AIR QUALITY DEVELOPMENT PART 1

AI AND ADS

- **1. Loading the Dataset:**
- To start, you need to acquire the dataset that is relevant to your AI or ADS project. This dataset can come from various sources, such as CSV files, databases, or APIs.
- **2. Preprocessing the Dataset:**
- **Data cleaning:** Remove any inconsistencies, missing values, or outliers from the dataset.
- **Data transformation:** Convert data types, normalize, or scale features as necessary.
- **Feature engineering:** Create new features or modify existing ones to improve the dataset's quality

- **3. Performing Different Analyses:**
- Exploratory Data Analysis (EDA): Visualize the data to gain insights and understand its characteristics.
- Statistical analysis: Use statistical methods to uncover patterns, correlations, and trends in the data.
- Machine learning: Depending on your project's goals, you might apply various machine learning algorithms to train models and make predictions.
- **4. Creating a Document:**
- After performing these activities, it's essential to document your work. You can use tools like Jupyter Notebook, Microsoft Word, or LaTeX for creating a report.
- Include details about the dataset, preprocessing steps, analysis results, and any models or algorithms you've implemented.

DATASET FOR AI AND ADS

- Loading a dataset for AI and ADS projects can vary depending on the data source and format. Here's a general outline of how to load a dataset from common sources.

1. CSV File:

If your dataset is in a CSV (Comma-Separated Values) file, you can use Python libraries like `pandas` to load it.

```
python
import pandas as pd
data = pd.read_csv('your_dataset.csv')
```

2. Excel File:

For Excel files, you can use the `pandas` library as well:

```
python
import pandas as pd
data = pd.read_excel('your_dataset.xlsx')
```

3. JSON Data:

To load data from a JSON file, you can use the `json` module in Python.

```
python
import json
with open('your_dataset.json') as json_file:
data = json.load(json_file)
```

4. Database:

If your dataset is stored in a database, you can use database connectors such as `pyodbc`, `sqlite3`, or libraries like `SQLAlchemy` to query and retrieve the data.

```
python
import sqlite3
conn = sqlite3.connect('your_database.db')
query = "SELECT * FROM your_table"
data = pd.read_sql_query(query, conn).
```

5. APIs:

To fetch data from web APIs, you can use Python libraries like `requests` to make HTTP requests and retrieve data in JSON or other formats.

```
python
import requests
response = requests.get('https://api.example.com/data')
data = response.json()
```

6. Custom Data Sources:

For custom data sources, you may need to write custom code to retrieve and load the data.

Remember to replace 'your_dataset' and 'your_table' with the actual file or table names in your case. Loading the dataset is the first step, and once you have the data in a suitable data structure, you can proceed with preprocessing and analysis.

DATA PREPROCESSING TECHNIQUES

1. Handling Missing Data:

Identify and handle missing values. You can remove rows with missing data or impute missing values with appropriate techniques like mean, median, or interpolation.

2. Data Cleaning:

Check for inconsistencies, errors, or outliers in your data and decide how to deal with them. You may need to remove or correct erroneous data points.

3. Data Transformation:

Convert data types if needed. For example, change categorical variables to numerical format using one-hot encoding or label encoding.

4. Scaling and Normalization:

- Scale numerical features to ensure they have similar scales. Common techniques include min-max scaling or standardization (z-score scaling).

5. Feature Selection:

- Select relevant features for your analysis. Remove redundant or irrelevant features that do not contribute to your project's goals.

6. Feature Engineering:

- Create new features based on domain knowledge or insights gained from your data. Feature engineering can improve model performance.

7.Handling Categorical Data:

Convert categorical data into numerical format using techniques like one-hot encoding or label encoding, depending on the nature of the data.

8. Data Splitting:

If your project involves supervised learning, split your dataset into training and testing sets to assess model performance.

9.Dealing with Imbalanced Data:

If your dataset has imbalanced classes, apply techniques like oversampling, undersampling, or using synthetic data to balance the dataset.

10. Handling Text Data (Natural Language Processing):

- If your dataset contains text data, preprocess it by tokenizing, removing stopwords, and applying techniques like TF-IDF or word embeddings.

11. Time-Series Data Handling:

- For time-series data, perform time-based feature engineering, smoothing, or resampling, depending on your project's goals.

12. Data Visualization:

- Visualize your data to gain insights and identify patterns or trends that can guide preprocessing decisions.