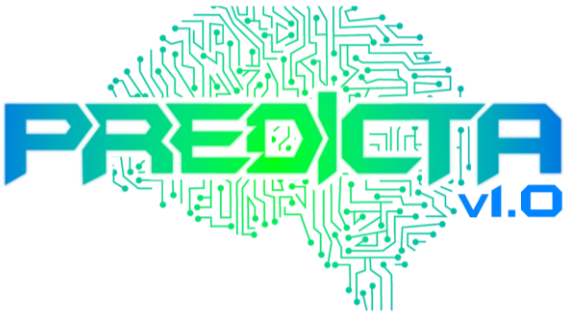

Predicta_Report_P047

PREDICTA 1.0

By OTTOON (P047)
University of Peradeniya

Table of Contents

---

1. Summary

1.1 Overview of Approach and Key Findings

Our approach to weather forecasting involved a comprehensive methodology encompassing both time series prediction and classification tasks. For time series prediction, we utilized historical weather data spanning from January 1, 2014, to December 31, 2018, across 100 cities worldwide [1].

Initially, we focused on thorough data understanding and preprocessing. This included converting date fields into datetime format and extracting additional features such as year, month, day, week, quarter, and day of the week. Handling missing data was critical; we employed mean imputation grouped by city_id, month, and day to maintain data integrity while filling gaps in the dataset. For feature engineering, we created statistical features to provide deeper insights into temperature distributions over time. Rolling window features computed to capture short-term trends, while lag features provided autocorrelation insights.

Model selection centered on the CatBoost Regressor [2], chosen for its robust handling of categorical features and complex interactions within structured data. Hyperparameter tuning using Optuna [3] optimized model performance, resulting in a competitive Root Mean Squared Error (RMSE) on the test set.

For the classification problem, we employed daily weather data [4] from daily_data.csv, spanning similar cities and variables but focusing on categorizing weather conditions into nine distinct types using the condition_text variable. Preprocessing involved selecting relevant features such as temperature, wind speed, precipitation, humidity, and air quality, and creating new features like daylight hours and interaction terms to capture nuanced weather patterns. Data transformations are done for better categorization and standardizing continuous variables. Various classifiers, including CatBoostClassifier , XGBoostClassifier, and RandomForestClassifier [5], were evaluated, with Optuna facilitating hyperparameter tuning to enhance model accuracy. An ensemble [6] approach using a Voting Classifier improved prediction robustness, achieving high classification accuracy. Insights gleaned from this approach highlighted the importance of feature engineering and model ensemble techniques in achieving accurate weather condition classification.

1.2 GitHub Repository

https://github.com/NishaniKasineshan/Predicta-1.0

---

2. Problem 1: Time Series Prediction

2.1 Data Understanding and Preprocessing

The historical_weather.csv, the historical weather dataset for 100 cities worldwide from January 1, 2014, to December 31, 2018, contains following columns:

- **city_id**: Unique identifier for each city.
- **date**: Date of the weather observation.
- **avg_temp_c**: Average temperature in Celsius.
- **min_temp_c**: Minimum temperature in Celsius.
- **max_temp_c**: Maximum temperature in Celsius.
- **precipitation_mm**: Precipitation in millimeters.
- **snow_depth_mm**: Snow depth in millimeters.
- **avg_wind_dir_deg**: Average wind direction in degrees.
- **avg_wind_speed_kmh**: Average wind speed in kilometers per hour.

The goal is to forecast the average temperatures for the first week of 2019 across 100 cities worldwide using historical weather data from January 1, 2014, to December 31, 2018. The target variable is **avg_temp_c**, a continuous variable.

The dataset was grouped based on city_id & date with mean aggregation of values.

Given the nature of time series data, the date variable is converted from object type to datetime type for further data transformations on it. We transformed date data into additional features such as year, month, day, week, quarter, and day of the week because they are crucial for extracting meaningful insights like seasonal trends, monthly, weekly patterns from time series data like historical weather records.

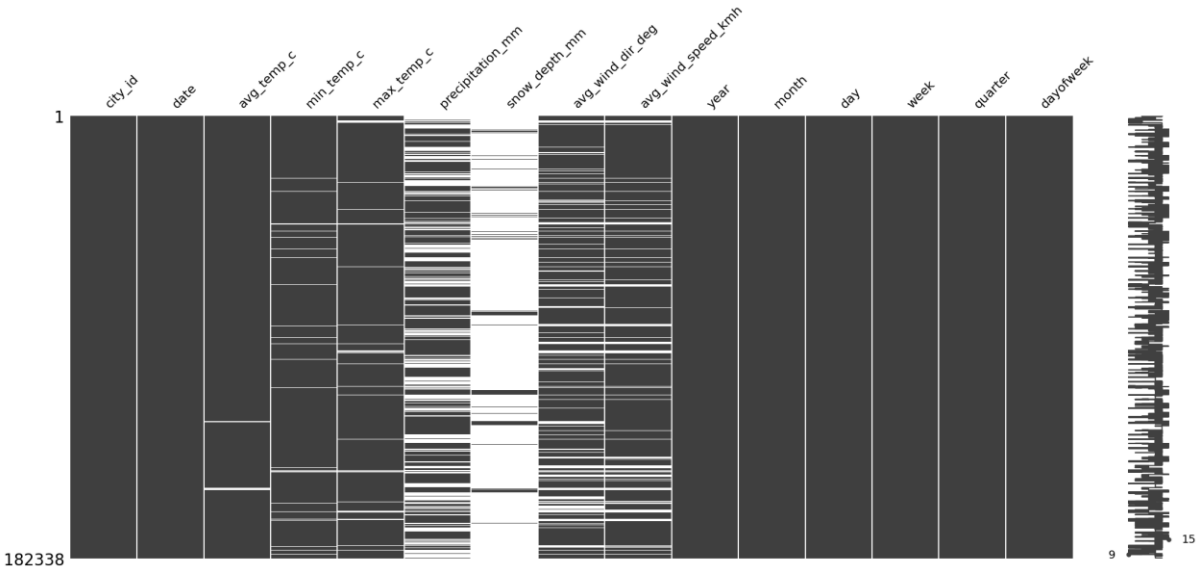

Fig 1. Visualization of missing values [7] across the dataset

We checked for missing values because missing values can significantly impact the accuracy and reliability of predictions, especially in time series data where continuity is crucial. As shown in Fig 1., it can be seen that features like **min_temp_c,max_temp_c, precipitation_mm,snow_depth_mm,avg_wind_speed_kmh,avg_wind_dir_deg including the target variable avg_temp_c.** Since there will be huge loss of necessary data, we didn't drop the missing data. In order to impute the missing values, first, we filled the null values based on feature respective mean values grouped based on city_id, month and day. This will ensure the month and day pattern across cities are distinctly aggregated over time. Next, forward fill method was used to fill the null values based on the last non-null value which ensures the continuity of data points. Finally, the rest of the null values were filled with 0 because they do not show any seasonal pattern. The dataset didn't contain any duplicated data.

2.2 Feature Selection and Engineering

Feature engineering techniques like data transformation, statistical data creation, lag features and window features, encoding of date related data, standardization scaling of continuous variables and dummy encoding [8] for categorical features were carried out to enhance the model's predictability of average temperature across cities. The process steps involved,

- Created Q1_temp_c,Q3_temp_c,mean_temp_c statistical features based on min_temp_c,max_temp_c to provide insights about the temperature distribution and central tendency to the model.
- Created rolling window features for continuous variables computing aggregated statistics like mean and standard deviation over preceding 3 days and 7 days grouped based on city_id,year and month. These rolling window features enables the model to capture cyclic behavior and trends in weather data.
- Created lag features for temporal variables with lags upto 7 days by shifting backward in time to capture the past values of the variables to encode trends and autocorrelation patterns over time.
- All the continuous variables were standardized to ensure equal contribution of features to the model.
- Encoded month, dayofweek, ,week, quarter into sine and cosine transformations to capture the cyclic trend of historical data.
- Performed dummy encoding on the categorical feature city_id to capture the information specific to each city.

2.3 Model Selection and Training

The dataset was split into two parts: data before December 23, 2018, for training, and data from December 23, 2018, onward for testing. This ensures that our model is trained on historical data and evaluated on more recent observations. For this weather forecasting task, we used the CatBoost Regressor, a powerful gradient boosting algorithm that handles categorical features automatically and provides strong performance on structured data.

CatBoost can effectively capture complex patterns in the weather data, making it a potential model for accurate weather forecasts.

To optimize the model, we employed Optuna for hyperparameter tuning. Optuna is a flexible and efficient hyperparameter optimization framework that uses a combination of Bayesian optimization and other advanced search techniques.

2.4 Results and Discussion

After training the CatBoost Regressor on the historical weather data and optimizing hyperparameters using Optuna, we evaluated the model's performance on the test set. The Root Mean Squared Error (RMSE), which measures the difference between the predicted and actual average temperatures, was computed to assess the model's accuracy. In this case, the test RMSE was found to be 0.9366. The CatBoost Regressor provides feature importance scores, indicating the impact of each feature on the model's predictions. Temperature statistical variables, window rolling features, lag features are identified as the top features by the model to highlight the importance of historical temperature metrics, trends & patterns, in accurately predicting average temperatures.

2.5 Conclusion

In tackling the weather forecasting challenge, we leveraged the CatBoost Regressor to predict average temperatures across 100 cities for the first week of 2019 using historical weather data. By splitting the dataset into training and testing sets at December 23, 2018, and optimizing the model's hyperparameters with Optuna, we ensured a robust and effective training process. In light of above, incorporating interaction features among the variables will account for combined effect of the variables on the average temperature and combining multiple forecasting models in an ensemble approach will enhance predictive accuracy and robustness, leveraging the strengths of different algorithms.

---

3. Problem 2: Classification Problem

3.1 Data Understanding and Preprocessing

The daily_data.csv, daily weather dataset contains the following columns:

- **day_id**: Unique identifier for each day.
- **city_id**: Identifier for each city.
- **temperature_celsius**: Temperature in Celsius.
- **condition_text**: Text description of the weather condition (target variable).
- **wind_kph**: Wind speed in kilometers per hour.
- **wind_degree**: Wind direction in degrees.
- **pressure_mb**: Atmospheric pressure in millibars.
- **precip_mm**: Precipitation in millimeters.
- **humidity**: Humidity percentage.
- **cloud**: Cloud cover percentage.

- **feels_like_celsius**: Feels-like temperature in Celsius.
- **visibility_km**: Visibility in kilometers.
- **uv_index**: UV index.
- **gust_kph**: Wind gust speed in kilometers per hour.
- **air_quality_us-epa-index**: Air quality index based on US EPA standards.
- **sunrise**: Time of sunrise.
- **sunset**: Time of sunset.

The target variable in this weather classification problem is **condition_text**. It is a categorical variable that represents the weather condition for each day in 100 cities into nine types: Clear and Sunny, Partly Cloudy, Light Precipitation, Cloudy and Overcast, Mist or Fog, Rain Showers, Light Rain with Thunder, Thunderstorms, Moderate to Heavy Rain. The provided dataset didn't have any missing values nor duplicated data.

3.2 Feature Selection and Engineering

Based on domain knowledge, following features were selected:

- **temperature_celsius & feels_like_celsius**: These features are crucial as they can differ significantly based on humidity and wind conditions.
- **wind_kph, wind_degree & gust_kph**: These features help to distinguish between calm and stormy conditions.
- **pressure_mb & precip_mm**: These are strong indicators of different weather patterns, especially for precipitation-related conditions.
- **humidity & cloud**: These features are critical for distinguishing between clear, cloudy, and foggy conditions.
- **visibility_km**: This directly impacts classifications involving mist or fog.
- **uv_index**: This feature is useful for identifying clear and sunny days.
- **air_quality_us-epa-index**: This feature helps to differentiate conditions like clear skies from hazy or foggy conditions due to pollutants.
- **sunrise & sunset**: These features can be used to calculate the length of daylight hours.

The feature engineering process involved creating new meaningful & informative features based on the given features by performing data transformations & statistical computations to improve the model's ability to accurately classify the weather conditions. The steps involved:

- Conversion of time related features **sunrise & sunset** into hours and created new feature **daylight_hours** to measure the length of daylight hours by calculating the difference between sunrise and sunset hours.
- Created a new feature **temperature_diff** to measure the variation between actual temperature in Celsius and the feel like temperature in Celsius.
- Combined the wind direction and speed components in order to get the wind speed vectors which will provide the combined effect information of the wind components.
- Binned the **temperature_celsius** into **"cold" (below 0°C), "mild" (0-15°C), "hot" (above 15°C)** , common human perceptions of temperature ranges and their effects on weather conditions.
- Binned the **air_quality_us-epa-index** into **"good" (below 2), "moderate" (3-5), "unhealthy" (>5)** to get clear distinctions in air quality impacts.
- Created interaction terms between several features to capture the potential non-linear relationships and account for combined effects on weather condition classification

task. (eg. humidity & temperature, temperature & pressure, cloud & visibility, precipitation & humidity).

- Created statistical features for each row of data combining all features to get their combined effect impact on weather conditions. (eg. mean, standard deviation, minimum, maximum, median, 1st quartile, 3rd quartile, inter quartile range)
- Perform standardized scaling on the continuous variables to ensure equal contribution during model training.
- Encoded categorical variables like **city_id ,tempearture_bin,air_quality_category** using dummy encoding.

3.3 Model Selection and Training

The model selection and training process began by splitting the dataset based on the availability of the target variable, **condition_text**. Records where **condition_text** was not null were designated as the training dataset, while those with null values formed the test dataset. The training dataset was further divided into **X_train and y_train** for model training, and **X_val** and **y_val** for validation purposes.

For model selection, we utilized a variety of classifiers, including **CatBoostClassifier, XGBoostClassifier, HistGradientBoostingClassifier, RandomForestClassifier, GradientBoostingClassifer, AdaBoostClassifier, DecisionTreeClassifier, ExtraTreesClassifier, & LGBMClassifier**.[5] We employed **Optuna** for hyperparameter tuning, leveraging its efficient search algorithm to find the optimal parameters for each model.

Optuna's hyperparameter tuning involved defining the search space for each model's parameters and running multiple trials to identify the best combination. For example, parameters like learning rate, max depth, number of estimators, and subsample ratios were fine-tuned for gradient boosting models, while max features and criterion were tuned for tree-based models.

After individual models were fine-tuned and validated, we employed an ensemble approach using a **VotingClassifier** with soft voting. This method combines the predicted probabilities of the individual classifiers to make the final prediction, which often improves accuracy by leveraging the strengths of each model. The ensemble model thus provided a more robust and reliable prediction by averaging the predictions of the diverse set of classifiers.

3.4 Results and Discussion

Accuracy was the primary evaluation metric used to measure the proportion of correctly predicted weather conditions out of the total predictions made. This metric was chosen for its straightforward interpretation and relevance to the classification task. The accuracy of each individual model, as well as the final blended ensemble model, was assessed to ensure robust performance and to demonstrate the effectiveness of combining multiple models to improve the prediction accuracy. The accuracies of the individual models and the ensemble model are shown in Fig 2.

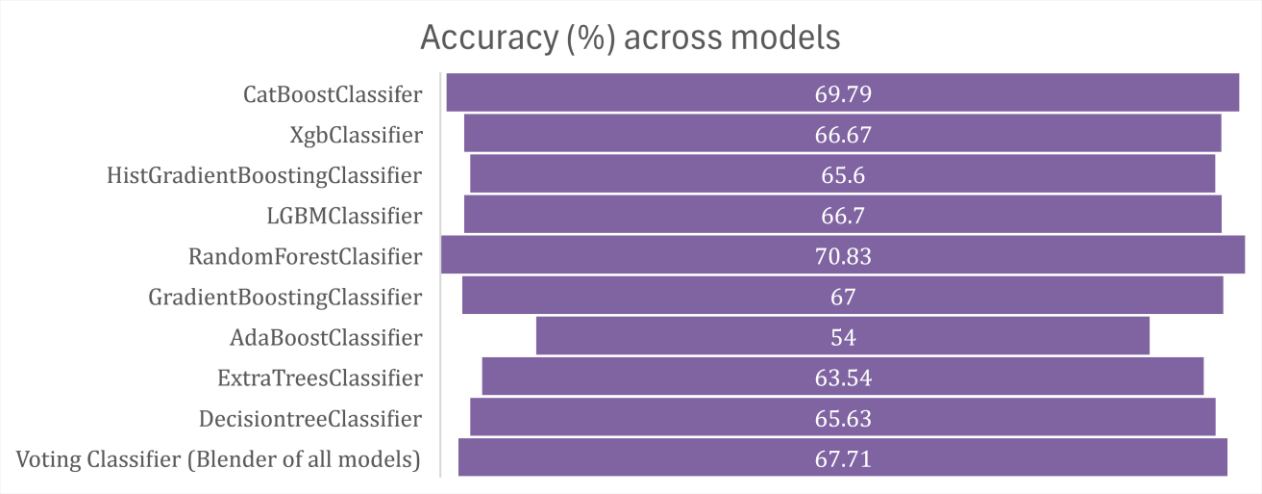

Fig 2. Performance comparison on the individual models & the blended model

The accuracy (%) results of various classifiers for weather condition classification provide valuable insights into their performance based on the dataset: RandomForestClassifier emerges as the top performer among individual models, showcasing its strength in capturing complex relationships within the data and handling noise effectively. CatBoostClassifier follows closely behind, leveraging its capability to handle categorical features robustly without extensive preprocessing, which is advantageous in weather prediction scenarios where categorical data such as city identifiers play a crucial role. GradientBoostingClassifier & LGBMClassifier perform consistently well, showcasing the reliability of gradient boosting techniques in capturing intricate patterns in the dataset. XGBClassifier, HistGradientBoostingClassifier, DecisionTreeClassifier, and ExtraTreesClassifier also provide competitive accuracies, each demonstrating varying degrees of effectiveness in handling different aspects of the weather condition classification task. AdaBoostClassifier shows comparatively lower accuracy, indicating potential challenges in adapting to the dataset's complexity or optimizing performance compared to other boosting algorithms.

3.5 Conclusion

Through experimentation, it can be seen that RandomForestClassifier demonstrated the highest accuracy among individual models, highlighting its efficacy in capturing complex weather patterns. Ensemble methods, particularly the Voting Classifier, further improved accuracy by leveraging diverse model strengths. Advanced feature engineering techniques like exploring more sophisticated features such as temporal patterns, geographical influences, or interaction effects between weather variables could provide deeper insights into weather dynamics. Further exploration of ensemble methods, including stacking or weighted averaging, could potentially yield higher predictive accuracy by combining complementary aspects of individual models. By addressing these recommendations, future enhancements can strengthen to better understanding and prediction of weather patterns to build highly accurate weather condition classification model.

4. References

4.1 List of References

[1] “Predicta 1.0: Predict the Unpredictable | Kaggle.” https://www.kaggle.com/competitions/predicta-1-0-predict-the-unpredictable.

[2] “CatBoostRegressor |.” https://catboost.ai/en/docs/concepts/python-reference_catboostregressor

[3] Y. Lim, “State-of-the-Art Machine Learning Hyperparameter Optimization with Optuna,” Medium, Apr. 08, 2022. Available: https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1

[4] “Predicta 1.0: Classify the Weather | Kaggle.” https://www.kaggle.com/competitions/predicta-1-0-predict-the-unpredictable-part-2

[5] “1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking,” Scikit-learn. https://scikit-learn.org/stable/modules/ensemble.html

[6] P. Shukla and P. Shukla, “How blending technique Improves machine learning model’s Performace - Dataaspirant,” Dataaspirant - A Data Science Portal For Beginners, Oct. 03, 2023. https://dataaspirant.com/blending-technique-machine-learning/

[7] “missingno,” PyPI, Feb. 26, 2023. https://pypi.org/project/missingno/

[8] Christopherfielding, “Feature scaling and dummy encoding, and their application in Scikit Learn,” Medium, Jan. 04, 2022. [Online]. Available: https://medium.com/@christopherfielding/feature-scaling-and-dummy-encoding-and-their-application-in-scikit-learn-a0c22ab8d3e1