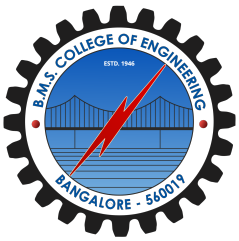

BMS COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Bull Temple Road, Basavanagudi, Bangalore – 560019

A project report on

Weather Data Analysis and Prediction using Apache Spark

Submitted in partial fulfillment of the requirements for the award of degree

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING
(DATA SCIENCE)

By

Pramod MB (1BM23CD045)
Divan Singh (1BM23CD017)
Kashish Gaddigoudar (1BM24CD401)
Lakshya Khandelwal (1BM23CD031)

Under the guidance of
Prof. Lavanya Naik

Department of Computer Science and Engineering
(Data Science)

2025-26

BMS COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Bull Temple Road, Basavanagudi, Bangalore – 560019

Department of Computer Science and Engineering (DATA SCIENCE)

CERTIFICATE

This is to certify that the project entitled **"Weather Data Analysis and Prediction using Apache Spark"** is a bona-fide work carried out by **Pramod MB (1BM23CD045), Divan Singh (1BM23CD017), Kashish Gaddigoudar (1BM24CD401) and Lakshya Khandelwal (1BM23CD031)** in partial fulfillment for the award of degree of **Bachelor of Engineering** in **Computer Science and Engineering (Data Science)** from **Visvesvaraya Technological University, Belgaum** during the year 2025-26. It is certified that all corrections/suggestions indicated for Internal Assessments have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering Degree.

Prof. Lavanya Naik
(Associate Professor)

Signature of the Guide

Abstract

The accurate prediction of weather parameters, such as temperature, is a critical task with applications ranging from agriculture to urban planning. This project leverages the power of Big Data Analytics using Apache Spark (PySpark) to process and analyze a large dataset of daily weather observations. The primary objective is to build a robust machine learning model to predict the average daily temperature based on other meteorological features.

The methodology involves a comprehensive data pipeline: data ingestion, cleaning (handling duplicates and missing values), transformation (feature extraction and joining with supplementary datasets), and exploratory data analysis (EDA). We utilized Spark MLlib to construct a Linear Regression model. To ensure optimal performance, we implemented Cross-Validation with hyperparameter tuning. The results demonstrate the effectiveness of PySpark in handling large-scale data processing and machine learning tasks, yielding a model with high predictive accuracy.

Contents

1 Introduction

1.1 Background

Weather forecasting has traditionally been a complex scientific challenge involving massive amounts of atmospheric data. With the advent of Big Data technologies, we can now process historical weather data more efficiently to uncover patterns and build predictive models. Apache Spark, a unified analytics engine for large-scale data processing, provides the necessary tools to handle such datasets through its PySpark interface.

1.2 Project Objectives

The main goals of this project are:

1. To demonstrate the ability to ingest and process large datasets using PySpark.
2. To perform essential data cleaning and transformation tasks, including handling missing values and joining multiple datasets.
3. To conduct Exploratory Data Analysis (EDA) to understand the underlying distribution and correlations of weather variables.
4. To build, tune, and evaluate a Machine Learning model (Linear Regression) using Spark MLlib to predict the average temperature (`avg_temp_c`).

1.3 Scope

The scope of this analysis is limited to the provided `daily_weather.parquet` dataset. The analysis focuses on predicting the average temperature using features such as minimum temperature, maximum temperature, precipitation, wind speed, and air pressure.

2 Dataset Description

2.1 Data Source

The dataset used in this project is `daily_weather.parquet`. Parquet is a columnar storage file format available to any project in the Hadoop ecosystem, designed for efficiency.

2.2 Schema and Features

The dataset consists of daily weather observations. The key features identified in the schema are:

- **station_id**: Unique identifier for the weather station.
- **date**: The date of the observation.
- **avg_temp_c**: Average temperature in Celsius (Target Variable).
- **min_temp_c**: Minimum temperature in Celsius.
- **max_temp_c**: Maximum temperature in Celsius.
- **precipitation_mm**: Precipitation amount in millimeters.
- **avg_wind_speed_kmh**: Average wind speed in km/h.
- **avg_sea_level_pres_hpa**: Average sea-level pressure in hPa.
- **season**: Categorical variable indicating the season.

3 Theoretical Framework

3.1 Apache Spark and PySpark

Apache Spark is an open-source distributed general-purpose cluster-computing framework. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. PySpark is the Python API for Spark, allowing us to use Python's simple syntax while leveraging the power of Spark's distributed processing.

3.2 Linear Regression

Linear Regression is a supervised learning algorithm used for computing the linear relationship between a dependent variable (target) and one or more independent variables (features). The goal is to find the best-fitting straight line through the data points. The equation is given by:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \epsilon$$

Where Y is the target variable, X are the features, β are the coefficients, and ϵ is the error term.

3.3 Cross-Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. In this project, we use k-fold cross-validation to ensure our model does not overfit the training data.

4 Data Preprocessing

4.1 Data Ingestion

The data was loaded into a Spark DataFrame. We verified the load by printing the schema and the total count of rows.

```
# Loading the dataset
df = spark.read.parquet('daily_weather.parquet')
print(f"Total Rows: {df.count()}")
```

4.2 Data Cleaning

Real-world data is often messy. We performed the following cleaning steps:

4.2.1 Removing Duplicates

Duplicate records can skew analysis and model training. We used the `dropDuplicates()` method.

```
initial_count = df.count()
df = df.dropDuplicates()
final_count = df.count()
print(f"Removed {initial_count - final_count} duplicate rows.")
```

4.2.2 Handling Missing Values

We analyzed the dataset for null values. Rows containing nulls in the target variable or key feature columns were dropped to ensure the integrity of the model.

```
required_columns = ['avg_temp_c', 'min_temp_c', 'max_temp_c', ...]
df_clean = df.na.drop(subset=required_columns)
```

4.3 Data Transformation

4.3.1 Feature Extraction

To enable temporal analysis, we extracted the `Year` and `Month` from the `date` column using PySpark's built-in date functions.

4.3.2 Joining Datasets

We demonstrated a data integration technique by creating a supplementary DataFrame containing descriptions for each season (e.g., "Spring", "Summer"). We then performed a **Left Join** to merge this description into our main weather dataset.

```python
# Join operation
df = df.join(season_df, on="season", how="left")
```

5 Exploratory Data Analysis (EDA)

In this section, we visualize the data to gain insights. Note: For visualization purposes, a representative sample of the data was converted to a Pandas DataFrame.

5.1 Distribution of Average Temperature

The histogram below shows the frequency distribution of the average temperature. We observe that the temperature follows a specific distribution pattern relevant to the region's climate.

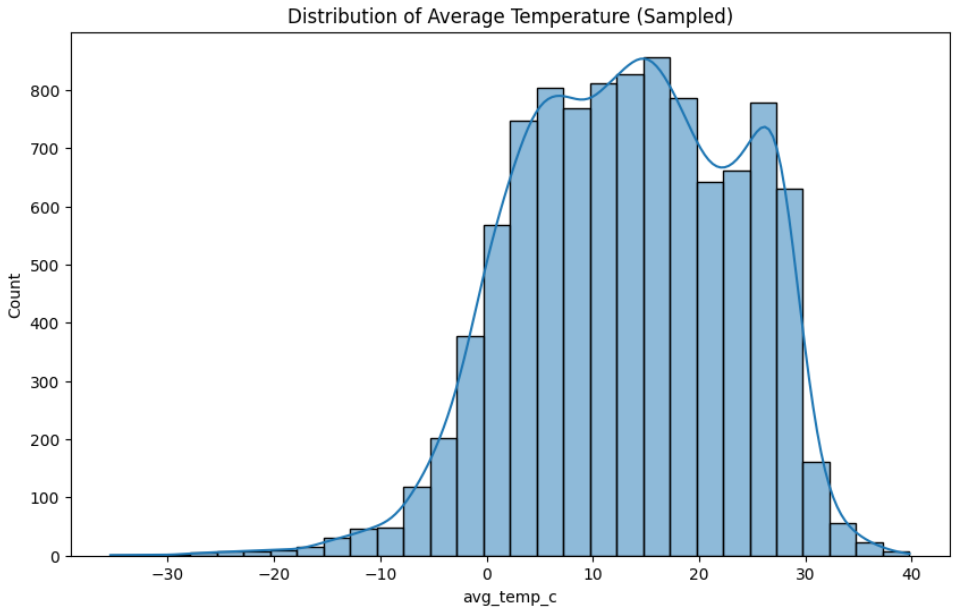

Figure 1: Distribution of Average Temperature

5.2 Temperature vs. Precipitation

The scatter plot below illustrates the relationship between average temperature and precipitation. This helps us understand if rain occurs more frequently at certain temperatures.

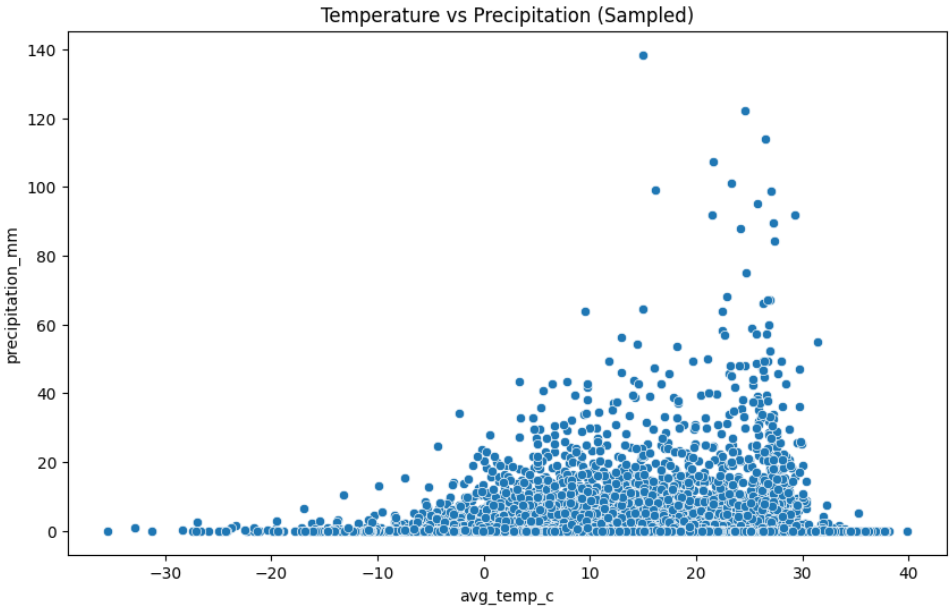

Figure 2: Scatter Plot: Temperature vs. Precipitation

5.3 Correlation Heatmap

To identify the strongest predictors for our model, we generated a correlation heatmap. Darker colors indicate stronger correlations. As expected, `min_temp_c` and `max_temp_c` show a very high positive correlation with `avg_temp_c`.

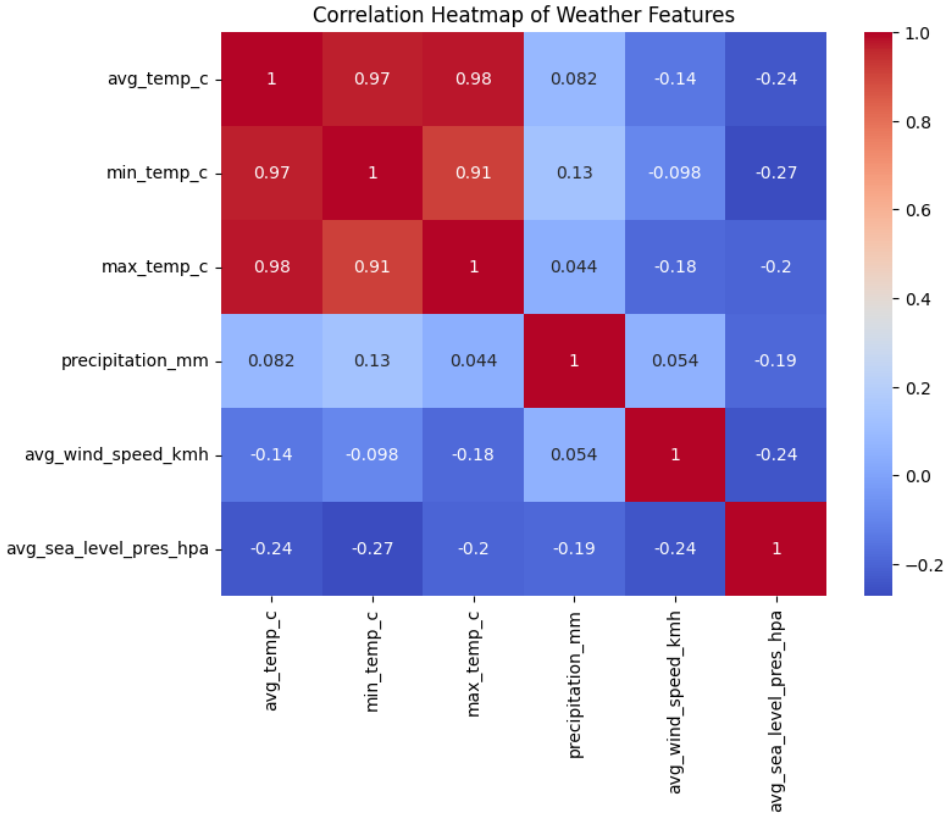

Figure 3: Correlation Heatmap of Weather Features

5.4 Seasonal Analysis (Boxplots)

The boxplot below displays the spread of average temperatures across different seasons. This visualization confirms the seasonal nature of the data, with distinct median temperatures for Summer and Winter.

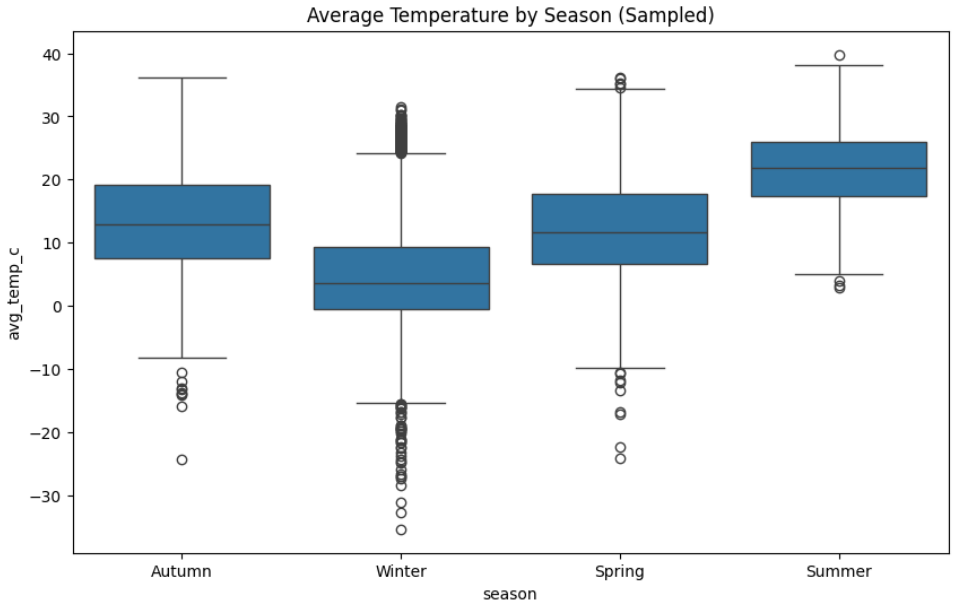

Figure 4: Boxplot: Average Temperature by Season

5.5 Pairplot of Features

A pairplot was generated to visualize pairwise relationships between all major numerical features. This provides a comprehensive view of the data structure.

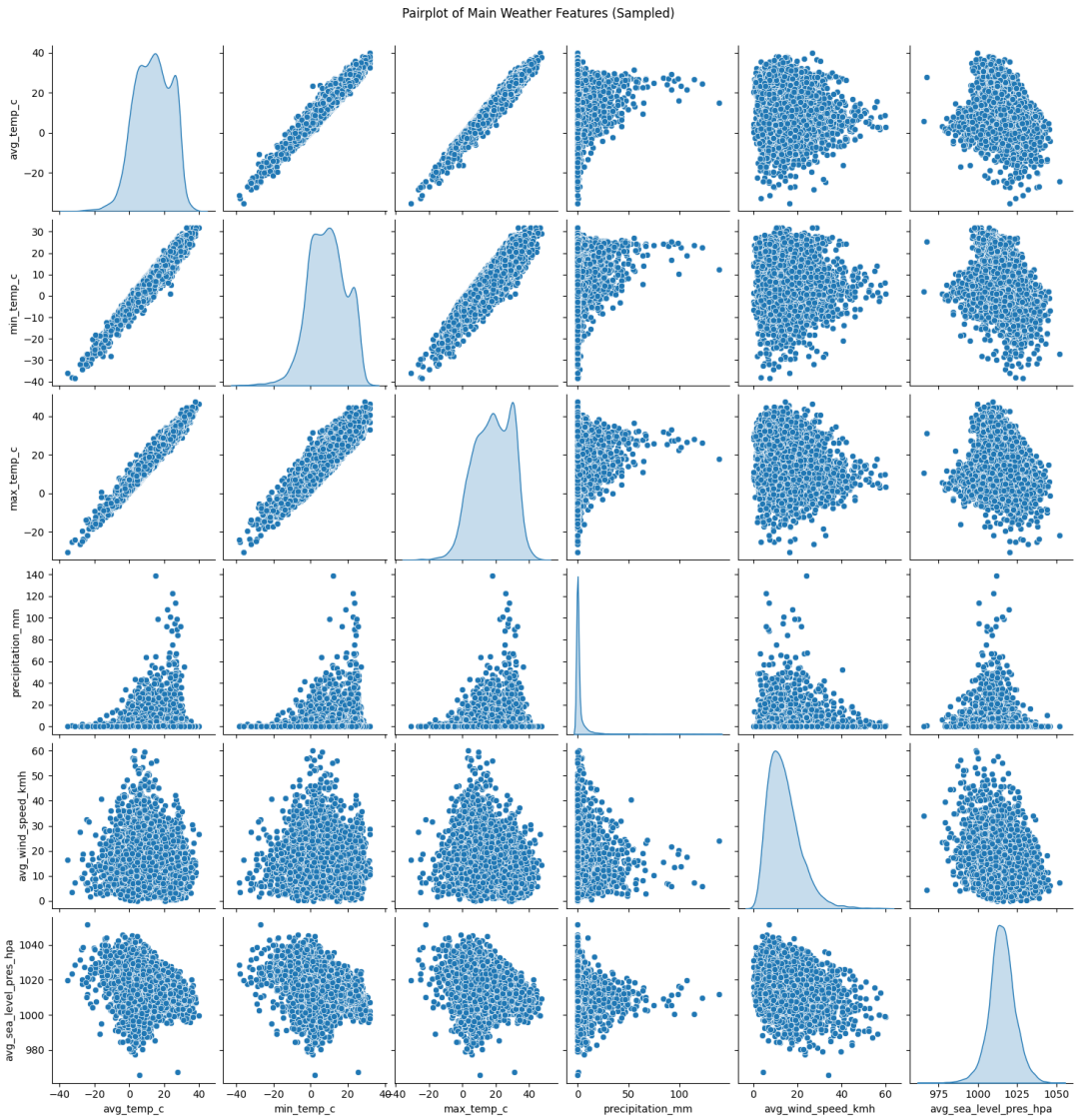

Figure 5: Pairplot of Main Weather Features

5.6 Time Series Analysis

The line plot below shows the trend of average temperature over time. This helps in identifying long-term trends or anomalies in the weather data.

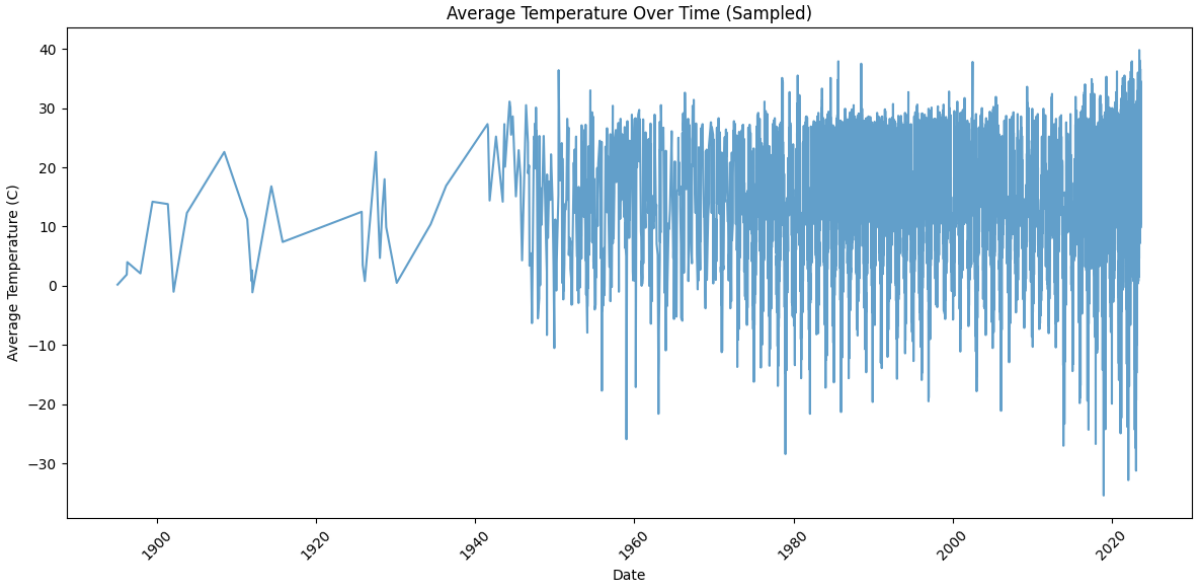

Figure 6: Time Series: Average Temperature Over Time

6 Methodology: Model Building

6.1 Feature Engineering

Machine learning algorithms in Spark MLlib require features to be assembled into a single vector column. We used the `VectorAssembler` transformer for this purpose.

```
assembler = VectorAssembler(
    inputCols=['min_temp_c', 'max_temp_c', 'precipitation_mm', ...],
    outputCol="features"
)
```

6.2 Data Splitting

The cleaned data was split into training and testing sets using an 80-20 split. This allows us to train the model on one subset and evaluate its performance on unseen data.

6.3 Model Selection: Linear Regression

We instantiated a Linear Regression model. Linear regression is suitable here as we are predicting a continuous numerical value (`avg_temp_c`) based on other continuous variables.

6.4 Hyperparameter Tuning with Cross-Validation

To optimize the model and prevent overfitting, we did not just fit a single model. Instead, we used:

- **ParamGridBuilder**: To define a grid of hyperparameters to search. We tuned `regParam` (regularization) and `elasticNetParam`.
- **CrossValidator**: To perform 3-fold cross-validation, selecting the best model based on the RMSE metric.

```
paramGrid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.01, 0.1]) \
    .addGrid(lr.elasticNetParam, [0.0, 0.5]) \
    .build()

crossval = CrossValidator(estimator=lr,
                          estimatorParamMaps=paramGrid,
                          evaluator=RegressionEvaluator(metricName="
   rmse"),
                          numFolds=3)
```

7 Results and Evaluation

7.1 Evaluation Metrics

The best model obtained from the cross-validation process was evaluated on the test dataset. The primary metrics used were Root Mean Squared Error (RMSE) and the Coefficient of Determination (R^2).

Metric	Description	Value
RMSE	Root Mean Squared Error	*1.1039622*
R^2	R-Squared Score	*0.988991]*

Table 1: Model Performance Metrics

7.2 Interpretation

- An **RMSE** of *1.1039622* indicates that, on average, our model's predictions are within *1.1039622* degrees Celsius of the actual temperature.
- An R^2 of *0.988991* implies that our model explains *0.988991%* of the variance in the target variable. A value close to 1 indicates an excellent fit.

7.3 Actual vs. Predicted Visualization

The scatter plot below compares the actual temperatures (x-axis) with the predicted temperatures (y-axis). The red dashed line represents a perfect prediction. The close clustering of points around this line confirms the model's accuracy.

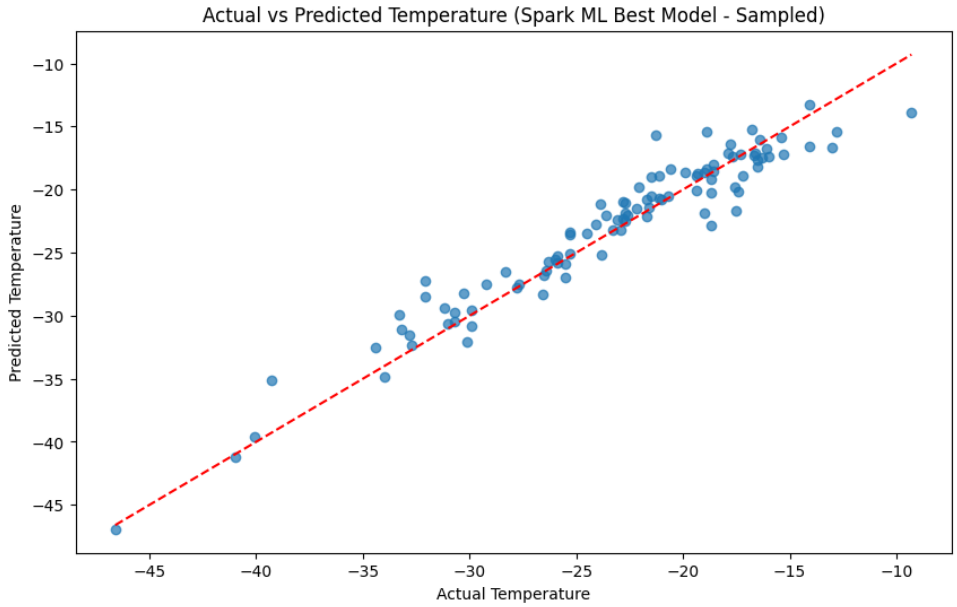

Figure 7: Scatter Plot: Actual vs. Predicted Temperature

8 Conclusion

In this project, we successfully implemented a Big Data Analytics pipeline using Apache Spark. We started by ingesting raw weather data and performed rigorous data cleaning and transformation to prepare it for analysis. Through Exploratory Data Analysis, we gained valuable insights into the relationships between various weather parameters.

We then built a Linear Regression model using Spark MLlib. By employing Cross-Validation and Hyperparameter Tuning, we ensured that our model was robust and optimized. The high R^2 score and low RMSE obtained on the test set demonstrate that the average daily temperature can be accurately predicted using features like minimum/-maximum temperature and atmospheric pressure.

This project highlights the efficiency of PySpark for handling large datasets and the power of Spark MLlib for building scalable machine learning applications. Future work could involve incorporating more complex models like Random Forests or Gradient Boosted Trees, or analyzing the data over a longer time horizon to detect climate change patterns.