Postgraduate Certificate in AI in Weather Prediction

Data Preprocessing And Visualization

Data collection is the first step in any weather-prediction workflow. It involves gathering observations from a variety of sources such as surface stations, radars, satellites, radiosondes, and numerical weather prediction (NWP) model outputs. Each source provides measurements at different temporal frequencies (e.G., Hourly, 3-hourly) and spatial resolutions (e.G., 0.25° Grid, 1 km radar pixels). Understanding the provenance of each dataset is essential because it influences later decisions about quality control, merging strategies, and bias correction. For example, satellite-derived temperature profiles often require calibration against radiosonde data to correct for sensor drift.

Quality control (QC) procedures aim to detect and flag erroneous observations before they enter the analysis pipeline. Common QC checks include range checks (ensuring temperature values lie within physically plausible limits), consistency checks (comparing wind direction and speed for logical coherence), and spatial checks (identifying isolated outliers that differ dramatically from neighboring grid points). Automated QC algorithms, such as the World Meteorological Organization (WMO) “Standard Atmosphere” checks, can be implemented in Python using the pandas library’s apply method to process each column efficiently.

Missing values are endemic in meteorological datasets due to instrument failure, transmission errors, or satellite orbital gaps. The choice of handling missing data depends on the downstream model. Simple approaches include listwise deletion, where any record containing a missing value is removed; however, this can lead to substantial data loss, especially in sparse networks. More sophisticated techniques involve imputation, where missing entries are estimated using statistical or machine learning methods. Linear interpolation is frequently used for short gaps in time series, while kriging or inverse distance weighting (IDW) may be employed for spatial gaps.

Outlier detection complements QC by identifying observations that deviate markedly from expected patterns. Statistical methods such as the Z-score, modified Z-score, or the Median Absolute Deviation (MAD) provide thresholds for flagging extreme values. Machine-learning based detectors, like isolation forests or one-class SVMs, can capture multivariate outliers that involve complex relationships among temperature, humidity, and wind. Once identified, outliers may be removed, corrected, or retained with reduced weight, depending on the analyst’s judgment and the model’s robustness.

Data smoothing helps reduce high-frequency noise that can obscure underlying climatic signals. Moving-average filters, Gaussian kernels, and Savitzky-Golay filters are common choices. For example, a 24-hour moving average applied to precipitation totals can reveal diurnal cycles while suppressing spurious spikes caused by sensor glitches. In practice, the scipy.Signal module provides ready-to-use functions such

as savgol_filter for this purpose.

Temporal alignment is required when merging datasets with different time stamps. A common approach is to resample all series to a common frequency (e.G., Hourly) using aggregation functions such as mean, sum, or maximum. The pandas resample method simplifies this task, allowing the analyst to specify the desired interval and aggregation rule in a single line of code. Temporal alignment also involves handling time zones and daylight-saving adjustments, which can be managed through the pytz library.

Spatial interpolation bridges gaps between irregularly spaced observations and regular model grids. Techniques range from simple nearest-neighbor interpolation to more advanced geostatistical methods like ordinary kriging, which models spatial autocorrelation through a variogram. For high-resolution radar mosaics, bilinear or bicubic interpolation may be sufficient, whereas climate reanalysis products often rely on sophisticated methods such as spherical harmonics. The pykrige package implements kriging with customizable variogram models, enabling practitioners to tailor interpolation to the specific spatial characteristics of their variable of interest.

Feature engineering transforms raw observations into informative predictors for AI models. In weather prediction, common engineered features include derived thermodynamic quantities such as dew point, potential temperature, and convective available potential energy (CAPE). Temporal features like hour-of-day, day-of-year, and lagged variables (e.G., Previous 6-hour wind speed) capture periodicity and persistence. Spatial features may involve gradients (e.G., Temperature gradient across a front) or neighborhood statistics (e.G., Mean wind speed within a 50 km radius). Feature engineering often requires domain knowledge to ensure that the derived variables retain physical meaning.

Feature selection reduces dimensionality by retaining only the most predictive variables. Filter methods, such as Pearson correlation or mutual information, rank features based on their statistical relationship with the target variable. Wrapper methods, like recursive feature elimination (RFE), evaluate subsets of features using a chosen model and iteratively discard the least important ones. Embedded methods, such as L1-regularized regression (Lasso), integrate selection directly into model training. In the context of weather AI, feature selection helps mitigate overfitting, especially when the number of predictors exceeds the number of training samples.

Dimensionality reduction techniques compress high-dimensional data into a lower-dimensional representation while preserving essential structure. Principal Component Analysis (PCA) is widely used to extract dominant modes of variability, such as the leading patterns of sea-surface temperature anomalies. Non-linear methods like t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) reveal clustering behavior in complex predictor spaces, aiding in the identification of distinct weather regimes. Autoencoders, a type of neural network, can learn compact latent representations that are subsequently fed into forecasting models.

Scaling and normalization standardize the range of numeric features, a prerequisite for many

machine-learning algorithms that assume comparable magnitudes across inputs. Standardization (z-score scaling) subtracts the mean and divides by the standard deviation, yielding a distribution with zero mean and unit variance. Min-max scaling rescales values to a fixed interval, typically [0, 1]. Logarithmic transformation is useful for variables with skewed distributions, such as precipitation, where the log of (value + 1) stabilizes variance. The scikit-learn library provides StandardScaler, MinMaxScaler, and PowerTransformer classes to automate these processes.

Encoding categorical variables is necessary when predictors include non-numeric attributes, such as weather station identifiers or cloud type descriptors. One-hot encoding creates binary columns for each category, preserving the lack of ordinal relationship. Label encoding assigns integer codes, which can be appropriate when categories possess a natural ordering (e.G., Beaufort wind force scale). For high-cardinality features like station IDs, target encoding—replacing each category with the mean of the target variable—can reduce dimensionality while retaining predictive power. Care must be taken to avoid data leakage by computing encodings only on the training split.

Target variable definition shapes the entire modeling effort. In short-range forecasting, the target may be a scalar field (e.G., Temperature at 2 m) at a future lead time. For probabilistic forecasts, the target could be the parameters of a distribution (mean and variance) or a set of quantiles. When training classification models for severe weather alerts, the target is binary (event vs. Non-event), often derived from thresholds applied to continuous variables such as wind speed $> 33\,\mathrm{m\,s^{-1}}$. Clearly defining the target ensures that evaluation metrics align with the intended application.

Data splitting partitions the dataset into training, validation, and test subsets. Random splitting is acceptable for independent observations, but weather data exhibit strong temporal autocorrelation, making random splits prone to leakage. Instead, a chronological split—using the earliest years for training, a subsequent period for validation, and the latest years for testing—preserves temporal integrity. For cross-validation, the TimeSeriesSplit iterator in scikit-learn creates successive folds that respect ordering, allowing robust hyperparameter tuning without contaminating future information.

Stratified sampling ensures that rare events, such as tornadoes or flash floods, are adequately represented in each split. By stratifying on the target class or on a derived severity index, analysts can avoid the situation where the validation set contains no positive examples, which would render performance metrics meaningless. In practice, the StratifiedShuffleSplit class can be adapted for time-series data by first grouping observations into blocks (e.G., Months) and then stratifying across those blocks.

Data augmentation expands the training set by generating synthetic examples. For spatial fields, techniques include random rotations, flips, or Gaussian noise injection, which help convolutional neural networks learn invariance to orientation. Temporal augmentation may involve jittering the time axis or creating synthetic sequences through interpolation between existing samples. Generative adversarial networks (GANs) have been employed to produce realistic radar reflectivity images, augmenting scarce extreme-event datasets. Augmentation must preserve physical realism; otherwise, the model may learn artifacts that degrade

real-world performance.

Handling imbalanced data is a recurring challenge when the frequency of extreme weather events is orders of magnitude lower than that of benign conditions. Resampling methods such as oversampling the minority class (SMOTE) or undersampling the majority class can rebalance the dataset. Cost-sensitive learning, where misclassifying a rare event incurs a higher penalty, can be implemented through class weights in loss functions. Evaluation metrics should reflect the imbalance, favoring skill scores like the Heidke Skill Score (HSS) or the Brier Skill Score (BSS) over raw accuracy.

Exploratory data analysis (EDA) provides the first visual insight into data characteristics. Time-series plots of temperature, humidity, and wind speed reveal diurnal cycles and seasonal trends. Histograms illustrate distribution shapes, while box plots expose outliers and interquartile ranges. Correlation matrices, visualized as heatmaps, identify linear relationships among variables, guiding feature selection. In weather domains, spatial visualizations—such as contour maps of pressure fields—are indispensable for recognizing synoptic patterns.

Time-series visualization often employs line plots with dual axes to display related variables (e.G., Temperature and dew point) simultaneously. Adding confidence bands, derived from moving-average standard deviations, helps communicate uncertainty. Interactive libraries like Plotly enable zooming and hovering to inspect specific timestamps, which is valuable when diagnosing model errors that cluster around particular events.

Heatmaps and contour plots convey spatial variability across a grid. For example, a heatmap of forecasted precipitation intensity can be overlaid with contour lines indicating isohyets (equal-rainfall lines). Using the cartopy library, one can project these visualizations onto geographic coordinate systems, adding coastlines and political boundaries for context. Color maps should be chosen carefully; perceptually uniform palettes such as “viridis” avoid misleading intensity interpretations.

Wind vector plots combine arrows representing direction and magnitude, often rendered on a geographic map. They are essential for assessing model performance in capturing jet streams or low-level wind shear. Scaling the arrow length proportionally to wind speed while maintaining a consistent reference vector ensures interpretability across different regions.

Radar and satellite imagery are rich sources of information for convective weather. Visualizing reflectivity (dBZ) or cloud-top temperature requires specialized colormaps and often a logarithmic scale to capture the wide dynamic range. Overlaying model-derived precipitation fields on observed radar can highlight systematic biases, such as underestimation of intense cores.

Scatter plots and pair plots explore relationships between two or more variables. In a scatter plot of observed versus predicted temperature, a 45° line indicates perfect agreement; deviations from this line quantify bias and dispersion. Pair plots, generated with Seaborn’s pairplot function, display all pairwise

relationships among a set of predictors, revealing multicollinearity that may necessitate dimensionality reduction.

Correlation matrix visualizations use color intensity to encode the magnitude of Pearson or Spearman coefficients. Annotating the matrix with numerical values assists in distinguishing strong from moderate correlations. When the matrix is large, hierarchical clustering can reorder variables to group similar features, facilitating the identification of redundant predictors.

Box plots and violin plots summarize distributional properties across categories, such as forecast errors for different seasons. Box plots convey median, quartiles, and outliers, while violin plots add a kernel density estimate, showing the full shape of the distribution. These plots are particularly helpful when comparing model performance across multiple lead times.

Histograms and cumulative distribution functions (CDFs) illustrate the frequency of error magnitudes. Plotting the CDF of absolute error allows the analyst to read off the proportion of forecasts within a given tolerance (e.G., 80 % Of temperature predictions within 2 °C).

Residual plots assess model fit by displaying the difference between observed and predicted values against predicted values or against a predictor. Randomly scattered residuals indicate that the model captures the systematic component, whereas patterns (e.G., Funnel shapes) suggest heteroscedasticity that may require variance-stabilizing transformations.

Skill score visualizations compare model performance against a reference, such as climatology or persistence. Plotting the Heidke Skill Score across lead times reveals the degradation of forecast skill as the horizon extends. Bar charts can compare multiple models side-by-side, while line charts show skill trajectories.

ROC curves and AUC are standard for binary classification tasks, such as predicting severe thunderstorm warnings. The Receiver Operating Characteristic curve plots the true-positive rate versus the false-positive rate for varying classification thresholds. The Area Under the Curve (AUC) provides a single scalar measure of discrimination ability; values close to 1 indicate excellent performance, while values near 0.5 Reflect random guessing.

Calibration plots assess probabilistic forecast reliability. By binning forecast probabilities and comparing the observed frequency within each bin, a well-calibrated model will align closely with the diagonal. Deviations indicate over- or under-confidence, prompting recalibration methods such as isotonic regression.

Model interpretability visualizations help explain complex AI models. SHAP (SHapley Additive exPlanations) values provide a unified measure of feature contribution; a SHAP summary plot shows each feature’s impact across the dataset, colored by the feature’s value. Partial dependence plots (PDPs) illustrate how the predicted outcome changes as a single feature varies, holding others constant. For convolutional networks applied to radar images, saliency maps highlight which pixels most influence the prediction, offering insight

into learned storm signatures.

Ensemble visualizations combine predictions from multiple models. Plotting the mean and spread of ensemble members as shaded regions conveys uncertainty. Rank histograms assess ensemble reliability by counting how often observations fall into each rank interval; a flat histogram indicates a well-calibrated ensemble, whereas a U-shaped histogram suggests under-dispersion.

Operational dashboards integrate many of the above visual components into a single interface for forecasters. Real-time updates of model output, verification metrics, and alert status can be delivered through web frameworks such as Flask combined with Plotly Dash. Designing an intuitive layout requires balancing information density with clarity, ensuring that critical alerts are prominent while detailed diagnostics remain accessible.

Computational challenges arise from the sheer volume of spatiotemporal data. Global reanalysis datasets can exceed several terabytes, demanding efficient I/O strategies. The xarray library, built on top of netCDF4, enables lazy loading and chunked processing, allowing analysts to work with subsets without loading entire files into memory. Parallel processing with Dask scales these operations across multiple cores or cluster nodes, reducing preprocessing time dramatically.

Non-stationarity is a fundamental issue in climate-aware weather prediction. Statistical relationships that hold in one decade may shift due to climate change, rendering historical training data less representative. Techniques such as detrending, adding climate indices (e.G., ENSO phase) as predictors, or employing transfer learning can mitigate non-stationarity. Visual monitoring of model bias over time, using rolling windows, helps detect emerging drifts.

Scale mismatches between predictors and targets can impair model learning. For instance, satellite radiance values are measured in Kelvin, while precipitation is expressed in millimeters. Converting all variables to a common physical basis—such as using potential temperature instead of absolute temperature—reduces the burden on the learning algorithm. Visual checks of variable ranges after scaling help verify that no feature dominates the loss function due to magnitude alone.

High dimensionality in gridded datasets leads to the “curse of dimensionality,” where the number of parameters grows exponentially with grid resolution. Dimensionality reduction techniques, mentioned earlier, become essential. Convolutional neural networks (CNNs) exploit local spatial coherence, dramatically reducing parameter count compared to fully connected layers. Visualizing learned filters in early CNN layers can reveal that the network is extracting physically meaningful patterns such as edges or gradients.

Data provenance and reproducibility are critical for scientific rigor. Recording the versions of data sources, preprocessing scripts, and library dependencies ensures that results can be replicated. Tools like git for version control and conda environment files for package management facilitate this practice. Generating a data processing pipeline diagram—often a directed acyclic graph (DAG)—helps communicate the sequence

of transformations and dependencies to collaborators.

Visualization of uncertainty goes beyond simple point estimates. Ensembles, Bayesian posterior samples, or Monte Carlo dropout predictions provide a distribution of possible outcomes. Visual representations include fan charts (shaded intervals), violin plots of forecast ensembles, and probability density maps. Communicating uncertainty effectively is vital for decision-makers who must weigh risk, for example, in flood-forecasting where a slight increase in predicted river stage can trigger evacuation orders.

Interactive exploration empowers analysts to probe data dynamically. Tools like Jupyter notebooks combined with ipywidgets or Plotly's interactive figures enable on-the-fly filtering by date, region, or threshold. Selecting a region on a map and instantly seeing corresponding time-series plots of temperature, humidity, and model error fosters a deeper understanding of localized model performance.

Geospatial considerations influence both preprocessing and visualization. Projection choice (e.G., Plate Carrée vs. Lambert Conformal) affects area representation, especially near the poles. When merging datasets on different grids, regridding methods such as bilinear interpolation, conservative remapping, or the ESMF library's tools help preserve integral quantities like total precipitation. Visual checks of regridded fields, using side-by-side difference maps, are essential to confirm that the transformation has not introduced spurious artifacts.

Data security and privacy may be relevant when incorporating proprietary observations, such as high-resolution wind lidar data from private operators. Preprocessing pipelines must enforce access controls, possibly anonymizing station identifiers or aggregating data to coarser resolutions before sharing. Visualization of such restricted data should omit sensitive metadata, and any public dashboards must comply with licensing agreements.

Case study: Preprocessing for a convective-storm AI model illustrates the integration of many concepts. The workflow begins with ingesting Level-II radar data (reflectivity, velocity) and surface observations (temperature, dew point) for a target region. QC filters remove non-meteorological echoes using texture-based classifiers. Missing radar sweeps are filled via temporal interpolation, while spatial gaps are addressed with IDW. Features such as maximum reflectivity, shear magnitude, and lifted index are engineered from the raw fields. A PCA step reduces the radar cube from 30 vertical levels to the first five principal components, preserving > 90 % variance. The resulting feature set is standardized, then split chronologically: 2015-2018 For training, 2019 for validation, and 2020 for testing.

Visualization of the preprocessing outcomes includes a heatmap of missing-data percentages across variables, a contour map of interpolated reflectivity, and a scatter plot of observed versus engineered CAPE values. Residual analysis after a baseline linear regression reveals heteroscedastic error, prompting a log transformation of precipitation. Finally, a SHAP summary plot of the trained gradient-boosted tree model highlights that shear and CAPE dominate predictions, guiding further refinement of feature engineering.

Case study: Visual verification of a seasonal-forecast AI system demonstrates the use of ensemble visualizations. A suite of 30 neural-network members predicts monthly mean temperature anomalies over Europe. The ensemble mean is plotted as a contour map, while the ensemble spread is shown as shading representing one standard deviation. A rank histogram assesses dispersion; a slight U-shape suggests under-dispersion, leading to post-processing via Bayesian Model Averaging. Skill scores, such as the Brier Skill Score for binary anomaly thresholds, are displayed in a bar chart comparing the AI system to a traditional statistical model.

These examples underscore how meticulous preprocessing and thoughtful visualization together enable robust AI models in weather prediction. By mastering the terminology and techniques outlined above, students can navigate the complexities of atmospheric data, transform raw observations into actionable information, and communicate results effectively to both scientific and operational audiences.