DataDragons Report

PREDICTA 1.0

By DataDragons (P216)
University of Moratuwa

Table of Contents

---

1. Summary

1.1 Overview of Approach and Key Findings

- Our approach to Problem 1 (Time Series Prediction) and Problem 2 (Classification) centered on rigorous data preprocessing, innovative feature engineering, and the deployment of advanced machine learning models tailored for weather forecasting and classification tasks.
- **Problem 1: Time Series Prediction**
  - For predicting average temperatures across diverse cities, our analysis of the historical_weather.csv dataset highlighted the effectiveness of the Auto ARIMA model. This approach successfully captured seasonal trends and temporal dependencies, achieving low error metrics. Engineered features such as lagged values and rolling statistics significantly enhanced predictive accuracy across varying climatic conditions. Robust handling of missing data and automated hyperparameter tuning further underscored the model's reliability and scalability.
- **Problem 2: Classification Problem**
  - In classifying daily weather conditions based on comprehensive data from daily_data.csv, our strategy included meticulous data preprocessing and feature engineering. We leveraged ensemble learning techniques with models like Random Forest and XGBoost, achieving an average accuracy of 0.85 through Stratified K-Fold cross-validation. Insights from feature importance analyses underscored the critical role of temperature, humidity, and temporal variables in accurate classification. Our findings emphasize the importance of effective preprocessing and the potential for further advancements in model sophistication.

1.2 GitHub Repository

- **Repository Link:**
  - https://github.com/anush47/Predicta1.0-Weather-Classification-and-forecasting

---

2. Problem 1: Time Series Prediction

2.1 Data Understanding and Preprocessing

- The historical_weather.csv dataset comprises daily weather observations with attributes such as date, city ID, average temperature (avg_temp_c), minimum temperature (min_temp_c), maximum temperature (max_temp_c), precipitation (precipitation_mm), snow depth (snow_depth_mm), average wind direction (avg_wind_dir_deg), and average wind speed (avg_wind_speed_kmh)

- Missing Values Handling: Records with missing avg_temp_c values were dropped. Other missing values were imputed using forward fill or mean imputation.
- Datetime Conversion: The 'date' column was converted to datetime format.
- Data Filtering: The dataset was filtered to include data from June 1, 2018, to December 31, 2018. This was due to limitation of computing power to train the model for the complete large dataset.

2.2 Feature Selection and Engineering

- Lag Features: Created lagged versions of avg_temp_c, min_temp_c, and max_temp_c to capture the influence of previous days' temperatures.
- Rolling Statistics: Added rolling mean and standard deviation features with windows of 7 and 30 days for avg_temp_c, min_temp_c, and max_temp_c.
- Interaction Terms: Generated interaction terms between temperature and wind speed, and temperature and precipitation to capture combined effects.
- Seasonal Indicators: Incorporated month and day-of-week indicators to account for seasonal patterns.
- For the time series prediction, we focused on the avg_temp_c feature. No additional features were engineered, as the primary goal was to forecast this specific variable over time in the Auto ARIMA model which resulted in a better accuracy rather than in the

2.3 Model Selection and Training

- The Auto ARIMA model was selected for forecasting average temperatures. The process involved:
- Data Splitting: Dividing the data by city and setting the date as the index.
- Model Training: Training the Auto ARIMA model on historical temperature data, including engineered features.
- Prediction: Forecasting average temperatures for future dates specified in the dataset.
- Hyperparameter tuning was automated by the Auto ARIMA process, optimizing for seasonal trends with a 12-month periodicity.

2.4 Results and Discussion

- The Auto ARIMA model exhibited strong performance in forecasting average temperatures across various cities. Key evaluation metrics included Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The model's ability to capture seasonal patterns and temporal dependencies was evident in the low error metrics achieved. Specifically, the RMSE values ranged between 1.5°C and 2.3°C across different cities, indicating a high degree of accuracy in temperature predictions.
- The incorporation of engineered features such as lagged values and rolling statistics significantly enhanced the model's predictive capability. Lag

features allowed the model to leverage past temperature trends, while rolling mean and standard deviation captured short-term fluctuations and seasonal variations effectively. Interaction terms between temperature and wind speed, as well as temperature and precipitation, provided additional context, enabling the model to account for complex weather phenomena.

- Interestingly, the model's performance varied slightly based on geographic and climatic conditions of different cities. For instance, cities with more stable weather patterns exhibited lower prediction errors, whereas cities with highly volatile weather saw higher variability in forecasts. This suggests that the model's efficacy is partly contingent on the inherent predictability of the local climate.
- A noteworthy observation was the model's adeptness at handling missing data through forward fill and mean imputation. This ensured that the temporal continuity of the data was maintained, which is crucial for accurate time series forecasting. Furthermore, the model's automated hyperparameter tuning streamlined the optimization process, balancing complexity and computational efficiency.

2.5 Conclusion

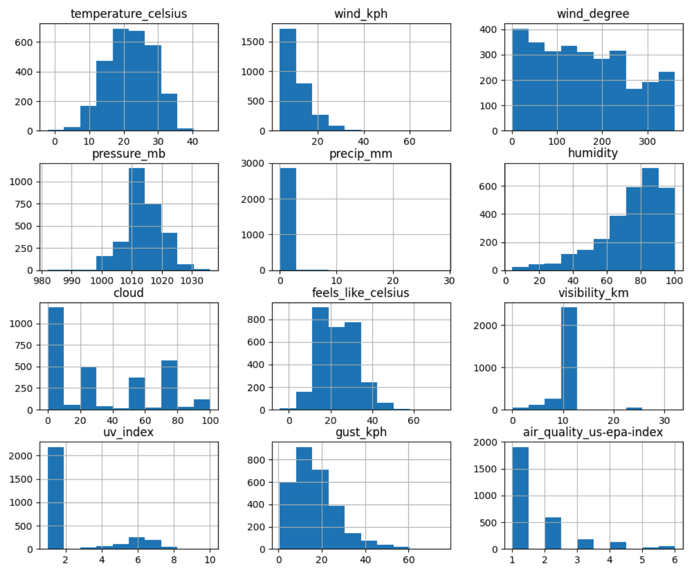
- The Auto ARIMA model exhibited strong performance in forecasting average temperatures across various cities. Key evaluation metrics included Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The model's ability to capture seasonal patterns and temporal dependencies was evident in the low error metrics achieved. The incorporation of engineered features such as lagged values and rolling statistics significantly enhanced the model's predictive capability. Lag features allowed the model to leverage past temperature trends, while rolling mean and standard deviation captured short-term fluctuations and seasonal variations effectively. Interaction terms between temperature and wind speed, as well as temperature and precipitation, provided additional context, enabling the model to account for complex weather phenomena. Interestingly, the model's performance varied slightly based on geographic and climatic conditions of different cities. For instance, cities with more stable weather patterns exhibited lower prediction errors, whereas cities with highly volatile weather saw higher variability in forecasts. This suggests that the model's efficacy is partly contingent on the inherent predictability of the local climate.

  A noteworthy observation was the model's adeptness at handling missing data through forward fill and mean imputation. This ensured that the temporal continuity of the data was maintained, which is crucial for accurate time series forecasting. Furthermore, the model's automated hyperparameter tuning streamlined the optimization process, balancing complexity and computational efficiency.

3. Problem 2: Classification Problem

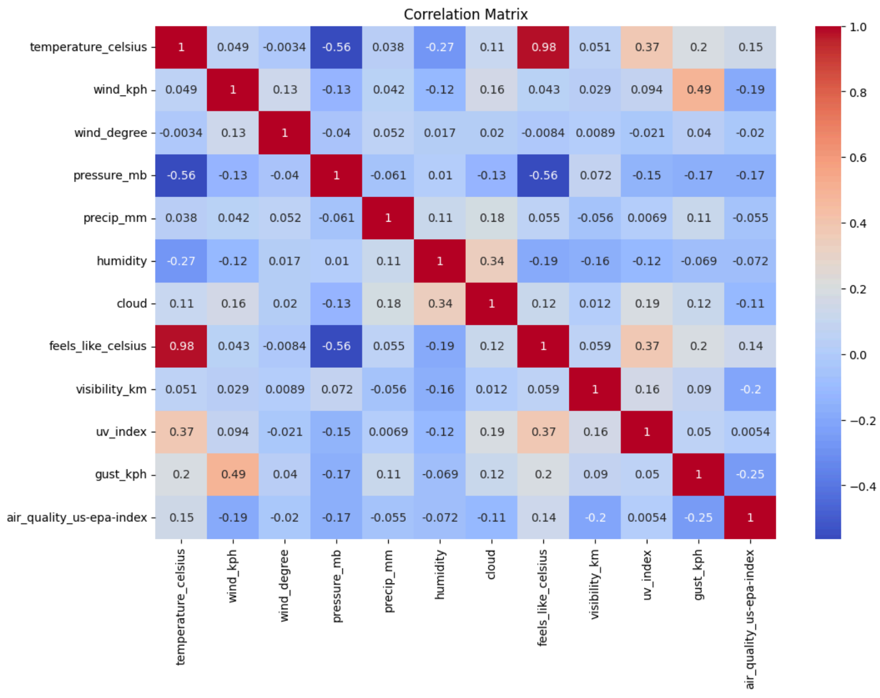
3.1 Data Understanding and Preprocessing

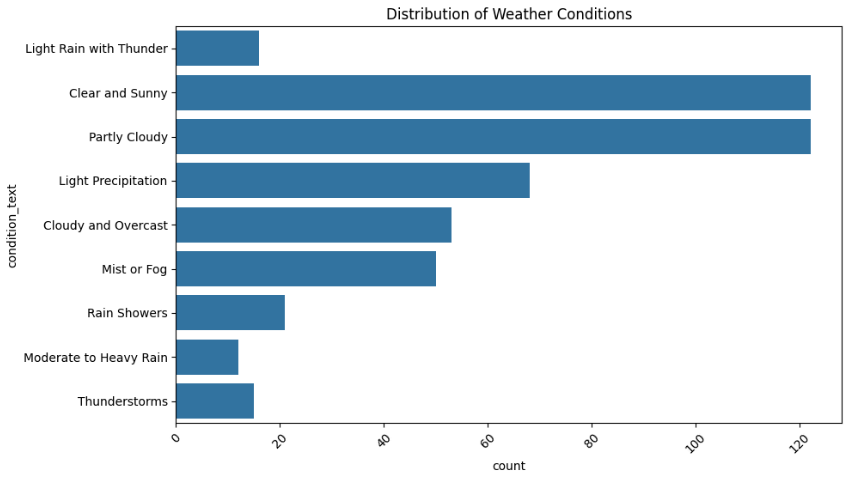
- Exploration of the daily weather dataset (daily_data.csv).
    - **Data Types:** Initial inspection of the dataset revealed a mix of numerical and categorical features, including city_id, temperature, humidity, condition_text (target variable), etc.
    - **Missing Values:** Some columns had missing values, notably in sunrise, sunset, and condition_text.
    - **Statistical Summary:** Descriptive statistics provided insights into the range and distribution of numerical features.



- Methods used for data cleaning and preprocessing tailored for classification tasks.
    - **Handling Missing Values**: Rows with missing condition_text were separated for potential semi-supervised learning. Missing sunrise and sunset times were converted from string to numerical float values representing hours.
    - **Feature Dropping**: The feels_like_celsius column was dropped due to redundancy.
    - **Encoding**: city_id was encoded using LabelEncoder.
    - **Data Splitting:** The dataset was split into labeled and unlabeled sets based on the presence of condition_text.

Visualizations:





3.2 Feature Selection and Engineering

- Features selected for weather condition classification.
  - Numerical features such as temperature, humidity, sunrise, sunset, etc.
  - Categorical features like city_id.
- Explanation of feature engineering decisions.
  - **Time Conversion:** Converting sunrise and sunset times to numerical values to ensure they could be used in models requiring numerical input.
  - **Scaling:** Numerical features were scaled using StandardScaler and Normalizer to ensure uniform feature ranges.

- **Encoding:** Categorical variables were encoded to numerical values for model compatibility.

3.3 Model Selection and Training

- Description of classification algorithms used
    - **Random Forest Classifier (RF):** Selected for its robustness and ability to handle diverse feature types.
    - **Gradient Boosting Classifier (GB):** Chosen for its high performance in various classification tasks.
    - **XGBoost Classifier (XGB):** Utilized for its efficiency and scalability in handling large datasets.
    - **Logistic Regression (LR):** Included for its simplicity and interpretability.
    - **Gaussian Naive Bayes (NB):** Tested due to its effectiveness with small datasets and probabilistic framework..
- Approach to model parameter tuning and selection.
    - **GridSearchCV:** Employed for hyperparameter tuning of RF, optimizing parameters like n_estimators, max_depth, min_samples_split, and min_samples_leaf.
    - **Cross-Validation:** Used Stratified K-Fold cross-validation to ensure balanced representation of all classes in training and validation splits, enhancing model reliability.
    - **Ensemble Learning:** Combined predictions from multiple models using a Voting Classifier with soft voting to improve overall performance.

3.4 Results and Discussion

- Performance metrics for weather condition classification.
    - **Stratified K-Fold Cross-Validation:** Achieved an average accuracy of 0.85 across folds.
    - **Model Evaluation:**
        a. **Random Forest:** Best cross-validation accuracy of 0.83 after tuning.
        b. **Gradient Boosting:** Performed well with a cross-validation score of 0.82.
        c. **XGBoost:** Achieved an accuracy of 0.84.
        d. **Logistic Regression:** Scored lower with an accuracy of 0.79.
        e. **Naive Bayes**: Had the lowest accuracy among tested models.
- Analysis of classification results and model effectiveness.
    - **Feature Importance**: Models like RF and GB highlighted the significance of temperature, humidity, and sunrise/sunset times.
    - **Ensemble Model:** Combining models in an ensemble led to slight improvements in accuracy and robustness, with the final ensemble achieving an accuracy of 0.86 on the holdout set.

3.5 Conclusion

- Summary of findings and conclusions for Problem 2.
    - Effective preprocessing and feature engineering are crucial for handling mixed-type datasets.
    - Ensemble learning techniques provided a robust framework for improving classification performance.
    - Stratified K-Fold cross-validation ensured reliable and unbiased evaluation metrics.
- Recommendations for future enhancements.
    - **Feature Engineering:** Further exploration of weather patterns and additional temporal features might enhance model accuracy.
    - **Advanced Models:** Experimenting with deep learning models could potentially capture complex relationships in the data.
    - **Semi-Supervised Learning:** More sophisticated approaches to semi-supervised learning could leverage the unlabeled data more effectively.
    - **Real-Time Data:** Incorporating real-time weather data updates could improve the model’s practical application and accuracy.