**Climate Data Analysis and Prediction Using Machine Learning**

**About dataset**: This project aims to analyze climate data for Jashore (2019-2021) using machine learning techniques. It focuses on predicting maximum temperature (Temp\_MAX), identifying climate patterns through clustering, and evaluating the impact of multiple factors using regression models. The insights gained will aid in understanding climate variability in the region.

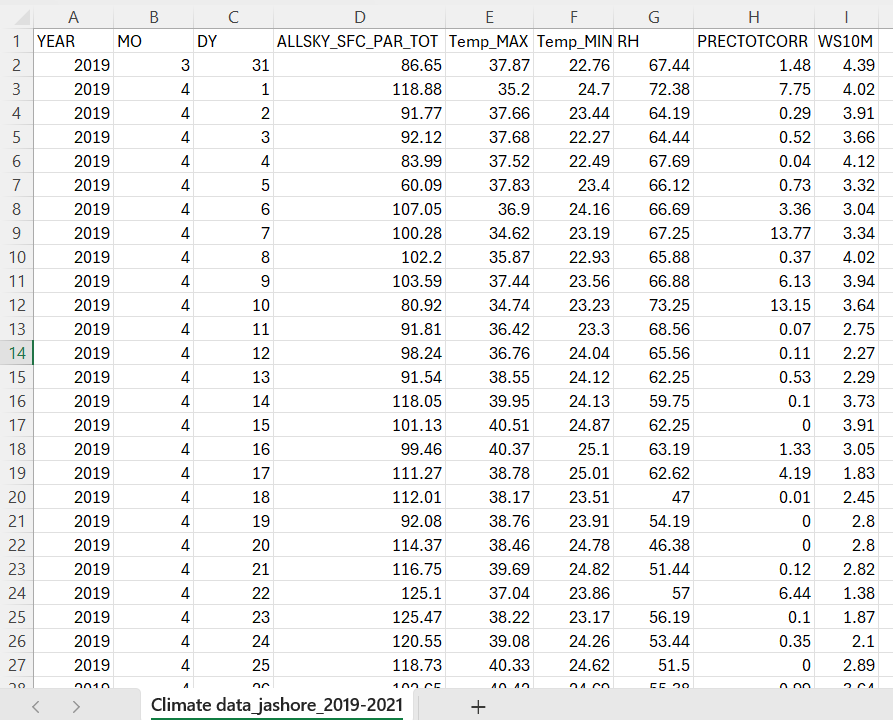


Figure 1 Climatic Dataset

**Linear Regression** was chosen for this dataset because of its simplicity and effectiveness in modeling the relationship between continuous variables. Given the dataset's focus on predicting Temp\_MAX (maximum temperature) using features such as Temp\_MIN (minimum temperature) and RH (relative humidity), Linear Regression is well-suited to determine and quantify the linear relationships between these variables. Additionally, it provides interpretable results, which are crucial for understanding how changes in one factor impact the target variable.

**Objectives**

* Predict Temp\_MAX using historical data.
* Classify climate patterns using clustering.
* Evaluate the influence of temperature, humidity, and seasonal changes through regression models.

**3. Data Description**

The dataset contains climate records from 2019 to 2021 for Jashore, with the following columns:

* **YEAR**: Year of observation
* **MO**: Month
* **Temp\_MAX**: Maximum temperature
* **Temp\_MIN**: Minimum temperature
* **RH**: Relative humidity

**CODE:**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
import matplotlib.pyplot as plt  
  
*# Load the dataset*file\_path = r"E:\Machine learning\Ml class\_cse\Climate data\_jashore\_2019-2021.csv"  
climate\_data = pd.read\_csv(file\_path)  
  
*# Selecting relevant columns and ensuring no missing values*data = climate\_data[['Temp\_MAX', 'Temp\_MIN', 'RH']].dropna()  
  
*# Splitting the data into features and target*X = data[['Temp\_MIN', 'RH']]  
y = data['Temp\_MAX']  
  
*# Splitting the data into training and testing sets*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
*# Creating the Linear Regression model*linear\_model = LinearRegression()  
  
*# Fitting the model*linear\_model.fit(X\_train, y\_train)  
  
*# Predicting on the test set*y\_pred = linear\_model.predict(X\_test)  
  
*# Calculating performance metrics*mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
  
*# Printing the metrics*print(f"Linear Regression MSE: {mse:.4f}")  
print(f"Linear Regression R2: {r2:.4f}")  
  
*# Visualizing the results: Predicted vs. Actual*plt.figure(figsize=(8, 6))  
plt.scatter(y\_test, y\_pred, color='blue', alpha=0.7, edgecolor='k', label='Predicted vs. Actual')  
plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', linewidth=2, label='Perfect Fit')  
plt.title('Linear Regression: Actual vs. Predicted Temp\_MAX')  
plt.xlabel('Actual Temp\_MAX')  
plt.ylabel('Predicted Temp\_MAX')  
plt.legend()  
plt.grid(True)  
plt.show()  
from sklearn.metrics import mean\_absolute\_error  
  
*# Calculating additional metrics*mae = mean\_absolute\_error(y\_test, y\_pred)  
rmse = mean\_squared\_error(y\_test, y\_pred, squared=False) *# squared=False gives RMSE  
  
# Printing all metrics*print(f"Linear Regression Metrics:")  
print(f"Mean Absolute Error (MAE): {mae:.4f}")  
print(f"Mean Squared Error (MSE): {mse:.4f}")  
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")  
print(f"R² Score: {r2:.4f}")

**Methodology:**

**Data Preprocessing**

* Handled missing data and ensured feature consistency.
* Split the dataset into training and testing sets (80:20 ratio).

**Linear Regression**

* Used Temp\_MIN and RH to predict Temp\_MAX.
* Evaluated the model using Mean Squared Error (MSE) and R² Score.

**Clustering**

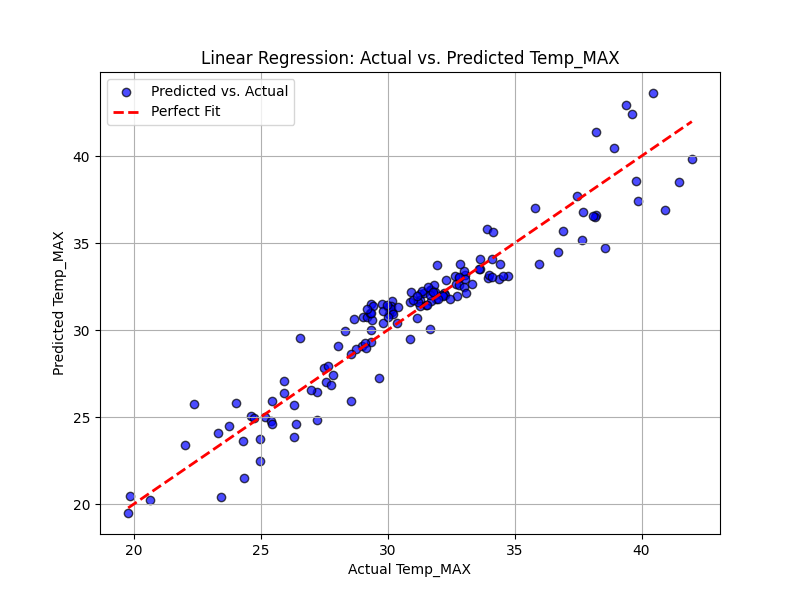
* Determined optimal clusters using the elbow method.
* Applied KMeans clustering with 3 clusters based on Temp\_MAX, Temp\_MIN, and RH.

**Results and Discussion**

Linear Regression Results

Linear Regression MSE: 2.0124

Linear Regression R2: 0.8983

****

**Figure 2 linear Regression**

**Clustering**

* Determined optimal clusters using the elbow method.
* Applied KMeans clustering with 3 clusters based on Temp\_MAX, Temp\_MIN, and RH.

CODE:

import pandas as pd  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
import matplotlib.pyplot as plt  
  
*# Load the dataset*file\_path = r"E:\Machine learning\Ml class\_cse\Climate data\_jashore\_2019-2021.csv"  
climate\_data = pd.read\_csv(file\_path)  
  
*# Selecting features for clustering and ensuring no missing values*clustering\_features = climate\_data[['Temp\_MAX', 'Temp\_MIN', 'RH']].dropna()  
  
*# Determining the optimal number of clusters using the elbow method*inertia = []  
for k in range(1, 11):  
 kmeans = KMeans(n\_clusters=k, random\_state=42)  
 kmeans.fit(clustering\_features)  
 inertia.append(kmeans.inertia\_)  
  
*# Plotting the elbow curve*plt.figure(figsize=(8, 5))  
plt.plot(range(1, 11), inertia, marker='o', linestyle='--')  
plt.title('Elbow Method for Optimal Clusters')  
plt.xlabel('Number of Clusters')  
plt.ylabel('Inertia')  
plt.grid(True)  
plt.show()  
  
*# Applying KMeans with an optimal number of clusters (e.g., 3)*optimal\_clusters = 3  
kmeans\_model = KMeans(n\_clusters=optimal\_clusters, random\_state=42)  
cluster\_labels = kmeans\_model.fit\_predict(clustering\_features)  
  
*# Adding cluster labels to the data*climate\_data['Cluster'] = cluster\_labels  
  
*# Displaying the first few rows with cluster labels*print(climate\_data[['Temp\_MAX', 'Temp\_MIN', 'RH', 'Cluster']].head())  
  
*# Calculating the silhouette score as the accuracy metric*silhouette\_avg = silhouette\_score(clustering\_features, cluster\_labels)  
print(f"Silhouette Score for {optimal\_clusters} clusters: {silhouette\_avg:.4f}")  
  
*# Visualizing the clusters*plt.figure(figsize=(10, 6))  
for cluster in range(optimal\_clusters):  
 cluster\_data = clustering\_features[cluster\_labels == cluster]  
 plt.scatter(cluster\_data['Temp\_MIN'], cluster\_data['Temp\_MAX'], label=f'Cluster {cluster}', alpha=0.7)  
  
*# Marking centroids*centroids = kmeans\_model.cluster\_centers\_  
plt.scatter(centroids[:, 1], centroids[:, 0], c='red', marker='X', s=200, label='Centroids')  
  
plt.title('KMeans Clustering: Temp\_MIN vs Temp\_MAX')  
plt.xlabel('Temp\_MIN')  
plt.ylabel('Temp\_MAX')  
plt.legend()  
plt.grid(True)  
plt.show()

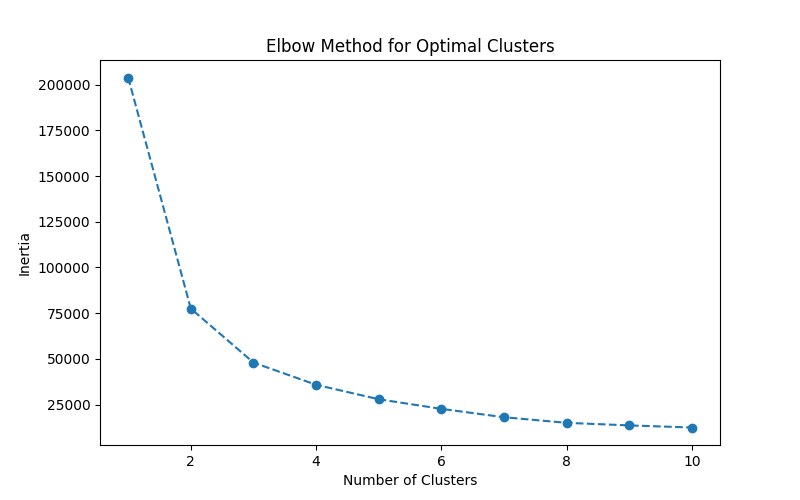


Figure 3 Elbow method

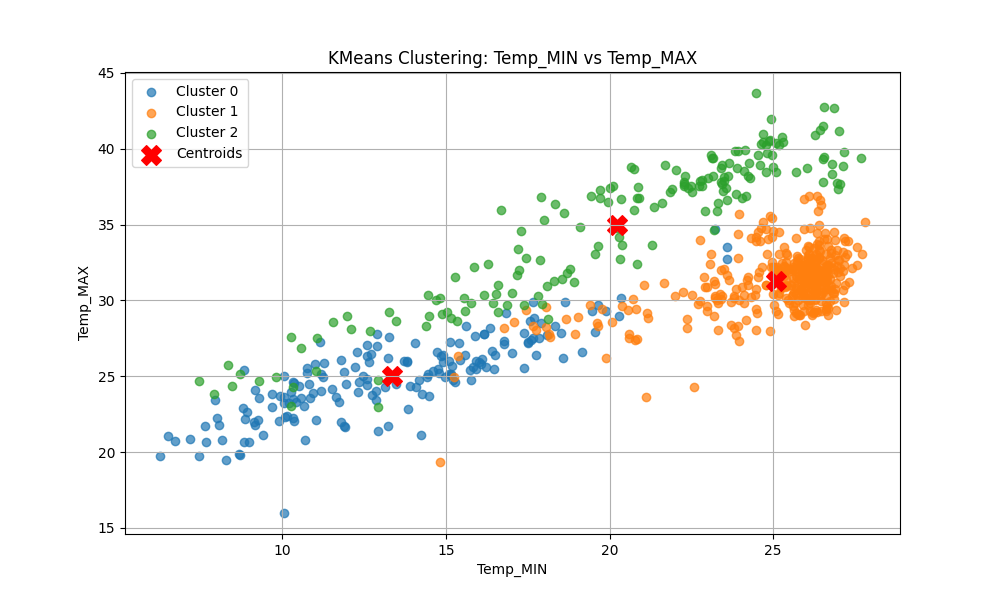


Figure 4 Clustering

**Clustering Results**

* Optimal number of clusters: 3
* Clusters represented distinct climate patterns based on temperature and humidity.

**Conclusion**

The project demonstrated the effectiveness of machine learning in analyzing climate data. Linear regression and clustering highlighted temperature trends and patterns, while multiple regression provided deeper insights into seasonal variations. Future work could explore advanced models and larger datasets for enhanced accuracy.