### Feature Extractor

The code is a comprehensive EEG feature extraction pipeline that takes preprocessed EEG signals stored in CSV files and computes a wide range of features using statistical, spectral, nonlinear, and connectivity-based methods. Below is a detailed explanation of every scientific and computational step involved:

**1. Data Ingestion and Preprocessing**

* **Loading Data:**  
  The extractor reads a CSV file where one column (by default named "Timestamp") is used for indexing while all other columns represent EEG channels.
  + **Pandas** is used to load the CSV.
  + The EEG data is transposed (channels × time) so that it fits the expected input shape for MNE.
* **Conversion to MNE Format:**  
  The transposed data is used to create an MNE RawArray.
  + **MNE Library:** Provides a standardized way to handle EEG data along with metadata (e.g., sampling frequency, channel names, channel types).

**2. Feature Registration and Modular Extraction**

* **Dynamic Registration:**  
  The extractor class maintains a dictionary mapping feature names to their corresponding functions. This design allows one to register any new feature extraction function easily.
* **Iterative Processing:**  
  For every channel, the extractor retrieves the 1D signal and applies all registered feature functions, catching any exceptions to avoid halting the process if one feature fails.

**3. Feature Extraction Methods**

The functions fall into several categories:

**A. Basic Statistical Features**

* **Mean, Median, Variance, Standard Deviation:**
  + These are standard statistical descriptors that summarize the central tendency and dispersion of the EEG signal.
* **Skewness and Kurtosis:**
  + These measures capture the asymmetry (skewness) and the "tailedness" (kurtosis) of the signal’s distribution.
* **Zero-Crossing Rate (ZCR):**
  + Counts how often the signal changes sign, which can be related to the frequency content.
* **Energy and Peak-to-Peak Amplitude:**
  + **Energy:** Sum of squared signal values (normalized by length), indicative of overall signal power.
  + **Peak-to-Peak:** Difference between maximum and minimum values, capturing amplitude dynamics.

**B. Frequency-Domain Features**

* **Power Spectral Density (PSD) via Welch’s Method:**
  + **Welch’s Method (via psd\_array\_welch):** This technique segments the signal, computes a periodogram for each segment, and averages them to estimate the PSD.
  + **Band-specific Power:**
    - **Delta (1–4 Hz), Theta (4–8 Hz), Alpha (8–12 Hz), Beta (12–30 Hz):** For each band, the code calculates the mean PSD.
  + **Total Power:**
    - Sum of the PSD in the 1–30 Hz range.
  + **Relative and Ratio Metrics:**
    - **Relative Alpha:** Alpha power divided by total power.
    - **Theta/Alpha and Beta/Alpha Ratios:** These ratios can indicate shifts in brain state.
* **Spectral Edge Frequency:**
  + The code computes the frequency below which a given percentage (e.g., 95%) of the total power is contained. This is determined by the cumulative sum of the PSD.

**C. Nonlinear Features**

* **Entropy Measures:**
  + **Sample Entropy:** (A placeholder implementation) Uses the log of the standard deviation to roughly quantify irregularity.
  + **Permutation Entropy:**
    - This method quantifies complexity by analyzing the ordinal patterns (i.e., the order of values) in the signal.
* **Fractal Dimension:**
  + **Higuchi Fractal Dimension:**
    - Estimates the fractal dimension by calculating the length of the curve at different scales and fitting a log-log line. This provides insight into the signal’s complexity.
* **Lempel-Ziv Complexity:**
  + Measures the complexity of a binary sequence (converted from the signal based on a median threshold), reflecting how “compressible” or unpredictable the signal is.

**D. Hjorth Parameters (Time-Domain Complexity)**

* **Activity, Mobility, and Complexity:**
  + **Activity:** Simply the variance (signal power).
  + **Mobility:** Ratio of the standard deviation of the signal derivative to that of the signal.
  + **Complexity:** Compares the mobility of the derivative with the original mobility, reflecting how the frequency content changes over time.
* **Spectral Entropy:**
  + Entropy calculated from the normalized PSD, indicating the disorder or randomness of the spectral distribution.

**E. Time-Frequency and Connectivity Features (Especially for Visual Imagery Studies)**

* **Phase Locking Value (PLV):**
  + **Hilbert Transform:** After band-pass filtering in the alpha range, the Hilbert transform provides the analytic signal to extract the instantaneous phase.
  + The signal is split into windows, and PLV is computed by comparing phase differences between consecutive windows.
* **Alpha Peak Frequency & Individual Alpha Power:**
  + **Alpha Peak Frequency:** Identifies the frequency within the 8–13 Hz band that has the highest power.
  + **Individual Alpha Power:** Computes the mean power in a narrow band around the detected alpha peak.
* **Wavelet Complexity:**
  + **Wavelet Decomposition (using pywt.wavedec):** Decomposes the signal into various frequency components.
  + Entropy is then calculated for each set of coefficients, and the mean entropy serves as a measure of complexity.
* **Functional Connectivity Measures:**
  + **Coherence:**
    - Splits the signal into overlapping segments and computes the coherence between adjacent segments within a specified frequency band.
  + **Dynamic Connectivity Variance:**
    - Measures how the coherence (and thus connectivity patterns) vary over time by calculating the variance of coherence values across segments.
* **Gamma Band Features:**
  + **Low Gamma (30–45 Hz) and High Gamma (55–80 Hz):**
    - These features capture power in gamma bands, which are important in attention and visual processing.
* **Detrended Fluctuation Analysis (DFA):**
  + DFA quantifies long-range temporal correlations by detrending the cumulative sum of the signal at multiple scales and fitting a log-log plot to obtain a scaling exponent (alpha).
* **Energy Ratio:**
  + The ratio of energy in the second half of the signal to that in the first half, potentially revealing changes in brain state or task engagement over time.
* **EEG Microstate Variability:**
  + Although true microstate analysis involves spatial information, here a proxy is computed by filtering in the alpha band, segmenting the signal into “states” based on amplitude relative to the median, and calculating the coefficient of variation of the state durations.

**4. Processing Pipeline and Execution**

* **File and Directory Handling:**
  + The script processes single CSV files and also supports recursively processing an entire directory structure.
  + **Progress Tracking:**
    - Uses tqdm for a visual progress bar.
  + **Output Handling:**
    - Computed features are saved as CSV files with a naming convention that mirrors the input files.
* **Command-Line Interface:**
  + An argument parser (via argparse) is provided, allowing the user to specify input/output directories and the sampling frequency.

**5. Summary of Scientific Methods and Their Purpose**

* **Statistical and Descriptive Analysis:**
  + Quantifies basic properties (mean, variance, skewness) that describe the signal’s distribution.
* **Spectral Analysis (Welch’s Method):**
  + Identifies the distribution of power across frequency bands, which is critical for understanding EEG rhythms related to different cognitive states (e.g., alpha with visual imagery).
* **Nonlinear Dynamics (Entropy, Fractal Dimensions):**
  + Captures the complexity and irregularity of EEG signals, which may reflect underlying neural processes.
* **Time-Frequency Analysis (Wavelets, Hilbert Transform):**
  + Decomposes the signal into time-localized frequency components and assesses synchronization (PLV) across windows.
* **Connectivity and Complexity Measures:**
  + Coherence and dynamic connectivity metrics offer insights into how different segments of the EEG interact or evolve over time.
* **DFA and Energy Ratios:**
  + Provide measures for long-range correlations and temporal evolution, important for assessing changes during tasks like visual imagery.

In summary, the code combines a suite of well-established methods from signal processing and nonlinear dynamics to extract a rich set of features from EEG data. Each feature is designed to capture a different aspect of the underlying neural activity—from simple statistical moments to complex time-frequency dynamics and connectivity patterns—thereby enabling detailed analysis of cognitive tasks such as visual imagery.

### Feature Merger

This code implements a comprehensive pipeline for EEG feature analysis that goes beyond simple extraction. It takes pre‐extracted EEG features (from earlier processing), merges them according to different experimental zones, and then computes feature importance rankings using several statistical and machine learning methods. Below is a detailed breakdown of every scientific and computational step and the methods used:

**1. Overview of the Pipeline**

The pipeline is organized into several key stages:

* **Feature Merging:**  
  The code first gathers individual feature CSV files (each corresponding to different EEG recordings and experimental conditions) and merges them into consolidated datasets grouped by “zone” (e.g., *imagery\_task*, *instruction*, *baseline\_post*).
  + **Metadata Extraction:** It extracts metadata such as the label (the EEG condition or task) and zone from the filenames.
  + **Session Information:** It also tags each record with session data (inferred from directory names) and ensures that the channel identifiers are standardized.
* **Feature Importance Calculation:**  
  After merging, the pipeline computes feature importance using three different methods:
  + **ANOVA F-Test (Univariate Analysis):**
    - Uses SelectKBest with the f\_classif function from scikit-learn to assess the discriminatory power of each feature based on variance between groups.
    - **Normalization:** The raw F-values are normalized to a 0–100 scale.
  + **Mutual Information:**
    - Uses mutual\_info\_classif to measure how much information about the target label is provided by each feature.
    - **Preprocessing:** Features are scaled using StandardScaler because mutual information is sensitive to feature scaling.
  + **Random Forest Feature Importance:**
    - Trains a RandomForestClassifier and leverages its built-in feature\_importances\_ attribute to rank features based on how much they reduce impurity in the trees.
    - **Ensemble Weighting:** The normalized scores are again scaled to 0–100.
* **Ensemble Importance Ranking:**  
  The three methods’ scores are combined using a weighted average (with weights 0.3 for ANOVA, 0.3 for mutual information, and 0.4 for random forest) to produce an ensemble feature importance ranking. This approach aims to provide a more robust and comprehensive evaluation by leveraging the strengths of each method.
* **Visualization and Reporting:**  
  Visualizations (using Matplotlib and Seaborn) and summary CSV files are generated for each zone. These provide bar plots of the top features and summary statistics to help interpret which EEG features best differentiate between experimental conditions.

**2. Detailed Components and Scientific Methods**

**A. Feature Merger (EEGFeatureMerger Class)**

* **Scanning and Metadata Extraction:**  
  The \_extract\_metadata\_from\_filename method parses filenames to extract the label and zone. For example, filenames formatted like features\_p\_af\_b\_imagery\_task\_1740854836.csv are split and processed to deduce that the zone is *imagery\_task*.
* **Merging Process:**
  + The merge\_features\_by\_zone method recursively scans the specified directory.
  + It reads each CSV file into a DataFrame, appending additional columns for label and session (extracted from the directory structure).
  + All files for a particular zone are concatenated into one merged DataFrame and saved to disk.
* **Summary Statistics:**
  + The create\_feature\_summary method computes basic statistics (minimum, maximum, mean, standard deviation, and percentage of missing values) for each feature across the merged dataset.
  + This summary helps in understanding the range and distribution of the features across different EEG recordings.

**B. Feature Importance Calculator (EEGFeatureImportance Class)**

* **Data Loading:**
  + The load\_merged\_data method reads the merged feature CSV file for a given zone. It expects the file to include columns like label, channel, and session alongside the EEG feature measurements.
* **ANOVA Importance Calculation:**
  + **Method:** Uses scikit-learn’s SelectKBest with the f\_classif function to compute the F-statistic for each feature.
  + **Scientific Basis:** The F-test determines whether the means between two or more groups are statistically different, providing insight into each feature’s discriminative power.
  + **Normalization:** The F-values are normalized to a 0–100 scale to facilitate comparison across methods.
* **Mutual Information Importance Calculation:**
  + **Method:** Uses mutual\_info\_classif to compute the mutual information between each feature and the target label.
  + **Scientific Basis:** Mutual information quantifies the reduction in uncertainty about the label given the feature, which is useful for capturing non-linear dependencies.
  + **Preprocessing:** Features are scaled to improve the accuracy of mutual information estimation.
* **Random Forest Importance Calculation:**
  + **Method:** Trains a Random Forest classifier and extracts the feature\_importances\_ attribute.
  + **Scientific Basis:** Random forests measure the decrease in impurity (or error) when a feature is used to split the data, thereby providing an importance metric that is robust to overfitting.
  + **Normalization:** Again, the importances are normalized to a 0–100 scale.
* **Ensemble Importance:**
  + **Method:** Combines the normalized scores from the three methods by calculating a weighted average.
  + **Weights:** The ensemble gives slightly higher weight to the random forest importance (0.4) compared to the other two (0.3 each). This reflects the idea that model-based importance might capture complex interactions better.
  + **Outcome:** The resulting ensemble score provides a robust ranking of features.
* **Visualization:**
  + **Bar Plots:** Using Seaborn’s barplot, the top N features (default is 20) are plotted based on their importance scores.
  + **Purpose:** These plots help researchers quickly identify which EEG features are most relevant for differentiating between experimental conditions.

**C. Integration in the Main Function**

* **Command-Line Interface:**  
  The script uses argparse to allow users to set directories, sampling frequencies, and to choose whether to skip certain stages (extraction, merging, or only run importance analysis).
* **Workflow Control:**  
  The main function orchestrates the following:
  1. Optionally run the feature extraction (if not skipped).
  2. Merge the extracted feature files using the merger class.
  3. Compute feature importance rankings using the importance calculator.
  4. Save and visualize the results, and print guidance on how these rankings might be used for further classification tasks (e.g., dimensionality reduction before training a Random Forest).

**3. Scientific Rationale**

* **Multifaceted Importance Analysis:**  
  By using three different methods (ANOVA, mutual information, and random forest), the pipeline combines linear statistical testing, information theory, and ensemble learning. This variety ensures that both linear and non-linear relationships between features and labels are considered.
* **Ensemble Method:**  
  The weighted average approach helps mitigate the bias of any single method, aiming to produce a more stable and reliable importance ranking.
* **Data Quality and Summarization:**  
  Merging feature files and summarizing statistics across zones provide context for how feature distributions vary across sessions and experimental conditions, which is critical for understanding EEG data variability.
* **Practical Usage:**  
  The pipeline ends by suggesting how these importance rankings can be applied (e.g., selecting the top features for classification tasks), and even provides example code for using the reduced feature set in a scikit-learn workflow.

In summary, this code is a sophisticated tool for analyzing EEG features: it first consolidates pre-extracted features from various recordings, then evaluates which features are most relevant for distinguishing between cognitive or behavioral states. By leveraging statistical tests, mutual information, and ensemble learning methods, it provides a robust, multi-dimensional ranking that can guide further analysis or classification model development.

### Label Differentiator

This code is a full-blown analysis and visualization pipeline that takes the merged EEG features (produced by the previous processing stages) and uses them for feature selection, classification, separability analysis, and data visualization. Below is a step‐by‐step explanation of the scientific methods and computational steps involved:

**1. Configuration and Data Loading**

* **File and Parameter Setup:**
  + The code begins by specifying the file path for the merged features (e.g., from an EEG imagery task).
  + It defines parameters such as the number of top labels to consider (e.g., the most diverse two labels), the number of features to select, and the channel handling approach (options include “pooled”, “separate”, or “features”).
* **Loading the Data:**
  + The merged CSV is read into a pandas DataFrame.
  + The script ensures that metadata (like the channel) is present and extracts the names of feature columns (i.e., all columns except metadata such as label, channel, and session).
  + It then prints a summary of the dataset, including unique labels, channels, and counts per group.

**2. Handling Channel Approaches**

* **Channel Handling Strategy:**
  + **Pooled Approach:** Treats each channel’s reading as an independent sample. This is often useful when sample sizes are small, as it “pools” the information.
  + **Separate Approach:** Would analyze each channel independently. (In this code, it is set up but not expanded further.)
  + **Features Approach:** Combines channels as additional features per sample; this requires that each recording (session) is represented by multiple rows, one per channel.
  + Based on the chosen option (here, “pooled”), the code assigns feature matrix X and labels y.

**3. Preprocessing**

* **Standardization:**
  + The feature matrix is standardized using a StandardScaler from scikit-learn. Standardization is crucial in EEG data analysis because it brings all features to a common scale, which improves the performance of many machine learning algorithms.
  + The scaled data is then converted back into a DataFrame for easier column-based operations.

**4. Enhanced Feature Selection**

* **Loading Feature Importance Rankings:**
  + The code attempts to load an external CSV file that contains feature importance rankings (likely generated from an ensemble of methods in the previous stage).
  + These rankings are sorted by an “ensemble score” which represents a weighted average of scores from ANOVA F-test, mutual information, and Random Forest importance.
* **Selecting Top Features:**
  + The top *N* features (as specified by the parameter) are selected based on the ensemble score.
  + **Supplemental Feature Selection:**
    - If some of the top features from the external ranking are missing from the current dataset, the code uses an ANOVA F-test (SelectKBest with f\_classif) to select additional features.
  + The final selected features are then extracted from the standardized data.
* **Fallback Strategy:**
  + If the feature importance file cannot be loaded, the code falls back to standard ANOVA-based feature selection.

**5. Finding the Most Separable Label Pairs**

* **Label Combinations:**
  + The script uses Python’s itertools.combinations to iterate over all possible combinations of labels (for example, pairs if TOP\_N\_LABELS is 2).
  + For each combination, a subset of the data containing only those labels is created.
* **Cross-Validation for Separation:**
  + **Leave-One-Out Cross-Validation (LOO CV):**
    - Given the small dataset size, LOO CV is employed to evaluate how well different classifiers can separate the labels.
  + **Classification Models:**
    - Two models are compared: a Random Forest and an SVM (with an RBF kernel).
    - The mean accuracy across all LOO CV folds is computed for each model.
  + **Separation Score:**
    - For each label combination, the best-performing model’s score is recorded. The combination with the highest score is identified as the “most separable” pair (or trio) of labels.

**6. Visualizations**

**A. PCA Visualization**

* **Dimensionality Reduction:**
  + Principal Component Analysis (PCA) is applied to the standardized features to reduce them to two dimensions for visualization.
  + The explained variance for each principal component is calculated and displayed on the axis labels.
* **Scatter Plot with Channel Markers:**
  + The reduced data is plotted with different colors for each label.
  + If channel information is available, different markers are used for different channels to indicate the distribution of readings across channels.
* **Confidence Ellipses:**
  + For each label, the code calculates and draws an ellipse representing the covariance of the data points, which helps visualize the spread and overlap of different groups.
  + Centroids for each label group are computed and marked with distinct symbols.
* **Legends and Annotations:**
  + Two legends are created—one for labels (colors) and one for channels (markers).
  + Text annotations on the plot display details about the best label combination and their separation score.

**B. Feature Importance Visualization**

* **ANOVA F-Test Ranking:**
  + For the subset of data corresponding to the most separable labels, an ANOVA F-test is applied.
  + F-scores and corresponding p-values are calculated for all features.
  + A bar plot is then created:
    - The F-scores (after a logarithmic transformation for better visualization) are plotted horizontally.
    - A red vertical line indicates a significance threshold (p-value < 0.05).
    - Each bar is annotated with the raw F-score and p-value.

**C. Channel Distribution Heatmap (Bonus)**

* **Heatmap Visualization:**
  + If channel information is available, the code computes a cross-tabulation of labels and channels.
  + The percentages of samples per channel for each label are calculated.
  + A heatmap is generated using Seaborn to show how samples are distributed across channels and labels.

**7. Saving and Output**

* **File Saving:**
  + The PCA visualization and feature importance plots are saved as high-resolution PNG files.
  + The channel distribution heatmap is also saved if applicable.
* **Summary Output:**
  + The code prints summaries at various stages: sample counts, selected features, model accuracies for each label combination, and the most separable label pair/trio along with their separation score and best-performing model.

**Scientific Rationale Behind Methods**

* **Standardization:**  
  Ensures that all features contribute equally to the analysis, which is critical when different features are on different scales.
* **Enhanced Feature Selection:**
  + **Ensemble Scoring:** Combines multiple feature ranking methods to robustly identify the most discriminative features.
  + **ANOVA F-test:** Identifies features with statistically significant differences between groups.
  + **Mutual Information & Random Forest:** Capture both linear and non-linear relationships between features and labels.
* **Leave-One-Out CV:**  
  Particularly useful for small datasets, this method provides an almost unbiased estimate of model performance by testing each sample individually.
* **Dimensionality Reduction (PCA):**  
  Helps visualize high-dimensional data in a two-dimensional plot, revealing patterns, clusters, or overlaps between groups.
* **Confidence Ellipses:**  
  These visually represent the spread (covariance) of each group, helping to interpret how distinct or overlapping the label clusters are.
* **Heatmap of Channel Distribution:**  
  Provides insights into potential biases or patterns in the data collection process across different channels.

In summary, this code ties together multiple advanced analytical steps—from preprocessing and feature selection to model evaluation and visualization—to interpret and assess EEG data. It not only selects the most discriminative features using ensemble and statistical methods but also evaluates which labels are most separable and visualizes both the data structure and feature importance. This integrated approach supports both the development of robust classification models and the extraction of insights about the underlying neural signals.