

WEATHERSENSE PROJECT

WEATHER SENSE

Presentation, April 2024

PRESENTATION

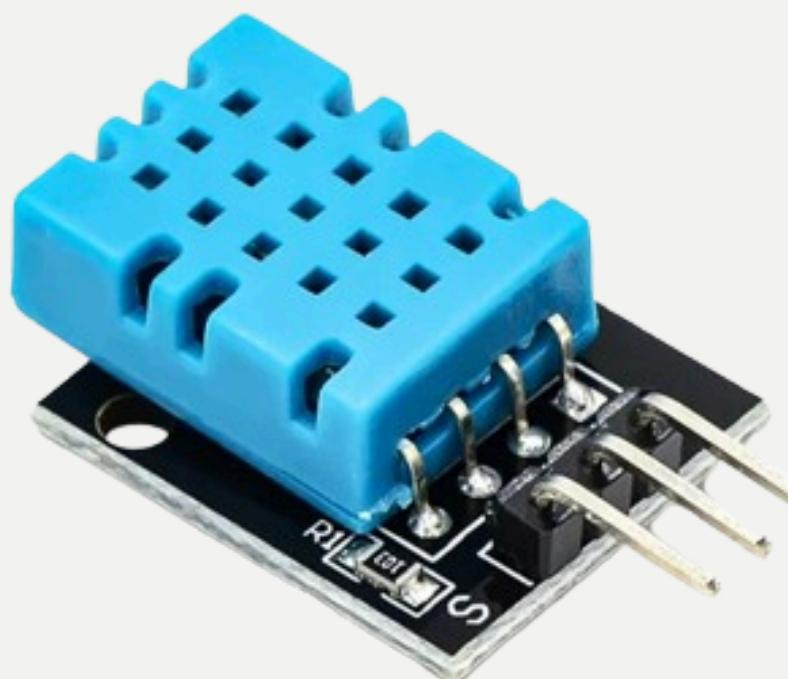


About Project

WeatherSense is a weather tracking website that updates weather data and predictions every 10 minutes. Leveraging our trained random forest model, WeatherSense provides accurate forecasts and historical weather information.

PRIMARY DATA

OUR PRIMARY DATA IS COLLECTED BY THE KIDBRIGHT BOARD, INCORPORATING A TEMPERATURE AND HUMIDITY SENSOR FROM THE KY-015 MODULE, AND USING MQTT TO SEND A DATA TO NODERED



SECONDARY DATA

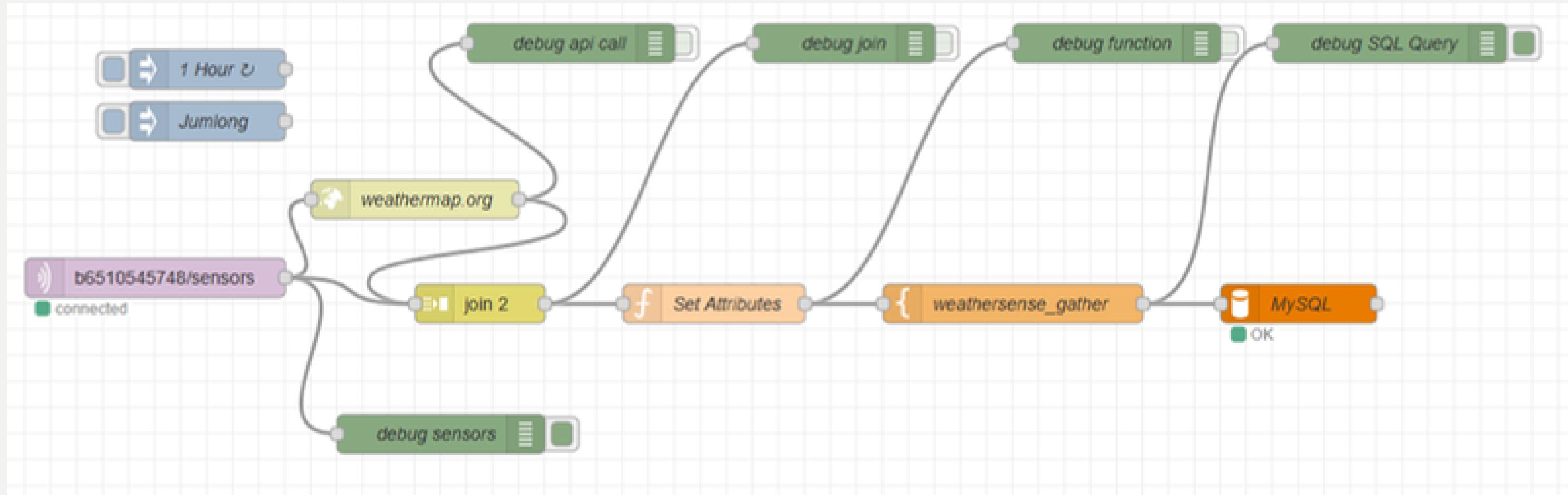
OUR SECONDARY DATA IS COLLECTED BY CALL API FROM OPENWEATHERMAP API
(CURRENT WEATHER) USING NODERED TO FETCH DATA FROM API

ATTRIBUTE SELECT:

- HUMIDITY (%)
- TEMPERATURE (DEGREE CELSIUS)
- PRESSURE (HPA)
- CLOUDINESS (%)
- WEATHER: SUCH AS CLOUD, FEW CLOUDS ETC.



NODE RED IMPLEMENTATION



DATA EXPLORLATION

USING PYTHON PANDAS TO EXPLORE AND PREPROCESS MORE
OVER TRAINING A MODEL.



DATA EXPLORATION

CHECKING DATA TYPE OF DATAFRAME

```
id                  int64
ts                  object
temp_sensor         int64
humidity_sensor    int64
temp_api           float64
humidity_api       int64
pressure           int64
wind_speed          float64
cloudiness          int64
weather             object
dtype: object
```

DATA EXPLORATION

CHANGE TYPE OF TS TO DATETIME TYPE

```
1 data['ts'] = pd.to_datetime(data['ts'])  
2 display(data.head())
```

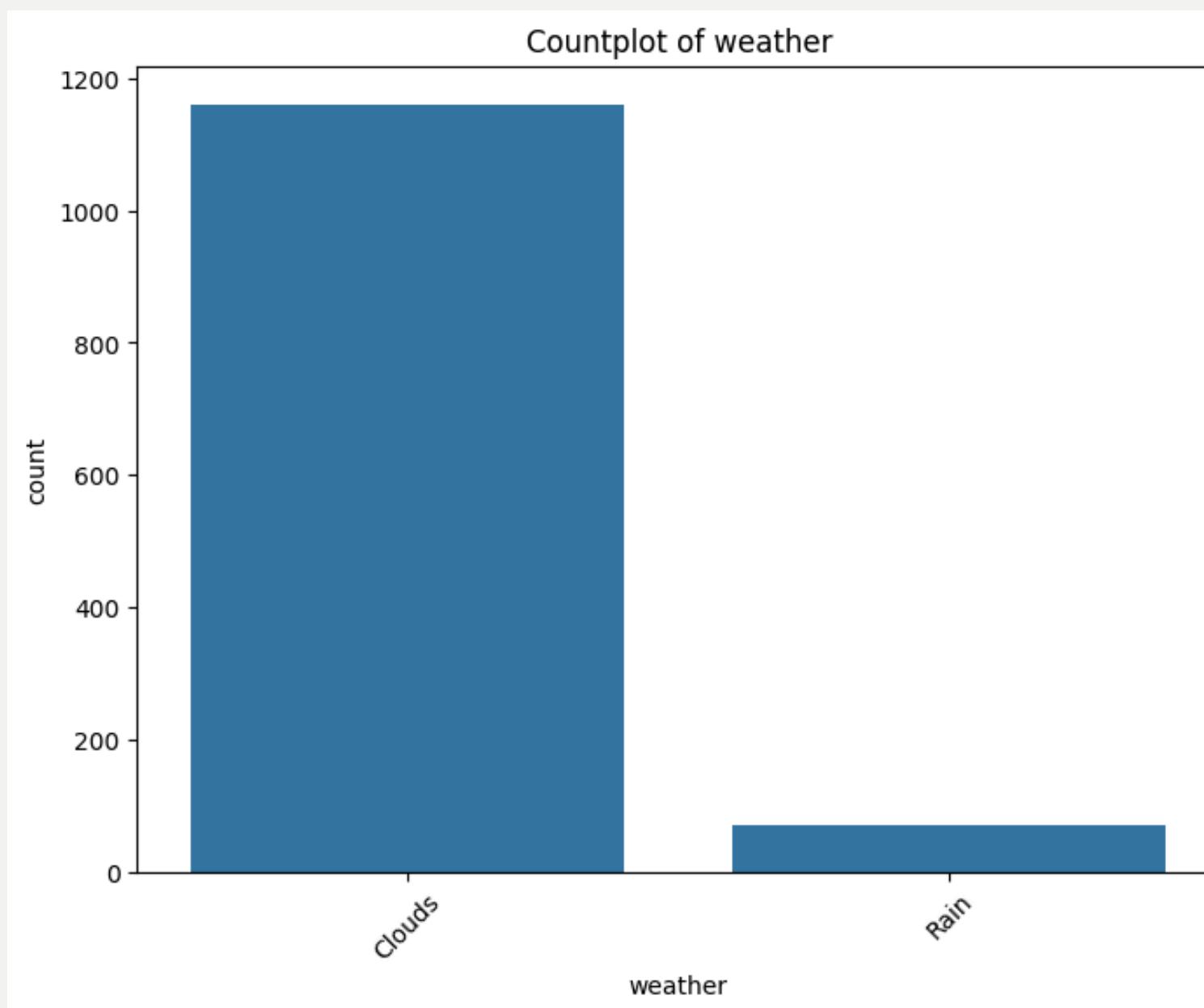
DATA EXPLORATION

- RETRIEVE A SUMMARY STATISTIC OF THIS DATASET.

	id		ts	temp_sensor	humidity_sensor	temp_api	humidity_api	pressure	wind_speed	cloudiness	
count	1232.000000		1232	1232.000000	1232.000000	1232.000000	1232.000000	1232.000000	1232.000000	1232.000000	
mean	616.500000	2024-04-28 05:42:56.026785792		33.039773	71.299513	33.911964	64.646916	1005.815747	4.595649	21.363636	
min	1.000000		2024-04-21 17:34:59	22.000000	26.000000	28.860000	26.000000	1000.000000	1.030000	20.000000	
25%	308.750000	2024-04-24 20:33:47.750000128		33.000000	68.000000	31.130000	52.000000	1005.000000	3.600000	20.000000	
50%	616.500000		2024-04-27 14:51:34	33.000000	73.000000	32.690000	70.000000	1006.000000	4.630000	20.000000	
75%	924.250000		2024-05-01 18:49:37.500000	34.000000	77.000000	37.190000	78.000000	1007.000000	5.140000	20.000000	
max	1232.000000		2024-05-07 02:21:56	35.000000	95.000000	41.220000	90.000000	1010.000000	8.230000	40.000000	
std	355.792074			NaN	1.904941	7.383676	3.152121	15.168092	1.842096	1.232721	5.043200

DATA EXPLORATION

- FOR CATEGORICAL FEATURES USING COUNT PLOTS.

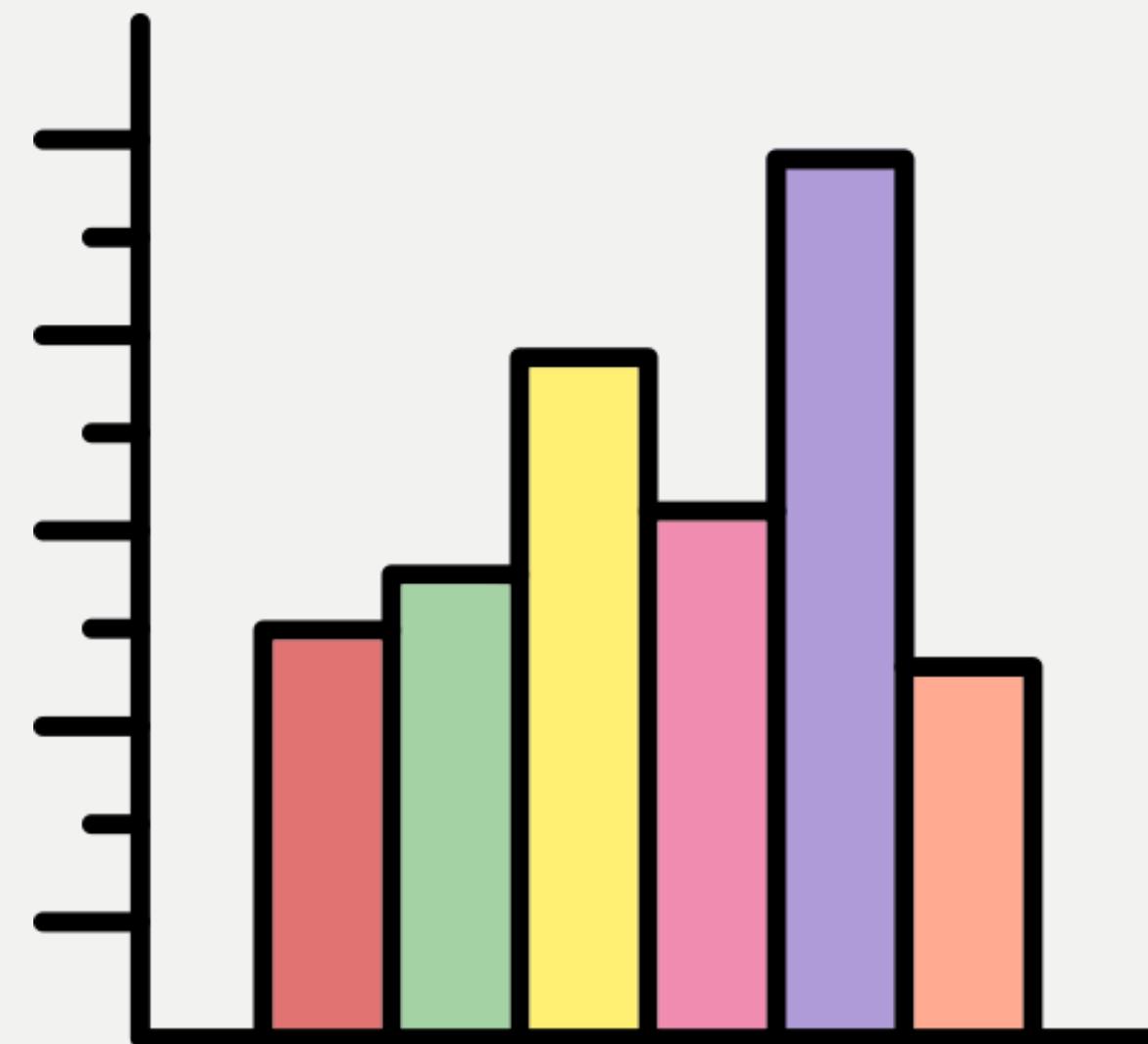


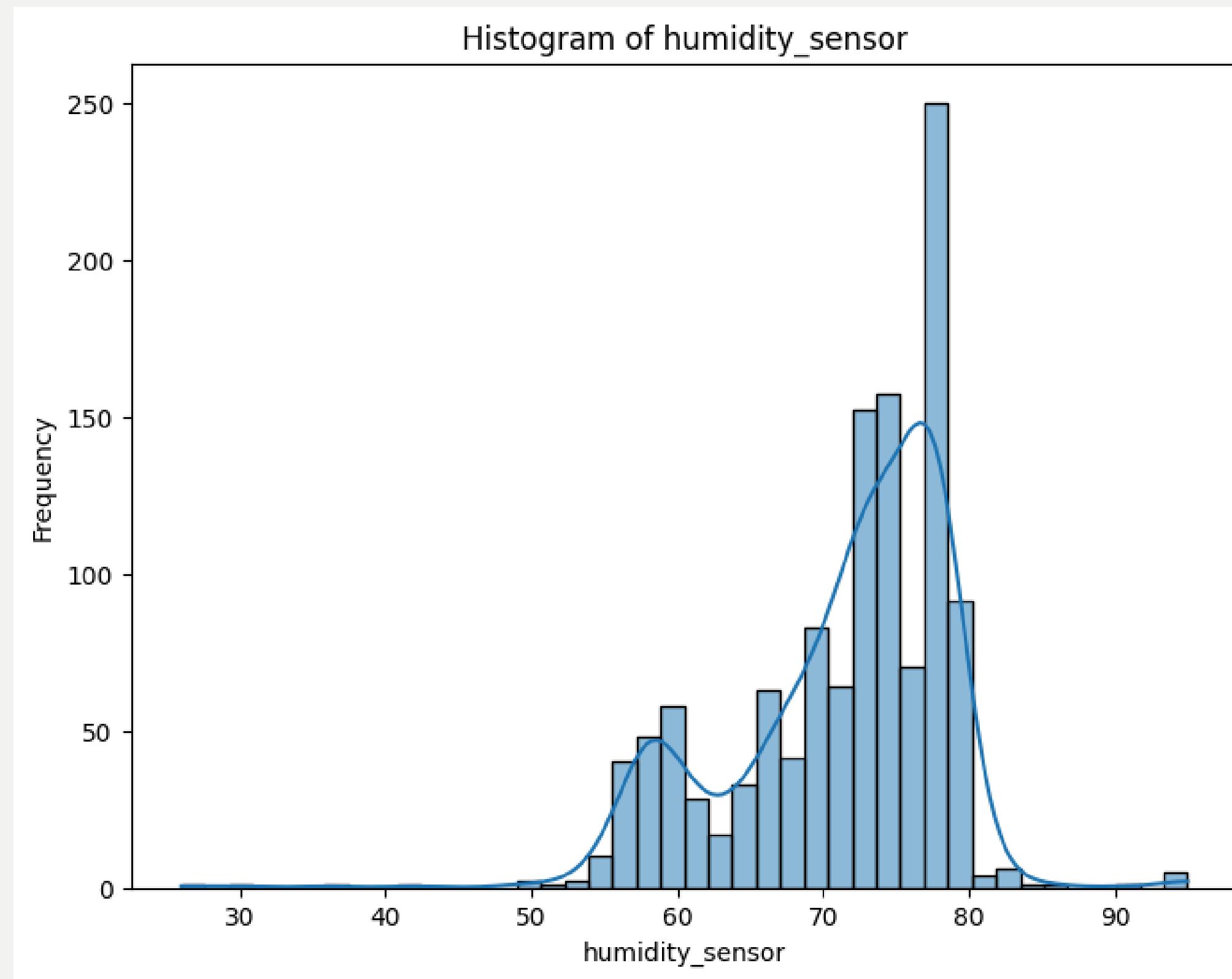
From the histogram plot, it's evident that the data clouds contain a large number of values, whereas instances of rain are scarce. This imbalance can lead to a data problem, which we need to address during the preprocessing stage.

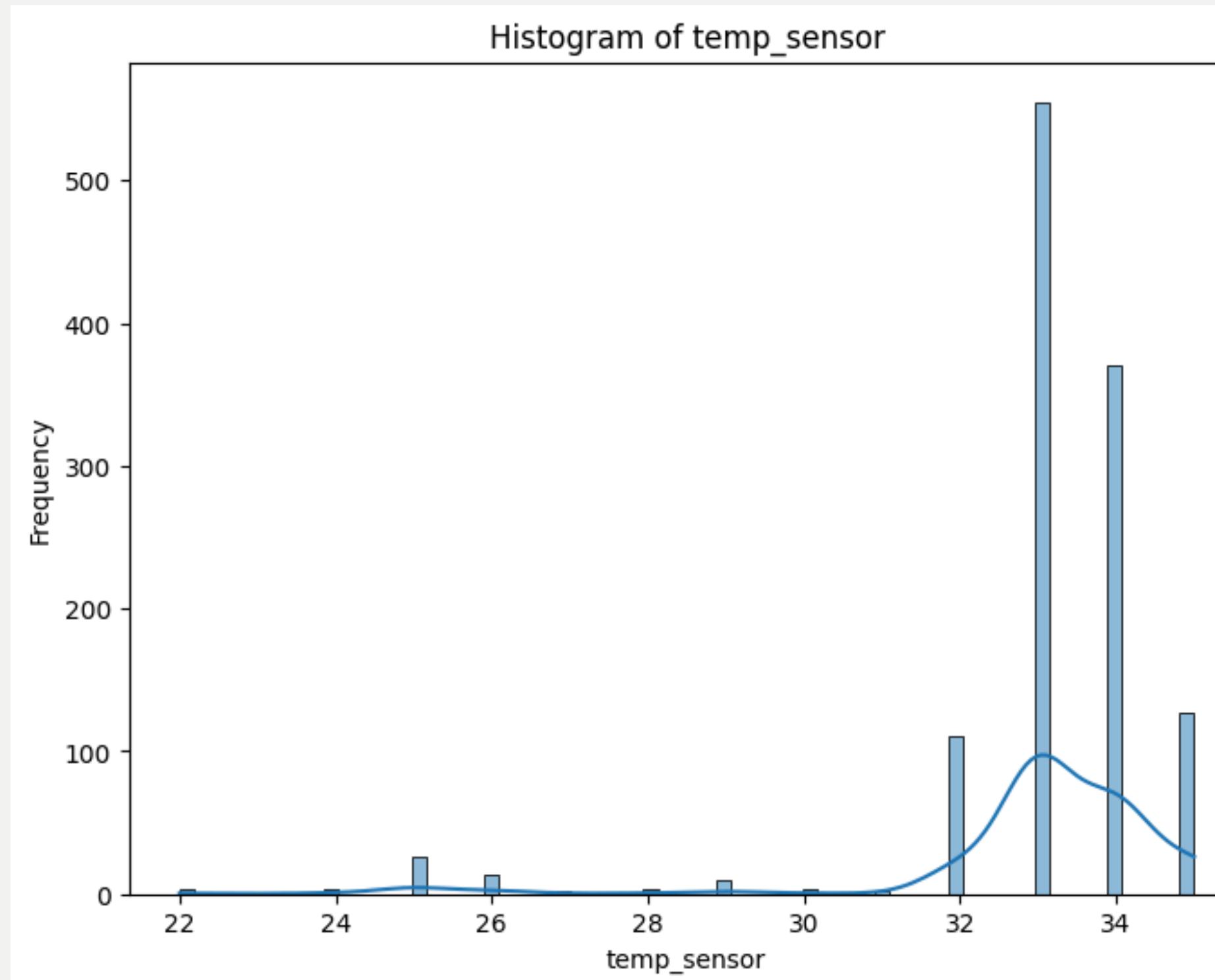


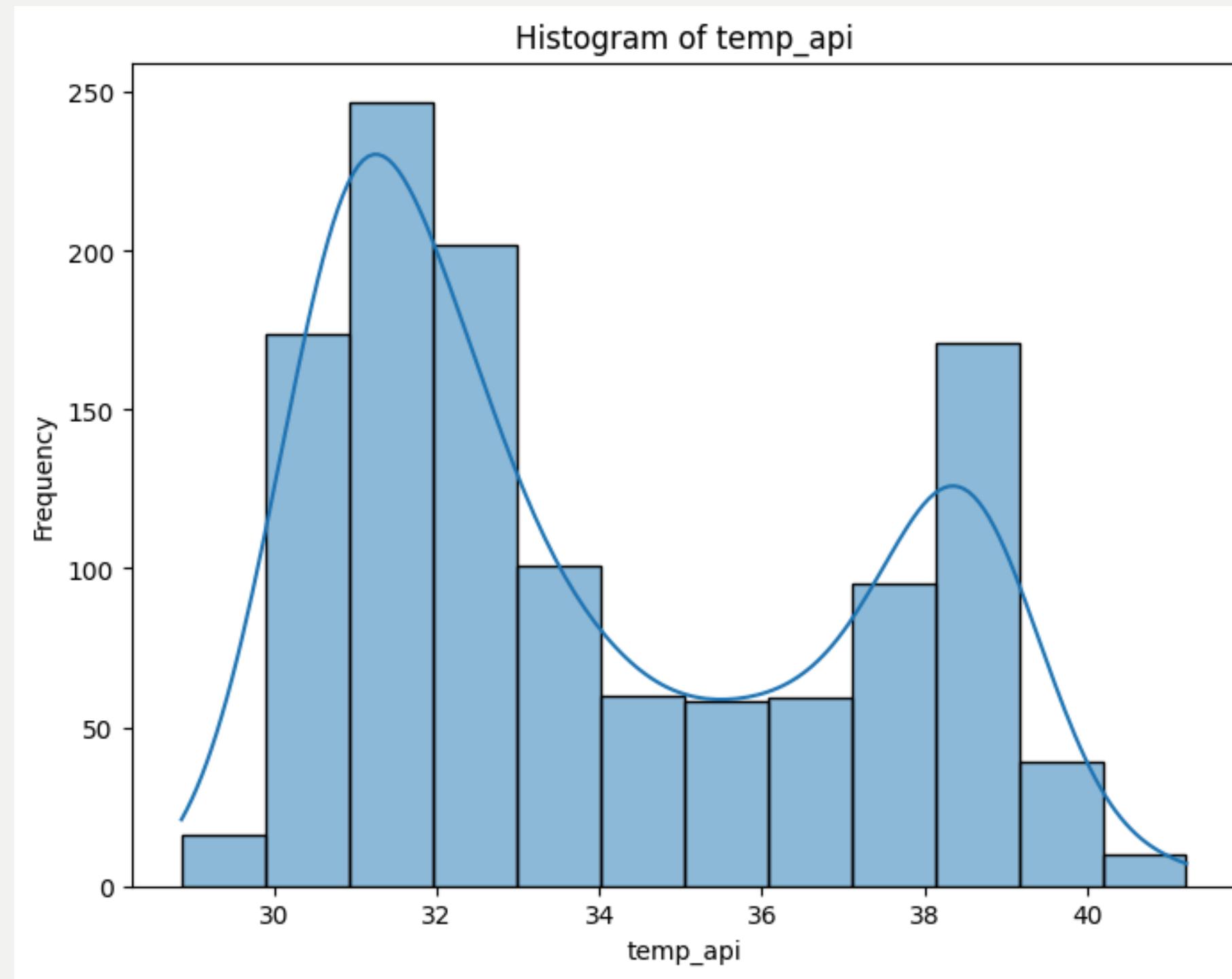
DATA EXPLORATION

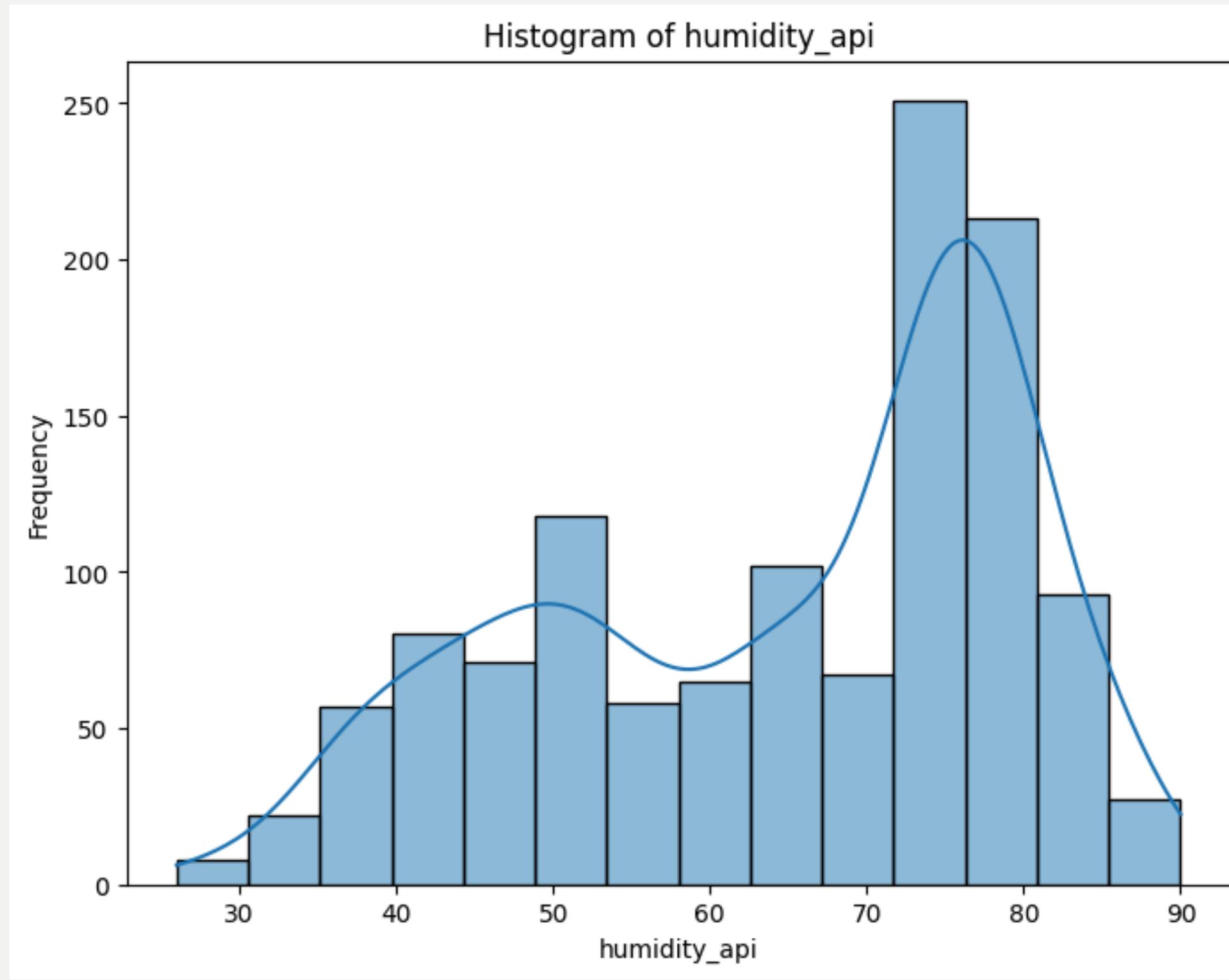
OR NUMERICAL FEATURES USING HISTOGRAM PLOT TO FIND DISTRIBUTION.

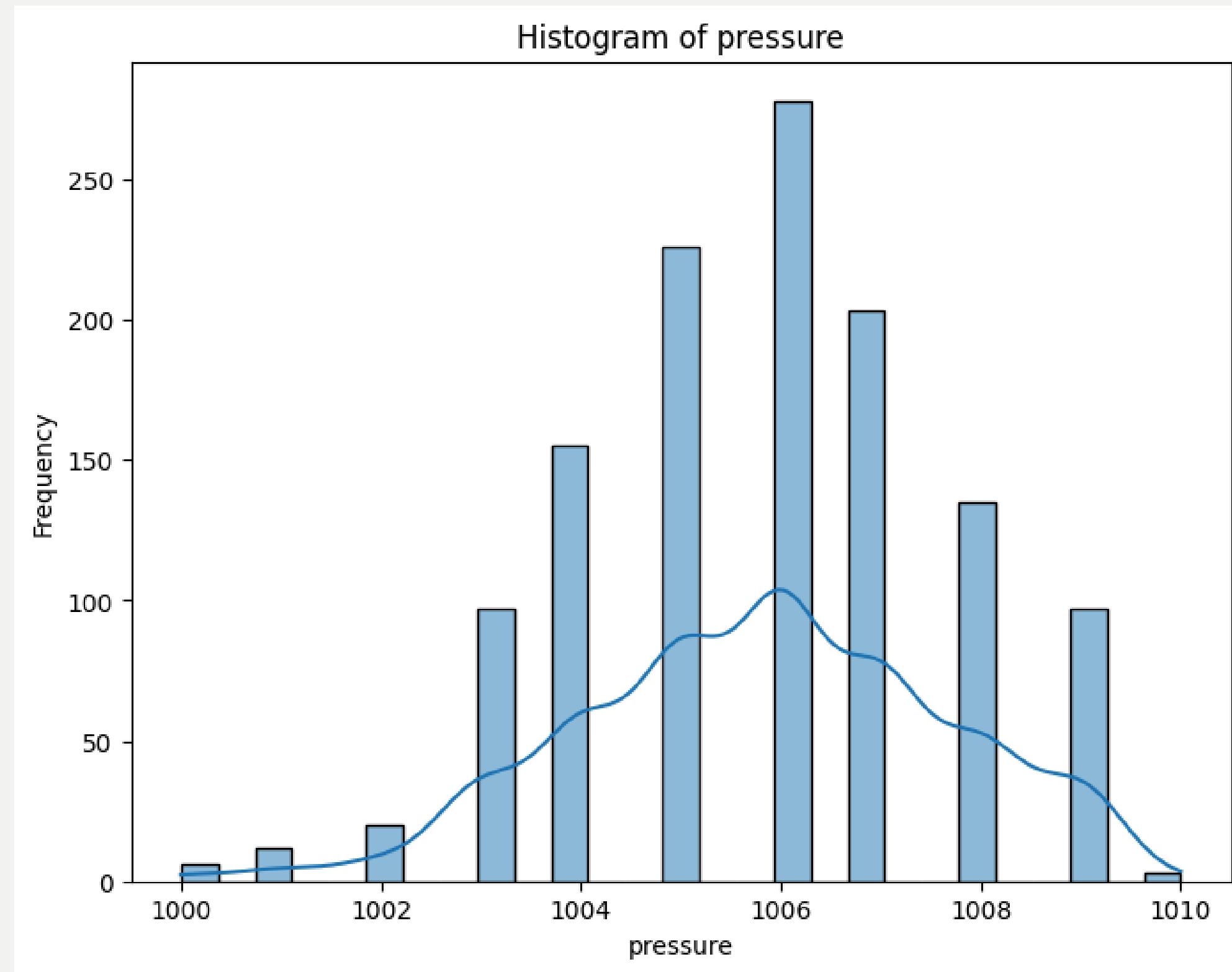


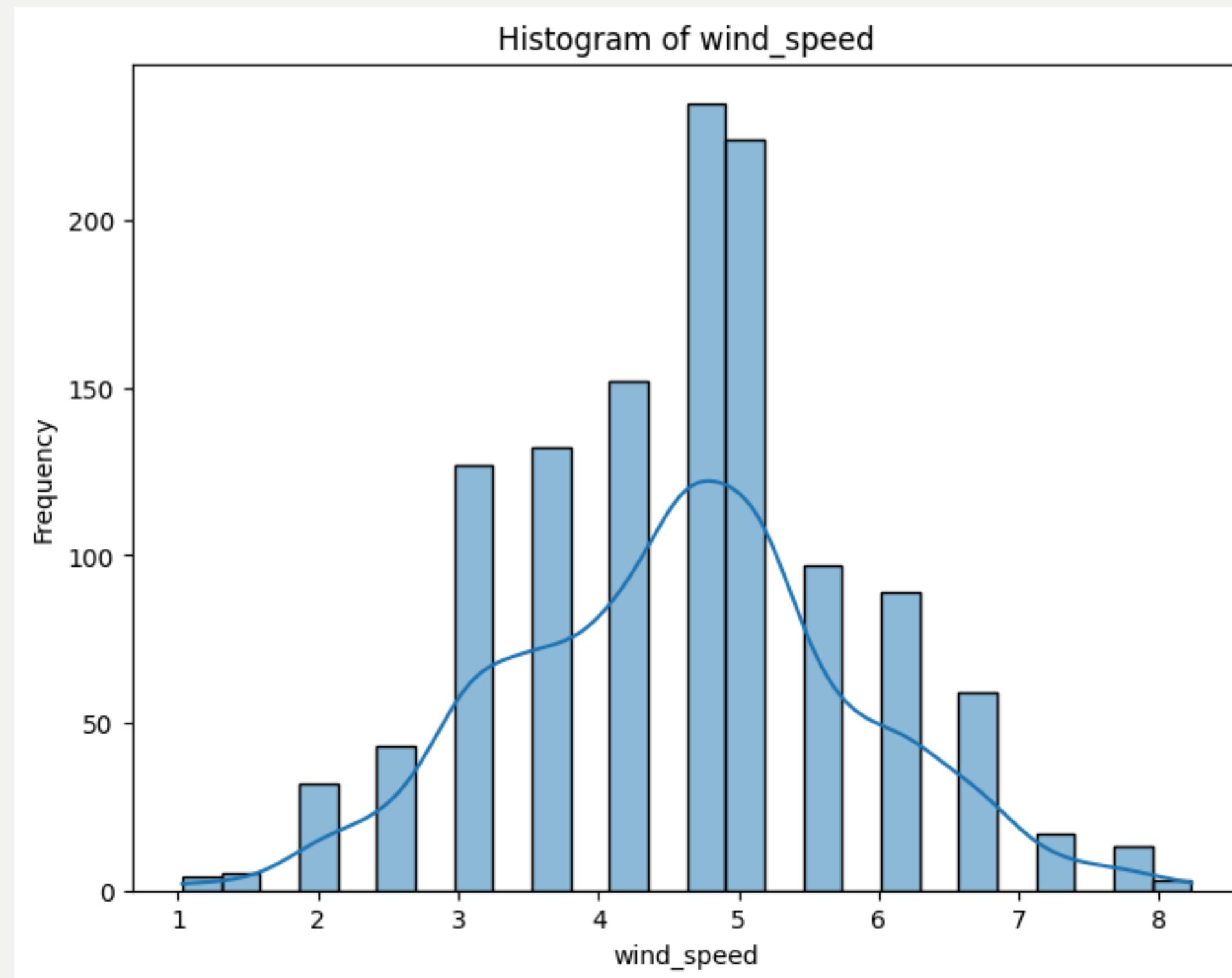


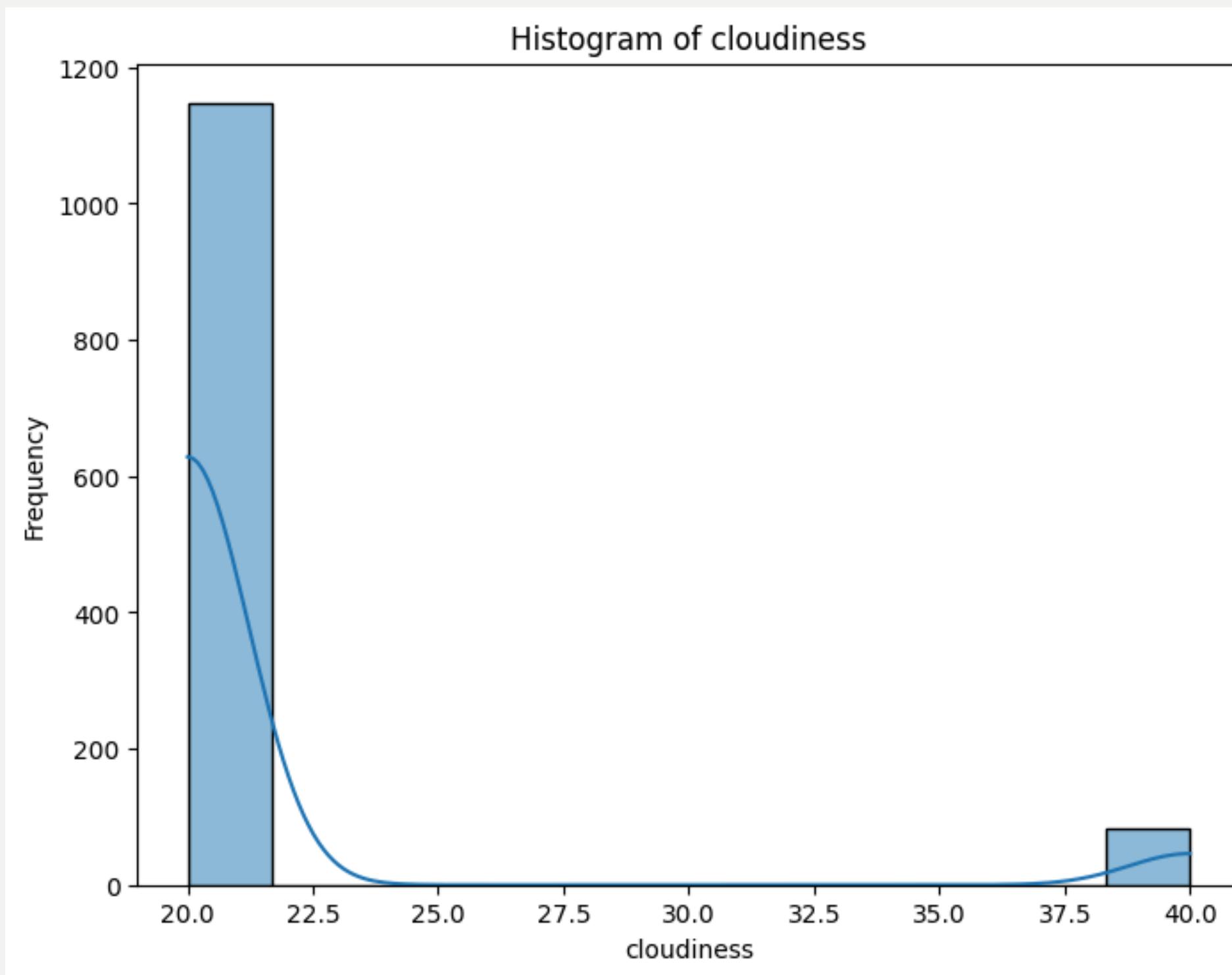






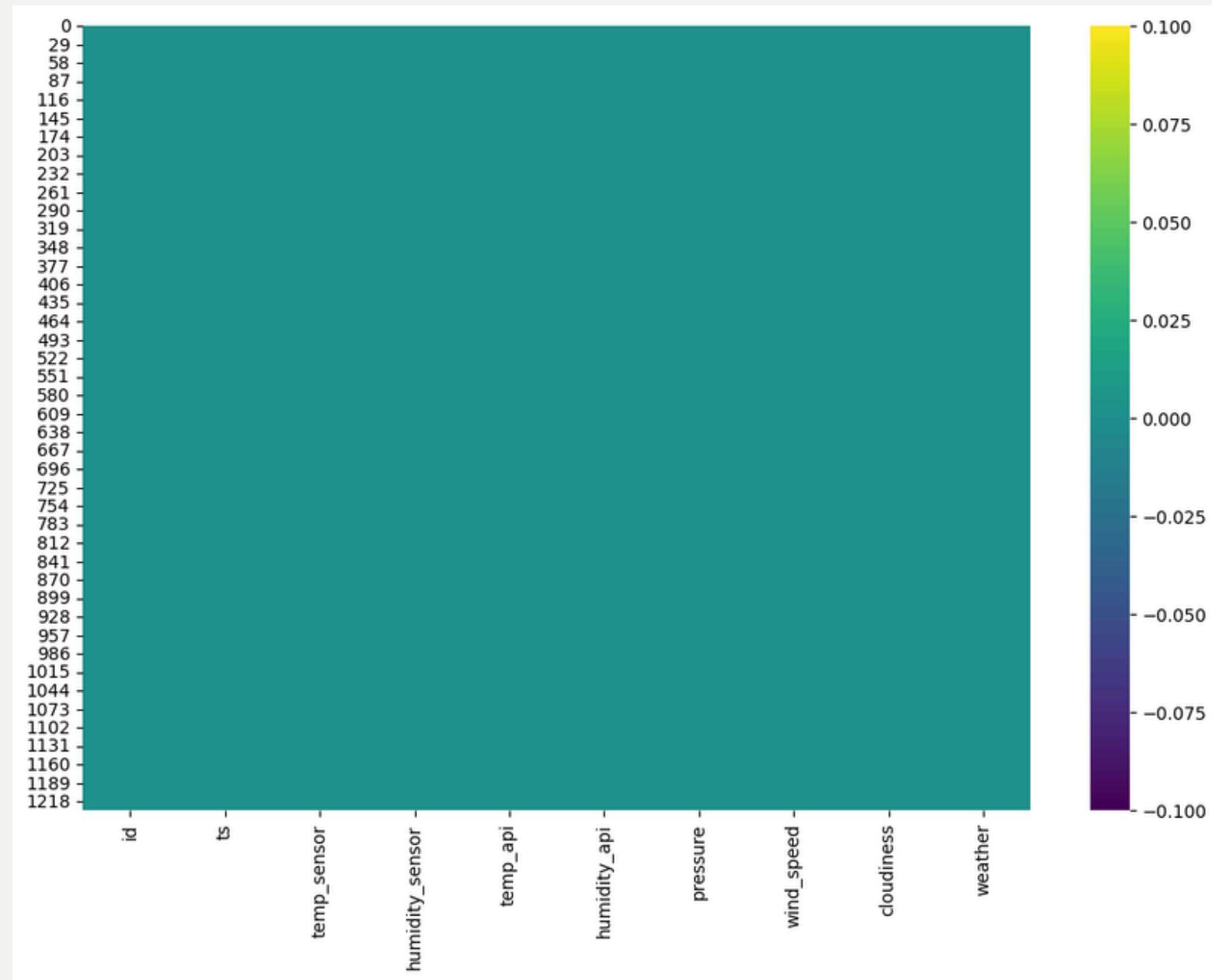






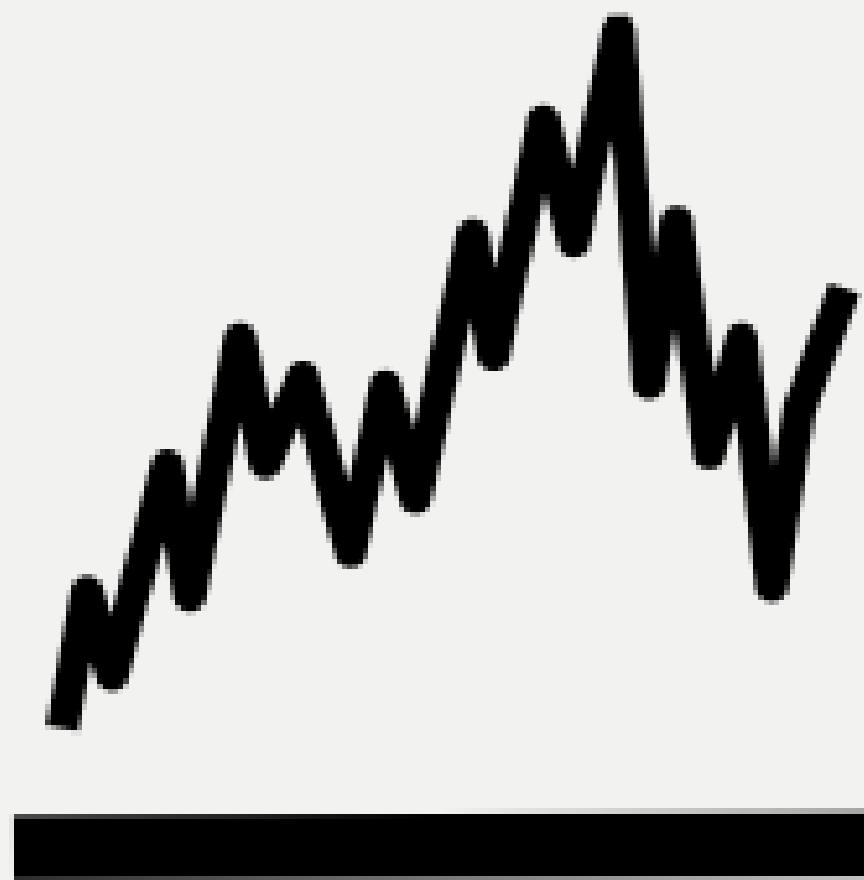
DATA EXPLORATION

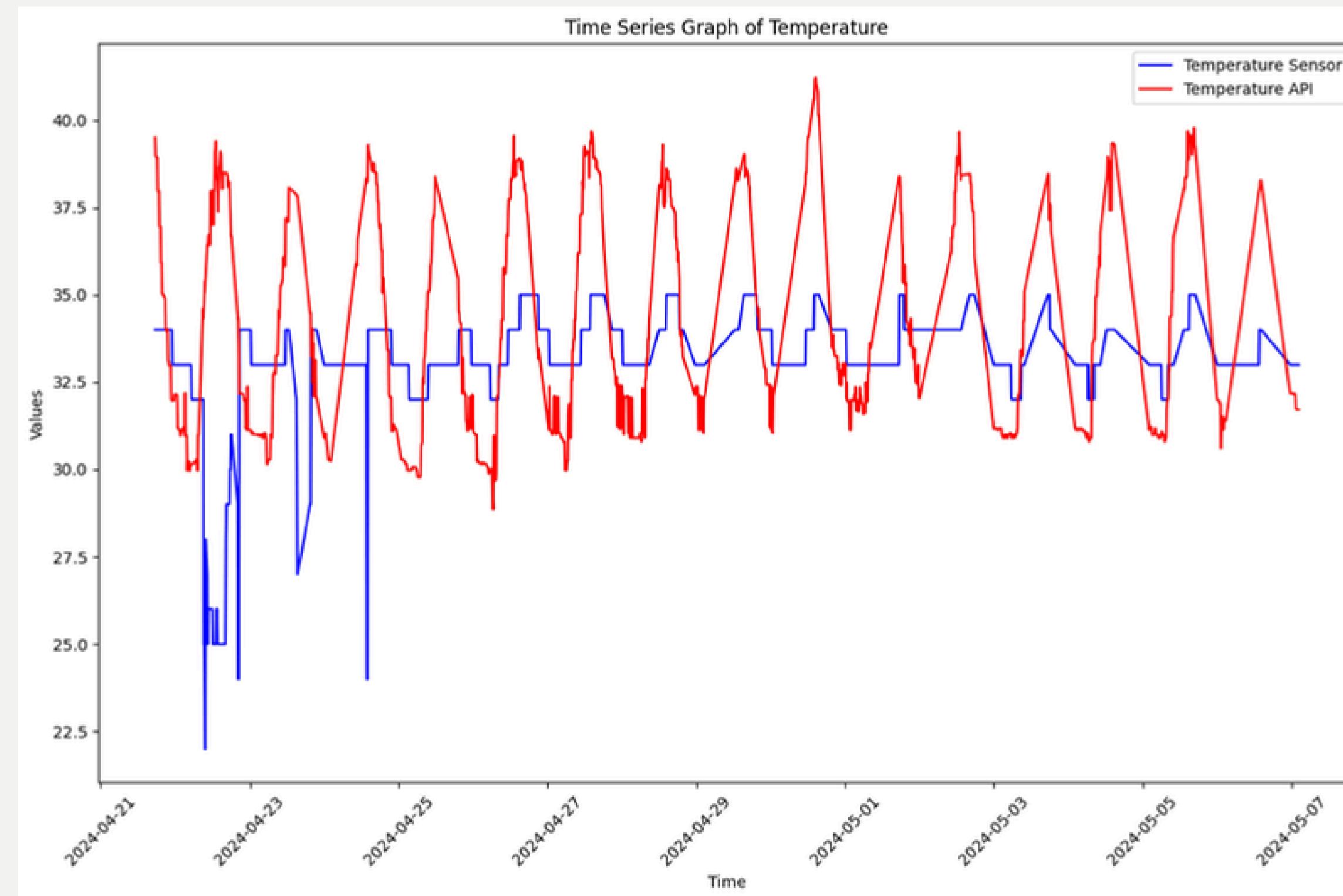
USE HEATMAP TO FIND MISSING DATA.

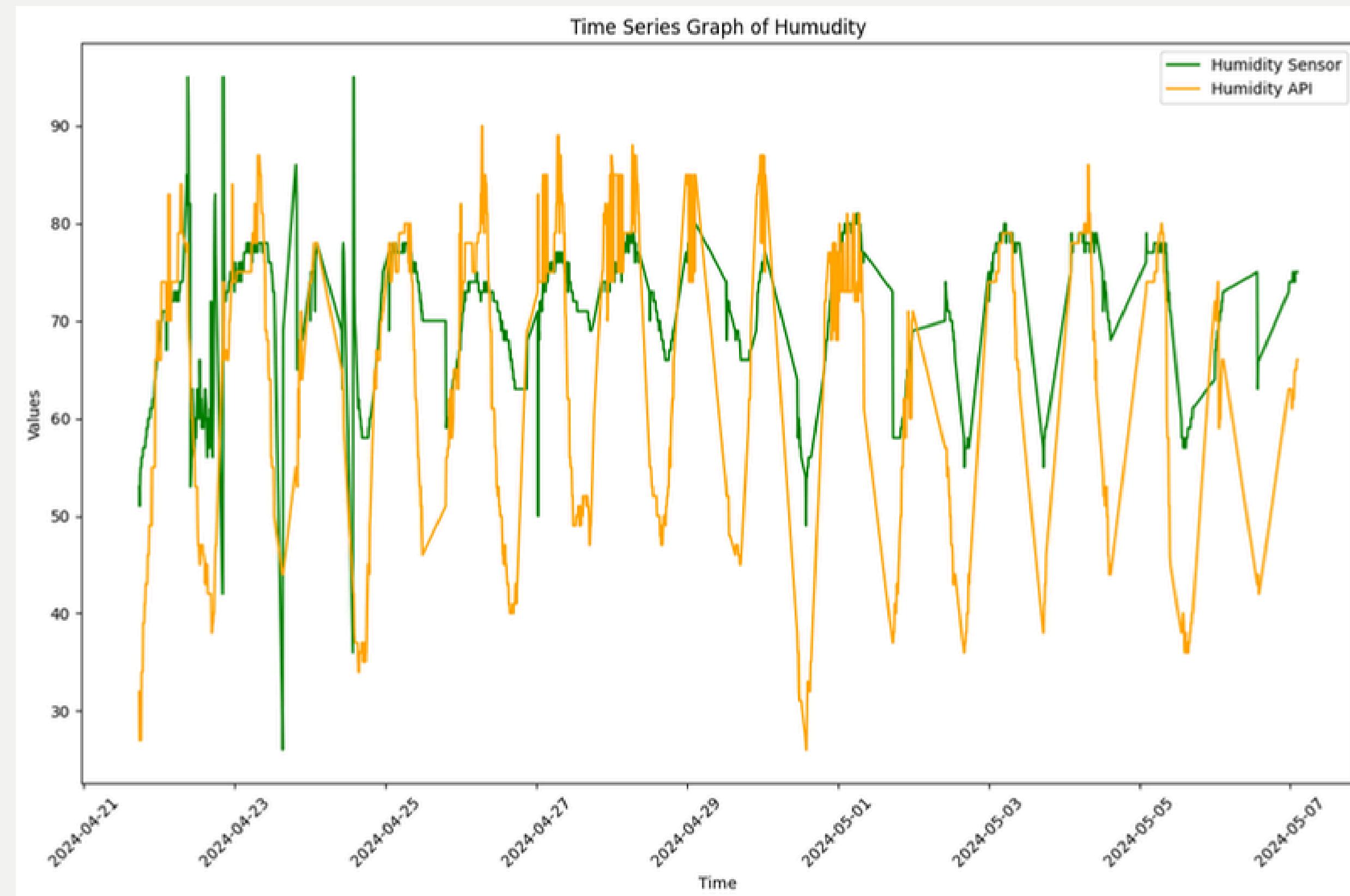


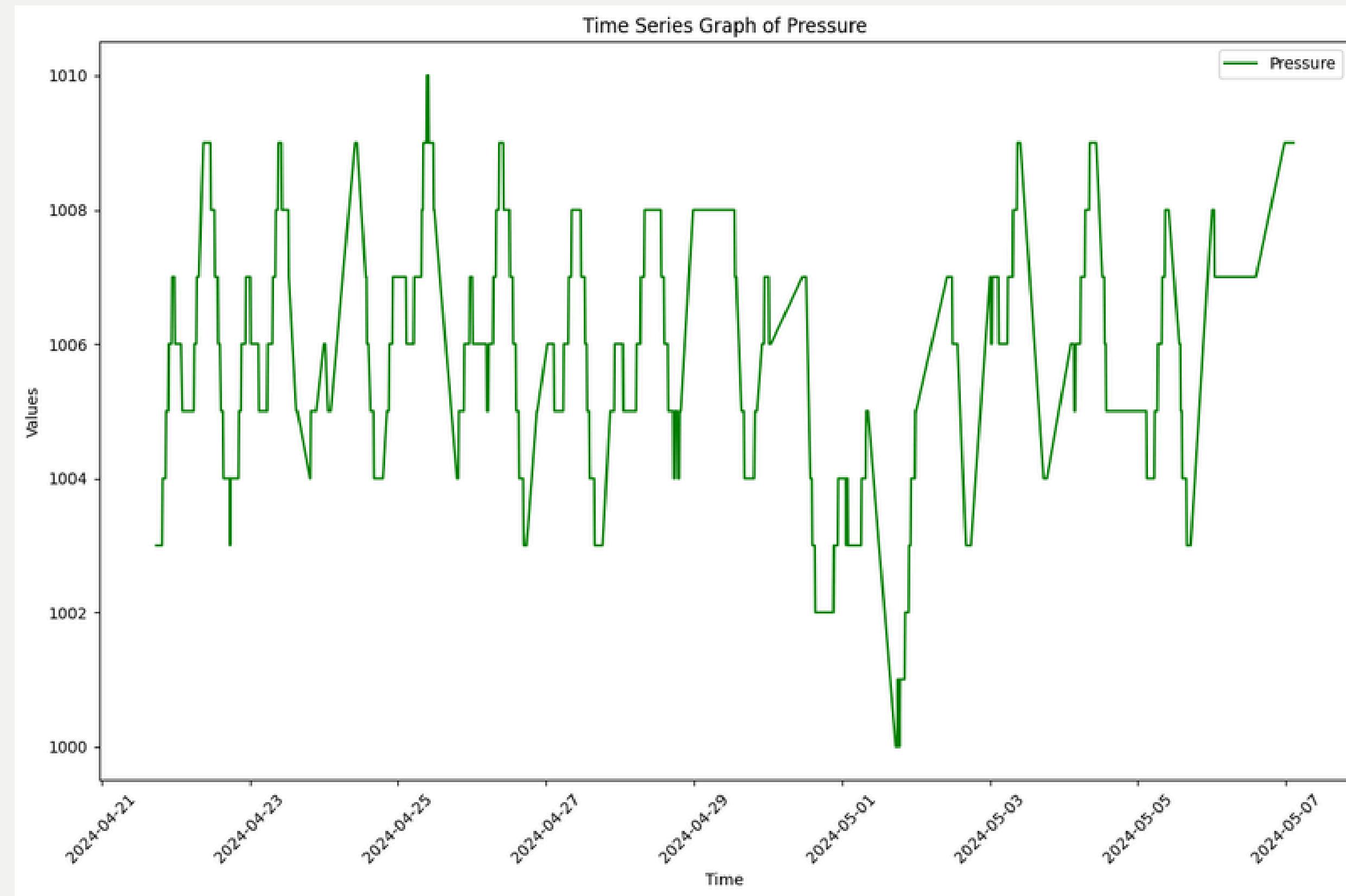
DATA EXPLORATION

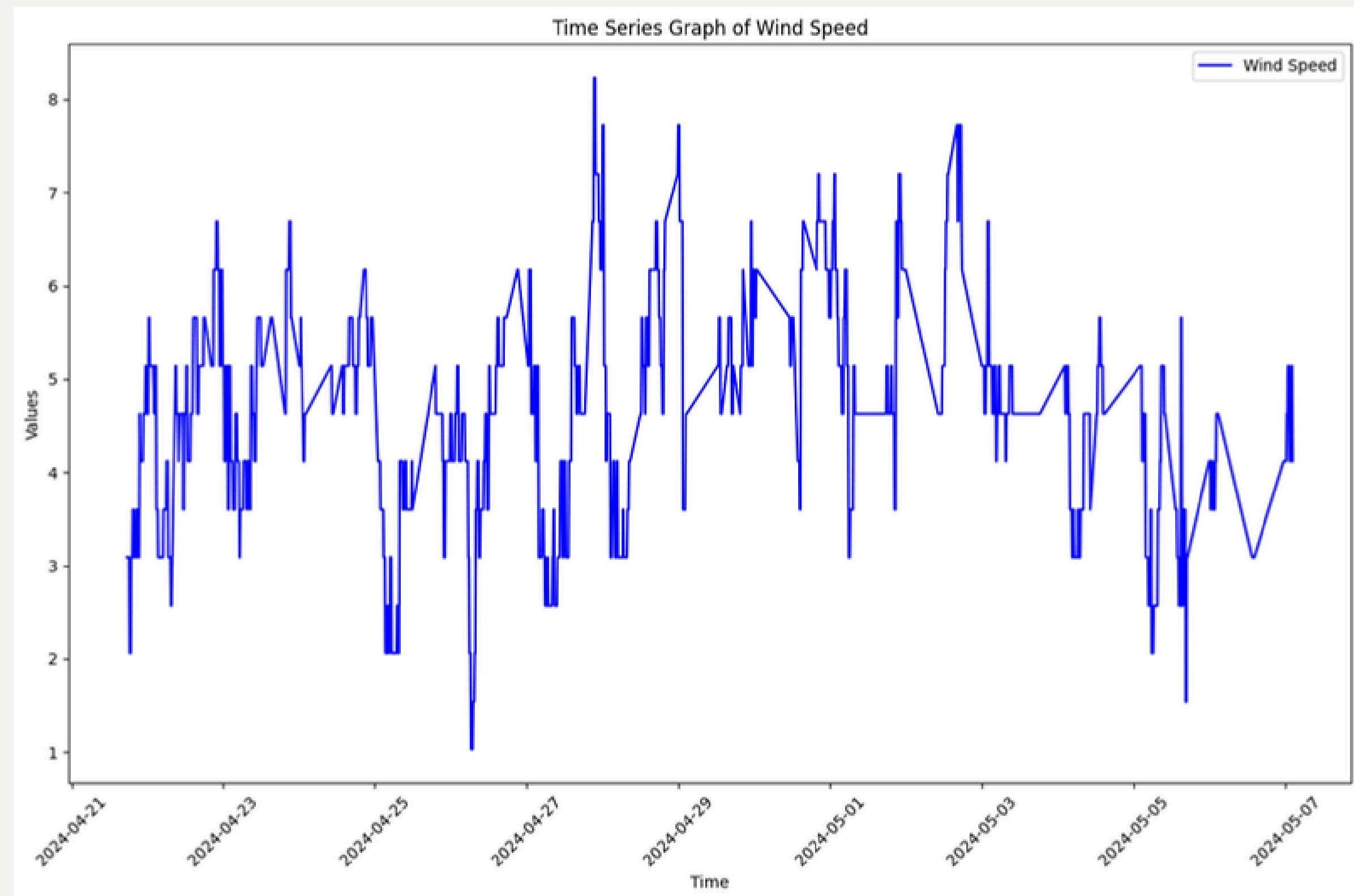
PLOT TIME SERIES GRAPH AMONG NUMERICAL FEATURES TO
FIND A TREND OF DATA IN THE FUTURE.





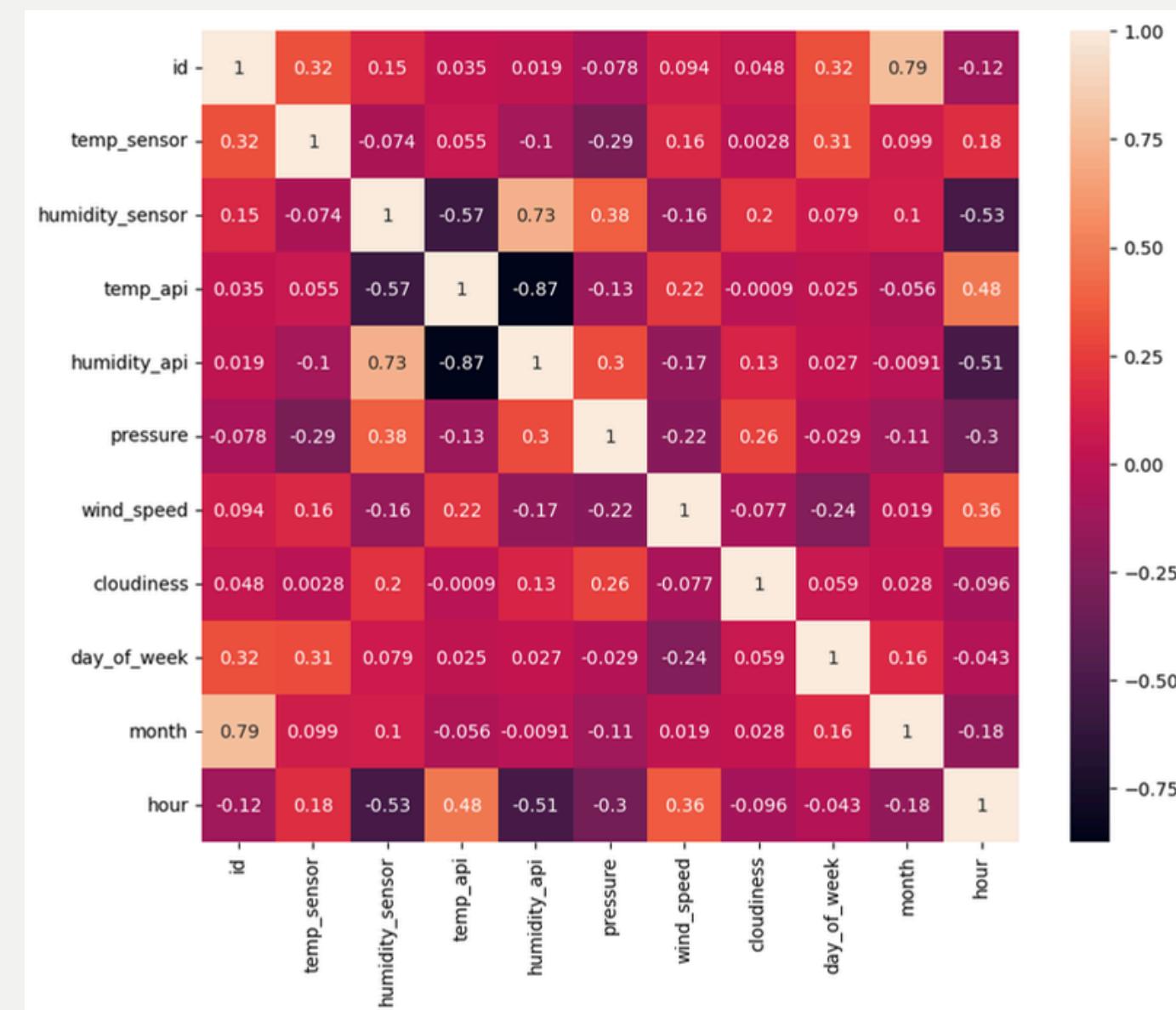






DATA EXPLORATION

USE HEATMAP TO SHOW CORRELATION AMONG NUMERICAL FEATURES



DATA PREPROCESSING

DATA PREPROCESSING

CONVERT TS TO NEW COLUMNS DAY_OF_WEEK, MONTH, HOUR THEN DROP TS



```
1 data["day_of_week"] = data["ts"].dt.dayofweek  
2 data["month"] = data["ts"].dt.month  
3 data["hour"] = data["ts"].dt.hour  
4  
5 data.drop("ts", axis=1, inplace=True)  
6 display(data.head())
```

BECAUSE WE NEED USE THESE DATA COLUMNS FOR TRAINING MODEL

DATA PREPROCESSING

PREDICTOR AND TARGET SPLIT



```
1 X = data.drop(["weather"], axis=1)
2 y = data["weather"]
```

DATA PREPROCESSING

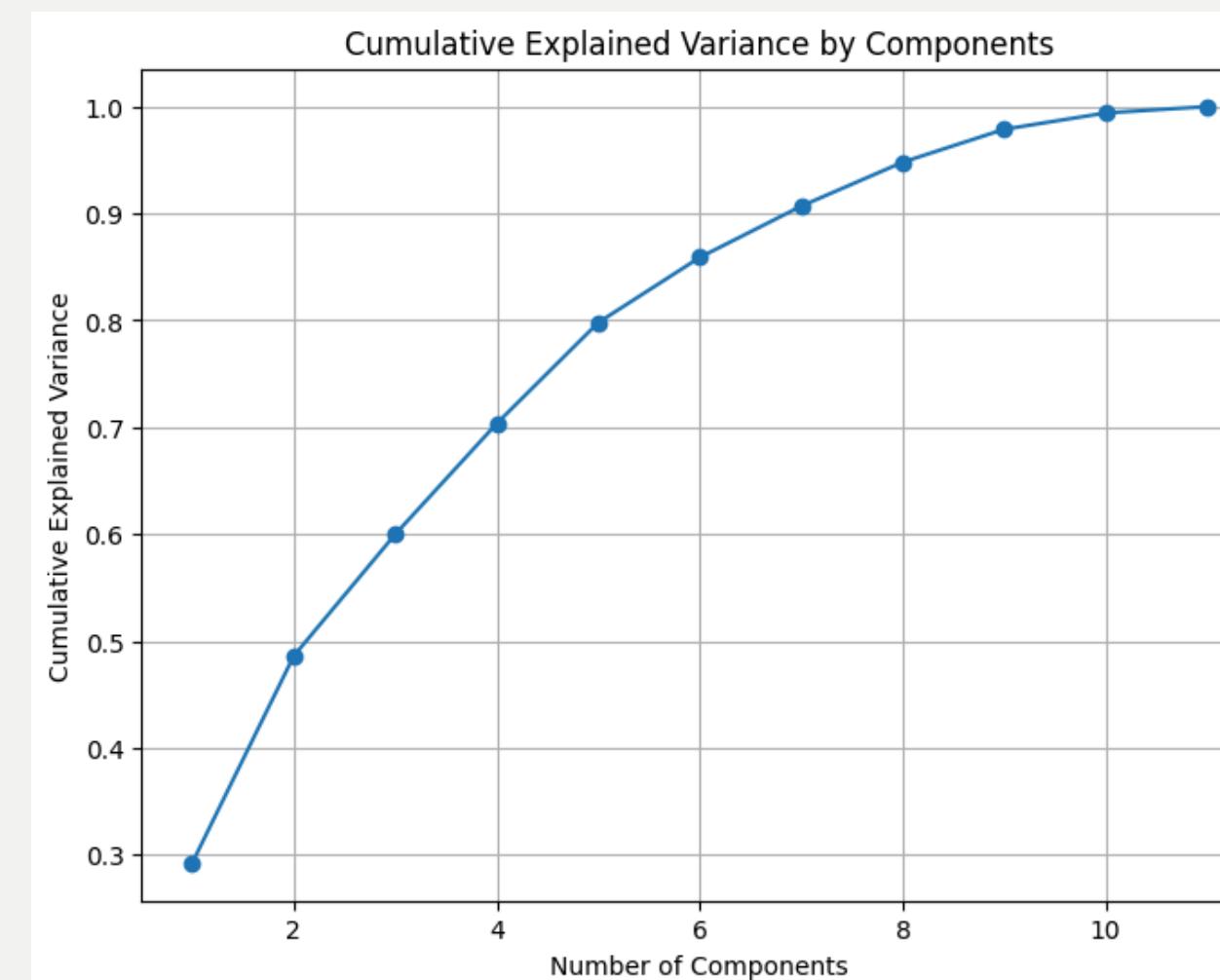
SCALE DATA WITH STANDARD SCALER



```
1 # Scale data (X)
2 standard_scaler = StandardScaler()
3 X_scaled = standard_scaler.fit_transform(X)
4 display(X_scaled)
```

DATA PREPROCESSING

PLOT CUMULATIVE VARIANCE BY COMPONENT



FROM PLOT WE SELECT NUMBER OF COMPONENT IS
5 BECAUSE IT NEAR TO 80 PERCENT

DATA PREPROCESSING

THEN WE ADAPT PCA 5 COMPONENTS TO OUR DATA

```
● ● ●  
1 pca = None  
2 pca = PCA(n_components=5)  
3 pca.fit(X_scaled)  
4  
5 X_pca = pca.transform(X_scaled)  
6 display(X_pca)
```

IMBALANCED DATA

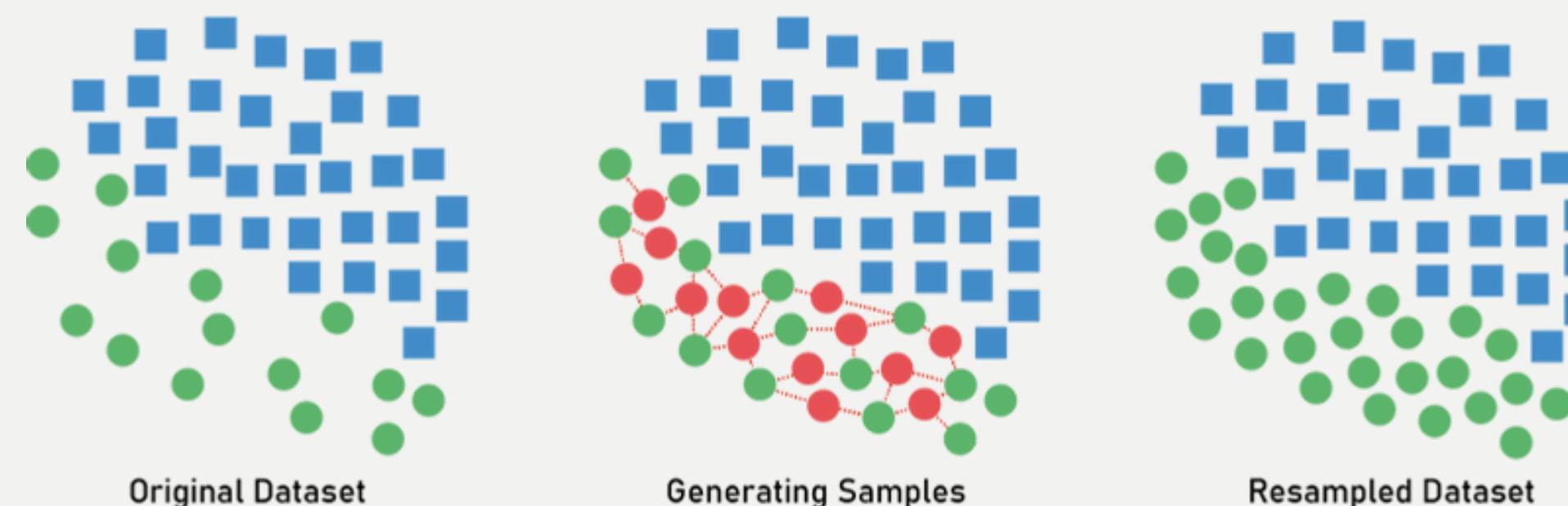
refers to a situation in classification problems where the distribution of classes in the dataset is highly skewed, meaning that one class is significantly more prevalent than the others. This imbalance can lead to biased models that perform poorly in accurately predicting the minority class, as the model may become overly biased towards the majority class.



SMOTE

(Synthetic Minority Over-sampling Technique) is a method used to address class imbalance in datasets by generating synthetic examples of the minority class. It works by creating new synthetic instances along the line segments joining existing minority class instances, thereby balancing the class distribution and improving the performance of machine learning models on imbalanced datasets.

Synthetic Minority Oversampling Technique



WHY SMOTE?

During summer, our data collection predominantly reflects sunny conditions with fewer instances of rain. This imbalance can arise due to the larger volume of sunny data compared to rainy data.



```
1 # Using smote
2 smote = SMOTE()
3 X_resampled, y_resampled = smote.fit_resample(X_pca, y)
4
5 display(X_resampled)
6 display(y_resampled)
```

WEATHERSENSE PROJECT

MODELING PART

RANDOM FOREST

We use Random Forest classification techniques to predict weather because Random Forest is a type of machine learning that creates a group of decision trees. It's straightforward to use and often gives excellent results without needing fine-tuning.

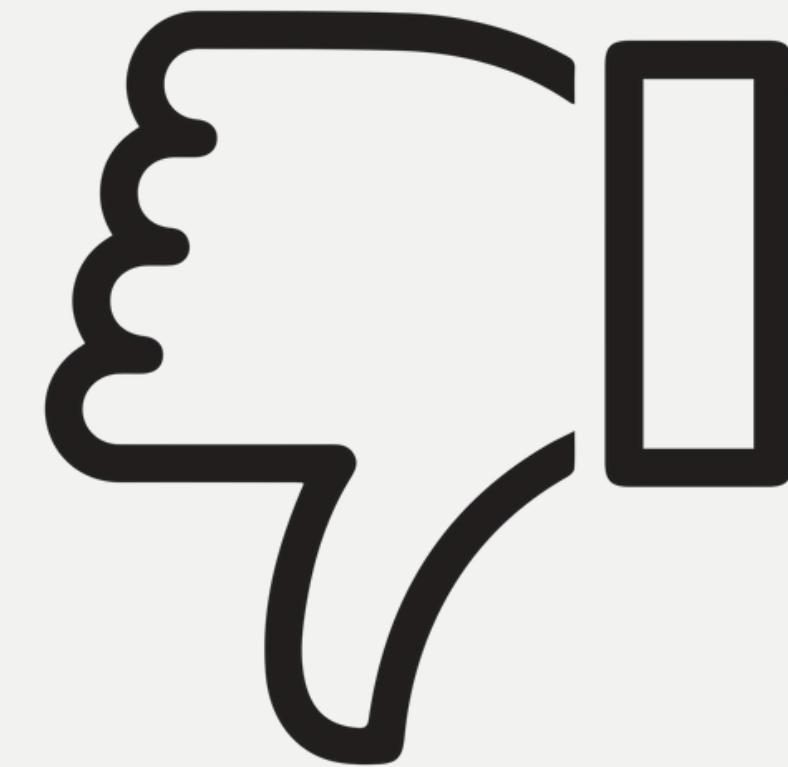


PROS

- Versatility: Random Forests can do both classification and regression tasks.
- Data Compatibility: Works with categorical and numerical data without needing scaling.
- Feature Selection: Automatically picks relevant features.
- Outlier Resilience: Handles outliers well.
- Relationship Handling: Works with linear and non-linear relationships.
- Accuracy: Often provides high accuracy.
- Bias-Variance Balance: Balances bias and variance effectively.

CONS

- Interpretability: Not easy to interpret like linear regression.
- Computationally Intensive: Can be slow for large datasets.
- Black Box Nature: Limited control over model workings.



MODELING

SPLIT TRAIN TEST TECHNIQUE



```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

MODELING

USE LABELENCODER TO ENCODE Y_TRAIN



```
1 label_encoder = LabelEncoder()  
2 y_train_encoded = label_encoder.fit_transform(y_train)  
3 display(y_train_encoded)
```

MODELING

GRID SEARCH CV TECHNIQUE

```
● ● ●  
1 param_grid = {  
2     'n_estimators': range(10, 201, 10)  
3 }  
4  
5 # Create a Random Forest classifier  
6 rf_classifier = RandomForestClassifier(random_state=1)  
7  
8 # Perform grid search with cross-validation  
9 grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy')  
10 grid_search.fit(X_train, y_train_encoded)  
11  
12 # Get the best parameters and best score  
13 best_n_estimators = grid_search.best_params_['n_estimators']  
14 best_score = grid_search.best_score_  
15  
16 print("Best Number of Estimators:", best_n_estimators)  
17 print("Best Accuracy Score:", best_score)
```

TO FIND BEST K FOR RANDOM FOREST

MODELING

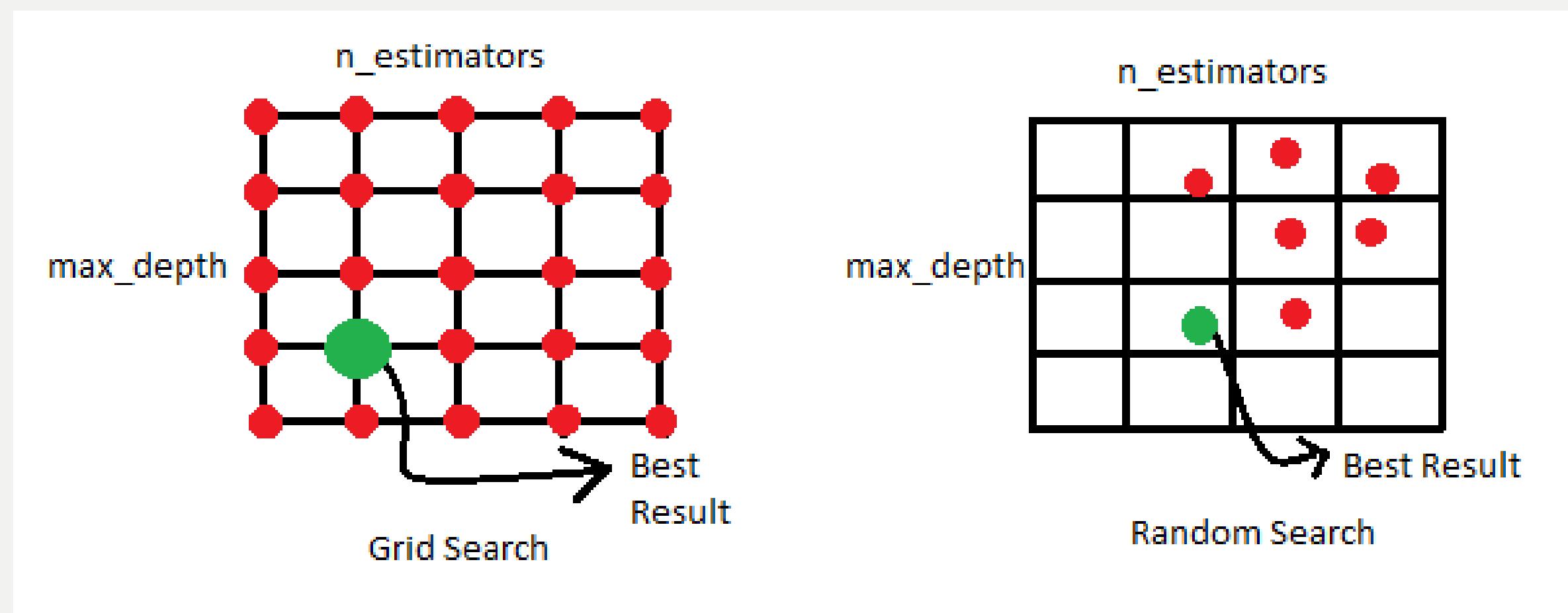
TRAIN MODEL WIL BEST K



```
1 rf_classifier = None  
2 rf_classifier = RandomForestClassifier(n_estimators=best_n_estimators, random_state=1)  
3 rf_classifier.fit(X_train, y_train_encoded)
```

GRID SEARCH CV

helps find the best settings for a model by trying different options and picking the one that works best. It's like testing different ingredients for a recipe to make the tastiest dish.



WEATHERSENSE PROJECT

EVALUATE THE MODEL PART

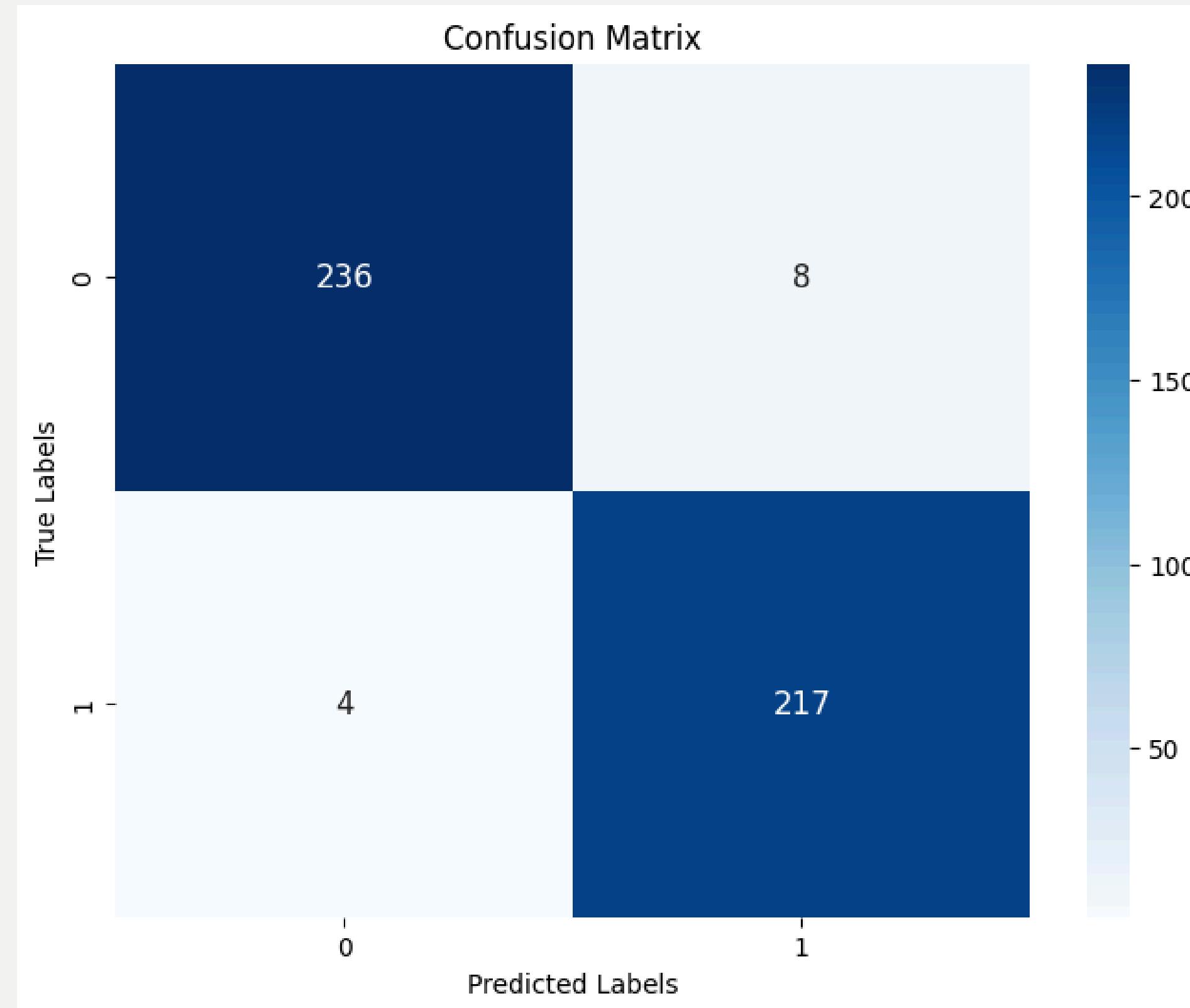
```
Accuracy: 0.9741935483870968
```

```
Classification Report:
```

	precision	recall	f1-score	support
Clouds	0.98	0.97	0.98	244
Rain	0.96	0.98	0.97	221
accuracy			0.97	465
macro avg	0.97	0.97	0.97	465
weighted avg	0.97	0.97	0.97	465

```
Confusion Matrix:
```

```
[[236  8]
 [ 4 217]]
```



WEATHERSENSE PROJECT

POSSIBLE APPLICATION

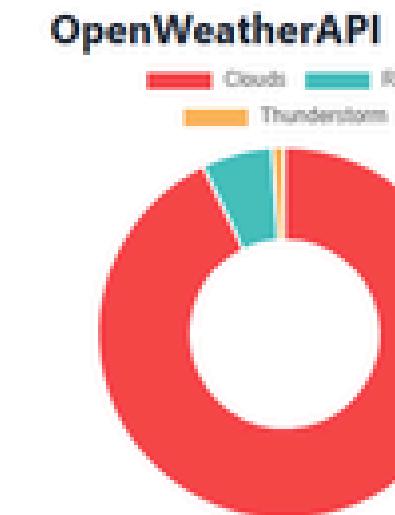
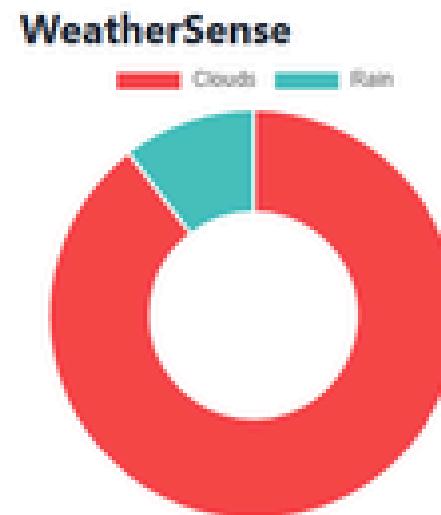
POSSIBLE APPLICATION

- Forecasting: Providing accurate weather forecasts for various locations and time intervals, helping individuals and organizations plan their activities accordingly.
- Agriculture: Assisting farmers in making informed decisions about planting, harvesting, irrigation, and pest control based on weather predictions.
- Travel : Helping travelers plan their trips by providing weather forecasts for their destinations, ensuring they have a pleasant experience.
- Transportation: Enhancing transportation safety and efficiency by predicting weather-related hazards such as storms, heavy rainfall, or snowfall.

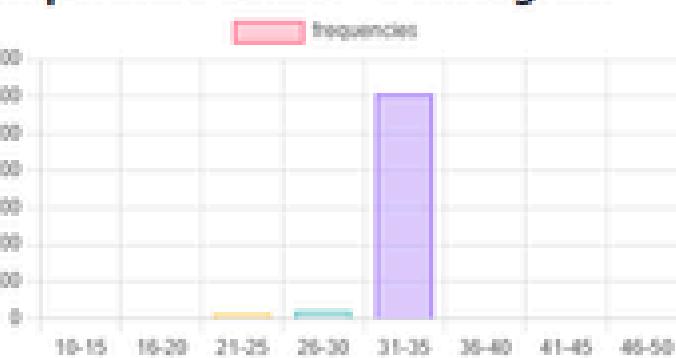
WEATHERSENSE SHOWCASE

WeatherSense

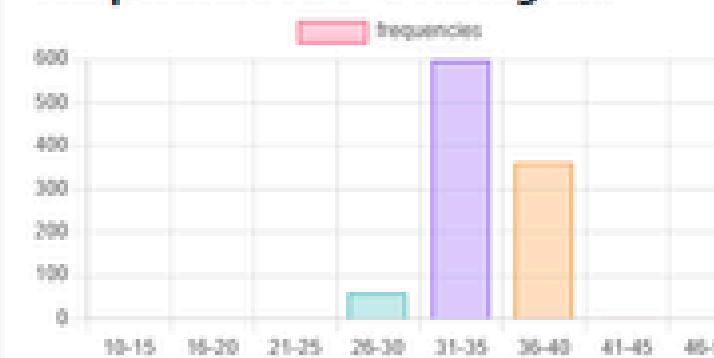
Home History Github Contact Data Visualize



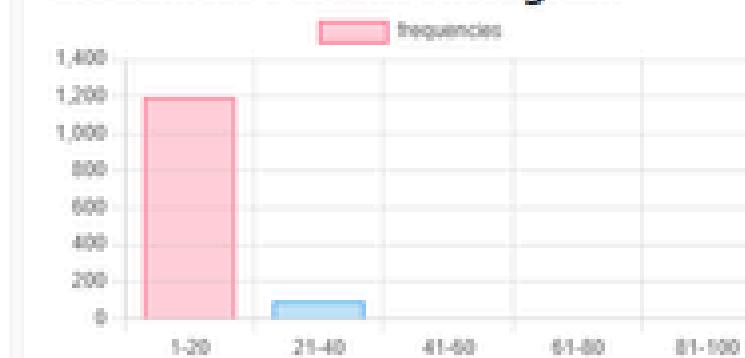
Temperature Sensor °C Histogram



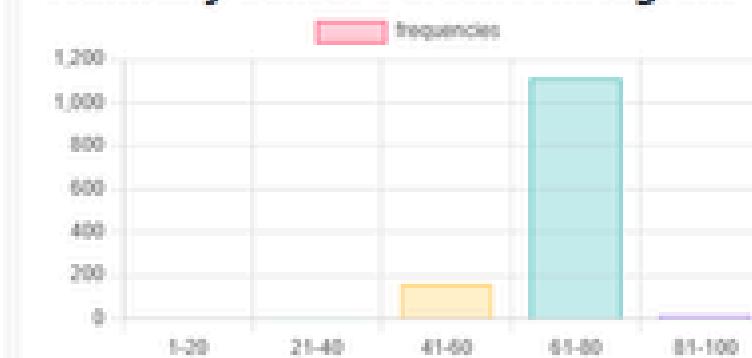
Temperature API °C Histogram



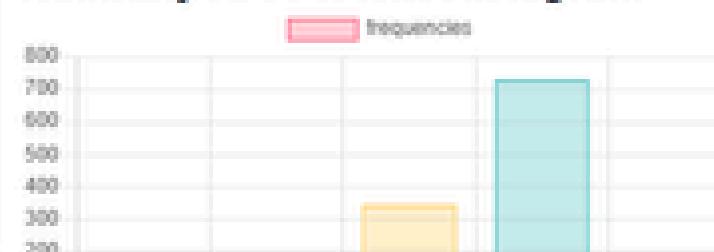
Cloudiness Percent Histogram



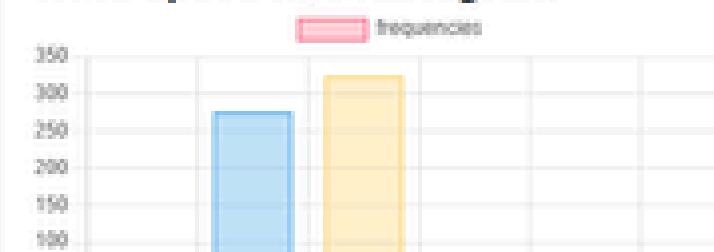
Humidity Sensor Percent Histogram



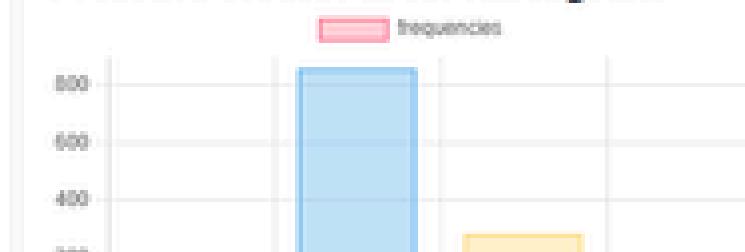
Humidity API Percent Histogram



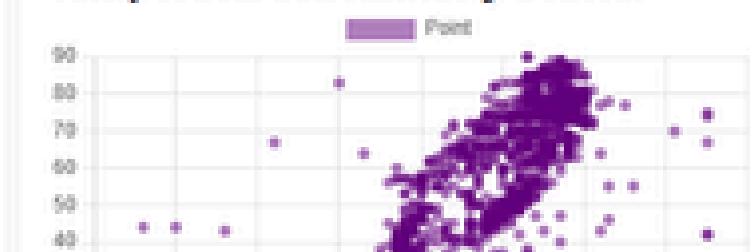
Wind Speed m/s Histogram



Pressure HectoPascal Histogram



Temperature&Humidity Scatter



WEATHERSENSE SHOWCASE

WeatherSense

[Home](#) [History](#) [Github](#) [Contact](#)

Latest update on April 23, 2024 at 3:25 PM

WeatherSens

Clouds

Temperature WeatherSense

27 °C

Humidity WeatherSense

69 Percent

Cloudiness

20 Percent

OpenWeatherMap

Clouds

Temperature OpenWeatherMap

37.83 °C

Humidity OpenWeatherMap

44 Percent

Pressure

1005 Hectopascal

Temperature Percentage Error

28.63 Percent

Humidity Percentage Error

56.82 Percent

Wind Speed

5.66 m/s

WEATHERSENSE SHOWCASE

WeatherSense	Home	History	Github	Contact	
Data on April 23, 2024 at 3:25 PM	View	Data on April 23, 2024 at 3:15 PM	View	Data on April 23, 2024 at 3:05 PM	View
Data on April 23, 2024 at 12:38 PM	View	Data on April 23, 2024 at 12:28 PM	View	Data on April 23, 2024 at 12:18 PM	View
Data on April 23, 2024 at 12:08 PM	View	Data on April 23, 2024 at 11:58 AM	View	Data on April 23, 2024 at 11:48 AM	View
Data on April 23, 2024 at 11:38 AM	View	Data on April 23, 2024 at 11:28 AM	View	Data on April 23, 2024 at 11:18 AM	View
Data on April 23, 2024 at 11:08 AM	View	Data on April 23, 2024 at 10:58 AM	View	Data on April 23, 2024 at 10:48 AM	View
Data on April 23, 2024 at 10:38 AM	View	Data on April 23, 2024 at 10:28 AM	View	Data on April 23, 2024 at 10:18 AM	View
Data on April 23, 2024 at 10:08 AM	View	Data on April 23, 2024 at 9:58 AM	View	Data on April 23, 2024 at 9:48 AM	View
Data on April 23, 2024 at 9:38 AM	View	Data on April 23, 2024 at 9:28 AM	View	Data on April 23, 2024 at 9:18 AM	View

REFERENCE

<https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04>

https://www.google.com/url?q=https://ntcloudsolutions.ntplc.co.th/knowledge/imbalanced-data-classification/&sa=D&source=editors&ust=1715032692718357&usg=AOvVaw2gaN-D_Gew2hcDg0nF9Z7Z



PROJECT MEMBER

1. WISSARUT KANASUB

**2. SUKPRACHOKE LEELAPISUTH
(DROPPED)**

