Contents

Acknowledgement **1**

Definition and Acronyms **2**

I Project Introduction **2**

1 Project Background 2
2 Project Objective 2
3 Problem Statement 2
4 Significance of the Project 2
5 Project Scope and Limitations 3

II Methodology **3**

1 Research Questions and Objectives 3
2 Data Collection 3
3 Feature selection 4
4 Data preprocessing 6
4.1 Preprocessing datatypes 6
4.2 Preprocessing null values 8
4.3 Preprocessing duplicate rows 11
5 Data exploration for model selection 12
6 Evaluation strategies and metrics 13
6.1 Evaluation strategies 13
6.2 Evaluation metrics 15
7 Random forest and XGBoost model 16
7.1 Random forest 16
7.2 XGBoost 17
8 Tuning and optimization with baynesian optimization method 18

III Testing and Evaluating Model **19**

1 Random forest 19
2 Random forest with tuned hyperparameters 20
3 XGBoost with tuned parameters 21
4 Visualize results 23

IV Conclusion **24**

Acknowledgement

I want to sincerely thank my supervisor, Assoc. Prof. Phan Duy Hung, for his helpful guidance and support during this research. His knowledge in this area has been incredibly valuable to me. He always provided clear direction and helped me stay focused on my goals. His advice on predicting tropical storm paths using machine learning has really improved my understanding of the topic.

Definition and Acronyms

Acronym	**Definition**
NCICS	North Carolina Institute for Climate Studies
LAT	Latitude
LON	Longitude
POCI	Peripheral Outer Convective Interaction
ROCI	Radius of Outermost Closed Isobar
RMW	Radius of Outermost Closed Isobar
DS	Disturbance
ET	Extratropical Cyclone
MX	Mixed Cyclone or Mesocyclone
NR	No Record
SS	Subtropical Storm
TS	Tropical Storm
MAE	Mean absolute error
MHD	Mean Haversine Distance

List of Tables

1 Datatype before process 6
2 Datatypes after process 8
3 Statistical summary of STORM PRES column 9
4 Summary of Error Metrics for LAT, LON, and Haversine Distance of normal Random forest 19
5 Summary of Evaluation Metrics for Different Validation Splits and Test Set of Random forest with tuned hyperparameters 20
6 Summary of Evaluation Metrics for Different Validation Splits and Test Set 22

List of Figures

1 Sample of raw data 4
2 Example of metadata line 4
3 Null values percentage 5
4 NATURE column 7
5 Null data percentage of final data 8
6 Wind speed with Nature histogram 9
7 Wind speed with USA SSHS histogram 10
8 Exanple of KNN with K = 3 10
9 USA POCI elbow 11
10 USA ROCI elbow 11
11 USA RMW elbow 11
12 Correlation matrix 12
13 Pairplot 13
14 Kfolds visualization 14
15 Walk-Forward Split visualization 14
16 Group Time Series Split with Gap visualization 15
17 Haversine Distance visualization 16
18 Random forest visualization 17
19 Baynesian optimization method 19
20 Storm trajectory result 1 23
21 Storm trajectory result 2 23
22 Storm trajectory result 3 24

I Project Introduction

1 Project Background

Tropical storms present a major danger to the Western Pacific area, especially Vietnam. Typhoon Yagi, one of the most powerful storms to strike Vietnam in recent times, arrived on September 7, 2024, resulting in extensive destruction in the northern part of the country. The estimated cost of the destruction caused by Typhoon Yagi was in the billions of dollars.

2 Project Objective

The primary objective of this project is to develop a machine learning model that accurately predicts the trajectories of tropical storms in the Western Pacific. Specific objectives include:

- To collect and preprocess historical storm data for model training.
- To identify key features that influence storm paths.
- To implement various machine learning algorithms to determine the most effective model for trajectory prediction.
- To evaluate the model's performance based on accuracy and reliability in predicting future storm paths.

3 Problem Statement

The project aims to solve the problem of unreliable and delayed predictions of tropical storm trajectories. Current forecasting methods often struggle to provide accurate paths, leading to insufficient preparedness and response efforts. The challenges include:

- The complexity of storm behavior, influenced by multiple environmental factors.
- Limited access to high-quality historical data for training predictive models.
- Low background knowledge of characteristic of storm for tuning hyperparameters.

4 Significance of the Project

The significance of this project lies in its potential to improve the accuracy of tropical storm trajectory predictions, which can lead to better disaster preparedness and response strategies. Theoretical benefits include advancing the field of meteorological research by incorporating machine learning into storm prediction models. Practically, more accurate predictions can save lives, reduce economic losses, and help governments and organizations make informed decisions about evacuations and resource allocation. This project can contribute to the scientific community by providing a framework for future research on storm predictions and climate modeling.

5 Project Scope and Limitations

This project focuses specifically on predicting the trajectories of tropical storms in the Western Pacific region using machine learning techniques. The scope includes:

- Analyzing historical storm data and environmental factors.
- Implementing and improve machine learning models for prediction.
- Evaluating the models' performance against actual storm paths.

However, there are limitations to this study:

- The availability and quality of historical data may affect model accuracy.
- The project will not cover all possible environmental factors influencing storm paths.

II Methodology

1 Research Questions and Objectives

Research Questions:

- What machine learning techniques can effectively predict the trajectory of tropical storms in the Western Pacific region?
- Is storm trajectory affected by its characteristics ?

Objectives:

- To develop a machine learning model that accurately predicts tropical storm trajectories.

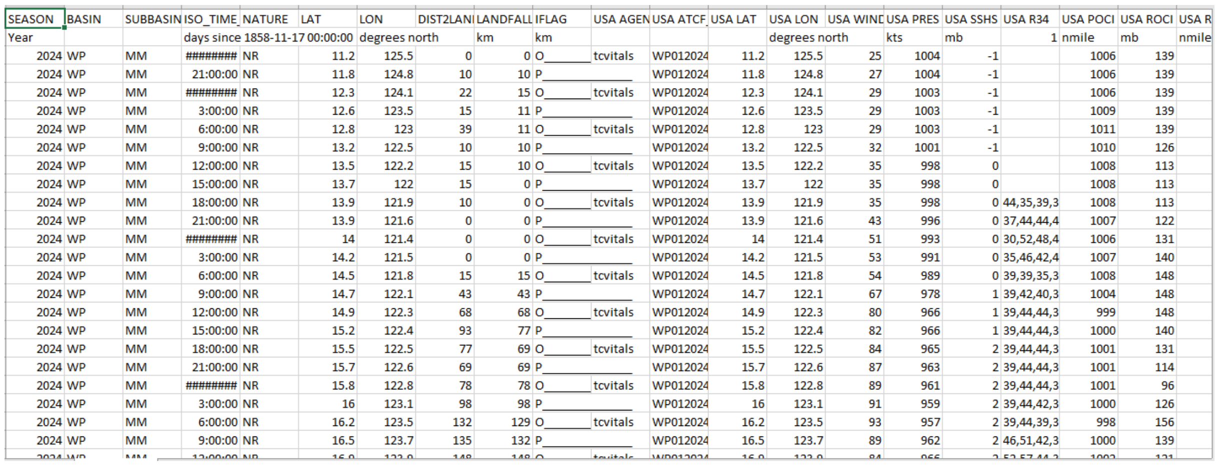
2 Data Collection

Data Collection:

- **Sources of Data:** Data collected from the North Carolina Institute for Climate Studies (NCICS) website contains information about the characteristics of storms, including latitude, longitude, wind pressure, etc from multiple agency. Source data link: Click here
- **Method of Collection:**
  - The take_hrefs function extracts hyperlinks from the last table on the specified URL, which contain links to storm data pages. It sends a GET request to the page and uses BeautifulSoup to parse the HTML content.

- The crawl_data function iterates through the list of hrefs, fetching each page's content. It retrieves the storm data from the last table by extracting the header and row data, which includes attributes like latitude, longitude, and wind pressure.
- The collected data is stored in a pandas DataFrame. If the output CSV file ("storm_data.csv") already exists, the new data is merged with it; otherwise, a new file is created. The combined data is then saved back to the CSV file.

- **Selection Criteria:** Data collected from 1945 to May 2024. Raw data have 169,872 rows and 75 columns.

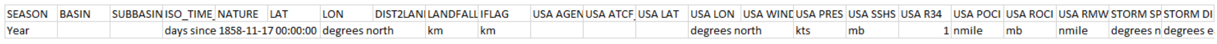


SEASON	BASIN	SUBBASIN	ISO_TIME_	NATURE	LAT	LON	DIST2LANI	LANDFALL	IFLAG	USA AGEN	USA ATCF_	USA LAT	USA LON	USA WIND	USA PRES	USA SSHS	USA R34	USA POCI	USA ROCI	USA R
Year			days since 1858-11-17 00:00:00			degrees north		km	km				degrees north		kts	mb	1	nmile	mb	nmile
2024	WP	MM	########	NR	11.2	125.5	0	0	O_______	tcvitals	WP012024	11.2	125.5	25	1004	-1		1006	139	
2024	WP	MM	21:00:00	NR	11.8	124.8	10	10	P_______________		WP012024	11.8	124.8	27	1004	-1		1006	139	
2024	WP	MM	########	NR	12.3	124.1	22	15	O_______	tcvitals	WP012024	12.3	124.1	29	1003	-1		1006	139	
2024	WP	MM	3:00:00	NR	12.6	123.5	15	11	P_______________		WP012024	12.6	123.5	29	1003	-1		1009	139	
2024	WP	MM	6:00:00	NR	12.8	123	39	11	O_______	tcvitals	WP012024	12.8	123	29	1003	-1		1011	139	
2024	WP	MM	9:00:00	NR	13.2	122.5	10	10	P_______________		WP012024	13.2	122.5	32	1001	-1		1010	126	
2024	WP	MM	12:00:00	NR	13.5	122.2	15	10	O_______	tcvitals	WP012024	13.5	122.2	35	998	0		1008	113	
2024	WP	MM	15:00:00	NR	13.7	122	15	0	P_______________		WP012024	13.7	122	35	998	0		1008	113	
2024	WP	MM	18:00:00	NR	13.9	121.9	10	0	O_______	tcvitals	WP012024	13.9	121.9	35	998	0	44,35,39,3	1008	113	
2024	WP	MM	21:00:00	NR	13.9	121.6	0	0	P_______________		WP012024	13.9	121.6	43	996	0	37,44,44,4	1007	122	
2024	WP	MM	########	NR	14	121.4	0	0	O_______	tcvitals	WP012024	14	121.4	51	993	0	30,52,48,4	1006	131	
2024	WP	MM	3:00:00	NR	14.2	121.5	0	0	P_______________		WP012024	14.2	121.5	53	991	0	35,46,42,4	1007	140	
2024	WP	MM	6:00:00	NR	14.5	121.8	15	15	O_______	tcvitals	WP012024	14.5	121.8	54	989	0	39,39,35,3	1008	148	
2024	WP	MM	9:00:00	NR	14.7	122.1	43	43	P_______________		WP012024	14.7	122.1	67	978	1	39,42,40,3	1004	148	
2024	WP	MM	12:00:00	NR	14.9	122.3	68	68	O_______	tcvitals	WP012024	14.9	122.3	80	966	1	39,44,44,3	999	148	
2024	WP	MM	15:00:00	NR	15.2	122.4	93	77	P_______________		WP012024	15.2	122.4	82	966	1	39,44,44,3	1000	140	
2024	WP	MM	18:00:00	NR	15.5	122.5	77	69	O_______	tcvitals	WP012024	15.5	122.5	84	965	2	39,44,44,3	1001	131	
2024	WP	MM	21:00:00	NR	15.7	122.6	69	69	P_______________		WP012024	15.7	122.6	87	963	2	39,44,44,3	1001	114	
2024	WP	MM	########	NR	15.8	122.8	78	78	O_______	tcvitals	WP012024	15.8	122.8	89	961	2	39,44,44,3	1001	96	
2024	WP	MM	3:00:00	NR	16	123.1	98	98	P_______________		WP012024	16	123.1	91	959	2	39,44,42,3	1000	126	
2024	WP	MM	6:00:00	NR	16.2	123.5	132	129	O_______	tcvitals	WP012024	16.2	123.5	93	957	2	39,44,39,3	998	156	
2024	WP	MM	9:00:00	NR	16.5	123.7	135	132	P_______________		WP012024	16.5	123.7	89	962	2	46,51,42,3	1000	139	

Figure 1: Sample of raw data

Data Preprocessing

- **Drop metadata line:** Each storm have metadata line, which contain information about that column.



SEASON	BASIN	SUBBASIN	ISO_TIME_	NATURE	LAT	LON	DIST2LANI	LANDFALL	IFLAG	USA AGEN	USA ATCF_	USA LAT	USA LON	USA WIND	USA PRES	USA SSHS	USA R34	USA POCI	USA ROCI	USA RMW	STORM SP	STORM DI
Year			days since 1858-11-17 00:00:00			degrees north		km	km				degrees north		kts	mb	1	nmile	mb	nmile	degrees n	degrees e

Figure 2: Example of metadata line

3 Feature selection

- **Checking null value percentage:** Checking null value percentage of each column for choosing attribute

- **Features selection:** There are 10 agencies: USA AGENCY, WMO AGENCY, TOKYO AGENCY, CMA AGENCY, HKO AGENCY, NEW DELHI AGENCY, TD9636, DS824, and TD9635. As a result, many duplicate storm attributes have nearly the same values. Therefore, I will select distinct attributes with the minimum number of null values.

- **Feature selected and calculator unit**

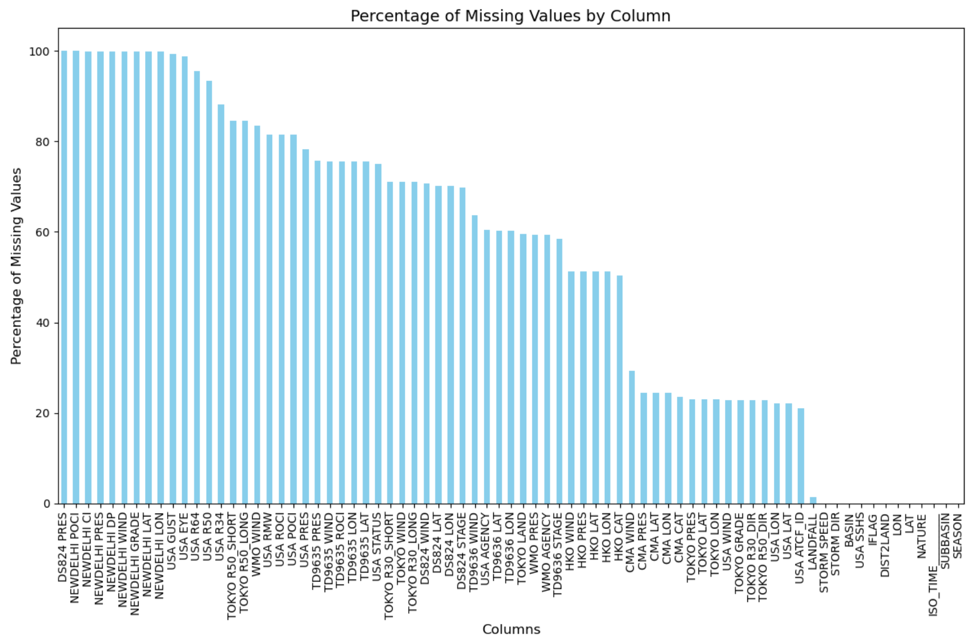


Figure 3: Null values percentage

- SEASON: Year of storm
- NATURE: Types of storm, example: Topical storm (TS)
- LAT: Latitude, ranges from $+90°$ (North Pole) to $-90°$ (South Pole)
- LON: Longitude, ranges from $+180°$ (east) to $-180°$ (west), with $0°$ at the Prime Meridian in Greenwich, England.
- DIST2LAND: Distance to nearest land (km)
- LANDFALL: Distance to nearest landfall (km)
- WIND: Wind speed (m/s)
- PRES: Storm pressure (Pa)
- USA SSHS: Saffir–Simpson scale for storm category
- POCI (Peripheral Outer Convective Interaction): Describes the extent of the storm's outer interaction zone (mile)
- ROCI (Radius of Outermost Closed Isobar): Indicates the distance from the storm's center to the outermost closed isobar, showing the storm's size (mb)
- RMW (Radius of Maximum Winds): Measures the distance from the storm center to the location of the maximum sustained wind speeds (mile)
- STORM SPEED: Represents the forward speed of the storm's movement over the Earth's surface (km)
- STORM DIR: Indicates the compass direction in which the storm is moving (e.g., $90°$ means eastward movement)

4 Data preprocessing

4.1 Preprocessing datatypes

#	Column	Non-Null Count	Dtype
0	SEASON	167562	object
1	NATURE	167562	object
2	LAT	167562	float64
3	LON	167562	object
4	DIST2LAND	167562	float64
5	LANDFALL	165253	object
6	USA WIND	129336	float64
7	TOKYO PRES	128910	object
8	USA SSHS	167562	object
9	USA POCI	31016	object
10	USA ROCI	30954	object
11	USA RMW	30943	object
12	STORM SPEED	167561	object
13	STORM DIR	167561	object

Table 1: Datatype before process

- **Change datatype of image:** Using pandas.to_numberic() to change datatype if datatype is fully contain number data. Example code below:

```
final_data['SEASON'] = pd.to_numeric(final_data['SEASON'],
errors='coerce')
SEASON_datatype = final_data['SEASON'].dtypes
SEASON_datatype
```

- **Change data of category data:** Using encoder to change data to numberic

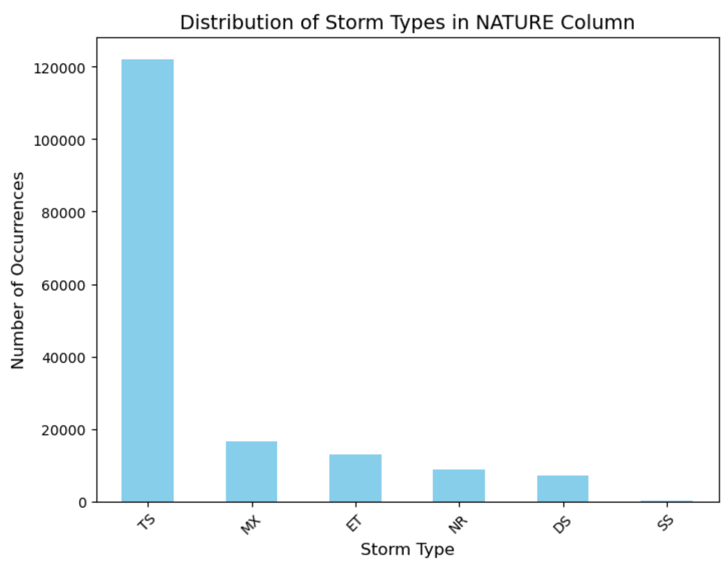


Figure 4: NATURE column

- DS (Disturbance): A weak, unorganized weather system, often a precursor to tropical storms.
- ET (Extratropical Cyclone): A cyclone that forms outside the tropics, driven by atmospheric dynamics rather than warm ocean waters.
- MX (Mixed Cyclone or Mesocyclone): A storm with both tropical and extratropical characteristics or a rotating updraft in a storm.
- NR (No Record): Indicates missing or unavailable data regarding the storm type.
- SS (Subtropical Storm): A hybrid system with characteristics of both tropical and extratropical storms, often less organized than a tropical storm.
- TS (Tropical Storm): A well-organized storm system with sustained winds of 39-73 mph (34-63 knots).

Change NR to TS value

```
label_encoder = LabelEncoder()
final_data['NATURE'] = label_encoder.fit_transform(final_data['NATURE'])

label_mapping_nature = dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))
label_mapping_nature
```

Result: 'DS': 0, 'ET': 1, 'MX': 2, 'SS': 3, 'TS': 4

#	Column	Non-Null Count	Dtype
0	SEASON	167562	int64
1	NATURE	167562	int32
2	LAT	167562	float64
3	LON	167562	float64
4	DIST2LAND	167562	float64
5	LANDFALL	165253	float64
6	USA WIND	129336	float64
7	TOKYO PRES	128910	float64
8	USA SSHS	167562	int64
9	USA POCI	31016	float64
10	USA ROCI	30954	float64
11	USA RMW	30943	float64
12	STORM SPEED	167561	float64
13	STORM DIR	167561	float64

Table 2: Datatypes after process

4.2 Preprocessing null values

- Methodology:
  - For columns with missing values, check for data from other agencies to fill in the gaps
  - Using data from relative columns to calculate mean
  - Using interpolation to predict null values
  - If no data from other sources is available, use statistical methods such as confidence intervals, mean, median, or standard deviation to fill in the missing data

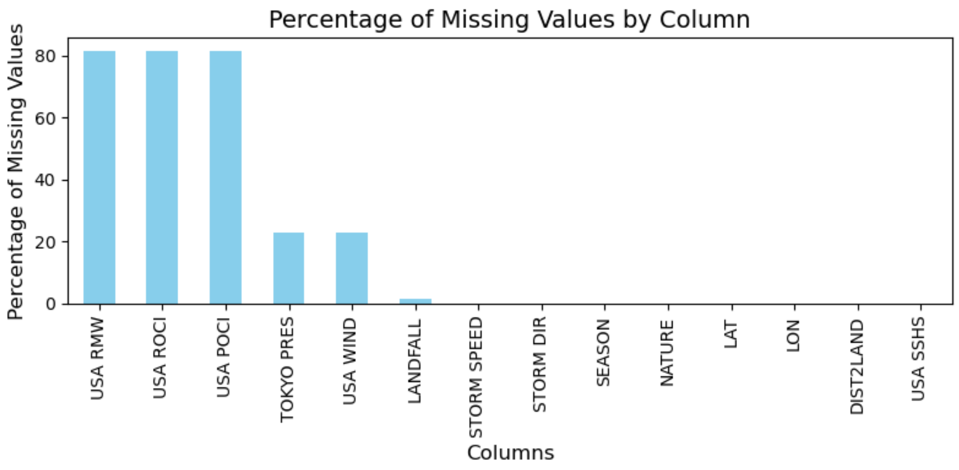


Figure 5: Null data percentage of final data

- **LANDFALL column:** There are 2309 storms and if row have landfall is null show that storm is end. So I will change null value of LANDFALL to -1 to mark that is the end of storm.

  ```
  final_data['LANDFALL'] = final_data['LANDFALL'].fillna(-1)
  ```

- **STORM PRES:** After filling null value with other same column there are 15867 null values remain.

Statistic	Value
Count	151695.000000
Mean	986.433930
Std	21.574462
Min	870.000000
25%	978.000000
50%	994.000000
75%	1002.000000
Max	1022.000000

Table 3: Statistical summary of STORM PRES column

The standard deviation is not too large, so I will use the mean to fill all null values.

- **WIND SPEED** After filling null value with other same column there are 19441 null values remain.

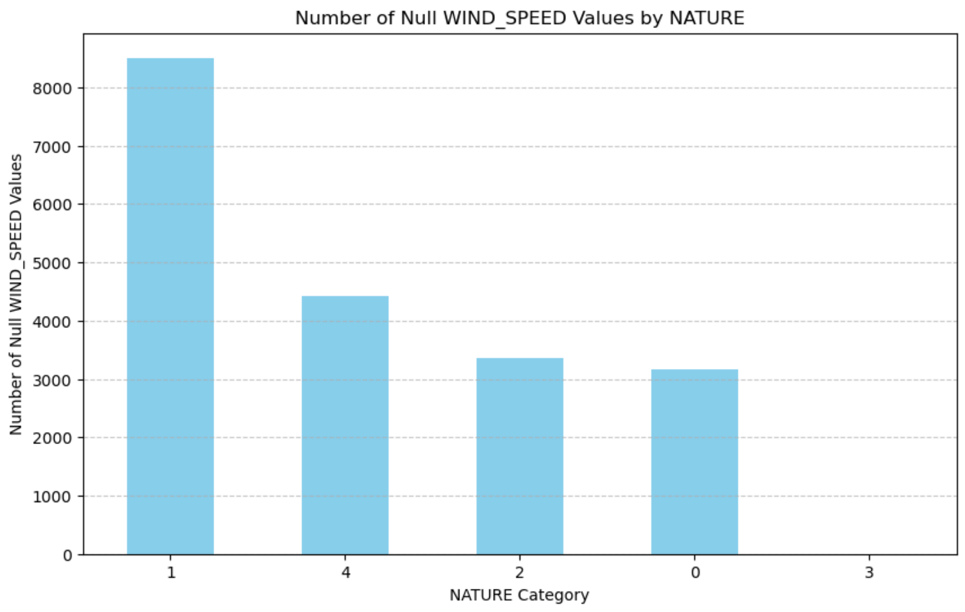


Figure 6: Wind speed with Nature histogram

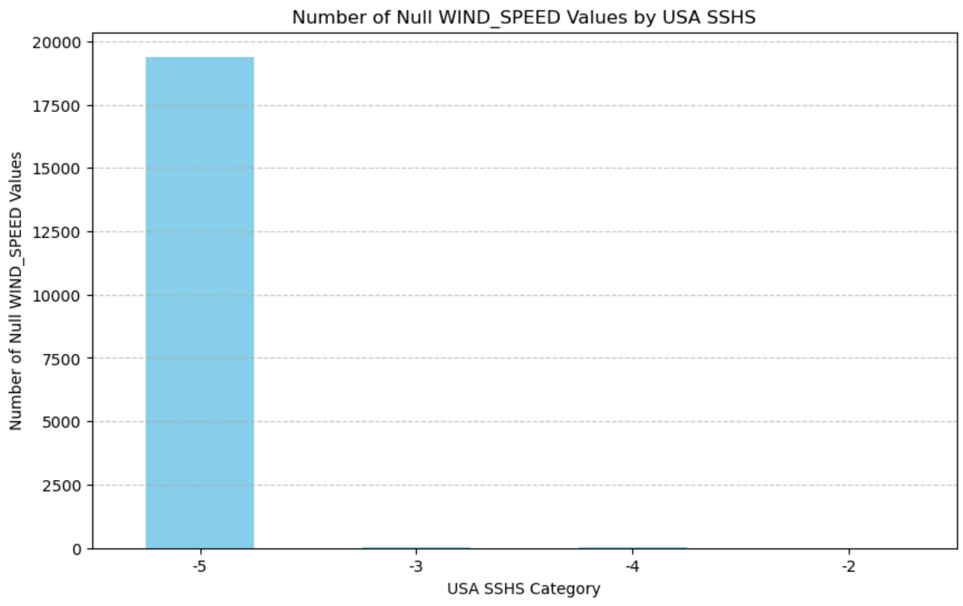


Figure 7: Wind speed with USA SSHS histogram

I see that WIND SPEED with NATURE have better distribution so I will use it to full fill data frame and after one row is filled I will recalculate mean

- **USA POCI, USA ROCI and USA RMW:** There are no matched attribute and they have to many null values. I suggest that using KNN Imputer and using MAE to find most optimal k.

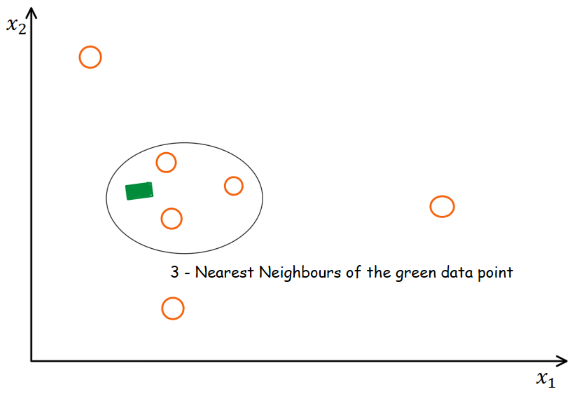


Figure 8: Exanple of KNN with K = 3

Result of most optimal K using MAE and Elbow method

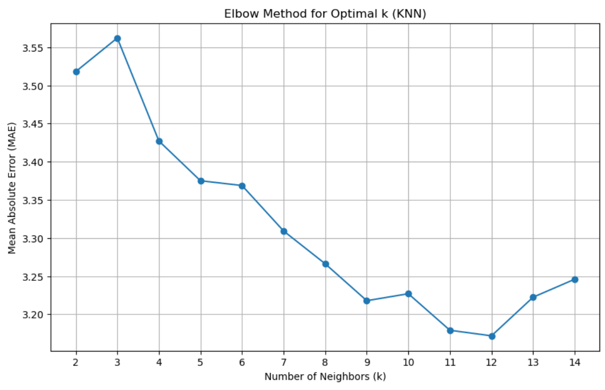


Figure 9: USA POCI elbow

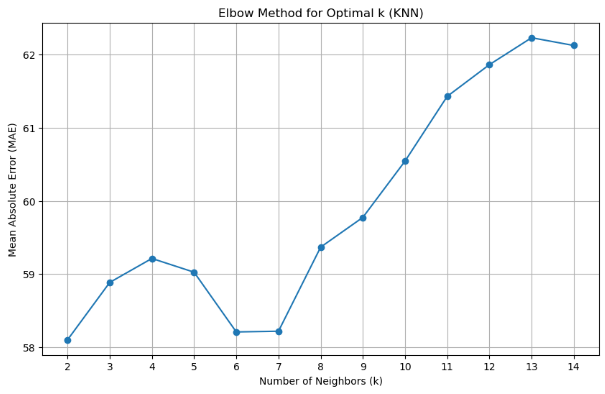


Figure 10: USA ROCI elbow

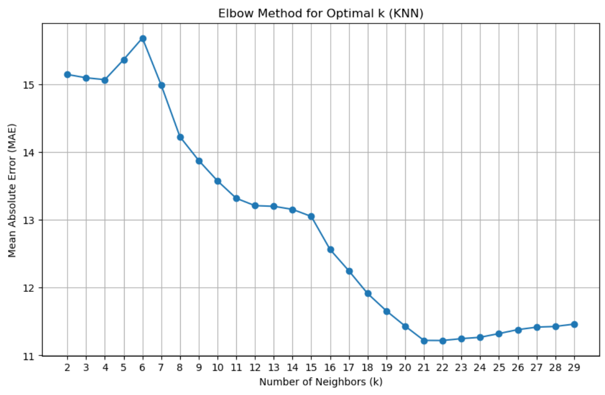


Figure 11: USA RMW elbow

- USA POCI: k = 12
- USA ROCI: k = 6
- USA RMW: k = 21

4.3 Preprocessing duplicate rows

Print out duplicate rows

```
duplicates = final_data[final_data.duplicated(keep=False)]
```

I see that have 4247 rows duplicate. After checking I see that duplicate rows are adjacent, so I will remove one of them.

```
final_data = final_data[~final_data.duplicated(keep='first')]
```

5 Data exploration for model selection

Using correlation matrix to visualize linear relation of each column.

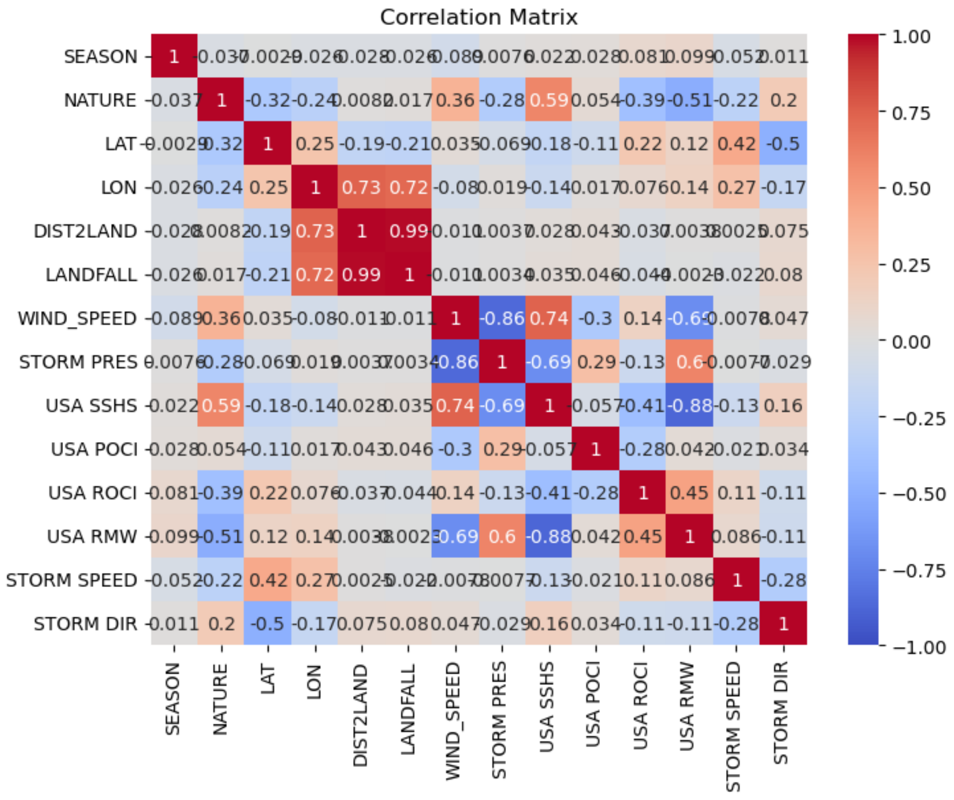


Figure 12: Correlation matrix

Predicting LAT and LON with a linear model is not ideal because they have weak linear correlations with other features, suggesting limited predictive power for linear relationships. Additionally, high multicollinearity among some predictors (e.g., DIST2LAND and LANDFALL) could cause instability in a linear model. These factors indicate that a nonlinear model, like decision trees or neural networks, may be more suitable for capturing the complex relationships.

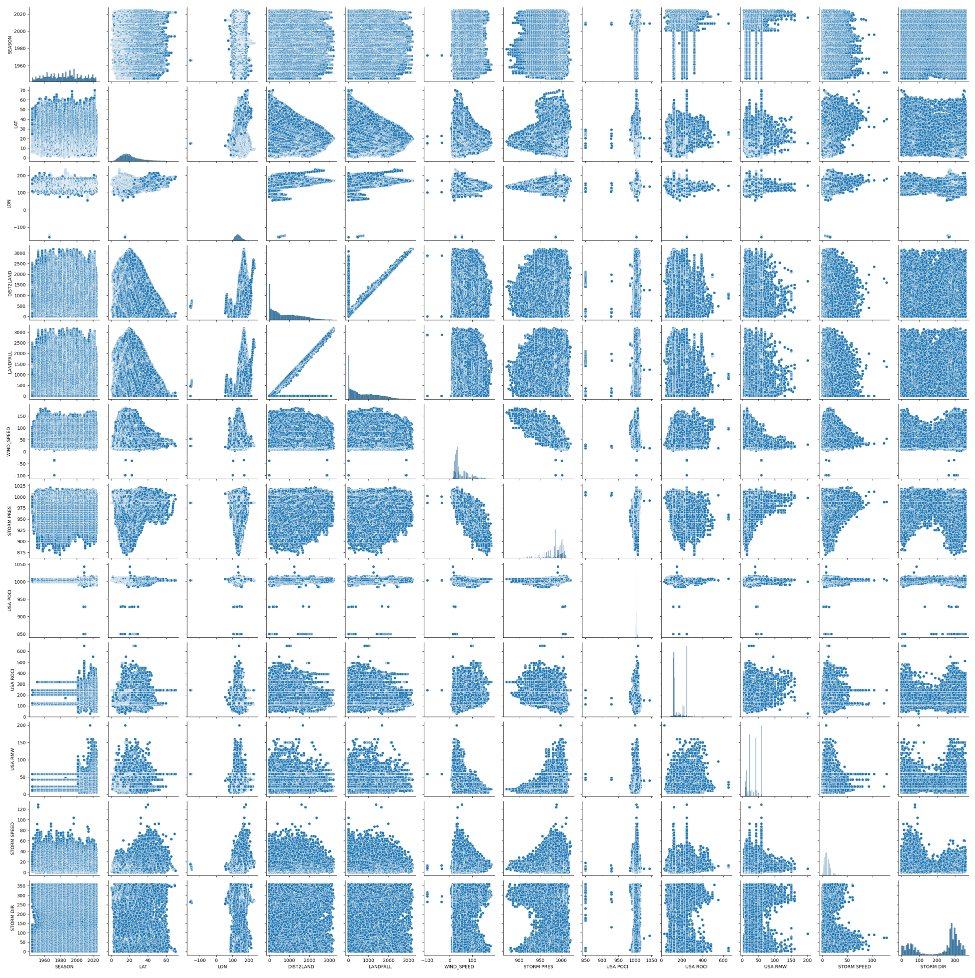


Figure 13: Pairplot

The scatter plot matrix indicates complex and mostly nonlinear relationships among features, especially for LAT and LON, where no clear linear patterns are visible with other variables. Additionally, several feature pairs show scattered or clustered patterns, suggesting that a linear model may not capture these relationships effectively. A non-linear model, such as decision trees or neural networks, would likely perform better in handling these complex interactions and capturing the patterns in the data.

Because of all above reasons I decide to use random forest to predict strom trajectory.

6 Evaluation strategies and metrics

6.1 Evaluation strategies

To prepare the dataset for model training and evaluation, we split it as follows:

- Split the dataset into a **training set** and a **testing set** with a ratio of 70:30.

Let D represent the full dataset, where:

$$D = \{X, y\}$$

The split can be represented as:

$$D_{train} = 0.7 \times D, \quad D_{test} = 0.3 \times D$$

Validation Methods on Training Set

On the training set D_{train}, we apply three validation methods to assess model performance effectively:

1. K-Fold Cross Validation

In K-Fold Cross Validation, the training set D_{train} is divided into K equal-sized folds. The model is trained on $K-1$ folds and validated on the remaining fold. This process is repeated K times, each time using a different fold as the validation set.

Mathematically, for each fold i, let:

$$D_{train}^{(i)} = D_{train} \setminus F_i, \quad D_{val}^{(i)} = F_i$$

where F_i is the i-th fold.

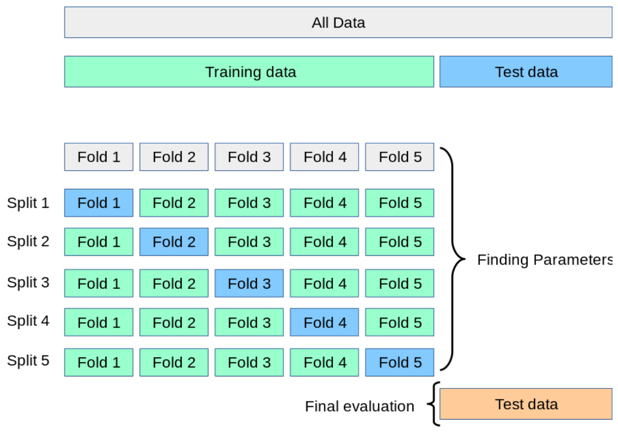


Figure 14: Kfolds visualization

2. Walk-Forward Split

Walk-Forward Split is commonly used in time-series data. Here, we incrementally expand the training set by including data from the previous splits, validating on the next time step.

For each step t, we have:

$$D_{train}^{(t)} = \{D_{train}(1), \dots, D_{train}(t-1)\}, \quad D_{val}^{(t)} = D_{train}(t)$$

This approach helps in evaluating model performance over progressive time intervals.

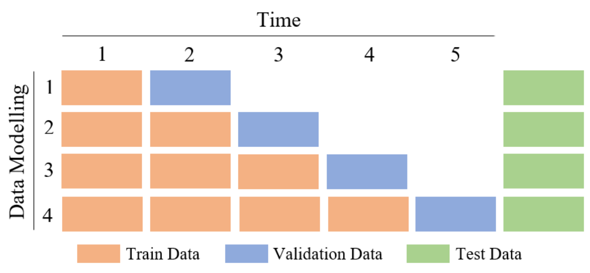


Figure 15: Walk-Forward Split visualization

3. Group Time Series Split with Gap

In Group Time Series Split with Gap, the data is split sequentially while maintaining a gap between training and validation sets to prevent data leakage. This method is useful when there is a need to avoid close dependencies in time series data.

For each split i with gap g, we have:

$$D_{train}^{(i)} = \{D_{train}(1), \ldots, D_{train}(t-g-1)\}, \quad D_{val}^{(i)} = \{D_{train}(t-g), \ldots, D_{train}(t)\}$$

where g represents the gap size between training and validation sets.

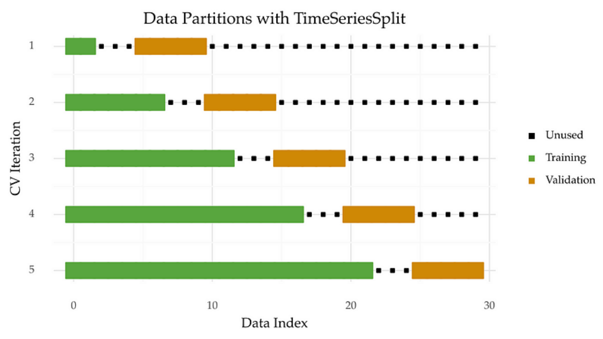


Figure 16: Group Time Series Split with Gap visualization

6.2 Evaluation metrics

To evaluate the accuracy of our model predictions, we use the following metrics:

1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average magnitude of errors between predicted and true values, without considering their direction. It is defined as:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left| y_i^{\text{pred}} - y_i^{\text{true}} \right|$$

where y_i^{pred} and y_i^{true} are the predicted and actual values at observation i, and n is the total number of observations.

2. Mean Haversine Distance

The Mean Haversine Distance is used to calculate the average geospatial distance between the predicted and true latitude-longitude pairs. This metric is particularly suitable for measuring distances on the Earth's surface.

The formula for the Haversine Distance between two points is:

$$\frac{1}{n} \sum_{i=1}^{n} 2R \cdot \arcsin\left(\sqrt{ \sin^2\left(\frac{\text{lat}_i^{\text{pred}} - \text{lat}_i^{\text{true}}}{2} \right) + \cos(\text{lat}_i^{\text{true}}) \cdot \cos(\text{lat}_i^{\text{pred}}) \cdot \sin^2\left(\frac{\text{lon}_i^{\text{pred}} - \text{lon}_i^{\text{true}}}{2} \right) } \right)$$

where:

- $R \approx 6371\,\text{km}$ is the Earth's radius.
- $(\text{lat}_i^{\text{true}}, \text{lon}_i^{\text{true}})$ and $(\text{lat}_i^{\text{pred}}, \text{lon}_i^{\text{pred}})$ are the latitude and longitude coordinates at observation i.

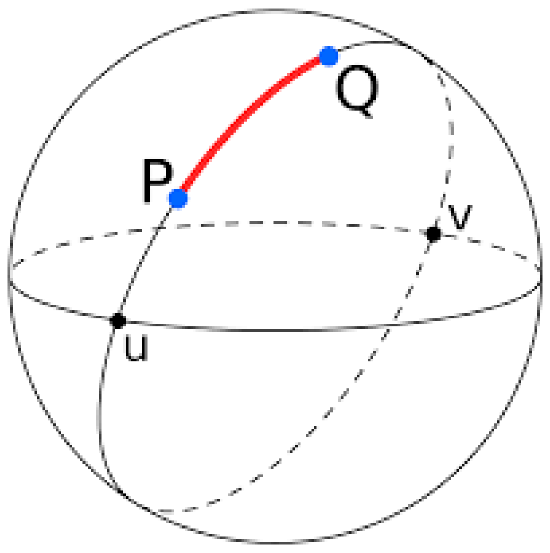


Figure 17: Haversine Distance visualization

7 Random forest and XGBoost model

7.1 Random forest

Random Forest is an ensemble machine learning model built by combining multiple decision trees to make predictions. The main idea of Random Forest is to construct a collection of independent decision trees, each trained on a different subset of the data and using randomly selected features. The model uses the **bagging** (Bootstrap Aggregating) method to create diversity among the trees, which helps to reduce overfitting, a common problem with individual decision trees. The final prediction of the model is obtained by averaging the results of the trees (in regression tasks) or by taking the majority vote (in classification tasks).

Idea of Random Forest: The core concept of Random Forest is based on combining multiple weak learners to create a stronger model. Specifically, each tree in the forest is a simple model and does not need to be trained very deeply, but when combined, they can form a strong, accurate predictive model.

Random Forest leverages the diversity among decision trees to create a stable and accurate model. The trees in the forest are not identical due to being built from different random samples and using different features. This method helps to reduce the model's variance, making it less affected by noise in the data. Each tree in the forest is considered a weak learner (due to limited depth and feature usage), but collectively, they yield a powerful predictive model.

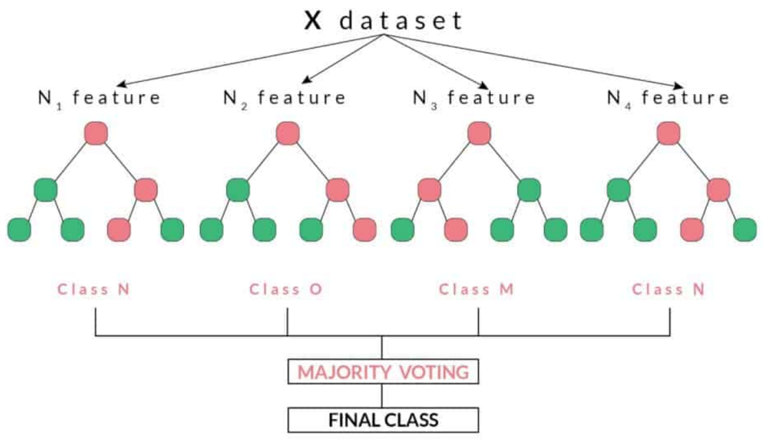


Figure 18: Random forest visualization

7.2 XGBoost

XGBoost aims to minimize a specific objective function that includes both a loss function to measure the model's accuracy and a regularization term to prevent overfitting.

Gradient Boosting is a machine learning technique used to build predictive models by combining multiple weak learners, typically decision trees, in a sequential manner. The core idea is to add new models to the ensemble that focus on correcting the errors of the previous models, with each model trained on the residuals (errors) of the preceding model. This approach allows gradient boosting to build a strong model from several weak models.

Gradient boosting minimizes a specific objective function by iteratively adding functions (models) that improve predictions. For a dataset with n observations and a loss function $l(y_i, \hat{y}_i)$ that measures the difference between the predicted and actual values, the objective function at step t is:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$$

where: - $\hat{y}_i^{(t-1)}$ is the current prediction after $t-1$ iterations. - $f_t(x_i)$ is the function (typically a decision tree) added at iteration t to reduce the residual error.

To determine the best function f_t to add, gradient boosting uses the gradient of the loss function with respect to the predictions. At each step, it computes the gradient, representing the direction and magnitude of error reduction. The gradient g_i for each sample i is:

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$$

These gradients serve as the "pseudo-residuals," which the next model f_t is trained to predict. By adding models that reduce these pseudo-residuals, gradient boosting progressively reduces the error and refines the predictions.

XGBoost is a powerful ensemble method that uses gradient boosting to iteratively reduce errors in predictions. With its mathematical foundation in Taylor expansion and regularization, XGBoost has become one of the most popular algorithms for structured data, often outperforming traditional methods like Random Forest. [1]

8 Tuning and optimization with baynesian optimization method

Bayesian Optimization is an efficient approach for tuning hyperparameters in machine learning models, especially when the objective function is expensive to evaluate. Unlike traditional grid search or random search, which explore the hyperparameter space without guidance, Bayesian Optimization builds a probabilistic model to predict the performance of different hyperparameter settings, enabling it to focus on the most promising areas of the space.

Mathematical Foundation: Bayesian Optimization uses a surrogate model to approximate the objective function $f(\mathbf{x})$, where $\mathbf{x}$ represents a set of hyperparameters. Commonly, a Gaussian Process (GP) is used as the surrogate model due to its flexibility and ability to quantify uncertainty.

1. **Define the Objective Function**: Let $f(\mathbf{x})$ represent the objective function we wish to minimize (or maximize), such as validation error or negative accuracy. Since evaluating $f(\mathbf{x})$ directly can be time-consuming, we build an approximation using a GP.

2. **Gaussian Process (GP)**: A GP models $f(\mathbf{x})$ as a collection of random variables, any finite number of which have a joint Gaussian distribution. Given observed data $\{\mathbf{x}_i, y_i\}$, where $y_i = f(\mathbf{x}_i)$, the GP can provide a prediction $\hat{f}(\mathbf{x})$ and an uncertainty measure at any new point $\mathbf{x}$.

3. **Acquisition Function**: To determine the next hyperparameter setting to evaluate, Bayesian Optimization uses an acquisition function $a(\mathbf{x})$, which balances exploration (trying less certain areas) and exploitation (focusing on areas with expected high performance). Common acquisition functions include: - **Expected Improvement (EI)**: Measures the expected improvement over the best observed value. - **Upper Confidence Bound (UCB)**: Selects points with high predicted mean and uncertainty.

4. **Optimization Process**: At each iteration, Bayesian Optimization selects the hyperparameter setting $\mathbf{x}_{\text{next}}$ that maximizes the acquisition function:

$$\mathbf{x}_{\text{next}} = \arg\max_{\mathbf{x}} a(\mathbf{x})$$

The true objective function $f(\mathbf{x}_{\text{next}})$ is then evaluated, and the surrogate model is updated with this new data. This process repeats until a stopping criterion is met, such as reaching a maximum number of evaluations.

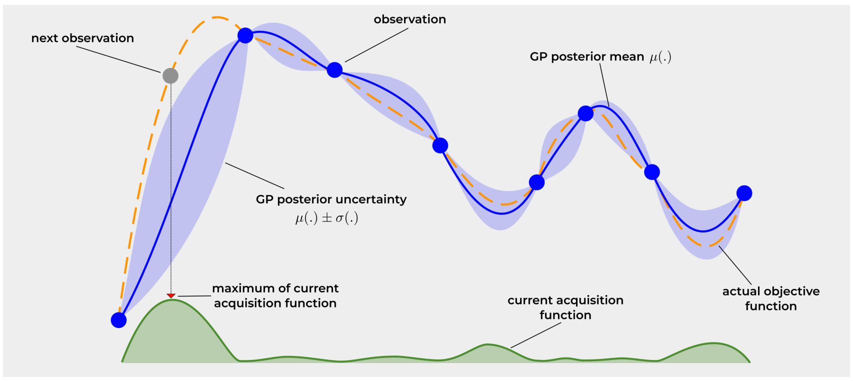

next observation
observation
GP posterior mean $\mu(.)$
GP posterior uncertainty
$\mu(.) \pm \sigma(.)$
maximum of current acquisition function
current acquisition function
actual objective function

Figure 19: Baynesian optimization method

Advantages of Bayesian Optimization: - **Sample Efficiency**: By focusing on promising areas, Bayesian Optimization can find optimal hyperparameters with fewer evaluations compared to grid or random search. - **Handling Complex Search Spaces**: Bayesian Optimization is effective for high-dimensional, non-convex, and costly objective functions.

Conclusion: Bayesian Optimization is a powerful and sample-efficient method for hyperparameter tuning, leveraging a probabilistic model to make informed choices about where to sample next. This approach is particularly useful in optimizing complex models where evaluating each hyperparameter configuration is computationally expensive.[2]

III Testing and Evaluating Model

1 Random forest

In this part, I will using normal random forest with no tuning parameter and use training data begin in 1945 and testing data begin in 2000.

```
split_data(data=training_df, train_start_year=1945, test_start_year=2000)
RandomForestRegressor(n_estimators=100, random_state=42)
```

After predicting I have this result

Metric	Error Value
Mean Absolute Error for LAT	4.926282084072047
Mean Absolute Error for LON	6.462165664807102
Mean Haversine Distance Error	922.673981090901

Table 4: Summary of Error Metrics for LAT, LON, and Haversine Distance of normal Random forest

The large error, with an average distance error of over 900 km, shows that the model's predictions are quite far from the true locations. This big gap means the model is not capturing the locations accurately, and it may need changes in the data, model settings, or adjustments to improve accuracy.

2 Random forest with tuned hyperparameters

The tuning process employed Bayesian Optimization with the `bayes_opt` library, focusing on four key hyperparameters of the Random Forest model:

- **n_estimators**: Number of trees in the forest, bounded between 50 and 300.
- **max_depth**: Maximum depth of each tree, ranging from 5 to 30.
- **min_samples_split**: Minimum number of samples required to split a node, set between 2 and 10.
- **min_samples_leaf**: Minimum number of samples required at a leaf node, set between 1 and 10.

The optimization function (`rf_evaluate`) calculates the **Mean Haversine Distance** on a K-fold split dataset. For each iteration, the function:

1. Trains the Random Forest model on training folds.
2. Predicts latitude and longitude on the test fold.
3. Calculates the Mean Haversine Distance between predicted and actual values.
4. Updates the optimization loop with the lowest observed Haversine distance.

After **10 iterations** with **2 initial random points**, the following optimal hyperparameters were found:

- **n_estimators**: 296
- **max_depth**: 30
- **min_samples_split**: 10
- **min_samples_leaf**: 10

The table below summarizes the evaluation metrics for the model across different validation methods and the test set:

Validation Method	**Mean Haversine Distance**	**MAE for LAT**	**MAE for LON**
K-Fold Split (Training Set)	867.0986	4.4507	6.2580
Walk Forward Split (Training Set)	1078.7983	5.9895	7.6256
Group Time Series Split with Gap (Training Set)	1017.1114	5.2332	7.3101
Test Set	875.2431	4.6623	6.1429

Table 5: Summary of Evaluation Metrics for Different Validation Splits and Test Set of Random forest with tuned hyperparameters

The new results consistently outperform the previous results across all metrics, indicating that the model has improved in predicting both latitude and longitude, as well as in reducing the overall geographic error measured by the Haversine distance.

3 XGBoost with tuned parameters

The tuning process focused on nine key hyperparameters of the XGBoost model:

1. **learning_rate**: The rate at which the model learns, controlling the contribution of each tree in the ensemble.
2. **max_depth**: The maximum depth of each tree, controlling the model complexity.
3. **n_estimators**: The number of boosting rounds or trees to build.
4. **min_child_weight**: The minimum sum of instance weight needed in a child node, used to control overfitting.
5. **subsample**: The fraction of samples used for training each tree.
6. **colsample_bytree**: The fraction of features used per tree.
7. **gamma**: Minimum loss reduction required to make a further partition on a leaf node, helping to control the complexity of the model.
8. **reg_alpha**: L1 regularization term on weights, useful to avoid overfitting.
9. **reg_lambda**: L2 regularization term on weights, also useful to avoid overfitting.

The optimization function `xgb_evaluate` calculates the Mean Haversine Distance for each hyperparameter configuration using a k-fold cross-validation approach. In each iteration:

1. The model is trained on the training folds.
2. Predictions are made on the test fold.
3. The Mean Haversine Distance between the predicted and actual coordinates is computed.

The Bayesian Optimization process ran for **50 iterations**, with an additional **2 initial random points** to explore the hyperparameter space.

After completing the optimization, the following hyperparameters were identified as optimal:

- **learning_rate**: 0.1136
- **max_depth**: 9
- **n_estimators**: 85
- **min_child_weight**: 2

- **subsample**: 0.9166
- **colsample_bytree**: 0.6802
- **gamma**: 18.12
- **reg_alpha**: 0.0
- **reg_lambda**: 18.99

The chosen hyperparameters reflect a balanced model that limits overfitting while capturing complex patterns:

- The **learning_rate** of 0.1136 is relatively low, allowing the model to learn gradually.
- A **max_depth** of 9 enables the model to capture non-linear relationships without making it overly complex.
- The **n_estimators** count of 85 and **subsample** of 0.9166 indicate a moderately-sized ensemble with a high sample usage rate, allowing each tree to be built with robust data coverage.
- **min_child_weight** of 2, **gamma** of 18.12, and **reg_lambda** of 18.99 add significant regularization, reducing the risk of overfitting on the training data.
- The **colsample_bytree** of 0.6802 restricts the number of features per tree, increasing feature diversity across trees.

The table below summarizes the evaluation metrics for the model across different validation methods and the test set:

Validation Method	**Mean Haversine Distance**	**MAE for LAT**	**MAE for LON**
K-Fold Split (Training Set)	64.2451	0.1810	0.5677
Walk Forward Split (Training Set)	64.2451	0.1810	0.5677
Group Time Series Split with Gap (Training Set)	137.9375	0.3751	1.2256
Test Set	64.2451	0.1810	0.5677

Table 6: Summary of Evaluation Metrics for Different Validation Splits and Test Set

The result is improved dramatically with using XGBoost, regulation and tuning hyperparameter.

4 Visualize results

In this part, I will visualize some storm trajectory of final model.

I will take random storm and visualize it using mathplotlib also calculate MHD of that storm.

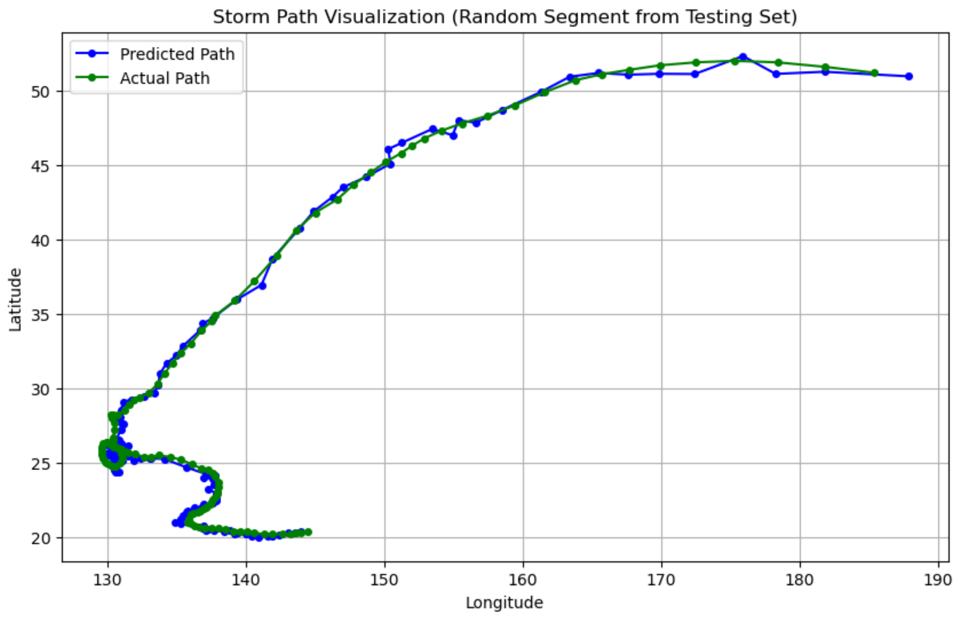


Figure 20: Storm trajectory result 1

Haversine Distance Error (Mean) for Selected Storm Path Segment: 55.80 km

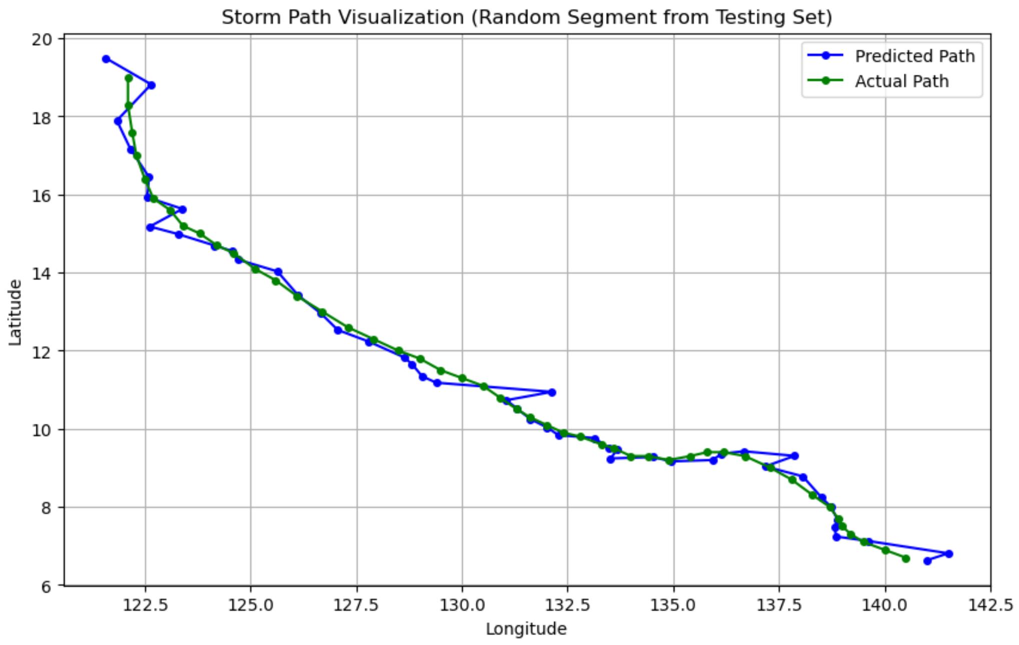


Figure 21: Storm trajectory result 2

Haversine Distance Error (Mean) for Selected Storm Path Segment: 37.35 km

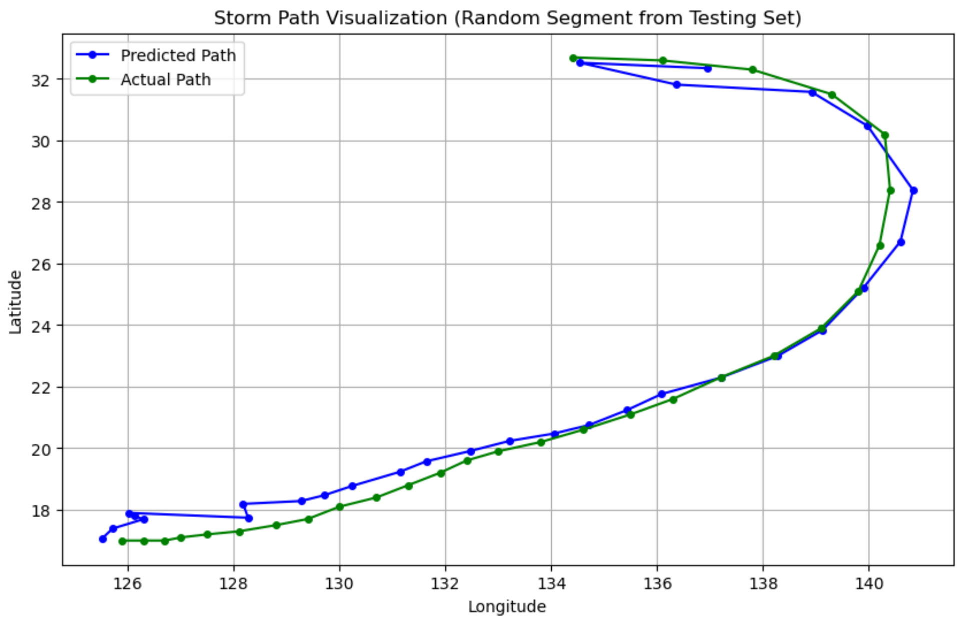


Figure 22: Storm trajectory result 3

Haversine Distance Error (Mean) for Selected Storm Path Segment: 63.87 km

IV Conclusion

The test set results show a strong accuracy level and alignment with the data splitting techniques employed in training, validating the model's ability to accurately predict locations. Nevertheless, the small difference seen in the Group Time Series Split indicates that additional thought may be needed for tasks that demand great time accuracy. In general, this research establishes a strong basis for integrating the model in real-world situations and improving the predictive performance of the system.

References

[1] C.-C. Wei and C.-C. Hsu, “Extreme gradient boosting model for rain retrieval using radar reflectivity from various elevation angles,” *Remote Sensing*, vol. 12, no. 14, p. 2203, 2020.

[2] P. I. Frazier, “A tutorial on bayesian optimization,” *arXiv preprint arXiv:1807.02811*, 2018.