**In-Lab**

**In-Lab Task 1**

|  |  |
| --- | --- |
| **Code:**  # Importing libraries needed  # Note that keras is generally used for deep learning as well from keras.models import Sequential  from keras.layers import Dense, Dropout  from sklearn.metrics import classification\_report, confusion\_matrix  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_squared\_error  import numpy as np  from sklearn import linear\_model  from sklearn import preprocessing  from sklearn import tree  from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor  import pandas as pd  import csv  import matplotlib.pyplot as plt  print("Libraries Imported Successfully")  **Output:**   |  | | --- | | Libraries Imported Successfully | |

**In-Lab Task 2**

|  |  |
| --- | --- |
| **Code:**  np.random.seed(7)  df = pd.read\_csv("Alumni Giving Regression (Edited).csv", delimiter=",")  dd\_df\_1 = df.head()  print("Imported Dataset Successfully")  **Output:**   |  | | --- | | Imported Dataset Successfully | |

**In-Lab Task 3**

|  |  |
| --- | --- |
| **Code:**   * **Data Pattern**   dd\_df\_1   * **Description of Data**   dd\_df\_1.describe().T  **Output:**   * **Data Pattern**  |  | | --- | |  |  * **Description of Data** |

**In-Lab Task 4**

|  |
| --- |
| **Code:**  corr = df.corr(method='pearson')  corr  **Output:** |

**In-Lab Task 5**

|  |  |
| --- | --- |
| **Code:**  Y\_POSITION = 5  model\_1\_features = [i for i in range (0, Y\_POSITION)]  X = df.iloc[:,model\_1\_features]  Y = df.iloc[:,Y\_POSITION]  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,Y,test\_size=0.20, random\_state=2020)  print("Data Readied for Model Training")  **Output:**   |  | | --- | | Data Readied for Model Training | |

**In-Lab Task 6**

|  |
| --- |
| **Code:**  model1 = linear\_model.LinearRegression()  model1.fit(X\_train, y\_train)  y\_pred\_train1 = model1.predict(X\_train)  print("Regression")  print("================================")  RMSE\_train1 = mean\_squared\_error(y\_train,y\_pred\_train1)  print("Regression Train set: RMSE {}".format(RMSE\_train1))  print("================================")  y\_pred1 = model1.predict(X\_test)  RMSE\_test1 = mean\_squared\_error(y\_test,y\_pred1)  print("Regression Test set: RMSE {}".format(RMSE\_test1))  print("================================")  coef\_dict = {}  for coef, feat in zip(model1.coef\_,model\_1\_features):  coef\_dict[df.columns[feat]] = coef  print(coef\_dict)  x\_values = np.arange(len(y\_test))  plt.scatter(x\_values, y\_test, color = 'red', label= 'Actual')  plt.scatter(x\_values, y\_pred1, color = 'green', label= 'Predicted')  plt.xlabel("Index or Sequence of Values")  plt.ylabel("Values")  plt.title("Actual vs Predicted Values")  plt.legend()  plt.show()  **Output:** |

**Post-Lab**

**Post-Lab Task**

|  |
| --- |
| To boost my linear regression model's accuracy, I'd consider these steps:   * **Feature Selection:** I'd check if all the features I'm using are important for predictions. Sometimes, less is more. * **Outlier Handling:** If there are weird data points messing with things, I'd investigate those and decide if they should stay or be removed. * **Normalization or Standardization:** Making sure all my features are on the same scale can help the model perform better. * **Polynomial Features:** If the relationship between the features and the target isn't exactly straight, I might try adding some polynomial features to catch those curves.   To check if my model is really doing well on new data, I would do the following:   * **Split the Data:** Divide my data into two parts – one for training and one for testing. Train my model on the training set and see how well it predicts the test set. * **Check for Overfitting:** If it's doing super well on the training set but poorly on the test set, I might need to simplify things a bit because it might have memorized the dataset rather than making predictions.   Doing these things helps me make sure my model isn't just memorizing the training data but can make good predictions on new stuff too. |