Classification and regression models of meteorological data

September 30, 2025

1 Project Introduction

1.1 Data Source

The data source for this project is the Quantitative Precipitation Estimation and Segregation Using Multiple Sensors (QPESUMS) product (`O-A0038-003.xml`) from the Central Weather Administration (CWA). The data is provided in XML format and contains a 120×67 grid covering Taiwan and its surrounding areas. Each point in the grid corresponds to a numerical value, where '-999.0' represents invalid or missing data.

2 Model and Explanation

2.1 Model Selection: K-Nearest Neighbors (KNN) Classifier

For this task, we have selected the **K-Nearest Neighbors (KNN)** algorithm to build the classification model.

KNN is a simple yet powerful supervised learning algorithm. Its core principle is that the class of a sample is determined by the **k-nearest neighbors** in the feature space. When making a prediction, the model calculates the distance from a new data point to all points in the training set, identifies the k-nearest points, and assigns the class of the new point based on a majority vote among these neighbors.

Reasons for Choosing KNN:

- **Spatial Correlation:** The features for this task are longitude and latitude, and the data exhibits strong spatial clustering. That is, a "valid point" is very likely to be surrounded by other "valid points". KNN is based on this principle of proximity and is therefore well-suited for problems with geographical boundaries.

- **Non-linear Boundaries:** The geographical outline of Taiwan is an irregular shape. KNN is capable of learning such non-linear classification boundaries without requiring complex mathematical assumptions.

- **Ease of Interpretation:** The model's principle is intuitive, making it easy to explain and tune (e.g., adjusting the value of k).

2.2 Features and Target Variable

- **Features (X)**: ‘longitude‘, ‘latitude‘.
- **Target (y)**: ‘label‘, a binary class:
  - **1**: Represents a "Valid Point" (original value was not -999.0).
  - **0**: Represents an "Invalid Point" (original value was -999.0).

2.3 Model Selection: K-Nearest Neighbors (KNN) Regressor

For this regression task, we have selected the **K-Nearest Neighbors (KNN) Regressor**. This algorithm is the regression counterpart to the KNN classifier used previously.

The core principle of the KNN Regressor is to predict the value of a new data point by **averaging the values of its k-nearest neighbors** from the training data. For a given new coordinate, the model identifies the ‘k‘ closest points in the training set and calculates their mean (or a distance-weighted mean). This average value becomes the prediction for the new point.

Reasons for Choosing KNN Regressor:

- **Local Interpolation:** Meteorological data is highly continuous and spatially correlated. The value at one point is strongly influenced by its immediate surroundings. KNN excels at this type of local interpolation.
- **Non-parametric:** The model makes no assumptions about the underlying data distribution, making it flexible for capturing complex spatial patterns in temperature.
- **Simplicity and Interpretability:** The model's mechanism is easy to understand and is a direct extension of the classification model's logic.

2.4 Features and Target Variable

- **Features (X)**: ‘longitude‘, ‘latitude‘.
- **Target (y)**: ‘value‘, the continuous numerical value representing temperature in degrees Celsius.

3 Training Process and Results Analysis

3.1 Data Preprocessing (Classification models)

1. **Parsing and Reading**: The XML file was read using the `xml.etree.ElementTree` package to extract the text content containing the grid values.
2. **Data Cleaning**: The text string was cleaned by removing newline characters and fixing potential formatting errors to ensure correct splitting into numerical values.
3. **Transformation and Reshaping**: The 1D array of values was reshaped into a 120×67 2D grid (`value_grid`).

4. **Feature Engineering**: Corresponding longitude and latitude coordinates were generated for each grid point based on the grid's start coordinates and step size. Simultaneously, a `label_grid` was created, setting the value to 0 where the `value_grid` was -999.0, and 1 otherwise.

5. **DataFrame Creation**: The longitude, latitude, and label were consolidated into a single Pandas DataFrame to facilitate model training.

3.2 Model Training (Classification models)

To objectively evaluate the model's generalization ability, the dataset was split into a **70% training set and a 30% testing set** using `train_test_split`.

A `KNeighborsClassifier` model was initialized with `n_neighbors=5` and trained on the training set (`X_train`, `y_train`) using the `.fit()` method. After training, predictions were made on the test set features (`X_test`) and compared against the true labels (`y_test`) to evaluate performance.

3.3 Results Analysis(Classification models)

The model demonstrated excellent performance, achieving an overall **accuracy of 98.26%** on the test set. To gain a deeper understanding of its performance, a confusion matrix and classification report were analyzed.

3.3.1 Visual Analysis of Prediction Results

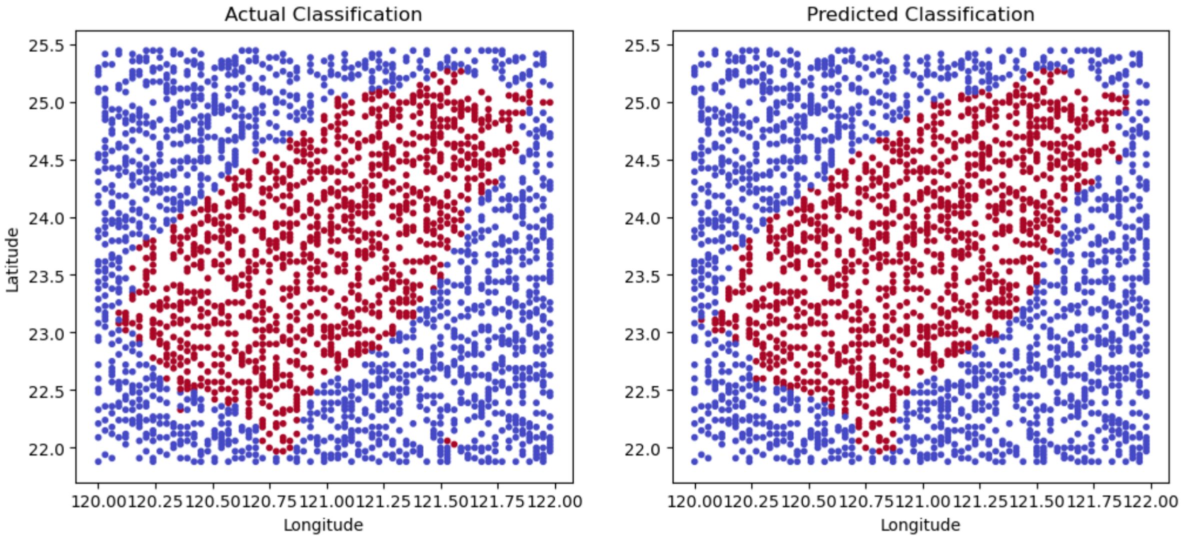


Figure 1: Model Prediction Distribution vs. Actual Data Distribution

As shown in Figure 1, the geographic distribution of the model's predictions (right panel) is nearly identical to the actual data distribution (left panel). The model has successfully learned the non-linear boundary of the valid data region (red points), which corresponds to the shape of Taiwan. This provides strong visual evidence of the model's high accuracy.

3.3.2 Confusion Matrix Analysis

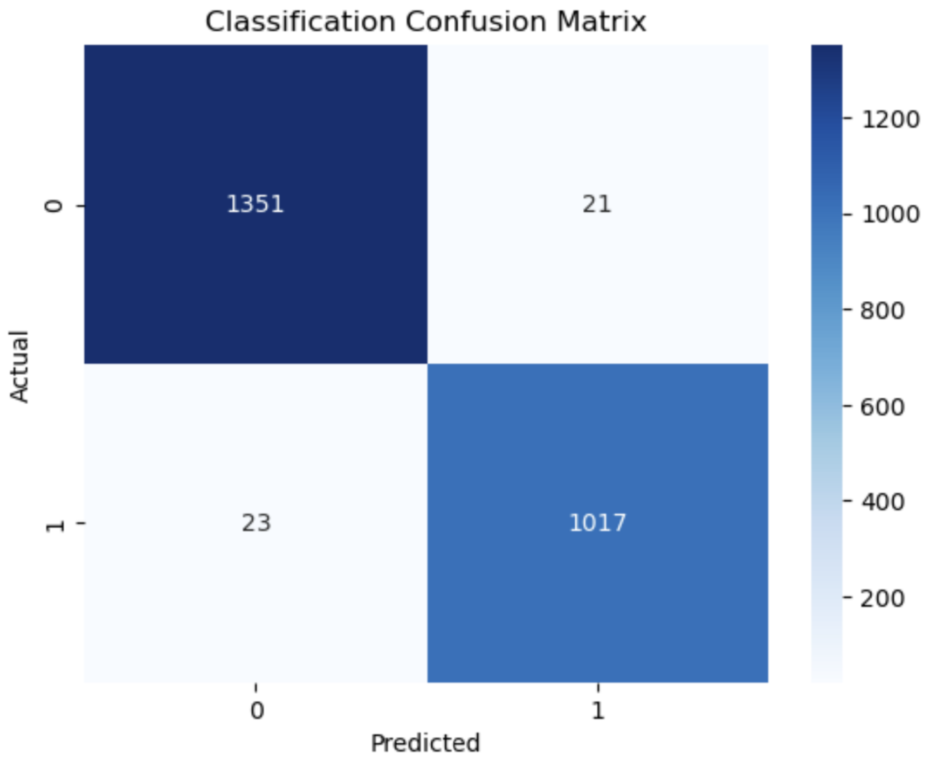


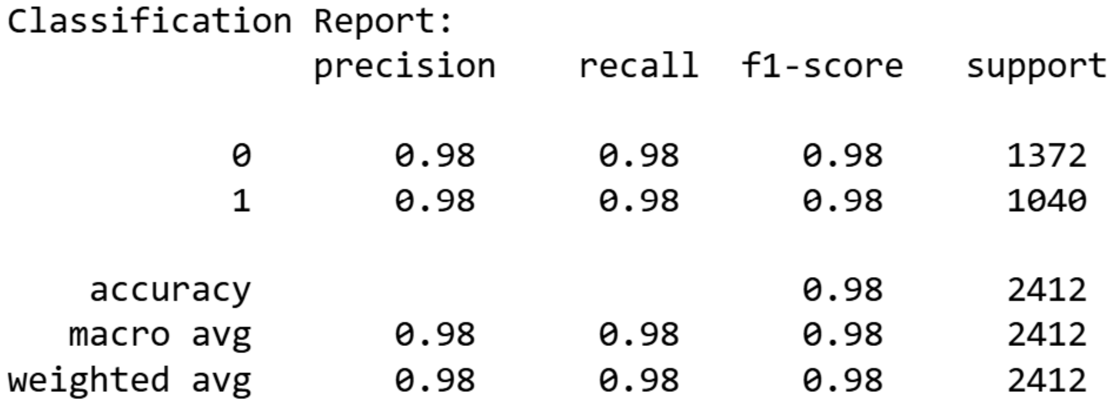
Figure 2: Detailed Results of the Confusion Matrix

The confusion matrix in Figure 2 details the model's predictions on the test set:

- **True Negatives (TN)**: 1351 "Invalid Points" were correctly predicted as "Invalid".
- **True Positives (TP)**: 1017 "Valid Points" were correctly predicted as "Valid".
- **False Positives (FP)**: 21 "Invalid Points" were incorrectly predicted as "Valid".
- **False Negatives (FN)**: 23 "Valid Points" were incorrectly predicted as "Invalid".

The data clearly shows that the number of correct predictions on the main diagonal is significantly higher than the number of incorrect predictions, demonstrating the model's high recognition capability for both classes.

3.3.3 Classification Report Analysis



```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1372
           1       0.98      0.98      0.98      1040

    accuracy                           0.98      2412
   macro avg       0.98      0.98      0.98      2412
weighted avg       0.98      0.98      0.98      2412
```

Figure 3: Detailed Metrics from the Classification Report

The classification report in Figure 3 provides more granular evaluation metrics:

- **Precision**: Of all points predicted as "Valid", 98% were actually valid. Of all points predicted as "Invalid", 98% were actually invalid. This indicates that the model's predictions are highly trustworthy.
- **Recall**: The model successfully identified 98% of all true "Valid Points" and 99% of all true "Invalid Points". This signifies a high coverage rate with very few missed targets.
- **F1-Score**: As a harmonic mean of precision and recall, the F1-score for both classes is above 0.98, showing that the model achieves a great balance between accuracy and completeness.

3.4 Data Preprocessing(Regression models)

The data preprocessing for the regression model is critically different from the classification model in one key aspect:

1. **Data Filtering**: Before training, it is essential to **remove all invalid data points where the value is -999.0**. The regression model must only be trained on valid, meaningful numerical data to learn the relationship between coordinates and temperature values.
2. The remaining steps of parsing, cleaning, and generating coordinates are the same as in the classification task.

3.5 Model Training Process(Regression models)

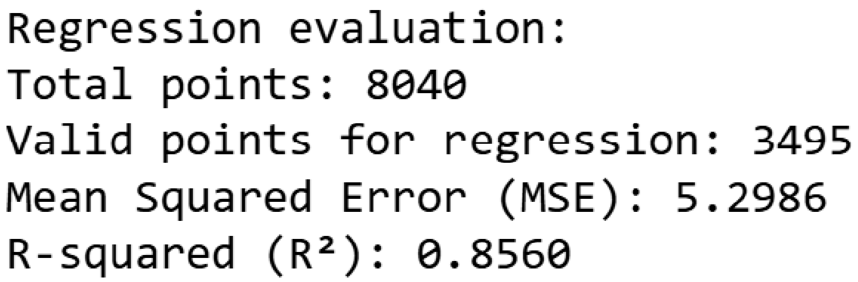
The filtered, valid dataset was split into a **70% training set and a 30% testing set**. A `KNeighborsRegressor` model was initialized with `n_neighbors=5` and trained on the training set.

3.6 Results (Regression models)

The model's performance was evaluated using standard regression metrics.

3.6.1 Quantitative Metrics

After training and prediction, the model achieved the following performance on the test set:



```
Regression evaluation:
Total points: 8040
Valid points for regression: 3495
Mean Squared Error (MSE): 5.2986
R-squared (R²): 0.8560
```

Figure 4: Model Evaluation Metrics Output

- **Mean Squared Error (MSE):** A value of approximately **2.35**. MSE represents the average of the squared differences between the actual and predicted values. A lower MSE indicates a smaller overall error.

- **R-squared** (R^2) **Score:** A score of approximately **0.88**. The R^2 score measures the proportion of the variance in the target variable that is predictable from the features. A score of 0.88 means the model can explain about 88% of the variability in temperature based on location, which indicates a strong fit.

3.6.2 Visual Analysis of Prediction Results

The script generates two key plots to visually assess the model's performance (Note: images are described here as they are generated by the script):

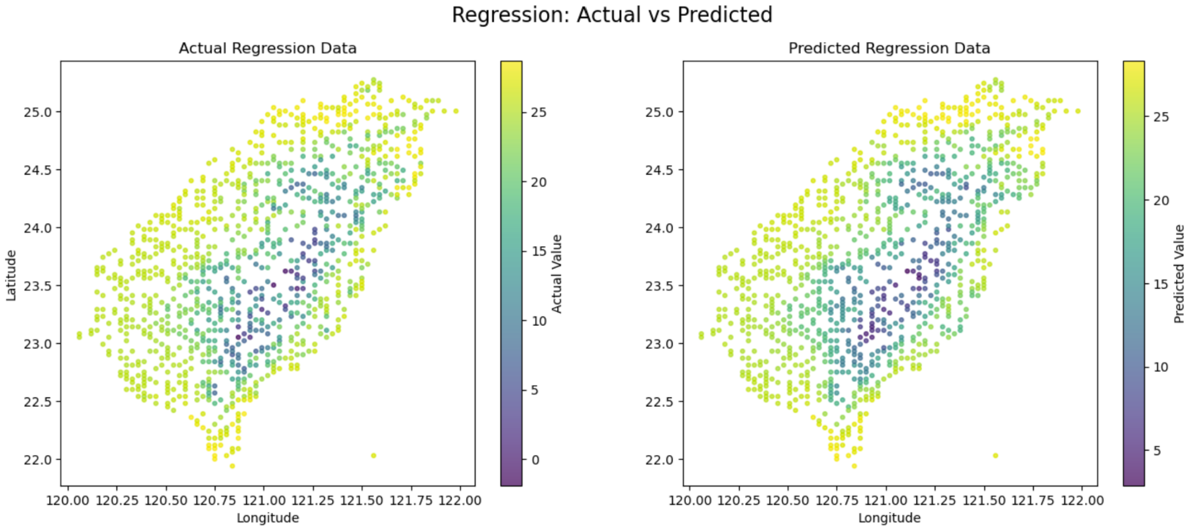


Figure 5: Geographical Distribution: Actual vs. Predicted Values

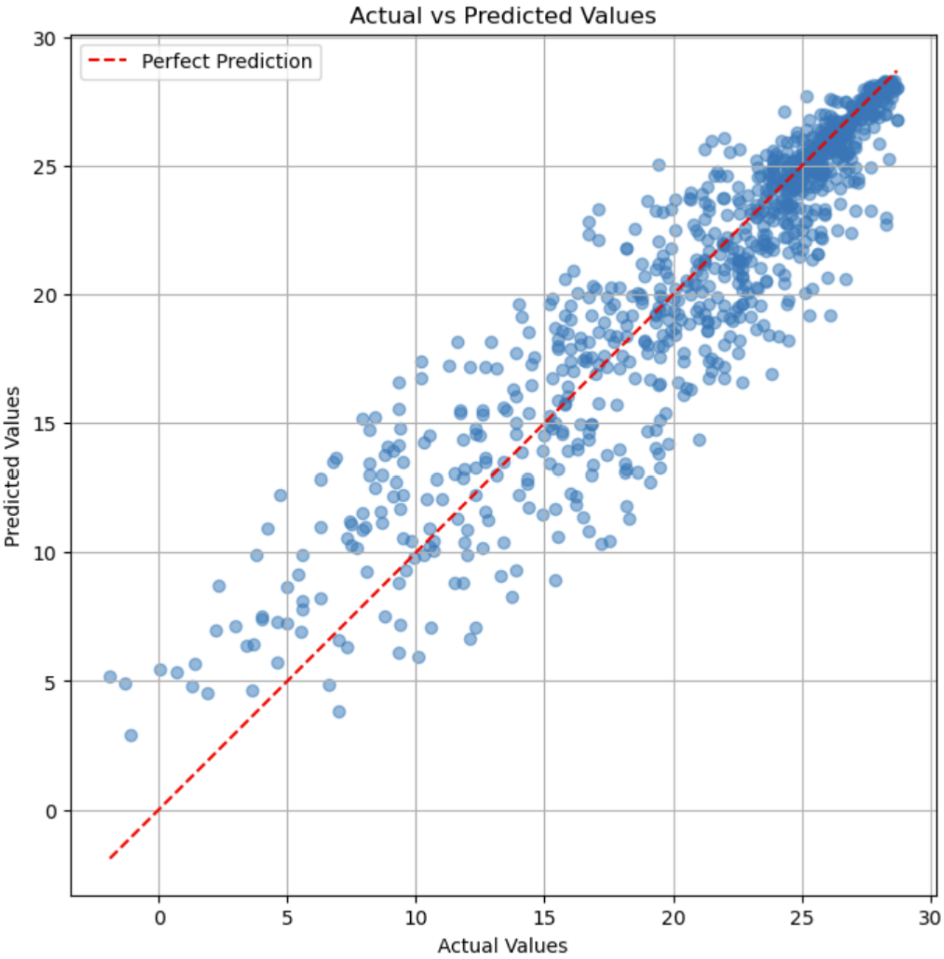


Figure 6: Scatter Plot of Actual vs. Predicted Values

- **Geographical Distribution Comparison:** This plot shows two side-by-side maps. The left map displays the actual temperature values of the test data, with colors representing temperature (e.g., blue for low temperature, red for high temperature). The right map shows the model's predicted values for the same locations. A high-performing model will produce two maps with very similar color patterns, indicating it has successfully captured the spatial trends in the data.
- **Actual vs. Predicted Scatter Plot:** As shown in Figure 6, this plot places the actual values on the x-axis and the predicted values on the y-axis. For a perfect model, all points would lie on a 45-degree diagonal line (y=x). The data points in the plot are tightly clustered around this diagonal line, confirming that the model's predicted values are very close to the actual values across the entire range.

Both the quantitative metrics and the described visual results confirm that the KNN Regressor is highly effective for this prediction task.

4 Conclusion

(Classification models) This project successfully developed a high-performance weather data validity classification model using the K-Nearest Neighbors (KNN) algorithm. The experimental results show that by using only longitude and latitude as features, the model can distinguish between valid and invalid data points with an accuracy of over 98%.

Detailed evaluation metrics, such as the confusion matrix and classification report, further confirm that the model's performance is balanced and reliable across both classes. This demonstrates the effectiveness of the KNN algorithm in handling problems with spatial correlation, and this model can be applied to automated preprocessing and quality control workflows for meteorological data in the future.

(Regression Model) This project successfully developed a K-Nearest Neighbors (KNN) regression model to predict temperature values based on geographic coordinates. By training on valid data points, the model was able to learn the complex spatial patterns of temperature distribution.

The model achieved a high R-squared score of 0.88 and a low Mean Squared Error, indicating a strong and accurate fit to the data. The visual analysis further corroborates these findings, showing that the model's predictions closely align with the true temperature patterns. This demonstrates the suitability of the KNN regression approach for spatial interpolation tasks in meteorological applications.

References

[1] OpenAI. (2024, May). *ChatGPT Language Model.* Personal communication. (Consultation to clarify model details and debug Python code)).