PROCESSING BIG DATA FOR ANALYTICS – FEATURE ENGINEERING

Liu Fan

isslf@nus.edu.sg

Jan 2023

Total Slides: 86

Feature Engineering

Agenda

Topics

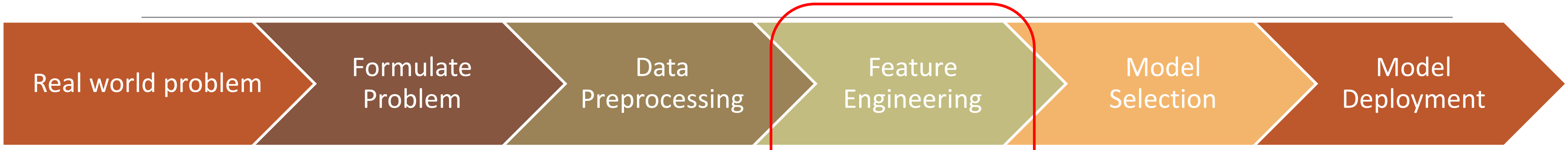
- Feature engineering techniques
- Feature selection/extraction techniques
- Synthetic data generation

Objectives

At the end of this module, you should be able to:

- Derive features for analytical tasks using techniques showcased in this session
- Understand the importance of generating simulated data for required use cases

Machine Learning Lifecycle



- Start with real-world problem

- Collect business requirements
- Analysis problems
- Collect raw data
- Set up main tasks

- Data integration
- Data cleaning
- Data exploration
- Sampling data

- Feature Extraction
- Feature Selection

- Model training
- Model evaluation
- Model tuning
- Model selection

- Model Deployment & Monitoring

Feature Engineering

Feature

- A feature is a **composite** numeric representation of raw data.

Feature engineering

- The process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unexplored data.

Importance of Feature Engineering

- Better features → better prediction results
- Better features → simpler models
- Better features → less computation time

Why

- Curse of dimensionality

Dimension Reduction

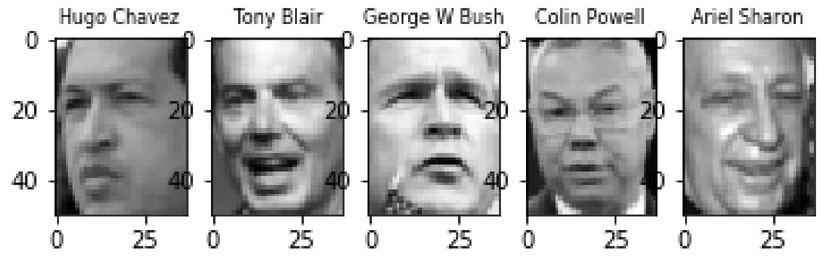
- **Dimension reduction** - transformation of data from a **high-dimensional space** into a **low-dimensional space** so that the low-dimensional representation retains some meaningful properties of the original data.

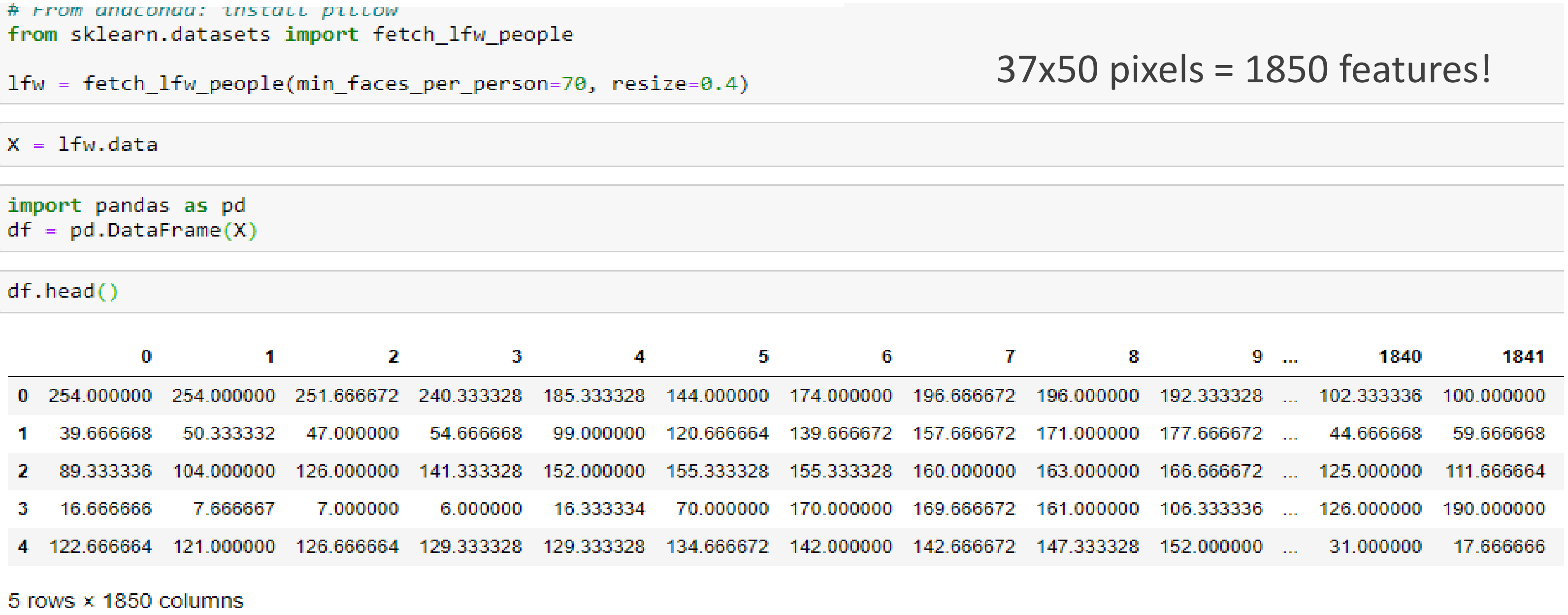
- Dimensionality reduction is common in fields that deal with large numbers of observations and/or large numbers of variables, such as signal processing, speech recognition, neuro-informatics, and bioinformatics.

- Approaches
  - Feature Selection – original features are maintained
  - Feature Extraction – transform the data onto a new feature space

https://en.wikipedia.org/wiki/Dimensionality_reduction

High Dimensional Features





```
# From anaconda: install pillow
from sklearn.datasets import fetch_lfw_people

lfw = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
```

```
X = lfw.data
```

```
import pandas as pd
df = pd.DataFrame(X)
```

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	1840	1841
0	254.000000	254.000000	251.666672	240.333328	185.333328	144.000000	174.000000	196.666672	196.000000	192.333328	...	102.333336	100.000000
1	39.666668	50.333332	47.000000	54.666668	99.000000	120.666664	139.666672	157.666672	171.000000	177.666672	...	44.666668	59.666668
2	89.333336	104.000000	126.000000	141.333328	152.000000	155.333328	155.333328	160.000000	163.000000	166.666672	...	125.000000	111.666664
3	16.666666	7.666667	7.000000	6.000000	16.333334	70.000000	170.000000	169.666672	161.000000	106.333336	...	126.000000	190.000000
4	122.666664	121.000000	126.666664	129.333328	129.333328	134.666672	142.000000	142.666672	147.333328	152.000000	...	31.000000	17.666666

5 rows × 1850 columns

37x50 pixels = 1850 features!

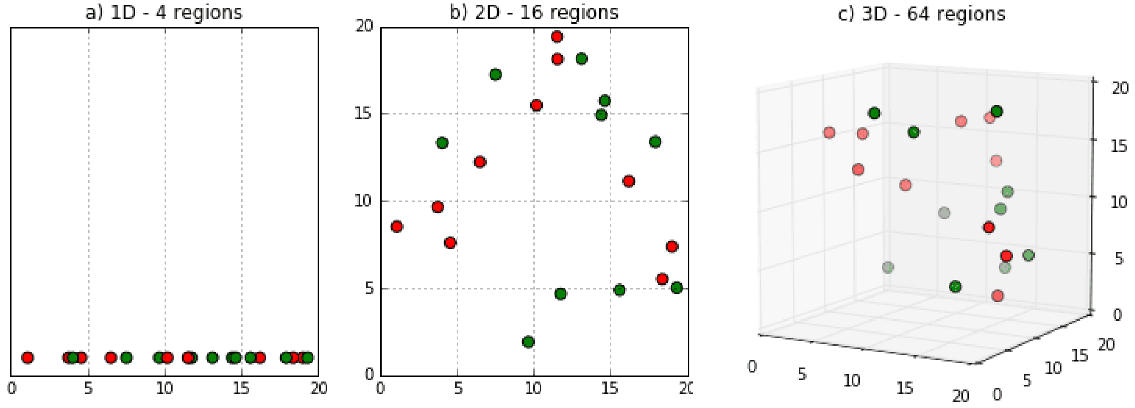
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html

Curse of Dimensionality

- The **Curse of Dimensionality** is termed by mathematician R. Bellman in his book “Dynamic Programming” in 1957.
- **Curse of Dimensionality** is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. In order to obtain a reliable result, the amount of data needed often grows exponentially with the dimensionality.

https://en.wikipedia.org/wiki/Curse_of_dimensionality

Curse of Dimensionality

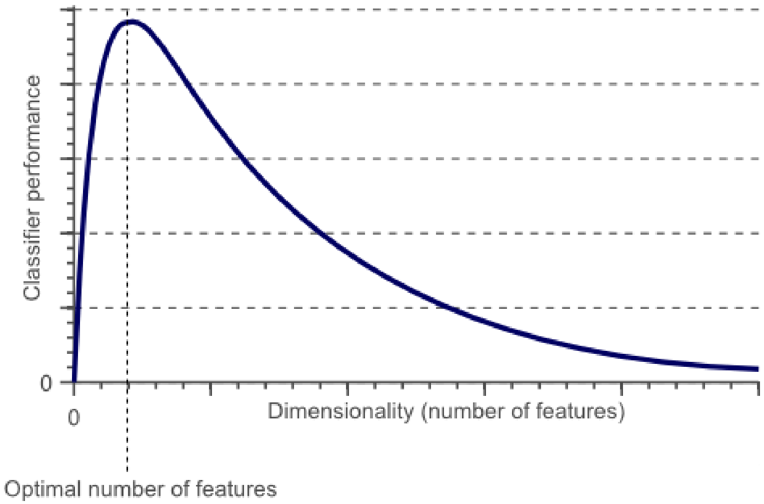


Features	Uniques
1	4
2	4^2
k	4^k

- Suppose we have one feature with 4 different values in one dimension(only one feature in the data set). There are total 4 combinations of feature values.
- Now if we add one more feature, we have 4*4 = 16 combinations of feature values for two dimensions. If we add one more feature to it, the combinations of feature values will increase to 4*4*4 = 64, and so on. As the dimensionality increases, the number of data points required for good performance of machine learning algorithm increases exponentially.
- When dimensionality increases the data becomes too sparse which creates problem.

https://medium.com/analytics-vidhya/the-curse-of-dimensionality-and-its-cure-f9891ab72e5c

Dimensionality vs. Model Performance



- As the dimensionality increases, the classifier's performance increases until the optimal number of features is reached. Further increasing the dimensionality without increasing the number of training samples results in a decrease in classifier performance.
- All the Machine Learning algorithms that rely on calculating distance between observation suffer most from the curse of dimensionality. Such as Segmentation and clustering algorithms like KNN, K-Means, etc.

https://miro.medium.com/max/450/0*fmwS1rDbutO2BWCW.png

Issues

Overfitting

- If the amount of available training data is fixed, then **overfitting** occurs if we keep adding dimensions. On the other hand, if we keep adding dimensions, the amount of training data needs to grow exponentially fast to maintain the same coverage and to avoid overfitting.

Algorithms with low generalization ability

- Neural Networks, KNN classifier, decision trees etc. Those classifiers that tend to model non-linear decision boundaries very accurately, do not generalize well and are prone to overfitting. Therefore, the dimensionality should be kept relatively low when these classifiers are used.
- If a classifier is used that generalizes easily (e.g. **naive Bayesian, linear classifier**), then the number of used features can be higher.

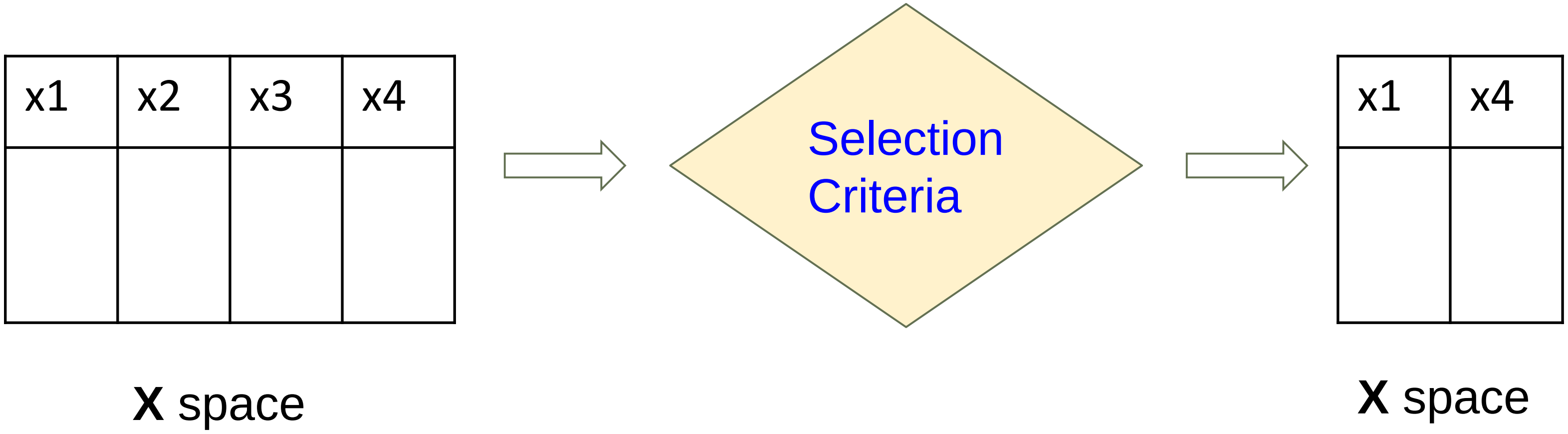
Solutions

- Feature Selection
- Feature Extraction

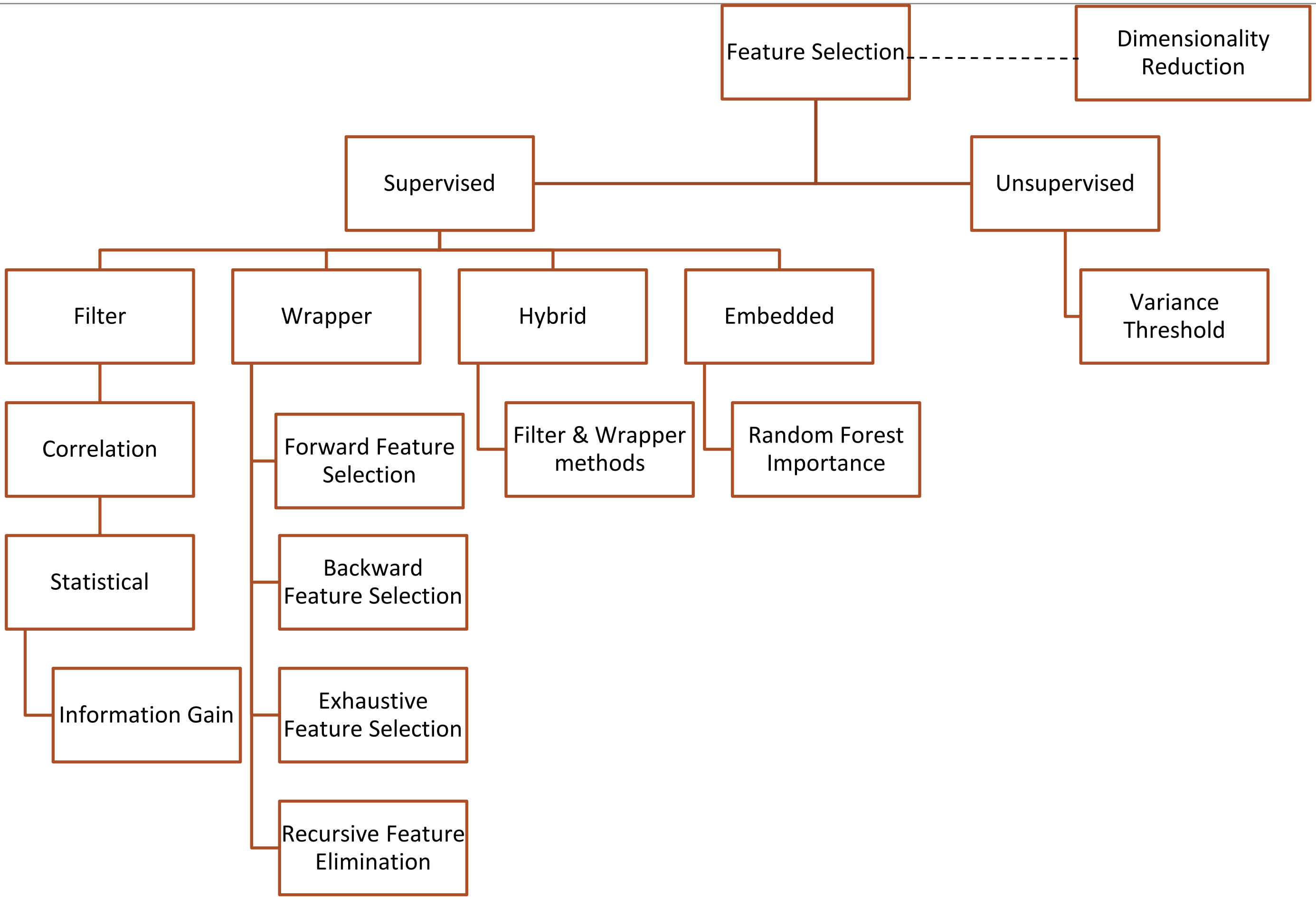
Feature Selection

Feature Selection

Feature selection is about selecting a **subset of features** out of the **original features** in order to reduce model complexity, enhance the computational efficiency of the models and reduce generalization error introduced due to noise by irrelevant features.



Feature Selection Techniques



Variance Threshold

- Constant features show similar/single values in all the observations in the dataset. These features provide no information that allow ML models to predict the target.
- Variance Threshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet the threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all observations.
- It looks only at the features (x), not the desired outputs (y), it can be used as unsupervised learning.
- Default value of threshold is 0
  - If Variance Threshold = 0 (Remove Constant Features)
  - If Variance Threshold > 0 (Remove Quasi-Constant Features)
- Note: Variance Threshold method is only a **baseline method**, it should be used with caution.

Supervised Method

- **Filter methods-** use **statistical techniques** to evaluate the relationship between each input variable and the target variable, and these scores are used as the basis to choose (filter) those input variables that will be used in the model. When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

- **Wrapper methods** – requires some method to search the space of **all possible subsets** of features, assessing their quality by learning and evaluating a classifier with that feature subset. It follows a greedy search approach by evaluating **all the possible combinations** of feature against the evaluation criterion.

- **Hybrid methods** – **Combine Filter and wrapper methods**. The main priority is to select the methods you're going to use, then follow their processes. The idea here is to use these ranking methods to generate a feature ranking list in the first step, then use the top k features from this list to perform wrapper methods.

- **Embedded methods** - **Integrate the feature selection machine learning algorithm as part of the learning algorithm**, in which classification and feature selection are performed simultaneously. The features that will contribute the most to each iteration of the model training process are carefully extracted. Random forest feature selection and decision tree feature selection are common embedded methods.

Filter methods

Statistics for Filter-based Feature Selection Methods

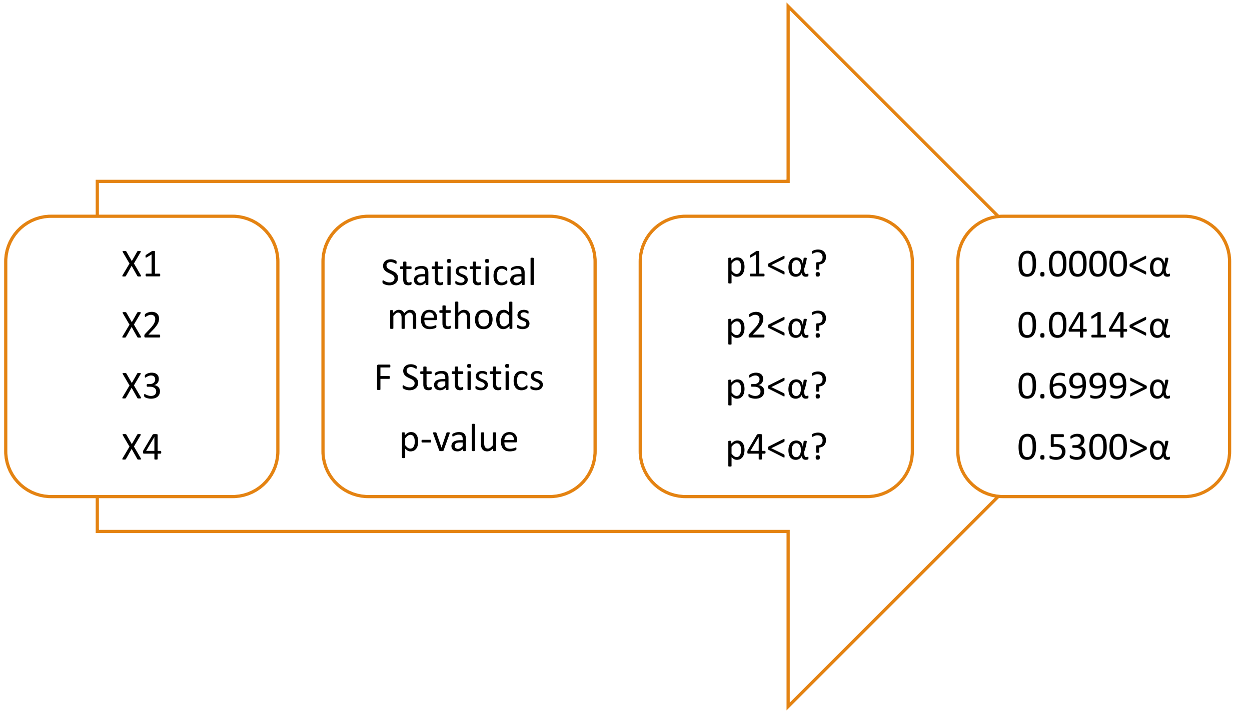
Input Variable	Output Variable	Feature Selection Model	Statistics
Numerical	Numerical	• Pearson's correlation coefficient (linear) • Spearman's rank coefficient (non-linear)	• Corr(Pearson's coefficient) • spearmanr (SciPy) • f_regression()
Numerical	Categorical	• ANOVA correlation coefficient (linear) • Kendall's rank coefficient (nonlinear)	• f_classif() • Chi2() • kendalltau(SciPy)
Categorical	Numerical	• Kendall's rank coefficient (linear) • ANOVA correlation coefficient (nonlinear)	• Very rare
Categorical	Categorical	• Chi-Squared test (contingency tables) • Mutual Information	• Chi2() • Mutual_info_classif()

Univariate Feature Selection

- Univariate feature selection examines each feature individually to determine the strength of the relationship of the feature with the response variable.

- Steps for Univariate Feature Selection
  - Get all independent features
  - Apply relevant statistical method
  - Get p-value and compare with the significance level
  - Select the feature if $p<\alpha$

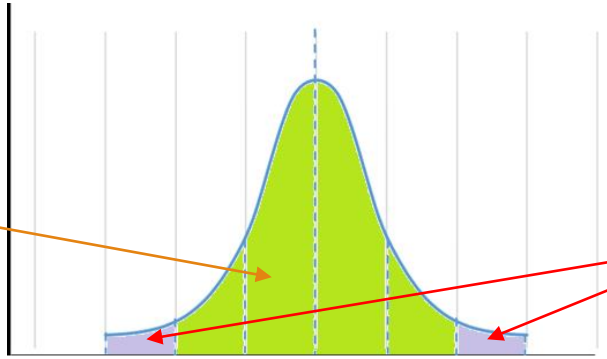
Univariate Feature Selection



Statistical Significance

H_0 The feature(predictor) has no impact on dependent variable (Y)

H_1 The feature(predictor) has significant impact on dependent variable (Y)



H_0

- We expect this to happen
- Null hypothesis is true with 100(1- α)% probability

- Typically α = 0.05

H_1

- We don’t expect this to happen
- Null hypothesis is rejected
- Rejection region
- p<α

Selection Method

- Univariate feature selection works by selecting the best features based on univariate statistical tests.

- Selection Methods
  - `SelectKBest` removes all but the k highest scoring features
  - `SelectPercentile` removes all but a user-specified highest scoring percentage of features
  - `GenericUnivariateSelect` allows to perform univariate feature selection with a configurable strategy.

- Statistical tests:
  - For regression: `f_regression`, `mutual_info_regression`
  - For classification: `chi2`, `f_classif`, `mutual_info_classif`

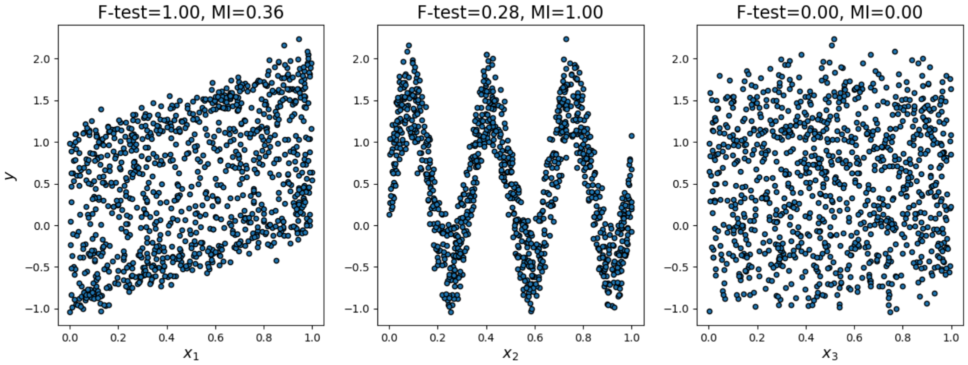
- F-test estimate the degree of linear dependency between two random variables while mutual information methods can capture any kind of statistical dependency.

- chi2 – Compute chi-squared stats between each non-negative feature and class. Cateogirical -> categorical

- f_classif - Compute the ANOVA F-value. Numeric -> categorical

F-test vs Mutual Information

- Select features based on F-test or Mutual Information
  - F-test captures only the linear dependency
  - Mutual information can capture any kind of dependency between variables.



- For the above graph, based on F-test, x1 is the most discriminative feature, based on MI, x2 is the most discriminative feature. Both methods marks x3 as irrelevant.

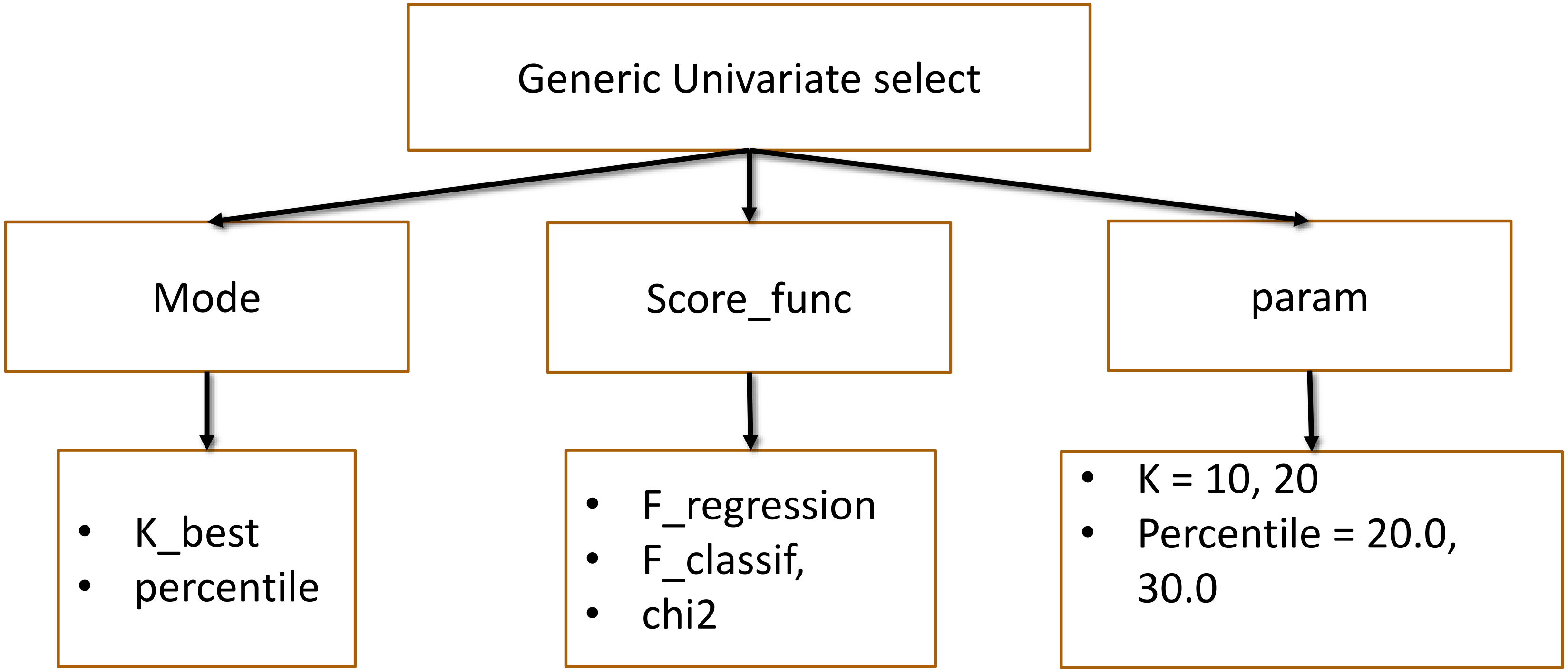
Source : https://scikit-learn.org/stable/modules/feature_selection.html

Example

```
>>> from sklearn.feature_selection import SelectKBest
>>> from sklearn.feature_selection import chi2
>>> X, y = load_iris(return_X_y=True)
>>> X.shape
(150, 4)
>>> X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
>>> X_new.shape]
(150, 2)
```

https://scikit-learn.org/stable/modules/feature_selection.html

Generic Univariate Select



Generic Univariate select

Mode

- K_best
- percentile

Score_func

- F_regression
- F_classif,
- chi2

param

- K = 10, 20
- Percentile = 20.0, 30.0

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.GenericUnivariateSelect.html

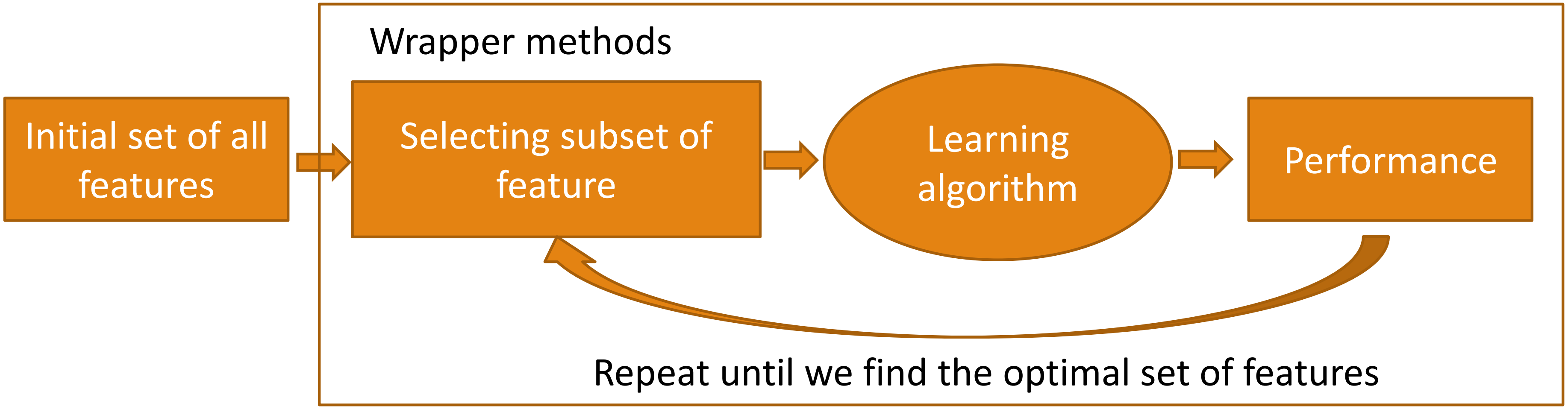
Example

```
>>> from sklearn.datasets import load_breast_cancer
>>> from sklearn.feature_selection import
GenericUnivariateSelect, chi2
>>> X, y = load_breast_cancer(return_X_y=True)
>>> X.shape
•   (569, 30)
>>> transformer = GenericUnivariateSelect(chi2, mode='k_best',
param=20)
>>> X_new = transformer.fit_transform(X, y)
>>> X_new.shape
(569, 20)
```

Wrapper methods

Wrapper Methods

- In wrapper methods, the feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset.
- It follows a **greedy search approach** by evaluating **all the possible combinations of features against the evaluation criterion**. The evaluation criterion is simply the performance measure which depends on the type of problem, for e.g. For regression evaluation criterion can be p-values, R-squared, Adjusted R-squared, similarly for classification the evaluation criterion can be accuracy, precision, recall, f1-score, etc. Finally, it selects the combination of features that gives the optimal results for the specified machine learning algorithm.


Wrapper methods
Initial set of all features
Selecting subset of feature
Learning algorithm
Performance
Repeat until we find the optimal set of features

Common Wrapper Methods

- Forward selection
- Backward elimination
- Exhaustive feature selection
- Recursive feature elimination
- Recursive feature elimination with cross-validation

Forward Selection

- The procedure starts with an **empty set of features**. In the first step, it evaluates all features individually and selects the one that generates the **best performance**, according to a pre-set evaluation criteria. In the second step, it evaluates all possible combinations of the selected feature and a second feature, and selects the pair that produce the best performing algorithm based on the same pre-set criteria. In each iteration, we keep adding the feature which best improves our model until an addition of a new variable does not improve the performance of the model.

Backward Selection

- In backward elimination, we start with **all the features** and removes the **least significant feature** at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.

Exhaustive feature selection

- In an exhaustive feature selection, the best subset of features is selected, over all possible feature subsets, by optimizing a specified performance metric for a certain machine learning algorithm.

Recursive feature elimination

- Recursive feature elimination performs a **greedy search** to find the best performing feature subset. It iteratively creates models and determines the best or the worst performing feature at each iteration. It constructs the subsequent models with the left features until all the features are explored. It then ranks the features based on the order of their elimination.

Recursive Feature Elimination with cross-Validation

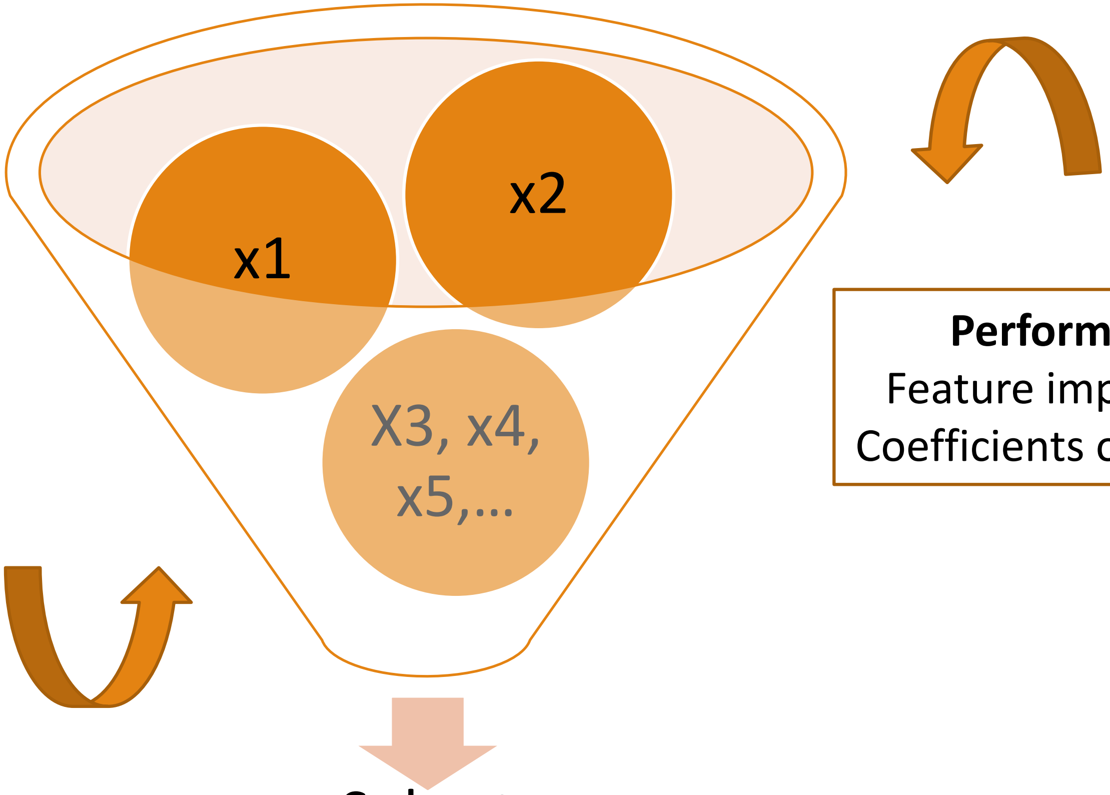
- Recursive Feature Elimination with Cross-Validated (RFECV) feature selection technique selects the best subset of features for the estimator by removing 0 to N features iteratively using recursive feature elimination.

- Then it selects the best subset based on the accuracy or **cross-validation** score or roc-auc of the model. Recursive feature elimination technique eliminates n features from a model by fitting the model multiple times and at each step, removing the weakest features.

Recursive Feature Elimination (RFE)

- There are two important configuration options when using RFE:
  - The choice in the number of features to select
  - The choice of the algorithm used to help choose features.
- RFE is a wrapper-type feature selection algorithm. Different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features.
- RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the **desired number** remains.
- This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.

Recursive Feature Elimination



Learning algorithms
Classification
regression

Performance
Feature importance
Coefficients or weights

Subset

X1, X2, X4

RFE in scikit-learn

- `sklearn.feature_selection.RFE(estimator, n_features_to_select=None, step=1, verbose=0)`
- `estimator`: a supervised learning estimator with a fit method that provides information about feature importance either through `coef_` `attribute` or through a `feature_importances_ attribute`
- `n_features_to_select`: int or None (default=None). The number of features to select. If None, half of the feature are selected.
- `step`: int or float, optional (default=1). The step corresponds to the number of features to remove at each iteration.
- `verbose`: int, (default=0). Verbose is a general programming term for produce lots of logging output. if you enable verbose > 0, that printing to the screen is generally a very slow process.

RFE Example

```
from sklearn.svm import SVC
from sklearn.datasets import load_digits
from sklearn.feature_selection import RFE
import matplotlib.pyplot as plt

# Load the digits dataset
digits = load_digits()
X = digits.images.reshape((len(digits.images), -1))
y = digits.target

# Create the RFE object and rank each pixel
svc = SVC(kernel="linear", C=1)
rfe = RFE(estimator=svc, n_features_to_select=1, step=1)
rfe.fit(X, y)
ranking = rfe.ranking_.reshape(digits.images[0].shape)

# Plot pixel ranking
plt.matshow(ranking, cmap=plt.cm.Blues)
plt.colorbar()
plt.title("Ranking of pixels with RFE")
plt.show()
```

https://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_digits.html#sphx-glr-auto-examples-feature-selection-plot-rfe-digits-py

Embedded Methods

Embedded Methods

- Using machine learning to select features:
  - The two main machine learning models for the purposes of feature selection are tree-based models and linear models.
  - They both have a notion of feature ranking that are useful when sub-setting feature sets.
  - **Tree-based model feature selection metrics:**
    - When fitting decision trees, the tree starts at the root node and greedily chooses the optimal split at every junction that optimizes a certain metric of node purity.
    - While each split is created, the model keeps track of how much each split helps the overall optimization goal.
    - In doing so, tree-based models that choose splits based on such metrics have a notion of feature importance.

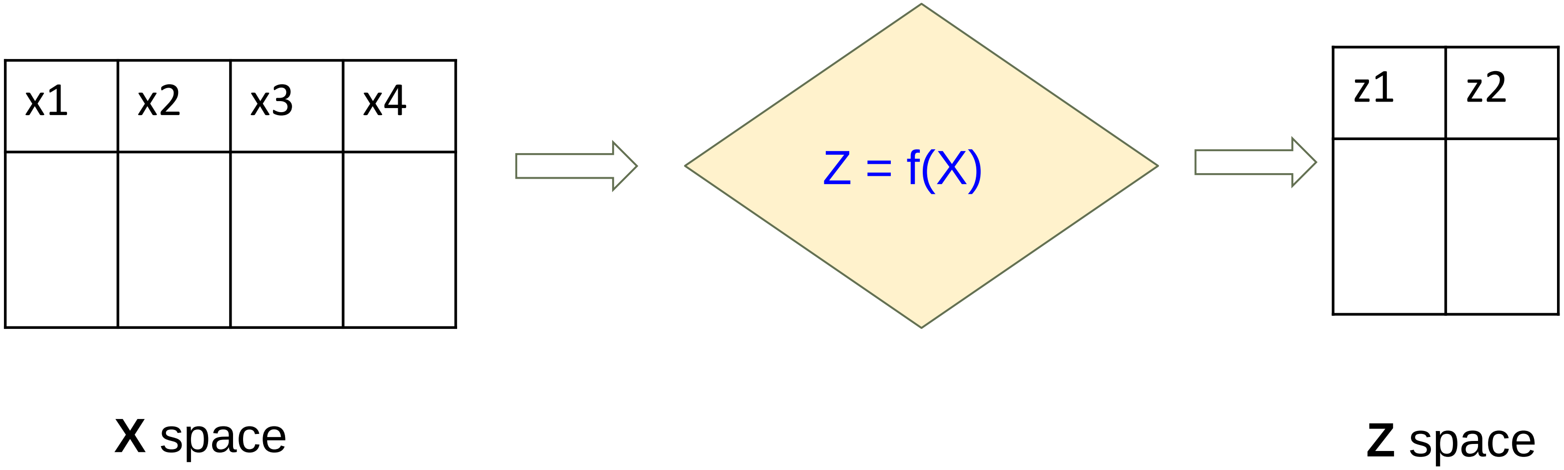
Comparison of Different Methods

	Filter methods	Wrapper methods	Embedded methods
Machine learning algorithm	Generic set of methods which do not incorporate a specific machine learning algorithm	Evaluates on a specific machine learning algorithm to find optimal features	Embeds(fix) features during model building process.
Computation time	Much faster	High computation time	Medium
Over-fitting	Less prone to over-fitting	High chances of over-fitting. Because it involves training of machine learning models with different combination of features	Generally reduce over-fitting by penalizing the coefficients of a model being too large

Feature Extraction

Feature Extraction

- Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations (Wiki).



x1	x2	x3	x4

Z = f(X)

z1	z2

X space

Z space

Feature Extraction

- **Why Feature Extraction is Useful?** - The technique of extracting the features is useful when you have a large data set and need to reduce the number of resources without losing any important or relevant information. Feature extraction helps to reduce the amount of redundant data from the data set.
- **Applications of Feature Extraction**
  - **Bag of Words-** Bag-of-Words is the most used technique for natural language processing. In this process they extract the words or the features from a sentence, document, website, etc. and then they classify them into the frequency of use. So in this whole process feature extraction is one of the most important parts.
  - **Image Processing** –Image processing is one of the best and most interesting domain. In this domain basically you will start playing with your images in order to understand them. So here we use many techniques which includes feature extraction as well and algorithms to detect features such as shaped, edges, or motion in a digital image or video to process them.

Feature Extraction Techniques

- **PCA** - PCA is a method of obtaining important variables (in form of components) from a large set of variables available in a data set. It tends to find the direction of maximum variation (spread) in data.

- **Kernel PCA** - PCA performs linear operations to create new features. PCA fails when the data is non-linear and is not able to create the hyperplane. Similar to SVM, Kernel PCA implements **Kernel–Trick** to convert the non-linear data into a higher dimension where it is separable.

- Others – LDA(Linear Discriminant Analysis), t-SNE, etc.

Principal Component Analysis

- Principal Component Analysis (PCA)
  - PCA is a **dimensionality-reduction** method that is often used to reduce the dimensionality of large data sets, by converting a huge quantity of variables into a small one and keeping most of the information preserved.

- PCA is such a **feature reduction** method where we create new uncorrelated features from the old features and select a subset (using a criteria known as ‘eigen value > 1’) of new features (called Principal Components) which are most important to explain the variability in the raw features.

PCA Pros and Cons

- Pros
  - Lack of redundancy of data
  - Principal components are uncorrelated with each other, and hence helps to eliminate multi-collinearity.
  - PCA helps in overcoming computational infeasibility issues by decreasing the number of features (refer to “Curse of Dimensionality”).
- Cons
  - Though PCA explains the maximum variance of the raw data features, sometimes it may skip a bit of information in comparison to the actual list of features as it selects the PCs with eigen value > 1 and eliminates the rest.
  - Implementing PCA over datasets leads to transforming raw features in principal components that are linear combinations of all raw features. So principle components are difficult to read or interpret as compared to raw features unless one takes help of a mathematical operation known as rotation.

PCA Steps

Step 1: Standardization or scaling

- The aim of this step is to standardize the range of the continuous initial raw variables so that each one of them contributes equally to the analysis.
- PCA is quite sensitive to the variances of the initial raw variables. If there are large differences between the ranges of initial raw variables, the variables with larger ranges will dominate over those with smaller ranges (e.g. a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results.

$$x\_scaled = \frac{X - mean(x)}{std(x)}$$

PCA Steps

Step 2: Covariance matrix computation

- The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them.
- Covariance matrix
- $$\begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}$$
- Covariance:
- $cov(x,y) = \frac{\sum(x-\bar{x})*(y-\bar{y})}{(n-1)}$

symmetric with respect to the main diagonal

- Sign of the covariance
  - if positive then : the two variables increase or decrease together (correlated)
  - if negative then : One increases when the other decreases (Inversely correlated)

PCA Steps

Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix:

- By ranking principle components in order of their eigenvalues, highest to lowest, we get the principal components in the order of significance (in terms of explaining the variability).

Step 4: Feature Vector

- In this step, we choose whether to keep all these components or discard those of lesser significance (of eigenvalues < 1), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

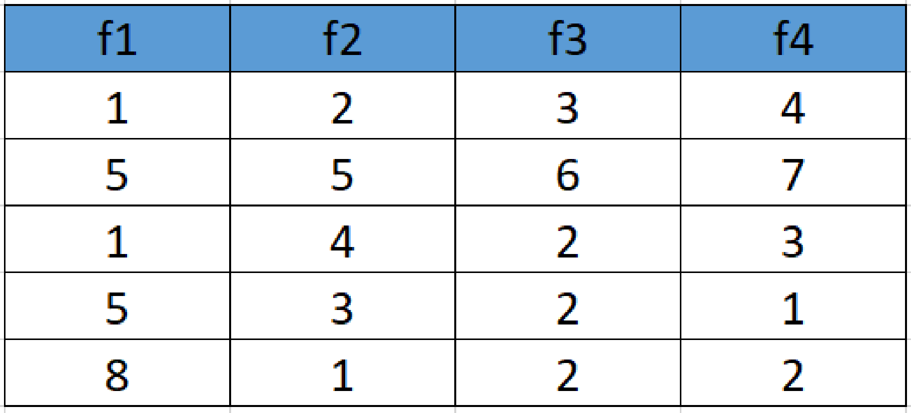
Step 5: Change base of the data along the principle component axes:

- $FinalDataSet = FeatureVector^T \times Standardized\ original\ Dataset^T$

PCA Algorithm

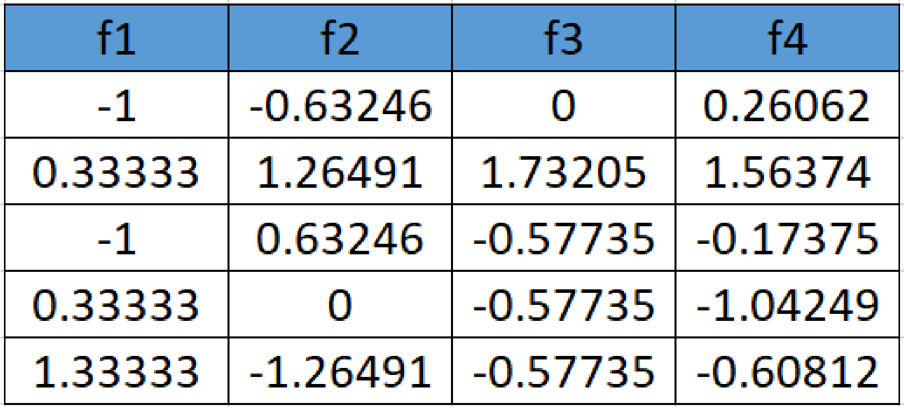
- Standardize the dataset – normalization
- Compute the covariance matrix using the centered data
- Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
- Take top K leading eigenvectors with largest eigenvalues
- Project the original data point to the K-dimensional new basis

1 Standardize the Dataset



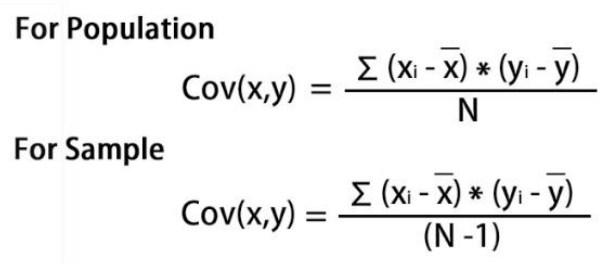
f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

$$x_{new} = \frac{x - \mu}{\sigma}$$



f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

2. Calculate the covariance matrix of the whole dataset

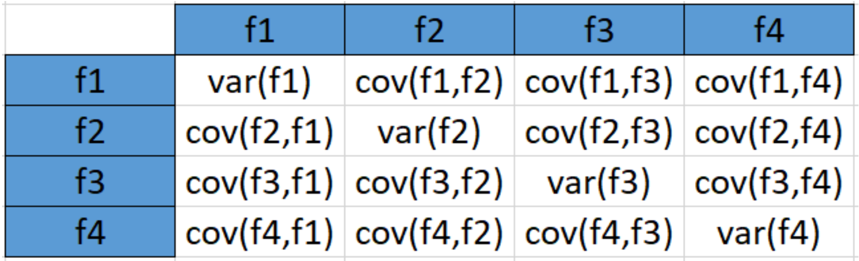


For Population

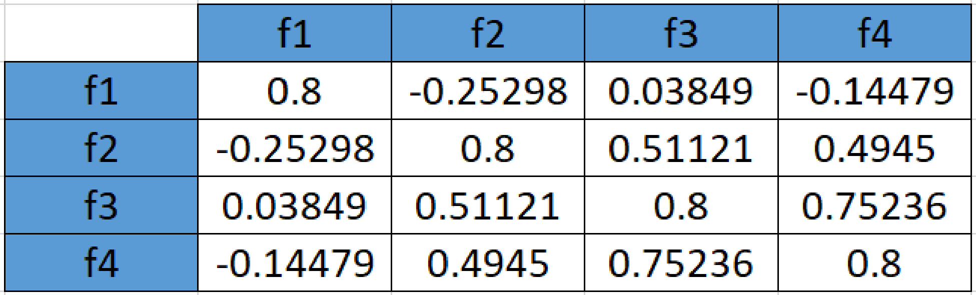
$$\mathrm{Cov}(x,y) = \frac{\Sigma\,(x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

For Sample

$$\mathrm{Cov}(x,y) = \frac{\Sigma\,(x_i - \bar{x}) * (y_i - \bar{y})}{(N - 1)}$$



	f1	f2	f3	f4
f1	var(f1)	cov(f1,f2)	cov(f1,f3)	cov(f1,f4)
f2	cov(f2,f1)	var(f2)	cov(f2,f3)	cov(f2,f4)
f3	cov(f3,f1)	cov(f3,f2)	var(f3)	cov(f3,f4)
f4	cov(f4,f1)	cov(f4,f2)	cov(f4,f3)	var(f4)



	f1	f2	f3	f4
f1	0.8	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8	0.51121	0.4945
f3	0.03849	0.51121	0.8	0.75236
f4	-0.14479	0.4945	0.75236	0.8

3 Calculate eigenvalue and eigenvectors

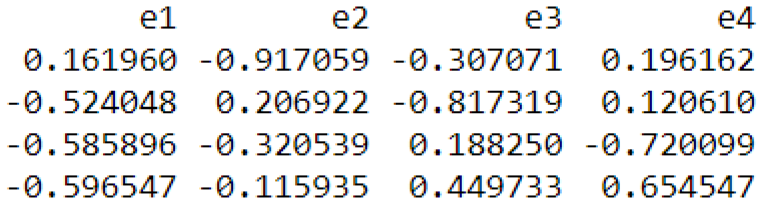
- Let **A** be a square matrix (in our case the covariance matrix), **v** is a vector and **λ** is a scalar that satisfies **Av = λv**, then **λ** is called eigenvalue associated with eigenvector **v** of **A**.

	f1	f2	f3	f4
f1	0.8 - λ	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8- λ	0.51121	0.4945
f3	0.03849	0.51121	0.8 - λ	0.75236
f4	-0.14479	0.4945	0.75236	0.8 - λ

Solving the equation det(A-λI) = 0, the eigenvalues of matrix **A** are:

λ = 2.51579324 , 1.0652885 , 0.39388704 , 0.02503121

And the corresponding eigenvectors are:



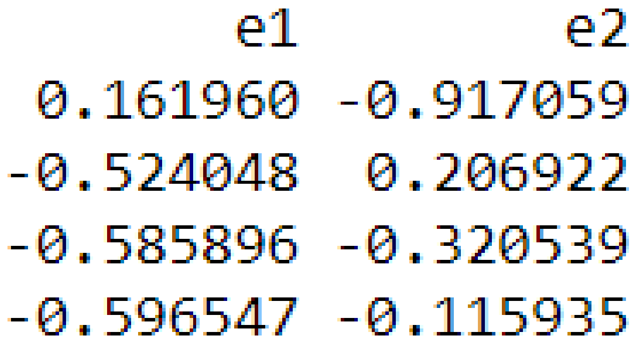
```
       e1        e2        e3        e4
 0.161960 -0.917059 -0.307071  0.196162
-0.524048  0.206922 -0.817319  0.120610
-0.585896 -0.320539  0.188250 -0.720099
-0.596547 -0.115935  0.449733  0.654547
```

4 Sort eigenvalue and their corresponding eigenvectors

- ***λ1> λ2> λ3> λ4***
- **e1->e2-**>e3->e4

5 Pick k eigenvalues and form a matrix of eigenvectors

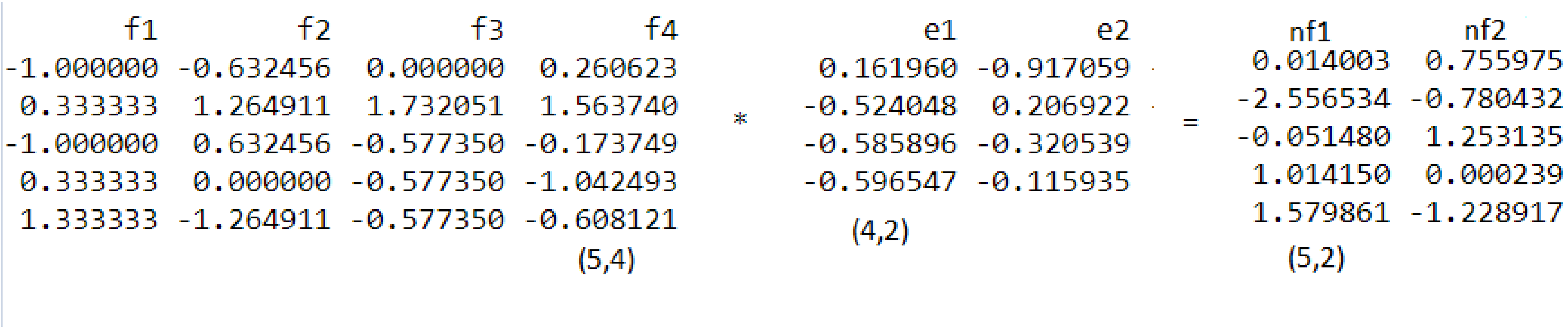
- If we choose the top 2 eigenvectors, the matrix will look like this:



```
       e1        e2
 0.161960 -0.917059
-0.524048  0.206922
-0.585896 -0.320539
-0.596547 -0.115935
```

6. Transform the original matrix

- Feature matrix * top k eigenvectors = Transformed Data



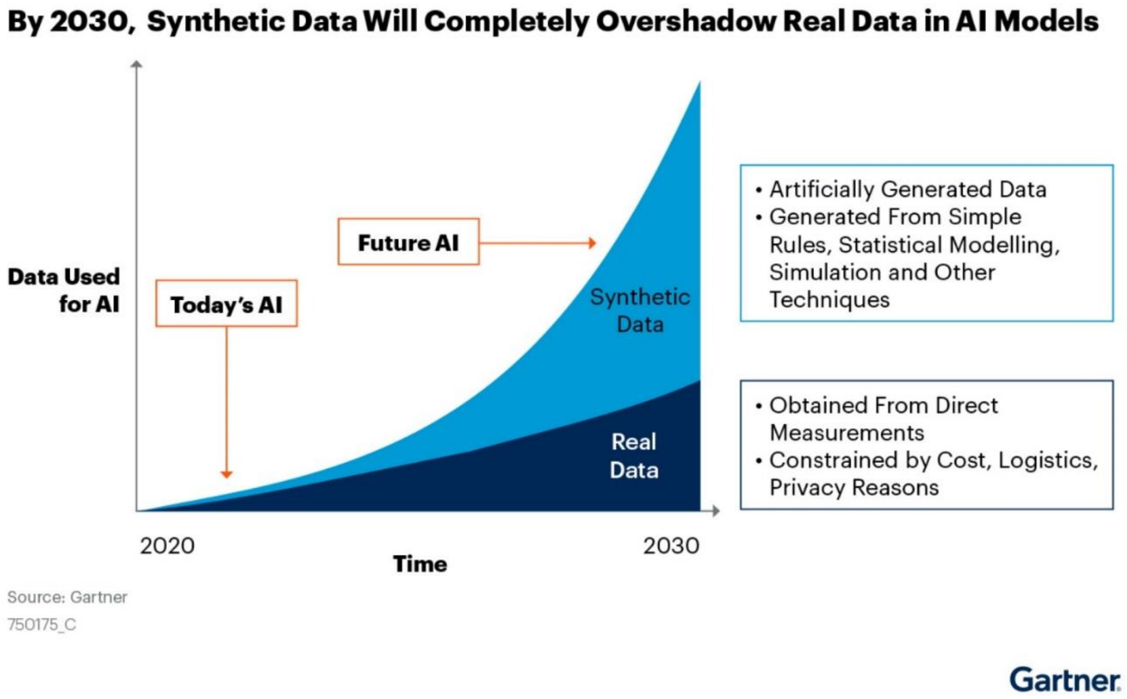
```
       f1        f2        f3        f4                e1        e2        nf1       nf2
-1.000000 -0.632456  0.000000  0.260623          0.161960 -0.917059     0.014003  0.755975
 0.333333  1.264911  1.732051  1.563740    *    -0.524048  0.206922  = -2.556534 -0.780432
-1.000000  0.632456 -0.577350 -0.173749         -0.585896 -0.320539    -0.051480  1.253135
 0.333333  0.000000 -0.577350 -1.042493         -0.596547 -0.115935     1.014150  0.000239
 1.333333 -1.264911 -0.577350 -0.608121                                 1.579861 -1.228917
                                 (5,4)               (4,2)                 (5,2)
```

- 4D reduced to 2D

Synthetic data generation

Synthetic Data Generation

- Synthetic data is annotated information that computer simulations or algorithms generate as an alternative to real-world data.



https://blogs.nvidia.com/wp-content/uploads/2021/06/Gartner-chart.jpg

Importance of Synthetic Data

Synthetic data is important because **it can be generated to meet specific needs or conditions that are not available in existing (real) data**. This can be useful in numerous cases such as

- **Privacy-enhancing technology**. When privacy requirements, limit data availability or how it can be used
- **Product testing**. Data is needed for testing a product to be released however such data either does not exist or is not available to the testers
- **Training ML model**. Training data is needed for ML algorithms. However, especially in the case of self-driving cars, such data is expensive to generate in real life
- **Dataset is too small for robust model building.**

Synthetic Data generation Techniques

- Generating according to the distribution
  - For cases where real data does not exist but data analyst has a comprehensive understanding of how dataset distribution would look like, the analyst can generate a random sample of any distribution such as Normal, Exponential, Chi-square, t, lognormal and Uniform etc.

- Fitting real data to a known distribution
  - If there exists ~~a~~ real-data, then businesses can generate synthetic data by determining the best fit distributions for given real-data. If businesses want to fit real-data into a known distribution and they know the distribution parameters, businesses can use Monte Carlo simulation method to generate synthetic data.

- Using deep learning methods
  - Deep generative models such as Variational Autoencoder(VAE) and Generative Adversarial Network (GAN) (for audio, video & image data) can generate synthetic data.

Generate Synthetic Data in Python

- Scikit-learn is one of the most widely-used Python libraries for machine learning tasks and it can also be used to generate synthetic data. One can generate data that can be used for regression, classification, or clustering tasks.
- SymPy is another library that helps users to generate synthetic data. Users can specify the symbolic expressions for the data they want to create, which helps users to create synthetic data according to their needs.
- Pydbgen: Categorical data can also be generated using Python's Pydbgen library. Users can generate random names, international phone numbers, email addresses etc.

Best Practice for Data Synthetic

- **Work with clean data**. Make sure you apply the following principles:
  - Data cleaning
  - Data aggregation: For example, same attributes from different sources need to be mapped to the same column
- Assess whether synthetic data is similar enough to real data for its application area
  - The utility of synthetic varies depending on the technique you use while generating it. You need to analyze their use case and decide if the generated synthetic data is a good fit the specific use case.
- Outsource support if necessary
  - Identify your organization's synthetic data capabilities and outsource based on the capability gaps. The TWO important steps are data preparation and data synthesis. Both steps can be automated by suppliers.

Summary

- Feature engineering is the process of formulating the most appropriate features given the data and the model.
- Features and models sit between raw data and the desired insight.
- Good features make the subsequent modeling step easy and the resulting model more capable of achieving the desired task. Bad features may require a much more complicated model to achieve the same level of performance.
- Feature engineering is a important stage in machine learning lifecycle. It includes handling missing value, categorical encoding techniques, feature transformation and feature scaling.
- Feature selection reduces the number of input variables to both reduce the computational cost of modeling and to improve the performance of the model.

References

- https://scikit-learn.org/stable/modules/feature_selection.html
- https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/sample_syntheticdata.html
- https://docs.opencv.org/4.5.2/d1/dee/tutorial_introduction_to_pca.html
- https://www.analyticssteps.com/blogs/introduction-principal-component-analysis-machine-learning
- https://research.aimultiple.com/synthetic-data/
- https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/
- https://en.wikipedia.org/wiki/Principal_component_analysis
- https://builtin.com/data-science/step-step-explanation-principal-component-analysis

APPENDIX

MISSING DATA IMPUTATION

Missing Data Imputation Methods

Numerical Variables	Categorical variables	Both
Mean/median imputation	Frequent category imputation	List-wise deletion
Special value imputation	Adding a “missing” category	Adding missing indicator
		Random sample imputation

- Mean/median imputation: replacing all occurrences of missing values within a variable by the mean or median.
- Special value imputation: replacing all occurrences of missing values within a variable by an extreme value.
- Frequent category imputation: replacing all occurrences of missing values within a variable by the mode, or the most frequent value.
- Adding a “missing” category: missing observations are grouped in the newly created category label “Missing”.
- Adding missing indicator : Similar to special value imputation/adding missing category
- List-wise deletion : Entire record is eliminated if any single variable value is missing.
- Random sample imputation
  - Take a random observation from the pool of available observations of the variables, and use that randomly extracted value to fill the NA.

Mean / Median Imputation

- Mean/median imputation: replacing all occurrences of missing values within a variable by the mean or median.
- Pros
  - Easy to implement
  - Fast way of obtaining complete datasets
  - Can be integrated in production environment
- Cons
  - Distortion of the original variable distribution
- Limitations
  - Data is missing at random
  - Only small number of data is missing (5%)

Special Value Imputation

- Special value imputation: replacing all occurrences of missing values within a variable by an extreme value. Typically values are 0,999,9999, -999, -9999 when data is not missing at random but due to a specific reason (e.g. not reported deliberately), we want to flag the missing values with a different (special extreme value).
- Pros
  - Easy to implement
  - Fast
  - Can be integrated into production environment
  - Capture the importance of being “missing”
- Cons
  - Distortion of the original data.
  - May create outliers
  - To be careful to use this method

Frequent Category Imputation

- Frequent category imputation : replacing all occurrences of missing values within a variable by the mode, or the most frequent value.
- Pros
  - Easy to implement
  - Fast way of obtaining complete datasets
  - Can be integrated in production environment
- Cons
  - Distorts the relation of the most frequent label with other variables within the dataset
  - May lead to an over-representation of the most frequent label if there is a big number of 'NA'.
- Limitation
  - Data is missing at random
  - Only small number of data is missing (5%)

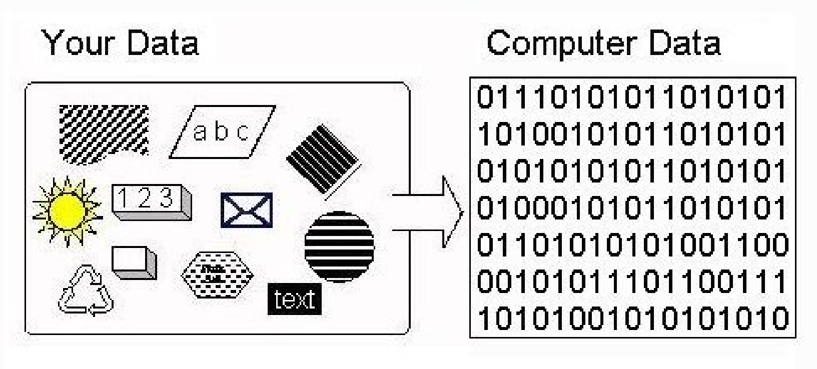
Adding A Missing Category

- Adding a missing category: missing observations are grouped in the newly created category label “Missing”.
- This is a widely used method for categorical variables.
- Pros
  - Easy to implement
  - Fast way of obtaining complete datasets
  - Can be integrated in production environment
  - Captures the importance of missing category
  - No assumption made on data

Categorical data encoding

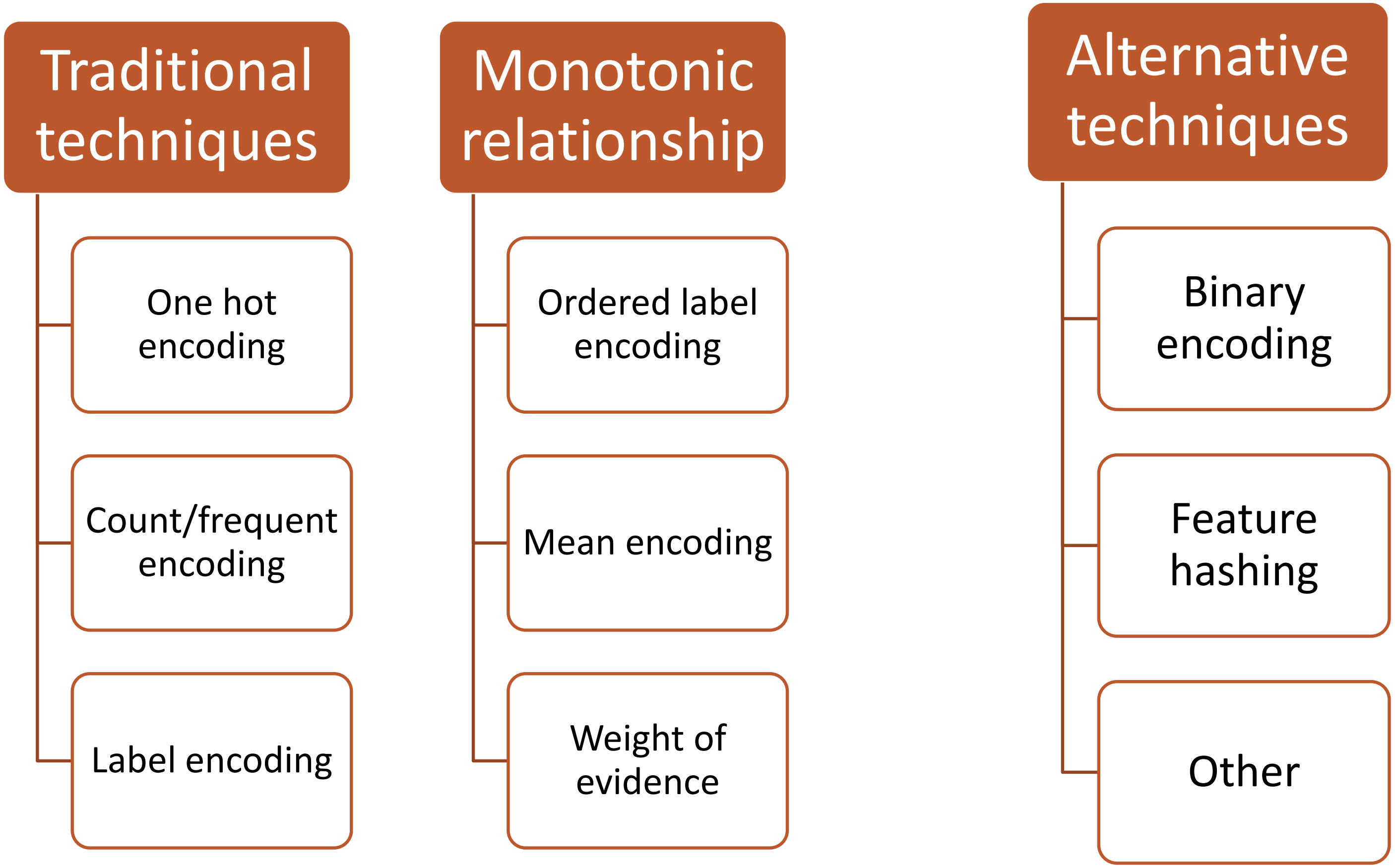
Categorical Encoding

- Categorical encoding is a process of converting categories to numbers
- Typically, any structured dataset includes multiple columns – a combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text.
- Categorical encoding techniques
  - Label Encode
  - One-Hot encode



https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/

Categorical Encoding Techniques

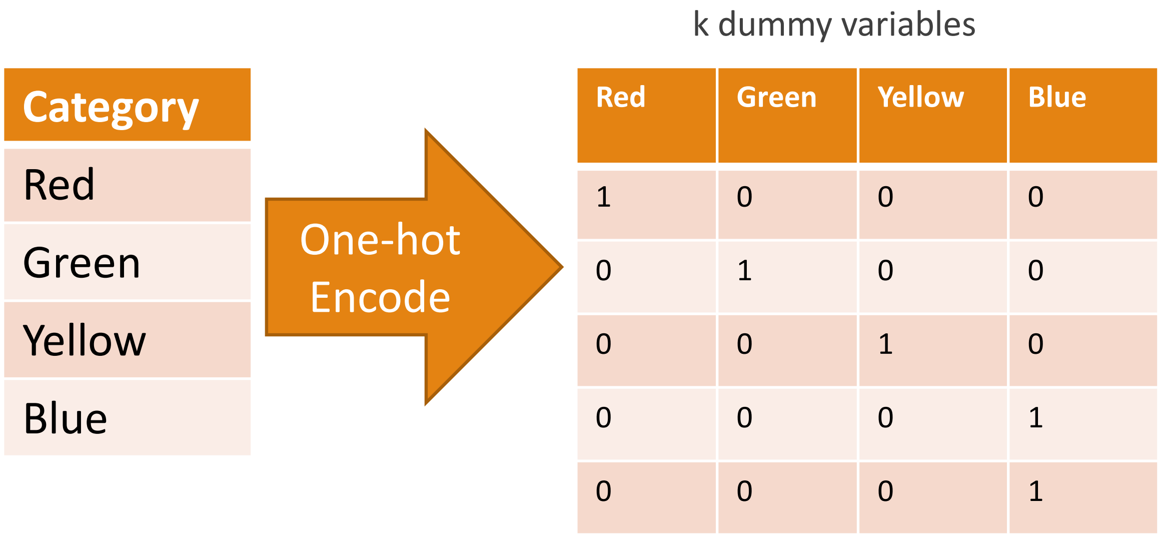


- Traditional techniques
  - One hot encoding
  - Count/frequent encoding
  - Label encoding
- Monotonic relationship
  - Ordered label encoding
  - Mean encoding
  - Weight of evidence
- Alternative techniques
  - Binary encoding
  - Feature hashing
  - Other

One-Hot Encode

One-Hot encode is a representation of categorical variables as binary vectors.

It consists in encoding each categorical variable with a set of Boolean variables which take values 0 or 1, indicating if a category is present for each observation.



Category
Red
Green
Yellow
Blue

One-hot Encode

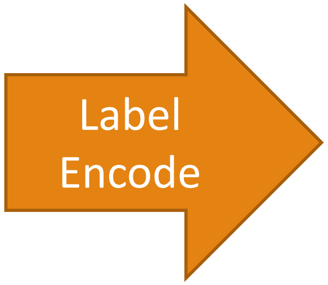
k dummy variables

Red	Green	Yellow	Blue
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	1

Label Encode

- Label Encode is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on ordering.
- It replaces the categories by whole number from 1 to n (or 0 to n-1),where n is the number of distinct categories of the variables.

Colour
Red
Green
Yellow
Blue
Blue


Label Encode

LabelEncode
1
2
3
4
4

Label Encode vs. One-hot Encode

- Label encode
  - Preserves feature dimension
  - Implicit bias in numerical value assigned. The numeric values can be misinterpreted by algorithms as having some sort of hierarchy/order in them. E.g. red = 1, green=2, so red < green?
  - Can handle out-of-range test values
  - The categorical feature is ordinal (like primary school, high school)

- One-hot encode
  - Expands dimensionality, especially you have many unique values in a category column
  - Less bias, each is its own feature
  - Cannot handle out-of-range test values
  - The categorical feature is not ordinal(like colour, countries)

Feature Transformation

- Feature transformation (FT) refers to family of algorithms that create **new features** using the existing features
- Feature transformation/scaling techniques:
  - MinMax Scaler
  - Standard Scaler
  - MaxAbsScaler
  - Robust Scaler
  - Quantile Transformer Scaler
  - Log Transformation
  - Power Transformer Scaler
  - Unit Vector Scaler/Normalizer

Feature Scaling : Rationale

The regression coefficient is directly influenced by the scale of the variable

Variables with bigger magnitude/value dominate over the ones with smaller magnitude

Gradient descent converges faster when features are on small scales

Euclidean distances are sensitive to feature magnitude

Machine learning algorithms sensitive to magnitude:

- Neural networks
- SVM
- KNN
- Kmeans
- Principal Component Analysis

Machine learning algorithms, insensitive to magnitude are the ones based on Trees where the nodes essentially have counts:

- Classification and regression Trees
- Random forest
- Gradient Boosted Trees

Feature scaling methods

- Standardization or Normalization
- Scale to maximum and minimum
- Scale to absolute maximum
- Robust scaling
- Etc..

Common Scaling methods

- Z-score $$x\_scaled = \frac{X - mean(x)}{std(x)}$$

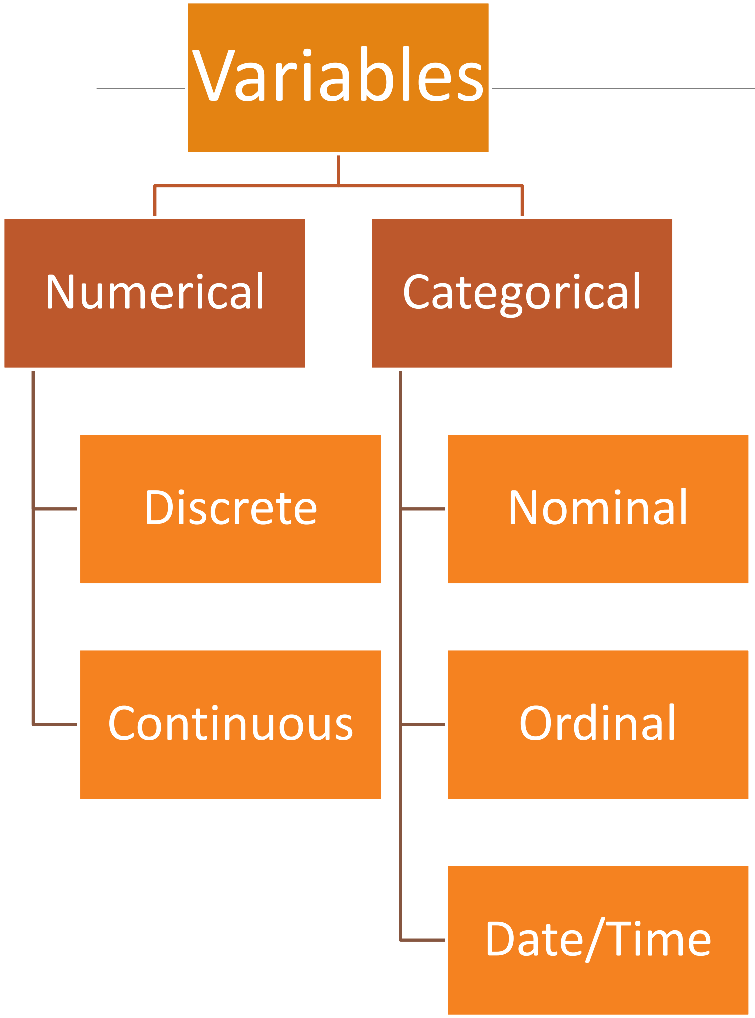
- Mean $$x\_scaled = \frac{X - mean(x)}{max(x) - \min(x)}$$

- MinMax $$x\_scaled = \frac{X - min(x)}{max(x) - \min(x)}$$

- MaxAbsScaling $$x\_scaled = \frac{X}{\max(|X|)}$$

- Robust Scaling $$x\_scaled = \frac{X - median(X)}{75th\ quant(X) - 25th\ quant(X)}$$

Variable Types (Recap)



- The values of a categorical variable are selected from a group of categories, also called labels.
- Examples:
  - Marital status (married, single,…)
  - Gender (male, female)
  - Telecomm provider(Singtel, M1, Starhub, …)

- Nominal Variables – show no intrinsic order of the labels.
- Examples:
  - Country
  - Postcode
  - Subject
  - Home ownership

- Ordinal Variables – categorical variables in which categories can be meaningfully ordered are called ordinal.
- Examples:
  - Student grade
  - Educational level

THE END