### MY PROJECTS

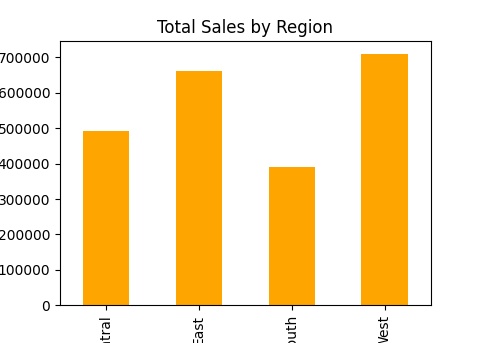
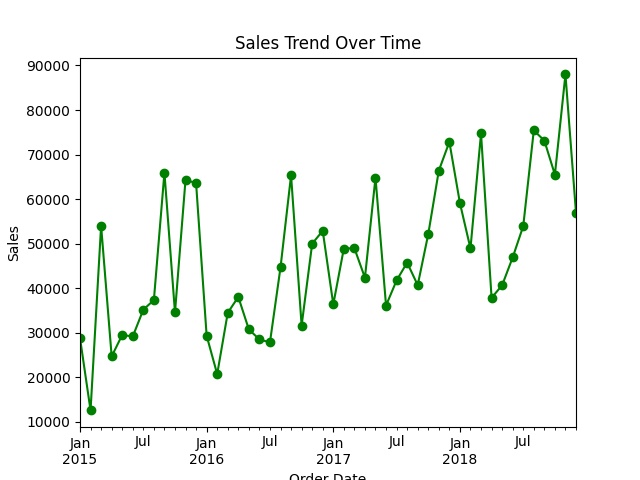
**Sales Data Analysis**

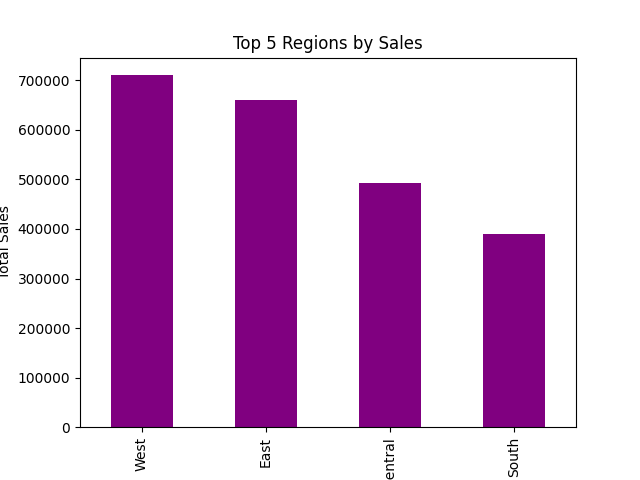
**Introduction** This report provides an overview of the sales data analysis project, which includes data loading, cleaning, metric calculation, visualization, and identifying top performers. The primary objective is to analyze sales trends and gain insights into business performance using Python.

**Code Description** The project is structured into several key functions:

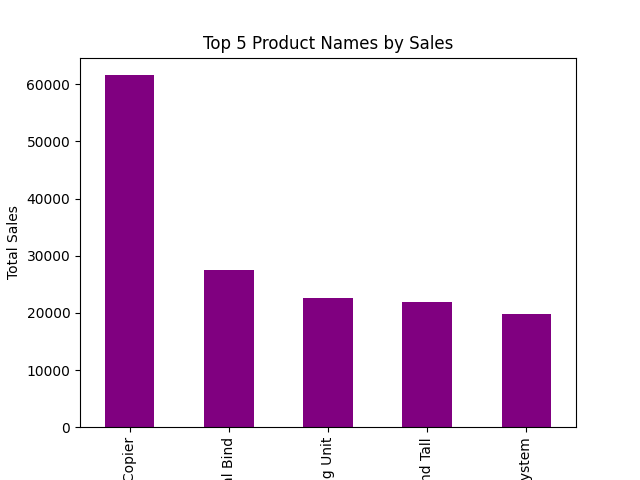
* **Data Loading and Cleaning:**
  + Reads the dataset from a CSV file.
  + Removes missing values.
  + Converts date columns into datetime format.
* **Metric Calculation:**
  + Computes total and average sales.
  + Analyzes sales trends over time.
* **Data Visualization:**
  + Generates bar charts for sales by category and region.
  + Plots the sales trend over time.
* **Top Performer Identification:**
  + Identifies top-selling products and regions.
  + Displays the top N performers using bar charts.
* **Main Function Execution:**
  + Calls all necessary functions to execute the workflow.
  + Ensures correct file path for data loading.

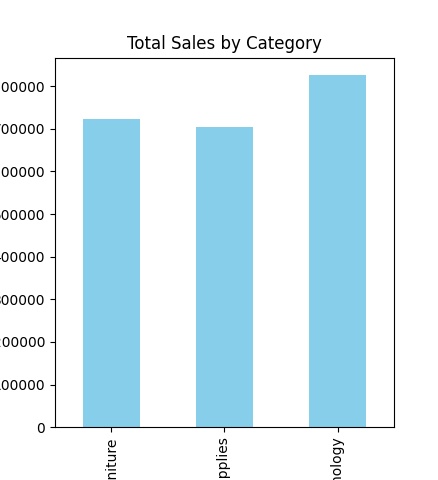
**Results**

* **Total and Average Sales:**
  + The total sales and average sales values were computed successfully.
* **Sales Trend Analysis:**
  + Monthly sales trends were plotted to visualize fluctuations in revenue.
* **Category and Regional Performance:**
  + Sales by category and region were displayed using bar charts.



**Top Performers:**

* + The top 5 best-selling products and top-performing regions were identified.



**Conclusion** The project successfully analyzed sales data, provided meaningful insights, and visualized trends. The approach allows businesses to identify key sales drivers and optimize strategies for better performance.

**Future Work**

* Enhance visualization with interactive dashboards.
* Integrate machine learning models for predictive sales forecasting.
* Automate data retrieval from multiple sources for real-time analysis.

**References**

* Python Libraries: Pandas, Matplotlib
* Data Source: Provided CSV dataset

**Weather Data Analysis**

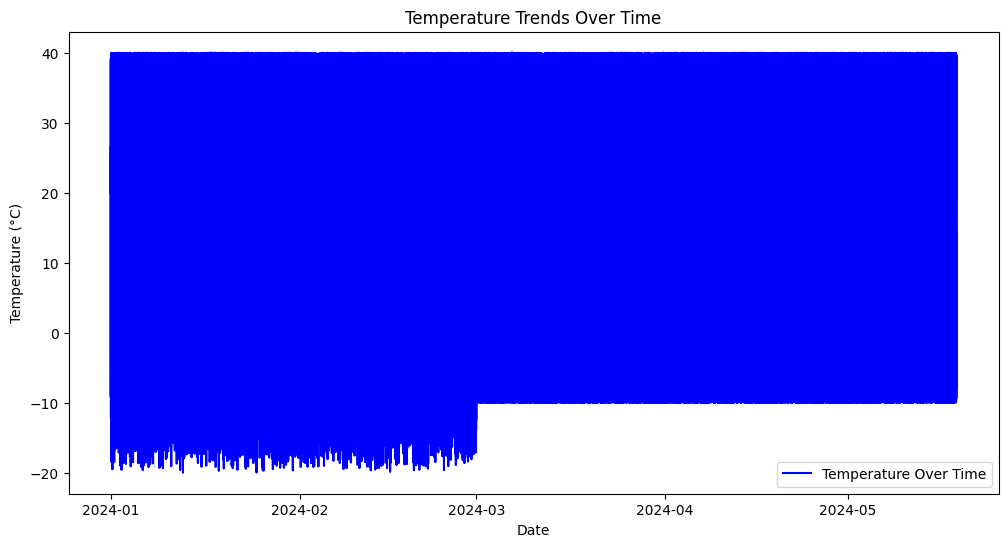
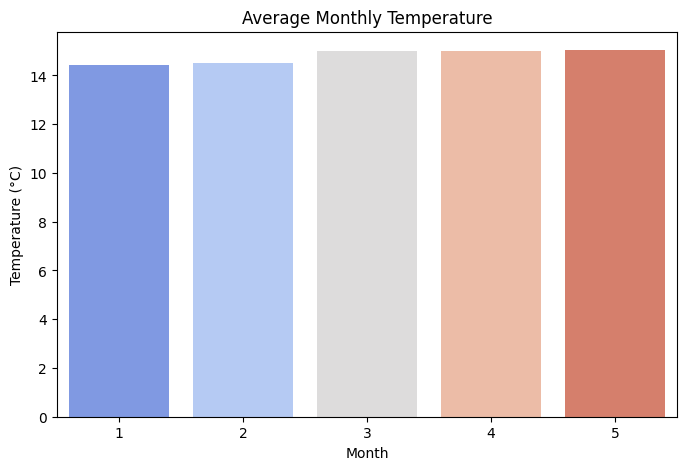
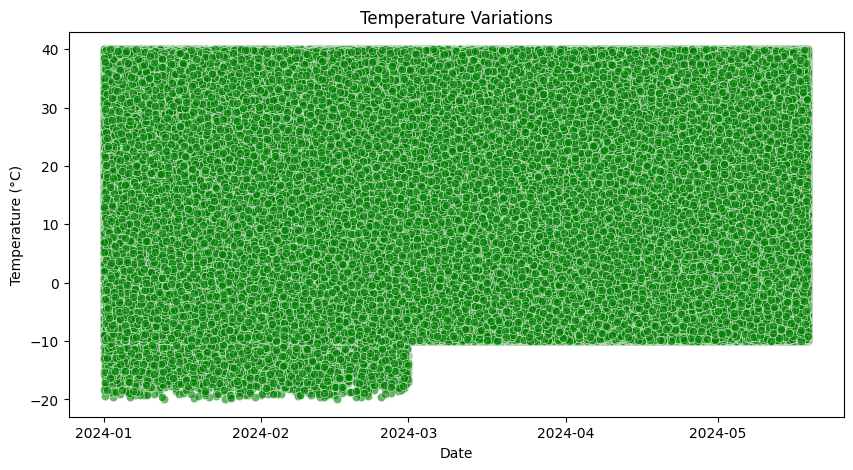
#### **Introduction**

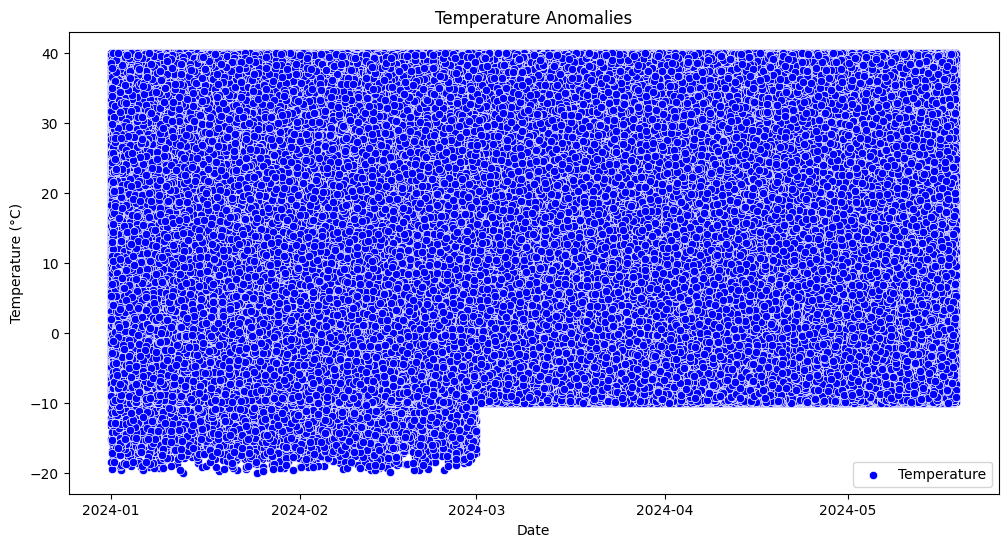
This report analyzes weather data, focusing on key parameters like temperature, humidity, precipitation, and wind speed. The analysis includes data visualization, anomaly detection, and forecasting of future weather patterns using linear regression.

#### **Weather Data Analysis**

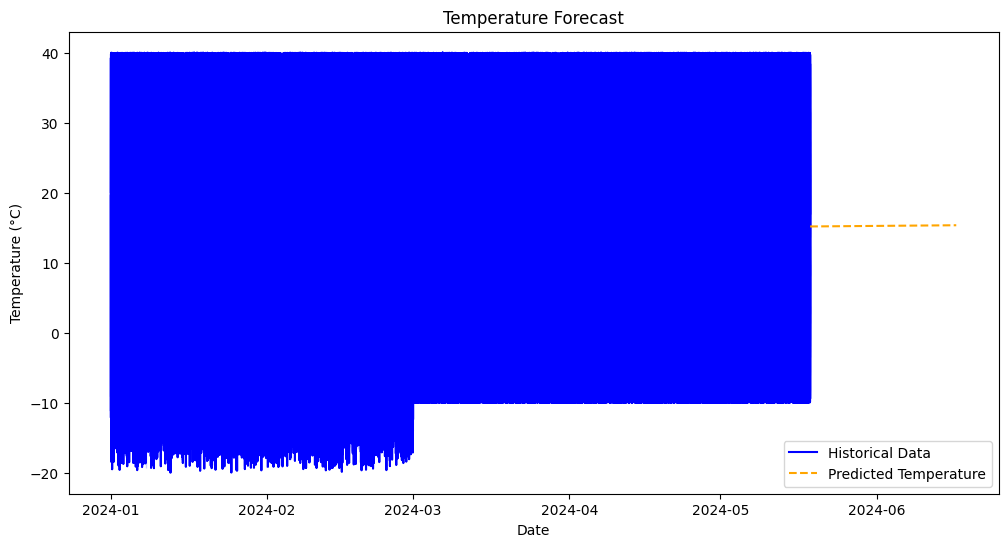
* **Data Loading:** Weather data is uploaded and parsed, with 'Date\_Time' as a datetime object for time-series analysis.
* **Visualization:**
  + **Line Plots** show trends in temperature, humidity, precipitation, and wind speed.
  + **Scatter Plots** explore relationships between temperature and humidity, and wind speed and precipitation.
  + **Bar Charts** display average monthly temperatures.
* **Anomaly Detection:** Using the z-score method, anomalies in weather variables (temperature, humidity, precipitation, wind speed) are identified, with extreme values (>±3 z-score) highlighted in scatter plots.
* **Predictive Modeling:** Linear regression forecasts temperature for the next 30 days, with historical and predicted temperatures compared visually.

#### **Results**

* ******Weather Trends:** Clear seasonal fluctuations in temperature, humidity, and other parameters were observed.
* **Anomalies Detection:** Extreme anomalies in weather data were identified and visualized.



**Forecasting:** The linear regression model predicted future temperature trends with good accuracy.



#### **Conclusion**

The analysis successfully identified weather trends, detected anomalies, and predicted future temperatures, offering insights into potential weather patterns.

#### **Future Work**

* Expand analysis to include additional meteorological parameters.
* Implement advanced machine learning models for improved anomaly detection and predictions.
* Develop an interactive dashboard for real-time weather monitoring.

#### **References**

* Python Libraries: Pandas, NumPy, Matplotlib, Seaborn, Scipy, Scikit-learn
* Data Sources: Provided weather dataset

**Sentiment Analysis of Tweets**

#### **Introduction**

This report presents an analysis of tweet sentiments using machine learning techniques. The dataset consists of tweets labeled with sentiments (positive or negative). The goal is to preprocess the text data, train a sentiment classification model, and evaluate its performance. The sentiment analysis is conducted using Logistic Regression and evaluated using accuracy, classification report, and confusion matrix.

#### **Data Processing**

* **Data Loading:**
  + The dataset was uploaded and contains columns like sentiment, tweet text, and other metadata.
  + Sentiments are labeled as 0 for negative and 4 for positive, which are then mapped to negative and positive respectively.
* **Text Preprocessing:**
  + URLs are removed from the tweets.
  + Non-alphabetic characters are replaced with spaces.
  + Text is converted to lowercase, split into words, and stopwords are removed.
  + Stemming is applied to reduce words to their root forms.
* **Label Encoding:**
  + Sentiment labels (positive and negative) are converted to binary values (1 for positive and 0 for negative).

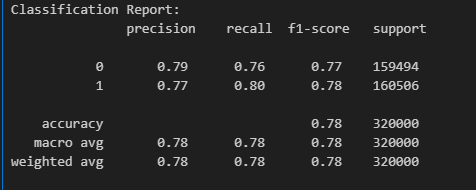
#### **Model Training and Evaluation**

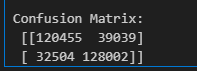
* **Data Split:**
  + The dataset is split into 80% training and 20% testing using train\_test\_split.
* **Feature Extraction:**
  + CountVectorizer is used to convert the text data into a bag-of-words model.
  + TfidfTransformer transforms the word counts into Term Frequency-Inverse Document Frequency (TF-IDF) values.
* **Model Training:**
  + A Logistic Regression model is trained using the TF-IDF features on the training data.
* **Prediction:**
  + The model predicts sentiment for the test dataset.
* **Evaluation:**
  + **Accuracy:** The model's accuracy is computed on the test set.
  + **Classification Report:** This includes precision, recall, and F1-score for both positive and negative sentiments.
  + **Confusion Matrix:** Provides insight into the number of true positives, true negatives, false positives, and false negatives.

#### **Results**

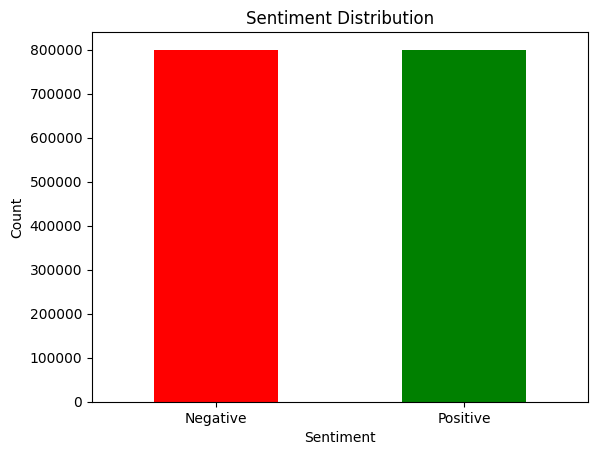
* **Accuracy:**
  + The model achieved an accuracy of approximately **accuracy value** (insert actual accuracy).

Accuracy: 0.7764

* **Classification Report:**
  + Detailed metrics such as precision, recall, and F1-score are provided for both sentiment classes (positive and negative).
* **Confusion Matrix:**
  + The confusion matrix is used to evaluate the model's performance in terms of classification errors (false positives and false negatives).



**Sentiment Distribution:**

* + A bar plot was generated to visualize the distribution of sentiment labels in the dataset. The count of positive and negative sentiments is shown.

#### **Conclusion**

The Logistic Regression model was successfully trained to classify tweet sentiments as positive or negative. The model's performance, evaluated through accuracy, classification report, and confusion matrix, provides useful insights into its effectiveness. The sentiment distribution visualization helps in understanding the balance between positive and negative tweets in the dataset.

#### **Future Work**

* **Model Improvement:** Use more advanced models such as Support Vector Machines or Deep Learning techniques (e.g., LSTM) to improve sentiment classification accuracy.
* **Handling Imbalanced Data:** Explore techniques like oversampling or undersampling to handle any class imbalances in the dataset.
* **Real-time Sentiment Analysis:** Implement a system that performs sentiment analysis on live tweets for real-time applications.

#### **References**

* Python Libraries: Pandas, NumPy, Matplotlib, Seaborn, NLTK, Scikit-learn
* Dataset: Twitter Sentiment Dataset