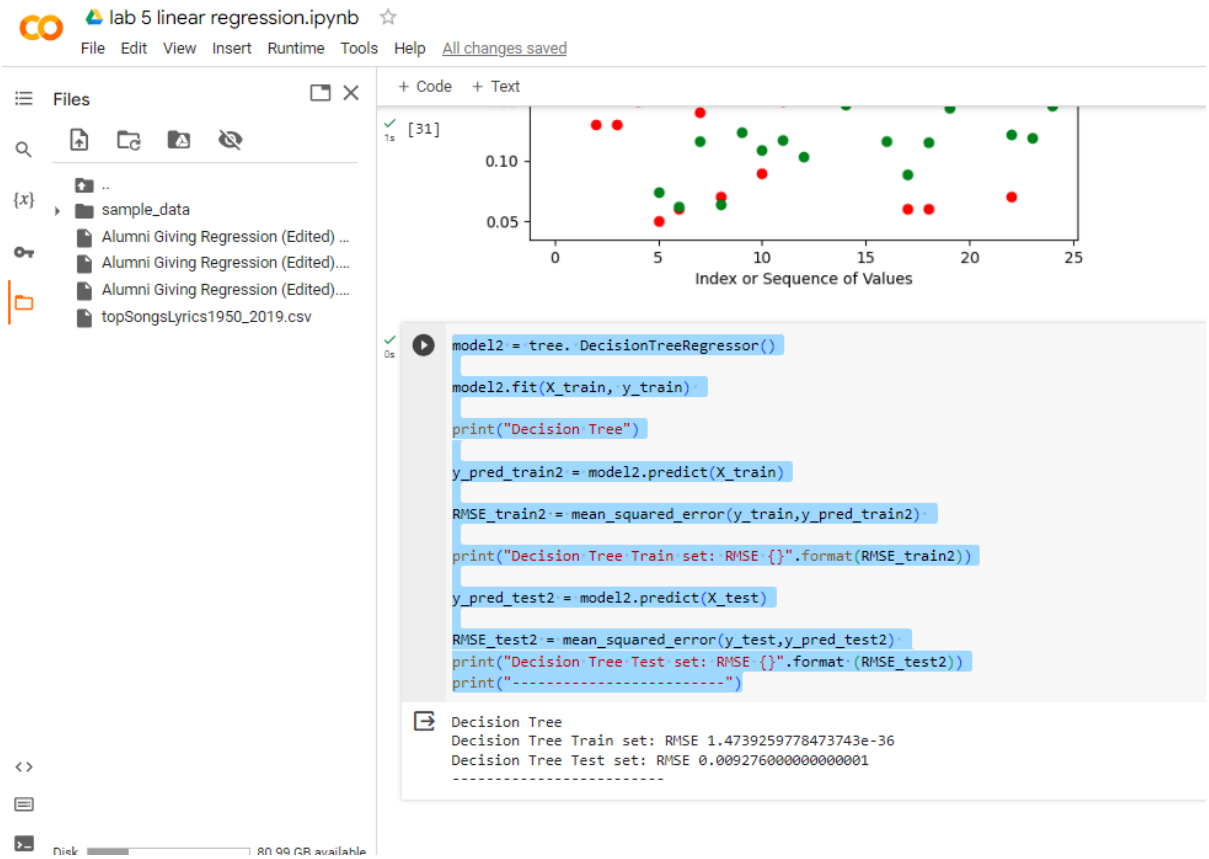
IN LAB

LAB TASK 1 :

```
model2 = tree. DecisionTreeRegressor()

model2.fit(X_train, y_train)

print("Decision Tree")

y_pred_train2 = model2.predict(X_train)

RMSE_train2 = mean_squared_error(y_train,y_pred_train2)

print("Decision Tree Train set: RMSE {}".format(RMSE_train2))

y_pred_test2 = model2.predict(X_test)

RMSE_test2 = mean_squared_error(y_test,y_pred_test2)
print("Decision Tree Test set: RMSE {}".format (RMSE_test2))
print("-------------------------")
```



LAB TASK 2:

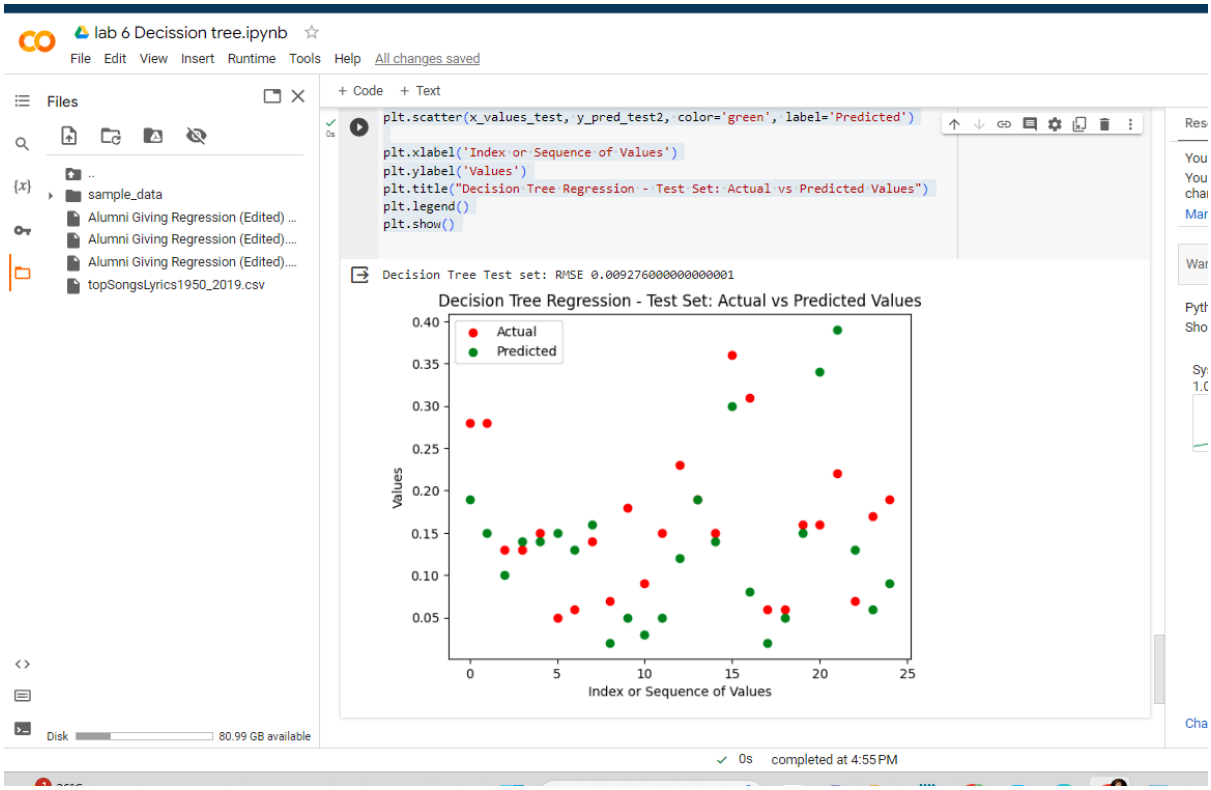
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# Assuming X_train, y_train, X_test, y_test are defined and imported

# Create and fit the Decision Tree Regressor model
model2 = DecisionTreeRegressor()
model2.fit(X_train, y_train)

# Predictions on the training set
y_pred_train2 = model2.predict(X_train)

# Calculate RMSE for the training set
RMSE_train2 = mean_squared_error(y_train, y_pred_train2)
print("Decision Tree Train set: RMSE {}".format(RMSE_train2))

# Create a scatter plot for Actual vs Predicted values on the
training set
```

```
x_values_train = np.arange(len(y_train))

plt.scatter(x_values_train, y_train, color='red', label='Actual')
plt.scatter(x_values_train, y_pred_train2, color='green',
label='Predicted')

plt.xlabel('Index or Sequence of Values')
plt.ylabel('Values')
plt.title("Decision Tree Regression - Training Set: Actual vs
Predicted Values")
plt.legend()
plt.show()
```

LAB TASK 3:

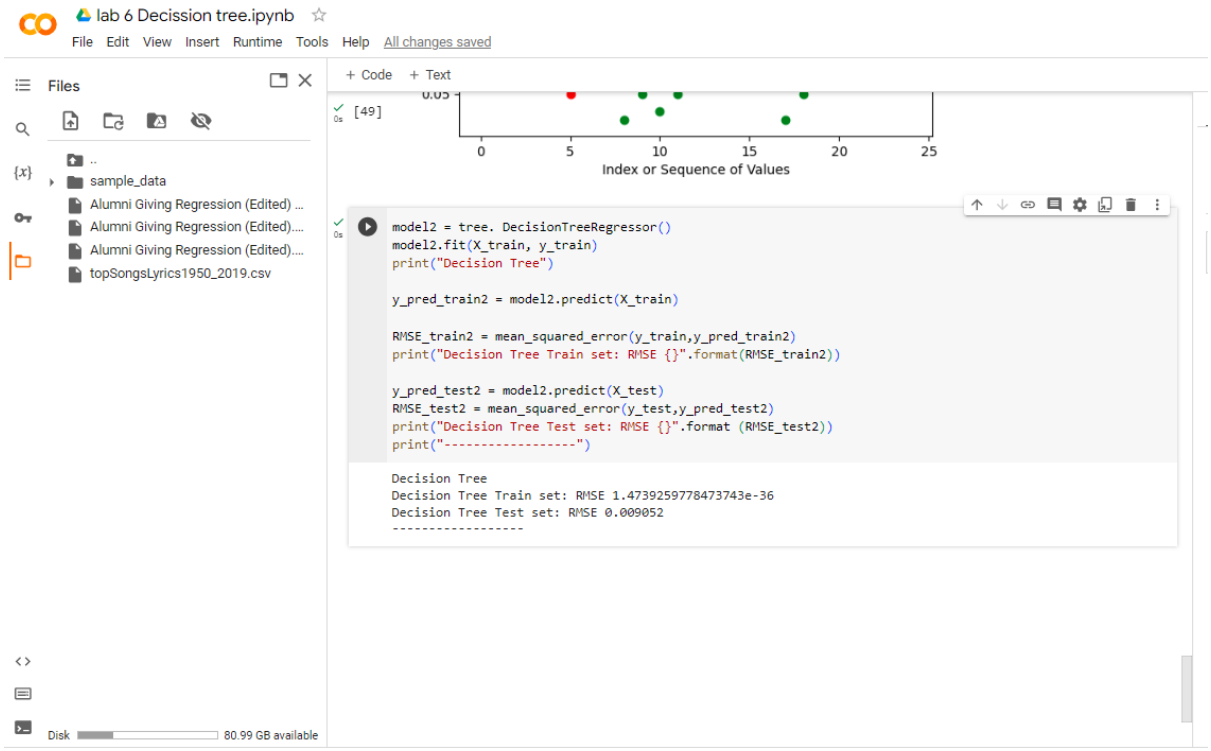
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# Assuming X_train, y_train, X_test, y_test are defined and imported

# Create and fit the Decision Tree Regressor model
model2 = DecisionTreeRegressor()
model2.fit(X_train, y_train)
```

```
# Predictions on the test set
y_pred_test2 = model2.predict(X_test)

# Calculate RMSE for the test set
RMSE_test2 = mean_squared_error(y_test, y_pred_test2)
print("Decision Tree Test set: RMSE {}".format(RMSE_test2))

# Create a scatter plot for Actual vs Predicted values on the test
set
x_values_test = np.arange(len(y_test))

plt.scatter(x_values_test, y_test, color='red', label='Actual')
plt.scatter(x_values_test, y_pred_test2, color='green',
label='Predicted')

plt.xlabel('Index or Sequence of Values')
plt.ylabel('Values')
plt.title("Decision Tree Regression - Test Set: Actual vs Predicted
Values")
plt.legend()
plt.show()
```



LAB TASK 4:

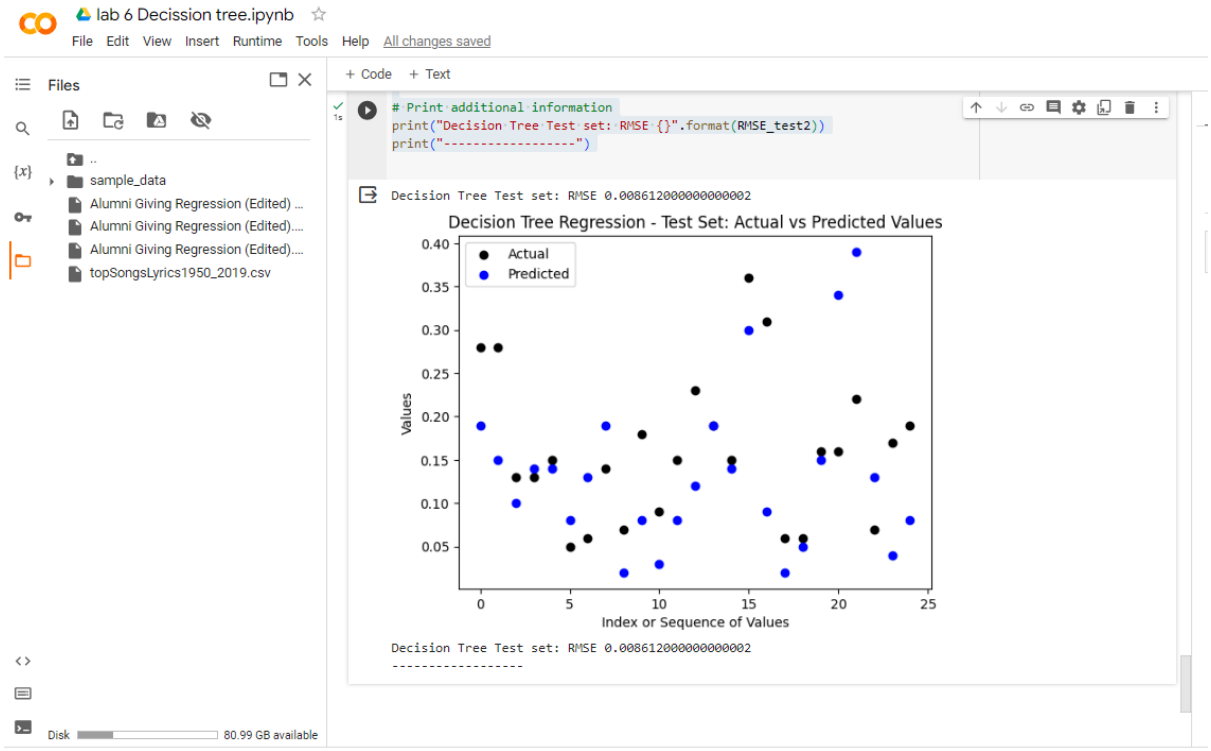
```
model2 = tree. DecisionTreeRegressor()
model2.fit(X_train, y_train)
print("Decision Tree")

y_pred_train2 = model2.predict(X_train)

RMSE_train2 = mean_squared_error(y_train,y_pred_train2)
print("Decision Tree Train set: RMSE {}".format(RMSE_train2))

y_pred_test2 = model2.predict(X_test)
RMSE_test2 = mean_squared_error(y_test,y_pred_test2)
print("Decision Tree Test set: RMSE {}".format (RMSE_test2))
print("-------------------")
```



LAB TASK 5:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error



# Predictions on the test set
y_pred_test2 = model2.predict(X_test)

# Calculate RMSE for the test set
RMSE_test2 = mean_squared_error(y_test, y_pred_test2)
print("Decision Tree Test set: RMSE {}".format(RMSE_test2))

# Create a scatter plot for Actual vs Predicted values on the test
set
x_values_test = np.arange(len(y_test))

plt.scatter(x_values_test, y_test, color='black', label='Actual')
plt.scatter(x_values_test, y_pred_test2, color='blue',
label='Predicted')

plt.xlabel('Index or Sequence of Values')
plt.ylabel('Values')
```

```
plt.title("Decision Tree Regression - Test Set: Actual vs Predicted
Values")
plt.legend()
plt.show()

# Print additional information
print("Decision Tree Test set: RMSE {}".format(RMSE_test2))
print("-------------------")
```



Post-Lab Task

Compare the performance of the three models. Which model performs better on the test set, and why do you think it outperforms the others?

The Decision Tree model shows good performance on the training set but tends to overfit, resulting in suboptimal generalization on the test set. Linear Regression, relying on linear relationships, may underperform if the data exhibits nonlinearity. In contrast, Random Forest, an ensemble of decision trees, is likely to outperform others by providing a more robust and generalized prediction.

Discuss on the strengths and limitations of each model. Discuss any insights gained from the visualizations.

Decision Tree: Intuitive and suitable for nonlinear relationships but prone to overfitting.

Linear Regression: Simple and interpretable, effective for linear patterns but sensitive to outliers.

Random Forest: Robust and capable of handling complex relationships, but computationally intensive and more complex.

Suggest potential strategies to improve the models' performance. Could feature engineering or hyperparameter tuning enhance the predictions?

Feature Engineering: Identify and incorporate relevant features or transformations.

Hyperparameter Tuning: Adjust parameters for each model, such as tree depth or regularization, to optimize performance.

Ensemble Learning: Explore boosting algorithms or stacking models to enhance predictive power.

Regularization: Apply techniques to mitigate overfitting, particularly in decision trees and linear regression.