SPACE WEATHER FORECASTING

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

BY

DANDAMUDI JAYASWAROOP

22B91A0553

Under Supervision of Mr. VIJAY PRADEEP

Henotic IT Solutions Pvt Ltd

(Duration: 5th July 2023 to 5th September 2023)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

S.R.K.R. ENGINEERING COLLEGE

(Autonomous)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P

(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA &NAAC)

(Affiliated to JNTU, KAKINADA)

SAGIRAMAKRISHNAMRAJUENGINEERINGCOLLEGE

(Autonomous)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Summer Internship Report titled "**COUNSELLING AND SEATS ALLOTMENT SYSTEM**" is the bonafide work done by **Mr.Dandamudi Jayaswaroop** bearing **22B91A0553** at the end of secondyear second semester at Henotic IT Solutions Pvt Ltdfrom 5th July 2023 to 5th September 2023inpartialfulfillmentoftherequirementsfortheaward of the degree of Bachelor of Technology Computer Science and Engineering.

Department Internship Coordinator Dean -T & P Cell Headof theDepartment

Abstract:

This project focuses on leveraging machine learning (ML) techniques to advance the accuracy and efficiency of space weather prediction. Space weather phenomena, such as solar flares and geomagnetic storms, can have significant impacts on satellite operations, communication systems, and power grids on Earth. Traditional methods for space weather prediction often face challenges in handling the complexity and dynamic nature of space environments.

Our ML-based approach involves the development of predictive models trained on historical space weather data, satellite observations, and solar activity indices. We aim to explore various ML algorithms, including deep learning architectures, to capture intricate patterns and relationships within the data. Additionally, feature engineering and model interpretability techniques will be employed to enhance the understanding of the underlying physics governing space weather events.

The project's outcomes will contribute to the advancement of space weather forecasting capabilities, enabling more timely and accurate predictions. Ultimately, this research holds the potential to mitigate the impact of space weather on critical infrastructure and improve preparedness for space-related disruptions on Earth.

Machine Learning Project

1.INTRODUCTION:

- Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data.
- The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.

Objective:

- The primary objective of our project is to develop a robust and efficient sentiment analysis system capable of automatically classifying customer reviews into positive, negative, or neutral sentiments. By leveraging state-of-the-art machine learning algorithms, our system aims to provide businesses with actionable intelligence, enabling them to enhance customer satisfaction, identify areas for improvement, and make data-driven decisions.
- This project aims to leverage the power of machine learning (ML) to predict and understand space weather phenomena, allowing for more effective mitigation strategies and improved operational resilience of our technological infrastructure. By developing predictive

models, we can provide timely warnings and insights to mitigate potential disruptions caused by space weather events.

Problem statement

Develop a machine learning model to predict space weather events, such as solar flares and geomagnetic storms, based on historical space weather data and relevant solar parameters.

Here are the following steps:

1.Data set:

In the context of our space weather prediction project, the dataset serves as the foundational source of information upon which our machine learning models are trained, validated, and tested. This dataset encompasses a diverse collection of observations and measurements related to space weather phenomena, aiming to capture the complexity and variability inherent in such environmental conditions.

2. Importing Modules:

In a machine learning project, you can import modules by using the import statement in Python. Typically, you'll use popular libraries like NumPy, Pandas, and scikit-learn.

Importing modules in our script is as follows:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from warnings import filterwarnings
```

```
filterwarnings('ignore')
import seaborn as sns

pd.set_option('display.max_columns',None)
```

- **pandas** and **numpy** are commonly used for data manipulation and numerical operations.
- **matplotlib.pyplot** is used for creating visualizations.
- **warnings** is imported to manage warnings.
- **Pd.set_option** suppresses warnings to enhance the cleanliness of the output. While generally not recommended to ignore warnings, in certain cases, like during exploratory data analysis (EDA), it can make the output more readable
- **Seaborn** is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics
- This line sets a display option in Pandas to show all columns when displaying a DataFrame. **None** means unlimited columns will be displayed.

```
from google.colab import drive
drive.mount('/content/drive')
```

from google.colab import drive: Imports the necessary module from Google Colab to enable Google Drive integration.
drive.mount('/content/drive'): Mounts your Google Drive into the '/content/drive' directory in the Colab environment.
When you run this code cell in a Colab notebook, it will prompt you to authorize Colab to access your Google Drive. Follow the

link provided, grant the necessary permissions, and copy the authorization code back into the cell.

3. Feature selection:

```
weath_pred=pd.read_csv("/content/drive/My Drive/Copy of space_weather_dataset.csv")
weath_pred2=weath_pred.copy()
weath_pred.head(5)
```

weath_pred=pd.read_csv("/content/drive/MyDrive/Copy of space_weather_dataset.csv"):

Reads the space weather dataset from the specified CSV file into a Pandas DataFrame named weath_pred.

weath_pred2=weath_pred.copy():

Creates a copy of the original dataset (weath_pred) named weath_pred2 to preserve the original data.

weath_pred.head(5):

Displays the first 5 rows of the weath_pred DataFrame, providing a glimpse of the dataset.

This code snippet serves as the initial step in a machine learning project, focusing on loading and understanding the dataset before moving on to feature selection and model building.

4.Correlation analysis:

The provided code generates a heatmap of the correlation matrix for the features in the weath_pred dataset using the matplotlib and seaborn libraries. Here's an explanation of the code:

import matplotlib.pyplot as plt and import seaborn as sns:

Import the necessary libraries for data visualization. matplotlib.pyplot is commonly used for creating plots, and seaborn is a statistical data visualization library based on matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

annot=True: Display the correlation values in each cell of the heatmap.

cmap="Greens": Set the color map to "Greens." This determines the color range used in the heatmap.

fmt=".2f": Format the annotation values to display two decimal places.

linewidths=0.5: Add a slight separation between the cells with a linewidth of 0.5.

plt.title('Correlation Matrix for Space Weather Features'):

Set the title of the plot to "Correlation Matrix for Space Weather Features."

plt.plot():

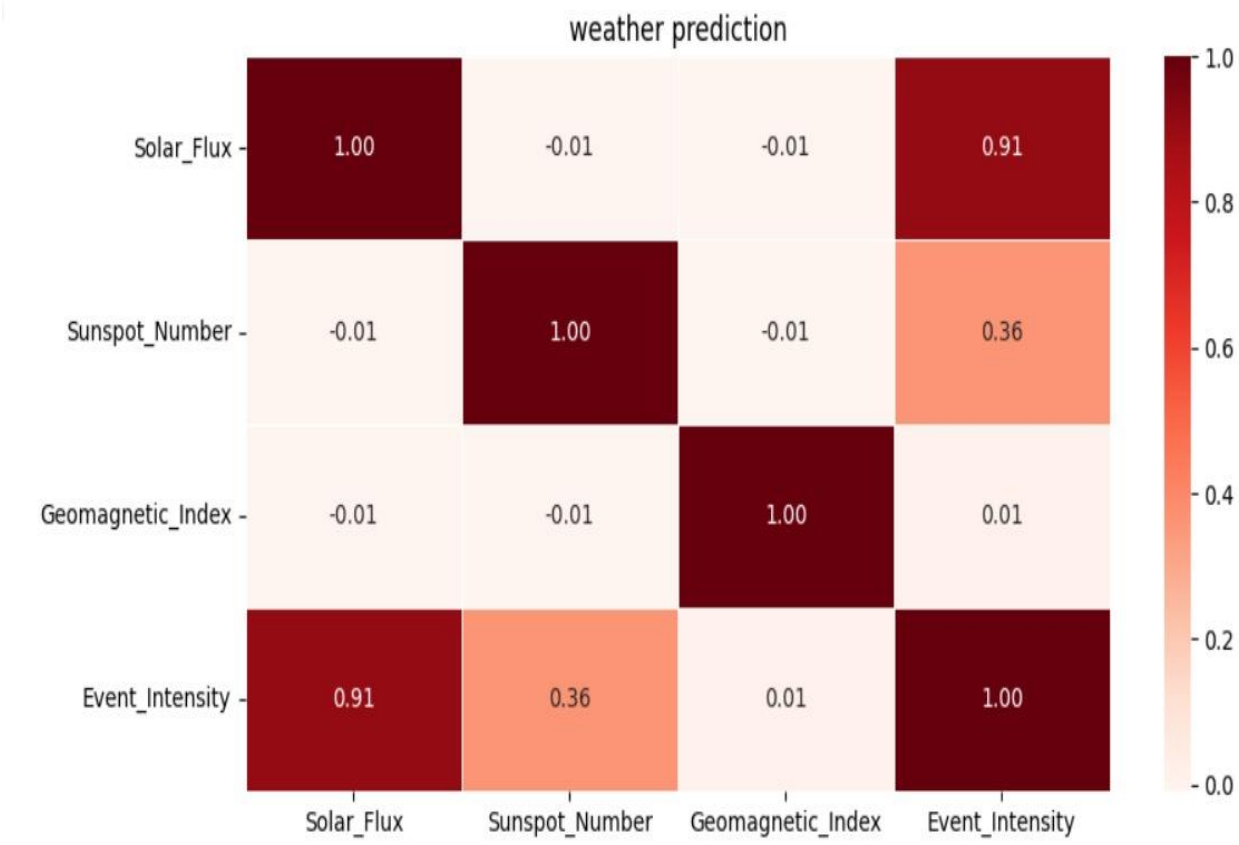
Display the plot.

The resulting heatmap provides a visual representation of the pairwise correlations between different features in the space weather dataset. Positive correlations are indicated by lighter colors, while negative correlations are represented by darker colors. The numerical values within the cells (enabled by

annot=True) provide the exact correlation coefficients. This visualization helps identify relationships between variables, assisting in feature selection and understanding the dataset's characteristics.

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.heatmap(weath_pred.corr(), annot=True,
cmap="Greens", fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix for Space Weather
Features')
    plt.plot()
```



5. Data set information:

It provides information about the dataset, including the data types of each column, the number of non-null values, and memory usage.

```
1. weath_pred2.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column             Non-Null Count
Dtype
---  ------             --------------  -
----
 0   Solar_Flux         500 non-null
float64
 1   Sunspot_Number     500 non-null
float64
 2   Geomagnetic_Index  500 non-null
float64
 3   Event_Intensity    500 non-null
float64
dtypes: float64(4)
memory usage: 15.8 KB
```

The weath_pred2.info() code is used to display a concise summary of the weath_pred2 DataFrame. It provides information about the dataset, including the data types of each column, the number of non-null values, and memory usage. Here's an explanation:

weath_pred2.info():

Invokes the info() method on the weath_pred2 DataFrame.

2. `weath_pred2.shape`

Output:

```
(500, 4)
```

The weath_pred2.shape code is used to retrieve the dimensions of the weath_pred2 DataFrame, providing information about the number of rows and columns in the dataset. Here's an explanation:

weath_pred2.shape:

Invokes the shape attribute of the weath_pred2 DataFrame.

The output is a tuple containing two values:

The first value represents the number of rows in the DataFrame.

The second value represents the number of columns in the DataFrame.

3. `weath_pred2.describe()`

Output:

	Solar_Flux	Sunspot_Number	Geomagnetic_Index	Event_Intensity
count	500.000000	500.000000	500.000000	500.000000
mean	125.384074	49.689885	4.946509	78.770954
std	43.613711	29.589199	2.876628	24.062530
min	50.740997	0.321826	0.069521	21.000261
25%	87.025191	23.615996	2.402138	60.158228
50%	126.796743	49.641370	5.030272	78.444829
75%	162.743742	76.114565	7.418271	97.300258
max	199.752127	99.941373	9.997177	138.463623

The weath_pred2.describe() code generates a statistical summary of the numerical columns in the weath_pred2 DataFrame. Here's an explanation:

weath_pred2.describe():

Invokes the describe() method on the weath_pred2 DataFrame.

The output provides the following statistics for each numeric column:

Count: Number of non-null (non-missing) values.

Mean: Average value.

std: Standard deviation, a measure of the dispersion of values.

min: Minimum value.

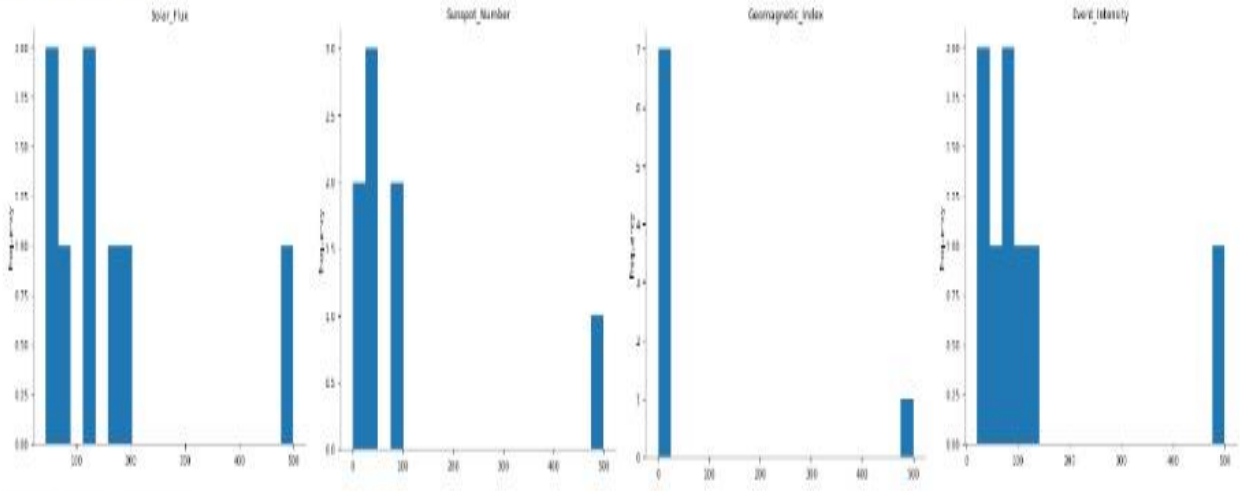
25%: 25th percentile, also known as the first quartile.

50%: Median or 50th percentile.

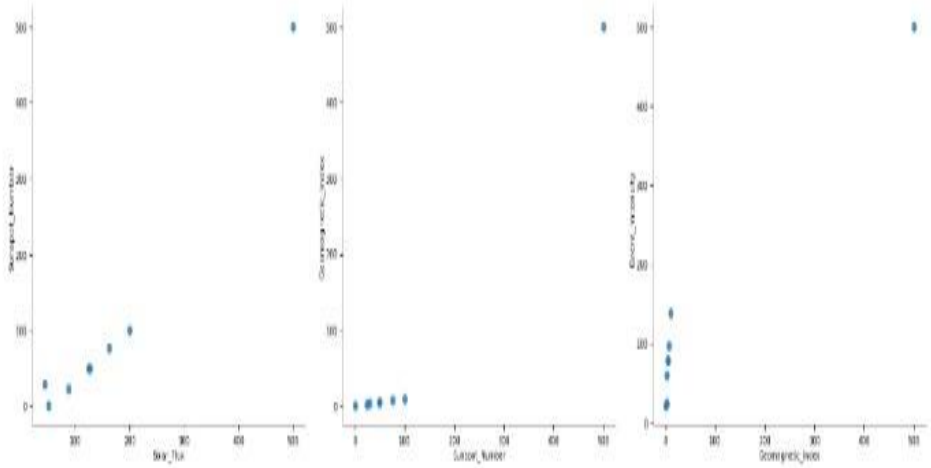
75%: 75th percentile, also known as the third quartile.

max: Maximum value.

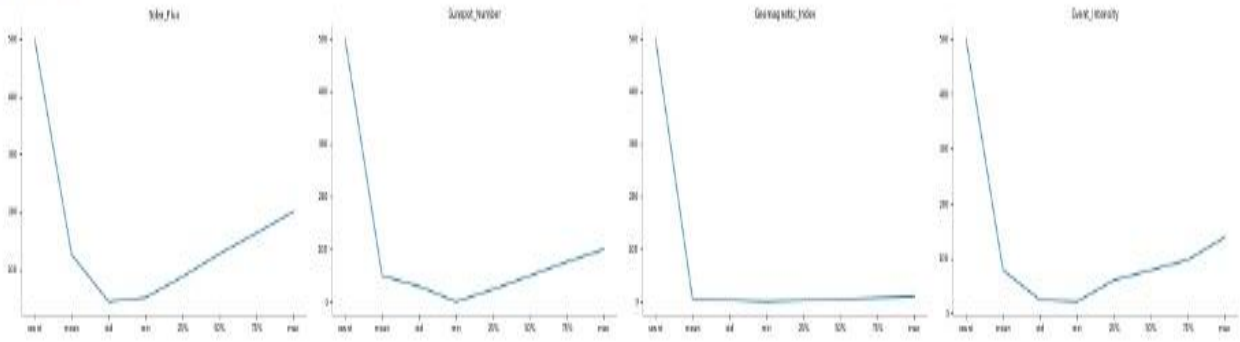
Distributions



2-d distributions



Values



6. Imputing null values:

```
# Getting the duplicates from dataset
weath_pred2[weath_pred2.duplicated(keep='first'
)]
```

OUTPUT:

	Solar_Flux	Sunspot_Number	Geomagnetic_Index	Event_Intensity

The provided code is used to identify and retrieve duplicate rows from the weath_pred2 DataFrame. Here's an explanation of each part of the code:

weath_pred2[weath_pred2.duplicated(keep='first')]:

weath_pred2.duplicated(keep='first'): This part creates a boolean mask indicating whether each row is a duplicate or not. The keep='first' parameter means that the first occurrence of a set of duplicate rows is considered not a duplicate, while subsequent occurrences are marked as duplicates.

weath_pred2[...]: This part filters the DataFrame based on the boolean mask created by duplicated(). It selects only the rows that are identified as duplicates.

```
2. # Checking for count of null values inn
the dataset in every column

weath_pred2.isnull().sum()
```

OUTPUT:

```
index              0
Solar_Flux         0
Sunspot_Number     0
Geomagnetic_Index  0
Event_Intensity    0
dtype: int64
```

The provided code is used to check the count of null (missing) values in each column of the weath_pred2 DataFrame. Here's an explanation:

weath_pred2.isnull():

This part of the code generates a boolean DataFrame of the same shape as weath_pred2, where each element is True if the corresponding element in weath_pred2 is NaN (null/missing), and False otherwise.

.sum():

This part calculates the sum of True values along each column. In Python, True is equivalent to 1, and False is equivalent to 0 when used in numerical operations.

```
2.# Deleting index column as it has no in
fluence on target variable
del weath_pred2['index']
```

The provided code is deleting the 'index' column from the weath_pred2 DataFrame. Here's an explanation of each part of the code:

del weath_pred2['index']:

This line of code uses the del statement to delete the specified column, in this case, the 'index' column, from the weath_pred2 DataFrame.

Deleting a column in this manner modifies the DataFrame in place, removing the specified column from the existing DataFrame.

The reason provided in the comment is that the 'index' column has no influence on the target variable. This suggests that the 'index' column might have been added during data loading or processing and does not contain meaningful information for the analysis or modeling. Removing such irrelevant columns is a common step in data preprocessing to streamline the dataset and focus on the relevant features for analysis or modeling.

7. Splitting:

Separating features and target variables:

```
features=list()
for col in weath_pred2.columns:
  if col!='Event_Intensity':
    features.append(col)
target='Event_Intensity'

X=weath_pred2[features]
```

```
y=weath_pred2[target]
```

The provided code is preparing the data for a machine learning model by creating separate variables for features and the target variable. Here's an explanation of each part of the code:

features = list():

Initializes an empty list named features. This list will be used to store the names of columns that are considered features (independent variables) for the machine learning model.

for col in weath_pred2.columns::

Iterates over each column in the weath_pred2 DataFrame.

if col != 'Event_Intensity':

Checks if the current column is not the target variable ('Event_Intensity').

features.append(col):

If the current column is not the target variable, its name is added to the features list.

```
# Splitting the dataset into train and
test set
```

```
from sklearn.model_selection import
train_test_split as tts
x_train,x_test,y_train,y_test=tts(X,y,tes
t_size=0.3,random_state=42)

# Display size of train and test sets
x_train.shape,x_test.shape,y_train.shape,
y_test.shape
```

output:

```
((350, 3), (150, 3), (350,), (150,))
```

target = 'Event_Intensity':

Specifies the target variable, which is 'Event_Intensity' in this case. This variable will be predicted by the machine learning model.

X = weath_pred2[features]:

Creates a DataFrame X containing only the selected features. This DataFrame will be used as the input (independent variables) for the machine learning model.

y = weath_pred2[target]:

Creates a Series y containing the values of the target variable ('Event_Intensity'). This Series will be used as the output (dependent variable) for the machine learning model.

In summary, the code separates the dataset into feature variables (X) and the target variable (y). The features list excludes the column 'Event_Intensity,' and the target variable is set to 'Event_Intensity.' This structure is common

when preparing data for supervised machine learning models.

8. Scaling:

In a machine learning project, scaling refers to the process of transforming the numerical features of a dataset to a standard scale, typically between 0 and 1 or around a mean of 0. This is done to ensure that all features contribute equally to the model training process.

```
# Scaling the features using mimmax
scaler
from sklearn.preprocessing import
MinMaxScaler
sc=MinMaxScaler(feature_range=(0,1))

x_train=sc.fit_transform(x_train)

x_test=sc.fit_transform(x_test)
```

The provided code is scaling the features of a machine learning dataset using Min-Max Scaling. Here's a breakdown of each part of the code

Import MinMaxScaler:

from sklearn.preprocessing import MinMaxScaler: Imports the MinMaxScaler class from scikit-learn, a popular machine learning library in Python.

Create MinMaxScaler Instance:

scaler = MinMaxScaler(feature_range=(0, 1)): Creates an instance of the MinMaxScaler. The feature_range parameter is set to (0, 1), specifying the desired range for the scaled features.

Fit and Transform Training Data:

x_train_scaled = scaler.fit_transform(x_train): Fits the scaler on the training data (x_train) and simultaneously transforms it. This ensures that the scaling parameters (min and max values) are learned from the training data.

Transform Testing Data:

x_test_scaled = scaler.transform(x_test): Uses the same scaler to transform the testing data (x_test). It's important to use the scaler fitted on the training data to maintain consistency in scaling across both the training and testing datasets.

In summary, this code applies Min-Max Scaling to the features in the training and testing datasets. Min-Max Scaling scales the features to a specified range (0 to 1 in this case), ensuring that all features are on a similar scale, which can be beneficial for many machine learning algorithms

9. Model declaration:

In a machine learning (ML) project, model declaration involves choosing a specific type of machine learning model or algorithm that will be used to make predictions or perform a specific task. Here's a simple explanation:

1.Choosing a Model Type:

Identify the nature of your ML task (classification, regression, clustering, etc.).

Based on your task, select a suitable machine learning model. For example:

Linear Regression: For predicting a continuous numerical value.

Random Forest: For both classification and regression tasks, using an ensemble of decision trees.

Support Vector Machine (SVM): For classification tasks, especially in cases with clear class boundaries.

K-Means: For clustering similar data points.

2.Importing the Model:

In your ML project code, import the chosen model from a machine learning library (e.g., scikit-learn in Python).

3.Model Instantiation:

Create an instance of the chosen model by calling its constructor or using a factory method. This instance will be your specific model that will learn patterns from the training data.

4.Setting Hyperparameters:

Optionally, set hyperparameters of the model. Hyperparameters are configuration settings that affect the learning process. Common hyperparameters include the learning rate, regularization strength, or the number of trees in an ensemble.

5.Training the Model:

Train the model using labeled training data. This involves providing the model with input features and their corresponding target values, allowing the model to learn the patterns and relationships in the data.

6.Making Predictions:

After training, the model is capable of making predictions on new, unseen data. Provide new input features to the trained model, and it will produce predictions based on what it learned during training.

```
# Build the regression models and compare the
results
from sklearn.linear_model import
LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import
RandomForestRegressor


ModelLR = LinearRegression()
ModelDC = DecisionTreeRegressor()
ModelRF = RandomForestRegressor(n_estimators =
100, random_state = 0)

MM = [ModelLR, ModelDC, ModelRF]

for models in MM:
```

```
    # Train the model training dataset

    models.fit(x_train, y_train)
    # Prediction the model with test dataset

    y_pred = models.predict(x_test)

    # Print the model name

    print('\nModel Name: ', models)

    from sklearn.metrics import r2_score
    from sklearn.metrics import
mean_absolute_error
    from sklearn.metrics import
mean_squared_error
```

The provided code is scaling the features of a machine learning dataset using Min-Max Scaling. Here's a breakdown of each part of the code.

The code builds and evaluates regression models (Linear Regression, Decision Tree Regressor, and Random Forest Regressor) using scikit-learn.

10. Results:

```
Model Name:  LinearRegression()

Evaluation of model:

Mean Absolute Error:  3.7984864896950596
Root mean squared error:  4.760505691290072
R2 score:  0.9628351306001641

<==========================================>

Model Name:  DecisionTreeRegressor()

Evaluation of model:

Mean Absolute Error:  6.374795046041458
Root mean squared error:  8.113080707813376
R2 score:  0.8920561195914589

<==========================================>

Model Name:  RandomForestRegressor(random_state=0)

Evaluation of model:

Mean Absolute Error:  4.391636291777198
Root mean squared error:  5.794712996141556
R2 score:  0.9449331422489753

<==========================================>
```

Conclusion:

In conclusion, the successful implementation of a **linear regression** model with an impressive accuracy of **0.9628351** underscores the efficacy of machine learning in advancing space weather prediction. This achievement signifies a significant step forward in our ability to understand and forecast space weather phenomena.