**In-Lab**

**In-Lab Task 1**

|  |  |
| --- | --- |
| **Code:**  model2 = DecisionTreeRegressor()  model2.fit(X\_train, y\_train)  print("Decision Tree")  print("===============================")  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("===============================")  **Output:**   |  | | --- | | Decision Tree  ===============================  Decision Tree Train set: RMSE 1.4739259778473743e-36  Decision Tree Test set: RMSE 0.008496  =============================== | |

**In-Lab Task 2**

|  |
| --- |
| **Code:**  x\_values = np.arange(len(y\_pred\_train2))  plt.scatter(x\_values, y\_pred\_train2, color = 'red', label= 'Train')  plt.scatter(x\_values, y\_train, color = 'purple', label= 'Actual')  plt.xlabel("Index or Sequence of Values")  plt.ylabel("Values")  plt.title("Decision Tree Regression - Training Set")  plt.legend()  plt.show()  **Output:** |

**In-Lab Task 3**

|  |
| --- |
| **Code:**  x\_values = np.arange(len(y\_pred\_test2))  plt.scatter(x\_values, y\_pred\_test2, color = 'blue', label= 'Predicted')  plt.scatter(x\_values, y\_test, color = 'purple', label= 'Actual')  plt.xlabel("Index or Sequence of Values")  plt.ylabel("Values")  plt.title("Decision Tree Regression - Testing Set")  plt.legend()  plt.show()  **Output:** |

**In-Lab Task 4**

|  |  |
| --- | --- |
| **Code:**  model3 = RandomForestRegressor()  model3.fit(X\_train, y\_train)  print("Random Forest")  print("===============================")  y\_pred\_train3 = model3.predict(X\_train)  RMSE\_train3 = mean\_squared\_error(y\_train, y\_pred\_train3)  print("Random Forest Train set: RMSE {}".format(RMSE\_train3))  y\_pred\_test3 = model3.predict(X\_test)  RMSE\_test3 = mean\_squared\_error(y\_test,y\_pred\_test3)  print("Random Forest Test set: RMSE {}".format(RMSE\_test3))  print("===============================")  **Output:**   |  | | --- | | Random Forest  ===============================  Random Forest Train set: RMSE 0.0004964972448979589  Random Forest Test set: RMSE 0.004843255999999997  =============================== | |

**In-Lab Task 5**

|  |
| --- |
| **Code:**  x\_values = np.arange(len(y\_pred\_train3))  plt.scatter(x\_values, y\_pred\_train3, color = 'green', label= 'Predicted')  plt.scatter(x\_values, y\_train, color = 'purple', label= 'Actual')  plt.xlabel("Index or Sequence of Values")  plt.ylabel("Values")  plt.title("Random Forest Regression - Training Set")  plt.legend()  plt.show()  **Output:** |

**Post-Lab**

**Post-Lab Task**

|  |
| --- |
| * **Model Comparison:**   In comparing the performance of **Linear Regression**, **Decision Tree Regression**, and **Random Forest Regressor**, each model has distinct strengths and limitations.   * + Linear Regression is simple and interpretable but assumes a linear relationship and is sensitive to outliers.   + Decision Tree Regression captures complex, non-linear relationships but is prone to overfitting.   + Random Forest Regressor, often performing well, reduces overfitting through aggregation but is less interpretable. * **Insights From Visualizations:**   Visualizations, such as scatter plots, offer insights into the model behaviors. Random Forest outperforms others due to its ability to handle complexity and reduce overfitting.   * **Performance Enhancement:**   To enhance performance, consider feature engineering, hyperparameter tuning, ensemble methods, handling outliers, cross-validation, and regularization. Ultimately, the choice depends on dataset characteristics, considering interpretability and computational cost. |