Project Description:

For this course, three projects have been defined. Students need to be divided into three groups, and a leader should be selected for each group for communication with me. The group leader is responsible for project presentation and communication with me. Progress reports for both projects need to be submitted every **two weeks**. Extra activities beyond the project will earn additional marks for this course.

Project Objectives:

The goal of these projects is to design a program (in Python and web-based) for flood prediction using a machine/deep learning algorithm. There is no need to collect data for these projects; the data will be provided to you in a CSV file. Your task is to design these programs in a user-friendly manner. One group will design the program based on Python, and the other group will design the web-based program. The design and user-friendliness of the program are of utmost importance, and a better-designed program will receive a higher grade.

Prerequisites:

- Basic familiarity with machine learning algorithms (regression) is required (Project 1 and Project 2).
- Basic familiarity with deep learning algorithms (regression) is required (Project 3).
- The choice of software, libraries, etc. is the responsibility of the student and there is no limit in this matter.

Please send your questions and projects to this email (vrazavi70@gmail.com).

Dr. Seyed Vahid Razavi-Termeh

****

Project 1: Python-based Machine Learning App

The objective of this project is to design a Python program that, in addition to being user-friendly, includes the following tasks:

1. Selecting input data (via CSV file) (Data.CSV)
2. Selecting independent and dependent data Independent data (all columns except the Flood column) Dependent data (Flood column)
3. Selecting the percentage ratio of training and testing data. The program should allow the selection of the following ratios for training and testing data:
    - 90:10 (90% training data, 10% testing data)
    - 80:20 (80% training data, 20% testing data)
    - 70:30 (70% training data, 30% testing data)
    - 60:40 (60% training data, 40% testing data)

4. Choosing a machine learning model for prediction on training and testing data: Machine learning models in regression mode (Random Forest or XGBoost)
5. Running the model on training and testing data and evaluating the results with the following metrics: RMSE, MAE, and R^2
6. Displaying a histogram of errors between actual (Flood) and predicted data in step 5
7. Displaying the determination of the importance of criteria using the machine learning model
8. Calling new data and predicting on this data: New data in CSV format is called initially. This data has two columns x and y (longitude and latitude) and other columns similar to independent data in step 2. After predicting the model on this data using the prediction output and values of x and y, display a density heat map here (you can use MapBox).

Extra Credit:

- Call and run a greater number of machine learning algorithms in step four.
- Display the boundary of the study area in step eight.

****

Project 2: Web-based Machine Learning App

The objective of this project is to design a web-based program (you can use this course for help: https://www.coursera.org/projects/machine-learning-streamlit-python) that, in addition to being user-friendly, includes the following tasks:

1. Selecting input data (via CSV file) (Data.CSV)
2. Selecting independent and dependent data Independent data (all columns except the Flood column) Dependent data (Flood column)
3. Selecting the percentage ratio of training and testing data. The program should allow the selection of the following ratios for training and testing data:
    - 90:10 (90% training data, 10% testing data)
    - 80:20 (80% training data, 20% testing data)
    - 70:30 (70% training data, 30% testing data)
    - 60:40 (60% training data, 40% testing data)
4. Choosing a machine learning model for prediction on training and testing data: Machine learning models in regression mode (Random Forest or XGBoost)
5. Running the model on training and testing data and evaluating the results with the following metrics: RMSE, MAE, and R^2
6. Displaying a histogram of errors between actual (Flood) and predicted data in step 5
7. Displaying the determination of the importance of criteria using the machine learning model
8. Calling new data and predicting on this data: New data in CSV format is called initially. This data has two columns x and y (longitude and latitude) and other columns similar to independent data in step 2. After predicting the model on this data (introducing independent columns) using the prediction output and values of x and y, display a density heat map here (you can use MapBox).

Extra Credit:

- Steps 6 and 7 are considered extra credit.
- Choose a greater number of algorithms in step four.
- Display the area boundary in step 8 using https://mapshaper.org/ or Leaflet.

****

Project 3: Python-based Deep Learning App

The objective of this project is to design a Python program that, in addition to being user-friendly, includes the following tasks:

1. Selecting input data (via CSV file) (Data.CSV)
2. Selecting independent and dependent data Independent data (all columns except the Flood column) Dependent data (Flood column)
3. Selecting the percentage ratio of training and testing data. The program should allow the selection of the following ratios for training and testing data:
    a. 90:10 (90% training data, 10% testing data)
    b. 80:20 (80% training data, 20% testing data)
    c. 70:30 (70% training data, 30% testing data)
    d. 60:40 (60% training data, 40% testing data)
4. Choosing a deep learning model for prediction on training and testing data: Deep learning models in regression mode (CNN or RNN)
5. Running the model on training and testing data and evaluating the results with the following metrics: RMSE, MAE, and R^2
6. Displaying a histogram of errors between actual (Flood) and predicted data in step 5
7. Calling new data and predicting on this data: New data in CSV format is called initially. This data has two columns x and y (longitude and latitude) and other columns similar to independent data in step 2. After predicting the model on this data using the prediction output and values of x and y, display a density heat map here (you can use MapBox).

Extra Credit:

- Call and run a greater number of deep learning algorithms in step four.
- Display the boundary of the study area in step seven.

****

An Example of Different Steps for Executing a Machine Learning Algorithm in Python:

1. **Import the Required Libraries:**

```
import numpy as np
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from permetrics.regression import RegressionMetric
from sklearn.ensemble import RandomForestRegressor
import sklearn.metrics as metrics
```

2. **Load the Data:**

```
ReadData = pd.read_csv('/content/drive/MyDrive/…../Data.csv')
```

3. **Split the Data into Training and Test Sets:**

```
X = ReadData.drop([Flood], axis = 1)
y = ReadData [Flood]
```

4. **Set the Data Ratio:**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=True,
random_state=1)
```

5. **Create and Train the Model:**

```
model = RandomForestRegressor(random_state = 1).fit(X_train, y_train)
```

6. **Predict Using the Model on Training and Test Data:**

```
yhat_train = model.predict(X_train)
yhat_test = model.predict(X_test)
```

7. **Evaluate the Results for Training and Test Data:**

```
r2 = metrics.r2_score(y_train, yhat_train)
mae = metrics.mean_absolute_error(y_train, yhat_train)
mse = metrics.mean_squared_error(y_train, yhat_train)
rmse = np.sqrt(mse)
```

```
r2 = metrics.r2_score(y_test, yhat_test)
mae = metrics.mean_absolute_error(y_test, yhat_test)
mse = metrics.mean_squared_error(y_test, yhat_test)
rmse = np.sqrt(mse)
```

8. **Load New Data:**

```
NewData = pd.read_csv('/content/drive/MyDrive/…../NewData.csv')
```

9. **Predict Using the Model on New Data:**

```
output_prediction = model.predict(Newdata)
```

10. **Calculate the Error Histogram:**

```
error = y_test - y_test_pred
plt.hist(error, bins=30)
plt.xlabel("Prediction Error")
plt.ylabel("Count")
plt.title("Error Histogram")
plt.show()
```