Data Science Guide By Abdul Rehman
Month 01 (Data Analysis)
Week 01

🔴 **Task 1 ->An introduction to Git.** This will include installing Git and learning some basic commands: init, add, commit, etc.

Here's what you need to do:

Install Git: Make sure Git is installed on your machine. You can find the installation instructions here: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
Basic Commands (for example):
git init: Initialize a new Git repository.
git add: Add files to the staging area.
git commit: Commit your changes to the repository.
Documentation:

1. Run the above commands on your machine.
2. Take screenshots of each step (installation, initializing a repository, adding files, and committing changes).
3. Compile these screenshots into a Word document.

Resource: https://www.w3schools.com/git/git_staging_environment.asp?remote=github

🔴 **Task 2 -> Git and GitHub**
Now that you have a bit of information regarding Git, it's time to learn about GitHub and how to use both Git and GitHub collectively. For now, I don't expect any extraordinary command on both of these, but you must complete the following steps:

Create a GitHub Account: If you don't already have one, sign up for a GitHub account.

Create a Public Repository:

Title it as follows: Data Science - YOUR NAME
Ensure the repository is public so we can all view it.
Organize Your Repository:

After creating the repository, create a folder within it named Task 1.
I suggest making a separate folder for each task on your local system. When you complete a task, simply drag and drop that folder into this repository.

🔴Task 3: Introduction to Jupyter Notebook and Installation

This will be our last task related to setting up everything. After this task, we will start with Python.

In this task, you need to set up Jupyter Notebook on your systems. Explore it and see how you can add fancy headings to it.

For this task, you will submit a Jupyter Notebook. Like the previous tasks, create a folder named Task 3, place the notebook in it, and drag and drop it into the same repository (I will check it there).

Resources:
1. https://www.geeksforgeeks.org/install-jupyter-notebook-in-windows/
2. https://www.geeksforgeeks.org/how-to-use-jupyter-notebook-an-ultimate-guide/

🔴 Task 4: Python Basics (Variable, Datatypes, Operators)

As I mentioned earlier, by Task 3, we have completed setting up the environment for future tasks. From Task 4, we will start with Python. I mentioned in the meeting that for every task I provide, try to think of a practical application you can create using the tools you have learned so far.

You can choose any project of your choice. Here is an example:

Simple DMAS Calculator:
Create a program that prompts the user to input two numbers. The program will then calculate and display the sum, difference, product, and quotient of the two numbers.

Submission: Jupyter Notebook on GitHub (Folder -> Task 4)
Resource: https://www.w3schools.com/python/default.asp

🔴 Task 5: If conditions, for loops, while loops, and functions using Python.

This will be our last task until Eid. I recommend you all to practice these concepts. Try to create as many mini-projects as possible.

For example, you can use one Jupyter notebook with multiple cells representing a mini-project.

The best way to master loops is by printing patterns. I will share some examples soon. You can consider those or try some yourself as well.

The deadline for all tasks (previous and this one) is Sunday, June 16, 2024. I will close the form on Sunday, and after that, no one will be allowed to submit their repository link. I have seen some repositories and have given personal suggestions to some fellows. I will share the collective results soon for the tasks we have completed so far.

Resources:
1. https://www.w3schools.com/python/gloss_python_else.asp
2. https://www.w3schools.com/python/python_while_loops.asp
3. https://www.w3schools.com/python/python_for_loops.asp
4. https://www.w3schools.com/python/python_functions.asp

Week 02

Deadline: 21/6/2024

🔴 **Task 6: File Handling, Exception Handling**

Look at this topic and try to create a small practical application. For example, you could create an application that records and retrieves data from a file. One such example is storing student data in a file. When the application starts, it should load the recorded data and provide options for searching and deleting records. You can use multiple attributes such as student_id, student_name, student_age, and student_grade.

The student_id should be unique, and searching/deleting will be done using this. When adding a new record, no duplicate IDs should be allowed. Also, display the list of available students.

This is the last task for basic programming. After this, we will move towards OOP and some common data structures in Python (dictionary, list, set, tuple).

Resource: https://www.w3schools.com/python/python_file_handling.asp

Deadline: 22/6/2024

🔴 **Task 7: Object-Oriented Programming (OOP) basics.**

This will include classes, objects, and methods.
A Practical Example:

Create a simple Library Management System using OOP concepts to manage books, members, and borrowing activities.

Key Features:

1. **Book Class:**
    - Attributes: title, author, ISBN, status (available/borrowed)
    - Methods: display info, mark as borrowed, mark as returned
2. **Member Class:**
    - Attributes: name, member ID, borrowed books (list)
    - Methods: borrow book, return book, display info

3. **Library Class:**
    - Attributes: list of books, list of members
    - Methods: add book, register member, issue book, return book, display all books, display all members

Steps:

1. Define `Book` class with necessary attributes and methods.
2. Define `Member` class with necessary attributes and methods.
3. Define `Library` class to manage books and members.
4. Test by creating instances, adding books, registering members, and simulating borrowing/returning books.

Resource: https://www.w3schools.com/python/python_classes.asp

Deadline: 23/6/2024

🔴 **Task 8: Simple Data Structures in Python (Dictionary, Set, List, Tuple)**

Description: This task will cover the basics of Python's built-in data structures: Dictionary, Set, List, and Tuple. You will learn how to create, manipulate, and use these data structures effectively in Python programming.

Key Topics:

1. **List:**
    - Creation: `my_list = [1, 2, 3, 4]`
    - Accessing elements: `my_list[0]`
    - Modifying elements: `my_list[1] = 5`
    - Adding elements: `my_list.append(6)`
    - Removing elements: `my_list.remove(3)`
2. **Tuple:**
    - Creation: `my_tuple = (1, 2, 3, 4)`
    - Accessing elements: `my_tuple[0]`
    - Immutable nature: Elements cannot be modified after creation.
3. **Set:**
    - Creation: `my_set = {1, 2, 3, 4}`
    - Adding elements: `my_set.add(5)`
    - Removing elements: `my_set.remove(2)`
    - Unique elements: No duplicates allowed.
4. **Dictionary:**
    - Creation: `my_dict = {'a': 1, 'b': 2, 'c': 3}`
    - Accessing elements: `my_dict['a']`
    - Modifying elements: `my_dict['b'] = 4`

- Adding elements: `my_dict['d'] = 5`
- Removing elements: `del my_dict['c']`

Practical Mini Project: Inventory Management System

Description: Develop a simple Inventory Management System using Python's data structures to manage products, categories, and stock levels.

Steps:

1. **Define Data Structures:**
   - Use a dictionary to store product details (e.g., `products = {'ID001': {'name': 'Product1', 'category': 'Category1', 'stock': 50}}`).
   - Use a list to maintain the order of product additions.
   - Use a set to track unique categories.
   - Use a tuple to store immutable product information (e.g., product name and category).
2. **Implement Functions:**
   - Add a product: `add_product(product_id, name, category, stock)`
   - Update stock: `update_stock(product_id, quantity)`
   - Display all products: `display_products()`
   - Display products by category: `display_by_category(category)`
3. **Testing:**
   - Add multiple products.
   - Update stock levels.
   - Display all products and filter by category.

Week 03

Deadline: 24/6/2024

🔴 **Task 9: Introduction to Statistics**

Statistics and probability are very important when we talk about data analysis. You should at least know mean, median, mode, some common distributions (for example: normal distribution, binomial distribution, Poisson distribution, and uniform distribution) and some concepts of probability. **You will learn and try to solve some related problems by hand, take screenshots, and submit a PDF.**

Resource: https://www.geeksforgeeks.org/introduction-of-statistics-and-its-types/

Deadline: 25/6/2024

🔴 **Task 10: Introduction to NumPy (arrays, basic operations)**

Now that you have a basic knowledge of statistics and probability, it is time to code it. Try to learn the basics of NumPy (arrays, etc.), perform some array operations in multiple dimensions, and try to solve the problems you solved in Task 9 using NumPy.

Everyone must do:

1. Create a 1D NumPy array containing the integers from 0 to 9.
2. Create a 2D NumPy array (3x3) containing random integers between 1 and 20.
3. Create a 3D NumPy array with dimensions (2, 3, 4) filled with ones.
4. Add two 1D arrays element-wise.
5. Multiply two 2D arrays element-wise.
6. Calculate the dot product of two matrices.
7. Calculate the mean, median, and standard deviation of a 1D array.
8. Find the maximum and minimum values in a 2D array.
9. Generate an array of 1000 random numbers from a normal distribution with a mean of 0 and a standard deviation of 1.
10. Create a 2D array of shapes (5, 5) with random integers from a uniform distribution between 10 and 50.
11. Calculate the cumulative sum of a 1D array.
12. Compute the correlation coefficient matrix of a 2D array.
13. Simulate rolling a six-sided die 1000 times and count the frequency of each outcome.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Deadline: 26/6/2024

🔴 Task 11: NumPy advanced operations (indexing, slicing, broadcasting)

Now that you have learned the basic concepts of NumPy and know how to create arrays, it's time to explore it further.

Everyone must do:

For serials 1 to 15:

Indexing and Slicing

1. Given a 2D array of shape (5, 5), extract a 3x3 sub-array starting from the element at position (1, 1).
2. From a 3D array of shape (4, 3, 2), extract all elements in the first two rows and all columns of the second slice along the third axis.
3. Given an array of integers, use fancy indexing to extract elements at positions [1, 3, 4, 7].
4. Given a 2D array, use fancy indexing to select rows [0, 2, 3] and columns [1, 3].
5. From a 1D array of random integers, extract all elements that are greater than 10.

6. Given a 2D array of shape (5, 5), replace all elements greater than 15 with the value 0.

Broadcasting

1. Add a 1D array of shape (3,) to each row of a 2D array of shape (4, 3).
2. Multiply a 2D array of shape (3, 3) by a 1D array of shape (3,).
3. Create two 2D arrays of shapes (3, 1) and (1, 4) respectively, and perform element-wise addition.
4. Given a 3D array of shape (2, 3, 4), add a 2D array of shape (3, 4) to each 2D slice along the first axis.

Some more

1. Given a 2D array, use slicing to extract every second row and every second column, then add a 1D array to each row of the sliced array.
2. From a 3D array of shape (4, 3, 2), extract a sub-array using slicing and then use broadcasting to subtract a 2D array from each slice along the third axis.
3. Given a 2D array, extract the diagonal elements and create a 1D array.
4. Use slicing to reverse the order of elements in each row of a 2D array.
5. Given a 3D array of shape (4, 5, 6), use slicing to extract a sub-array of shape (2, 3, 4) and then use broadcasting to add a 1D array of shape (4,) to each row along the third axis.
6. Create a 2D array and use both slicing and broadcasting to set the last column to the sum of the first two columns for each row.

For serial 16 to last:

Indexing and Slicing

7. Given a 2D array of shape (6, 6), extract a 2x2 sub-array starting from the element at position (1, 1).
8. From a 3D array of shape (3, 2, 1), extract all elements in the first two rows and all columns of the second slice along the third axis.
9. Given an array of integers, use fancy indexing to extract elements at positions [1, 3, 4, 6].
10. Given a 2D array, use fancy indexing to select rows [0, 2, 2] and columns [1, 3].
11. From a 1D array of random integers, extract all elements that are greater than 8.
12. Given a 2D array of shape (6, 6), replace all elements greater than 13 with the value 0.

Broadcasting

5. Add a 1D array of shape (3,) to each row of a 2D array of shape (4, 3).
6. Multiply a 2D array of shape (3, 3) by a 1D array of shape (3,).
7. Create two 2D arrays of shapes (3, 1) and (1, 4) respectively, and perform element-wise addition.

8. Given a 3D array of shape (2, 3, 4), add a 2D array of shape (3, 4) to each 2D slice along the first axis.

Some more

7. Given a 2D array, use slicing to extract every second row and every second column, then add a 1D array to each row of the sliced array.
8. From a 3D array of shape (3, 2, 1), extract a sub-array using slicing and then use broadcasting to subtract a 2D array from each slice along the third axis.
9. Given a 2D array, extract the diagonal elements and create a 1D array.
10. Use slicing to reverse the order of elements in each row of a 2D array.
11. Given a 3D array of shape (7, 6, 5), use slicing to extract a sub-array of shape (2, 3, 4) and then use broadcasting to add a 1D array of shape (4,) to each row along the third axis.
12. Create a 2D array and use both slicing and broadcasting to set the last column to the sum of the first two columns for each row.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Deadline: 27/6/2024

🔴 **Task 12: Introduction to Pandas (Series, DataFrame basics)**

Now that you have a basic understanding of Python, NumPy, and statistics, it's time to explore the powerful tools for data manipulation offered by the Pandas library. Start by learning about Series and DataFrames and their operations.

1. Create a Pandas Series from a Python list, numpy array, and a dictionary.
2. Assign a custom index to the Series.
3. Perform basic arithmetic operations on Series.
4. Access elements using index labels and positions.
5. Filter the Series to include only values greater than a specific threshold.
6. Create a DataFrame from a dictionary of lists.
7. Create a DataFrame from a numpy array, specifying column and index names.
8. Load a DataFrame from a CSV file.
9. Display the first and last five rows of the DataFrame.
10. Get a summary of the DataFrame including the mean, median, and standard deviation of numeric columns.
11. Extract a specific column as a Series.
12. Filter rows based on column values.
13. Select rows based on multiple conditions.
14. Add a new column to the DataFrame.
15. Delete a column from the DataFrame.
16. Rename columns in the DataFrame.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Deadline: 28/6/2024

🔴 **Task 13: Data manipulation with Pandas (indexing, selection, grouping)**

1. Load a DataFrame from a CSV file. Display the first and last five rows of the DataFrame.
2. Set a specific column as the index of the DataFrame.
3. Select a specific column and display its values.
4. Select multiple columns and display the resulting DataFrame.
5. Select a subset of rows using the `.loc` method.
6. Select a subset of rows and columns using the `.iloc` method.
7. Filter rows based on a condition.
8. Group the DataFrame by a specific column and calculate the mean of each group.
9. Group the DataFrame by multiple columns and calculate the sum of each group.
10. Use the `agg` method to apply multiple aggregation functions to grouped data.
11. Calculate the size of each group.
12. Select rows based on multiple conditions.
13. Use the `query` method to filter rows.
14. Use `isin` to filter rows based on a list of values.
15. Select specific columns and rename them.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Deadline: 30/6/2024

🔴 **Task 14: Data cleaning and preprocessing with Pandas**

Develop a comprehensive understanding of data cleaning and preprocessing techniques using Pandas, which are crucial for preparing data for analysis and machine learning.

1. Identify missing values in the DataFrame.
2. Drop rows with any missing values.
3. Drop columns with any missing values.
4. Fill missing values with a specific value.
5. Fill missing values using forward fill and backward fill methods.
6. Interpolate missing values.
7. Convert a column to a different data type.
8. Apply a function to transform the values of a column.
9. Normalize a column using Min-Max scaling.
10. Standardize a column (z-score normalization).
11. Identify duplicate rows in the DataFrame.

12. Drop duplicate rows.
13. Drop duplicate rows based on specific columns.
14. Convert all string values in a column to lowercase.
15. Remove leading and trailing spaces from string values in a column.
16. Replace a specific substring in a column with another substring.
17. Extract a substring from each value in a column.
18. Convert a column to datetime format.
19. Extract year, month, and day from a datetime column.
20. Filter rows based on a date range.
21. Convert a categorical column to numerical using one-hot encoding.
22. Convert a categorical column to numerical using label encoding.
23. Group values in a categorical column and create a new column with grouped categories.
24. Merge two DataFrames based on a common column.
25. Concatenate two DataFrames vertically.
26. Concatenate two DataFrames horizontally.
27. Create a new column based on existing columns.
28. Discretize a continuous column into bins.
29. Create polynomial features from existing numerical columns.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Week 04

Deadline: 2/6/2024

🔴 **Task 15: Data Wrangling: Join, Combine, and Reshape.**

Now that you know about NumPy and Pandas, it is time to learn data wrangling: join, combine, and reshape.

1. Merge two DataFrames on a single key.
2. Merge two DataFrames on multiple keys.
3. Perform an outer join, inner join, left join, and right join.
4. Concatenate two DataFrames along rows.
5. Concatenate two DataFrames along columns.
6. Concatenate a list of DataFrames.
7. Reshape data using the `melt` function to go from wide to long format.
8. Create a pivot table to summarize data.
9. Group data by one or more columns and perform aggregation functions (e.g., sum, mean, count).
10. Apply multiple aggregation functions to grouped data.
11. Use the `groupby` function to group data and apply custom functions.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Deadline: 3/6/2024

🔴 **Task 16: Introduction to Matplotlib (basic plots: line, scatter, bar)**

It is time to visually observe and find insights, so we will be learning Matplotlib. It is particularly useful for generating graphs and charts to visualize data. Matplotlib is highly customizable, offering a variety of plot types including line plots, scatter plots, bar charts, histograms, and more.

1. **Basic Plots:**
- Create a simple line plot.
- Plot multiple lines on the same graph.
- Add markers to a line plot.
- Customize line styles and colors.
2. **Scatter Plots:**
- Create a basic scatter plot.
- Customize marker styles, sizes, and colors.
- Plot multiple scatter plots on the same graph.
- Add labels and titles to the scatter plot.
3. **Bar Charts:**
- Create a vertical bar chart.
- Create a horizontal bar chart.
- Customize bar colors and add edge colors.
4. **Pie Charts:**
- Create a basic pie chart.
- Customize slice colors and add a legend.
- Explode a slice of the pie chart.
- Add percentage labels to slices.

Resource: https://matplotlib.org/stable/plot_types/index.html

Deadline: 4/6/2024

🔴 **Task 17: Advanced plotting techniques (subplots, histograms, box plots)**

1. **Subplots:**
- Generate a figure with 2x2 subplots, each containing a different type of plot (e.g., line plot, scatter plot, bar plot, and histogram).
- Customize the subplots with titles, axis labels, and legends.
2. **Histograms:**
- Create a histogram for a given numerical data set. Customize the number of bins.
- Plot multiple histograms on the same figure to compare different data sets.
- Normalize the histograms and add appropriate labels and titles.

3. **Box Plots:**

- Create a box plot for a given numerical data set.
- Generate multiple box plots in a single figure to compare different data sets.
- Customize the box plots with titles, labels, and change the colors.

From the previous task and this task, you know some of the basic and necessary plots. Now it's time to practice more. I recommend you draw as many useful plots as you can so that you can truly understand the power of visualization.

Resources: https://matplotlib.org/stable/plot_types/index.html

Month 01 (Mini-project) Data Analysis

It is time for a project that covers all the concepts you have learned so far: **Data Analysis**. I have provided you with a **dataset of university enrollment**. The university has started offering online courses to reach a wider range of students and wants your help to understand enrollment trends. You are asked to use the concepts you have learned, such as s**tatistics, data cleaning, transformations (if needed), and visualizations**, to find important insights from the dataset. Specifically, **identify what contributes to higher enrollment and determine if the course type (online or classroom) is a significant factor.** This is an **unguided project**, so you will need to apply your understanding independently as no guided questions will be provided.
Dataset:
https://drive.google.com/file/d/1rfNe-FDgGJwXNwH4Ia04s61gIzbpIjxk/view?usp=sharing

Month 02 (Towards Baseline Models)

Week 01

Deadline: 8/7/2024

🔴 **Task 18 ->Introduction to Seaborn (visualization with Seaborn)**

Read about these plots from the documentation and use them on different datasets.

1. Scatter Plot
2. Line Plot
3. Bar Plot
4. Count Plot
5. Box Plot

Customization

1. Setting the Style use: **sns.set_style()**
2. Adding Titles and Labels
3. Changing Palette

Resource: https://seaborn.pydata.org/

Deadline: 10/7/2024

🔴 **Task 19 ->Statistical plotting with Seaborn (regression plots, distribution plots)**

1. Linear Regression Plot (`sns.regplot`)
2. Multiple Linear Regression Plot (`sns.lmplot`)
3. Residual Plot (`sns.residplot`)

1. Histogram (`sns.histplot`)
2. Kernel Density Estimate (KDE) Plot (`sns.kdeplot`)
3. Distribution Plot (`sns.distplot`)
4. Box Plot (`sns.boxplot`)
5. Violin Plot (`sns.violinplot`)
6. Pair Plot (`sns.pairplot`)

1. Heatmap (`sns.heatmap`)
2. Joint Plot (`sns.jointplot`)
3. Facet Grid (`sns.FacetGrid`)

Resources: https://seaborn.pydata.org/
Deadline: 11/7/2024
🔴 **Task 20->Feature Engineering**
Note: Please read the blog post in the link provided as a resource.

Feature engineering is the process of using domain knowledge to select, modify, or create new features (variables) that make machine learning algorithms work better. It is one of the most critical steps in building effective machine-learning models because it directly influences the model's ability to learn and make accurate predictions.

Key steps in feature engineering include:

1. **Feature Selection**: Identifying the most relevant features that contribute to the model's accuracy.
2. **Feature Transformation**: Applying mathematical transformations to features to improve the model's performance.
3. **Feature Creation**: Creating new features from existing data to provide more information to the model.

Tasks to Understand Feature Engineering

Here are some tasks that will help you get hands-on experience with feature engineering:

Task 1: Feature Selection

1. **Objective**: Identify the most relevant features from a dataset.
2. **Dataset**: Use a sample dataset like the Titanic or Iris datasets.
3. **Steps**:
   - Load the dataset.
   - Analyze the correlation between features and the target variable.
   - Use methods like mutual information or chi-squared test to select important features.
   - Document the selected features and explain why they were chosen.

Task 2: Feature Transformation

1. **Objective**: Apply transformations to existing features to improve model performance.
2. **Dataset**: Use the same dataset as Task 1.
3. **Steps**:
    - Load the dataset.
    - Apply transformations such as log transformation, normalization, or standardization.

Task 3: Feature Creation

1. **Objective**: Create new features from existing data.
2. **Dataset**: Use the same dataset as Task 1.
3. **Steps**:
    - Load the dataset.
    - Create new features by combining or modifying existing ones (e.g., creating an interaction term, extracting date features, etc.).

Task 4: Polynomial Features

1. **Objective**: Generate polynomial features to capture non-linear relationships.
2. **Dataset**: Use a dataset with numerical features, such as the Boston housing dataset.
3. **Steps**:
    - Load the dataset.
    - Use the `PolynomialFeatures` class from `sklearn.preprocessing` to generate polynomial features.

Task 5: Handling Categorical Features

1. **Objective**: Convert categorical features into numerical features.
2. **Dataset**: Use a dataset with categorical features, such as the Titanic dataset.
3. **Steps**:
    - Load the dataset.
    - Apply encoding techniques like one-hot encoding, label encoding, or target encoding.

Resource:
https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423

Deadline: 13/7/2024

🔴 **Task 21-> Linear algebra and calculus in NumPy**

Linear Algebra Tasks

1. **Matrix Creation and Manipulation**
    - Create various types of matrices (zero matrix, identity matrix, random matrix).
    - Perform basic matrix operations (addition, subtraction, multiplication).
    - Transpose a matrix and find the determinant and inverse of a matrix.
2. **Solving Linear Equations**

    - Use NumPy to solve a system of linear equations.
    - Implement matrix factorization methods (LU decomposition, QR decomposition).
3. **Eigenvalues and Eigenvectors**
    - Calculate the eigenvalues and eigenvectors of a given matrix.
    - Verify the results by reconstructing the original matrix.
4. **Vector Operations**
    - Perform basic vector operations (addition, dot product, cross product).
    - Normalize a vector and compute vector norms.
5. **Matrix Decomposition**
    - Understand and implement Principal Component Analysis (PCA) using SVD.

Calculus Tasks

1. **Numerical Differentiation**
    - Use NumPy to compute the numerical derivative of a given function.
    - Implement forward, backward, and central difference methods for differentiation.
2. **Numerical Integration**
    - Use NumPy to compute the numerical integral of a given function.
    - Implement the trapezoidal rule and Simpson's rule for integration.
3. **Partial Derivatives**
    - Calculate partial derivatives of multivariable functions using NumPy.
    - Verify results by comparing with analytical solutions.
4. **Optimization**
    - Use NumPy to solve optimization problems with constraints.

Resource:
https://drive.google.com/file/d/1j9BePhvRWPMnBPnxHFP6vlRnWQAr-yZy/view?usp=sharing

Deadline: 14/7/2024

🔴 Self Learning-> Introduction to Machine Learning

Read this blog completely. We will have a short interview evaluation based on it. The date for interviews will be announced soon.

Resource:
https://monkeylearn.com/machine-learning/#:~:text=Machine%20learning%20(ML)%20is%20a,to%20make%20their%20own%20predictions.

Week 02

Deadline: 18/7/2024

🔴 Task 22-> Linear Regression from scratch

Creating Linear Regression from scratch involves initializing parameters (weights and bias), using a hypothesis function ($\hat{y} = w \cdot x + b$), and optimizing these parameters to minimize a loss function, typically Mean Squared Error (MSE). This is achieved using gradient descent, which

iteratively updates the weights and bias based on the gradient of the loss function to each parameter. This process helps in understanding the underlying mechanics of Linear Regression and how predictions are refined through optimization.

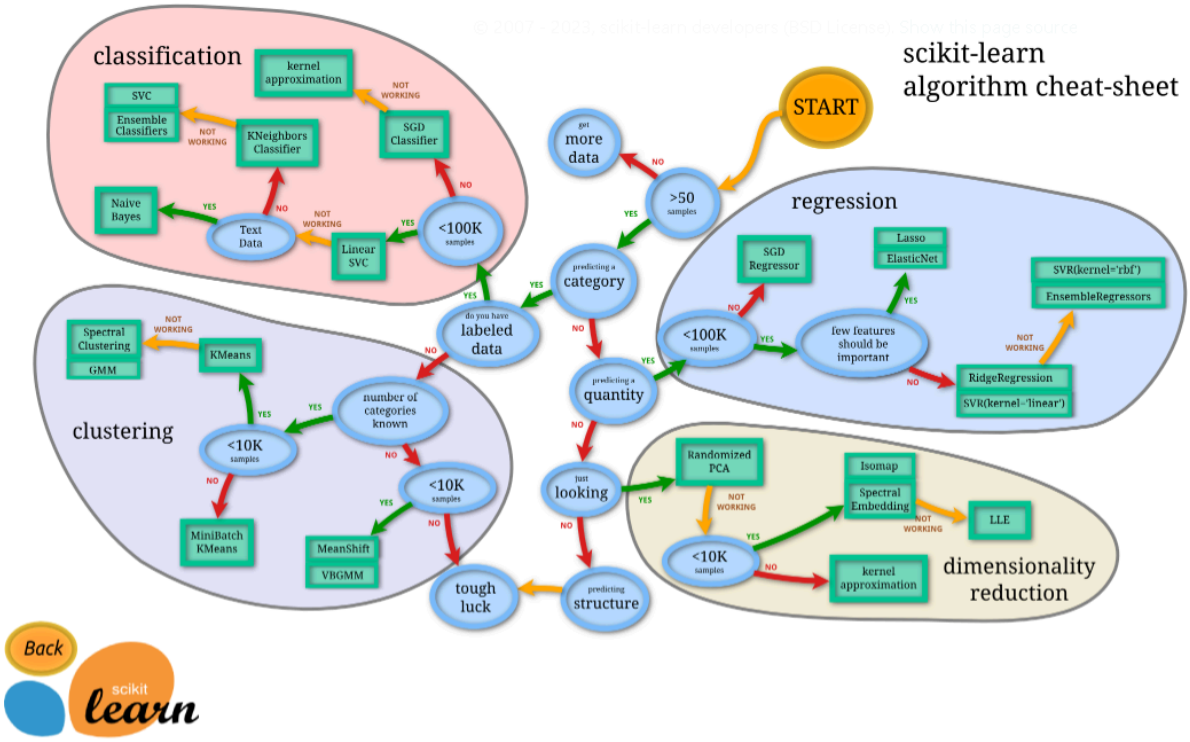
Resource:

https://towardsdatascience.com/linear-regression-from-scratch-cd0dee067f72

Deadline: 19/7/2024

🔴 Task 23-> Regression Models using scikit-learn

Explore different regression models and apply them using the sklearn library.



Resource:

https://scikit-learn.org/stable/

Deadline: 20/7/2024

🔴 Task 24-> Evaluation Techniques for Regression Models

Evaluation techniques for regression models are crucial for assessing their performance. Common metrics include Mean Absolute Error (MAE), which measures the average magnitude of errors, and Mean Squared Error (MSE), which emphasizes larger errors due to squaring the differences. Root Mean Squared Error (RMSE) provides error estimates in the same unit as the target variable. R-squared (R^2) indicates the proportion of variance explained by the model, while Adjusted R-squared accounts for the number of predictors. Mean Absolute Percentage Error (MAPE) offers a percentage-based accuracy metric, and Median Absolute Error provides a robust measure of central tendency for errors, less influenced by outliers. These metrics help in comparing and refining regression models.

Resource:

https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

Week 03

Deadline: 24/7/2024

🔴 Task 25-> Logistic Regression from scratch

Logistic Regression from scratch involves building a classification model by implementing the logistic function and optimization algorithm manually. You start with defining the sigmoid function to model probabilities, then use a cost function (like cross-entropy) to measure prediction error. Optimization, typically using gradient descent, adjusts model weights to minimize the cost function and improve accuracy. This process helps you understand the core mechanics of logistic regression beyond using built-in libraries.

Resource:

https://philippmuens.com/logistic-regression-from-scratch

Deadline: 25/7/2024

🔴 Task 26-> Classification Algorithms using scikit-learn

Classification algorithms in scikit-learn are used to categorize data into predefined classes. Some common ones include:

1. **Logistic Regression**: Models the probability of a binary outcome using the logistic function.
2. **k-Nearest Neighbors (k-NN)**: Classifies data based on the majority class of its nearest neighbors.
3. **Support Vector Machines (SVM)**: Finds the optimal hyperplane that separates classes with the maximum margin.
4. **Decision Trees**: Splits data into subsets based on feature values to make decisions.
5. **Random Forests**: An ensemble of decision trees that improves classification accuracy by averaging multiple trees.
6. **Gradient Boosting**: Builds models sequentially, each correcting errors of the previous one to improve performance.
7. **Naive Bayes**: Applies Bayes' theorem with an assumption of feature independence to classify data.

Resource:

https://scikit-learn.org/stable/

Deadline: 26/7/2024

🔴 Task 27-> Evaluation Techniques for classification models

Evaluation techniques for classification models help assess their performance and ensure they make accurate predictions. Key techniques include:

1. **Confusion Matrix**: A table showing true positives, true negatives, false positives, and false negatives, providing a detailed breakdown of classification results.
2. **Accuracy**: The ratio of correctly predicted instances to the total number of instances. It gives a general sense of how often the model is correct.
3. **Precision**: The ratio of true positives to the sum of true positives and false positives. It measures the accuracy of positive predictions.
4. **Recall (Sensitivity)**: The ratio of true positives to the sum of true positives and false negatives. It measures the ability to identify all positive instances.
5. **F1 Score**: The harmonic mean of precision and recall. It balances precision and recall, especially useful when dealing with imbalanced datasets.
6. **ROC Curve**: A graphical representation of the true positive rate versus the false positive rate at various thresholds. It helps evaluate the trade-offs between true positives and false positives.
7. **AUC (Area Under the Curve)**: Measures the overall performance of the ROC curve. An AUC of 1 indicates a perfect model, while an AUC of 0.5 indicates a model with no discrimination ability.
8. **Cross-Validation**: Splits the dataset into multiple folds to train and test the model on different subsets, providing a more robust estimate of model performance.
9. **Classification Report**: A summary of precision, recall, F1 score, and support for each class, offering a comprehensive view of the model's performance across different classes.

Resources:
https://openclassrooms.com/en/courses/6401081-improve-the-performance-of-a-machine-learning-model/6519011-evaluate-the-performance-of-a-classification-model

Deadline: 27/7/2024

🔴 **Task 28-> Exploring Cross-Validation, Overfitting, and Underfitting**

Cross-Validation

Cross-validation is a technique used to assess the performance and generalization ability of a model by partitioning the dataset into multiple subsets or folds.

Overfitting

Overfitting occurs when a model learns the details and noise in the training data to the extent that it negatively impacts its performance on new, unseen data. Characteristics include:

Underfitting

Underfitting occurs when a model is too simple to capture the underlying patterns in the data. Characteristics include:

Resources:

https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

https://www.geeksforgeeks.org/cross-validation-machine-learning/

Deadline: 28/7/2024

🔴 Task 29-> Hyperparameter Tuning Techniques

When working on a machine learning project, selecting and fine-tuning the right model is essential for optimal performance. Common models from `sklearn` include Linear Regression for predicting continuous values, Logistic Regression for binary classification, and Decision Trees, Random Forests, SVMs, and KNNs for various classification and regression tasks. Hyperparameter tuning techniques such as Grid Search, Random Search, and Bayesian Optimization can be employed to find the best parameters for these models, enhancing their performance and predictive accuracy. Apply these techniques and note down the results to evaluate the impact of different models and hyperparameters on your dataset's performance.

Resource:
https://www.kdnuggets.com/hyperparameter-tuning-gridsearchcv-and-randomizedsearchcv-explained

Week 04

Deadline: 30/7/2024

🔴 Task 30-> Some preprocessing Using scikit-learn

Preprocessing is a crucial step in preparing your data for machine learning models. Using `scikit-learn`, you can scale features with `StandardScaler`, encode categorical variables with `OneHotEncoder`, and impute missing values with `SimpleImputer`. For example, numerical features can be standardized to have zero mean and unit variance, while categorical features can be converted into binary variables. Once preprocessing is complete, you can apply various machine learning models from `sklearn` like Linear Regression, Logistic Regression, and Random Forest. To optimize these models, perform hyperparameter tuning using techniques such as Grid Search and Random Search. Apply these models to your datasets and note down the results based on the techniques you applied to evaluate their performance.

Resource:

https://medium.com/@chyun55555/data-preprocessing-methods-with-scikit-learn-python-98437e8d93cb

Deadline: 31/7/2024

🔴 Task 31-> Dimensionality Reduction Techniques

Dimensionality reduction techniques simplify complex datasets, reduce computational costs, and help prevent overfitting in machine learning models. Common methods include Principal Component Analysis (PCA), which transforms data into principal components with the greatest variance; Linear Discriminant Analysis (LDA), which finds linear combinations of features that best separate classes; and t-Distributed Stochastic Neighbor Embedding (t-SNE), a non-linear technique for visualizing high-dimensional data. Other techniques include Independent Component Analysis (ICA) for separating a multivariate signal into independent components, and feature selection methods like SelectKBest and Recursive Feature Elimination (RFE). To see the impact of these techniques, pick a high-dimensional dataset and observe how the results of a model change when dimensionality reduction is applied. This will help you understand the effectiveness of each technique in improving model performance.

Resources:

https://medium.com/nerd-for-tech/dimensionality-reduction-techniques-pca-lca-and-svd-f2a56b097f7c

Deadline: 2/8/2024

🔴 **Task 32-> Clustering (KMeans)**

Clustering with KMeans partitions a dataset into groups based on feature similarities. To use KMeans, choose a dataset and preprocess it, then apply the KMeans algorithm to group data points into clusters. The Elbow Method helps determine the optimal number of clusters by plotting the within-cluster sum of squares against the number of clusters and finding the 'elbow point.' **Once the optimal number is chosen, use KMeans to segment the data. Visualize the clusters with scatter plots, using different colors to represent each cluster for clear and intuitive understanding.**

Resources: https://www.geeksforgeeks.org/k-means-clustering-introduction/

Month 03 (ADVANCE)
Week 01

Deadline: 5/8/2024

🔴 **Task 33-> Neural Networks Basics (Perceptron, Activation Functions)**

Neural networks, inspired by the human brain, consist of interconnected layers of neurons. A perceptron, the simplest neural network, uses a weighted sum of inputs and an activation function to produce an output. Activation functions, like sigmoid and ReLU, introduce non-linearity, allowing the network to learn complex patterns. **Make a simple neural network from scratch for a regression task, the Mean Squared Error (MSE) measures prediction accuracy, while gradient descent optimizes the model's weights to minimize this error. A basic neural network with one input layer, one hidden layer, and one output layer can effectively perform regression by using these principles.**

Resources: https://www.geeksforgeeks.org/activation-functions-neural-networks/

🔴 **Self-Learning -> Loss functions & Exploring Optimizers**

Loss Functions

Loss functions are essential in training machine learning models as they measure the discrepancy between the model's predictions and the actual values. The goal is to minimize this loss to improve model accuracy. Here are common types of loss functions:

1. **Mean Squared Error (MSE)**: Used for regression tasks, it penalizes larger errors more heavily.
2. **Mean Absolute Error (MAE)**: Also for regression, it measures the average of absolute errors and is less sensitive to outliers.
3. **Cross-Entropy Loss**: Common in classification tasks, it measures the difference between predicted and actual probability distributions.
4. **Hinge Loss**: Used in binary classification, particularly with Support Vector Machines (SVM), to ensure a margin between classes.

Optimizers

Optimizers adjust the model's weights to minimize the loss function. Here are some widely used optimizers:

1. **Stochastic Gradient Descent (SGD)**: Simple and straightforward, it updates weights based on a small batch of data.
2. **Momentum**: Builds on SGD by accelerating convergence using a velocity term to navigate the loss landscape more efficiently.
3. **Adam (Adaptive Moment Estimation)**: Combines the benefits of Momentum and RMSProp, adapting learning rates for each parameter.
4. **RMSProp**: Adapts learning rates for each parameter by dividing the learning rate by a moving average of recent gradients.

Submission is Not Compulsory

This means that submitting the task is optional. You can choose to complete and submit it if you wish to receive feedback or further your understanding, but there is no obligation, and there will be no negative consequences if you decide not to submit it.

Resources:
https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Deadline: 7/8/2024

🔴 **Task 34-> Deep Learning with TensorFlow/Keras**

Learn how to create an ANN using TensorFlow and complete task 33 using TensorFlow. You are free to use more hidden layers than mentioned.

Resources: https://www.geeksforgeeks.org/artificial-neural-network-in-tensorflow/

Deadline: 8/8/2024

🔴 **Task 35-> Convolutional Neural Networks (CNNs)**

Convolutional Neural Networks (CNNs) are a type of deep learning model particularly well-suited for image and video recognition tasks. They work by using convolutional layers to automatically and adaptively learn spatial hierarchies of features from input images. Basic tasks that can be done from scratch with CNNs include image classification, where the model learns to categorize images into predefined classes, and object detection, where the model identifies and locates objects within an image. **Other fundamental tasks include image segmentation, which involves partitioning an image into regions of interest, and image denoising, where the model learns to remove noise from images to enhance their quality.**

Resources: https://www.ibm.com/topics/convolutional-neural-networks

Deadline: 9/8/2024

🔴 **Task 36-> Implement with TensorFlow/Keras**

Learn how to create a CNN using TensorFlow and complete an image classification task.

Resources:
https://www.kaggle.com/code/prashant111/comprehensive-guide-to-cnn-with-keras

Month 02 (Mini-project) Models

Deadline: 14/8/2024

Here is mini project 2. This time, pick a tough dataset related to a regression problem, clean it, perform EDA, apply transformations, conduct statistical tests (hypothesis, etc.), and prepare the data for ML models.

Models:

1. Baseline regressors (SVM, linear regression, Random Forest Regression, decision tree, gradient boosting, etc.)
2. An ANN, and display the error curve to identify if there is any overfitting or underfitting.
3. A model with some convolutional layers at the start (1D convolutional layer) followed by ANN layers; display the error curve and identify if there is any overfitting or underfitting.

Please note: try to provide as much information as possible from each plot or finding from any test performed.

🔴 Self-Learning -> Data Augmentation Techniques for Image Data

Please have a look at Data Augmentation. For tabular data, we primarily use SMOTE or Bootstrapping, etc. What can be used for image augmentation?

Resources:

https://machinelearningmastery.com/image-augmentation-deep-learning-keras/

🔴 Self-Learning -> Introduction to Natural Language Processing (NLP)

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and human language. It involves developing algorithms and models that enable machines to understand, interpret, and generate human language in a way that is both meaningful and useful. NLP combines linguistics, computer science, and machine learning to process and analyze large amounts of natural language data. Applications of NLP include tasks such as sentiment analysis, language translation, chatbots, and speech recognition, making it a critical technology for enabling more intuitive and natural interactions between humans and machines.

Resources:

https://www.ibm.com/topics/natural-language-processing#:~:text=Natural%20language%20processing%2C%20or%20NLP,and%20generate%20text%20and%20speech.&text=often%20in%20real%20time.

Deadline: 20/8/2024

🔴Task 37-> NLP Preprocessing

Go through some common text preprocessing techniques and demonstrate them by applying them to different datasets.

Resources:

https://medium.com/@maleeshadesilva21/preprocessing-steps-for-natural-language-processing-nlp-a-beginners-guide-d6d9bf7689c9

Deadline: 21/8/2024

🔴Task 38-> Advanced NLP Techniques (NER, Sentiment Analysis)

Try to work on some of the common problems in NLP. For Named Entity Recognition (NER), you are allowed to use the NLTK library.

Resources:

https://www.geeksforgeeks.org/named-entity-recognition/
https://www.geeksforgeeks.org/nlp-sentiment-analysis-for-us-election/?ref=ml_lbp

🔴 Self-Learning -> Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a special type of neural network used for processing sequences of data, like text or time series. They work differently from regular neural networks because they have loops that allow them to remember information from previous steps in the sequence. This means that an RNN can use past information to make predictions about the future.

Imagine reading a sentence: an RNN can remember the words you've read so far to understand the context and make better predictions about the next word. However, traditional RNNs sometimes struggle to remember long sequences, which can make them less effective for tasks where long-term context is important.

To address this, more advanced versions of RNNs, like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), have been developed. These versions are designed to remember long-term dependencies better and handle sequences more effectively.

Resources:

https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

Deadline: 22/8/2024

🔴Task 39-> Implement with TensorFlow/Keras (RNN)

A simple application using RNNs is **predicting the next word in a sentence**.

Example:

1. **Collect Data**: Use a small set of text, like sentences from a book or articles.
2. **Prepare Data**: Convert sentences into sequences of words or characters.
3. **Build Model**: Create an RNN that learns to predict the next word based on the previous ones.
4. **Generate Text**: Input a few words, and the model predicts the next word, continuing to generate a sequence.

This basic task helps understand how RNNs handle sequential data and make predictions based on learned patterns.

Resources:
https://machinelearningmastery.com/understanding-simple-recurrent-neural-networks-in-keras/

Capstone Project

Deadline: 22/8/2024

1. Decide whether you would like to work individually, with a partner, or as part of a group with me.
2. Without needing further details at this stage, simply choose whether you prefer to create a model from scratch or develop an application using ChatGPT. Don't worry about the "how" for now, just focus on the "what"—the concept you'd like to work on.
3. Make your experience meaningful by coming up with a unique idea that will make your fellowship productive and stand out.