PCLP3 Final Project (2025)

Grigore Mihnea – Andrei, 311CA

Synthetic Data Generation

Configuration and Library Imports

The code begins by configuring the *Matplotlib* library to use the *Agg* backend. This non-interactive backend allows the generation of plots in environments without graphical interfaces (to avoid run errors). Following this, several essential libraries are imported: *pandas* and *numpy* for data manipulation and numerical operations, *random* for randomness, *seabron* and *matplotlib.pyplot* for visualization, and AI tools from *scikit-learn*.

Definition of Dataset Parameters

A list of ten major cities serve as possible categorical values for the City feature, a collection of typical weather conditions such as 'Clear', 'Cloudy', 'Rainy', 'Stormy', and 'Snowy', and a range of wind speeds representing different weather intensities. The temperature range is defined as integers between -10 and 40 degrees Celsius.

Synthetic Data Generation Process

The *generate_weather_data* function takes three parameters: the number of samples to generate, the probabilities controlling the introduction of missing data and outliers. For each sample, the function randomly selects values for each feature according to the defined parameters. Numeric features like temperature, wind speed, humidity, and pressure are generated within realistic bounds, but also include the possibility of extreme outlier values, such as unrealistic temperatures (-100, 100, 200) or abnormal wind speeds (50, 100). This outlier injection, controlled by the *outlier_prob* parameter, helps to simulate the anomalies that may appear in real-world datasets.

Additionally, the function randomly assigns missing values to the numeric features independently, done by the *missing_prob* parameter. This approach realistically simulates incomplete data often encountered in practice. Categorical features, such as *Visibility*, are selected from a limited set of values, while *DateTime* is randomly sampled from a range.

Dataset Creation and Export

Using the described generation function, the script creates two distinct datasets: a training set with 500 instances and a testing set with 200 instances. Both datasets incorporate missing values and outliers with equal probability. Finally, the datasets are exported to CSV files named *'train_data.csv'* and *'test_data.csv'*. These files exclude row indices to maintain an easy operation forward in the code.

Handling Missing Values

The function *fill_missing_values* processes each column of the dataframe to impute missing values. For categorical columns, it fills missing entries with the column most frequent value. For numerical columns, it replaces missing values with the column mean.

The function is applied to both training and testing datasets separately. The complete resulting datasets are saved as *'train_data_filled.csv'* and *'test_data_filled.csv'*.

Interpretation of Descriptive Statistics

The following interpretation is based on the data extracted in Documentation folder. Since the code uses RANDOM datasets, the following interpretation will not be correct on another run of the code.

Numerical Features:

- *Temperature* ranges from -100 to 200. The mean and median both sit near 13.9°C, with a high standard deviation (~26), indicating wide variability. The interquartile range (2 to 26) captures typical temperature values.
- *Wind* varies between 1 and 100 km/h, again with outliers present. The mean and median (~18) are aligned, and the data shows moderate spread.
- *Humidity* spans 5% to 110%, exceeding physical bounds due to outliers. The mean and median are about 59%, with typical values mostly between 42% and 79%.
- *Pressure* ranges from 950 to 1050, and includes some extremes. The mean (~1002) and median are close, and variability is moderate.
- *DateTime* covers the full range from 2000 to 2024, centered around 2012.

Categorical Features:

- *City* includes all 10 cities, with *'Berlin'* most frequent (61 samples).
- *Condition* includes five weather types; *'Snowy'* is most common (110 samples).
- *Visibility* categories are *'Good'*, *'Moderate'*, and *'Poor'*, with *'Good'* most frequent (182 samples).

Descriptive Statistics and EDA

Distribution Analysis

The *plot_distributions* function generates visual summaries for both numerical and categorical variables. For numerical columns, it creates histograms to reveal data spread and modality. For categorical columns, countplots visualize the frequency distribution of categories. A barplot shows average temperature per city, highlighting possible differences in temperature patterns across locations.

Outlier Detection

The *detect_outliers* function applies the IQR method to identify outliers in each numerical column. It computes Q1 and Q3 and calculates the IQR, lower and upper bounds. These bounds define the range beyond which values are considered outliers. For each numerical feature, boxplots including outliers are generated and saved, providing visual identification of extreme values.

Correlation Analysis

The *correlation_analysis* function calculates the correlation matrix among numerical features. A heatmap visualizes these correlations with color gradients, annotated with numeric values for clarity. Both the heatmap image and raw correlation matrix are saved in the folder.

Relationship with Target Variable

Finally, the *plot_target_relationships* function explores how selected numerical features relate to the target variable (*Temperature*). It generates scatter plots to show continuous relationships between each feature and the target, highlighting trends or patterns. If the target were categorical, violin plots would be used instead. Each plot is saved individually.

Analysis of Histograms and Plots

Histogram for Wind

- **What do we observe?** The distribution is right-skewed, most values are below 40, but there are some extreme values up to 100.
- **What suspicions/ideas can we formulate?** Presence of outliers or abnormal high values.
- **What preprocessing should be applied?** Detect and possibly remove or transform outliers; apply scaling.

Histogram for Temperature

- **What do we observe?** Temperature roughly follows a normal distribution but with extreme negative and positive outliers.
- **What suspicions/ideas can we formulate?** Presence of abnormal values that may affect models.

- **What preprocessing should be applied?** Truncate or replace outliers; impute missing values if any.

Histogram for Pressure

- **What do we observe?** Atmospheric pressure is centered around the mean with slight extremes at the tails.
- **What suspicions/ideas can we formulate?** Possible outliers at both low and high ends.
- **What preprocessing should be applied?** Analyze and treat outliers.

Histogram for Humidity

- **What do we observe?** Humidity is fairly uniformly distributed between 20 and 100, with some very low and very high values.
- **What suspicions/ideas can we formulate?** Possible erroneous values.
- **What preprocessing should be applied?** Verify and correct erroneous values; impute missing values.

Countplot for Visibility

- **What do we observe?** ‘Good’, ‘Moderate’, and ‘Poor’ visibility classes are relatively evenly distributed, with ‘Good’ slightly more frequent.
- **What suspicions/ideas can we formulate?** ‘Good’ class might dominate model learning due to higher frequency.
- **What preprocessing should be applied?** Possible class balancing; encode categories numerically for modeling.

Countplot for Condition

- **What do we observe?** Weather conditions are evenly distributed, with ‘Snowy’ slightly more frequent.
- **What suspicions/ideas can we formulate?** No major imbalance, but possible correlation with other features.
- **What preprocessing should be applied?** Apply categorical encoding.

Countplot for City

- **What do we observe?** Cities are evenly represented with no dominant city.
- **What suspicions/ideas can we formulate?** No balancing needed due to relatively uniform counts.
- **What preprocessing should be applied?** Categorical encoding for city.

Correlation Matrix Heatmap

- **What do we observe?** No strong correlations between numerical variables; all coefficients near zero.
- **What suspicions/ideas can we formulate?** Features are mostly independent or weakly correlated.
- **What preprocessing should be applied?** Consider nonlinear relationships or advanced feature selection techniques.

Barplot for City vs Temperature

- **What do we observe?** Average temperature varies by city; Madrid has the highest mean, Sydney the lowest.
- **What suspicions/ideas can we formulate?** City is an important factor influencing temperature and should be included in modeling.
- **What preprocessing should be applied?** Encode city variable; consider grouping effects.

Training and Evaluating a Basic Regression Model

The code trains a Ridge Regression model to predict the target variable *"Temperature"*.

It reads preprocessed train and test CSV files, separates features and target, and applies one-hot encoding to categorical features. The feature sets are aligned to ensure matching columns, then standardized.

The Ridge model is trained on scaled training data and used to predict test targets.

Model quality is evaluated with RMSE, MAE, and R^2 metrics. Two plots are generated and saved: a scatter plot of predicted vs. actual values, and a histogram of residuals to assess error distribution.

Since the datasets are RANDOM generated, we can predict that the Regression Model would not have the same results every time. For example, in the current run, the Regression Model had an accuracy of -0,8 in terms of R^2, which is quite good because it aproaches 0. Other tests gave results lower than -1 (even -2 or -3), meaning a worse prediction of the input data.