DAYANANDA SAGAR UNIVERSITY

School of Engineering

Department of Computer Science & Engineering(AIML)

Kudlu Gate, Bangalore – 560068

Karnataka, India

Special Topic 1

Report

on

"Earth Observation using ML to Analyze Satellite Data to Monitor and Predict Weather Patterns, Natural Disasters and Climate Change"

Department of Computer Science Engineering (AI & ML)

SUBMITTED BY

Sayli Pankaj Bande (ENG21AM0112)

Ratan Ravichandran (ENG21AM0093)

Sri Bharath Sharma P (ENG22AM3005)

Shruti Nigam (ENG21AM0120)

Under the guidance of

Dr. Rangaraj B S

Research Professor, Dept. of CSE (AIML)

2022 - 2023

DAYANANDA SAGAR UNIVERSITY

School of Engineering

Department of Computer Science & Engineering (AIML)

Kudlu Gate, Bangalore – 560068

Karnataka, India

Department of Computer Science Engineering (AI & ML)

CERTIFICATE

This is to certify that the Special Topic-1(21CS2409) project work titled "Earth Observation using ML to Analyze Satellite Data to Monitor and Predict Weather Patterns, Natural Disasters and Climate Change" is carried out by Sayli Pankaj Bande (ENG21AM0112), Ratan Ravichandran (ENG21AM0093), Sri Bharath Sharma P (ENG22AM3005**)**, Shruti Nigam (ENG21AM0120) bonafide students of Bachelor of Technology in Computer Science and Engineering (AI&ML) at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering(AI&ML), during the year 2022-2023.

Signature of Guide

Dr.Rangraj B S, Ph.D
Research Professor

Dept. of AIML, SOE, DSU

Signature of Chairperson

Dr. Jayavrinda Vrindavanam, Ph.D
Professor and Chairperson,

Dept. of AIML, SOE, DSU

DECLARATION

We, Sayli Pankaj Bande (ENG21AM0112), Ratan Ravichandran (ENG21AM0093), Sri Bharath Sharma P (ENG22AM3005**)**, Shruti Nigam (ENG21AM0120)**,** are student's of the fourth semester B.Tech in **Computer Science and Engineering(AI&ML)**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Special Topic-1 titled **"**Earth Observation using ML to Analyze Satellite Data to Monitor and Predict Weather Patterns, Natural Disasters and Climate Change**"** has been carried out by us and submitted in partial fulfillment for the award of degree in **Bachelor of Technology in Computer Science and Engineering(AI&ML)** during the academic year **2022-2023**.

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr. Udaya Kumar Reddy K R.**, Dean, **School of Engineering & Technology**, **Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Jayavrinda Vrindavanam, Department Chairperson**, **Computer Science and Engineering (AI&ML)**, **Dayananda Sagar University,** for providing the right academic guidance that made our task possible.

We would like to thank our guide **Dr.Rangraj B S**, **Research Professor**, **Dept. of Computer Science and Engineering(AI&ML)**, **Dayananda Sagar University**, for sparing his valuable time to extend help in every step of our Special Topic-1 work, which paved the way for smooth progress and the fruitful culmination of the research.

We would like to thank our **Special Topic-1 Coordinator Jeevaraj R, Assistant Professor** and all the staff members of Computer Science and Engineering (AI&ML) for their support.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Special Topic-1 work.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Earth observation using machine learning has emerged as a promising approach to analysing satellite data for monitoring and predicting weather patterns, natural disasters, and climate change. Hence, the accurate forecasting of weather is integral to planning and implementing change efforts. This study indicates the significance of weather forecasting and also addresses the challenge of analysing satellite data to identify patterns, trends, and predictions related to weather and climate.

Our project utilizes machine learning techniques to analyse satellite data for earth observation. We plan to use various algorithms such as linear regression, naive Bayesian classifiers, ARIMA and others to preprocess, train, and test the model.

CHAPTER 1 INTRODUCTION

Earth observation satellites generate vast amounts of data that can be analyzed using machine learning algorithms to monitor and predict weather patterns, natural disasters, and climate change. Machine learning models can detect patterns in the data and provide early warning signals of potential climate-related events, enabling timely and effective responses. The integration of machine learning into the analysis of satellite data has the potential to revolutionize our understanding of the Earth's climate and support decision-making processes.

To predict weather, we analyze historical data based on these factors:

Year: Long-term climate variations influence weather patterns.

Month: Seasonal variations impact sunlight, temperature, and precipitation.

Day: Certain days may exhibit recurring patterns like diurnal temperature changes or specific weather events.

Temperature: It affects air pressure, wind patterns, cloud formation, and precipitation.

Specific Humidity: Moisture content in the air influences cloud formation and precipitation.

Relative Humidity: It measures air's moisture compared to its maximum capacity, affecting comfort, cloud formation, and energy exchange between Earth's surface and the atmosphere.

Objectives:

The objective of this project is to leverage machine learning techniques to analyze satellite data for the purpose of monitoring and predicting weather patterns, natural disasters, and climate change. The project aims to tackle this challenge through the development and implementation of a comprehensive machine learning framework. This framework uses Earth observation data analysis to accomplish several key objectives:

Enhancing our Understanding of Weather Phenomena: By using machine learning, we can analyze weather patterns more deeply, improving our forecasting models and accuracy.

Improving Disaster Preparedness and Response with Early Warnings: By combining Earth observation data and machine learning, the framework offers timely and accurate early warnings for weather-related disasters. This allows emergency teams and communities to prepare, mobilize resources, and implement effective strategies to minimize damage to lives and infrastructure.

Understanding the Impact of Climate Change: Climate change affects our planet, altering weather patterns and intensifying extreme events. This project acknowledges the significance of understanding these changes. By analyzing Earth observation data using machine learning, we can assess the impacts of climate change on weather systems and develop strategies to mitigate and adapt to these transformations.

Formulating Effective Adaptation Strategies and Resource Allocation: The project helps formulate robust adaptation strategies by understanding weather and climate change. It determines optimal resource allocation for sectors vulnerable to weather fluctuations. By providing valuable insights, decision-makers can minimize risks and optimize resource allocation for better outcomes.

The machine learning project on weather forecasting is limited by resource availability and technical expertise. It focuses on utilizing existing satellite data and expertise to develop and enhance forecasting models. Acknowledging these limitations, the project aims to maximize its potential within the constraints of available data and expertise to improve weather predictions to the best extent possible.

CHAPTER 2 PROBLEM DEFINITION

In recent years, the unpredictability of climate change and the occurrence of natural disasters have reached alarming levels, presenting a substantial and ever-growing threat to human life and property. However, the current methods employed to monitor and predict these events often fall short in terms of accuracy and timeliness. This inadequacy poses significant challenges for individuals and organizations who rely on reliable and up-to-date information to make informed decisions.

Recognizing the critical need to address this issue, our project endeavors to bridge the gap by developing innovative approaches rooted in machine learning and advanced data analysis techniques. By leveraging the power of these technologies, we aim to enhance the monitoring and prediction capabilities pertaining to climate change and natural disasters.

The primary objective of our project is to create a comprehensive framework that combines satellite data, historical records, and real-time observations to generate accurate and timely insights. Through this framework, people will gain access to a wealth of information that can empower them to make well-informed decisions. Whether it involves implementing evacuation plans, strengthening infrastructure, or taking other preventive measures, having reliable and precise data is crucial to mitigating the adverse impact of these events on human lives and property.

By pushing the boundaries of existing methods and striving for greater accuracy and timeliness, our project seeks to revolutionize the way we monitor, predict, and respond to climate change and natural disasters. Through our efforts, we aim to equip individuals and organizations with the necessary tools and knowledge to navigate the unpredictable nature of these events and safeguard the well-being of communities around the world.

CHAPTER 3 LITERATURE REVIEW

Aravind M ,Thilak et al.[1] proposes that random forest models excel in accuracy and computational efficiency compared to SVMs and neural networks. They exhibit superior predictive accuracy, capturing complex relationships and non-linear patterns. Aggregating predictions from multiple decision trees reduces overfitting, leading to more robust and accurate results. Random forests efficiently process large datasets, making them suitable for high-dimensional or large-scale applications. Moreover, they handle missing values and outliers well, minimizing the need for extensive data preprocessing.

C. M. KISHTAWAL [2] proposes that Satellite data assimilation is a critical component in weather prediction. While numerical data enables faster predictions, it does not guarantee accuracy. By integrating satellite observations into numerical models, the assimilation process improves the accuracy of predictions. Satellite data provides valuable information about the Earth's atmosphere, oceans, and land surface, which cannot be obtained through numerical models alone. By incorporating this data, forecasters can enhance the accuracy of weather predictions and make more informed decisions.

Ferreira, B.; Silva et al. [3] proposes that the combination of Earth observation (EO) and machine learning (ML) is highly advantageous. EO provides extensive and continuous data on a global scale, while ML techniques excel in analyzing big data and identifying complex patterns. By integrating the two, we can extract valuable insights, predict environmental phenomena, classify land cover, optimize data collection processes, and uncover new knowledge. This synergy enhances our understanding of the Earth's systems and empowers decision-making in fields such as agriculture, disaster management, and urban planning.

CHAPTER 4 PROJECT DESCRIPTION

This project leverages machine learning techniques to analyze satellite data for earth observation, focusing on monitoring and predicting weather patterns, natural disasters, and climate change. Accurate and timely forecasting is crucial due to the increasing unpredictability of climate change and the rise in the frequency and intensity of natural disasters.

The primary objective is to develop a comprehensive framework that utilizes satellite data analysis and machine learning algorithms. Three models - linear regression, random forest, and decision tree - will be employed to predict weather conditions based on historical and real-time data.

Data from Bangalore spanning 2009 to the present will be used, enabling diverse training and testing of the models on various weather patterns and events. The aim is to create accurate and reliable weather prediction models that improve preparedness and decision-making for residents and stakeholders.

By analyzing satellite data, the project addresses the challenge of weather prediction accuracy by identifying patterns, trends, and predictions related to weather and climate. Machine learning algorithms enable the models to learn from historical data and make predictions based on available input features.

The project's outcomes are significant for weather-dependent sectors such as agriculture, transportation, and disaster management. Accurate weather predictions assist farmers in decision-making for crop cultivation and irrigation scheduling. Transportation and logistics companies can optimize routes and schedules based on forecasts, improving efficiency and safety. Reliable weather prediction models also aid proactive disaster management and response planning, minimizing loss of life and property damage.

4.1 Software and Hardware Requirements

Hardware/software	Elements	Version
Hardware	Processor	Intel i5 8th gen/ AMD Ryzen 5 3000 series
	RAM	4GB
Software	OS	Windows/Linux/MacOS
	Browser	Chrome/Brave/Edge
	Libraries	sklearn, numpy, pandas, matplotlib

Table 1 Software and Hardware requirements

CHAPTER 5 METHODOLOGY

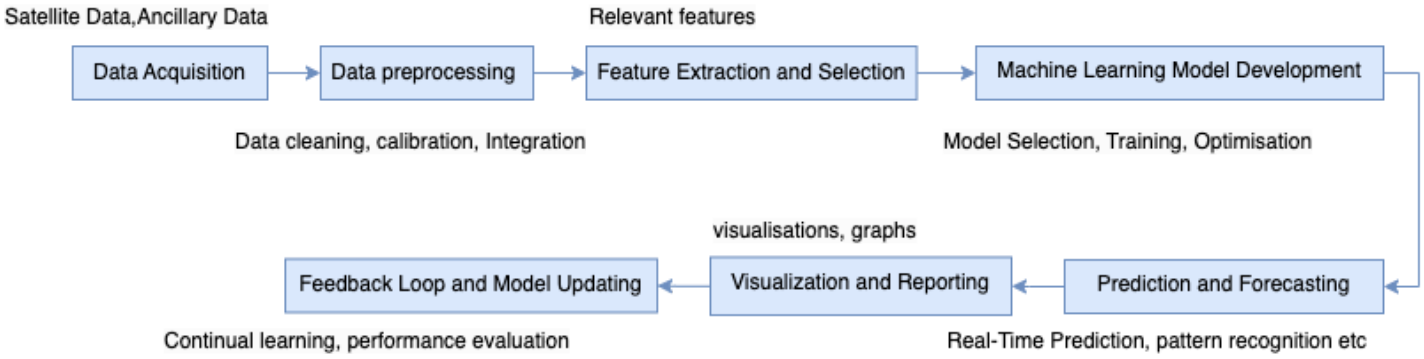

Figure 1 Architecture Diagram

5.1. Architecture Of Proposed System

Data Collection: Satellite data, Historical records- Collect historical weather data from reliable sources, covering a significant timeframe.

Data Preprocessing: Clean and preprocess satellite data to remove noise, artifacts, and inconsistencies.

Feature Extraction: Perform feature engineering to extract relevant features from the data.

Model Training and Development: Split the preprocessed data into training and testing sets. Utilize machine learning algorithms such as linear regression, random forest, and decision tree.Train the models using the training set, optimizing their parameters to minimize prediction errors. Test the models using the validation set to assess their performance and fine-tune them if necessary.

Model Evaluation: Evaluate the trained models using evaluation metrics such as accuracy, root mean square error (RMSE), or mean absolute error (MAE).

5.2. Architecture Workflow

Data Preparation:

1. Import necessary libraries:
    a. pandas: for data manipulation and analysis
    b. numpy: for numerical operations
    c. matplotlib.pyplot: for data visualization
    d. seaborn: for advanced data visualization
    e. sklearn: for data preprocessing and model selection
    f. Read the dataset:
    g. pd.read_csv(): to read the dataset from a CSV file

Data preprocessing:

1. Handle missing values:
2. Feature selection: use a correlation heat map
3. weather_df_num=weather_df.loc[:,['maxtempC','mintempC','HeatIndexC','uvIndex.1','FeelsLikeC','WindChillC']]: select relevant features
4. Split data into features and target:
5. weather_y=weather_df_num.pop("tempC"): assign the target variable to weather_y and remove it from weather_df_num
6. weather_x=weather_df_num: assign the remaining features to weather_x
7. Split data into training and testing sets:
8. train_X,test_X,train_y,test_y=train_test_split(weather_x,weather_y,test_size=0.2,random_state=4): split the data into 80% training and 20% testing sets

Model Training and Evaluation:

Linear Regression:

1. Import the LinearRegression model from sklearn.linear_model
2. Initialize the model: model=LinearRegression()
3. Train the model: model.fit(train_X,train_y)
4. Make predictions: prediction = model.predict(test_X)

5. Evaluate the model:
6. np.mean(np.absolute(prediction-test_y)): calculate the mean absolute error
7. model.score(test_X, test_y): calculate the variance score (R^2)

Decision Tree Regression:

1. Import the DecisionTreeRegressor model from sklearn.tree
2. Initialize the model: regressor=DecisionTreeRegressor(random_state=0)
3. Train the model: regressor.fit(train_X,train_y)
4. Make predictions: prediction2=regressor.predict(test_X)
5. Evaluate the model:
6. np.mean(np.absolute(prediction2-test_y)): calculate the mean absolute error
7. regressor.score(test_X, test_y): calculate the variance score (R^2)

Random Forest Regression:

1. Import the RandomForestRegressor model from sklearn.ensemble
2. Initialize the model: regr=RandomForestRegressor(max_depth=90, random_state=0, n_estimators=100)
3. Train the model: regr.fit(train_X,train_y)
4. Make predictions: prediction3=regr.predict(test_X)
5. Evaluate the model:
6. np.mean(np.absolute(prediction3-test_y)): calculate the mean absolute error
7. regr.score(test_X, test_y): calculate the variance score (R^2)

Model Evaluation and Visualization:

Create a table to compare the actual and predicted values:

Visualize the table: Use plotly.graph_objs to create a table visualization

Create line plots to compare actual and predicted values:

Create traces for actual and predicted values:

Predictions with Trained Models:

Use the trained models to make predictions on new input data:

Linear Regression: model.predict(new_data)

Decision Tree Regression: regressor.predict(new_data)

Random Forest Regression: regr.predict(new_data)

5.3. Modules Used

1. NumPy: A fundamental library for scientific computing, used for data manipulation and numerical operations.
2. Pandas: A powerful library for data manipulation and analysis, used for loading, preprocessing, and transforming data.
3. Scikit-learn: A popular machine learning library, used for training and evaluating regression models.
4. Matplotlib: A plotting library for creating static and interactive visualizations.
5. Seaborn: A statistical data visualization library built on Matplotlib, used for enhancing visualizations and analyzing relationships between variables.

CHAPTER 6 EXPERIMENTATION

6.1. Dataset Definition

The dataset used comprises data collected over the last 10 years, starting from January 1 2019. It contains values of the minimum and maximum temperatures, UV indices, Sunrise and sunset, moonrise and moonset, heat index, dew point, etc.

The dataset comprises of information regarding the following:

Parameter	**Significance**
Date/Time	signifies the date and time at which the data was recorded
MaxTempC	gives the highest recorded temperature at the specific location and time
MinTempC	gives the lowest recorded temperature at the specific location and time
TotalSnow	gives the total snow recorded for the specified date and time
sunHour	gives the amount of sunlight or daylight available during the particular hour,
HeatIndexC	measure of how hot it feels when relative humidity is factored in with the actual air temperature.
uvIndex1	helps assess the risk level associated with UV radiation exposure by providing a standardized value.
Moon_illumination	represents the percentage of the moon's visible surface that is illuminated by the sun at a specific time or hour.
FeelsLikeC	provides an estimate of how the weather conditions will feel to humans in terms of perceived temperature or discomfort.
Moonrise	gives the time when the moon becomes visible above the horizon as observed from a specific location on Earth
Moonset	It represents the time when the moon disappears below the horizon as observed from a specific

	location on Earth.
Sunrise	It represents the specific moment when the upper edge of the sun appears above the eastern horizon, marking the beginning of daylight.
Sunset	It represents the specific moment when the upper edge of the sun disappears below the western horizon, marking the end of daylight.
WindChillC	prediction holds significance as it provides information about the perceived temperature, taking into account the combined effect of wind and air temperature.
DewPointC	It represents the dew point temperature in degrees Celsius. It indicates the level of moisture or humidity in the air.
WindGustC	It represents the wind gust speed in kilometers per hour. It provides information about the strength or intensity of the wind during gusts.
CloudCover	It gives the percentage of the sky covered by clouds. It represents the amount of cloudiness at a given time or location
Humidity	It represents the percentage of moisture in the air relative to the maximum amount of moisture the air can hold at a given temperature
precipMM	It represents the amount of precipitation, such as rainfall or snowfall, recorded in millimeters during a specific time period
Pressure	It represents the atmospheric pressure level at the given date/time
TempC	It represents the air temperature in degrees Celsius. It indicates the actual temperature of the air at a specific time or location

Visibility	it represents the horizontal distance over which objects can be seen clearly in the atmosphere.
WinddirDegree	It gives the angle or direction from which the wind is blowing.
WindspeedKmph	It represents the wind speed in kilometers per hour. It indicates the rate at which the air is moving horizontally.

Table 2 Significance of the dataset

When plotting a correlation heatmap, the columns MaxTempC, HeatIndexC, uvIndex1,FeelsLikeC and WindchillC are found to have the most corelation with target TempC, so they are considered for the model.

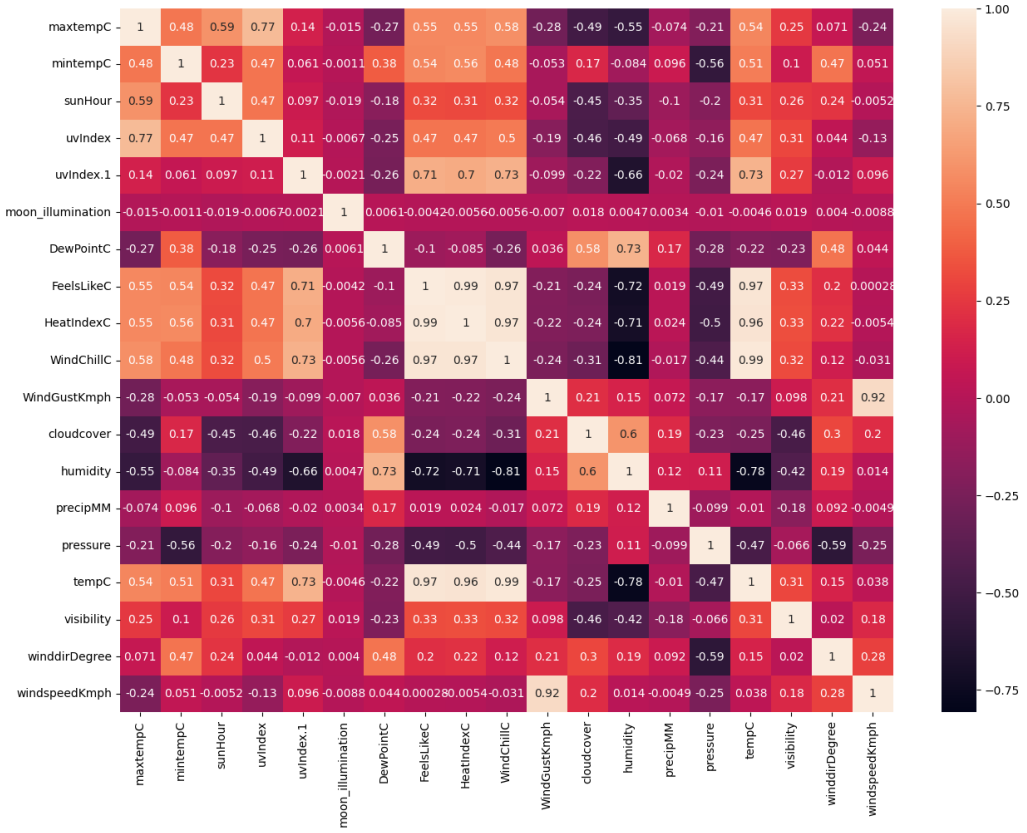

Figure 2 Heatmap for all features

The dataset is as follows:

date_time	maxtempC	mintempC	HeatIndexC	uvIndex.1	FeelsLikeC	WindChillC	tempC
2019-01-01 00:00:00	26	16	18	1	18	18	18
2019-01-01 01:00:00	26	16	18	1	18	18	18
2019-01-01 02:00:00	26	16	17	1	17	17	17
2019-01-01 03:00:00	26	16	17	1	17	17	17
2019-01-01 04:00:00	26	16	17	1	17	17	17

Figure 3: Preview of the dataset

6.2. Model Creation

The model is created by:

- Importing necessary libraries such as numpy, pandas, matplotlib, and sklearn for data analysis, visualization, and machine learning.
- Reading and displaying necessary information from the dataset
- Creating a subset of the dataset, including specific columns ('maxtempC', 'mintempC', 'HeatIndexC', 'uvIndex.1', 'FeelsLikeC', 'WindChillC', 'tempC') and stored in the "weather_df_num" DataFrame.

6.2.1. Linear Regression

1. It works by fitting a linear equation to the given data, it establishes a relationship between the features and the target variable. It works by:
2. The algorithm will fit a linear equation to the training data by adjusting the slope and intercept.
3. Once the model is trained, it will use the learned slope and intercept to predict the temperature for new data points. It will multiply each feature variable by its corresponding slope, sum up these values, and then add the intercept to obtain the predicted temperature. (Y=MX+C)
4. After the model is trained, it can be used to predict the temperature for new instances where the feature variables (maxtempC, mintempC, HeatIndexC, uvIndex.1, FeelsLikeC, WindChillC) are known. By using

the values of these features into the equation, the algorithm will calculate the predicted temperature.

6.2.2. Decision Tree

1. The decision tree algorithm constructs a tree-like model by recursively partitioning the data based on the feature variables. In this case, it would:
2. It selects the best feature and split point that maximizes the information gain or minimizes impurity at each node.
3. Making predictions- Starting from the root node, each instance traverses down the tree by following the selected feature values. At each internal node, the algorithm compares the instance's feature value with the split point and decides which child node to move to. Eventually, the instance reaches a leaf node that contains the predicted temperature value.
4. By comparing the predicted temperatures with the actual temperatures from the test or validation data, the accuracy and generalization capability of the model can be assessed.

6.2.3. Random Forest

1. The random forest algorithm creates an ensemble of decision trees. It randomly selects subsets of the training data and feature subsets to train individual decision trees. Each tree is trained independently using a subset of the data. For this use-case, it would:
2. To make a prediction, the random forest algorithm aggregates the predictions of all the decision trees in the forest. For a new instance, each decision tree in the forest provides its prediction for the temperature based on the selected feature values. The final prediction is typically the average or majority vote of all the individual tree predictions.
3. After the random forest model is built, it can be used to predict the temperature for new instances with known feature values. The random

forest algorithm aggregates the predictions from all the decision trees in the ensemble to provide the final predicted temperature.

6.3. Model Training

The models are trained using the training data, represented by "train_X" (input features) and "train_y" (target variable), using the fit() method. Predictions are made using the trained models on the test data, represented by "test_X" (input features), using the predict() method, and the predictions are stored in separate variables ("prediction", "prediction2", "prediction3" for Linear Regression, Decision Tree Regressor, and Random Forest Regressor, respectively).The accuracy or error of the predictions is evaluated by computing the mean absolute error (MAE), which is calculated as the mean of the absolute differences between the predicted values and the true values ("test_y").

6.3.1. Evaluation

1. Linear Regression

   The performance of the linear regression model is assessed by comparing the predicted values with the actual values. The most effective evaluation metric for linear regression is the mean squared error (MSE), which calculates the average squared difference between the predicted and actual values. The lower the MSE, the better the model fits the data.
2. Decision Tree

   The performance of the decision tree model is evaluated using various metrics, such as accuracy or mean squared error. By comparing the predicted temperatures with the actual temperatures from the test or validation data, the accuracy and generalization capability of the model can be assessed.
3. Random Forest

   Similar to decision trees, the accuracy of the random forest model is calculated by comparing the predicted temperatures with the actual temperatures from the test data, using metrics such as R2 scores or MSE.

CHAPTER 7 RESULTS AND ANALYSIS

The results obtained were as follows:

```
[ ]  model = DecisionTreeRegressor()
     new_value = np.array([[35,23,32.1,1,32.1,32.4]])
     model.fit(train_X,train_y)
     prediction = model.predict(new_value)
     print("Predicted value:", prediction)

     Predicted value: [32.]
```

```
[ ]  print('Variance score: %.2f' % regressor.score(test_X, test_y))
     from sklearn.metrics import r2_score

     print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction2 - test_y)))
     print("Residual sum of squares (MSE): %.2f" % np.mean((prediction2 - test_y) ** 2))
     print("R2-score: %.2f" % r2_score(test_y,prediction2 ) )#decision tree

     Variance score: 0.98
     Mean absolute error: 0.38
     Residual sum of squares (MSE): 0.34
     R2-score: 0.98
```

Figure 4: Results for Decision Tree

```
     model = LinearRegression()
     new_value = np.array([[35,23,32.1,1,32.1,32.4]])
     model.fit(train_X,train_y)
     prediction = model.predict(new_value)
     print("Predicted value:", prediction)

     Predicted value: [31.45130049]
```

```
[ ]  print('Variance score: %.2f' % model.score(test_X, test_y))
     from sklearn.metrics import r2_score
     print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction - test_y)))
     print("Residual sum of squares (MSE): %.2f" % np.mean((prediction - test_y) ** 2))
     print("R2-score: %.2f" % r2_score(test_y,prediction ))# linear regression

     Variance score: 0.97
     Mean absolute error: 0.48
     Residual sum of squares (MSE): 0.57
     R2-score: 0.97
```

Figure 5: Results for Linear Regressions

```
model = RandomForestRegressor()
model.fit(train_X,train_y)
new_value = np.array([[35,23,32.1,1,32.1,32.4]])
prediction = model.predict(new_value)

print("Predicted value:", prediction)
```

```
Predicted value: [31.695]
```

+ Code + Text

```
[ ] print('Variance score: %.2f' % regr.score(test_X, test_y))
    from sklearn.metrics import r2_score

    print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction3 - test_y)))
    print("Residual sum of squares (MSE): %.2f" % np.mean((prediction3 - test_y) ** 2))
    print("R2-score: %.2f" % r2_score(test_y,prediction3 ) )#random forest
```

```
Variance score: 0.98
Mean absolute error: 0.38
Residual sum of squares (MSE): 0.34
R2-score: 0.98
```

Figure 6: Results for Random Forest

Based on the provided results, it appears that both the RandomForestRegressor and DecisionTreeRegressor models perform slightly better than the linear regression model. This can be explained by the following:

- **Variance score**: The variance scores for RandomForestRegressor and DecisionTreeRegressor are 0.98, while the linear regression model has a slightly lower variance score of 0.97. A higher variance score indicates better performance in capturing the target variable's variance.
- **Mean absolute error (MAE):** Both the RandomForestRegressor and DecisionTreeRegressor models have a MAE of 0.38, while the linear regression model has a slightly higher MAE of 0.48. A lower MAE suggests better accuracy in predicting the target variable.
- **Residual sum of squares (MSE)**: The RandomForestRegressor and DecisionTreeRegressor models have an MSE of 0.34, whereas the linear regression model has a higher MSE of 0.57. A lower MSE indicates better accuracy in predicting the target variable.
- **R2-scor**e: The R2-scores for RandomForestRegressor and DecisionTreeRegressor are 0.98, while the linear regression model achieves a slightly lower R2-score of 0.97. A higher R2-score indicates a better fit of the model to the data.

Decision tree was more accurate for the prediction for the past week over random forest. However the difference between both of them are very close and both have performed very well over historic data.

Linear Regression Visualisations

Actual	Prediction	Difference	Percentage Error
26	25.84	0.16	0.62%
21	20.63	0.37	1.76%
27	27.6	-0.6	-2.22%
29	27.83	1.17	4.03%
20	19.77	0.23	1.15%
20	19.65	0.35	1.75%
25	24.48	0.52	2.08%
24	26.06	-2.06	-8.58%
24	24.71	-0.71	-2.96%
29	28.64	0.36	1.24%
23	23.94	-0.94	-4.09%
25	24.39	0.61	2.44%
26	25.78	0.22	0.85%
20	20.45	-0.45	-2.25%
26	25.79	0.21	0.81%
35	35.17	-0.17	-0.49%

Figure 7: Numerical Actual vs Predicted values for Linear Regression

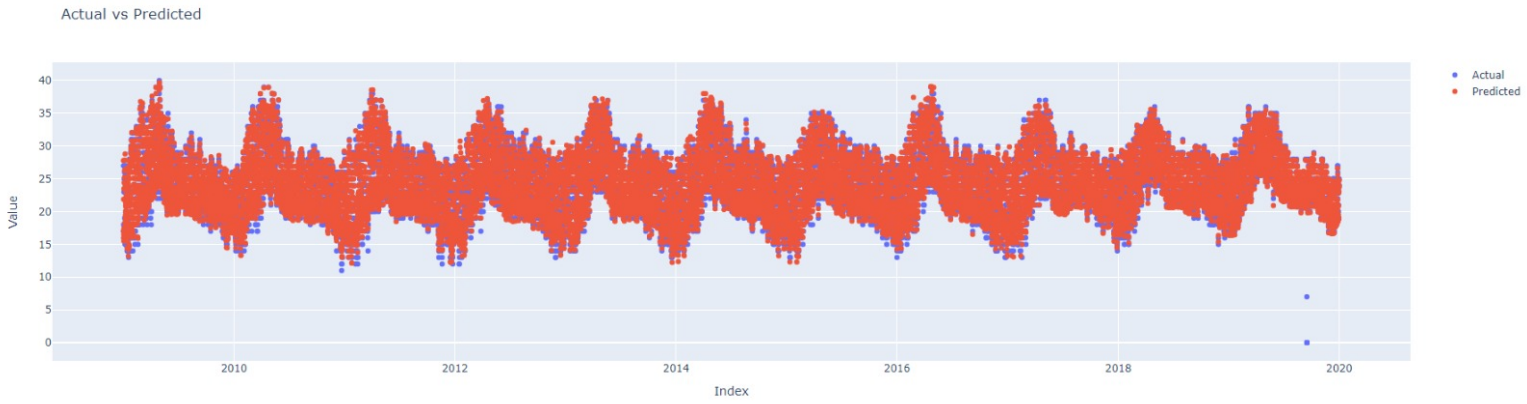

Figure 8: Scatter Plot- Actual vs Predicted Values for Linear Regression

Decision Tree Visualisation

Actual	Prediction	Difference	Percentage Error
26	25.89	0.11	0.42%
21	20.81	0.19	0.9%
27	27.91	-0.91	-3.37%
29	28.22	0.78	2.69%
20	19.98	0.02	0.1%
20	19.61	0.39	1.95%
25	25	0	0.0%
24	26.5	-2.5	-10.42%
24	24.85	-0.85	-3.54%
29	28	1	3.45%
23	23.88	-0.88	-3.83%
25	24.86	0.14	0.56%
26	25.77	0.23	0.88%
20	20.39	-0.39	-1.95%
26	26	0	0.0%
35	34.67	0.33	0.94%

Figure 9: Numerical Actual vs Predicted Values for Decision Trees

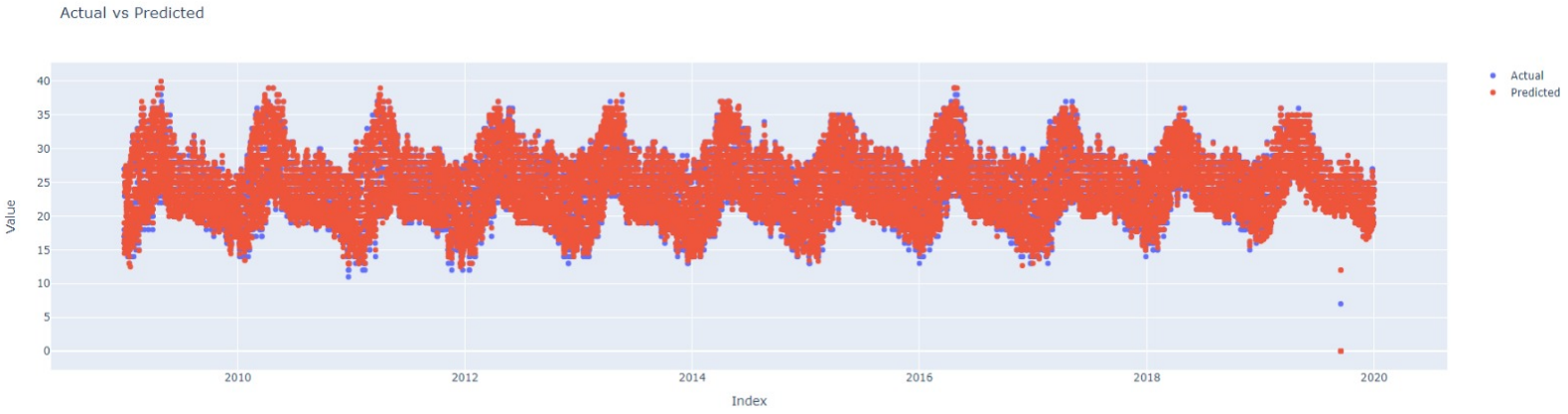

Figure 10: Scatter Plot- Actual vs Predicted Values for Decision Trees

Random Forest Visualizations

Actual	Prediction	Difference	Percentage Error
26	25.89	0.11	0.42%
21	20.81	0.19	0.9%
27	27.91	-0.91	-3.37%
29	28.17	0.83	2.86%
20	19.98	0.02	0.1%
20	19.61	0.39	1.95%
25	25	0	0.0%
24	26.44	-2.44	-10.17%
24	24.85	-0.85	-3.54%
29	28.31	0.69	2.38%
23	23.89	-0.89	-3.87%
25	24.85	0.15	0.6%
26	25.77	0.23	0.88%
20	20.39	-0.39	-1.95%
26	26	0	0.0%
35	34.69	0.31	0.89%

Figure 11: Numerical Actual vs Predicted Values for Random Forest

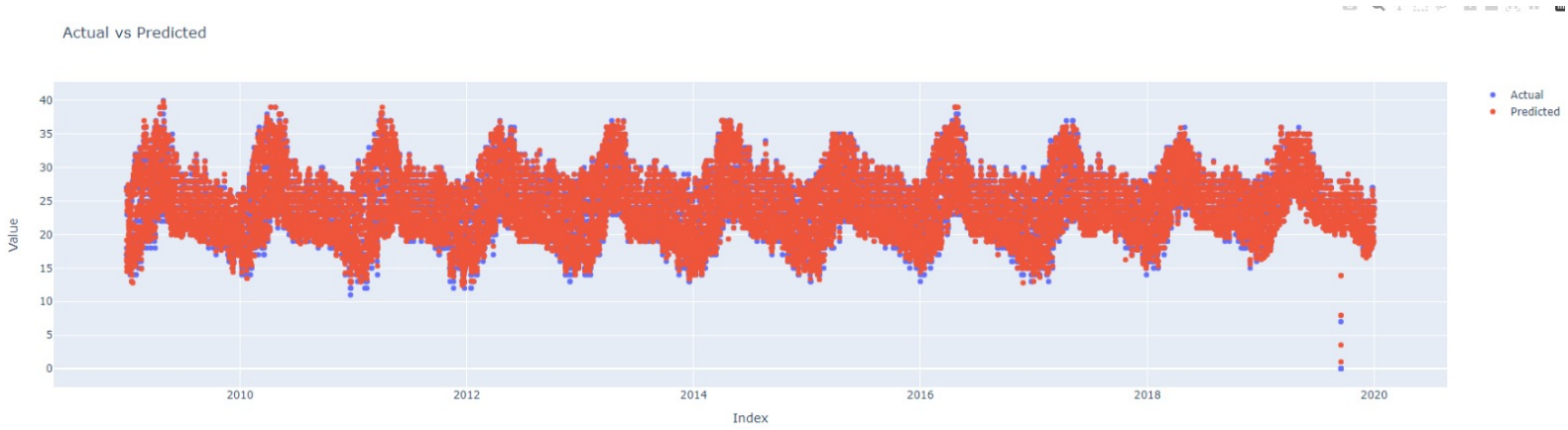

Figure 12: Numerical Actual vs Predicted Values for Random Forest

CHAPTER 8 CONCLUSION AND FUTURE WORK

Weather prediction using satellite data has limitations due to its limited vertical resolution, impacting the accuracy of predictions for variables like cloud formation and precipitation. While satellite imagery offers distinct advantages with its higher spatial resolution, allowing detailed observations of localized weather features, it also has a major drawback. Its effectiveness is highly dependent on weather conditions, such as cloud cover, which can result in data gaps and potentially affect the accuracy of weather predictions in regions experiencing heavy cloud cover or adverse weather conditions.

Combining satellite numerical data and imagery yields the best weather prediction model. Numerical data captures large-scale patterns, while imagery offers higher spatial resolution for localized features. This integration compensates for their respective limitations, resulting in more comprehensive and accurate assessments. The numerical data fills gaps in imagery due to cloud cover, while imagery provides details where numerical data lacks vertical resolution. The combination optimizes prediction accuracy and reliability

Based on the conclusion, our next step in developing our project is as follows:

- To enhance the accuracy of our predictions, we will incorporate satellite image data into our existing numerical data trained model. We will leverage our prediction capabilities to analyze climate change patterns and predict natural disasters.
- Our focus will be on implementing the models into a web application that provides real-time predictions and displays comprehensive analysis.
- We will dedicate efforts to creating more refined models by combining ground-based data, satellite numerical data, and satellite image data. This integrated approach aims to produce the most optimal and accurate version of our weather prediction model.

CHAPTER 9 REFERENCES

[1] Aravind M ,Thilak S ,Vigneshwaran B ,Dr.J.B.Jona, Weather prediction Using Random Forest Methods, IJCRT 2022

[1]C. M. KISHTAWAL,Use of satellite observations for weather prediction, MAUSAM 2019.

[3]Ferreira, B.; Silva, R.G.; Iten, M. Earth Observation Satellite Imagery Information Based Decision Support Using Machine Learning. Remote Sens. 2019

CHAPTER 10 PROGRAM/CODE

```
#importing basic libraries
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#Data spliting tools
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import preprocessing

#importing all the models
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

#reading the dataset & decribing it
weather_df = pd.read_csv('/content/bengaluru.csv', parse_dates=['date_time'],
index_col='date_time')
weather_df.tail(5)
weather_df.columns
weather_df.describe()
weather_df.isnull().any()
```

```
weather_df_num=weather_df.loc[:,['maxtempC','mintempC','HeatIndexC','uvIndex.1','FeelsLikeC','WindChillC','tempC' ]]

weather_df_num.head()

weather_df_num.shape

weather_df_num.columns


#heatmap of the correlation matrix

weather_df = weather_df.drop('totalSnow_cm', axis=1)

import seaborn as sns

weather_df_corr = weather_df.corr()

plt.figure(figsize=(15,11))

sns.heatmap(weather_df_corr, annot = True)

plt.show()

#used for extracting features


weather_y=weather_df_num.pop("tempC")

weather_x=weather_df_num


#spliting the data set for training and testing

train_X,test_X,train_y,test_y=train_test_split(weather_x,weather_y,test_size=0.2,random_state=4)

train_X.shape

train_y.shape

train_y.head()


model=LinearRegression()

model.fit(train_X,train_y)

prediction = model.predict(test_X)
```

```
np.mean(np.absolute(prediction-test_y))
print('Variance score: %.2f' % model.score(test_X, test_y))

import plotly.graph_objs as go

for i in range(len(prediction)):
    prediction[i] = round(prediction[i], 2)

results = pd.DataFrame({'Actual': test_y, 'Prediction': prediction})
results['Difference'] = (results['Actual'] - results['Prediction']).round(4)
results['Percentage Error'] = ((results['Difference'] / results['Actual']) *
100).round(2).astype(str) + '%'

fig = go.Figure(data=[go.Table(
    header=dict(values=list(results.columns)),
    cells=dict(values=[results.Actual, results.Prediction, results['Difference'],
results['Percentage Error']])
)])
fig.show()



import plotly.graph_objs as go

# Round the predicted values
rounded_predictions = [round(pred, 2) for pred in prediction]

# Create a trace for actual values
trace_actual = go.Scatter(
```

```
    x=test_y.index,
    y=test_y,
    mode='markers',
    name='Actual'
)

# Create a trace for predicted values
trace_predicted = go.Scatter(
    x=test_y.index,
    y=rounded_predictions,
    mode='markers',
    name='Predicted'
)

# Create the layout
layout = go.Layout(
    title='Actual vs Predicted',
    xaxis=dict(title='Index'),
    yaxis=dict(title='Value')
)

# Create the data list
data = [trace_actual, trace_predicted]

# Create the figure
fig = go.Figure(data=data, layout=layout)
```

```
# Show the figure
fig.show()

import matplotlib.pyplot as plt

# Plot for predicted values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index,prediction, width=0.4, color='blue')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Predicted Values')
plt.show()

# Plot for actual values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, test_y, width=0.4, color='green')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Actual Values')
plt.show()

# Plot with both actual and predicted values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, test_y, width=0.4, color='green', label='Actual')
plt.bar(test_y.index, prediction, width=0.4, color='#ADD8E6', label='Predicted')
plt.xlabel('Index')
plt.ylabel('Value')
```

```
plt.title('Actual vs Predicted')
plt.legend()
plt.show()

model = LinearRegression()
#['maxtempC','mintempC','HeatIndexC','uvIndex.1','FeelsLikeC','WindChillC']
new_value = np.array([[32.4,22.7,34.2,8,34.2,33.4]])

model.fit(train_X,train_y)
prediction = model.predict(new_value)
print("Predicted value:", prediction)

from sklearn.metrics import r2_score
print("Mean absolute error: %.4f" % np.mean(np.absolute(prediction - test_y)))
print("Residual sum of squares (MSE): %.4f" % np.mean((prediction - test_y) ** 2))
print("R2-score: %.4f" % r2_score(test_y, prediction))

from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor(random_state=0)
regressor.fit(train_X,train_y)
prediction2=regressor.predict(test_X)
np.mean(np.absolute(prediction2-test_y))

import numpy as np
import plotly.graph_objs as go

for i in range(len(prediction2)):
```

```
    prediction2[i] = round(prediction2[i], 2)


results = pd.DataFrame({'Actual': test_y, 'Prediction': prediction2})

results['Difference'] = results['Actual'] - results['Prediction']

results['Difference'] = np.round(results['Difference'], 4)


# Calculate percentage error

results['Percentage Error'] = ((results['Difference'] / results['Actual']) *
100).round(2).astype(str) + '%'


fig = go.Figure(data=[go.Table(

    header=dict(values=list(results.columns)),

    cells=dict(values=[results.Actual, results.Prediction, results['Difference'],
results['Percentage Error']])

)])

fig.show()


import plotly.graph_objs as go


# Round the predicted values

rounded_predictions2 = [round(pred, 2) for pred in prediction2]


# Create a trace for actual values

trace_actual = go.Scatter(

    x=test_y.index,

    y=test_y,

    mode='markers',

    name='Actual'
```

```
)

# Create a trace for predicted values
trace_predicted = go.Scatter(
    x=test_y.index,
    y=rounded_predictions2,
    mode='markers',
    name='Predicted'
)

# Create the layout
layout = go.Layout(
    title='Actual vs Predicted',
    xaxis=dict(title='Index'),
    yaxis=dict(title='Value')
)

# Create the data list
data = [trace_actual, trace_predicted]

# Create the figure
fig = go.Figure(data=data, layout=layout)

# Show the figure
fig.show()

import matplotlib.pyplot as plt
```

```
# Plot for predicted values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, rounded_predictions2, width=0.4, color='blue')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Predicted Values')
plt.show()

# Plot for actual values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, test_y, width=0.4, color='green')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Actual Values')
plt.show()

# Plot with both actual and predicted values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, test_y, width=0.4, color='green', label='Actual')
plt.bar(test_y.index, rounded_predictions2, width=0.4, color='#ADD8E6',
label='Predicted')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```

```
model = DecisionTreeRegressor()
new_value = np.array([[32.4,22.7,34.2,8,34.2,33.4]])
model.fit(train_X,train_y)
prediction = model.predict(new_value)
print("Predicted value:", prediction)

print('Variance score: %.2f' % regressor.score(test_X, test_y))
from sklearn.metrics import r2_score

print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction2 - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction2 - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y,prediction2 ) )#decision tree

from sklearn.ensemble import RandomForestRegressor
regr=RandomForestRegressor(max_depth=90,random_state=0,n_estimators=100)
regr.fit(train_X,train_y)
prediction3=regr.predict(test_X)
#mean absolute error
np.mean(np.absolute(prediction3-test_y))

import numpy as np
import plotly.graph_objs as go

for i in range(len(prediction3)):
    prediction3[i] = round(prediction3[i], 2)

results = pd.DataFrame({'Actual': test_y, 'Prediction': prediction3})
```

```
results['Difference'] = results['Actual'] - results['Prediction']
results['Difference'] = np.round(results['Difference'], 4)

# Calculate percentage error
results['Percentage Error'] = ((results['Difference'] / results['Actual']) *
100).round(2).astype(str) + '%'

fig = go.Figure(data=[go.Table(
    header=dict(values=list(results.columns)),
    cells=dict(values=[results.Actual, results.Prediction, results['Difference'],
results['Percentage Error']])
)])
fig.show()

import plotly.graph_objs as go

# Round the predicted values
rounded_predictions3 = [round(pred, 2) for pred in prediction3]

# Create a trace for actual values
trace_actual = go.Scatter(
    x=test_y.index,
    y=test_y,
    mode='markers',
    name='Actual'
)

# Create a trace for predicted values
```

```
trace_predicted = go.Scatter(
    x=test_y.index,
    y=rounded_predictions3,
    mode='markers',
    name='Predicted'
)

# Create the layout
layout = go.Layout(
    title='Actual vs Predicted',
    xaxis=dict(title='Index'),
    yaxis=dict(title='Value')
)

# Create the data list
data = [trace_actual, trace_predicted]

# Create the figure
fig = go.Figure(data=data, layout=layout)

# Show the figure
fig.show()
import plotly.graph_objs as go

# Round the predicted values
rounded_predictions3 = [round(pred, 2) for pred in prediction3]
```

```
# Create a trace for actual values
trace_actual = go.Scatter(
    x=test_y.index,
    y=test_y,
    mode='markers',
    name='Actual'
)

# Create a trace for predicted values
trace_predicted = go.Scatter(
    x=test_y.index,
    y=rounded_predictions3,
    mode='markers',
    name='Predicted'
)

# Create the layout
layout = go.Layout(
    title='Actual vs Predicted',
    xaxis=dict(title='Index'),
    yaxis=dict(title='Value')
)

# Create the data list
data = [trace_actual, trace_predicted]

# Create the figure
```

```
fig = go.Figure(data=data, layout=layout)

# Show the figure
fig.show()
import matplotlib.pyplot as plt

# Plot for predicted values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, rounded_predictions3, width=0.4, color='blue')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Predicted Values')
plt.show()

# Plot for actual values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, test_y, width=0.4, color='green')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Actual Values')
plt.show()

# Plot with both actual and predicted values
plt.figure(figsize=(8, 4))
plt.bar(test_y.index, test_y, width=0.4, color='green', label='Actual')
plt.bar(test_y.index, rounded_predictions3, width=0.4, color='#ADD8E6',
label='Predicted')
plt.xlabel('Index')
```

```
plt.ylabel('Value')
plt.title('Actual vs Predicted')
plt.legend()
plt.show()

model = RandomForestRegressor()
model.fit(train_X,train_y)
#['maxtempC','mintempC','HeatIndexC','uvIndex.1','FeelsLikeC','WindChillC']

new_value = np.array([[32.4,22.7,34.2,8,34.2,33.4]])
prediction = model.predict(new_value)

print("Predicted value:", prediction)

print('Variance score: %.2f' % regr.score(test_X, test_y))
from sklearn.metrics import r2_score

print("Mean absolute error: %.2f" % np.mean(np.absolute(prediction3 - test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((prediction3 - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y,prediction3 ) )#random forest
```