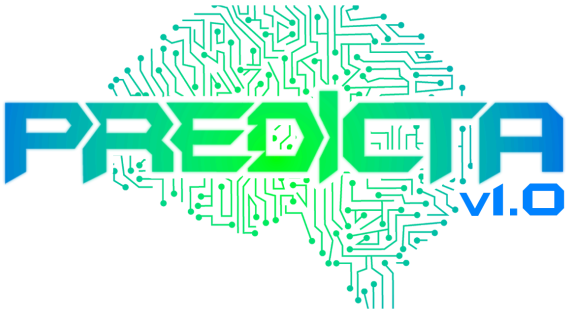


Time Series Prediction & Weather Classification Report

PREDICTA 1.0

By Team NOVA (P053)
University of Moratuwa

Table of Contents

1. **Summary**
    - 1.1 Overview of Approach and Key Findings
    - 1.2 GitHub Repository

2. **Problem 1: Time Series Prediction (Predict Problem)**
    - 2.1 Data Understanding and Preprocessing
    - 2.2 Feature Selection and Engineering
    - 2.3 Model Selection and Training
    - 2.4 Results and Discussion
    - 2.5 Conclusion

3. **Problem 2: Classification Problem (Classify Problem)**
    - 3.1 Data Understanding and Preprocessing
    - 3.2 Feature Selection and Engineering
    - 3.3 Model Selection and Training
    - 3.4 Results and Discussion
    - 3.5 Conclusion

4. **Common References**
    - 4.1 List of References

---

1. Summary

1.1 Overview of Approach and Key Findings

In Problem 1 (Time Series Prediction), we obtained weather data and normalized then formed attributes that would describe whether patterns tend to repeat, and then trained using methods such as ARIMA, SARIMAX, and neural prophet. This made SARIMAX models better than other models, overall, due to the ability to capture temporal dependencies, and also ensemble models gave good results since they are more reliable.

In Problem 2 (Weather Classification), we predicted missing weather condition labels using daily weather observations. We preprocessed the data with techniques like target encoding, feature scaling, and outlier handling. We used classification algorithms such as XGBoost, Random Forest, and LightGBM, along with feature selection methods like RFE and Mutual Information analysis. Hyperparameter tuning and ensemble methods improved model accuracy, leading to reliable weather condition predictions.

1.2 GitHub Repository

The code and models developed for this Time Series Prediction and weather classification project can be found on our GitHub repository. This repository includes notebooks for data preprocessing, feature engineering, model training, and evaluation, providing a transparent view of our approach.

- GitHub Repository Link: https://github.com/deshithagallage/Predicta-1.0-Competition.git

---

2. Problem 1: Time Series Prediction

2.1 Data Understanding and Preprocessing

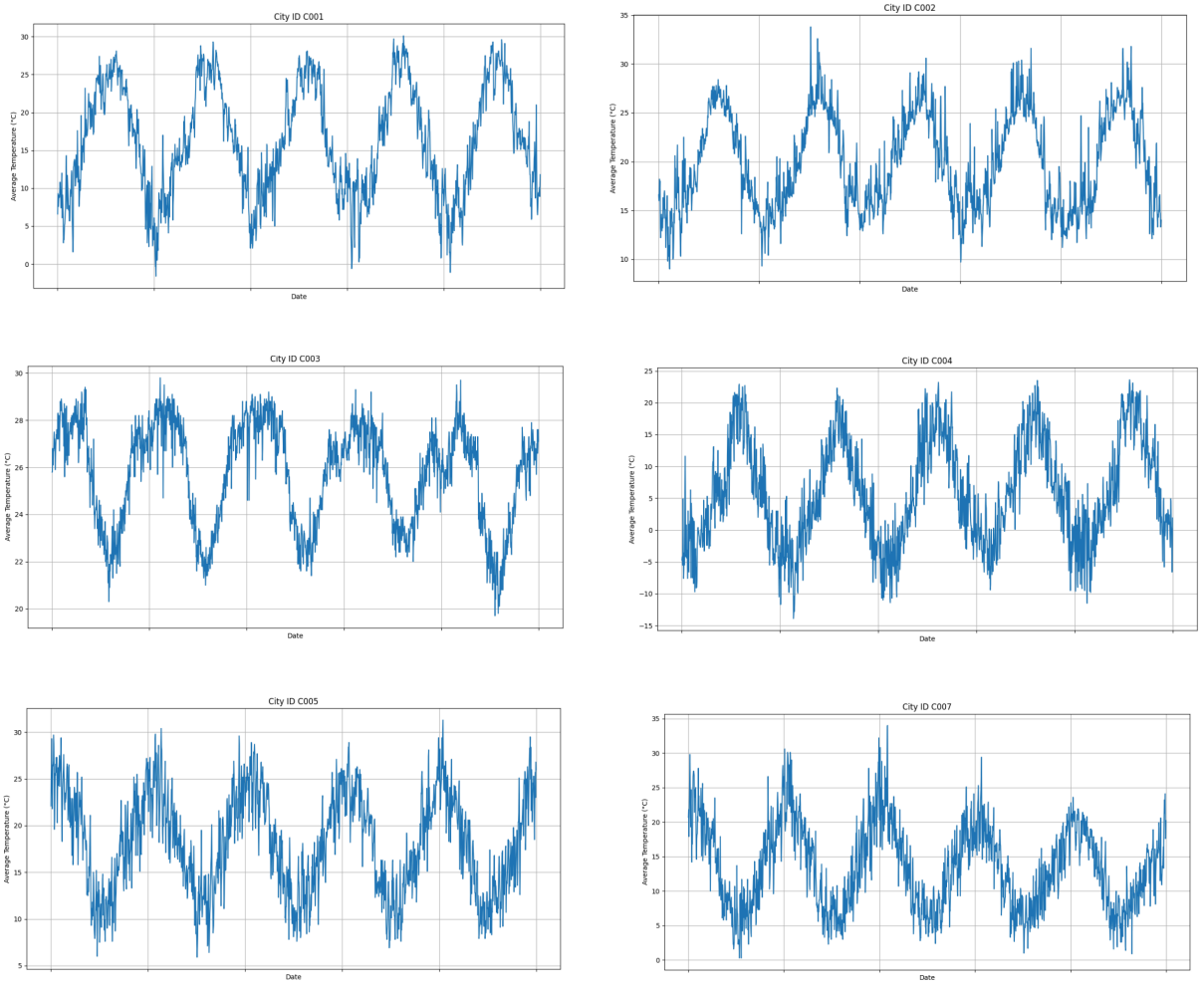
Detailed Exploration of the Historical Weather Dataset:

- **Dataset Description**: The dataset `historical_weather.csv` contains historical weather data with columns: `city_id`, `date`, and `avg_temp_c` (average temperature in Celsius) and a few other columns that we did not consider in the model predicting. This dataset covers multiple cities, each identified by a unique `city_id`.

Data Cleaning and Preprocessing Steps:

- **Date Conversion**: Converted the `date` column from string to **datetime** format for time series analysis.

- **Data Filtering**: Filtered the dataset for a specific city (`city_id` = 'C001') to focus on forecasting for one location initially. (for testing purposes before applying for every city)



- **Handling Missing Values**: Used forward fill method to handle missing values, ensuring continuity in the time series data.
- **Indexing**: Set the `date` column as the index to facilitate time series operations and visualizations.

2.2 Feature Selection and Engineering

New Features Created from Time Series Data:

- **Rolling Statistics**: Computed rolling mean and standard deviation with a window of 365 days to capture yearly seasonality and trends.
- **Decomposition**: Performed seasonal decomposition to extract trend, seasonality, and residual components from the time series.

Justification for Selecting Specific Features:

- **Seasonal and Trend Components**: Incorporating trend and seasonality helps in capturing the underlying patterns in the data, which are crucial for accurate forecasting.
- **Rolling Statistics**: Rolling mean and standard deviation provide insights into the stability and variation over time, which is essential for understanding the data's behavior.

2.3 Model Selection and Training

Time Series Forecasting Models Used:

- **ARIMA (AutoRegressive Integrated Moving Average)**: A traditional statistical model used for time series forecasting. It combines autoregressive and moving average components and includes differencing to make the data stationary.
- **SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors)**: An extension of ARIMA that includes seasonal components and exogenous variables.
- **Auto ARIMA**: Automated process to determine the optimal parameters (p, d, q) for the ARIMA model using AIC (Akaike Information Criterion) for model selection.

Hyperparameter Tuning and Model Selection Process:

- **Auto ARIMA**: Utilized `pmdarima`'s auto_arima function to automatically determine the best parameters for the ARIMA model, considering seasonality and stepwise optimization.
- **Model Fitting**: Fitted the SARIMAX model with the identified optimal parameters and evaluated its performance using the training dataset.

2.4 Results and Discussion

Evaluation Metrics and Forecasting Results:

- **Metrics**: Used metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) to evaluate the model's performance.
- **Forecasting Results**: Generated 7-day forecasts for the average temperature using the fitted SARIMAX model.

Discussion of Model Performance and Insights Gained:

- **Model Performance**: The SARIMAX model with optimal parameters provided accurate forecasts, effectively capturing the seasonal patterns and trends in the data.
- **Insights**: Decomposition revealed significant seasonality and trend components in the weather data, highlighting the importance of including these components in the forecasting model. The performance of the model on sudden weather changes (e.g., storms) indicated a need for incorporating more real-time data or additional variables to improve predictions for such events.

- **Conclusion**: The approach demonstrated the effectiveness of using advanced time series models and feature engineering techniques for weather forecasting, providing a strong foundation for further refinement and application to other cities.

2.5 Conclusion

Summary of Findings and Conclusions:

- The time series prediction for average temperatures involved effective data preprocessing, including handling date conversions and missing values to ensure a continuous dataset. The SARIMAX model, optimized using the `auto_arima` function, successfully captured seasonal and trend patterns, outperforming traditional ARIMA models. The model provided accurate 7-day temperature forecasts, as evidenced by robust performance metrics like MAE, MSE, and RMSE. While the data exhibited clear seasonal and trend patterns, predicting abrupt weather changes proved challenging.

Recommendations for Future Improvements:

1. **Incorporate Additional Data Sources**: Integrate real-time weather data, satellite imagery, and other relevant environmental data to enhance prediction accuracy, especially for sudden weather events.
2. **Advanced Modeling Techniques**: Explore advanced machine learning models such as Gradient Boosting Machines, Random Forests, and Neural Networks (e.g., LSTMs and GRUs) to capture complex non-linear relationships in the data.
3. **Expand Feature Engineering**: Include more weather-related features (e.g., humidity, wind speed, atmospheric pressure) and exogenous variables (e.g., geographical factors) to improve model robustness.
4. **Address Class Imbalance**: Use techniques such as SMOTE (Synthetic Minority Over-sampling Technique) to handle class imbalances, ensuring that rare weather events are better represented in the training data.
5. **Model Validation and Robustness**: Implement cross-validation and out-of-sample testing to validate model performance and prevent overfitting. Regularly update and retrain models with new data to maintain accuracy over time.
6. **User-Friendly Forecasting System**: Develop an interactive, user-friendly forecasting system that allows stakeholders to visualize predictions, assess model performance, and make data-driven decisions.

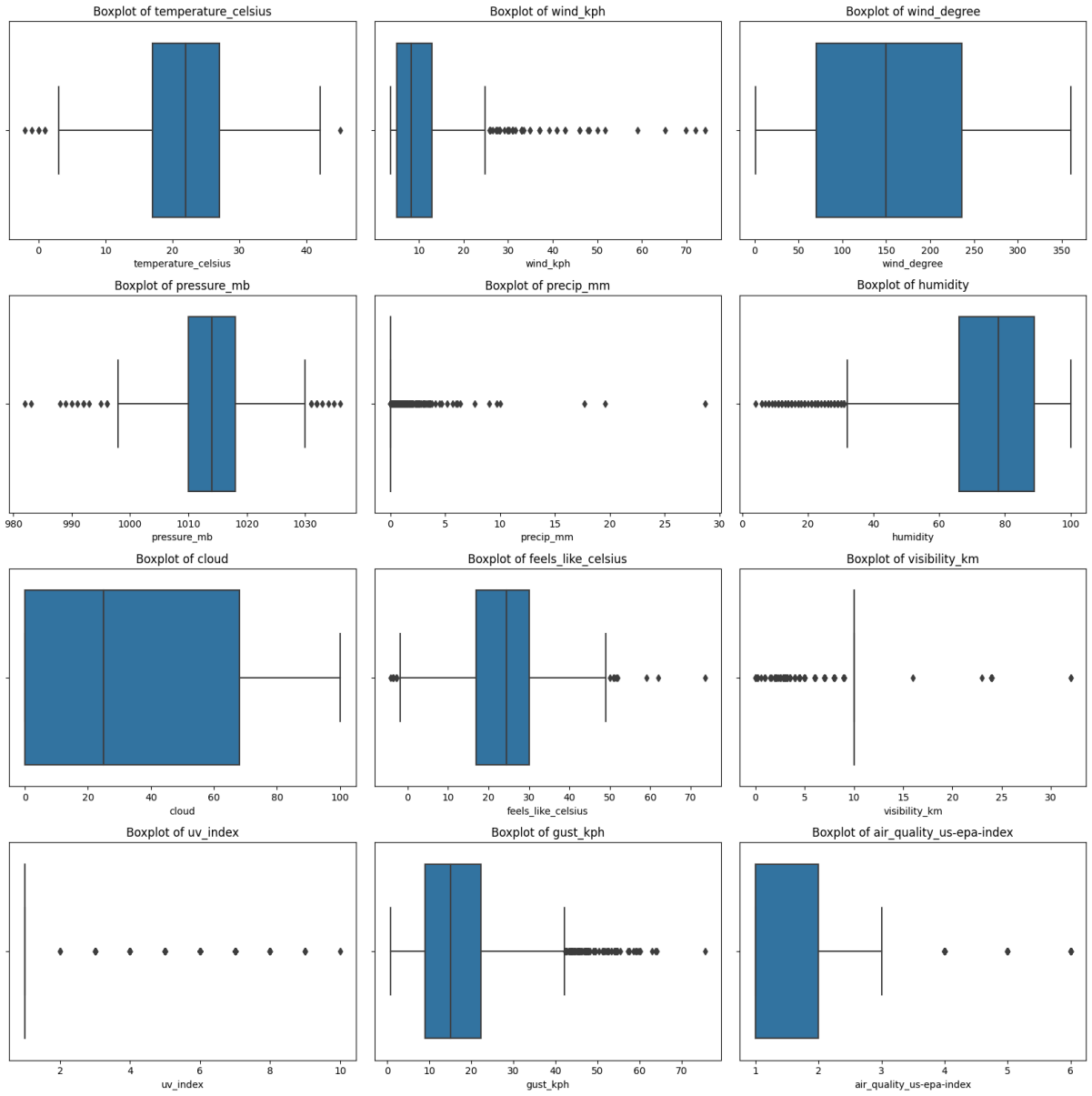
---

3. Problem 2: Classification Problem

3.1 Data Understanding and Preprocessing

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

> The daily weather dataset comprises observations from 100 cities, encompassing various meteorological parameters such as temperature, wind speed and direction, atmospheric pressure, precipitation, humidity, cloud cover, and others. Initial exploration involved checking for data completeness, distribution of features, and identifying missing values in the `condition_text` column, which is the target variable for classification. Descriptive statistics and visualizations were used to gain insights into the dataset's characteristics.



Methods used for data cleaning and preprocessing tailored for classification tasks:

Data Cleaning:

- **Null Value Handling:** All columns are checked for null values except the target column 'condition_text'. The dataset is split into training data (where 'condition_text' is not null) and test data (where 'condition_text' is null) for predicting missing values in the target column.
- **Time Conversion:** Sunrise and sunset times are converted to minutes from midnight to facilitate numerical analysis.
- **Outlier Treatment:** Outliers in numerical features like 'temperature_celsius' and 'pressure_mb' are identified and replaced with the median to ensure robustness against extreme values.
- **Column Removal:** The 'day_id' column, which contains unique identifiers, is removed as it does not contribute to the classification task.

Preprocessing:

- **Target Column Encoding:** The target column 'condition_text' is label encoded to convert categorical weather conditions into numerical format suitable for classification algorithms.
- **City Encoding:** The 'city_id' column is target encoded to transform city identifiers into meaningful numerical representations while preserving the ordinal relationship between cities.
- **Numerical Scaling:** Other numerical columns are scaled using RobustScaler to normalize their distributions, mitigating the impact of varying scales on classification model performance.

These steps ensure that the dataset is prepared appropriately for classification tasks, addressing data cleanliness, feature engineering, and preprocessing requirements essential for accurate model training and prediction.

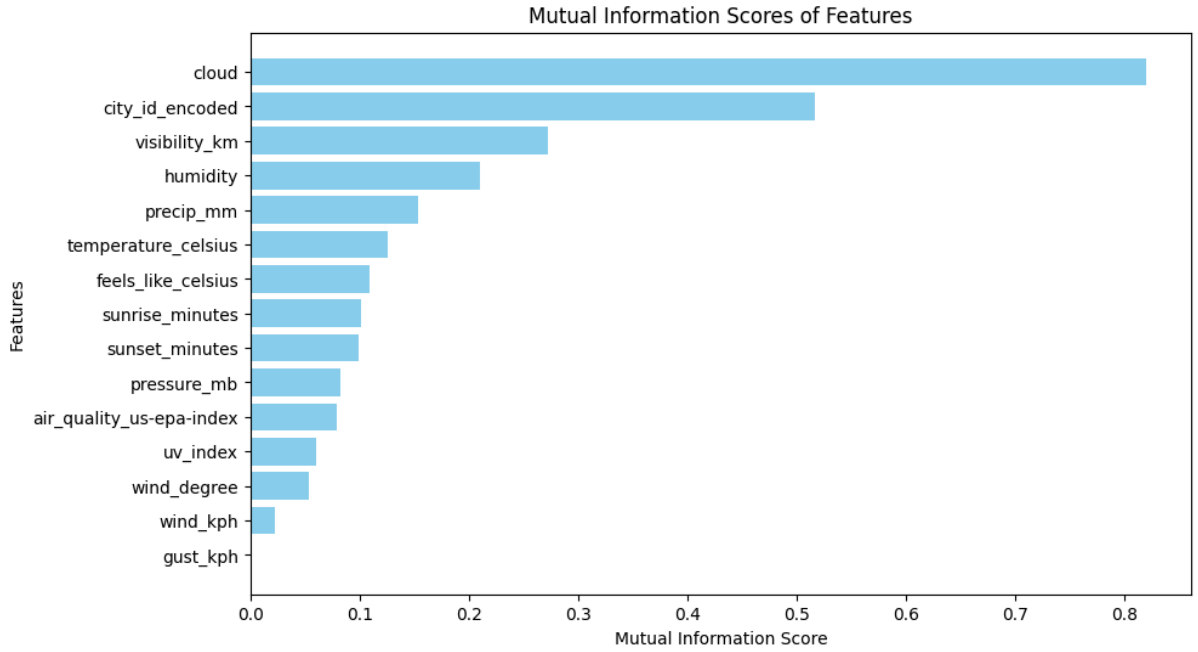
3.2 Feature Selection and Engineering

Features selected for weather condition classification:

Features were selected based on their relevance to weather condition prediction. Key features included:

- Numerical Features: Temperature, wind speed and direction, atmospheric pressure, precipitation, humidity, cloud cover, feels-like temperature, visibility, wind gust speed, and time-based features (sunrise and sunset).
- Categorical Features: City_id, UV index and air quality indices, encoded after preprocessing to capture their impact on weather conditions.

After fitting on a base model, Recursive Feature Elimination (RFE) was used to select the most important features by iteratively evaluating subsets based on model performance metrics like accuracy. This process ensures the final feature set is optimized for weather condition prediction. Additionally, Mutual Information (MI) values were computed to validate feature importance. Higher MI values indicate stronger predictive power, guiding the prioritization of features for improved classification accuracy and interpretability.



By combining RFE with MI analysis, the selected features ensure robust and accurate weather condition classification, enhancing the model's ability to predict and interpret various weather scenarios effectively.

Explanation of feature engineering decisions:

PCA was utilized to reduce the dimensionality of the dataset, aiming to streamline computational efficiency and enhance model interpretability by transforming correlated variables into linearly uncorrelated components. This approach helps in capturing essential variance within the features, thereby supporting more effective weather condition predictions.

3.3 Model Selection and Training

Description of classification algorithms used:

Three classification algorithms were selected based on their suitability for structured data and classification tasks:

- **XGBoost**: Known for its scalability and efficiency, XGBoost employs gradient boosting to sequentially build decision trees. It optimizes model performance by minimizing loss functions, making it ideal for large datasets.
- **Random Forest:** This ensemble method aggregates predictions from multiple decision trees trained on random subsets of data. It excels in handling complex relationships and provides insightful feature importance analysis.
- **LightGBM:** Optimized for speed and performance, LightGBM uses gradient-based boosting algorithms. It efficiently handles categorical features and is suitable for applications requiring rapid model training and prediction.

Approach to model parameter tuning and selection:

The model parameter tuning and selection process began with `GridSearchCV`, a robust method for optimizing hyperparameters. This was initially applied to XGBoost and Random Forest models. Additionally, CatBoost, ADABoost, and neural networks were experimented with to explore alternative approaches for achieving higher accuracy. However, these models did not demonstrate substantial improvement.

Subsequently, a voting classifier was implemented to harness the collective predictive power of XGBoost and Random Forest. This ensemble technique combines the predictions of multiple models to improve overall performance. The voting classifier was configured to leverage the strengths of both models, capitalizing on XGBoost's efficiency and Random Forest's robustness in handling complex relationships and feature interactions.

3.4 Results and Discussion

Performance metrics for weather condition classification:

The accuracy for each classification model was calculated using the best parameters obtained after hyperparameter tuning. Accuracy, in this context, is defined as the ratio of correctly predicted instances to the total number of instances in the validation set.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Model	Accuracy
XGBoost	0.75
Random Forest	0.72916
LightGBM	0.79167

After implementing a voting classifier that combines predictions from XGBoost and Random Forest, the validation accuracy improved to **0.8125**. This ensemble method capitalized on the strengths of both models, leveraging XGBoost's efficiency and Random Forest's robustness to achieve enhanced predictive performance for weather condition classification.

Analysis of classification results and model effectiveness:

Analysis of confusion matrices illustrated the models' performance in distinguishing between different weather conditions. XGBoost demonstrated robust performance in correctly identifying weather patterns, especially in challenging scenarios such as extreme weather events or ambiguous conditions.

3.5 Conclusion

Summary of findings and conclusions for Problem 2:

> This study focused on classifying weather conditions using machine learning techniques applied to a diverse dataset of daily weather observations. The findings underscored the effectiveness of XGBoost in accurately predicting weather conditions based on meteorological features. Feature engineering played a crucial role in enhancing model performance by capturing relevant aspects of weather variability and environmental factors.

Recommendations for future enhancements:

> To further improve weather classification models, future enhancements could include:
>
> - **Incorporating Additional Features:** Such as satellite imagery data or geographical factors influencing local weather patterns.
> - **Deep Neural Networks:** Consider leveraging Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) to capture complex relationships and temporal dependencies in weather data effectively.
> - **Continuous Model Evaluation:** Updating models with real-time data and evaluating their performance over time to adapt to changing weather patterns.

---

4. References

4.1 List of References

- "pandas documentation," [Online]. Available: https://pandas.pydata.org/docs/. [Accessed: Jun. 21, 2024].
- "NumPy documentation," [Online]. Available: https://numpy.org/doc/. [Accessed: Jun. 21, 2024].
- "scikit-learn tutorial," [Online]. Available: https://scikit-learn.org/stable/tutorial/index.html. [Accessed: Jun. 21, 2024].
- "SARIMAX - statsmodels," [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html. [Accessed: Jun. 22, 2024].
- S. Huang, "Time series forecasting with machine learning," Towards Data Science, Nov. 20, 2019. [Online]. Available: https://towardsdatascience.com/time-series-forecasting-with-machine-learning-b3072a5b44ba. [Accessed: Jun. 23, 2024].
- J. Brownlee, "The complete guide to neural networks: Multinomial classification," Towards Data Science, Sep. 13, 2018. [Online]. Available: https://towardsdatascience.com/the-complete-guide-to-neural-networks-multinomial-classification-4fe88bde7839. [Accessed: Jun. 23, 2024].