**In-Lab**

**In-Lab Task 1**

|  |
| --- |
| **Code:**  **Part1:**  # 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 seaborn as sns  import matplotlib.pyplot as plt  print("Libraries Imported Successfully!")  **Output1:**    **Part2:**  # Read Data and fix seed  # Fix random seed for reproducibility  np.random.seed(7)  df = pd.read\_csv("Alumni Giving Regression (Edited).csv", delimiter=",")  df = df.dropna()  df.head()  **Output2:**    **Part3:**  df.info()  **Output3:**  **Part4:**  df.describe()  **Output4:**    **Part5:**  sns.boxplot(data=df).set(xlabel='Columns',ylabel='Values', title='Box Plot of All Values')  plt.show()  **Output5:** |

**In-Lab Task 2**

|  |
| --- |
| **Code:**  **Part1:**  quantile99 = df.iloc[:,0].quantile (0.99)  df1 = df[df.iloc[:,0] < quantile99]  sns.boxplot(data=df1).set(xlabel='Columns',ylabel='Values', title='Box Plot of Top 99% of Values')  plt.show()  **Output1:**    **Part2:**  quantile1 = df.iloc[:,0].quantile (0.01)  quantile99 = df.iloc[:,0].quantile (0.99)  df2 = df[(df.iloc[:,0] > quantile1) & (df.iloc[:,0] <quantile99)]  sns.boxplot(data=df2).set(xlabel='Columns',ylabel='Values', title='Box Plot of 1%-99% of Values')  plt.show()  **Output2:**    **Part3:**  model3 = RandomForestRegressor()  X\_train = df[['A','B','C','D','F']]  y\_train = df[['E']]  y\_train = np.array(y\_train).ravel() # Makes the target variable "y\_train" into a 1D array  model3.fit(X\_train, y\_train)  RF = model3  importances = RF.feature\_importances\_  std = np.std([tree.feature\_importances\_ for tree in RF.estimators\_], axis = 0)  indices = np.argsort(importances)[::-1]  # Print the feature ranking  print("========================")  print(" Feature Ranking")  print("========================")  for f in range(X\_train.shape[1]):  print("{Feature#%s}=>(%f)" %(indices[f], importances[indices[f]]\*100))  print("========================")  **Output3:** |

**In-Lab Task 3**

|  |
| --- |
| **Code:**  **Part1:**  indices\_top3= indices[:3]  print("The top 3 features indexes:",indices\_top3)  Y\_position=5  TOP\_N\_FEATURE = 3  X = df.iloc[:, indices\_top3]  Y = df.iloc[:,Y\_position]  # create model  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.20, random\_state = 2020)  #Model 1 linear regression  model1 = linear\_model.LinearRegression()  model1.fit(X\_train, y\_train)  y\_pred\_train1 = model1.predict(X\_train)  print(" ======================")  print(" Regression")  print("================================================")  RMSE\_train1 = mean\_squared\_error(y\_train, y\_pred\_train1)  print("Regression TrainSet: RMSE {}".format(RMSE\_train1))  print("================================================")  y\_pred1 = model1.predict(X\_test)  RMSE\_test1 = mean\_squared\_error(y\_test,y\_pred1)  print("Regression Testset: RMSE {}".format(RMSE\_test1))  print("================================================")  **Output1:** |