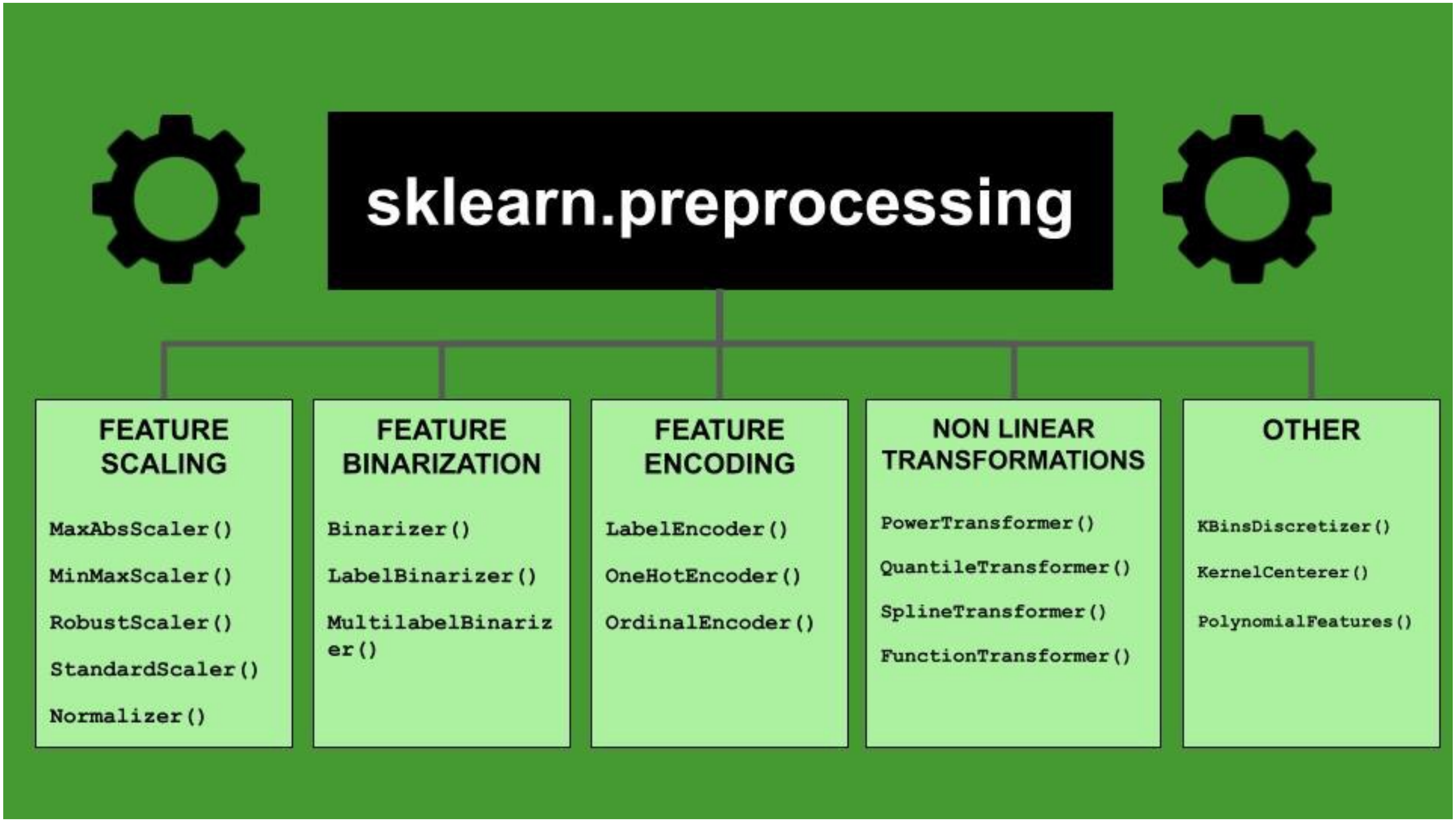
Scikit-Learn Library

- Provides several estimators: machine learning algorithms, models, and metrics.
- Built on top of NumPy, SciPy, and Matplotlib
- Open-source python library, tools for data mining and data analysis
- Supports classification, regression and clustering algorithms
    - Support Vector Machine (SVM), Random Forest, gradient boosting, k-means, etc.
- Seaborn for heatmap analytics

Scikit-Learn Capabilities



Towards Data Science

Using scikit-learn

List of features, libraries, methods available

[API Reference — scikit-learn 1.5.0 documentation]

Scikit-Learn Modules that We Use?

- Metrics: Accuracy, Area Under Curve (AUC)
- Principal Component Analysis
- Classifier and Regressor: K-Nearest Neighbor
- Classifier: Logistic Regression
- Classifier: Gradient Boosting (time permits)

Principal Component Analysis

- In multivariate analytics, we want to know which variables (or features) play more significant role over others?
- What is the issue?
  - Computationally intensive. Data represented by multiple features is a non-linear problem.
  - Interdependence of data is often hard to assess
  - Lesser the number of features → better would be
    - Data interpretation, and
    - explain underlying physical processes.
- How do we select significant features (instead of all features)?

Terminology

- **Rank Order**. In PCA, we list parameters by the variation in descending order. This process helps with selecting top performers.
- **Standardized Data**. Transform data into [0,1] interval. Note, if the distribution is skewed, standardization is not accurate.
- **Dimensionality Reduction**. Reduce number of features (from original list). This involves dropping no/low-variance features without compromising for prediction accuracy.
- **Anomaly** (detection). Unusual variations that are present in low-variance components
- **Noise.** Small and repetitive parts of the signal (data) that doesn't exhibit a pattern.
- **Decorrelation**. Transform highly correlated components (aka features or their combination) into uncorrelated components.
- **Label**. Text representing the categorical data

Terminology (Contd.)

- **Square Matrix.** Coefficients of a system of linear equations. Also represents a linear transformation.
- **Covariance Matrix**. A square matrix that captures the magnitude of the variance and its direction of a variable (feature) in multivariate analytics.
- **Vector.** Quantity that has both direction and magnitude (or length)
- **Linear Transformation**. When multiplied by a matrix, if length of the vector changes, it’s called linear transformation.
- **Non-linear Transformation.** When multiple by a matrix, the length of the vector and/or its direction change, then the transformation is called non-linear.
- **Eigenvector.** A vector whose direction remains unchanged when multiplied by a matrix.
- **Eigenvalue**. The magnitude (or length) of the eigenvector. Represents variance in the direction of largest spread of the data.

How Do We go about PCA?

- First and foremost, make sure the data is not skewed. Standardize the range of continuous initial variables (to eliminate technical bias)
- Compute the covariance matrix to identify correlations among features
- Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
- You can think of eigenvectors as those that do not change over a transformation.
- Create a feature vector to decide which principal components to keep
- Recast the data along the principal component’s axes

Algorithm for PCA

Step 1: Setup a DataFrame for the dataset (X,Y) with X and Y representing features and Target of the sample

Step 2: Standardize dataset by subtracting the mean and dividing by the std. deviation

Step 3: Calculate the covariance matrix of features (X)

Step 3: Use Covariance matrix to find eigenvalues and eigenvectors

Step 4: Pick the top contributor eigenvalues → principal components.

Step 5: Pick the set of eigenvalues together describe data to the max (>85% is a good estimate). For this rank order percent eigenvalues.

Step 6: Pick top contributing eigenvalues (Dimensionality reduction)

Find % variance (information) accounted for by each eigenvalue

%ev1 = ev1/(sum of eigenvalues) * 100

%ev2 = ev2/(sum of eigenvalues) * 100, so on.

Step 7: Merge the eigenvectors into a matrix and apply* it to the data. Principal components are now aligned with the axes of our features. Keep reduced # of features that explain the most variation in the data!

*Note: Applying eigenvectors and eigenvalues will scale and rotate data

Review and follow [diabetes-pca.ipynb]

[Principal Component Analysis (PCA) - easy and practical explanation (youtube.com)]