rf.predict(X_val)
        scores[i, j] = balanced_accuracy_score(y_val, y_pred_rf)

# Visualization for a random forest
plt.figure(figsize=(10, 8))
for idx, depth in enumerate(depths):
```

```
    plt.plot(n_estimators_range, scores[idx, :], label=f'Depth: {depth}')
plt.legend()
plt.xlabel('Number of trees')
plt.ylabel('Balanced Accuracy')
plt.title('Dependence of accuracy on random forest parameters for wine cl
plt.show()

# Selection of a gradient boosting model with visualization
model_gb = XGBClassifier(n_estimators=100, max_depth=3, learning_rate=0.1
eval_set = [(X_train, y_train), (X_val, y_val)]
model_gb.fit(X_train, y_train, eval_set=eval_set, verbose=False)

y_pred_gb = model_gb.predict(X_val)
print('Balanced Accuracy (Gradient boosting):', balanced_accuracy_score(y

# Getting learning outcomes for gradient boosting
results = model_gb.evals_result()

# Visualization for gradient boosting
epochs = len(results['validation_0']['mlogloss'])
x_axis = range(0, epochs)

fig, ax = plt.subplots(figsize=(12,6))
ax.plot(x_axis, results['validation_0']['mlogloss'], label='Train')
ax.plot(x_axis, results['validation_1']['mlogloss'], label='Val')
ax.legend()

plt.ylabel('Log Loss')
plt.title('XGBoost Log Loss')

# Saving a graph to a PDF file
plt.savefig('res.pdf')
print(os.getcwd())
```






```
Balanced Accuracy (Gradient boosting): 0.9583333333333334
/home/jovyan/work/check 2.3
```



In [ ]: