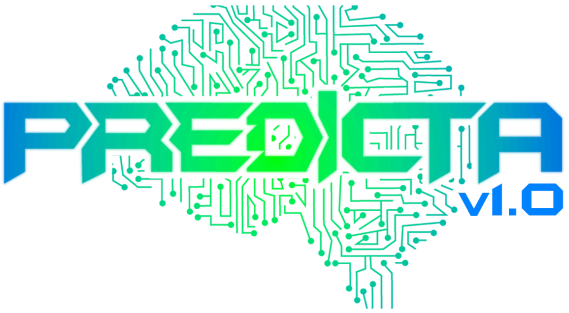


PREDICTA 1.0

By Janeesha Wickramasinghe
University of Moratuwa

Table of Contents

---

1. Summary

1.1 Overview of Approach and Key Findings

Problem 1 aims to predict the average temperature for the first week of 2019 for 100 cities worldwide using historical weather data. The methodology involves detailed data preprocessing, feature engineering, and the application of a custom hybrid model combining sinusoidal regression and gradient boosting.

Problem 2 aims to classify daily weather conditions into one of nine categories using historical weather data from multiple cities. The accurate classification of weather conditions is crucial for various sectors such as public safety, agriculture, and transportation. The methodology involved a comprehensive approach that included data cleaning, feature engineering, model selection, and evaluation. Through this systematic approach, I identified the best performing algorithms and further enhanced their performance using ensemble techniques and hyperparameter tuning. Key findings from the analysis revealed that the XGBClassifier achieved the highest accuracy, indicating its robustness in handling complex weather patterns.

1.2 GitHub Repository

- Task 1: https://github.com/JaneeshaJ2001/Time-Series-Weather-Prediction.git
- Task 2: https://github.com/JaneeshaJ2001/Weather-Classification.git

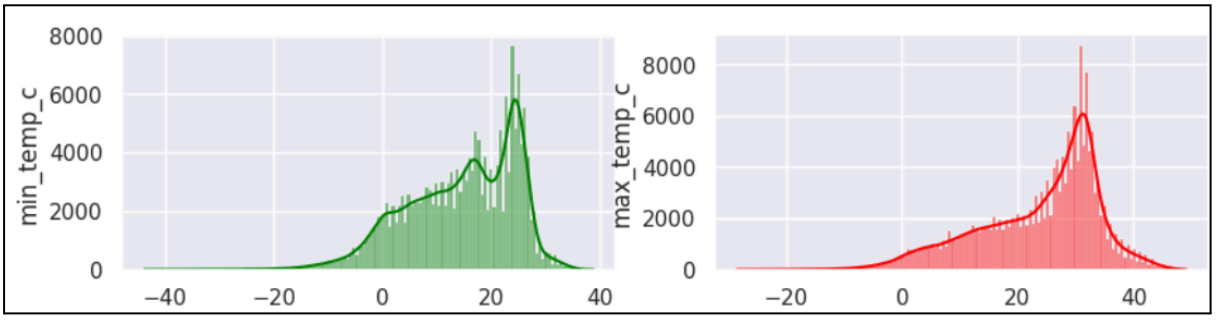
All the EDA visualizations and model trained codes for both tasks are included in the repository.

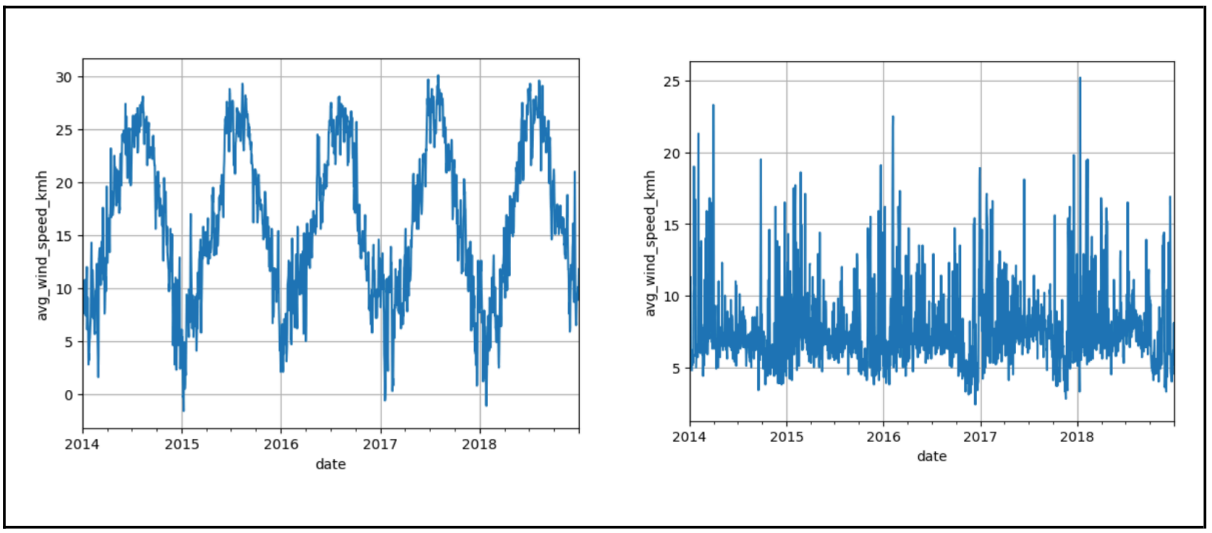
---

2. Problem 1: Time Series Prediction

2.1 Data Understanding and Preprocessing

The historical weather dataset comprises various weather attributes, including average, minimum, and maximum temperatures, precipitation, snow depth, and wind characteristics. A thorough exploration revealed missing values, which were handled using a combination of forward filling, linear interpolation, and last year's temperature values.





After visualization, it is clear that there's an annual pattern in these data. All the EDA process visualizations are included in the GitHub repository.

Steps Involved:

- **Handling Missing Values:** Missing values in the target variable (avg_temp_c) were first filled using the values from the same day in the previous year. Any remaining missing values were then filled using linear interpolation. For other features, forward filling was applied. Due to the huge percentage of missing values in 2 columns, those were removed.
- **Data Sorting:** The data was sorted by city_id and date to facilitate time series analysis.

2.2 Feature Selection and Engineering

For time series forecasting, specific features were engineered to capture temporal patterns:

Engineered Features:

- **Time Index:** A sequential time index representing each day.
- **Sinusoidal Components:** Sine and cosine transformations of the time index to model seasonal variations.

Justification:

- **Time Index:** Essential for any time series analysis to maintain the order of observations.
- **Sinusoidal Components:** Capture seasonal effects, which are significant in weather data due to annual cycles.

2.3 Model Selection and Training

The model selection process involved a custom hybrid model combining sinusoidal regression and a gradient boosting regressor. The sinusoidal regressor captures periodic trends, while the gradient boosting model handles complex patterns and residuals.

Models Used:

- **Sinusoidal Regressor:** Captures seasonal variations using sine and cosine transformations of the time index.
- **XGBoost Regressor:** Addresses residuals from the sinusoidal model and improves prediction accuracy.

Training Process:

- **Model Training:** Each city's data was used to train the models. The sinusoidal regressor was first fitted, followed by fitting the XGBoost model on the residuals.
- **Prediction:** The trained model was used to predict temperatures for the first week of 2019 for each city.

2.4 Results and Discussion

The evaluation metrics for the model's performance included Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The hybrid model demonstrated improved accuracy compared to using standalone models.

Performance Analysis:

- **MAE and RMSE:** Metrics were calculated to quantify the prediction error. The hybrid model showed lower error rates.
- **Insights:** Seasonal patterns were effectively captured by the sinusoidal components, and residual complexity was managed by the gradient boosting model.

2.5 Conclusion

The hybrid modeling approach combining sinusoidal regression and gradient boosting proved effective for time series forecasting of temperatures. Key conclusions include:

- **Effectiveness of Hybrid Models:** Combining models that capture different aspects of the data (seasonality and complex residual patterns) improves accuracy.
- **Importance of Feature Engineering:** Creating features that represent underlying patterns in the data is crucial for model performance.

As the future improvements,

Explore Advanced Models: Investigate the use of deep learning models like LSTM for capturing long-term dependencies in the data.

Cross-Validation: Implement robust cross-validation techniques to ensure the model's generalizability to unseen data.

---

3. Problem 2: Classification Problem

3.1 Data Understanding and Preprocessing

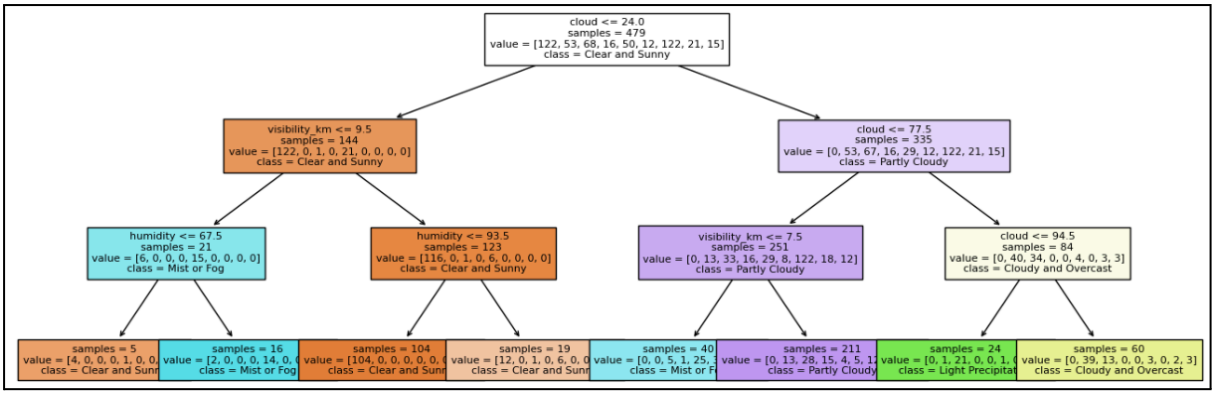
The dataset used for this analysis was `daily_data.csv`, which contained various weather-related features. Key features included temperature, humidity, wind speed, pressure, and visibility, among others. The dataset also had categorical variables such as city identifiers and textual weather conditions that were essential for classification.

Methods Used for Data Cleaning and Preprocessing:

- **Irrelevant Columns Removal:** Columns like 'sunrise', 'sunset', 'sunrise_x', 'sunset_x', 'day_id', and 'city_id' were dropped as they were not directly relevant to weather condition classification.
- **Handling Outliers:** Found extreme outliers in the precipitation column after plotting, which significantly improved model performance after handling them.
- **Datetime Conversion**: The `sunrise` and `sunset` times were converted to datetime objects to facilitate the creation of new features.

3.2 Feature Selection and Engineering

- **Creation of New Features:**
  - **Day Length**: Calculated by subtracting `sunrise` from `sunset`, providing an indication of the duration of daylight.
  - **Relative Temperature**: Defined as the difference between `feels_like_celsius` and `temperature_celsius`, capturing the perceived temperature.
  - **Wind Chill**: Estimated using a formula that accounts for the cooling effect of wind on perceived temperature.
  - **Temperature-Humidity Interaction**:Created by multiplying `temperature_celsius` and `humidity`, capturing the combined effect of these two variables.
- **Decision Tree Analysis:** Identified important predictive features using a decision tree. Drew a decision tree of depth 3 to identify the most important predictive features, discovering that `cloud`, `visibility_km`, and `humidity` were the most significant.



- **Correlation analysis** using a heatmap helped identify and remove highly correlated features such as `feels_like_celsius`, `relative_temperature`, and `wind_chill` to reduce redundancy and multicollinearity. Additionally, numerical features were **scaled using StandardScaler**, and categorical features (city identifiers) were **one-hot encoded**.

The final features selected for the classification task included:

- Temperature in Celsius
- Wind speed in kph
- Wind direction in degrees
- Atmospheric pressure in mb
- Precipitation in mm
- Humidity percentage
- Cloud cover percentage
- Visibility in km
- UV index
- Gust speed in kph
- Air quality index (US EPA)
- Day length
- Interaction between temperature and humidity

Explanation of Feature Engineering Decisions:

Feature engineering involved creating new features to capture complex relationships in the data.

- Day length feature captures the duration of daylight, which can significantly impact weather conditions and various dependent activities. For instance, longer daylight hours might be correlated with higher temperatures and specific weather patterns.
- The interaction between temperature and humidity can significantly influence weather conditions, such as the formation of dew or fog.
- The `relative_temperature` provides insight into how external factors like humidity and wind affect perceived temperature.

These feature engineering decisions were critical in refining the dataset, ensuring that the models trained on this data could learn effectively and generalize well to new, unseen data.

3.3 Model Selection and Training

After performing dataset preprocessing and feature engineering, the dataset was split into `train_df` and `test_df`, with the target variable already defined in `train_df`. To identify the best-performing algorithms, various machine learning models were trained and evaluated using cross-validation for improved accuracy. Cross-validation was employed using ShuffleSplit to ensure robust training and validation.

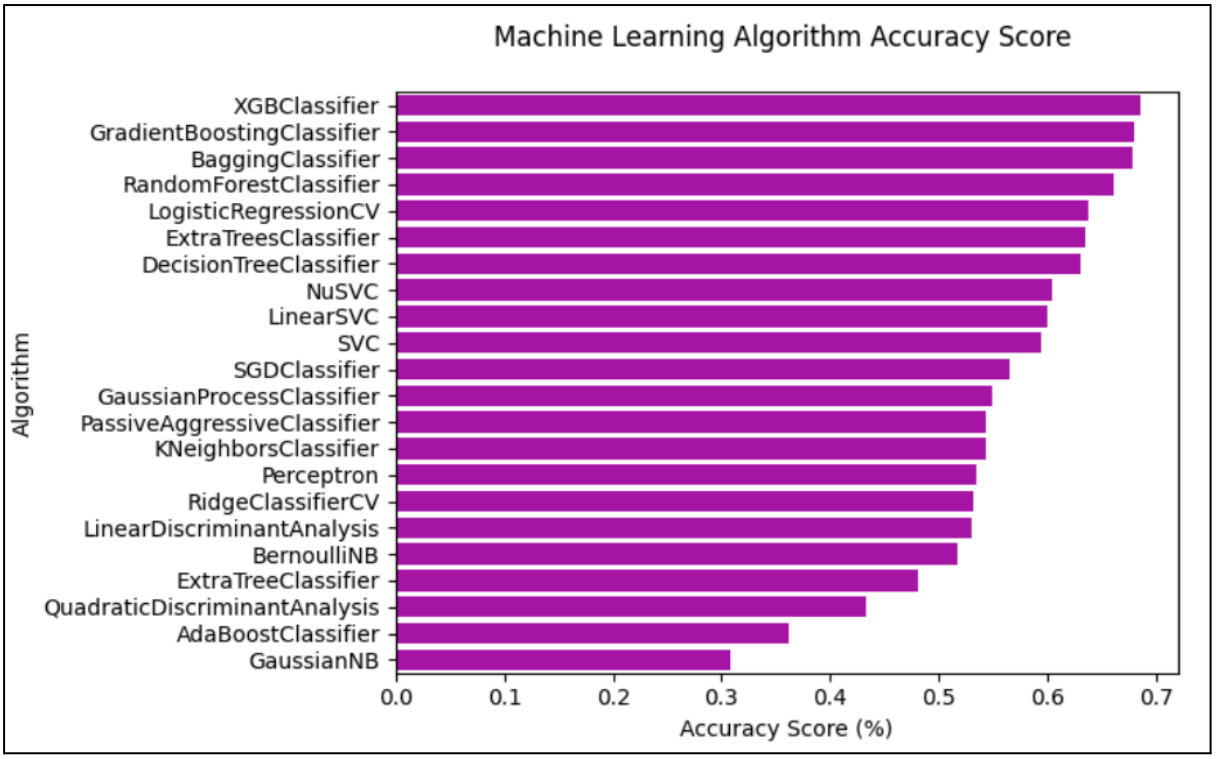
A variety of classification algorithms were evaluated, including:

- Ensemble Methods: AdaBoost, Bagging, Extra Trees, Gradient Boosting, Random Forest
- Gaussian Processes: Gaussian Process Classifier
- Generalized Linear Models: Logistic Regression, Passive Aggressive Classifier, Ridge Classifier, SGD Classifier, Perceptron
- Naive Bayes: BernoulliNB, GaussianNB
- Nearest Neighbor: K-Nearest Neighbors
- Support Vector Machines: SVC, LinearSVC
- Trees: Decision Tree, Extra Tree
- Discriminant Analysis: Linear Discriminant Analysis, Quadratic Discriminant Analysis
- XGBoost: XGBClassifier

A table was printed displaying the training and test accuracies for each algorithm.

	MLA Name	MLA Parameters	MLA Train Accuracy Mean	MLA Test Accuracy Mean	MLA Test Accuracy 3*STD	MLA Time
21	XGBClassifier	{'objective': 'binary:logistic', 'base_score':...	1.0	0.686111	0.076149	0.638637
3	GradientBoostingClassifier	{'ccp_alpha': 0.0, 'criterion': 'friedman_mse'...	1.0	0.679861	0.081463	1.78991
1	BaggingClassifier	{'base_estimator': 'deprecated', 'bootstrap': ...	0.982578	0.677778	0.067315	0.059988
4	RandomForestClassifier	{'bootstrap': True, 'ccp_alpha': 0.0, 'class_w...	1.0	0.660417	0.121995	0.279594
6	LogisticRegressionCV	{'Cs': 10, 'class_weight': None, 'cv': None, '...	0.893728	0.6375	0.115545	2.02259
2	ExtraTreesClassifier	{'bootstrap': False, 'ccp_alpha': 0.0, 'class_...	1.0	0.634722	0.084471	0.220185
17	DecisionTreeClassifier	{'ccp_alpha': 0.0, 'class_weight': None, 'crit...	1.0	0.630556	0.071443	0.02089
15	NuSVC	{'break_ties': False, 'cache_size': 200, 'clas...	0.990592	0.604861	0.092913	0.096493
16	LinearSVC	{'C': 1.0, 'class_weight': None, 'dual': True,...	0.902091	0.6	0.108733	0.121584
14	SVC	{'C': 1.0, 'break_ties': False, 'cache_size': ...	0.77561	0.59375	0.141068	0.088182
9	SGDClassifier	{'alpha': 0.0001, 'average': False, 'class_wei...	0.834495	0.564583	0.071595	0.083565

Additionally, a graph was generated to show the performance ranking of the algorithms.



- The results indicated that **XGBClassifier, Gradient Boosting Classifier, Bagging Classifier, Random Forest Classifier** and **Extra Trees Classifier** were the top performers.

Some algorithms achieved 100% training accuracy, likely due to the small size of the training dataset, indicating potential overfitting. The top five performing algorithms were selected for further analysis.

Ensemble techniques, including hard voting (majority vote) and soft voting, were applied to these selected models. The train and test accuracies for these ensemble methods were calculated and compared.

Approach to Model Parameter Tuning and Selection:

Next, hyperparameter tuning was conducted using GridSearchCV on the selected algorithms. It optimizes parameters such as learning rates, max depth, and the number of estimators to improve model performance. After tuning, hard voting and soft voting were performed again, and their train and test accuracies were recorded. Ensemble methods were combined using voting classifiers to leverage the strengths of multiple models.

3.4 Results and Discussion

Performance Metrics for Weather Condition Classification:

A final chart was created to summarize all the results. The results clearly indicated an overfitting issue, as evidenced by the disparity between training and test accuracies. To address this, soft voting with tuned hyperparameters was chosen as the best approach, as it provided a better balance between train and test accuracy, reducing the overfitting problem. Models were evaluated based on their

accuracy scores. The accuracy scores were validated using cross-validation, ensuring that the models generalized well to unseen data.

Machine Learning Technique	Train Accuracy	Test Accuracy
Hard Vote	100	70.56
Hard Vote- After Hyper parameter tuning	67.7	62.08
Soft Vote	100	70.21
Soft Vote- After Hyper parameter tuning	80.15	68.06

Analysis of Classification Results and Model Effectiveness:

The analysis showed that ensemble methods outperformed single classifiers, highlighting their ability to capture complex patterns in the data. Feature scaling significantly improved the performance of algorithms like SVM and logistic regression. The use of one-hot encoding for categorical variables also contributed to the effectiveness of the models.

3.5 Conclusion

The study explored the classification of weather conditions using various machine learning algorithms. Through detailed preprocessing, feature engineering, and model evaluation, the following conclusions were drawn:

- **Effective Feature Engineering**: Creating new features such as `day_length`, `relative_temperature` and `temperature_humidity_interaction` significantly enhanced the predictive power of the dataset.
- **Outlier Handling**: Addressing extreme outliers in the precipitation column was crucial for improving model robustness and performance.
- **Model Selection**: The ensemble methods, particularly voting with hyperparameter tuning, provided the best balance between training and test accuracy, effectively mitigating overfitting.
- **Importance of Scaling and Encoding**: Standard scaling of numerical features and one-hot encoding of categorical variables were essential preprocessing steps ensured that the models could learn effectively from the data.

Recommendations for Future Enhancements:

Increasing the size of the training dataset can help models generalize better and reduce overfitting. Future work could explore the inclusion of additional weather features, such as historical weather trends and geographical data, to further improve model accuracy. Experimenting with deep learning models could also provide insights into complex weather patterns. Additionally, deploying the model in a real-time prediction system could offer practical benefits for weather-dependent sectors.

---

4. References (If applicable)

4.1 List of References

- Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts: Melbourne, Australia. Available at: https://otexts.com/fpp3/
- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, 785-794. DOI: 10.1145/2939672.2939785
- Analytics Vidhya. Retrieved from Analytics Vidhya
- Raschka, S., & Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow (3rd ed.). Packt Publishing.