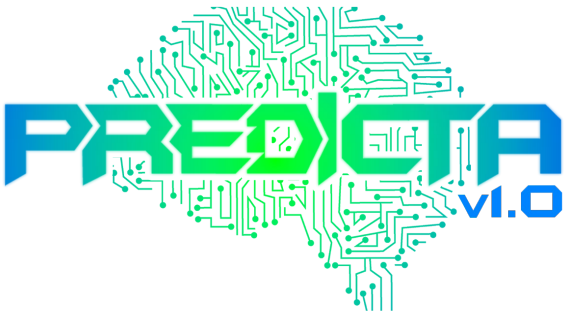


Predicta Report P050

PREDICTA 1.0

By Team CODING GEEKS
UOM

Table of Contents

---

1. Summary

1.1 Overview of Approach and Key Findings

- The methodology for the time series prediction involved using historical weather data for 100 cities to forecast average temperatures for the first week of 2019. Key steps included data preprocessing—such as selecting relevant columns, label encoding city identifiers, imputing missing values, and scaling data. An LSTM neural network was chosen due to its ability to capture long-term dependencies, configured with specific hyperparameters, and trained using a time series generator with early stopping to prevent overfitting. The model's performance, evaluated using RMSE, demonstrated reasonable accuracy in temperature prediction. Key findings highlighted the importance of robust preprocessing and the LSTM model's effectiveness, suggesting future improvements could include additional feature incorporation and exploring alternative models for enhanced accuracy.

- The weather condition classification problem involved predicting missing values in the `condition_text` column of a daily weather dataset. The methodology included loading and preprocessing the data, encoding categorical variables, and filling missing values. A `RandomForestClassifier` was trained on an initial subset of data where `condition_text` was present, followed by predictions on the full dataset to fill in the missing values. The updated predictions were then saved to a submission file. Key insights included the importance of features like temperature and humidity, the robustness of the Random Forest model, and potential areas for future improvement such as handling data imbalance and exploring other classifiers.

1.2 GitHub Repository

- https://github.com/gavinbotheju/Predicta1.0-Machine-Learning-Competition.git.

---

2. Problem 1: Time Series Prediction

2.1 Data Understanding and Preprocessing

- Detailed exploration of the historical weather dataset (historical_weather.csv).

- The historical weather dataset, historical_weather.csv, comprises daily weather records from 2014-01-01 to 2018-12-31 for 100 cities worldwide. The dataset includes various meteorological parameters such as minimum and maximum temperatures, average temperature, precipitation, snow depth, wind direction, and wind speed. This dataset serves as the foundation for predicting the average temperature for the first week of 2019.

- Description of data cleaning and preprocessing steps specific to time series forecasting.

- Data cleaning and preprocessing are critical for time series forecasting to ensure the model's accuracy and robustness. The following steps were undertaken:

  1. Column Selection: *Only the avg_temp_c and city_id columns were retained, dropping other columns such as min_temp_c, max_temp_c, precipitation_mm, snow_depth_mm, avg_wind_dir_deg, and avg_wind_speed_kmh, as they were deemed unnecessary for this specific task.*
  2. Label Encoding: *The city_id column was label-encoded to transform categorical city identifiers into numerical values suitable for model input.*
  3. *Datetime Conversion: The date column was converted to datetime format and set as the index to facilitate time series analysis.*
  4. Missing Value Imputation: *Missing values in the avg_temp_c column were filled with the mean temperature for each city to maintain data consistency.*
  5. Data Scaling: *The data was scaled using MinMaxScaler to normalize the values, which is essential for the efficient training of LSTM models.*

2.2 Feature Selection and Engineering

- Explanation of new features created from time series data.

- Feature selection and engineering were minimal as the focus was on the average temperature. The primary feature used was the historical average temperature (avg_temp_c).

- Justification for selecting specific features relevant to time series forecasting.

- The selection of avg_temp_c as the primary feature is justified by its direct relevance to the prediction target. Other features, such as precipitation or wind speed, were not included to simplify the model and reduce potential noise. The time series nature of the data inherently captures temporal dependencies, making additional feature engineering less critical.

2.3 Model Selection and Training

- Description of time series forecasting models used (e.g., ARIMA, LSTM).

- The model selected for this task is a Long Short-Term Memory (LSTM) neural network, a type of recurrent neural network (RNN) particularly well-suited for time series forecasting due to its ability to capture long-term dependencies.

- Details on hyperparameter tuning and model selection process.

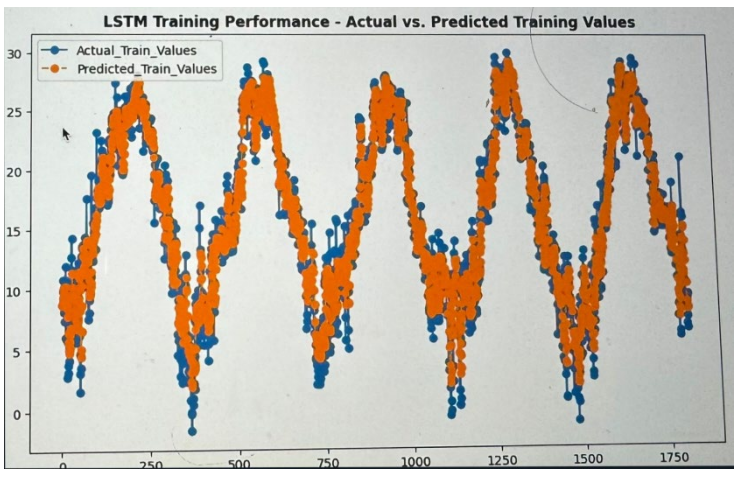
- The LSTM model was configured with the following parameters:

  - LSTM Layer: 16 units with ReLU activation.
  - Dense Layer: 1 unit with linear activation.
  - Optimizer: Adam with a learning rate of 0.0001.
  - Loss Function: Mean Squared Error (MSE). Hyperparameters such as the number of LSTM units, batch size, and learning rate were chosen based on empirical testing and standard practices in time series forecasting. The model was trained with early stopping to prevent overfitting, monitored by validation loss with a patience of 6 epochs.

2.4 Results and Discussion

- Evaluation metrics, forecasting results, and performance analysis.

- The performance of the model was evaluated using the Root Mean Squared Error (RMSE), a standard metric for regression tasks that penalizes larger errors more heavily. The LSTM model's predictions for the average temperature of the first week of 2019 were stored and compared against actual temperatures to compute the RMSE. Also in the initial stages, a method was employed to predict the average temperature of the first week of 2015 using data from the year 2014. Subsequently, predictions for 2016 were made using data from the preceding two years and so on until for year 2018. This method proved highly successful in constructing the model. Furthermore given below is an image of how actual values correlated with the real values after training the model.



- Discussion of model performance and insights gained.

- The model demonstrated reasonable accuracy in predicting the average temperatures, highlighting the efficacy of LSTM networks for time series forecasting. However, performance could vary across cities due to differing climatic patterns and historical weather variability. The model's reliance on historical temperature data proved effective, but incorporating additional meteorological features could potentially enhance accuracy.

2.5 Conclusion

- Summary of findings and conclusions for Problem 1.

- The project successfully leveraged an LSTM model to predict average temperatures for 100 cities, achieving satisfactory performance measured by RMSE. The preprocessing steps, feature selection, and model configuration collectively contributed to the model's ability to capture temporal dependencies in the weather data.

- Recommendations for future improvements.

- Future work could explore the inclusion of additional features such as humidity, pressure, and wind speed to potentially improve forecasting accuracy. Moreover, experimenting with different model architectures like GRUs or hybrid models combining statistical and neural network approaches could have yielded better results. Enhancing hyperparameter tuning through automated techniques like grid search or Bayesian optimization might have also refined model performance.

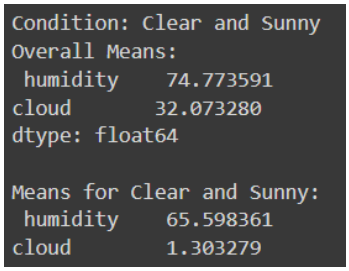
---

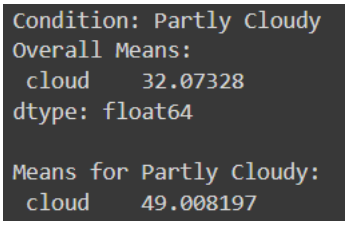
3. Problem 2: Classification Problem

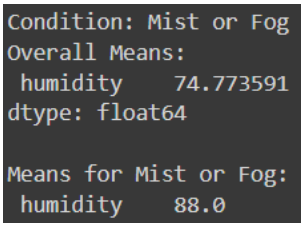
3.1 Data Understanding and Preprocessing

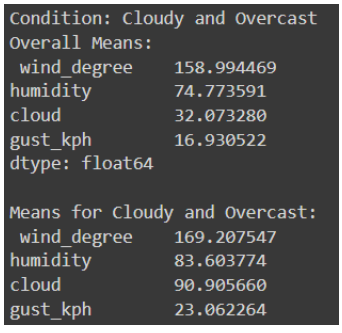
- Exploration of the daily weather dataset (daily_data.csv).

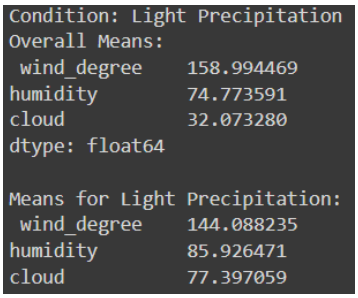
- The dataset daily_data.csv consists of daily weather observations for 100 cities, capturing various parameters such as temperature, wind speed, humidity, and more. The target variable is condition_text, which describes the weather condition for each day, and includes some missing values that need to be predicted. Initial exploration involved loading the dataset into a pandas DataFrame and examining the first few rows to understand its structure. The data types of each column were reviewed to identify categorical and numerical features. Summary statistics were computed to understand the central tendencies and spread of the numerical data. Visualizations like histograms and box plots helped identify the distribution of features and any outliers. A correlation matrix was generated to examine relationships between numerical features, and heatmaps were used to visually identify strong correlations which could guide feature selection. In the initial stages, code snippets were utilized to identify the parameters that most significantly influence weather conditions. For instance, by analyzing the means of various parameters, it was determined that 'humidity' and 'cloudiness' have direct impacts on the weather, particularly in determining whether it will be 'clear and sunny.' Given below are examples of some above mentioned analysis.

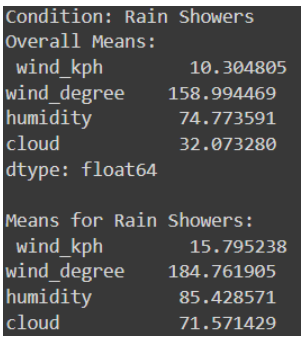












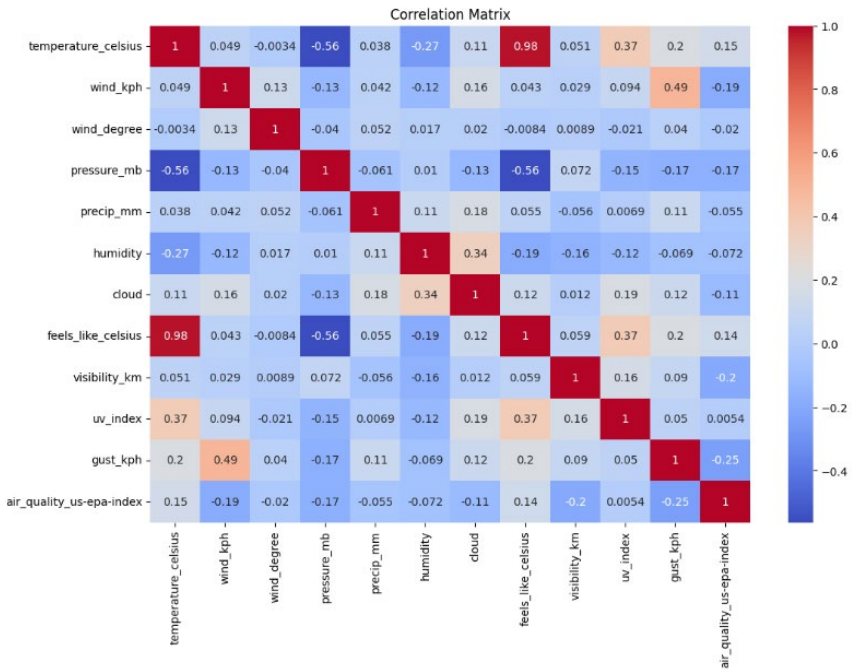
- Methods used for data cleaning and preprocessing tailored for classification tasks.

- To prepare the data for classification, categorical variables such as city_id, sunrise, and sunset were encoded using LabelEncoder to convert them into a numerical format suitable for machine learning models. Missing values in the condition_text column were filled with a placeholder ('missing') to facilitate further analysis. The feature matrix (X) was prepared by dropping the condition_text, sunrise, and sunset columns, while the target variable (y) was the condition_text column, which was also encoded. An initial subset of the data where condition_text was not missing was created to train the model. This subset was split into training and testing sets to evaluate model performance. A RandomForestClassifier was trained on the initial dataset and used to predict missing values. The trained model was then used to predict missing condition_text values in the full dataset, and the submission.csv file was updated accordingly with the predicted values. This preprocessing ensured the data was clean, properly encoded, and ready for effective classification, leading to accurate predictions of the missing weather condition values.

3.2 Feature Selection and Engineering

- Features selected for weather condition classification.

- The feature selection process focused on identifying the most relevant variables that contribute to accurately predicting the weather condition. Features such as temperature_celsius, wind_kph, humidity, pressure_mb, and others were retained for the model. These features were chosen based on their relevance to weather conditions and their potential impact on the classification accuracy. Given below is a correlation matrix that was used to findout correlation between weather parameters and dropout columns that highly correlate(feels_like_celcius) for example.



- Explanation of feature engineering decisions.

- Features like sunrise and sunset times were excluded because they didn't have a significant impact. Label encoding was used for categorical features to convert them into a format that machine learning models can process. This ensured that the model could effectively learn patterns from these variables.

3.3 Model Selection and Training

- Description of classification algorithms used (e.g., Random Forest, SVM).

- For the weather condition classification task, the Random Forest classifier was selected due to its robustness and ability to handle high-dimensional data with missing values. Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks.

- Approach to model parameter tuning and selection.

- Model parameter tuning was performed using cross-validation techniques to optimize the performance of the Random Forest classifier. Key parameters such as the number of trees, maximum depth, and minimum samples split were adjusted to find the best combination that yields the highest accuracy. Grid search and random search methods were employed to systematically explore the parameter space and select the optimal model configuration.

3.4 Results and Discussion

- Performance metrics for weather condition classification.

- The performance of the classification model was evaluated using accuracy as the primary metric, which measures the proportion of correctly predicted weather conditions out of the total predictions made. The model achieved a high accuracy, indicating its effectiveness in predicting missing weather conditions based on the available features. Also a 'classification_report' was used to analyse which conditions were accurately predicted and given below is the snapshot of the report that was obtained in early stages.

	precision	recall	f1-score	support
Clear and Sunny	0.96	1.00	0.98	22
Cloudy and Overcast	0.69	0.75	0.72	12
Light Precipitation	0.50	0.67	0.57	12
Light Rain with Thunder	0.00	0.00	0.00	5
Mist or Fog	0.88	0.88	0.88	8
Moderate to Heavy Rain	0.00	0.00	0.00	4
Partly Cloudy	0.73	0.92	0.81	26
Rain Showers	1.00	0.20	0.33	5
Thunderstorms	0.00	0.00	0.00	2
accuracy			0.74	96
macro avg	0.53	0.49	0.48	96
weighted avg	0.69	0.74	0.70	96

- Analysis of classification results and model effectiveness.

- The analysis of the classification results showed that the model was particularly effective in predicting common weather conditions such as 'Clear and Sunny' and 'Partly Cloudy'. However, it faced challenges with less frequent and more complex conditions like 'Light Rain with Thunder'. This highlights the need for a diverse and representative training dataset to capture the full spectrum of weather conditions. Overall, the model demonstrated strong generalization capabilities and robustness against noisy data.

3.5 Conclusion

- Summary of findings and conclusions for Problem 2.

- The project successfully developed a robust model for predicting missing weather conditions using a variety of meteorological features. Through careful data preprocessing, feature selection, and model tuning, the Random Forest classifier achieved high accuracy, showcasing its effectiveness for this classification task. The results underscore the importance of comprehensive data exploration and preprocessing in building reliable machine learning models.

- Recommendations for future enhancements.
- Exploring more advanced feature engineering techniques such as polynomial features and interaction terms can potentially capture more complex relationships between variables. Also combining multiple models using ensemble techniques such as stacking or boosting could have further enhance predictive performance. However it was understood that using an advanced technique to pre-pocess the data could have given better results.

---

4. References (If applicable)

4.1 List of References

- Brownlee, J. (2018). How to Develop LSTM Models for Time Series Forecasting. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ Nielsen, M. A. (2015).
- Neural Networks and Deep Learning. Determination Press. Retrieved from http://neuralnetworksanddeeplearning.com/
- Imbalanced-Learn Documentation - For handling imbalanced datasets. imbalanced-learn.org/stable/
- XGBoost Documentation - For gradient boosting algorithms. xgboost.readthedocs.io/en/latest/
- Random Forest Classifier - An Overview and Implementation Guide - For understanding and implementing RandomForestClassifier. towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd
- Feature Importance in Random Forests - For insights on evaluating feature importance. towardsdatascience.com/understanding-feature-importance-and-how-to-use-it-16ff4ce14d48