UNIVERSITI TEKNOLOGI PETRONAS

AI INDIVIDUAL PROJECT

MAY 2025 SEMESTER

'AFFAN NAJIY BIN RUSDI

22010453

BACHELOR OF COMPUTER SCIENCE

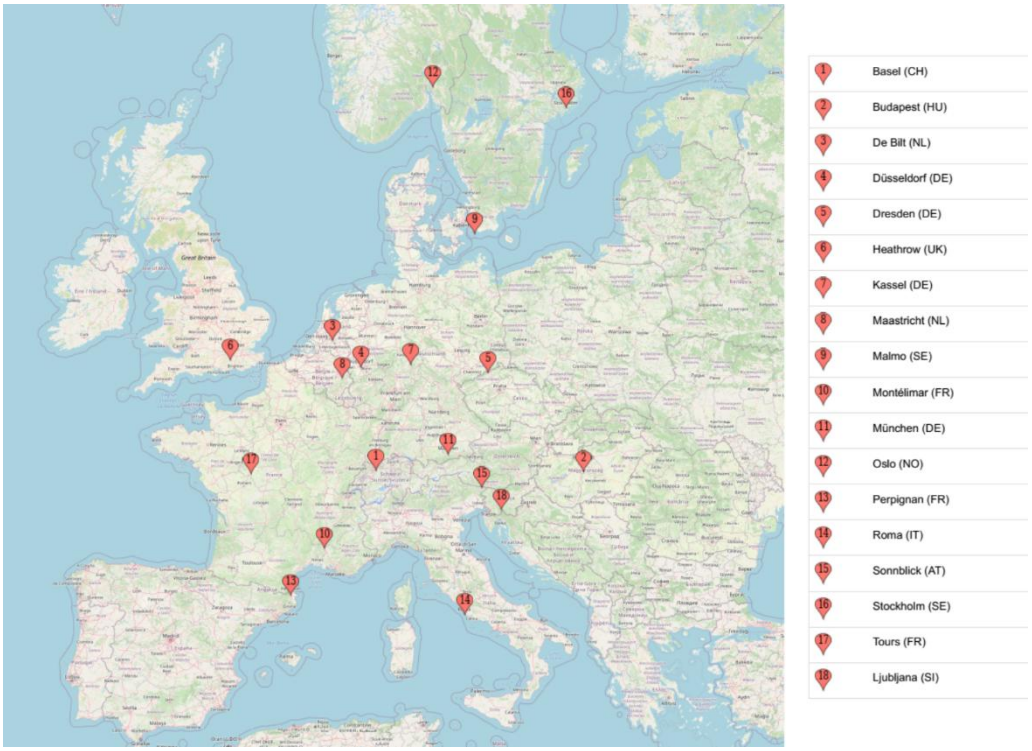
ARTIFICIAL INTELLIGENCE

TEB2023

Contents

Project Overview

The project is to develop a machine learning model to predict weather conditions. Historical weather data from the years 2000 to 2010 from 18 European cities were used in this project (Huber, 2021). Based on variables available in the dataset, the machine learning model is to predict the weather conditions for temperature (°C), humidity, and pressure (hPa). This project was implemented using the Python language in Jupyter Notebook with a machine learning library called scikit-learn.



The map of the locations that provided the data for the dataset

Objectives

1. To predict temperature, humidity, and pressure using linear regression on historical weather data.
2. To evaluate model performance with MSE and R^2 scores for each weather parameter.
3. To visualize correlations and prediction accuracy through heatmaps and scatter plots.

Methodology

Data Preparation

Data cleaning involved dropping missing values (null) from the dataset by using the `df.dropna()` function.

```
# 1. Handle Missing Values
df = df.dropna() # Drop rows with missing values
```

Code of Dropping Missing Data

The dataset is then encoded for the categorical values using the one-hot encoding. The one-hot encoding turns categorical data into numerical data. This is the line of code that does that: `df = pd.get_dummies(df, drop_first=True)`.

```
# 2. Encode Categorical Data (One-Hot Encoding Method) - Turn categorical data
into numerical data
df = pd.get_dummies(df, drop_first=True) # Drop first column
```

Code of Encoding Categorical Data

The dataset is then visualized using a correlation matrix using a heatmap to show the relationship between the pairs of variables in the dataset. Thus, using a heatmap with coefficients ranging from -1 to 1 to show the strength of the correlations.

```
# 3. Visualize Correlation Matrix
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(numeric_only=True), cmap='coolwarm', annot=False)
plt.title("Correlation Matrix of Numerical Features")
plt.show()
```

Code of Correlation Matrix

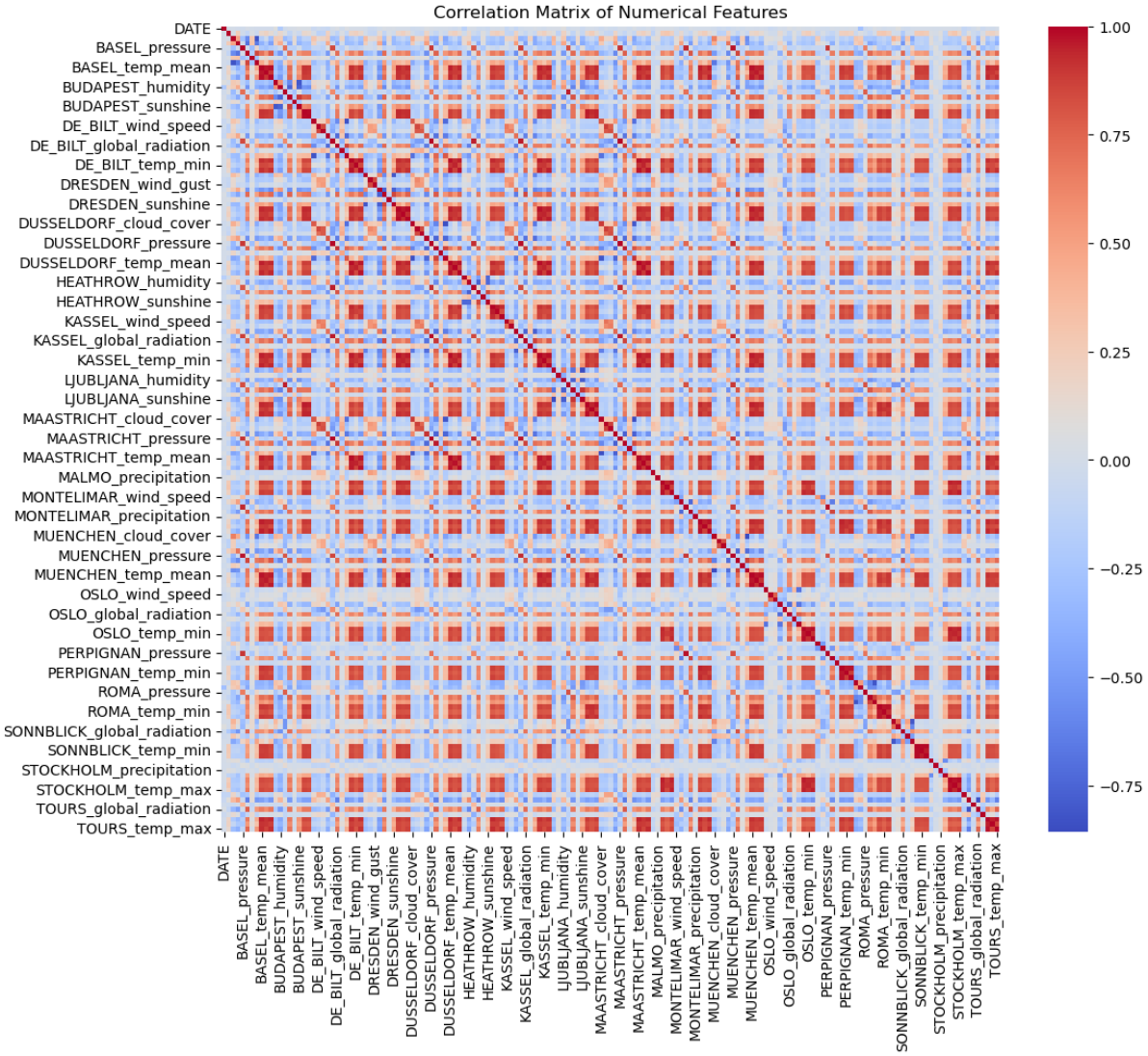


Image of Correlation Matrix

Correlation Value (r) Interpretation Table

Correlation Value (r)	Interpretation
+1.0	Perfect positive correlation
+0.7 to +0.9	Strong positive correlation
+0.4 to +0.6	Moderate positive correlation
+0.1 to +0.3	Weak positive correlation
0	No correlation
-0.1 to -0.3	Weak negative correlation
-0.4 to -0.6	Moderate negative correlation
-0.7 to -0.9	Strong negative correlation
-1.0	Perfect negative correlation

So, as the table would suggest and based on the heatmap provided, a +1.0-correlation value relationship exists between BASEL_temp_mean and BASEL_temp_mean.

Modelling

The linear regression model is used in this program to train 3 separate models, with 1 model per target variable, which focused on BASEL_temp_mean, BASEL_humidity, and BASEL_pressure, which were chosen based on the correlation matrix with the best correlation of +1.0.

Whereas,

X1 and y1 = BASEL_temp_mean

X2 and y2 = BASEL_humidity

X3 and y3 = BASEL_pressure

Data is split first to allow for model evaluation and general optimization for the model. The `test_size=0.2` means that 80% of the model learns patterns and the remaining 20% is to check for accuracy.

```
from sklearn.model_selection import train_test_split

# Split Data Temperature
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2,
random_state=42)

# Split Data Humidity
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2,
random_state=42)

# Split Data Pressure
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.2,
random_state=42)
```

Code of Splitting Data

The linear regression model is then implemented into the code based on the split data.

```
from sklearn.linear_model import LinearRegression

# Init Temperature Model
model_temp = LinearRegression()
# Train Temperature Model
model_temp.fit(X1_train, y1_train)

# Init Humidity Model
model_humidity = LinearRegression()
# Train Humidity Model
model_humidity.fit(X2_train, y2_train)

# Init Pressure Model
model_pressure = LinearRegression()
# Train Pressure Model
model_pressure.fit(X3_train, y3_train)
```

Code of Build and Training the Linear Regression Model

Evaluation Metrics

The models are then evaluated based on the one initialized. There will be 3 models evaluated, which are (*X1 and y1 = BASEL_temp_mean), (X2 and y2 = BASEL_humidity), and (X3 and y3 = BASEL_pressure)*. These models are evaluated by using the same metrics, which are the Mean Squared Error (MSE) and R-squared score (R2). The mean square error measures how far the predicted value is from the actual value, the lower the value, the better the model. Meanwhile, the R-squared score measures how well the model fits the data, which is the variance of the dataset.

```
# -----Temperature Predictions-----
y1_pred = model_temp.predict(X1_test)

# Calc Mean Squared Error
mse_1 = mean_squared_error(y1_test, y1_pred)

# Calc R-Squared Score
r2_1 = r2_score(y1_test, y1_pred)

# Print Results
print("Temperature Evaluation")
print(f"Mean Squared Error: {mse_1}")
print(f"R2 Score: {r2_1}")
print()
```

Code for Evaluating BASEL_temp_mean

An output will be displayed for each variable calculated. Such as the BASEL_temp_mean:

```
Temperature Evaluation
Mean Squared Error: 0.19665724668451487
R2 Score: 0.9963551645217845
```

This means for BASEL_temp_mean the R^2 is nearly excellent as it approaches closer to 1, and the MSE is also good as it is low, approaching 0.

Results

Evaluated Models Results

The results of the 3 models are then displayed.

Target	Mean Squared Error (MSE)	R^2 Score	Interpretation
Temperature	0.20 °C²	0.996	Near-perfect prediction
Humidity	0.0017 %²	0.864	Strong prediction
Pressure	3.04×10^{-7} hPa²	0.995	Near-perfect prediction

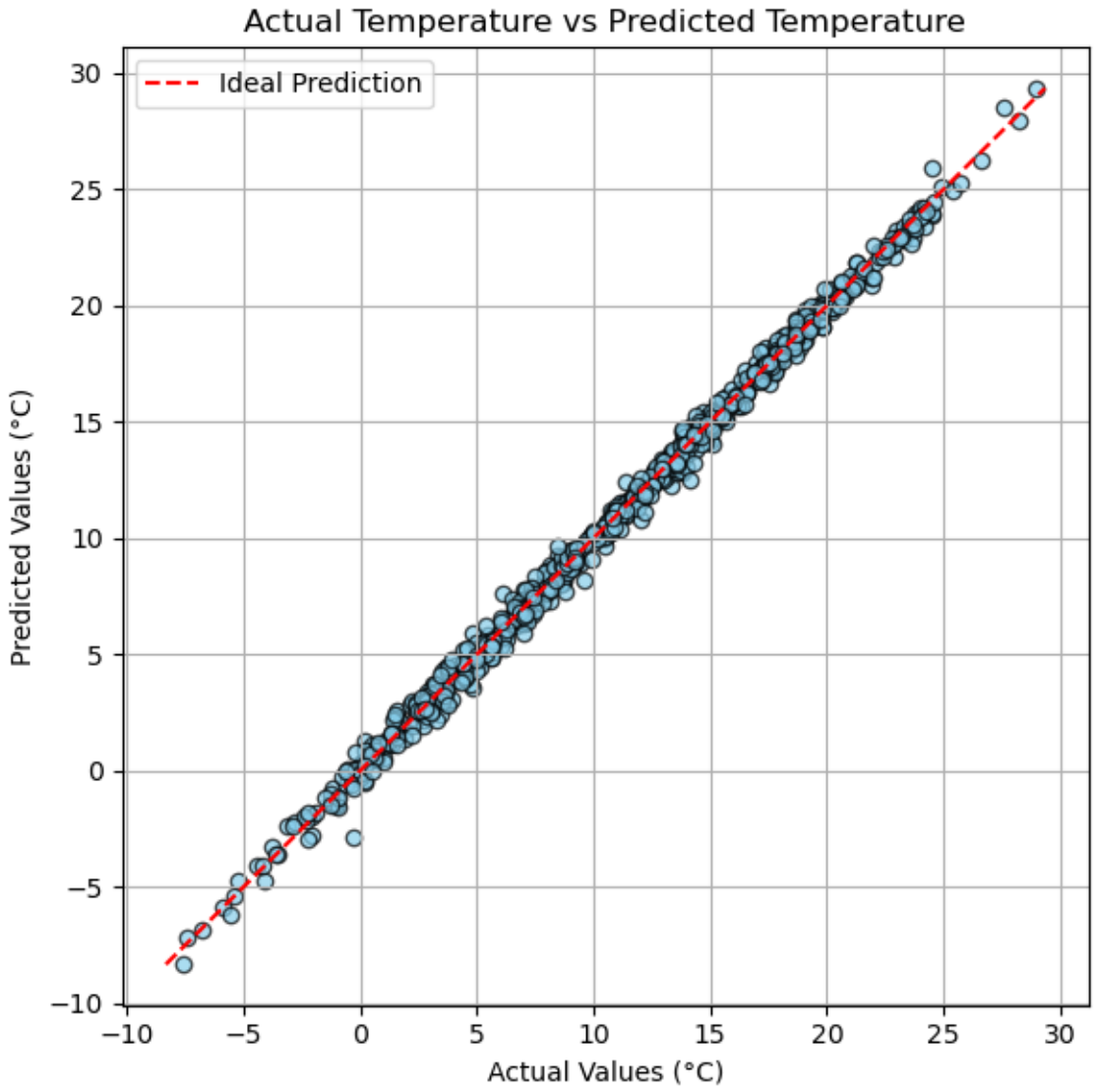
Table shows the calculated results for each of the trained models

Findings

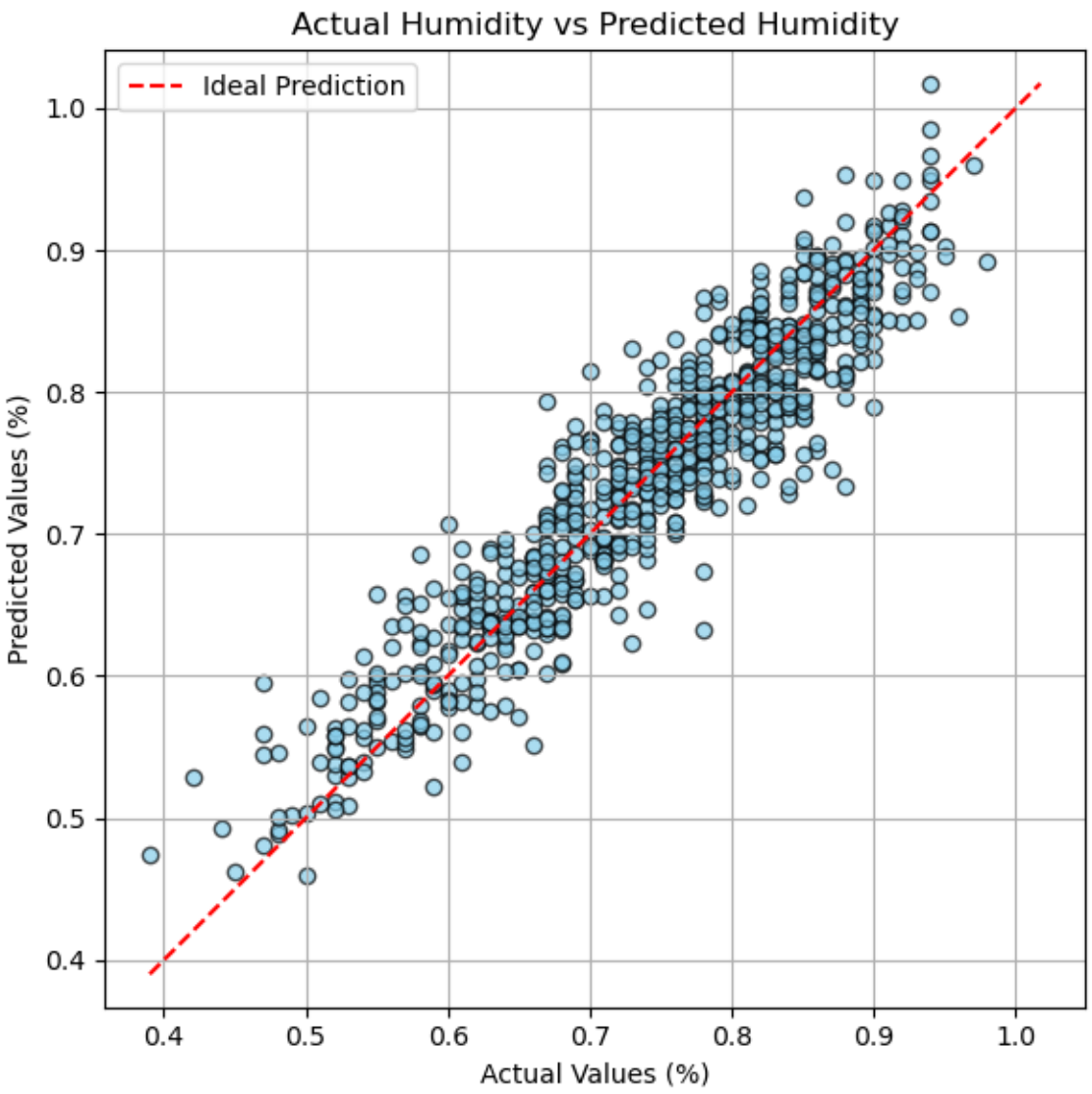
Temperature was the easiest to predict due to its high R^2 score, which is near-perfect. While humidity had the highest error rate due to its variability. Lastly, the model for pressure was stable and reliable.

Visualisation

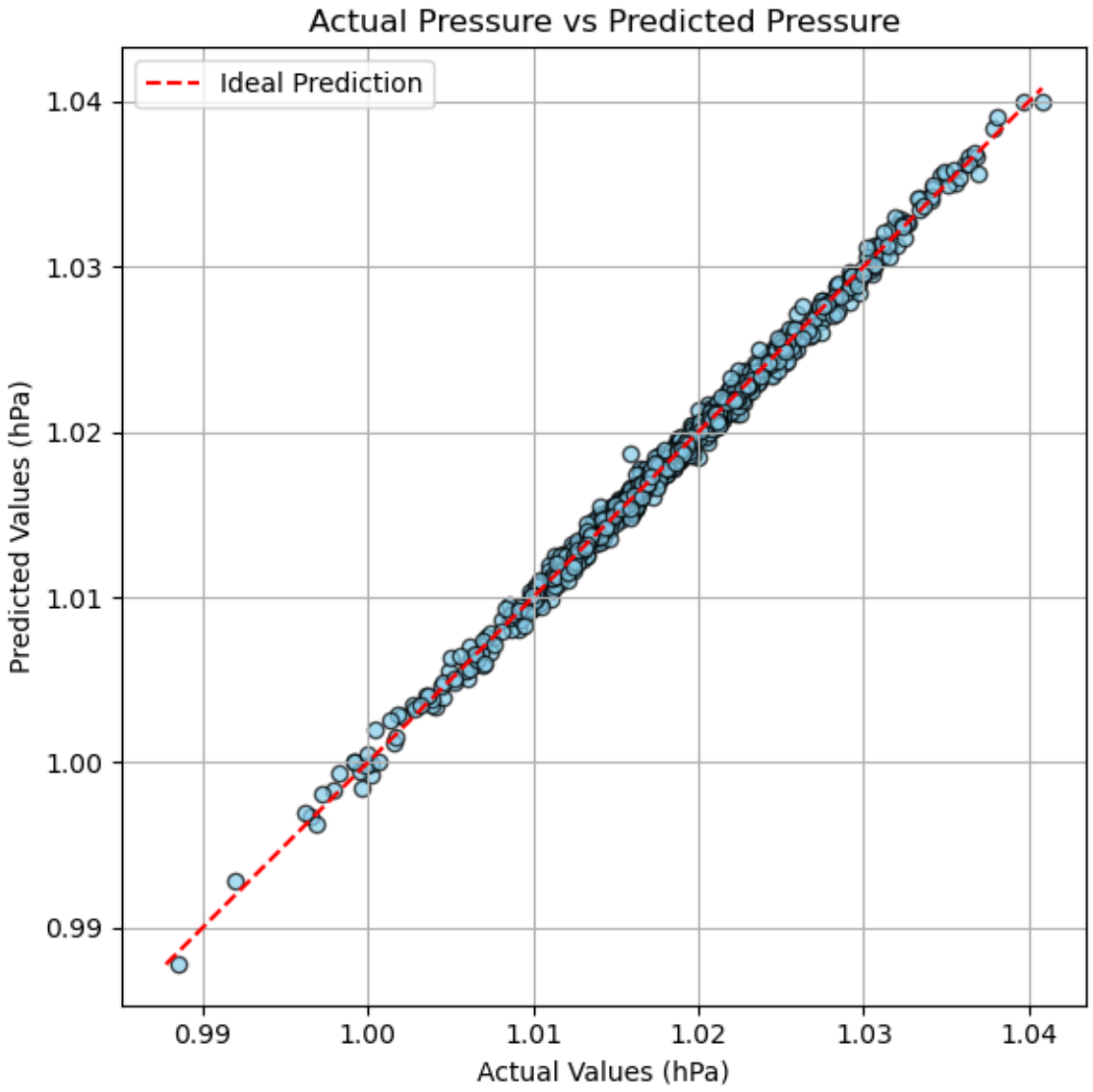
The data were visualized into a scatter plot with the x-axis being the actual data and the y-axis being the predicted data.



Scatter Plot shows the Temperature Model



Scatter Plot shows the Humidity Model



Scatter Plot shows the Pressure Model

Conclusion

The linear regression model works well for simple weather predictions, and it is suitable for the dataset that was obtained. A suggestion for future works is to try other models such as Random Forest and SVM while also including more features (variables) such as wind speed and precipitation.

References

'Affan Najiy Bin Rusdi. (2025, July). *AI Weather Prediction App [Computer software]*. Retrieved from GitHub: https://github.com/affannajiy/AI-Weather_Prediction_App

Huber, F. (2021, May 18). *Weather prediction dataset*. Retrieved from Zenodo: https://zenodo.org/records/4770937