## Project Documentation: Build and Train a Simple Machine Learning Model

### Project Title: Build and Train a Simple Machine Learning Model

### Step 1: Importing Required Libraries

The essential libraries for data handling, statistical analysis, and machine learning were imported as follows:

- \*\*Pandas\*\* and \*\*NumPy\*\* were utilized for data manipulation and numerical operations.

- \*\*Scikit-Learn\*\* provided machine learning utilities for model training, encoding, scaling, and error calculations.

- \*\*Statsmodels\*\* was used for detailed statistical modeling and to apply the Ordinary Least Squares (OLS) regression.

- \*\*LinearRegression\*\* from Scikit-Learn and additional algorithms like Decision Tree were also imported for comparative model training.

### Step 2: Loading and Merging Data Files

Two datasets related to World War II weather were loaded: `Summary of Weather.csv` and `Weather Station Locations.csv`. These files were concatenated to create a comprehensive dataset for predicting mean temperatures. This merging provided a unified dataset containing both weather summaries and station-specific information, which enabled a richer analysis and facilitated predictive modeling.

### Step 3: Exploratory Data Analysis (EDA)

A thorough exploratory data analysis (EDA) was conducted to understand the structure, distribution, and quality of the data:

1. \*\*Data Overview\*\*: The dataset’s structure was inspected by checking data types, dimensions, and null values.

2. \*\*Descriptive Statistics\*\*: Basic statistical metrics (mean, standard deviation, etc.) were computed to gain initial insights into numeric distributions.

3. \*\*Feature Engineering\*\*: Type conversions were applied as needed, and unnecessary columns were dropped to streamline the dataset.

4. \*\*Missing Values\*\*: Missing data points were identified to determine appropriate imputation strategies.

5. \*\*Duplicate Entries\*\*: The dataset was examined for duplicates, which were removed to maintain data integrity.

6. \*\*Column Partitioning\*\*: Columns were divided into numeric and categorical groups to allow tailored preprocessing steps.

### Step 4: Data Preprocessing Pipeline

A data preprocessing pipeline was designed to manage data transformation in a systematic and reproducible manner. The steps included:

1. \*\*Outlier Removal\*\*: The Winsorization technique was applied to limit extreme values in numeric features, reducing the effect of outliers without significantly altering the data’s structure.

2. \*\*Mean Imputation\*\*: Missing numeric values were imputed with the mean, preserving the overall data distribution.

3. \*\*One-Hot Encoding\*\*: Categorical features were transformed into binary columns, preventing any ordinal bias in model training.

4. \*\*Scaling\*\*: MinMaxScaler was employed to normalize numeric features between 0 and 1, making the dataset more compatible with machine learning algorithms sensitive to value ranges.

All preprocessing steps were integrated into a pipeline and managed through a \*\*ColumnTransformer\*\* for efficient data processing.

### Step 5: Conversion to DataFrame and Concatenation

Post-preprocessing, the transformed numeric and categorical features were converted back into a unified DataFrame, merging them into a single structure. This consolidated format ensured consistency and readiness for modeling.

### Step 6: Post-Processing Data Analysis

Following preprocessing, additional analysis was conducted to validate data transformations and examine relationships:

1. \*\*Skewness and Kurtosis\*\*: These measures were analyzed to understand data distribution symmetry and peakedness, providing insights into the effectiveness of outlier handling.

2. \*\*Correlation Analysis\*\*: Correlations were calculated to assess feature relationships, which helped in understanding dependencies and potential multicollinearity issues.

3. \*\*Automated EDA with Sweetviz\*\*: An automated report was generated using Sweetviz, summarizing the dataset’s structure, distributions, and relationships visually, which streamlined data documentation.

### Step 7: Data Partitioning for Model Training and Testing

The dataset was partitioned into training and testing sets, with 80% allocated for training and 20% for testing. This split ensured that a sufficient amount of data was available for training while leaving ample data for model evaluation.

### Step 8: Preparing Data for Statsmodels

To facilitate training with \*\*Statsmodels\*\*, the training features and target variable were combined into a single DataFrame. This format aligned with the requirements of the OLS model in Statsmodels, allowing for seamless model training.

### Step 9: Model 1 - Ordinary Least Squares (OLS) Regression

An Ordinary Least Squares (OLS) regression model was initialized using Statsmodels. The model was trained on the training data with the predictor (`X`) and target (`y`) variables specified, and subsequently fitted. A summary of the model was generated, providing details on coefficients, R-squared values, and statistical significance, which offered insights into feature importance and model fit.

### Step 10: Model 1 Predictions

The trained OLS model was used to predict target values on the test set. These predictions provided an initial assessment of the model’s ability to generalize to unseen data.

### Step 11: Model 1 Evaluation Metrics

The model’s performance was evaluated using error metrics:

- \*\*Mean Error\*\* (ME): Captured average prediction error.

- \*\*Mean Squared Error\*\* (MSE): Penalized larger errors, highlighting areas where predictions were further from actual values.

- \*\*Root Mean Squared Error\*\* (RMSE): Provided a normalized error value, indicating how well the model predicted mean temperature values.

These metrics enabled a detailed understanding of the model’s accuracy and effectiveness in handling the prediction task.

### Step 12: Model 2 - Decision Tree Regression

A Decision Tree Regressor was initialized and trained on the training dataset (features `X\_train` and target `y\_train`). This model served as a comparative approach to the OLS regression, potentially capturing complex, non-linear relationships in the data.

### Step 13: Model 2 Predictions

The Decision Tree model generated predictions for the test set, allowing for a side-by-side performance comparison with the OLS regression model.

### Step 14: Model 2 Evaluation Metrics

Similar error metrics were computed for the Decision Tree model:

- \*\*Mean Error\*\* (ME)

- \*\*Mean Squared Error\*\* (MSE)

- \*\*Root Mean Squared Error\*\* (RMSE)

These metrics offered insights into the performance of the Decision Tree model, specifically in comparison to the OLS regression.

### Step 15: Error Comparison DataFrame

A summary DataFrame was created to compile and compare training and testing error metrics for both the OLS regression and Decision Tree models. This comparative analysis highlighted each model’s strengths and weaknesses, allowing for informed conclusions regarding model selection.