Weather Data Analysis Report

Purpose and Insights from Data Processing and Advanced Analysis

Assignment 2

Muhammad Abdullah
Roll No: 22P-9371
BCS-6B
Submitted to: Instructor Muhammad Zulqarnain

March 12, 2025

Abstract

This report details the analysis of weather event data from `weather_records.csv`, focusing on data preprocessing, feature engineering, and advanced analytical tasks. Each task's purpose is articulated, alongside methods used (e.g., Python with Pandas, Matplotlib, Seaborn, Scikit-learn, Cartopy) and insights derived. The objectives include enhancing data usability, uncovering patterns, and preparing the dataset for predictive modeling or further research.

1 Introduction

Weather data analysis is critical for understanding environmental patterns, improving forecasting, and informing decision-making. This report outlines a series of tasks applied to a dataset containing weather events with attributes like `StartTime(UTC)`, `EndTime(UTC)`, `Precipitation(in)`, `Severity`, `LocationLat`, and `LocationLng`. The tasks range from cleaning and basic feature extraction to advanced feature engineering and analytical techniques.

2 Task A: Data Preprocessing and Feature Engineering

2.1 A.1: Data Cleaning

Purpose: Ensure data integrity by handling missing values and inconsistencies, making the dataset suitable for analysis.
Method: Forward-filled time-related columns (`StartTime(UTC)`, `EndTime(UTC)`, `Precipitation(in)`), used mode-based imputation for location fields (e.g., `City`, `ZipCode`) based on `AirportCode`, and dropped rows with missing `Type` or `Severity`.
Outcome: A complete dataset with no missing values in critical fields, preserving sequential weather patterns and geographical consistency.

2.2 A.2: Temporal Feature Extraction

Purpose: Extract temporal features to enable time-based analysis of weather patterns.
Method: Converted `StartTime(UTC)` from `MM/DD/YY HH:MM` format to datetime, extracted `Hour`, `DayOfWeek`, and `Month`, and preserved the slash-based format in the output CSV.
Outcome: Added features:

- `Hour`: Captures diurnal cycles (e.g., afternoon storms).
- `DayOfWeek`: Identifies weekly patterns or biases (e.g., 0 = Monday).
- `Month`: Highlights seasonal trends (e.g., winter snow).

3 Task B: Encoding Categorical Variables

Purpose: Transform categorical data into numerical form for statistical or machine learning applications.
Method: Applied label encoding to `Severity` using Scikit-learn's `LabelEncoder`, mapping ordinal categories (e.g., Light = 0, Moderate = 1, Severe = 2).
Choice Justification: Label encoding was chosen due to `Severity`'s ordinal nature, where order reflects intensity. One-hot encoding would ignore this hierarchy, making it less suitable.
Outcome: Added `Severity_Encoded`, enabling ordinal-aware analysis or modeling.

4 Task C: Advanced Feature Engineering

4.1 C.1: Create New Features

Purpose: Enhance the dataset with derived features to capture event duration and seasonal effects, facilitating deeper temporal analysis.
Method: Calculated `Duration` as the difference between `EndTime(UTC)` and `StartTime(UTC)` in hours using Pandas datetime operations. Extracted `Month` from `StartTime(UTC)` if not present, then mapped it to `Season` (Winter: Dec-Feb, Spring: Mar-May, Summer: Jun-Aug, Fall: Sep-Nov) using a custom function. Time columns were converted back to `MM/DD/YY HH:MM` format for consistency.
Outcome: Added `Duration` and `Season`, enabling analysis of event persistence and seasonal patterns, crucial for weather modeling.

4.2 C.2: Encoding Categorical Variables

Purpose: Convert the categorical `Severity` column into a numerical format suitable for advanced statistical or predictive tasks.
Method: Used Scikit-learn's `LabelEncoder` to transform `Severity` into `Severity_Encoded`, assigning integers to ordinal categories (e.g., Light = 0, Moderate = 1, Severe = 2).
Choice Justification: Label encoding was selected because `Severity` exhibits an ordinal relationship, where numerical order reflects increasing intensity. One-hot encoding would treat categories as independent, disregarding this inherent hierarchy.
Outcome: Added `Severity_Encoded`, enhancing the dataset's compatibility with machine learning algorithms while preserving ordinal meaning.

5 Task D: Advanced Analysis

5.1 D.1: Correlation Analysis

Purpose: Investigate the relationship between `Precipitation(in)` and `Duration` to understand if longer events correlate with higher precipitation.
Method: Computed Pearson correlation using `data.corr()` and visualized it with a Seaborn heatmap (`coolwarm` colormap, annotated values).
Insight: A positive correlation (e.g., 0.4) suggests longer events may involve more precipitation, though the strength varies (weak: < 0.3, strong: > 0.7). This informs weather intensity models.

5.2 D.2: Outlier Detection

Purpose: Identify anomalies in `Precipitation(in)` to assess data quality and extreme events.
Method: Used the IQR method (Q1 - 1.5*IQR, Q3 + 1.5*IQR) to detect outliers, visualized with a styled Seaborn boxplot (dark red fliers for outliers).
Insight: Outliers (e.g., values > upper bound) may represent heavy storms. Handling options include capping at bounds for normalization or retaining them as valid extremes, avoiding data loss.
Outcome: Quantified outliers (e.g., 5 out of 1000 points) and visualized their distribution.

5.3 D.3: Geospatial Analysis

Purpose: Map weather events geographically to reveal spatial patterns and clustering on a real-world map.
Method: Plotted `LocationLat` and `LocationLng` using Cartopy with a PlateCarree projection, overlaying scatter points on a map with land, ocean, and border features. Points were colored by `Precipitation(in)` and sized by `Duration`.
Insight: Clustering occurs around specific coordinates (e.g., Saguache, CO: 38°N, -106°W; Albert Lea, MN: 43°N, -93°W), likely airport stations. Darker colors (high precipitation) and larger markers (long duration) indicate intense, persistent events in these areas, visualized against a geographical backdrop.
Outcome: A real-world map highlighting localized weather activity with clear spatial context.

6 Conclusion

The tasks collectively transform raw weather data into an analytically rich dataset. Preprocessing ensures reliability, feature engineering uncovers temporal and spatial dimensions, encoding prepares data for modeling, and advanced analyses reveal correlations, outliers, and geographical distributions. These steps support applications like weather prediction, anomaly detection, and regional planning.

Appendix: Tools Used

- **Python Libraries**: Pandas (data manipulation), Matplotlib/Seaborn (visualization), Scikit-learn (encoding), Cartopy (geospatial plotting).
- **Environment**: Jupyter Notebook, data from `weather_records.csv`.