**Zelestra x AWS ML Ascend Challenge**

**Performance Optimization of Solar Panels**

**Team Details:**

**Team Name: Cloud-Catalysts**

**Team members:**

1. **Kumari Vaishnavi**
2. **Anubhav Kumar**
3. **Ashish Rawat**
4. **Vansh Khattar**

**Approach**

The pipeline employs a multi-step ensemble strategy, where several base models are learned independently, and their predictions are stacked together as inputs for a meta-model. The process includes:

1. **Data Loading and Preprocessing**: We load the training and test data using Pandas and remove any identifier columns to prevent them from interfering with model training. Numeric columns (such as "humidity", "wind\_speed", and "pressure") are converted into numeric types (with coercion of non-numeric values to NaN) and then imputed with the median from the training set. Categorical features (like "installation\_type" and "error\_code") are converted to strings and one-hot encoded to transform them into binary features. Finally, the train and test sets are aligned so that they have identical columns, with any missing features filled with zeros.
2. **Feature Scaling:** StandardScaler is used to standardize the feature values so that the resulting features have zero mean and unit variance. Scaling plays an important role, particularly for the linear meta-model (Ridge), which combines the outputs of the base models.
3. **Train–Validation Split:** The dataset is partitioned using train\_test\_split into a training set (90%) and a validation set (10%) which helps in evaluating the model performance and ensuring that the ensemble generalizes well to unseen data.
4. **Training Base Models:** The code trains several base models with different strengths:

* **XGBoost:** A powerful gradient boosting technique well-suited for tabular data.
* **LightGBM:** Another high-performance gradient boosting method that uses a histogram-based approach and early stopping to prevent overfitting.
* **CatBoost:** Known for its capability in handling categorical features, optimized here with loss function RMSE.
* **RandomForestRegressor:** An ensemble tree algorithm that reduces variance through bagging.
* **AdaBoostRegressor:** Uses boosting to combine weak learners (e.g., shallow trees) to enhance performance.

1. **Stacking Predictions:** Each base model generates predictions on the validation and test sets. These predictions are then stacked (using np.column\_stack) to form a new feature matrix. This “meta” dataset encapsulates diverse forecasts from different modeling philosophies.
2. **Training the Meta-Model:** A Ridge regression model is used as the meta-learner. Its regularization parameter (alpha) is tuned using GridSearchCV over a set of candidate values. The meta-model learns to optimally blend the base models' predictions. Final evaluation is performed using RMSE (converted into a custom score).
3. **Final Predictions and Submission:** The meta-model is trained on the validation predictions, and its learned weights are applied to the test set predictions, culminating in final outputs that are formatted and saved as a submission file.

**Feature Engineering Details**

* **Numeric Processing:** Numeric columns are explicitly cast to numeric types with error handling (errors="coerce"), ensuring that any inappropriate string values are set to NaN. Missing values are then imputed using the median, which is a robust measure against outliers.
* **Categorical Transformation:** Categorical variables are converted to strings and one-hot encoded via pd.get\_dummies(). This creates binary columns for each unique category, ensuring that the models can handle nominal data appropriately.
* **Final Alignment:** After one-hot encoding, train and test datasets are aligned so that they include the same features, managing any features that may appear in one set and not the other.
* **Scaling:** Feature scaling with StandardScaler standardizes the data. This is critical for any models sensitive to the scale of input (in our case, the Ridge meta-model).

**Tools Used**

* **Pandas and NumPy: T**hese libraries provide essential data manipulation and numerical computing capabilities, allowing efficient data reading, cleaning, and transformation.
* **Scikit‑learn:** Tools from scikit‑learn used in this pipeline include:
  + **StandardScaler:** For feature standardization.
  + **train\_test\_split:** For partitioning the dataset.
  + **GridSearchCV:** For exhaustive hyperparameter tuning using cross-validation.
  + **Ridge:** The linear regression model with L2 regularization serves as the meta-model.
  + **mean\_squared\_error:** For evaluating model performance via RMSE.
* **XGBoost, LightGBM, and CatBoost:** These are state‑of‑the‑art gradient boosting frameworks that provide high predictive power and handle large, structured datasets efficiently.
* **RandomForestRegressor and AdaBoostRegressor:** Ensemble methods based on decision trees that help to reduce variance (RandomForest) and bias (AdaBoost).

**Final Outcome**

In our experiments, after meticulously preprocessing the data, engineering features, training multiple base models, and optimally blending their predictions with a tuned Ridge meta-model, we secured a score of **89.87023.**