Table of Contents

1 Feature Engineering In Time Series Analysis

Feature Engineering can be defined as the process of selection and pre-processing meaning-full features and eliminating features from the time series that are irrelevant to the task.

More generally feature engineering includes the following tasks.

- **Exploratory Data Analysis, EDA** (data visualization, preliminary manual analysis (unsupervised), e.t.c.).
- **Feature Representation** (and Time Series Representation), (trend-seasonal decomposition, bag of words, spectrum domain, wavelet decomposition, acf, e.t.c.).
- **Feature Extraction** (trend-seasonal-irregualr part decomposition, PCA, SSA, Intrinsic mode decomposition, Auto-encoder based extraction, representation learning, ARIMA approximation).

- **Feature Transformation** (and Time Series Transformation), (Box-Cox transform, differencing, trend linearization or elimination, seasonality elimination, data normalization).
- **Feature Selection** (filtering methods, wrappped methods).
- **Data Anomaly Detection**(novelty detection, outliers missing values).
- **Data Denosing** (same as feature extraction but only for such process as white noise and e.t.c.).

Feature engineering facilitates data understanding, reduces the computational time and storage requirements, so that model learning becomes an easier process. As usual, it can allow increasing accuracy of father processing.

1.1 Exploratory Data Analysis (EDA)

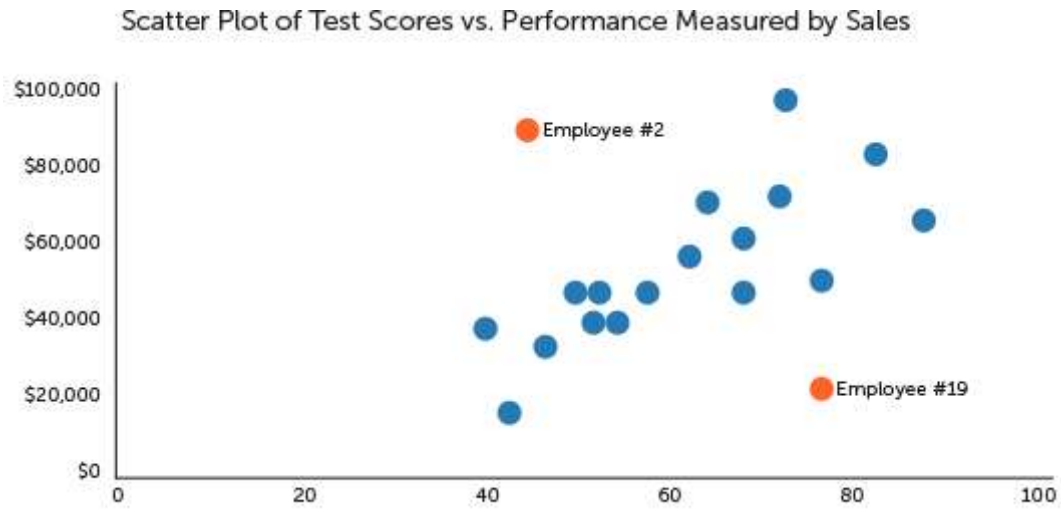
Exploratory Data Analysis (EDA)

EDA refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

EDA includes such parameters estimation as:

- Minimum, quartiles, Median, mean, Maximum values.
- values deviation, normal-distribution hypnosis checking, skewness, kurtosis.
- stationarity analysis (adf-test, ACF-PCF analysis).
- data dependences analysis for multivariate data (correlation, causality analysis).
- data visualization (box-plot, histogram-distribution, scatterplot).
- missing and outliers value searching.

Example of scatterplot analysis. A scatterplot can reveal data symmetry, clusters, correlation between variables, and extreme values or outliers. .

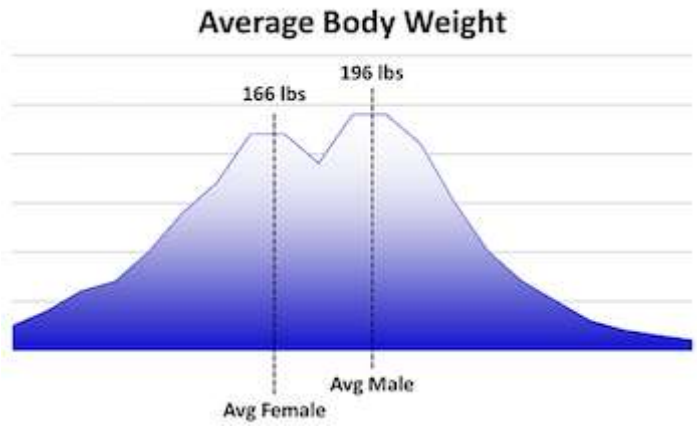


Example of histogram analysis.

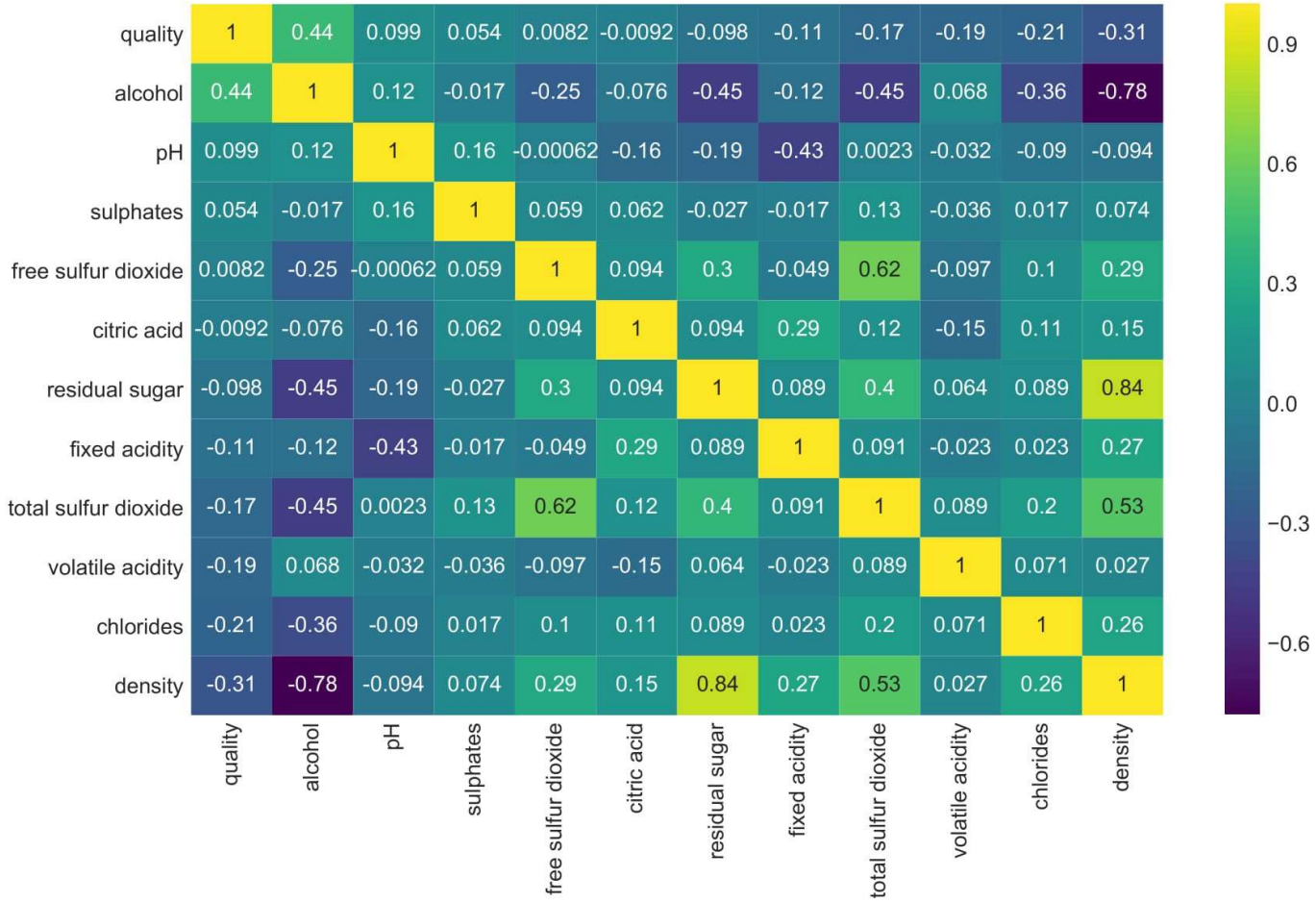
Histogram represents the underlying structure in the form of a frequency distribution; that is, how often a particular value occurs.

Histograms help us see data symmetry, peaks, outliers or data error through omission.

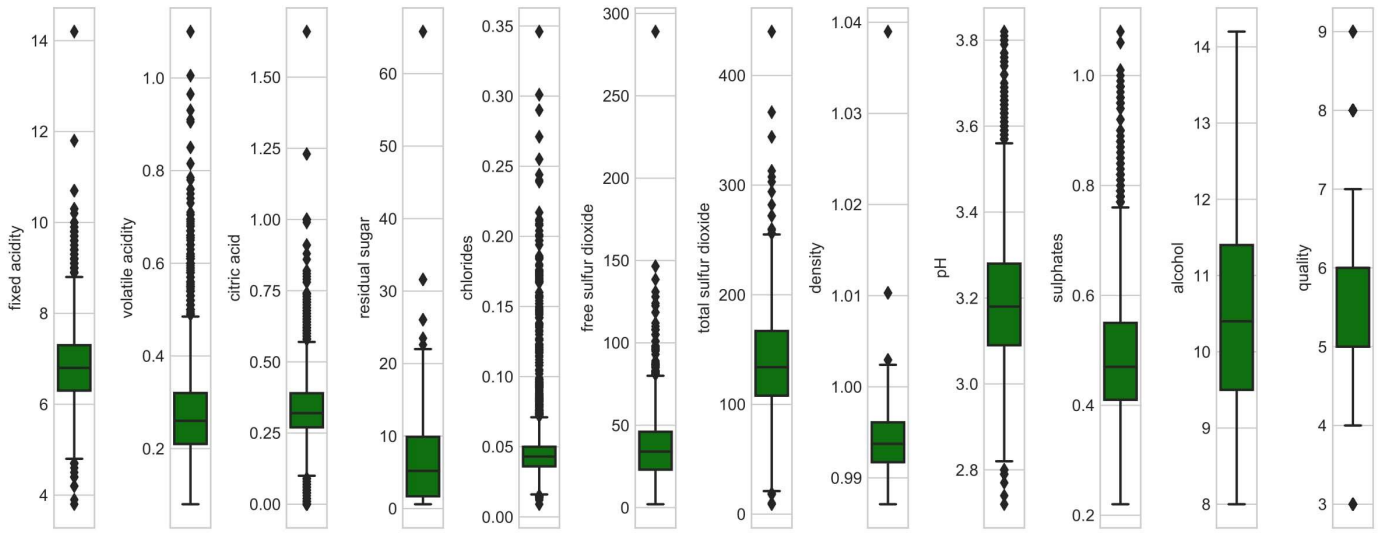
When multimodal histograms is presence (more than one peak), there's room to split the data. For every peak, we can build a different model.



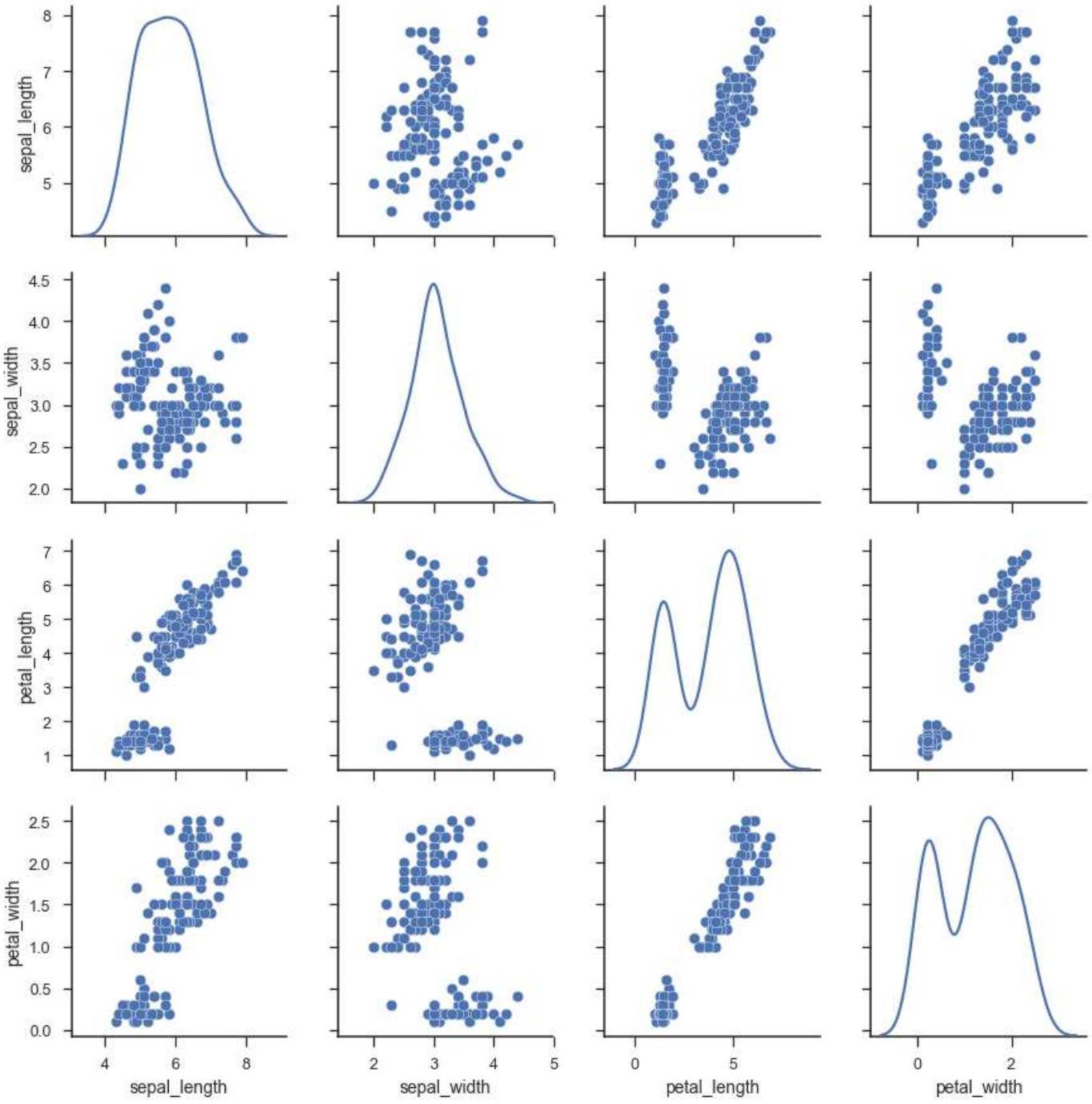
Example of correlation analysis - here we can eliminate features with high correlation or independent for our aim.



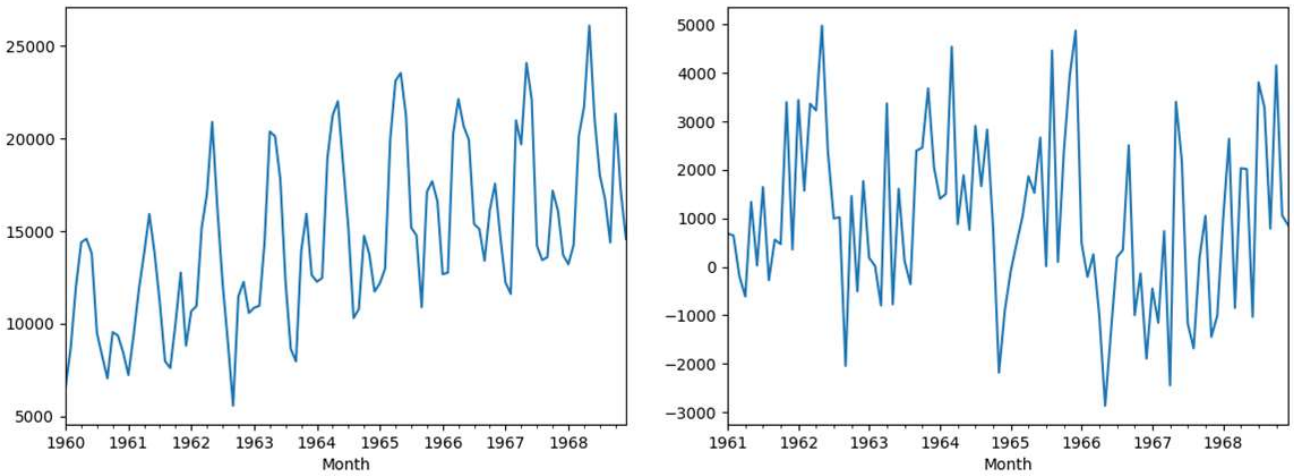
The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the five number summary: Minimum, First quartile, Median, Third quartile, Maximum.



Example of data transformation - make data stationary by its difference.

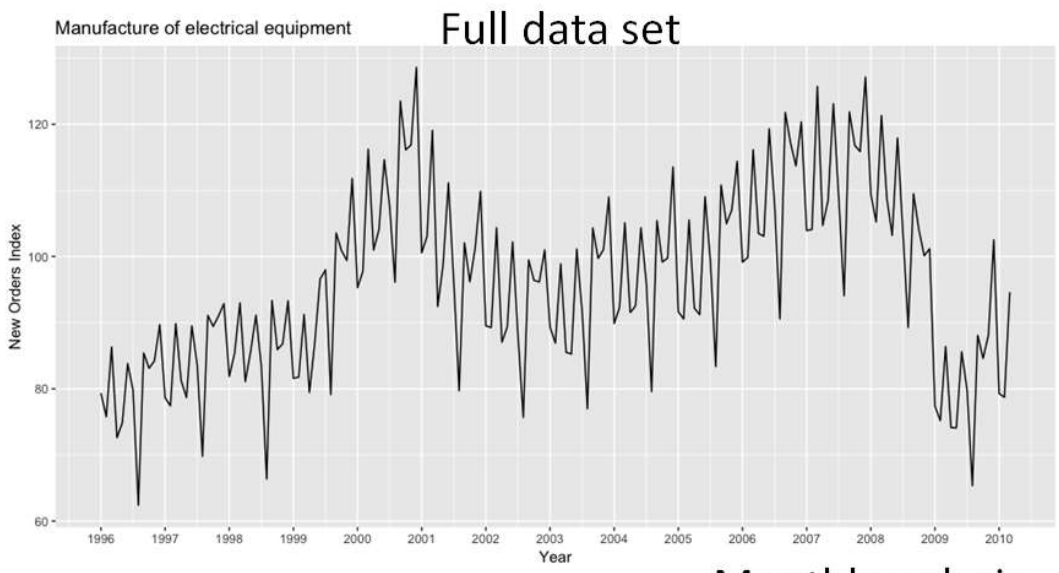


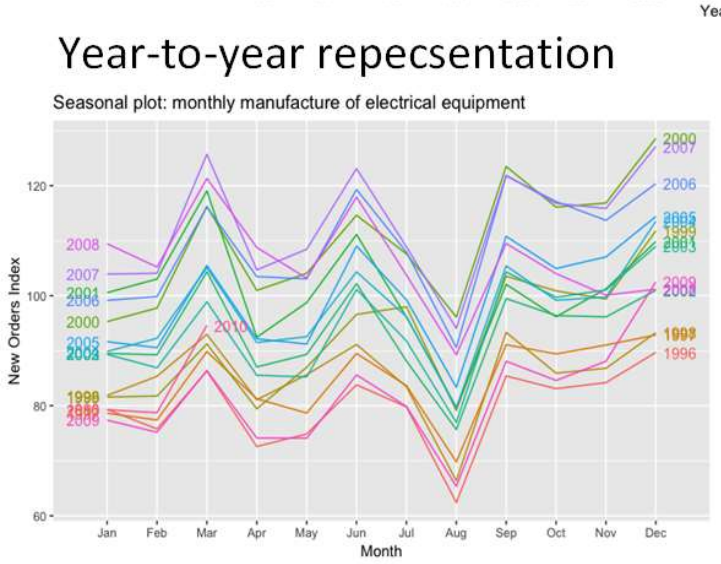
Example of check data stationary.

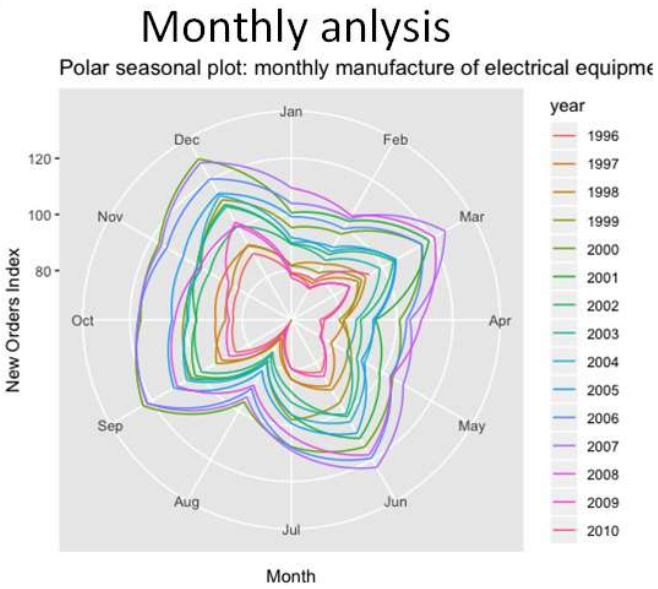


Example of seasonal dependency checking - year dependency by corresponding segmentation of the data.

It is also high monthly dependence is presence (especially in august, where are holidays).

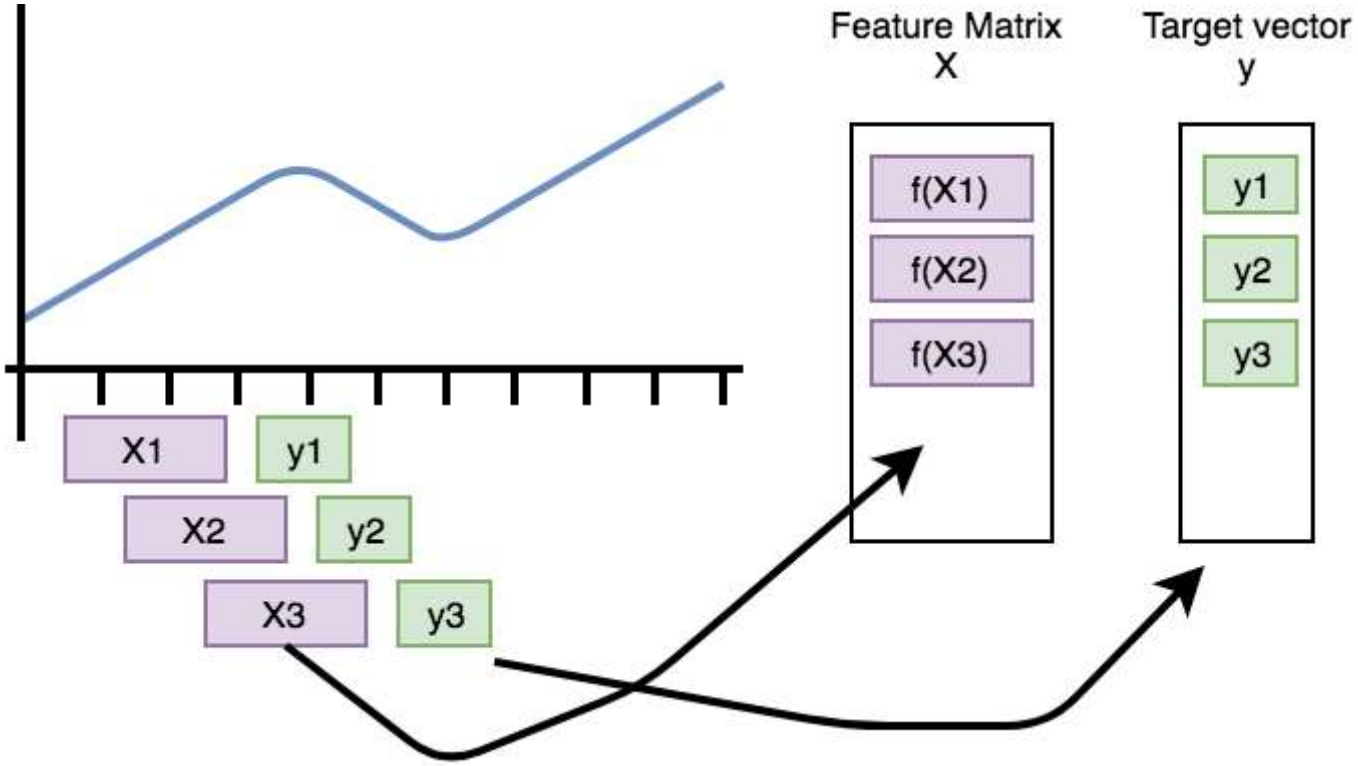






1.2 Feature Representation

Example of time series **Rolling Representation**

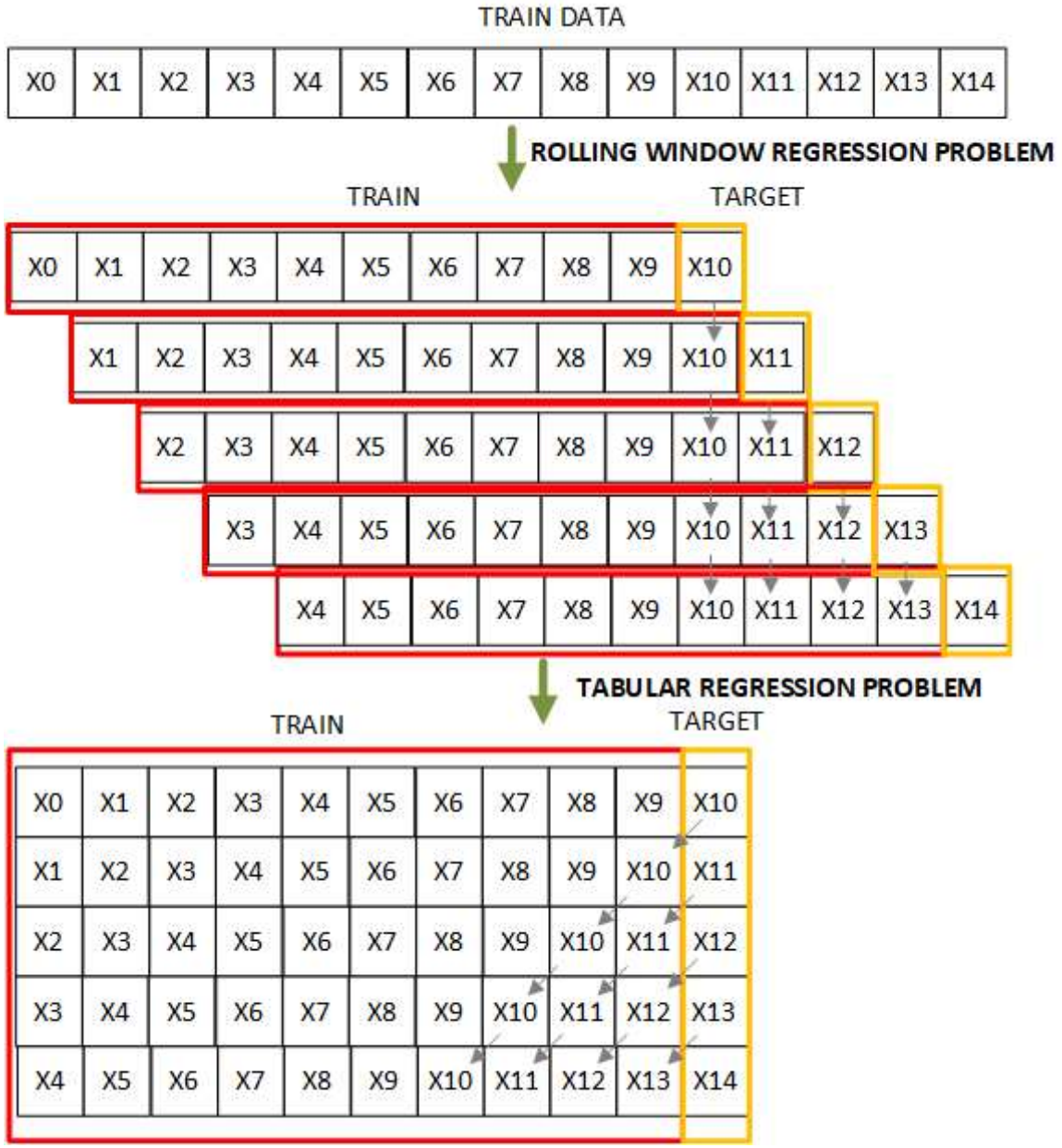


The better way to make forecast using conventional regression - is to use so-called **Forecast Reduction** technique, which is transformation of the implicitly accompanying long-horizon regression problem to the sliding window based problem.

This can be performed by method `make_reduction` as it is shown below.

The idea of Reduction is to reduce the regression through the time series problem to the tabular regression problem like it is shown below.

Please Note that in the tabular or rolling window regression problem, during the training stage you may use the predicted values instead of known in the corresponding positions. Like it is shown by gray arrows.

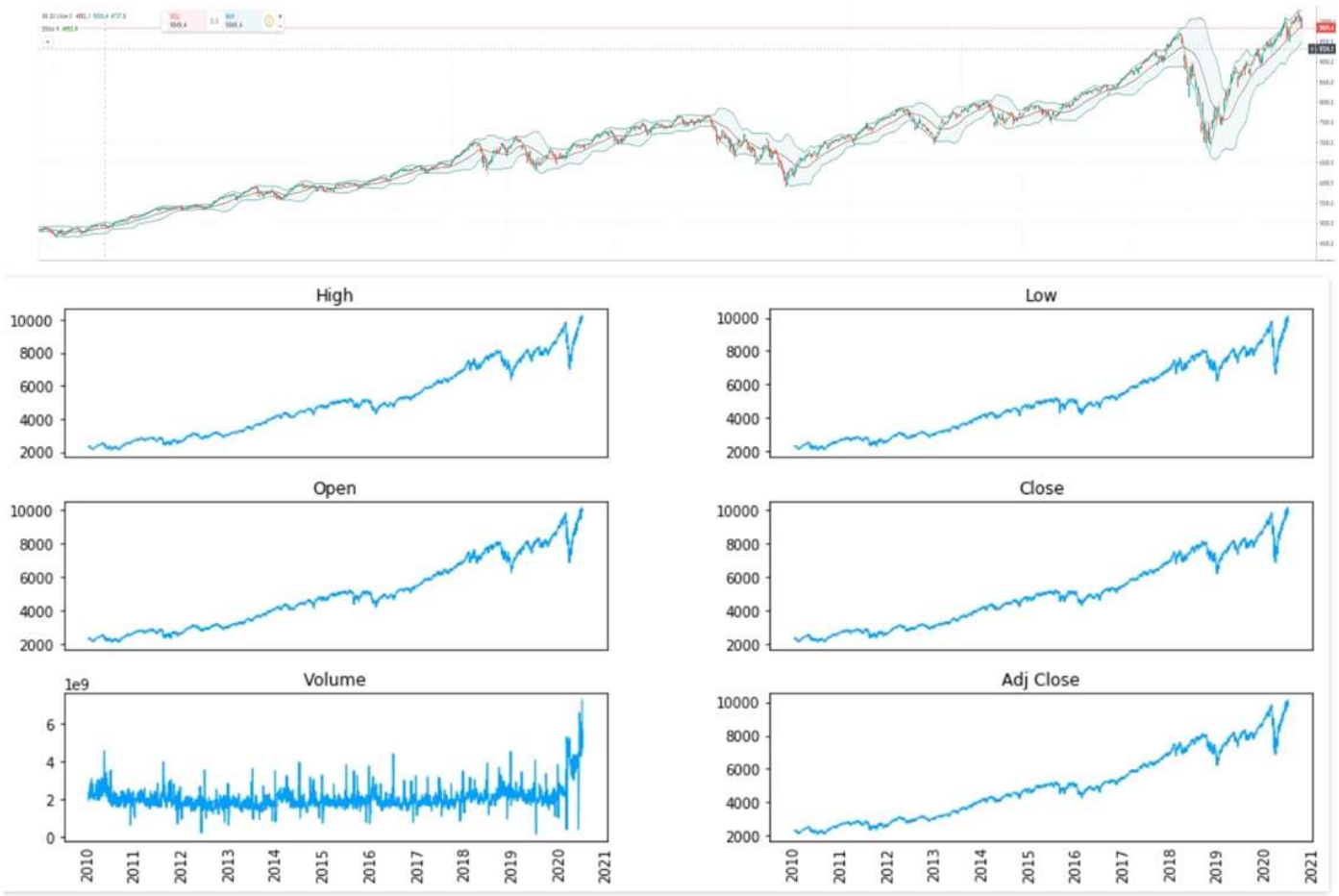


Let's see how method `make_reduction` work. There are "direct", "recursive" and "multioutput" strategies of forecasting.

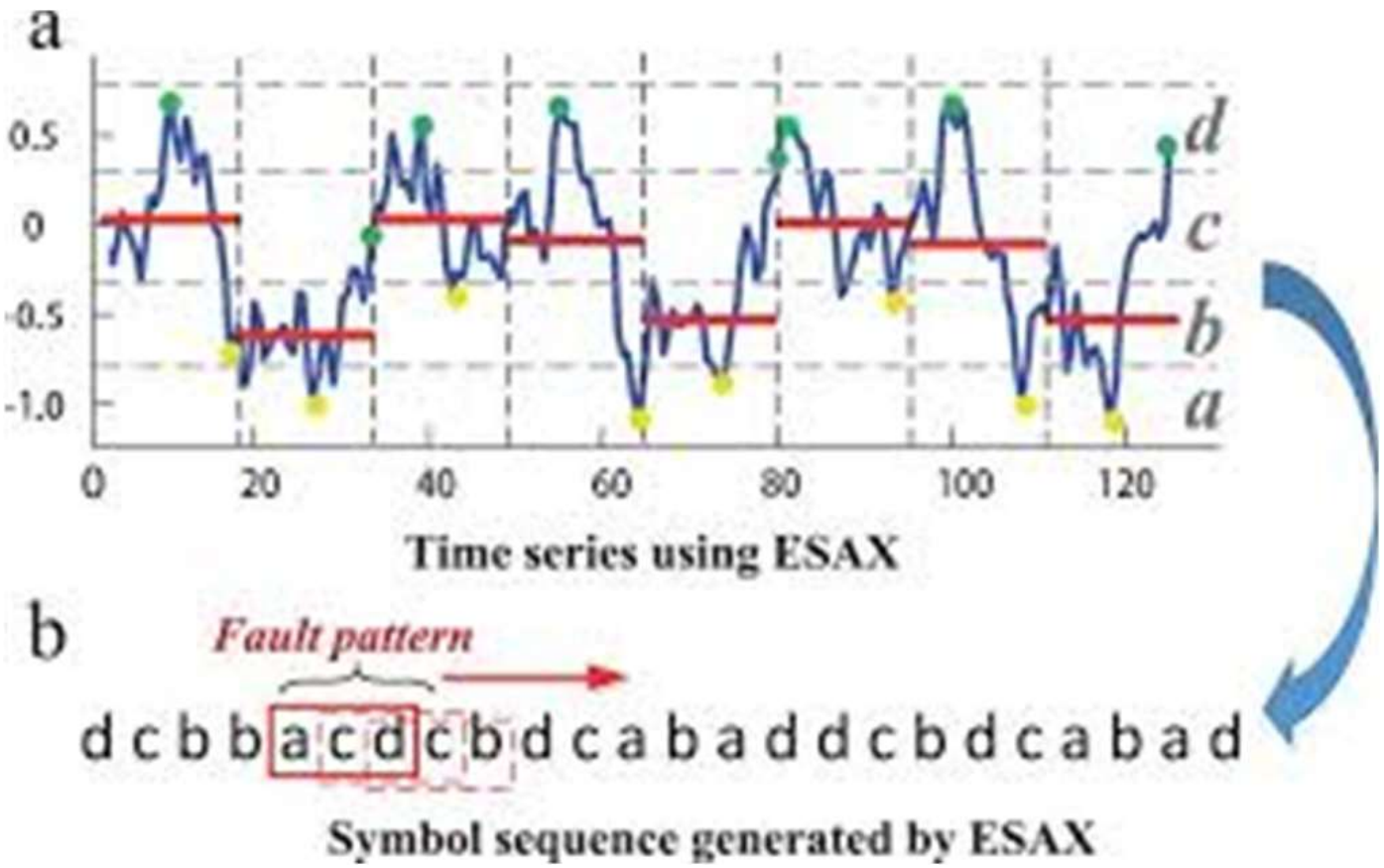
- In the direct strategy we use different forecasts for each output (target) (without gray arrows).
- In the recursive strategy we use previous results in the forecasts for each next output (with gray arrows).
- In the multioutput strategy we directly predict several steps.

Example -

Build a Multivariate Time Series form the Univariate to estimate feature parameter importance



Example of **Symbol-Based (coarse quantization) Time Series Representation**.



Frequency domain representation

The time series can be also represented as its spectrum - i.e. in the frequency domain (in opposite to the time domain).

For the series with unchanged trend behavior and other parameters the time and frequency representation will be equally informative.

If there are exits behavior to time relation, than combined time-frequency representation can be applied.

Both the time and the frequency domains are allow to fully represent the series.

In some cases frequency or time-frequency representation could be more

convenient.

The examples of this approach are wavelet transform, Hilbert-Huang transform (empirical mode decomposition) and other techniques of fine-grain (precision) time series analysis.

The simplest way to obtain frequency representation is using Digital Fourier Transform:

$$Y(\omega_k) = \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N} y_n \cdot \exp\left(-j\omega_k \cdot n\right),$$

where $Y(\omega_k)$ is the spectrum (complex-valued) of the time series y_n;

$\omega_k = 2\pi \cdot k/N$ are the set of frequencies, $k = 0, \ldots, N-1$; $j = \sqrt{-1}$.

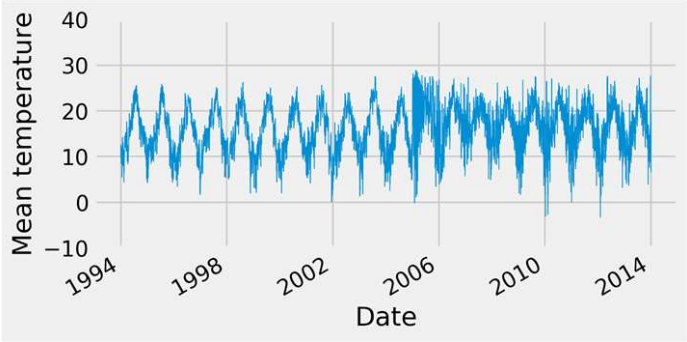
In practice we instead of Digital Fourier Transform the **Fast Fourier Transform (FFT)** is applied. This approach is much more faster (has the order of computational complexity $Nlog_2N$ (while Digital Fourier Transform has the order N^2).

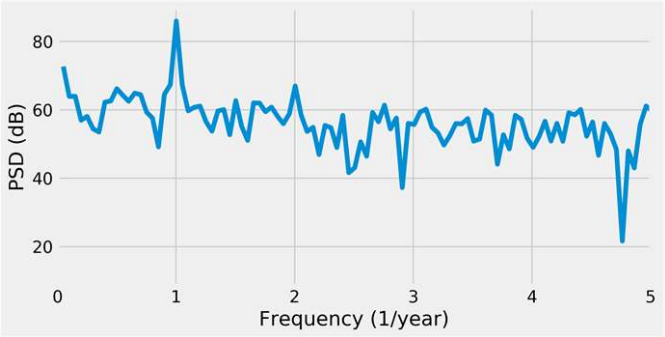
As it can be seen the spectrum representation has its module and phase part (as each complex-valued series),

$$Y(\omega_k) = |Y(\omega_k)| \exp(j \cdot \arg(Y(\omega_k))),$$

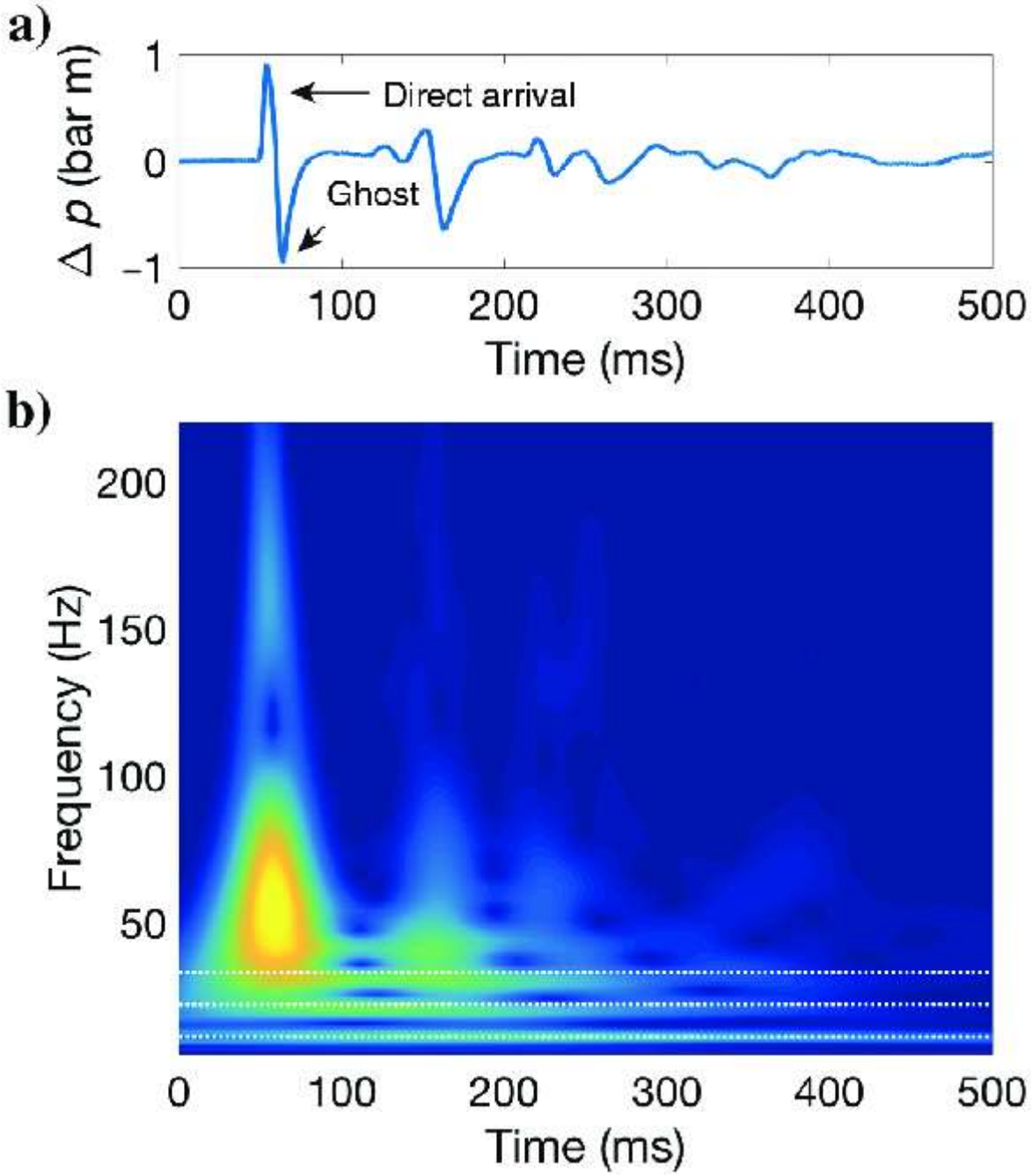
where $\arg(Y(\omega_k) = \arctan(image(Y(\omega_k))/real(Y(\omega_k)))$.

But in the following we will usually mean only $|Y(\omega_k)|$.





By taken Spectrum with some moving window and time series we can obtain time-frequency (3-dimensional) representation of time series.


a)
Δ p (bar m)
1
0
−1
Direct arrival
Ghost
0
100
200
300
400
500
Time (ms)
b)
Frequency (Hz)
200
150
100
50
0
100
200
300
400
500
Time (ms)

Pleas note:

In the frequency domain such "abstract" concepts as trend, seasonality and e.t.c. obtain there physical meaning.

The trend (including cyclic part) is the most low-frequency part of the spectrum with relatively high intensity(amplitude).

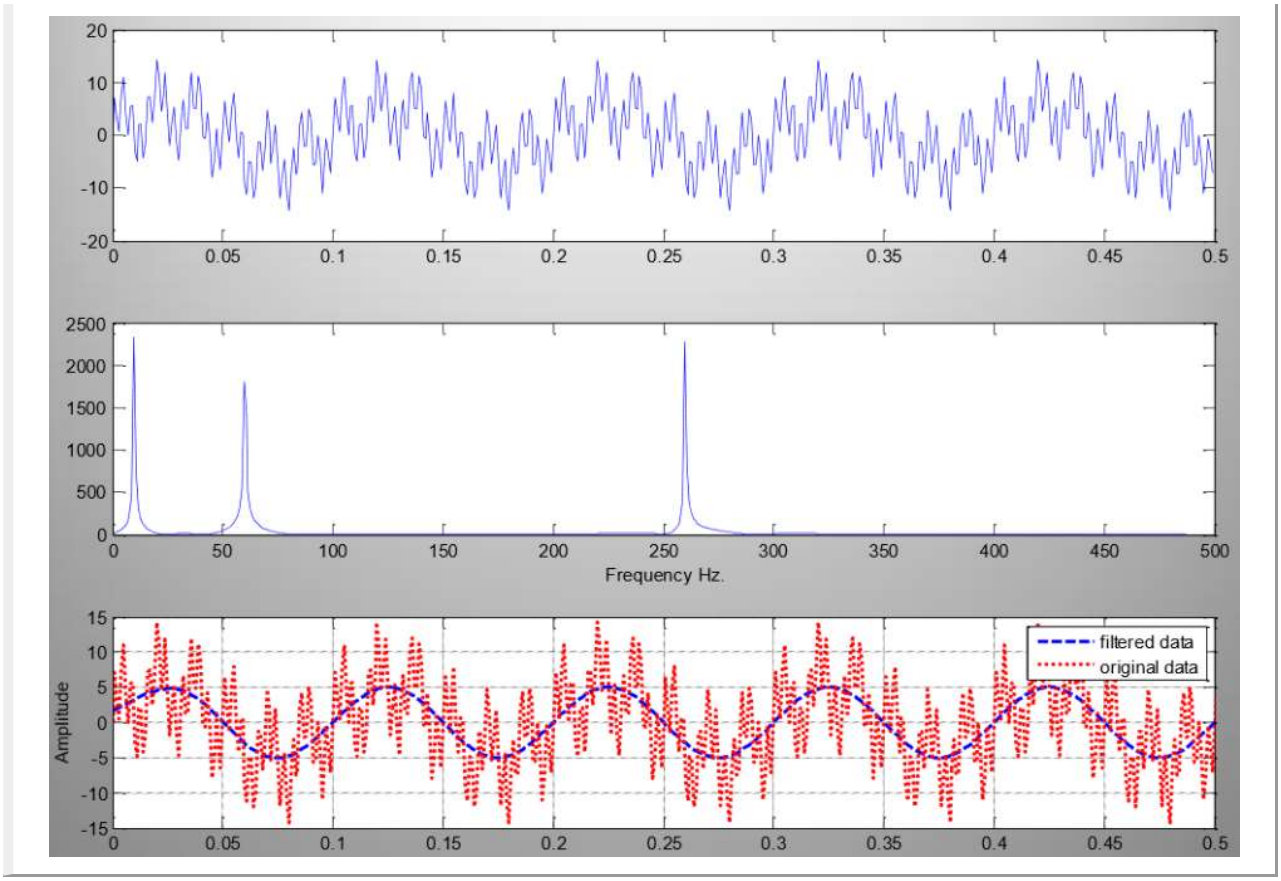
The level is the zero-frequency part of the spectrum.

The seasonality is the relatively high-frequency part of the spectrum but with relatively small intensity(amplitude).

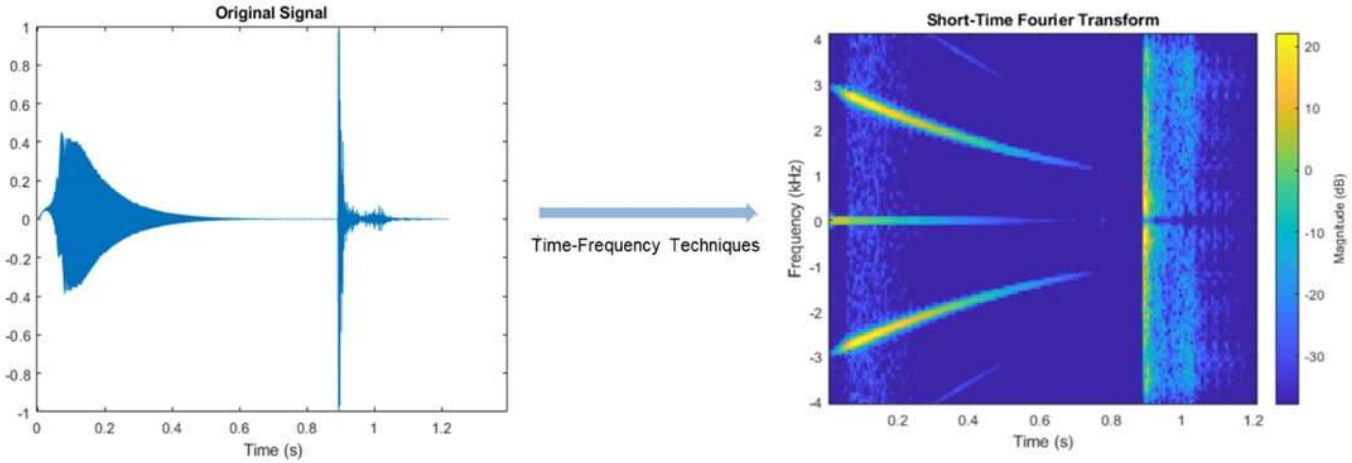
The change of the series behavior mean change of its time-frequency representation behavior.

Extracting some part of time series (i.e. trend or cyclic) is equal to low-pass filtration in frequency representation.

Example of removing seasonal component using low-pass filtration in the spectrum domain

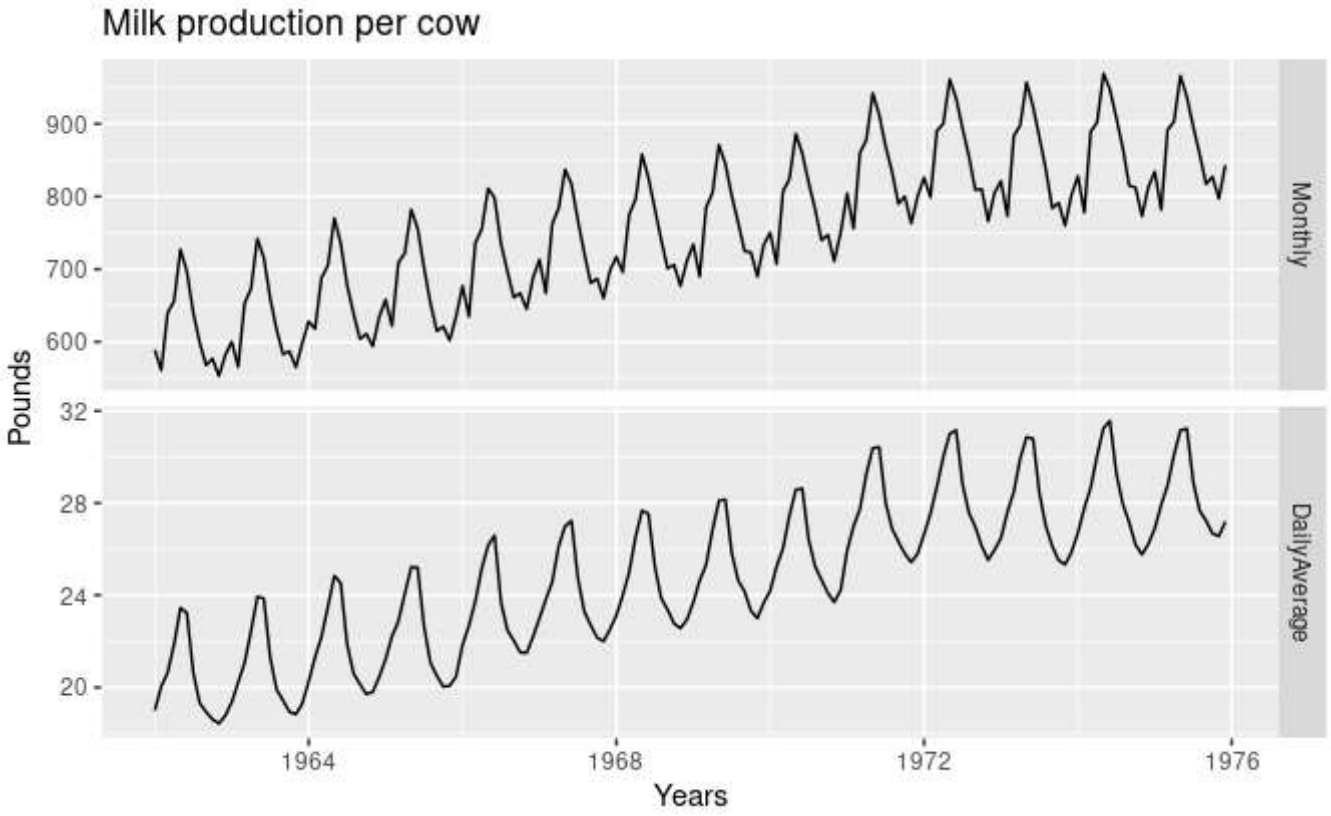


Example of time series short-Fourier transform representation

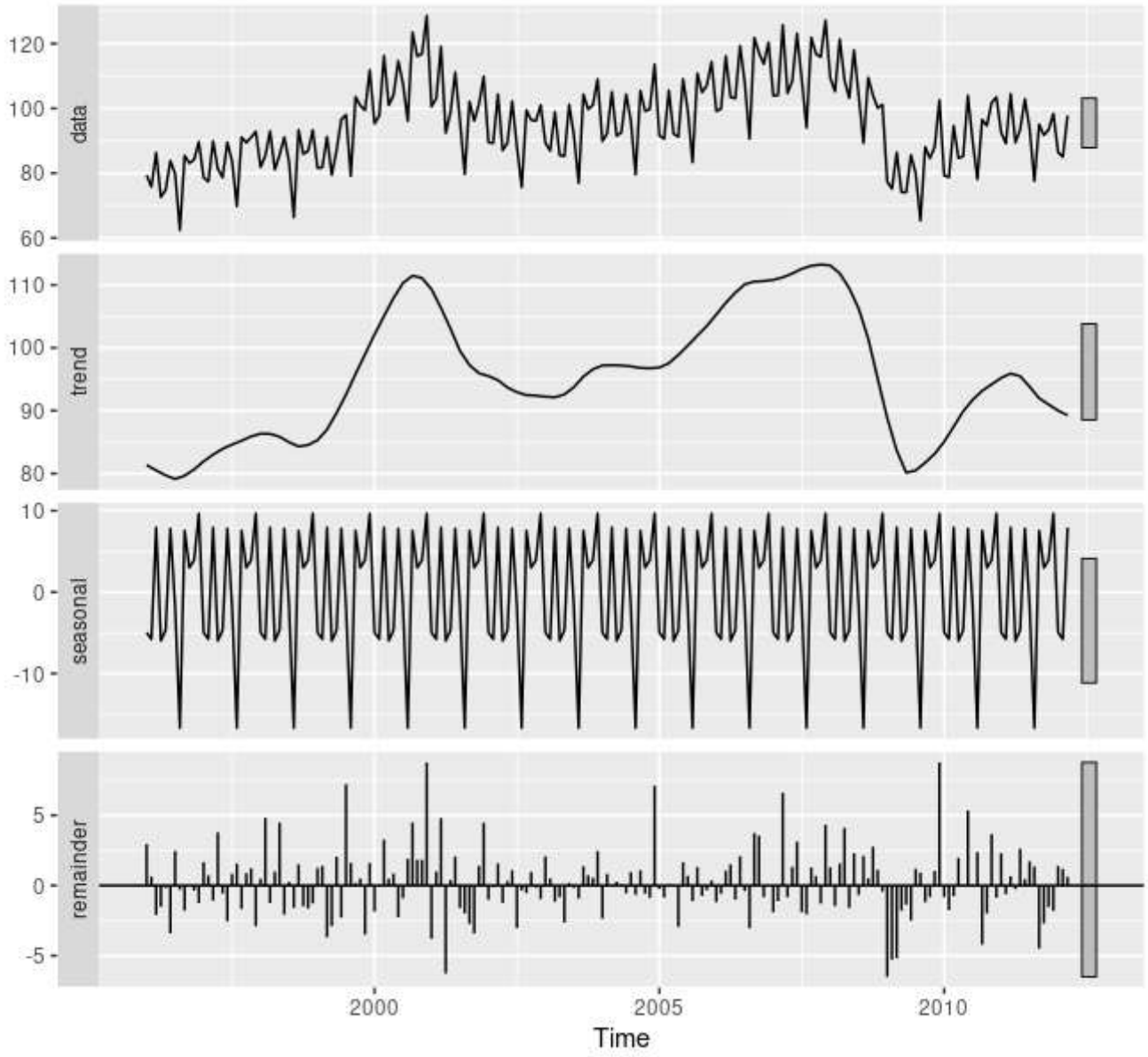


1.3 Feature Extraction

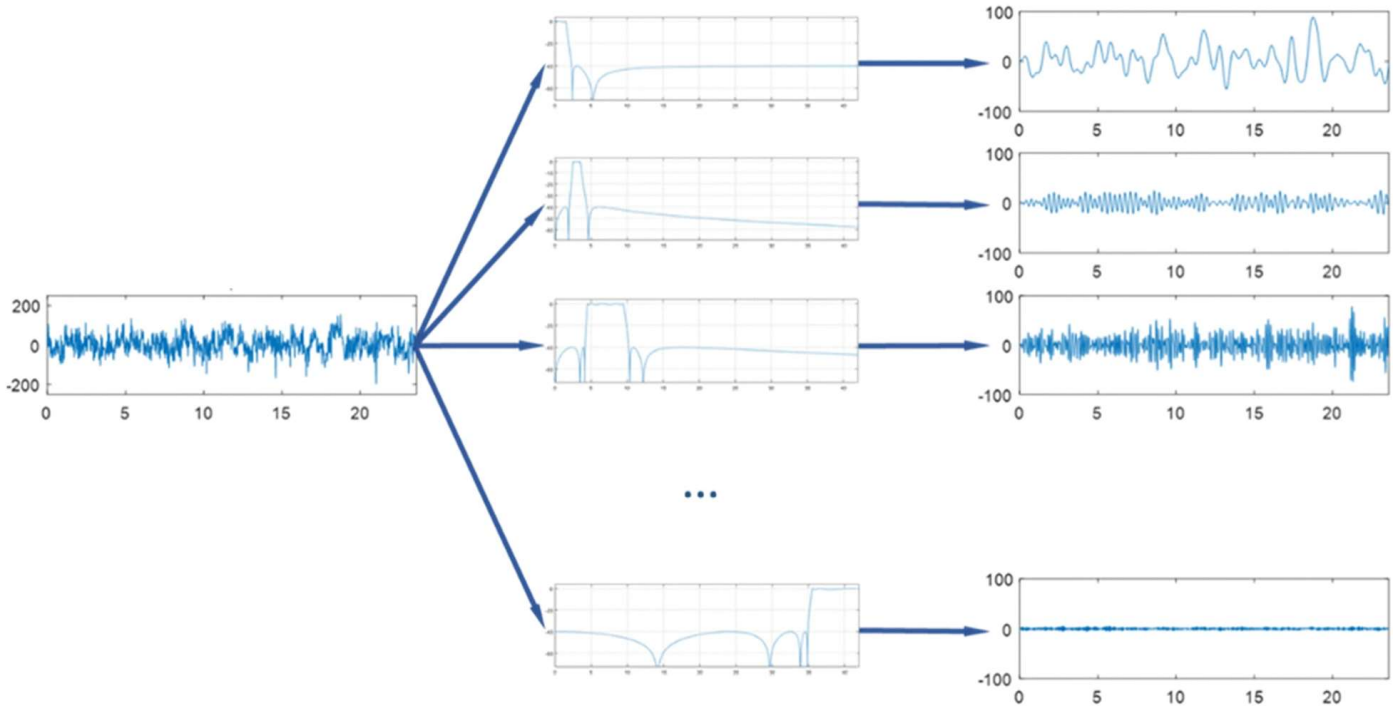
Example where the data are taken simply (upper case) and with averaging by number of days in month



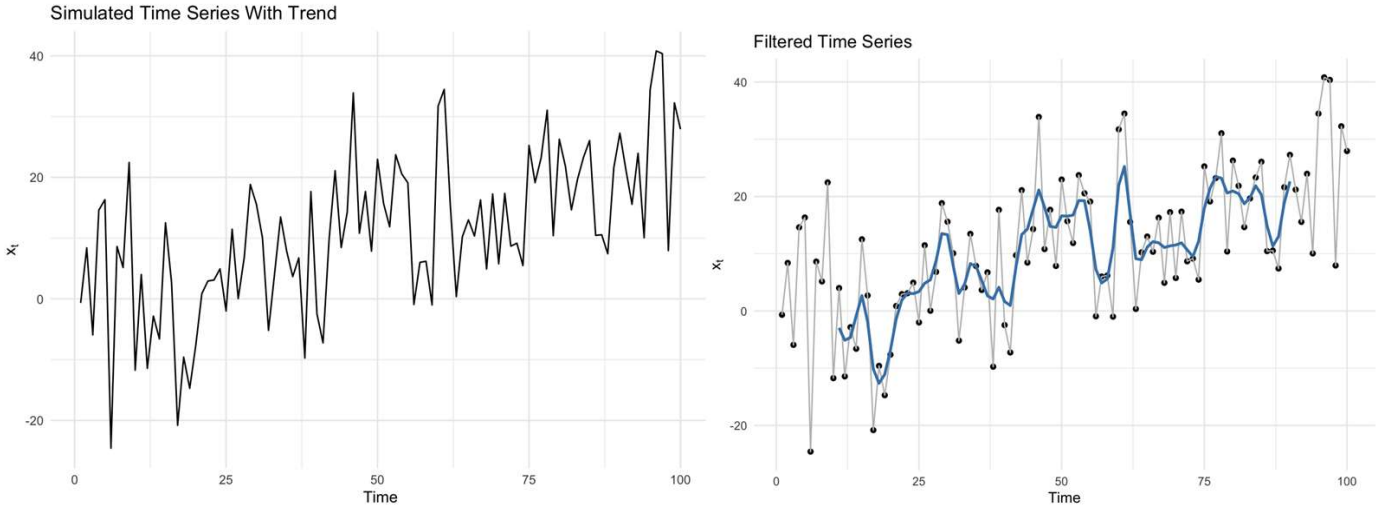
Example of STL - seasonal, trend decomposition,



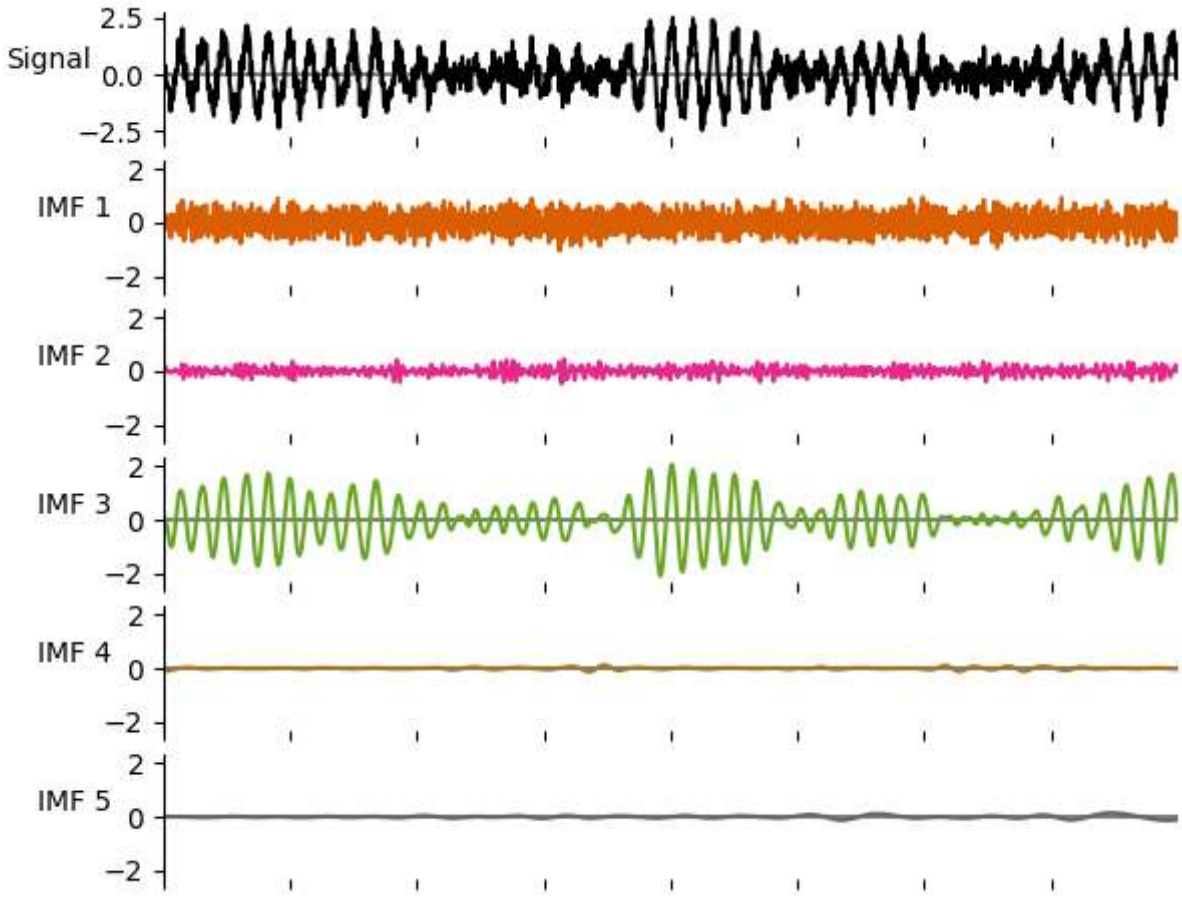
Example of feature extraction using filtration



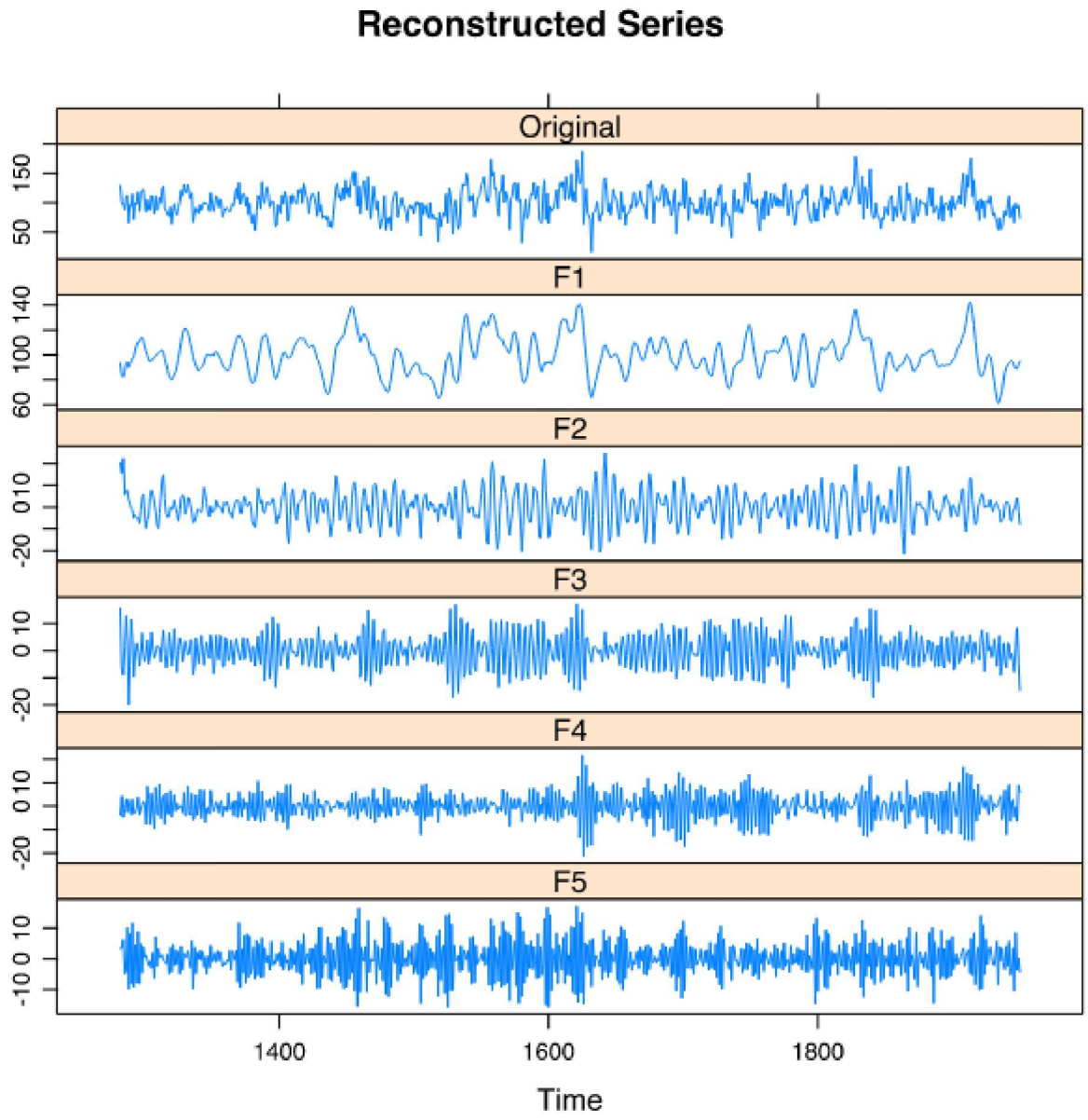
Example of time series low-frequency denoising



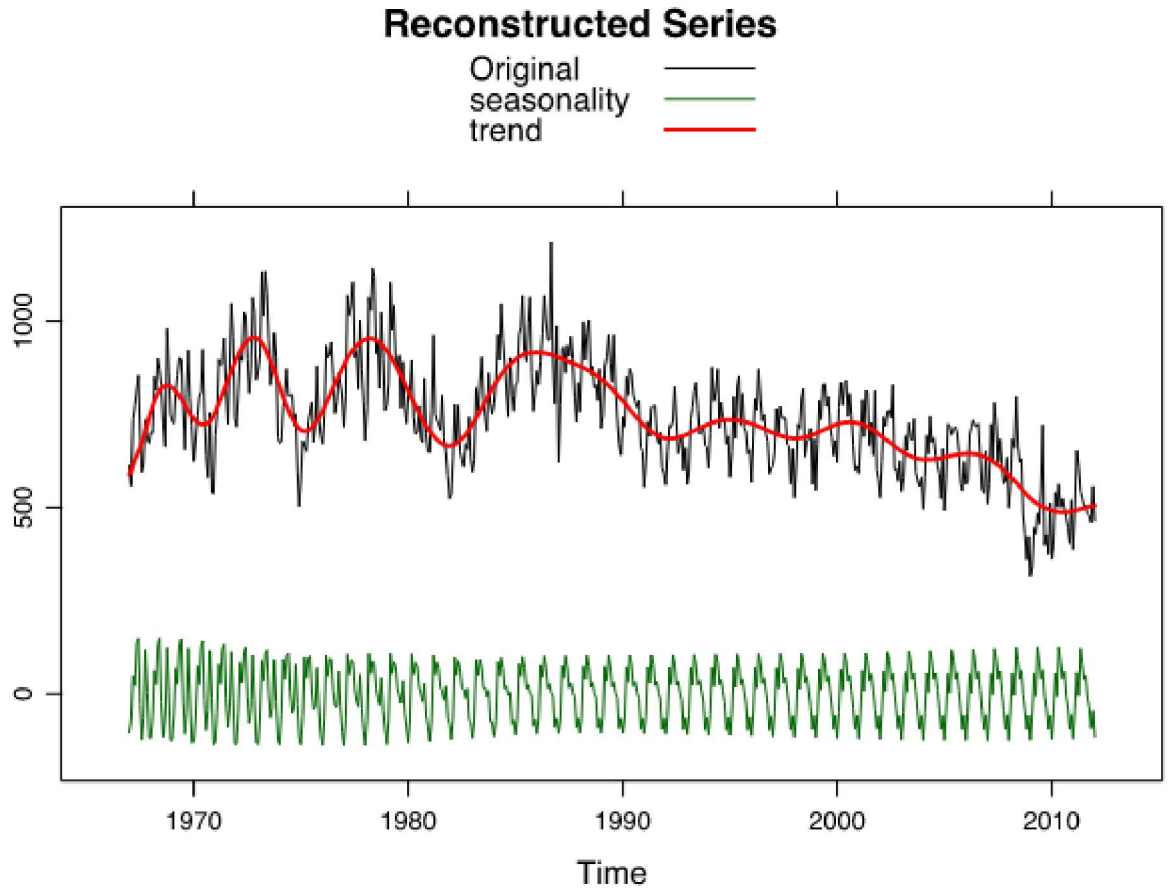
Example of feature extraction using intrinsic mode decomposition and Hilbert-Huang transform



Example of feature extraction using Singular-SPectrum Analysis



Example of feature extraction using Singular-SPectrum Analysis for complex trend extraction



1.4 Time Series Transformations

Types of **time series transformations**:

- Power Transform.
- log-exp Transform.
- Box-Cox Transformation.
- Seasonal Difference.
- Trend Difference.
- Standardization.
- Normalization.

Box-Cox transformation

The Box Cox transformation is an exponent, lambda (λ), which varies from -5 to 5. All values of λ are considered and the optimal value for your data is selected.

The "optimal value" is the one which results in the best approximation of a normal distribution curve. The transformation of Y has the form: boxcox formula

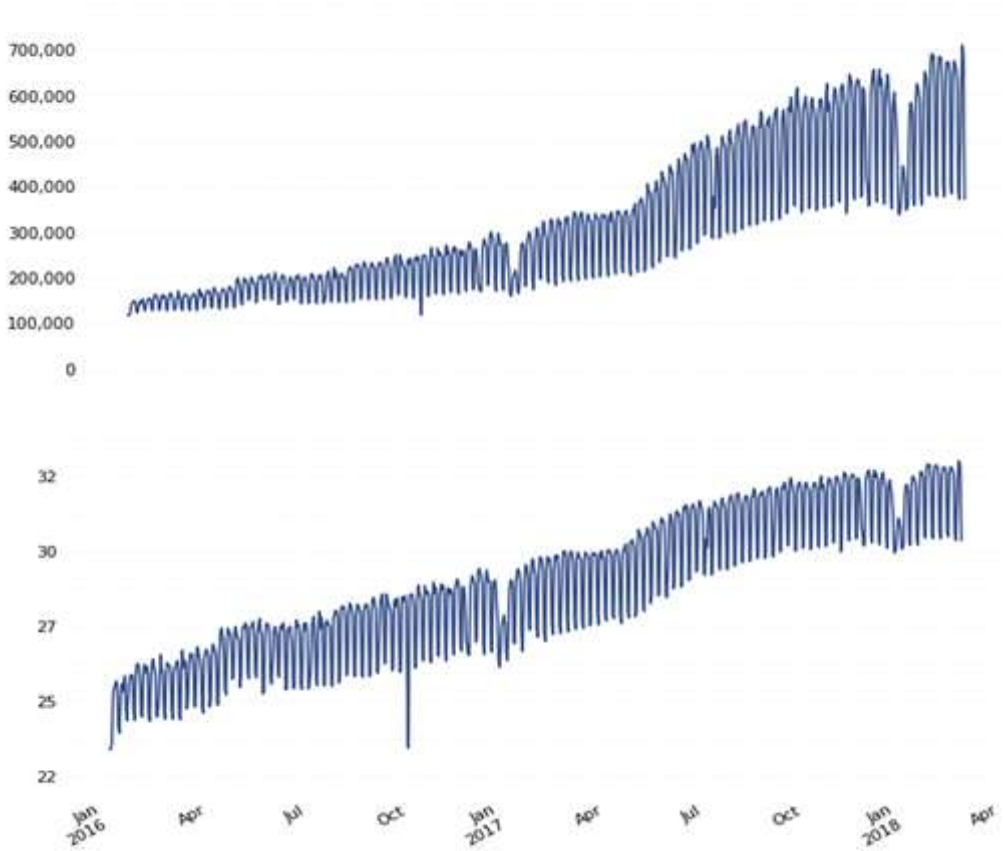
$$y = \begin{cases} \log y, & \text{if } \lambda = 0; \\ \frac{(y^{\lambda}-1)}{\lambda}, & \text{otherwise} \end{cases}$$

This test only works for positive data. However, Box and Cox did propose a second formula that can be used for negative y-values:

$$y = \begin{cases} \log (y + \lambda_2), & \text{if } \lambda_1 = 0; \\ \frac{((y+\lambda_2)^{\lambda_1}-1)}{\lambda_1}, & \text{otherwise} \end{cases}$$

> *Note*: the transformation for zero is $log(0)$, otherwise all data would transform to $Y0 = 1$. The transformation doesn't always work well, so make sure you check your data after the transformation with a normal probability plot.

Example of trend linearization using Box-Cox transform



1.5 Feature Selection

In scope of **feature selection** the technique are typically categorized into wrapper, filter and embedded, univariate and multivariate methods.

- **Wrapper methods** use a predetermined learning algorithm to determine the quality of selected features according to an evaluation metric.
- **Filter methods** apply statistical measures to evaluate the set of attributes.

- **Embedded methods** achieve model fitting and feature selection simultaneously.
- **Multivariate** methods evaluate features in batches.
- **Univariate methods** evaluate each feature independently.

The feature selection methods allowing:

- Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy: Less misleading data means modeling accuracy improves.
- Reduces Training Time and dimension (size): fewer data points reduce algorithm complexity and algorithms train faster.

Filtration based feature selection:

Select subsets of features based on their relationship with the target.

- **correlation analysis** (cross-correlation, canonical-correlation analysis),
- **Mutual Information** Feature Selection (Kulback-Liber or Mutual-entropy analysis, DICE over distributions and other).

- **Variance analysis** (ANOVA, eigenvalues,).
- **Filtration based** on the feature importance (tree-based supervised method).

Correlation analysis and Mutual Information

- The methods are used as a measure linear dependency between pairs of variables (features).
- If normolized correlation coefficient is measured (cos-distance),

  Ranges between -1 to 1 mean, that values closer to 1 shows that they are highly correlated and value closer to -1 indicates that they are negatively correlated.
- As usual features with values above some threshold (± 0.9) are ejected.
- Any other distance function can be allied instead of correlation (for instance Chi-Square).
- In the case of Mutual Information Feature Selection information function (entropy, KL divergence or other are applied instead of cross-correlation).

Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.

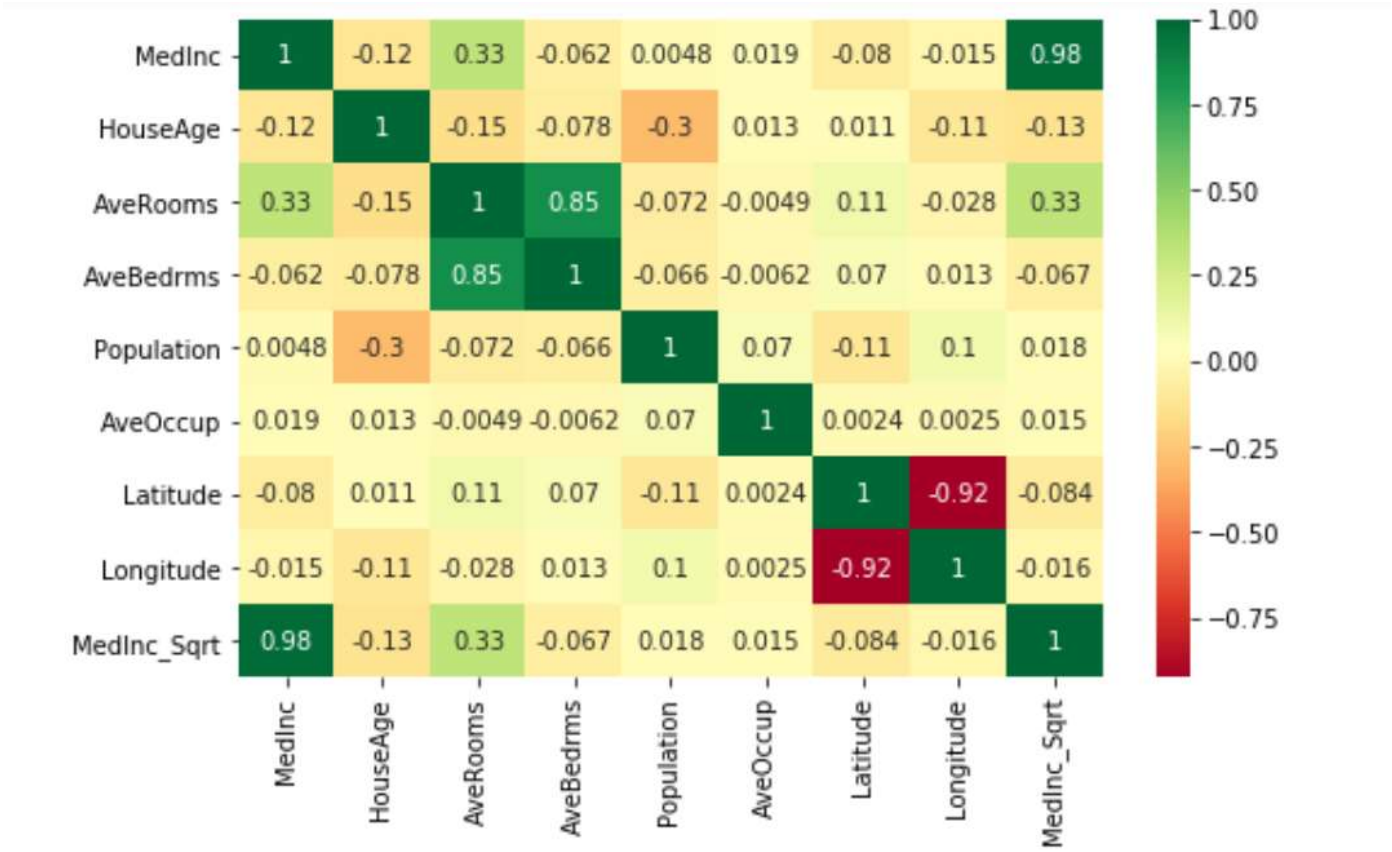
Note

- In other interpretation the **k-best (most independent) feature** can be rest based on the Correlation or distance analysis.
- In the classification supervised case the importance of each feature can be estimated for each class, for instance Fisher Score in this case can be given as

$$F_i = \frac{\sum_{j=1}^{J} n_j(\mu_{ij} - \mu_i)^2}{\sum_{j=1}^{J} n_j \sigma_{ij}^2},$$

  where μ_{ij} and σ_{ij}^2 are mean and variance of i-th feature in j-th class, n_j is the number of instances in j-th class and μ_i is the mean of i-th feature.
- For Regression task feature importance can be also estimated by analyzing the similarity of features values in the training and test datasets.

Example of correlation based feature selection



ANalysis Of VAriance (ANOVA):

ANOVA is a statistical method, used to check the means of two or more groups (features in time series that are significantly different from each other.

It assumes Hypothesis on testing feature (or variable or column of data)

- Null: Means of all groups are equal.
- Alternate: At least one mean of the groups are different - we need this.

Features of ANOVA

- By using ANOVA we are want to determine the set of features which are explain (terms of) the variance of the data.
- ANOVA allow one to select a set of features which mostly independent one form other and in sum produce full variance of the data.
- Use ANOVA when you have one categorical feature and one numerical (like sex (Male, Female) and age 0-100 or like class labels and data).
- In ANOVA we divied our features on the groups in correspondence with categorical feature (Male-Ages, Female-Ages).
- For Feature (or variable) selection we test the hypothesis that distribution of this feature differs from other for each group.

For determining the ANOVA test the Fisher criteria of each feature j can be give as

$$F_j = \frac{MS_{bg}}{MS_{wg}^j} = \frac{N-J}{J-1}\frac{\sum_{j=1}^{J}(M_j - M_{tot})}{\sum_{i=1}^{N}(x_{ij} - M_j)},$$

where:

- MS_{bg} is the sum of squares between groups,

  $MS_{bg} = \sum_{j=1}^{J}(M_j - M_{tot})/(J-1)$,

  M_j is the mean of each feature j and M_{tot} is the mean of all M_j,

  J is number of features (groups);

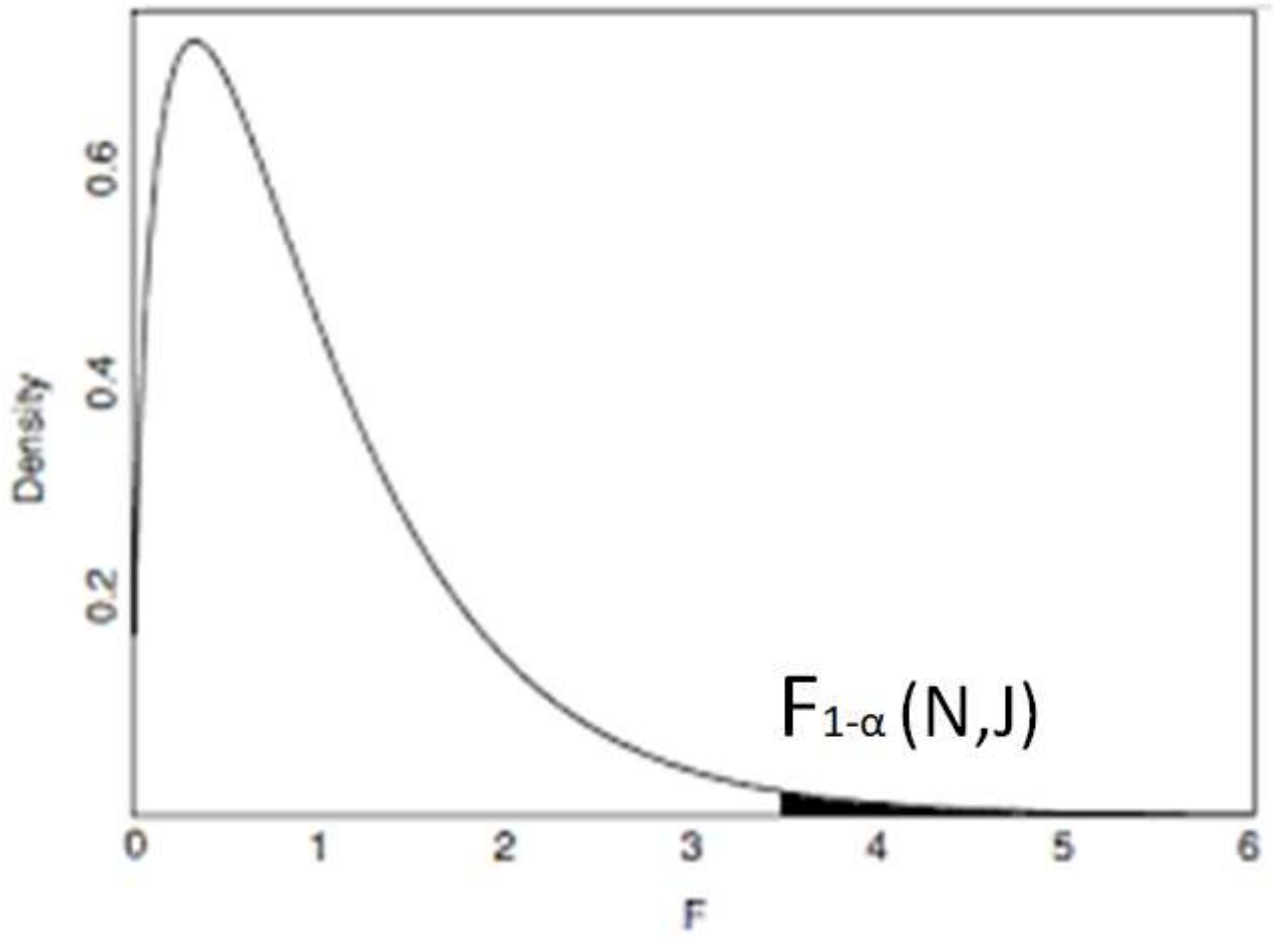
- MS_{wg}^{j} is the sum of squares within group j,

  $MS_{wg}^{j} = \sum_{i=1}^{N}(x_{ij} - M_j)/(N-J)$, N is the size of feature.

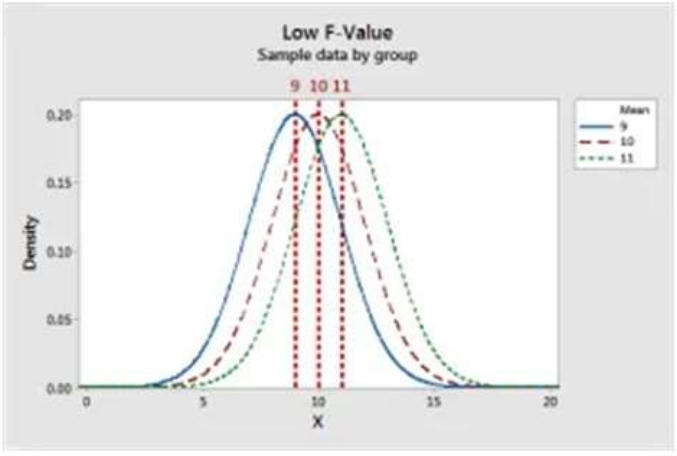
  After F_j calculation the influence of variables can be calculated with predefined confident interval using F-tables.

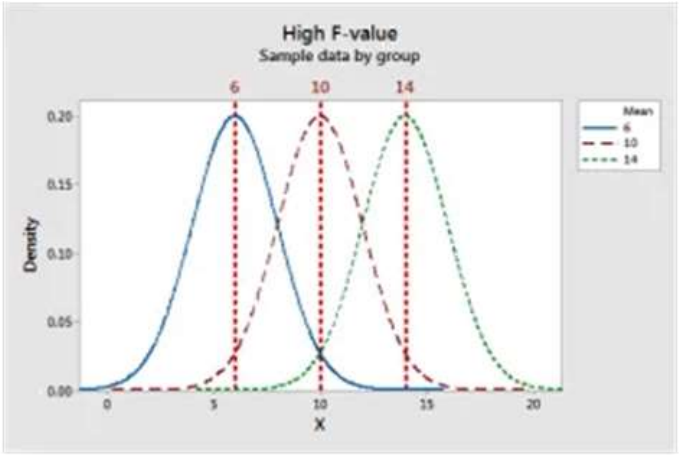
  Note, the threshold F-value is the function of confident interval and degrees of freedom (Js and N).

Here if value is higher then threshold ($F_{1-\alpha}(N, J)$, then both features are important.



Example of low-F-value (high coincidence, remove all expect one features) and high F-value (low coincidence, rest all features).





Example of case where features are important

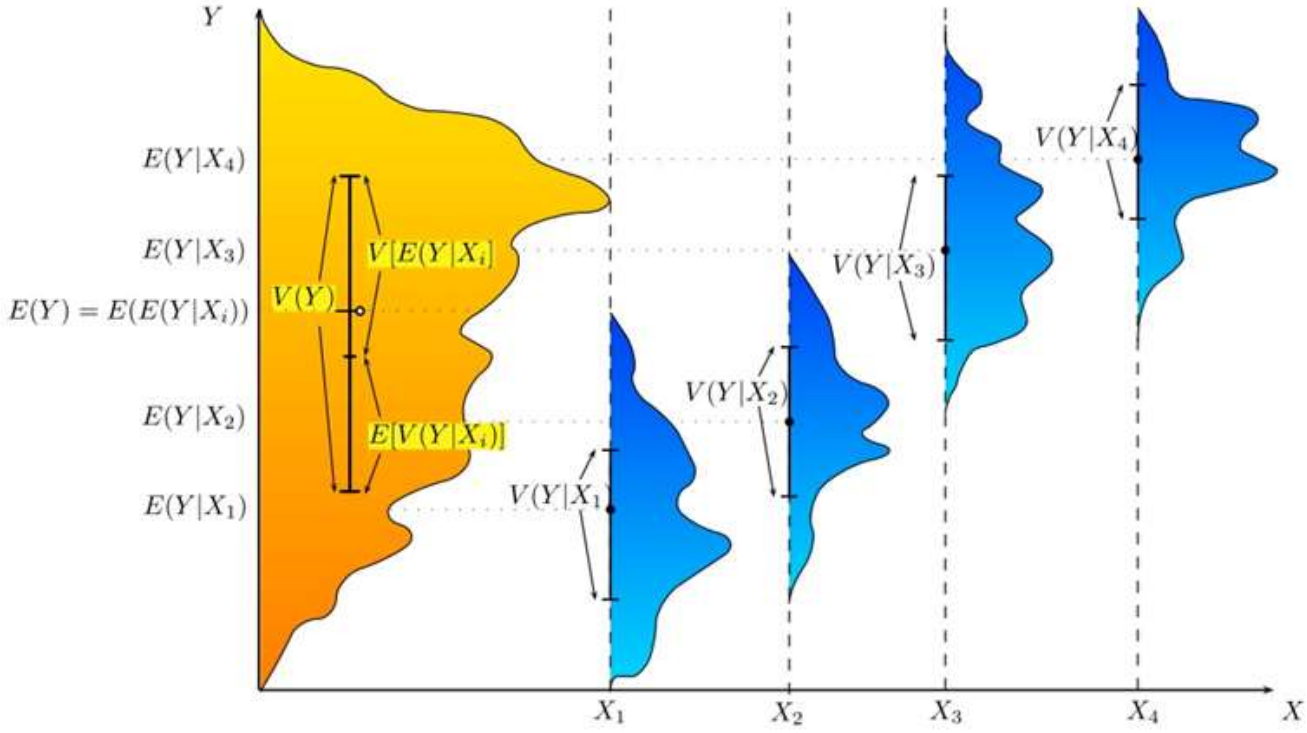


Figure 1: ANOVA : Fair fit

Note

- In the case of two features ANOVA is analogue of t-test.
- Beside ANOVA variance-based feature selection can be done using **variance threshold**.
  - **Variance threshold method** is based on the assumption that when a feature doesn't vary much within itself, it generally has very little predictive power.

    Thus almost constant features can be eliminated.

- The analogue of variance threshold is **eigenvalues analysis and PCA**

> - As a rule, ANOVA is used when one variable is numeric and one is categorical, such as numerical input variables and a classification target variable in a classification task.
> - ANOVA is analogue of t-test but allow to check dependencies between 2,3 or more features

Feature importance

The idea of the approach is to describe the feature relative importance to a model.

After select highest importance and eliminate weak features or combinations of features and re-evalute to see if the model fairs better during cross-validation.

In scikit-learn, Decision Tree models and ensembles of trees such as Random Forest, Gradient Boosting, and Ada Boost provide a `feature_importances_` attribute when fitted.

In general Gradient Boosting is better than Random Forest.

Most common way to select a feature by importance is to use tree-based methods

- The feature importance in tree is based models are calculated based on Gini Index (Gini Impurity), Entropy or Chi-Square value.
- Then higher the feature position in tree, than it more important (then it more decreases impurity).
- Then more near split by feature to the tree trunk, than it more important (then it more decreases impurity).
- *The zero importance of feature mean that it is not used to split any nodes, and so we can remove it without affecting the model performance.*

- *Correlated features show similar importance, so if one removes one of the correlated feature, the other feature shows grows of importance.*
- In theory of decision trees then more a feature decreases impurity, more important the feature is.

> Tree based models calculate feature importance for they need to keep the best performing features as close to the root of the tree.
>
> The approach is inverse of three-based anomaly detection.

Note

- **The straightforward feature selection based on its importance not always provide correct results.**

  In some case it is recommended to make feature importance estimation iteratively, with change on of the feature (in cycle) to noise or with removing it for one iteration.
- If Random Forest is used for feature selection the importance can be calculated using:

- analysis of mean decrease in impurity (MDI, using sum of Gini impurity) for each feature;
- mean decrease in accuracy (MDA) by replacing of feature (variable, column) on the noise with similar distribution

  or by permutation (shuffle) its values (permutaed feature is almost the noise with the same distribution as the feature);
- by performs on all variables at the same time and concatenates the noised features(shuffled) with the original ones (Boruta).

- If gradient-boosting is used for feature selection the importance can be calculated using:

  (`LightGBM`) importance_type (string, optional (default= `split`)) — How the importance is calculated. If `split` , result contains numbers of times the feature is used in a model. If `gain` , result contains total gains of splits which use the feature.

  (`XGBoost`) `weight` — the number of times a feature is used to split the data across all trees. `gain` — the average gain of the feature when it is used in trees `cover` — the average coverage of the feature when it is used in trees, where coverage is defined as the number of samples affected by the split

Wrapper methods

Wrapper methods are based on greedy search algorithms as they evaluate

all possible combinations of the features and select the combination that produces the best result for a specific machine learning algorithm.

- **Step-Forward wrapper selection**
  - Start with having no feature in the model.
  - In each iteration, search to adding the feature which best improves an accuracy
  - Repeat till an addition of a new variable does not improve the performance of the model or by the certain number of features are selected.
  - The best depends entirely on the defined by such evaluation criteria as AUC, prediction accuracy, RMSE, etc.

Stepwise/Subset Selection

The algorithm is similar to the forward selection process, but a variable can also be dropped if it's deemed as not useful any more after a certain number of steps.

Step-Backwards wrapper selection (Recursive Feature Elimination, RFE)

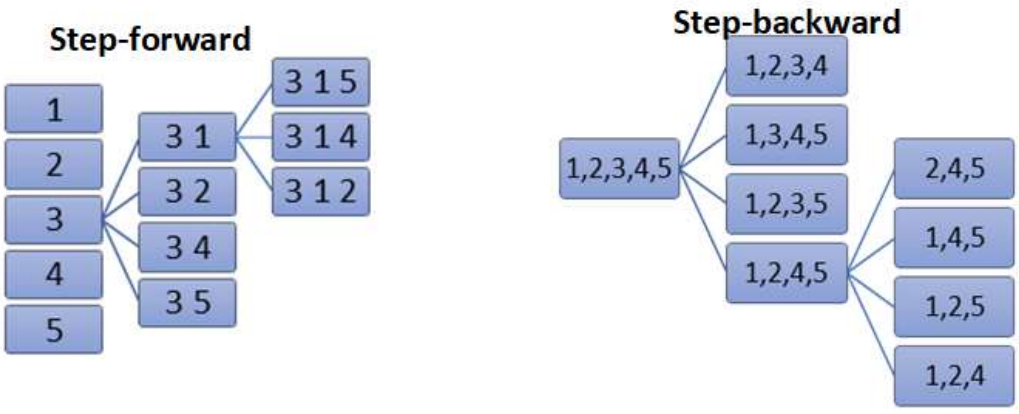
- Start with having all feature in the model.

- In each iteration remove a feature that the smallest influence on the accuracy.
- Repeat till find the optimal subset of a predefined size.

Exhaustive feature selection:

- At each iteration repeatedly creates a several models (algorithm) and evaluate performance among all the possible combination of features.
- Keeps aside the best or the worst performing feature at each iteration.
- Ranks the features based on the order of their elimination.
- Leave the best combination of the predefined size.

Illustration of wrapped methods



Note

Wrapper methods can be applied only for test (or validation) part. In this case instead of removing a feature we can replace it with random noise - feature column is still there, but it no longer contains useful information. This method works if noise is drawn from the same distribution as original feature values (as otherwise estimator may fail). The simplest way to get such noise is to shuffle values for a feature, i.e. use other examples' feature values - this is how permutation importance is computed.

Embedded Method

Embedded Method is inbuilt variable selection method.

- Use regularization (L1, L2, Elastic) for each feature during algorithm training.

  frequently L1(Lasso) penalization is used here.
- After training each feature obtain its own regularization weight.
- Not important feature are given very low weight(close to zero).

- Select the features with non-zero coefficients (or coefficient above some threshold or k-best feature).

> Embedded technique is faster and less expensive than wrapper and more accurate than filter.

In [ ]: