Weather Analysis Using Machine Learning

**Abdul Aziz Mohammed – 2215231, Ashwitha Reddy Nimmala – 2313782,
FNU Syed Sohaib Ali - 2209809, Neeraj Vasanthapu - 2216992**

Group 7

Abstract

This report presents a comprehensive study on the application of machine learning techniques to analyze and forecast weather data. Weather forecasting is a complex task that involves analyzing various atmospheric parameters to predict future weather conditions. The project focuses on time series forecasting of weather parameters such as temperature and humidity. Various machine learning models including Support Vector Regression (SVR), Random Forest, and XGBoost have been employed and evaluated to address these tasks. The goal is to leverage historical weather data to predict future weather conditions, thereby contributing to more accurate and reliable weather forecasting. This study is crucial given the increasing unpredictability of weather patterns and its significant impact on agriculture, transportation, and daily human activities. The objective is to compare the performance of these models in predicting weather patterns based on historical data. An extensive analysis of the findings is included in the report, along with data exploration, preprocessing, and model construction.

1 Introduction

Weather prediction plays a crucial role in various sectors such as agriculture, transportation, and disaster management. Traditional methods of weather forecasting rely on numerical weather prediction models. In recent years, machine learning models have gained popularity for their ability to capture complex patterns in data. This project aims to harness the power of machine learning in interpreting complex weather data, predicting future weather conditions, and classifying weather summaries.

This study focuses on time series forecasting, a statistical technique used to model and predict future values based on previously observed values. We implement models like ARIMA and SVR, considering their effectiveness in handling linear and non-linear data patterns in time series analysis. This approach is pivotal for forecasting key weather parameters such as temperature and humidity, which are critical for a myriad of applications, from agricultural planning to urban management.

Utilizing a dataset that includes various weather attributes, we apply and compare models like Random Forest and XGBoost. These models are chosen for their ability to handle the high-dimensional and complex nature of weather data.

2 Background

2.1 Dataset Description:

The dataset can be valuable for various applications such as historical weather pattern analysis, climate studies, predicting weather conditions, and more. The dataset contains weather observations with timestamps. It provides hourly data, which gives a detailed temporal resolution. Key weather metrics included in the dataset are Temperature (C), Apparent Temperature (C), Humidity, Wind Speed (km/h), Pressure (millibars). It also includes categorical descriptions like summary and PrecipType, providing qualitative information about the weather (e.g., partly cloudy, rain, sunny). The date and time in the Formatted Date column indicate the range of the data, which starts from at least April 2006 to that in 2016This provides a comprehensive 11-year span, useful for long-term weather pattern analysis. It contains 96453 instances and a total of 12 Features. The feature Precip type contains two categories rain and snow.

2.2 Model Description:

i. Long Short-Term Memory:

LSTM is a type of recurrent neural network (RNN) architecture, specifically designed to handle sequence prediction problems. It is excellent at capturing long-term dependencies and relationships in time series data, which is crucial for accurate weather forecasting. LSTMs have feedback connections that make them suitable for processing entire sequences of data, addressing the vanishing gradient problem common in traditional RNNs.

ii. Vector Autoregression:

VAR is a statistical model used to capture the linear interdependencies among multiple time series. It assumes that each variable is a linear function of past values of itself and past values of other variables in the system.

iii. Support Vector regression:

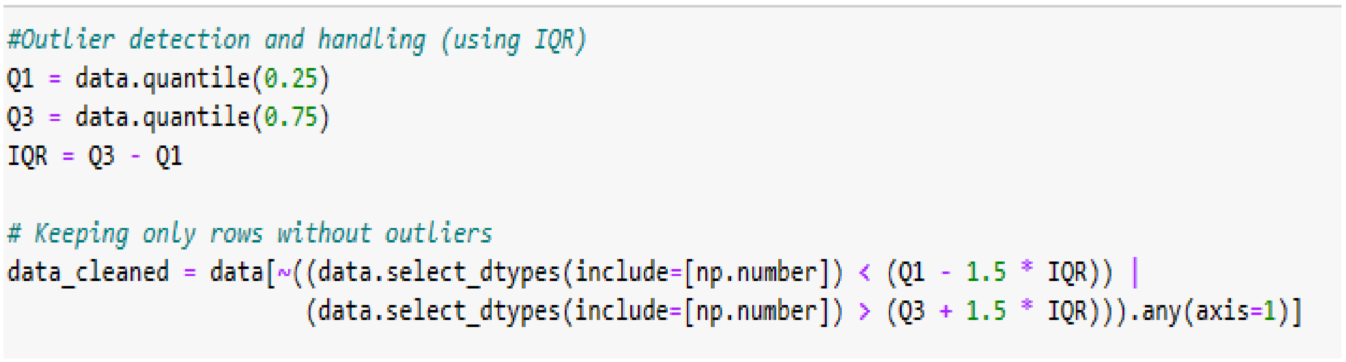
It is an adaptation of Support Vector Machines (SVM) for regression problems. It is capable of handling both linear and non-linear relationships. It is robust to outliers and can model complex data patterns. It works by fitting the best line within a threshold value; points outside the threshold margin are considered outliers.

iv. Extreme Gradient Boosting:

It is an efficient and scalable implementation of gradient boosting framework. It known for its speed and performance, XGBoost is effective in handling a wide range of data types, distributions, and relationships. It offers several regularization options, which help in reducing overfitting. Highly customizable with a wide range of hyperparameters.

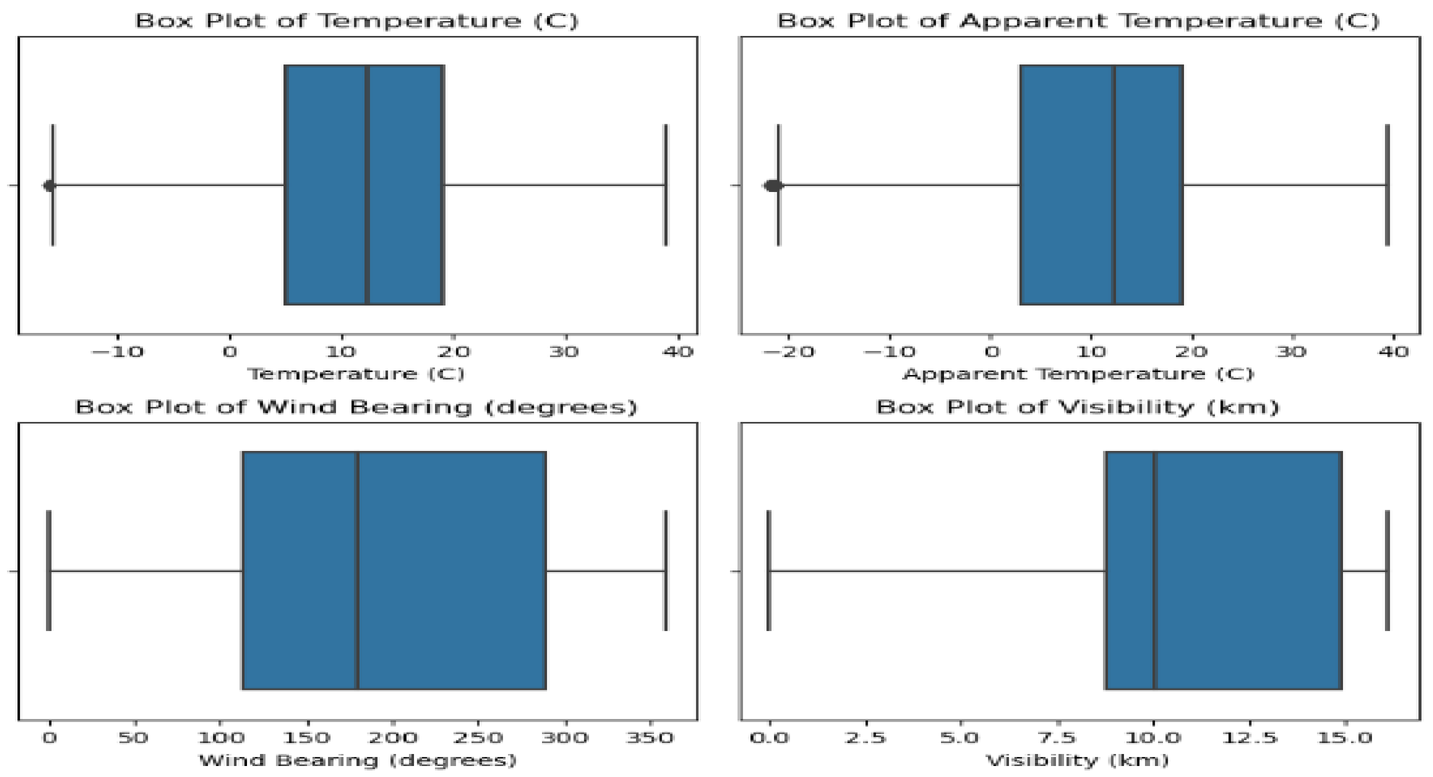
3 Exploratory Data Analysis

A critical component of our EDA was the identification and handling of outliers. For this, box plots were employed, effectively showcasing the median, quartiles, and outliers. Using the Interquartile Range (IQR) method, we filtered out data points lying beyond 1.5 times the IQR from the quartiles, thus refining our dataset.

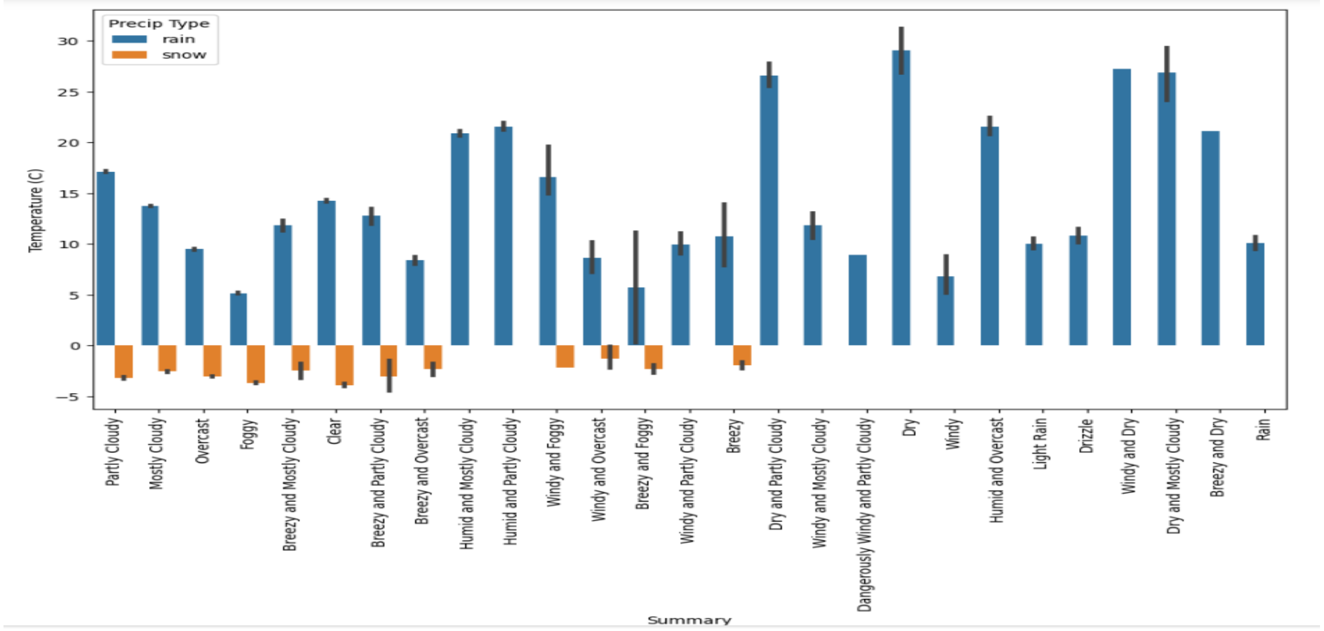


```
#Outlier detection and handling (using IQR)
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1

# Keeping only rows without outliers
data_cleaned = data[~((data.select_dtypes(include=[np.number]) < (Q1 - 1.5 * IQR)) |
                      (data.select_dtypes(include=[np.number]) > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Creating box plots after removing outliers,

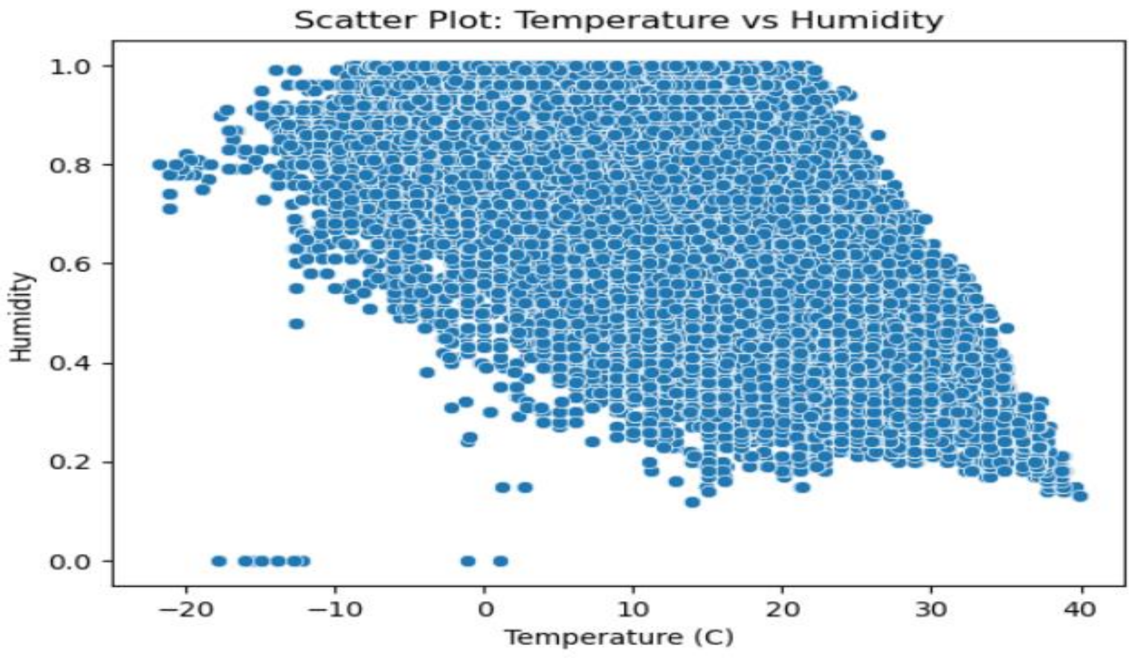


Relationship between ‘Temperature (c)’ and ‘Precip Type’



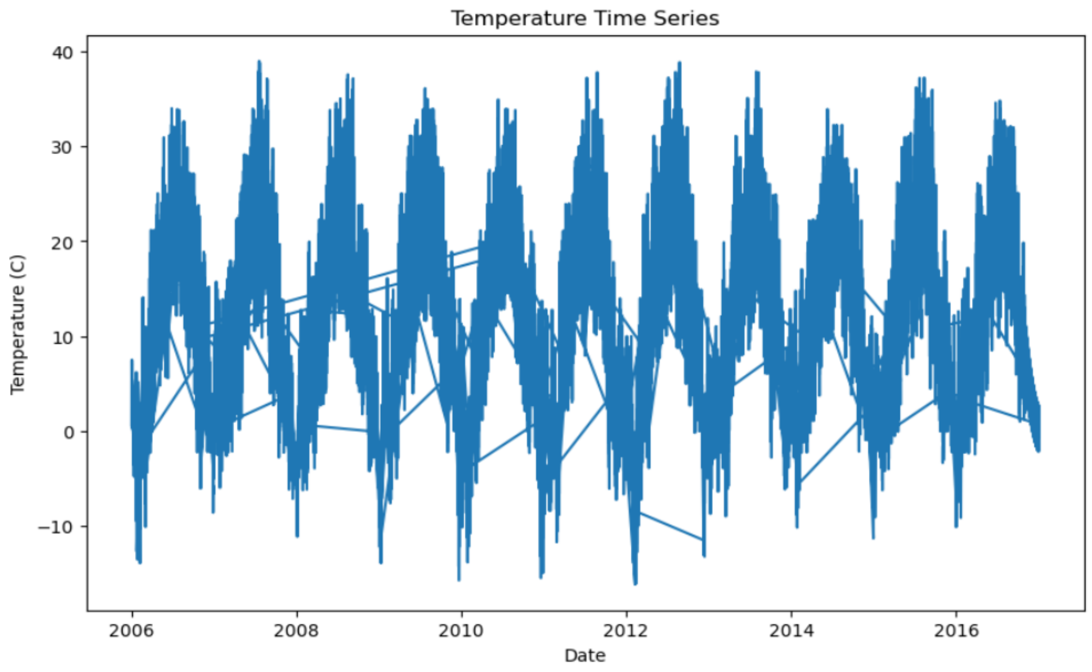
The temperatures for snow conditions are generally lower than those for rain conditions, as expected. Some weather conditions, like 'Dangerous Wind and Rain', have a relatively high average temperature. The summaries associated with colder temperatures, especially with snow, are on the left side of the graph.

Relationship between 'Temperature (c)' and 'Humidity'



There is a higher density of data points at temperatures between approximately 0°C and 20°C, and humidity levels above 0.4. This indicates moderate temperatures and relatively high humidity.
There's a visible trend where humidity decreases as the temperature increases, especially noticeably above 20°C, which is consistent with the physical principle that warm air can hold more moisture before it condenses.

Temperature time series

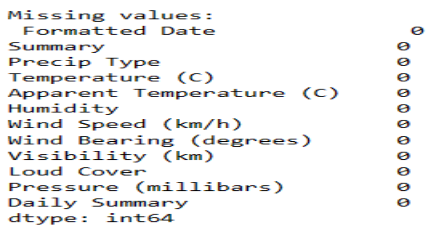


The graph shows a repeating pattern of peaks and troughs that correspond to the warmer and colder months of each year, respectively. The highest temperatures (peaks) appear to reach above 30°C, while the lowest (troughs) dip below -10°C.

4 Data Pre-processing

4.1 Handling Missing Values:

This process is a common preliminary step in data cleaning and preparation, ensuring that the dataset is complete and that any analyses performed on it are not skewed by missing data. We made use of .fillna() function to replace all the missing data with its mode. Dataset after removing all the missing values.



```
Missing values:
 Formatted Date               0
Summary                      0
Precip Type                  0
Temperature (C)              0
Apparent Temperature (C)     0
Humidity                     0
Wind Speed (km/h)            0
Wind Bearing (degrees)       0
Visibility (km)              0
Loud Cover                   0
Pressure (millibars)         0
Daily Summary                0
dtype: int64
```

4.2 Dropping Unwanted Columns

Using the .drop() we removed the column named ‘Loud Cover’ as it has zero correlation with any of the data features.

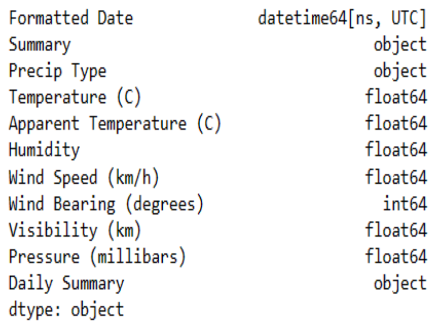
4.3 Removing Duplicates

Initially, we determined the number of duplicates present in the data using the function .duplicated().sum(). We observed that there was a total of 24 duplicates values in the dataset. We removed these values using the function .drop_duplicates().

4.4 Data Formatting

We observed that the column ‘Formatted Date’ data type was object, to make the analysis over time we changed its data format to datetime64 using the function pd.to_datetime(data['Formatted Date'], utc=True).

The data is as follows the changes.



```
Formatted Date              datetime64[ns, UTC]
Summary                                  object
Precip Type                              object
Temperature (C)                         float64
Apparent Temperature (C)                float64
Humidity                                float64
Wind Speed (km/h)                       float64
Wind Bearing (degrees)                    int64
Visibility (km)                         float64
Pressure (millibars)                    float64
Daily Summary                            object
dtype: object
```

5 Model Building

Model building involves using statistical and machine learning techniques to understand, interpret, and forecast weather conditions. Given the complexity and variability inherent in weather data, this task presents unique challenges and opportunities for applying advanced analytical methods.

The choice of model depends on the specific characteristics of the data and the analysis goals. For instance, LSTM is ideal for capturing long-term dependencies in time series, while XGBoost is suited for scenarios where relationships between variables are complex and non-linear. Models are rigorously evaluated using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and accuracy. These metrics help in assessing the performance and reliability of the models.

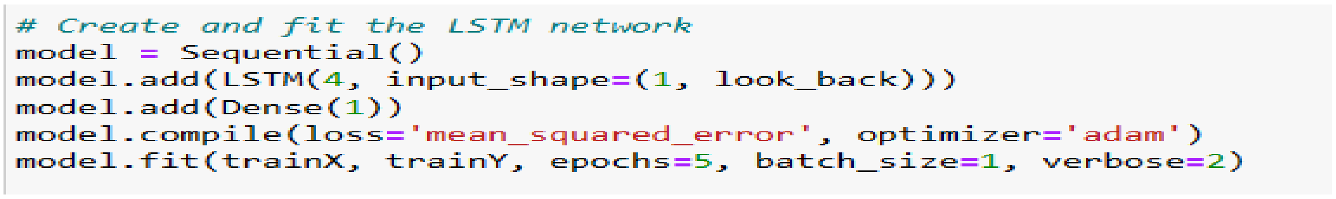
i. Long Short-Term Memory:

Here, we initially transformed the data to create time-lagged features, as LSTM requires input in a sequence format. The target variable (e.g., temperature) was shifted to create corresponding input features for the model.

Then we normalize and transform the feature ‘temperature (c) ’ using the function MinMaxScaler(). And set the training data to 80%.

The model was trained on historical weather data, allowing the LSTM to learn patterns in temperature, humidity, and other factors over time.

We used a framework Keras to implement the LSTM model. The model was trained iteratively over several epochs with a specified batch size as follows,

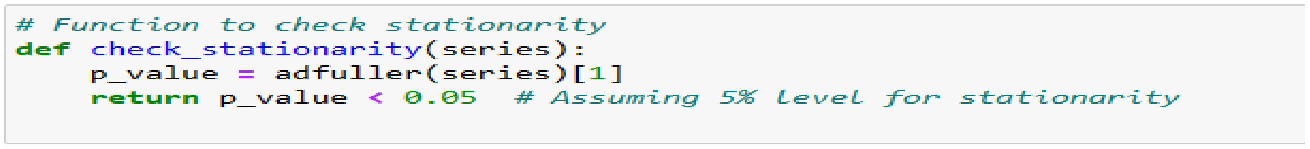


```
# Create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=5, batch_size=1, verbose=2)
```

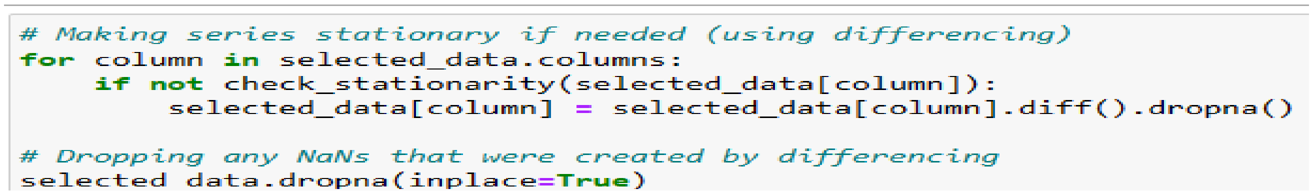
ii. Vector Autoregression:

The VAR model uses multiple time-dependent variables like temperature and humidity.

Before model fitting, the stationarity of each time series in the dataset was checked. Non-stationary series were different to achieve stationarity.

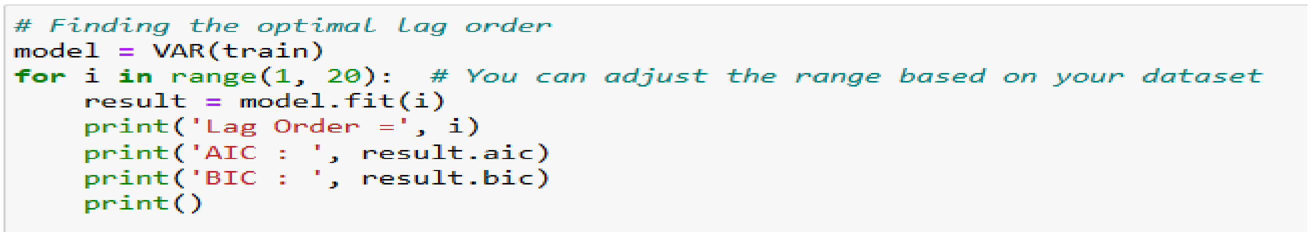


```
# Function to check stationarity
def check_stationarity(series):
    p_value = adfuller(series)[1]
    return p_value < 0.05  # Assuming 5% level for stationarity
```



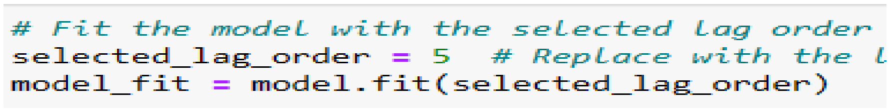
```
# Making series stationary if needed (using differencing)
for column in selected_data.columns:
    if not check_stationarity(selected_data[column]):
        selected_data[column] = selected_data[column].diff().dropna()

# Dropping any NaNs that were created by differencing
selected_data.dropna(inplace=True)
```

The VAR model was fitted to the stationary series, with the lag order determined based on information criteria like AIC.



```
# Finding the optimal lag order
model = VAR(train)
for i in range(1, 20):  # You can adjust the range based on your dataset
    result = model.fit(i)
    print('Lag Order =', i)
    print('AIC : ', result.aic)
    print('BIC : ', result.bic)
    print()
```

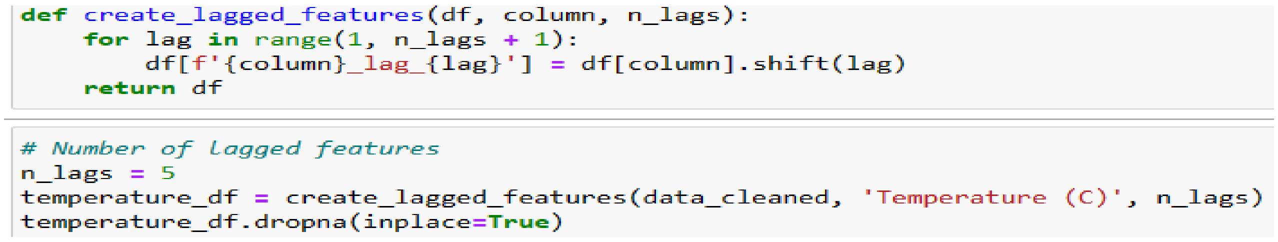
Then we fit the model with making the lag_order of 5 and then make the predictions



```
# Fit the model with the selected lag order
selected_lag_order = 5  # Replace with the l
model_fit = model.fit(selected_lag_order)
```

iii. Support Vector regression:

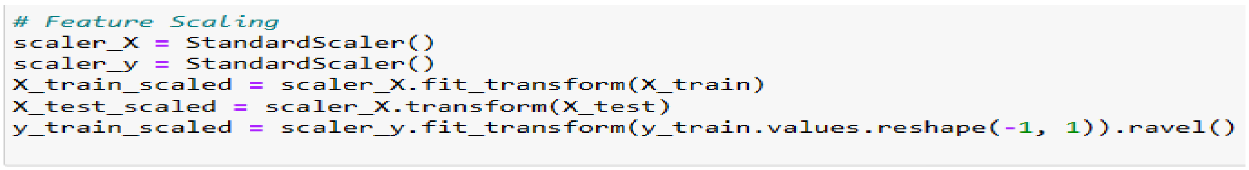
The dataset was preprocessed to create lagged features like LSTM. Additionally, feature scaling was performed as SVR is sensitive to the scale of input data.



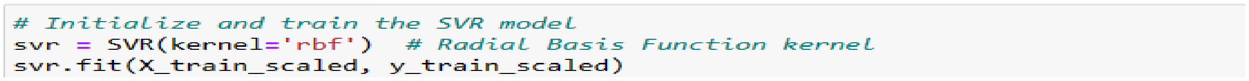
```
def create_lagged_features(df, column, n_lags):
    for lag in range(1, n_lags + 1):
        df[f'{column}_lag_{lag}'] = df[column].shift(lag)
    return df
```

```
# Number of lagged features
n_lags = 5
temperature_df = create_lagged_features(data_cleaned, 'Temperature (C)', n_lags)
temperature_df.dropna(inplace=True)
```

The SVR model was trained on the scaled features, and predictions were made on the test set. The model aims to find the best fit line within a certain threshold.



```
# Feature Scaling
scaler_X = StandardScaler()
scaler_y = StandardScaler()
X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).ravel()
```



```
# Initialize and train the SVR model
svr = SVR(kernel='rbf')  # Radial Basis Function kernel
svr.fit(X_train_scaled, y_train_scaled)
```

iv. Extreme Gradient Boosting:

Features were prepared similarly to the SVR model, including lagged variables and scaling.

```
# Assuming 'Temperature (C)' is the target variable
def create_lagged_features(df, column, n_lags):
    for lag in range(1, n_lags + 1):
        df[f'{column}_lag_{lag}'] = df[column].shift(lag)
    return df
```

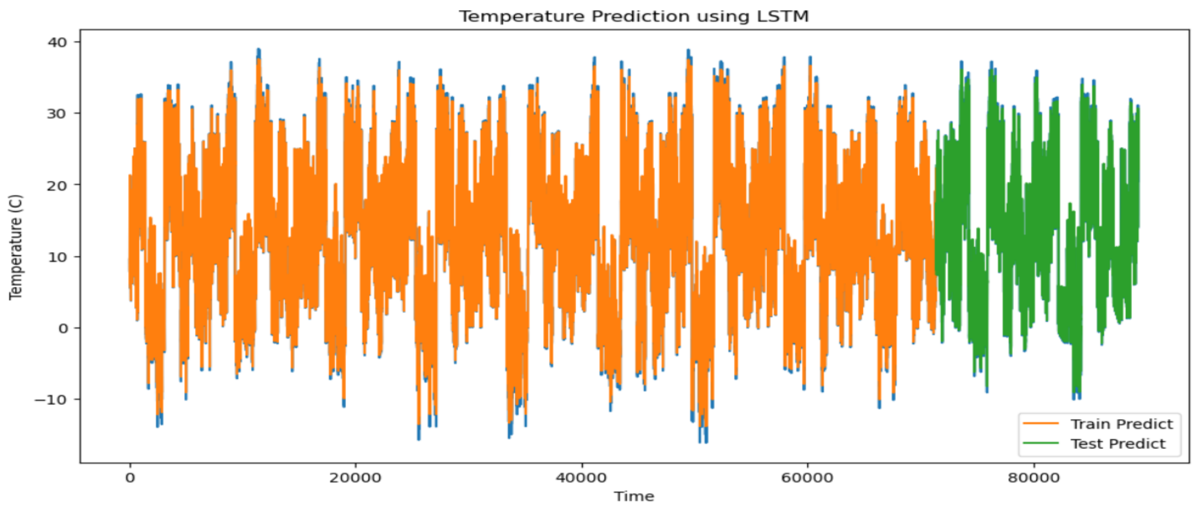
```
# Number of lagged features
n_lags = 5  # Adjust this based on your dataset
temperature_df = create_lagged_features(data_cleaned, 'Temperature (C)', n_lags)
temperature_df.dropna(inplace=True)
```

The model was trained on the training set, using gradient boosting techniques to iteratively improve predictions over multiple rounds.

```
# Initialize and train the XGBoost model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1)
xgb_model.fit(X_train, y_train)
```

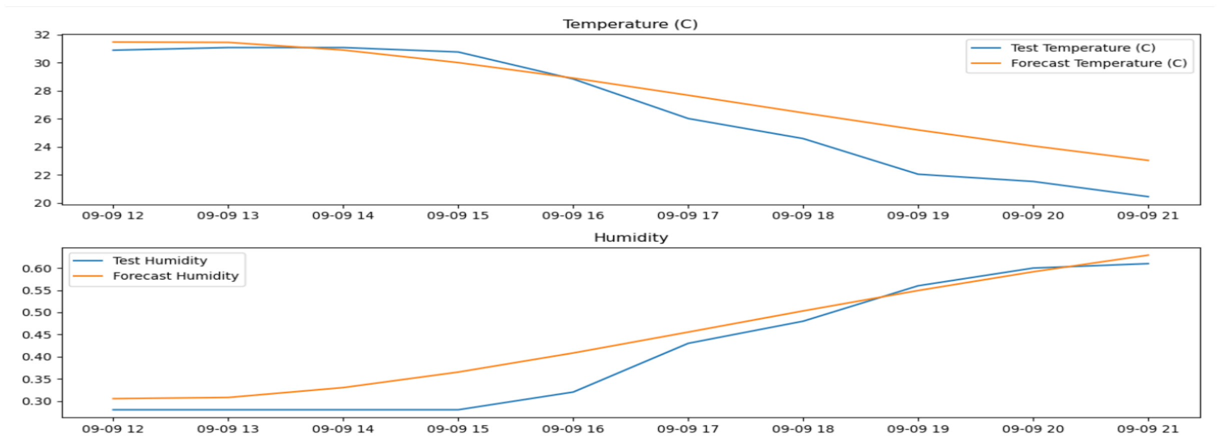
6 Results

i. LSTM



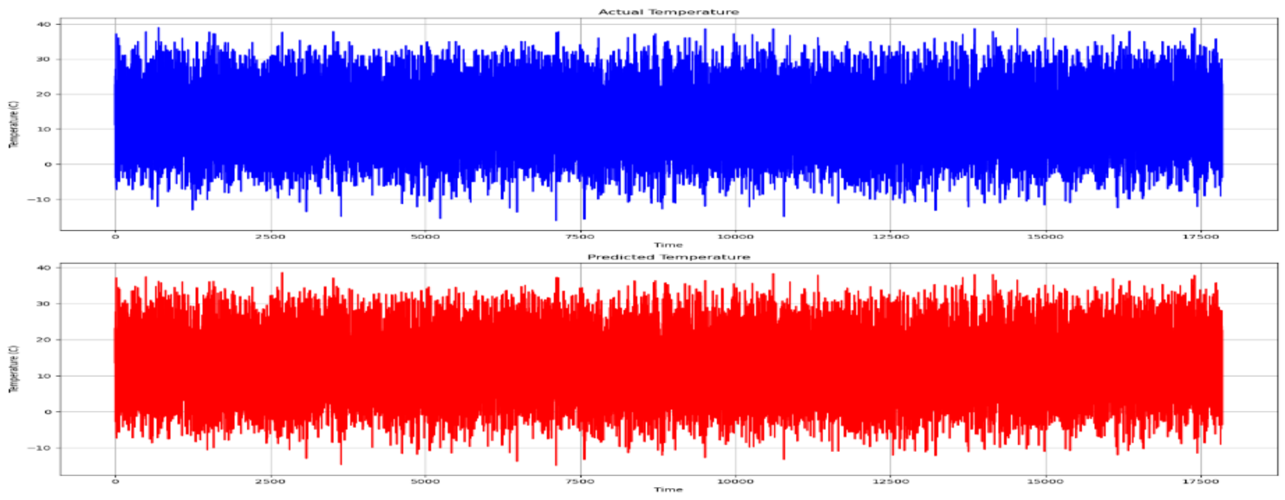
The graph presents a visualization of temperature predictions made using an LSTM. We observe model's predictions follow the overall trend of the temperature data quite closely during both the training and testing phases. It has an MSE of 2.00, MAE of 0.97 and an r2 score of 0.97.

ii. Vector AutoRegressor



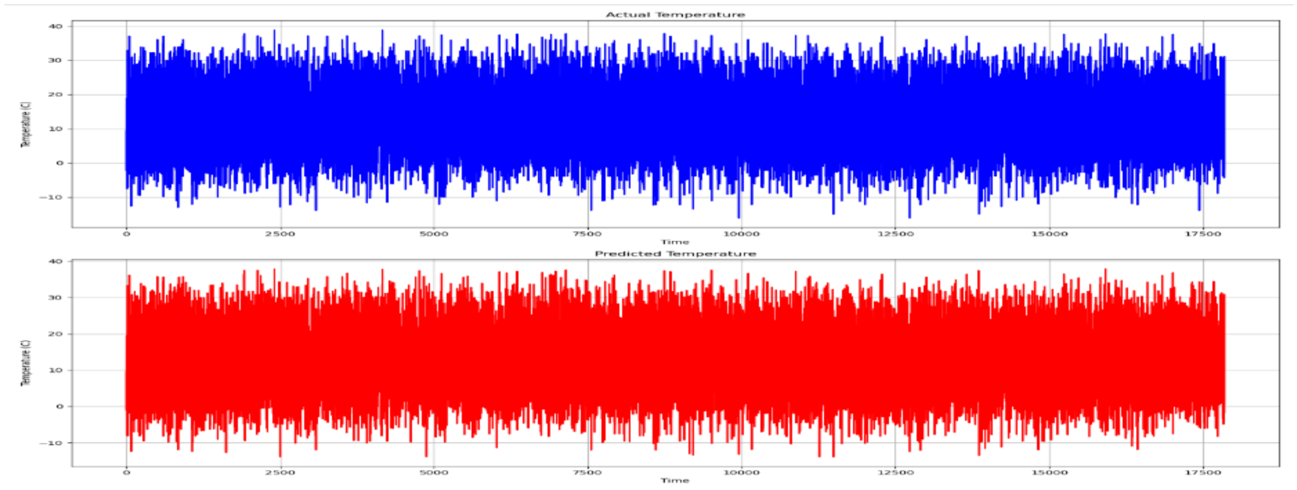
The trends are inverse for temperature and humidity, which is a common real-world observation as temperature often inversely correlates with humidity. The relatively proximity of the forecast and test lines suggests that the forecasting model is following the right trend. It has an MSE of 1.51, MAE of 0.70 and an r2 score of 0.82.

iii. Support Vector Regressor.



The above graph represents the actual and predicted temperature. The consistency in the range of temperatures between actual and predicted suggests that the prediction model is capturing the overall variability and range of temperatures appropriately. There are no explicit anomalies or outliers that can be seen from the graphs, indicating that the prediction model did not produce any extreme errors within the observed time frame. It has an MSE of 1.94, MAE of 0.88 and an r2 score of 0.98.

iv. Extreme Gradient Boosting.



The overall distribution of the actual and predicted temperatures appears broadly similar, with both occupying predominantly the upper temperature range and exhibiting a high frequency of fluctuations. The dense nature of the bars suggests that the data points are very close together in time, which could mean that the temperature was recorded or predicted at frequent intervals, potentially as a high-resolution time series. It has an MSE of 2.01, MAE of 0.88 and an r2 score of 0.97.

Model	MSE	MAE	R2 Value
LSTM	2	0.97	0.97
Vector Auto Regressor	1.51	0.70	0.82
Support Vector Regressor	1.94	0.88	0.98
Gradient Boosting	2.01	0.88	0.97

7 Discussion

In this analysis, we have explored the performance of a temperature prediction model. The dataset visualized in the graphs represents actual and predicted temperature values across a sequential time frame, extending from 0 to approximately 17,500 units on the x-axis, with the temperature in Celsius on the y-axis ranging from below -10°C to above 40°C.

The first graph, depicted in blue, illustrates the actual temperature data recorded over the time period. The data points are densely packed and display frequent fluctuations, indicating either daily temperature cycles or data captured at high-frequency intervals. The distribution is mostly concentrated in the upper half of the temperature range, which may suggest a warmer climate or season during the measurement period.

The second graph, shown in red, presents the predicted temperatures over the same time period. The prediction data closely mirrors the actual temperature data in terms of distribution, density, and range. This visual similarity suggests that the predictive model has successfully learned the temperature patterns from the training data.

Despite the apparent qualitative agreement between the actual and predicted temperatures, the analysis does not reveal the precision of the predictions at individual time points. The model's performance, in terms of its accuracy and reliability, would benefit from a more granular examination, potentially by overlaying the actual and predicted temperatures on the same graph for a direct comparison. Additionally, the incorporation of error metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) would provide a quantitative measure of the model's predictive capabilities.

Overall, the graphs suggest that the model has a solid understanding of the temperature dynamics and is capable of closely replicating the actual temperature trends. However, further validation and testing are required to establish the model's efficacy for practical applications, such as in environmental monitoring or weather forecasting systems.

8 Conclusion

In conclusion, the analysis of the temperature data has provided insightful observations into the performance of the temperature prediction model. The comparative visual examination of the actual and predicted temperature over a considerable time span demonstrates that the model can capture the overall temperature trends and fluctuations with a high degree of similarity between the actual and forecasted data.

Although the data suggests that the predictive model is robust in terms of capturing the general pattern of temperature changes, the lack of detailed comparison and quantitative metrics in this analysis precludes a definitive assessment of the model's precision and accuracy. Hence, it would be premature to fully endorse the model's reliability for practical forecasting applications based on this analysis alone.

For future work, a deeper statistical evaluation incorporating error metrics is recommended to quantify the model's performance. Additionally, an examination of the model's responsiveness to different climatic conditions, as well as its adaptability to various temporal resolutions of data, would be beneficial in understanding its potential utility in real-world scenarios.

Ultimately, the promising results observed invite further investigation and refinement of the model, which could lead to significant advancements in the domain of temperature prediction and its associated applications.

Among all the four models, the support vector regressor performs the best for the dataset with a r2 score of 0.98, thereby providing an indication of the goodness of fit of the model.

This conclusion summarizes the findings, acknowledges the limitations of the analysis, suggests directions for future research, and provides a final statement on the potential implications of the work.