Data Preprocessing Steps

Tushar B. Kute,
http://tusharkute.com

Statistical Data Analysis

- Statistical data analysis is a procedure of performing various statistical operations.
- It is a kind of quantitative research, which seeks to quantify the data, and typically, applies some form of statistical analysis.
- Quantitative data basically involves descriptive data, such as survey data and observational data.

Qualitative Data Type

- Qualitative or Categorical Data describes the object under consideration using a finite set of discrete classes.
- It means that this type of data can’t be counted or measured easily using numbers and therefore divided into categories.
- The gender of a person (male, female, or others) is a good example of this data type.

Qualitative Data Type

- These are usually extracted from audio, images, or text medium.
- Another example can be of a smartphone brand that provides information about the current rating, the color of the phone, category of the phone, and so on.
- All this information can be categorized as Qualitative data. There are two subcategories under this:
  – Nominal
  – Ordinal

Nominal

- These are the set of values that don’t possess a natural ordering.
- Example: The color of a smartphone can be considered as a nominal data type as we can’t compare one color with others.
- It is not possible to state that ‘Red’ is greater than ‘Blue’.
- The gender of a person is another one where we can’t differentiate between male, female, or others.
- Mobile phone categories whether it is midrange, budget segment, or premium smartphone is also nominal data type.

Ordinal

- These types of values have a natural ordering while maintaining their class of values.
- If we consider the size of a clothing brand then we can easily sort them according to their name tag in the order of small < medium < large.
- The grading system while marking candidates in a test can also be considered as an ordinal data type where A+ is definitely better than B grade.

Ordinal

- These categories help us deciding which encoding strategy can be applied to which type of data.
- Data encoding for Qualitative data is important because machine learning models can’t handle these values directly and needed to be converted to numerical types as the models are mathematical in nature.
- For nominal data type where there is no comparison among the categories, one-hot encoding can be applied which is similar to binary coding considering there are in less number and for the ordinal data type, label encoding can be applied which is a form of integer encoding.

Quantitative Data Type

- This data type tries to quantify things and it does by considering numerical values that make it countable in nature.
- The price of a smartphone, discount offered, number of ratings on a product, the frequency of processor of a smartphone, or ram of that particular phone, all these things fall under the category of Quantitative data types.

Quantitative Data Type

- The key thing is that there can be an infinite number of values a feature can take.
- For instance, the price of a smartphone can vary from x amount to any value and it can be further broken down based on fractional values.
- The two subcategories which describe them clearly are:
  - Discrete
  - Continuous

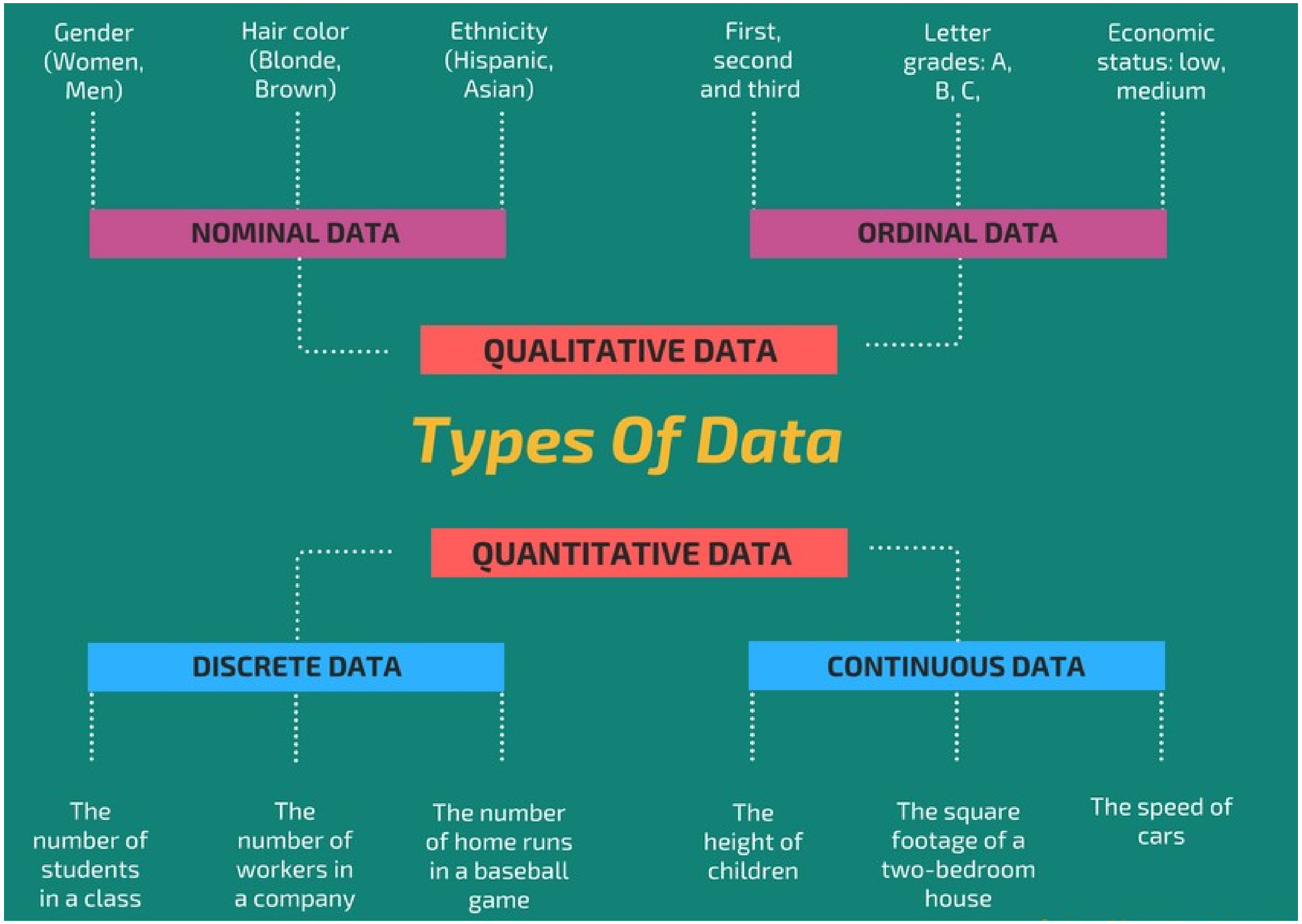
Discrete

- The numerical values which fall under are integers or whole numbers are placed under this category.
- The number of speakers in the phone, cameras, cores in the processor, the number of sims supported all these are some of the examples of the discrete data type.

Continous

- The fractional numbers are considered as continuous values.
- These can take the form of the operating frequency of the processors, the android version of the phone, wifi frequency, temperature of the cores, and so on.

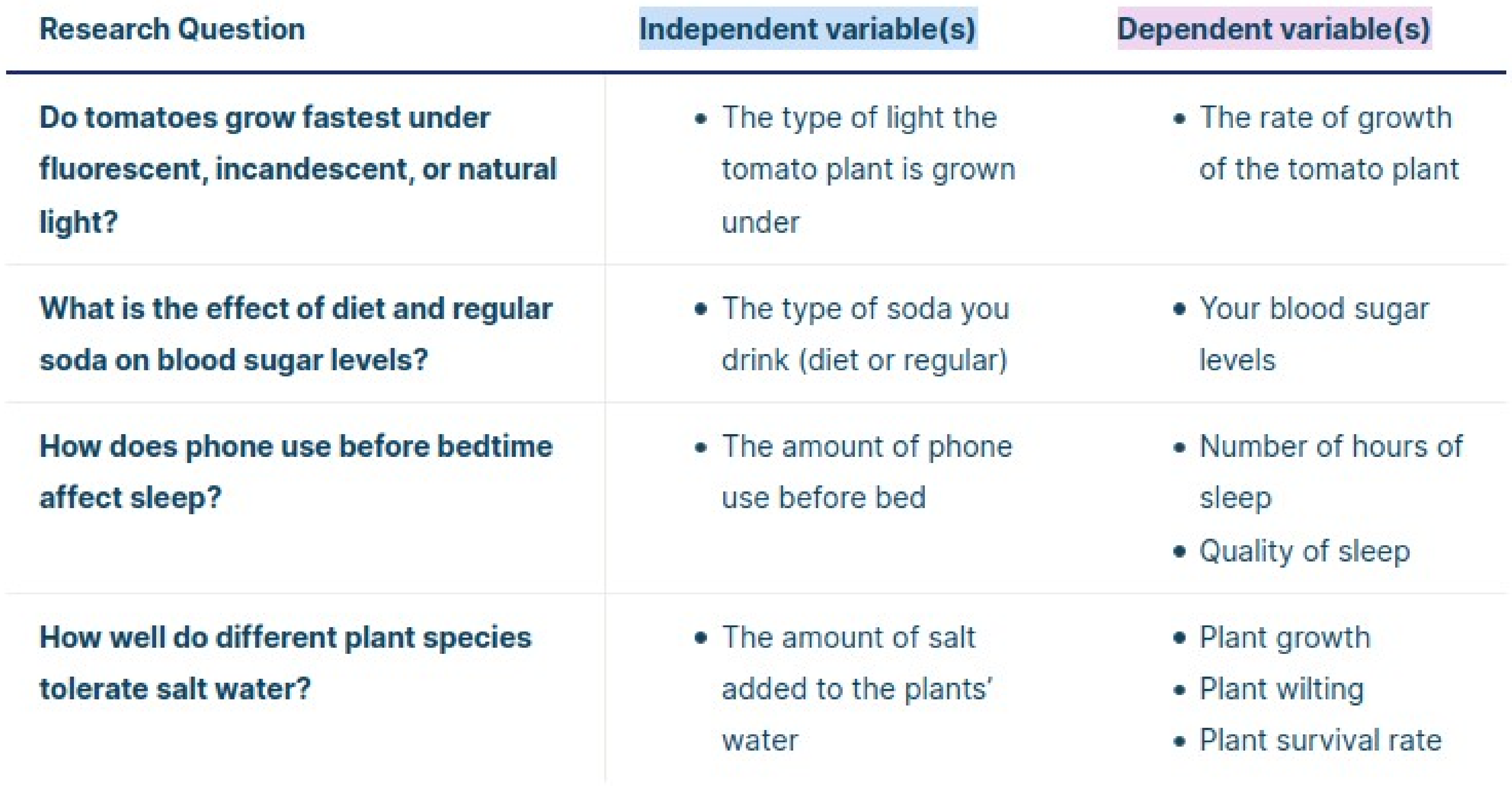
Summary


Gender (Women, Men)
Hair color (Blonde, Brown)
Ethnicity (Hispanic, Asian)
First, second and third
Letter grades: A, B, C,
Economic status: low, medium
NOMINAL DATA
ORDINAL DATA
QUALITATIVE DATA
Types Of Data
QUANTITATIVE DATA
DISCRETE DATA
CONTINUOUS DATA
The number of students in a class
The number of workers in a company
The number of home runs in a baseball game
The height of children
The square footage of a two-bedroom house
The speed of cars

Types of variables

- In research, variables are any characteristics that can take on different values, such as height, age, species, or exam score.
- In scientific research, we often want to study the effect of one variable on another one. For example, you might want to test whether students who spend more time studying get better exam scores.
- The variables in a study of a cause-and-effect relationship are called the independent and dependent variables.
  - The independent variable is the cause. Its value is independent of other variables in your study.
  - The dependent variable is the effect. Its value depends on changes in the independent variable.

Types of variables

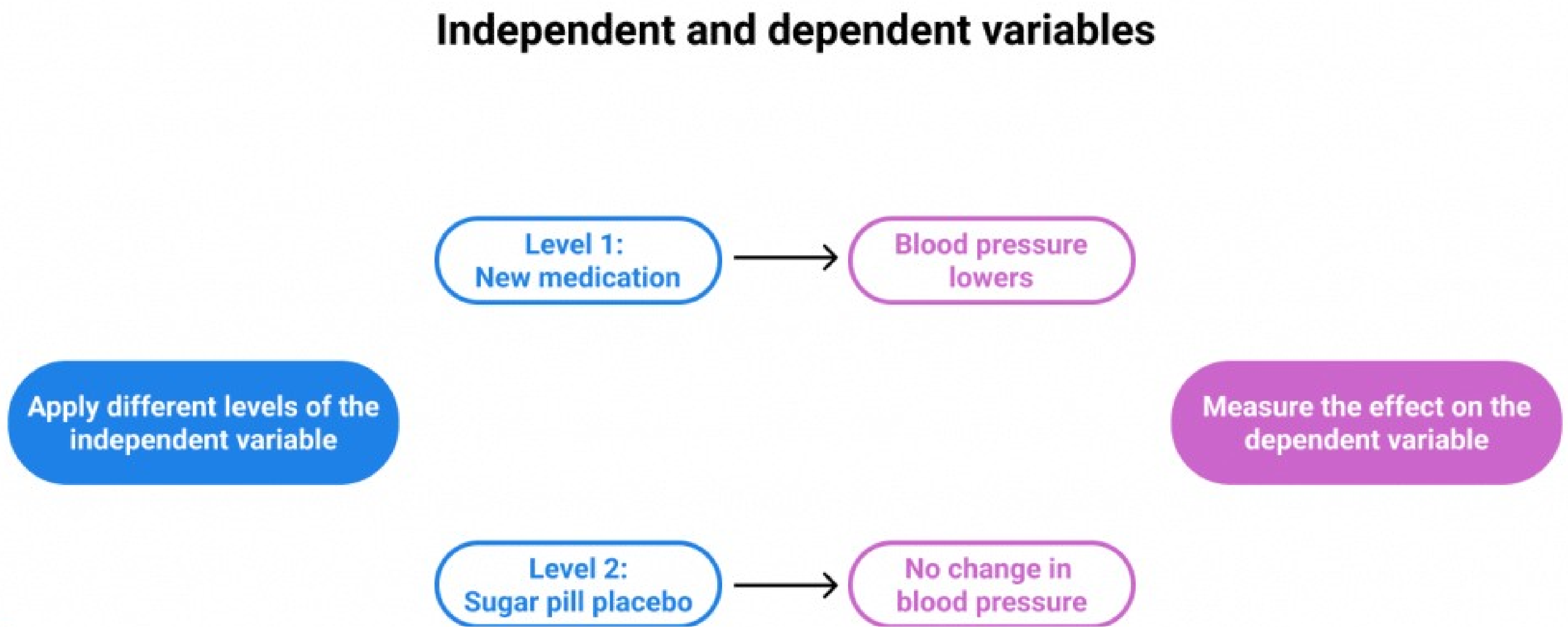


Research Question	Independent variable(s)	Dependent variable(s)
Do tomatoes grow fastest under fluorescent, incandescent, or natural light?	• The type of light the tomato plant is grown under	• The rate of growth of the tomato plant
What is the effect of diet and regular soda on blood sugar levels?	• The type of soda you drink (diet or regular)	• Your blood sugar levels
How does phone use before bedtime affect sleep?	• The amount of phone use before bed	• Number of hours of sleep • Quality of sleep
How well do different plant species tolerate salt water?	• The amount of salt added to the plants' water	• Plant growth • Plant wilting • Plant survival rate

Example:

- You are studying the impact of a new medication on the blood pressure of patients with hypertension.
- To test whether the medication is effective, you divide your patients into two groups. One group takes the medication, while the other group takes a sugar pill placebo.
  - Your independent variable is the treatment that you vary between groups: which type of pill the patient receives.
  - Your dependent variable is the outcome that you measure: the blood pressure of the patients.

Example:


Independent and dependent variables
Level 1: New medication
Blood pressure lowers
Apply different levels of the independent variable
Measure the effect on the dependent variable
Level 2: Sugar pill placebo
No change in blood pressure

Example:

- Imagine that a tutor asks 100 students to complete a maths test. The tutor wants to know why some students perform better than others. Whilst the tutor does not know the answer to this, she thinks that it might be because of two reasons:
  - (1) some students spend more time revising for their test; and (2) some students are naturally more intelligent than others. As such, the tutor decides to investigate the effect of revision time and intelligence on the test performance of the 100 students.
- Dependent Variable: Test Mark (measured from 0 to 100)
- Independent Variables: Revision time (measured in hours) Intelligence (measured using IQ score)

Other names for independent variables

- Sometimes, the variable you think is the cause might not be fully independent – it might be influenced by other variables. In this case, one of these terms is more appropriate:
  - Explanatory variables (they explain an event or outcome)
  - Predictor variables (they can be used to predict the value of a dependent variable)
  - Right-hand-side variables (they appear on the right-hand side of a regression equation).

Other names for dependent variables

- Dependent variables are also known by these terms:
  - Response variables (they respond to a change in another variable)
  - Outcome variables (they represent the outcome you want to measure)
  - Left-hand-side variables (they appear on the left-hand side of a regression equation)

Univatiate Data

- This type of data consists of only one variable.
- The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes.
- It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it.
- The example of a univariate data can be height.

Univatiate Data

Heights (in cm)	164	167.3	170	174.2	178	180	186

Univatiate Data

- Suppose that the heights of seven students of a class is recorded, there is only one variable that is height and it is not dealing with any cause or relationship.
- The description of patterns found in this type of data can be made by drawing conclusions using central tendency measures (mean, median and mode), dispersion or spread of data (range, minimum, maximum, quartiles, variance and standard deviation) and by using frequency distribution tables, histograms, pie charts, frequency polygon and bar charts.

Bivatiate Data

- This type of data involves two different variables.
- The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship among the two variables.
- Example of bivariate data can be temperature and ice cream sales in summer season.

Bivatiate Data

TEMPERATURE(IN CELSIUS)	ICE CREAM SALES
20	2000
25	2500
35	5000
43	7800

Bivatiate Data

- Suppose the temperature and ice cream sales are the two variables of a bivariate data.
- Here, the relationship is visible from the table that temperature and sales are directly proportional to each other and thus related because as the temperature increases, the sales also increase.
- Thus bivariate data analysis involves comparisons, relationships, causes and explanations.
- These variables are often plotted on X and Y axis on the graph for better understanding of data and one of these variables is independent while the other is dependent.

Multivatiate Data

- When the data involves three or more variables, it is categorized under multivariate.
- Example of this type of data is suppose an advertiser wants to compare the popularity of four advertisements on a website, then their click rates could be measured for both men and women and relationships between variables can then be examined.

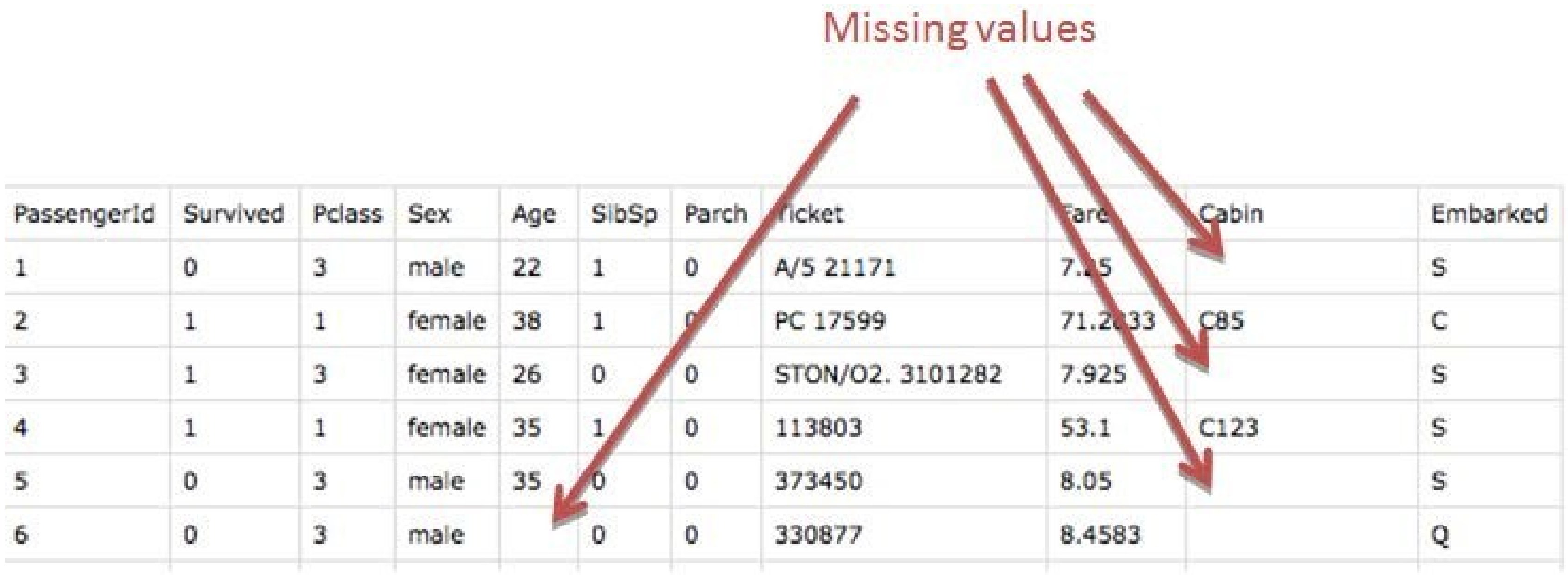
Multivatiate Data

- It is similar to bivariate but contains more than one dependent variable.
- The ways to perform analysis on this data depends on the goals to be achieved.Some of the techniques are regression analysis,path analysis,factor analysis and multivariate analysis of variance (MANOVA).

Missing Values

- In real world data, there are some instances where a particular element is absent because of various reasons, such as, corrupt data, failure to load the information, or incomplete extraction.
- Handling the missing values is one of the greatest challenges faced by analysts, because making the right decision on how to handle it generates robust data models.
- Let us look at different ways of imputing the missing values.

Missing Values


Missing values

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

Missing Values – Replacement policy

- Ignore the records with missing values.
- Replace them with a global constant (e.g., “?”).
- Fill in missing values manually based on your domain knowledge.
- Replace them with the variable mean (if numerical) or the most frequent value (if categorical).
- Use modeling techniques such as nearest neighbors, Bayes’ rule, decision tree, or EM algorithm.

Missing Data – Basic Handling

```
# Find the missing values
print data.isnull().sum()

# Drop the missing values
print data.dropna()

# Fill all NaN elements with 0
print data.fillna(0)

# Previous value will be replaced with NaN
print data.fillna(method='pad')

# Next value will be replaced with NaN
print data.fillna(method='backfill')
```

Missing data – deleting rows

- This method commonly used to handle the null values.
- Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values.
- This method is advised only when there are enough samples in the data set.
- One has to make sure that after we have deleted the data, there is no addition of bias.
- Removing the data will lead to loss of information which will not give the expected results while predicting the output.

Missing data – deleting rows

```
# Replace the NA values
print data.dropna(inplace=True)

# Print NA values
print data.isnull().sum()
```

Missing data – mean/median/mode

- This strategy can be applied on a feature which has numeric data like the age of a person or the ticket fare.
- We can calculate the mean, median or mode of the feature and replace it with the missing values.
- This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns.
- This method is also called as leaking the data while training.
- Another way is to approximate it with the deviation of neighbouring values. This works better if the data is linear.

Missing data – mean/median/mode

```
# Find mean of missing values
print data['age'].mean()

# Replace NaN by mean of others
print data['age'].replace(np.NaN, data['age'].mean())

# Replace NaN by median of others
print data['age'].replace(np.NaN, data['age'].median())

# Replace NaN by mode of others
print data['age'].replace(np.NaN, data['age'].mode())
```

Missing Data – Assign unique values

- A categorical feature will have a definite number of possibilities, such as gender, for example.
- Since they have a definite number of classes, we can assign another class for the missing values.
- Here, the features Cabin and Embarked have missing values which can be replaced with a new category, say, U for ‘unknown’.
- This strategy will add more information into the dataset which will result in the change of variance. Since they are categorical, we need to find one hot encoding to convert it to a numeric form for the algorithm to understand it.

Missing Data – Assign unique values

```
# Assign undefined values
print data.fillna('U')
```

Missing Data – Using regression

- Using the features which do not have missing values, we can predict the nulls with the help of a machine learning algorithm.
- This method may result in better accuracy, unless a missing value is expected to have a very high variance.
- We will be using linear regression to replace the nulls in the feature 'age', using other available features.
- One can experiment with different algorithms and check which gives the best accuracy instead of sticking to a single algorithm.

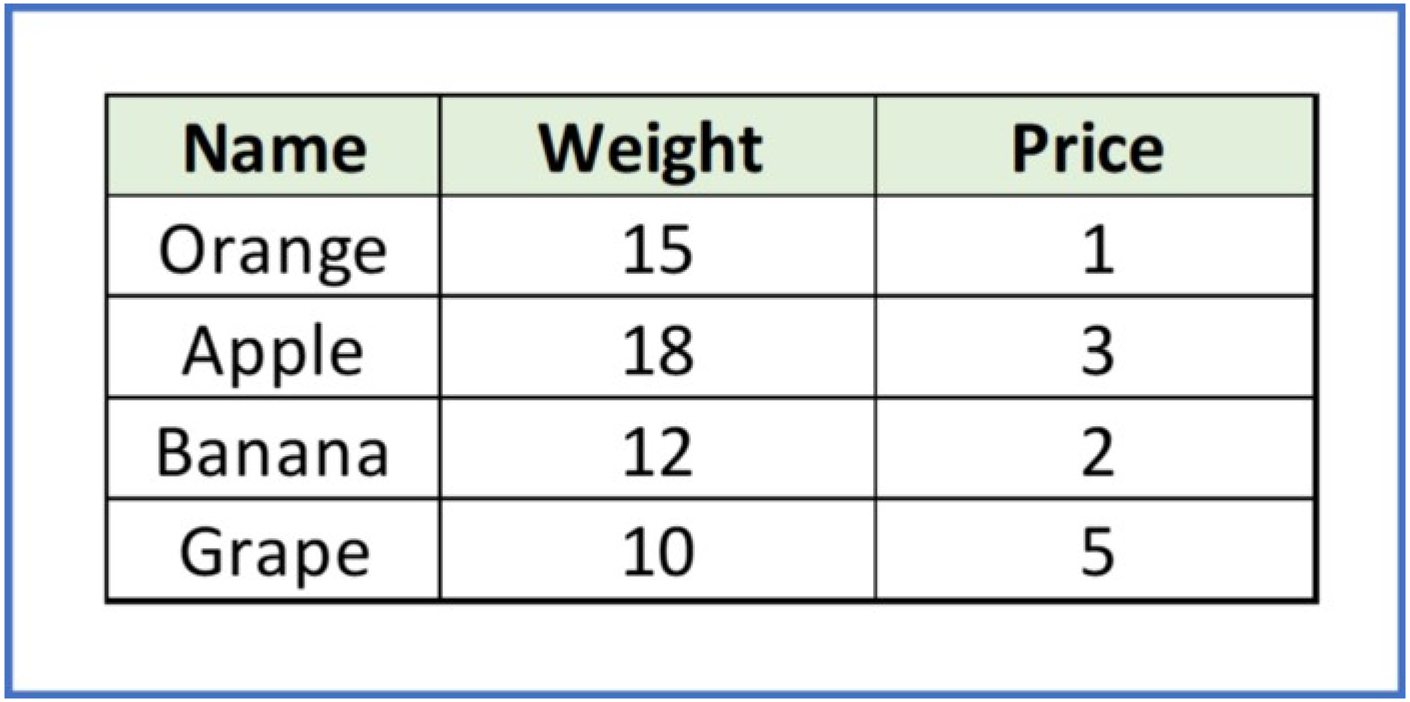
Normalization

- Real world dataset contains features that highly vary in magnitudes, units, and range.
- Normalization should be performed when the scale of a feature is irrelevant or misleading and not should Normalize when the the scale is meaningful.
- The algorithms which use Euclidean Distance measure are sensitive to Magnitudes. Here feature scaling helps to weigh all the features equally.
- Formally, If a feature in the dataset is big in scale compared to others then in algorithms where Euclidean distance is measured this big scaled feature becomes dominating and needs to be normalized.

Normalization

- Normalization is a technique often applied as part of data preparation for machine learning.
- The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.
- For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

Normalization : Example



Name	Weight	Price
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5

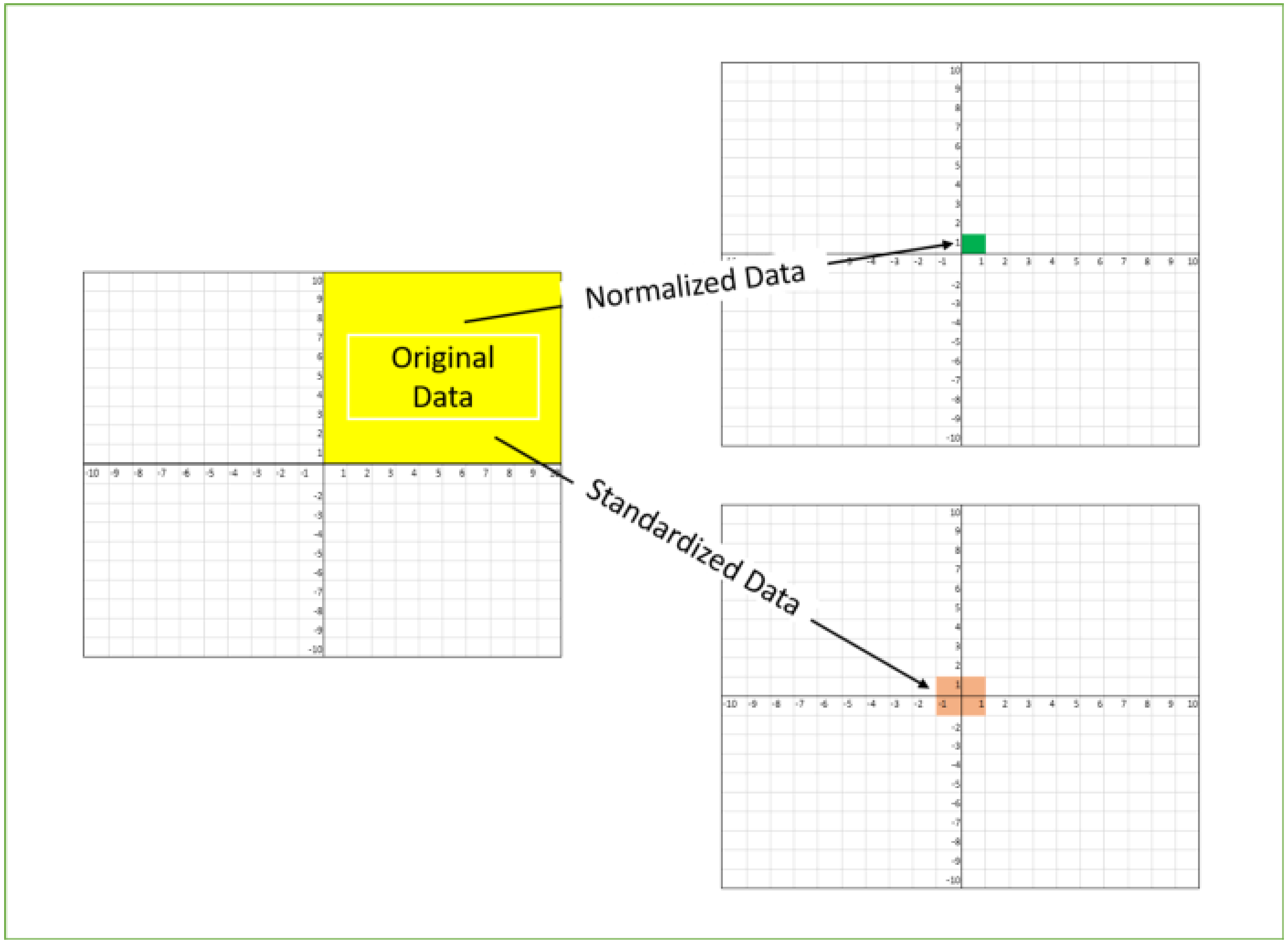
Normalization

- Suppose we have two features of weight and price, as in the below table.
- The “Weight” cannot have a meaningful comparison with the “Price.” So the assumption algorithm makes that since “Weight” > “Price,” thus “Weight,” is more important than “Price.”
- So these more significant number starts playing a more decisive role while training the model.
- Thus feature scaling is needed to bring every feature in the same footing without any upfront importance. Interestingly, if we convert the weight to “Kg,” then “Price” becomes dominant.

Normalization / Scaling Techniques

- Basic Normalization Techniques are-
  - Min-Max Scaling
  - Feature Scaling
  - Robust Scaling
  - Max Abs Scaling
  - Quantile Transformer Scaling
  - Power Transformer Scaling
  - Unit Vector Scaling

Normalization Techniques



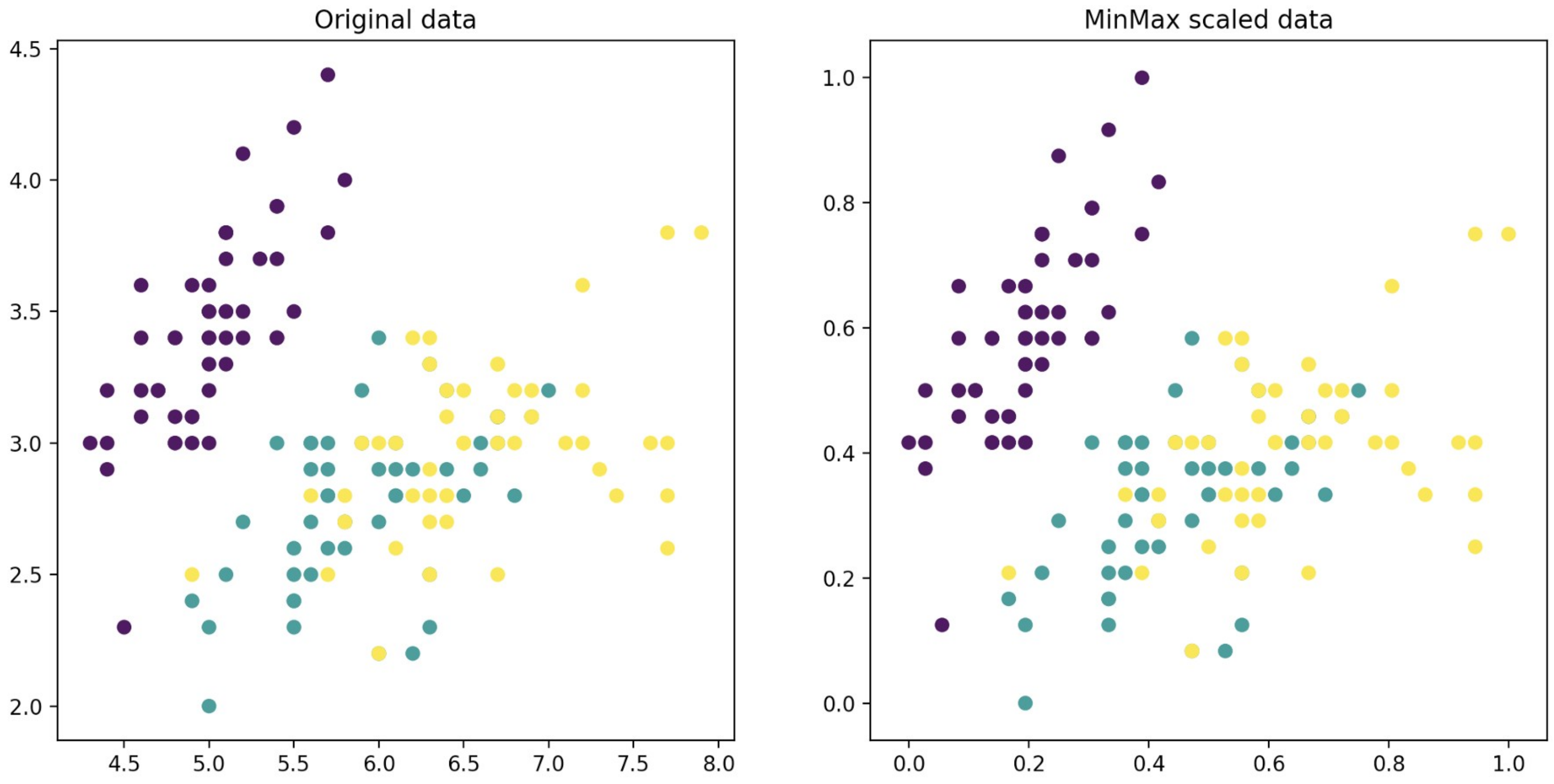
Min-Max Scaling

- It is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.
- Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.
  - When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
  - On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
  - If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

Min-Max Scaling



Example:

- For example, for a dataset, we could guesstimate the min and max observable values as 30 and -10.
- We can then normalize any value, like 18.8, as follows:
  - y = (x – min) / (max – min)
  - y = (18.8 – (-10)) / (30 – (-10))
  - y = 28.8 / 40
  - y = 0.72

Feature Scaling or Standardization

- Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation.
- This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.
- Here's the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

Which one to use?

- Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
- Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

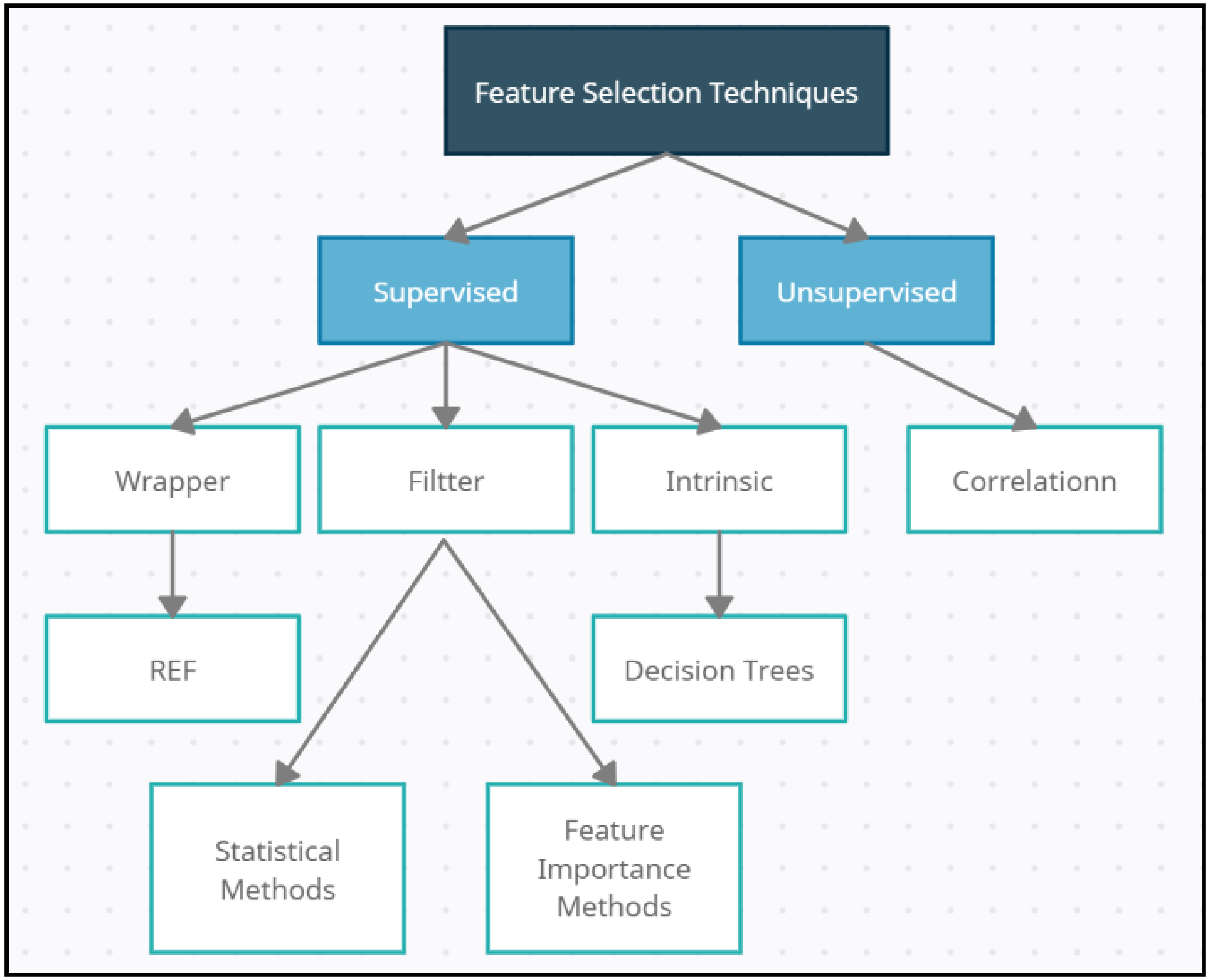
Feature Selection

- The training time and performance of a machine learning algorithm depends heavily on the features in the dataset. Ideally, we should only retain those features in the dataset that actually help our machine learning model learn something.
- Unnecessary and redundant features not only slow down the training time of an algorithm, but they also affect the performance of the algorithm. The process of selecting the most suitable features for training the machine learning model is called "feature selection".

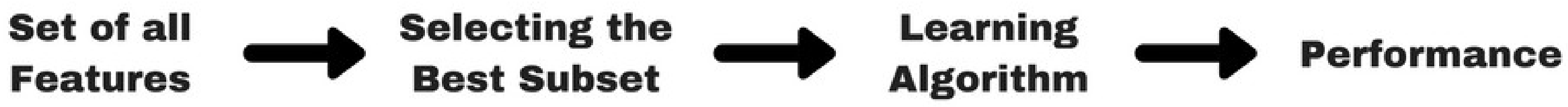
Why feature selection ?

- There are several advantages of performing feature selection before training machine learning models, some of which have been enlisted below:
  - Models with less number of features have higher explainability
  - It is easier to implement machine learning models with reduced features
  - Fewer features lead to enhanced generalization which in turn reduces overfitting
  - Feature selection removes data redundancy
  - Training time of models with fewer features is significantly lower
  - Models with fewer features are less prone to errors

Types


Feature Selection Techniques
Supervised
Unsupervised
Wrapper
Filtter
Intrinsic
Correlationn
REF
Decision Trees
Statistical Methods
Feature Importance Methods

Filter Methods



- Filter method relies on the general uniqueness of the data to be evaluated and pick feature subset, not including any mining algorithm.
- Filter method uses the exact assessment criterion which includes distance, information, dependency, and consistency.
- The filter method uses the principal criteria of ranking technique and uses the rank ordering method for variable selection.

Filter Methods

- Filters methods belong to the category of feature selection methods that select features independently of the machine learning algorithm model. This is one of the biggest advantages of filter methods.
- Features selected using filter methods can be used as an input to any machine learning models.
- Another advantage of filter methods is that they are very fast. Filter methods are generally the first step in any feature selection pipeline.
- They are broadly categorized into two categories:
  - Univariate Filter Methods
  - Multivariate Filter Methods.

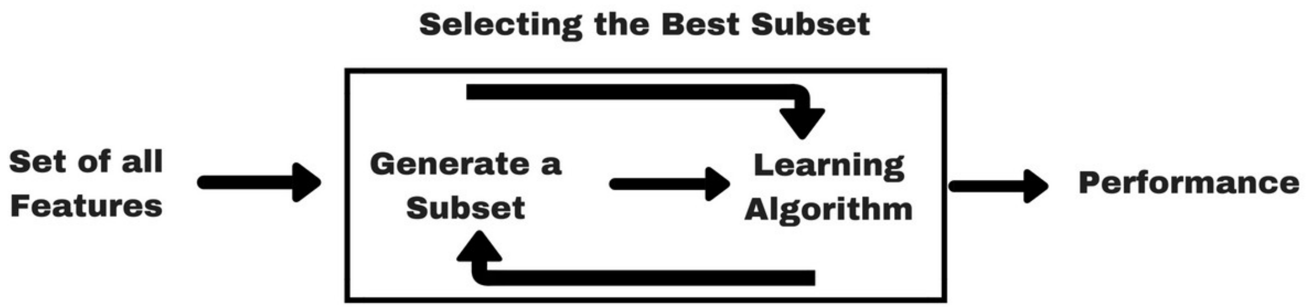
Univariate Filter Methods

- The univariate filter methods are the type of methods where individual features are ranked according to specific criteria.
- The top N features are then selected. Different types of ranking criteria are used for univariate filter methods, for example fisher score, mutual information, and variance of the feature.
- One of the major disadvantage of univariate filter methods is that they may select redundant features because the relationship between individual features is not taken into account while making decisions.
- Univariate filter methods are ideal for removing constant and quasi-constant features from the data.

Multivariate Filter Methods

- Multivariate filter methods are capable of removing redundant features from the data since they take the mutual relationship between the features into account.
- Multivariate filter methods can be used to remove duplicate and correlated features from the data.

Wrapper Methods



- A wrapper method needs one machine learning algorithm and uses its performance as evaluation criteria.
- This method searches for a feature which is best-suited for the machine learning algorithm and aims to improve the mining performance.
- To evaluate the features, the predictive accuracy used for classification tasks and goodness of cluster is evaluated using clustering.

Common Filter Methods

- Remove constant features
- Remove quasi-constant features
- Remove duplicate features and
- Remove correlated features

Remove Constant Features

- Constant features are the type of features that contain only one value for all the outputs in the dataset.
- Constant features provide no information that can help in classification of the record at hand. Therefore, it is advisable to remove all the constant features from the dataset.
- Let's see how we can remove constant features from a dataset. The dataset that we are going to use for this example is the Santandar Customer Satisfaction dataset, that can be downloaded from Kaggle.

Let’s Code

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
```

```
df = pd.read_csv('satandar.csv', nrows=40000)
df.shape
```

```
(40000, 371)
```

```
df.columns;
```

```
X_train, X_test, y_train, y_test =train_test_split(
    df.drop(labels=['TARGET'], axis=1),
    df['TARGET'],
    test_size=0.2,
    random_state=0)
```

Variance Threshold

- VarianceThreshold(threshold=0.0)
  - Feature selector that removes all low-variance features.
  - threshold
    - Features with a training-set variance lower than this threshold will be removed.
    - The default is to keep all features with non-zero variance, i.e. remove the features that have the same value in all samples.

Variance Threshold

```
constant_filter = VarianceThreshold(threshold=0)
```

```
constant_filter.fit(X_train)
```

```
VarianceThreshold(threshold=0)
```

```
len(X_train.columns[constant_filter.get_support()])
```

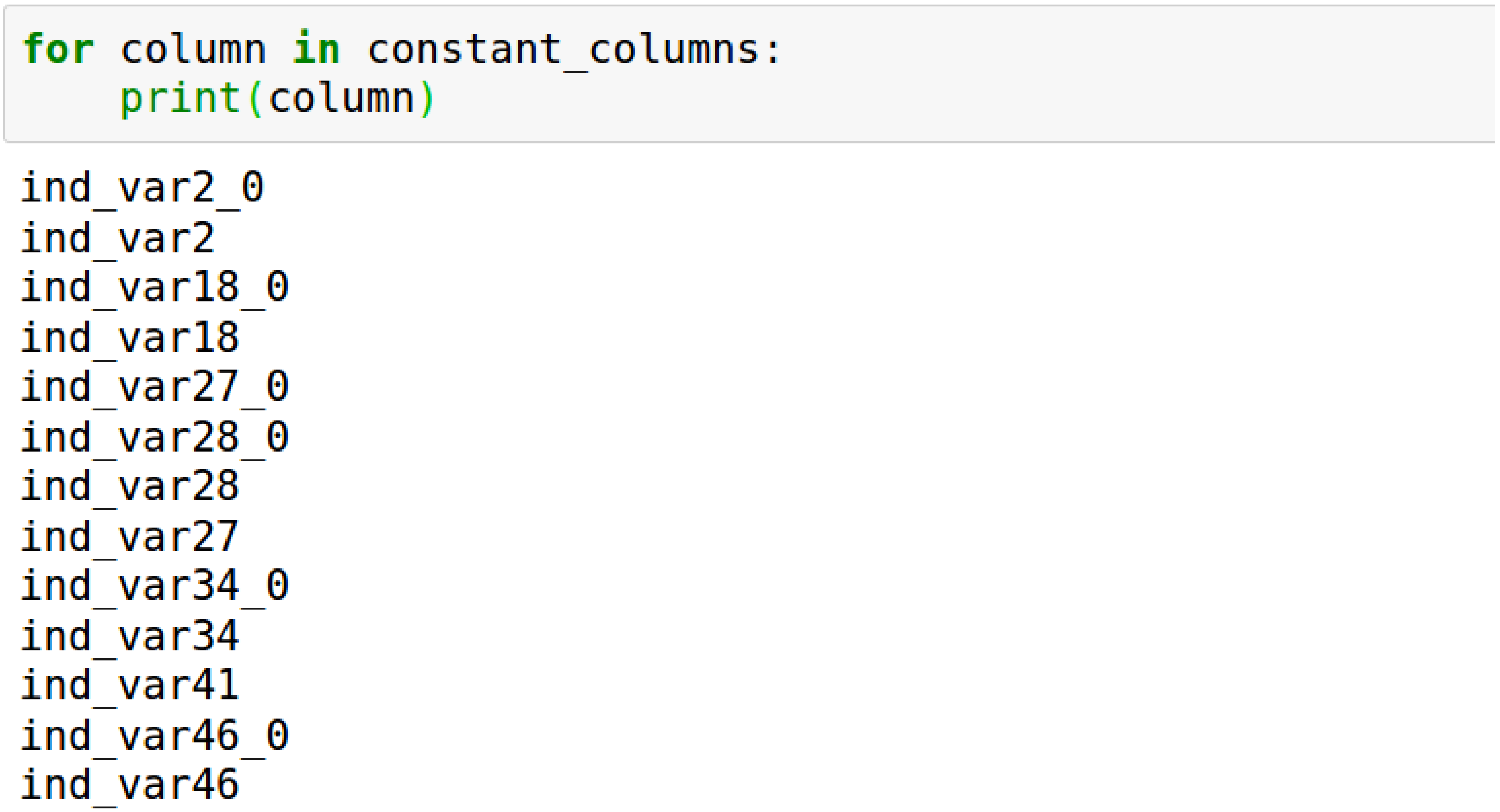
```
314
```

```
constant_columns = [column for column in X_train.columns
   if column not in X_train.columns[constant_filter.get_support()]]

print(len(constant_columns))
```

```
56
```

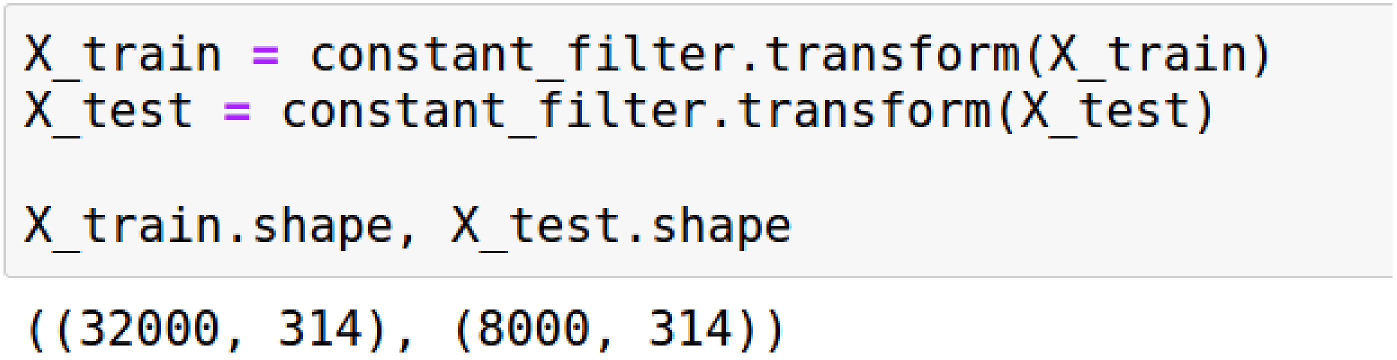
Constant Columns



```python
for column in constant_columns:
    print(column)
```

```
ind_var2_0
ind_var2
ind_var18_0
ind_var18
ind_var27_0
ind_var28_0
ind_var28
ind_var27
ind_var34_0
ind_var34
ind_var41
ind_var46_0
ind_var46
```

Extracted Features



```
X_train = constant_filter.transform(X_train)
X_test = constant_filter.transform(X_test)

X_train.shape, X_test.shape
```

```
((32000, 314), (8000, 314))
```

Remove duplicate features

- Duplicate features are the features that have similar values.
- Duplicate features do not add any value to algorithm training, rather they add overhead and unnecessary delay to the training time.
- Therefore, it is always recommended to remove the duplicate features from the dataset before training.
- Removing duplicate columns can be computationally costly since we have to take the transpose of the data matrix before we can remove duplicate features.

Let's Code

```
# Remove duplicates
df = pd.read_csv('satandar.csv', nrows=40000)
df.shape
```

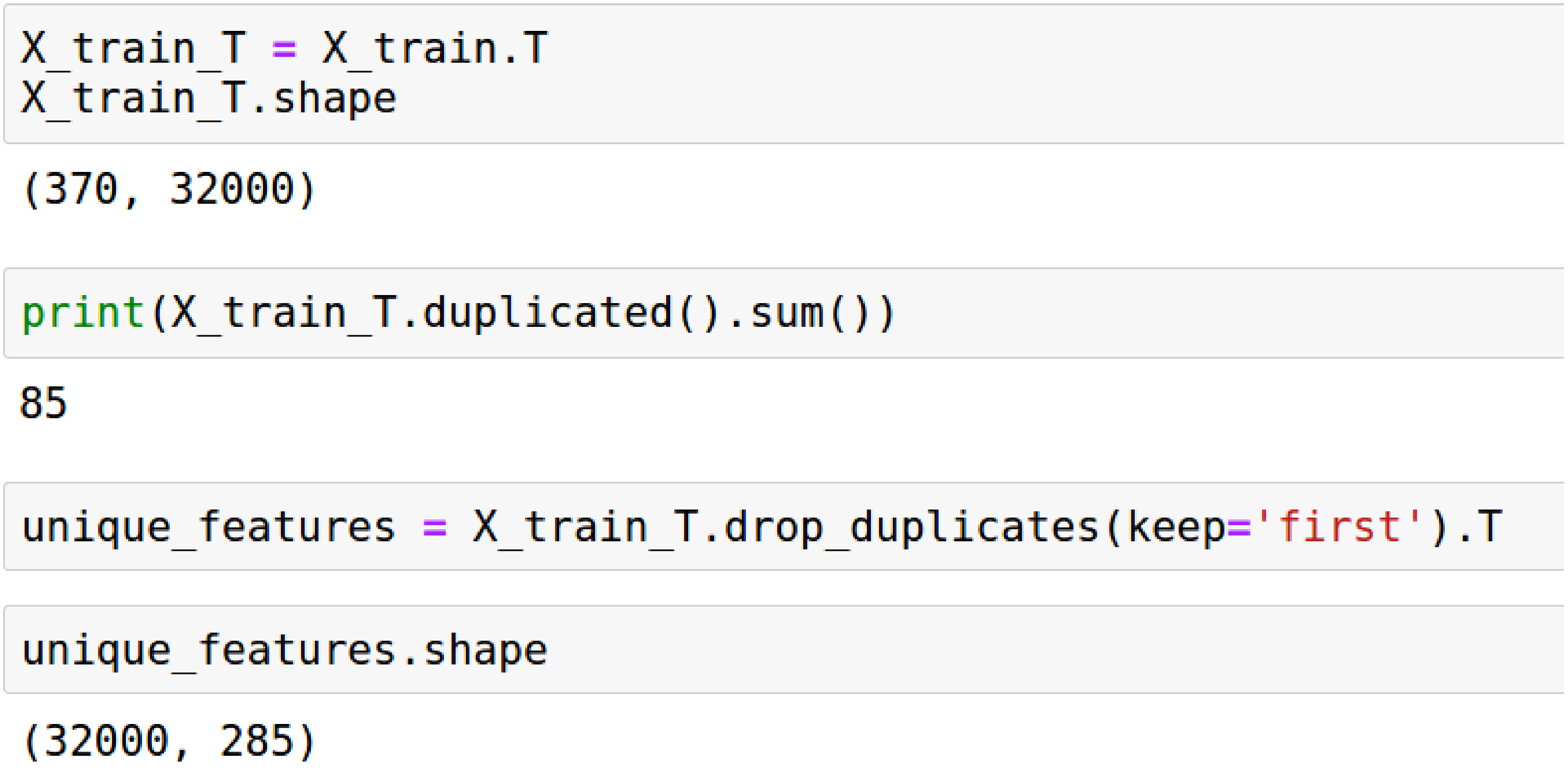
```
(40000, 371)
```

```
X_train, X_test, y_train, y_test =train_test_split(
    df.drop(labels=['TARGET'], axis=1),
    df['TARGET'],
    test_size=0.2,
    random_state=0)
```

```
train_features_T = train_features.T
train_features_T.shape
```

```
(370, 32000)
```

Unique and duplicated features



```python
X_train_T = X_train.T
X_train_T.shape
```

```
(370, 32000)
```

```python
print(X_train_T.duplicated().sum())
```

```
85
```

```python
unique_features = X_train_T.drop_duplicates(keep='first').T
```

```python
unique_features.shape
```

```
(32000, 285)
```

Get the duplicated features

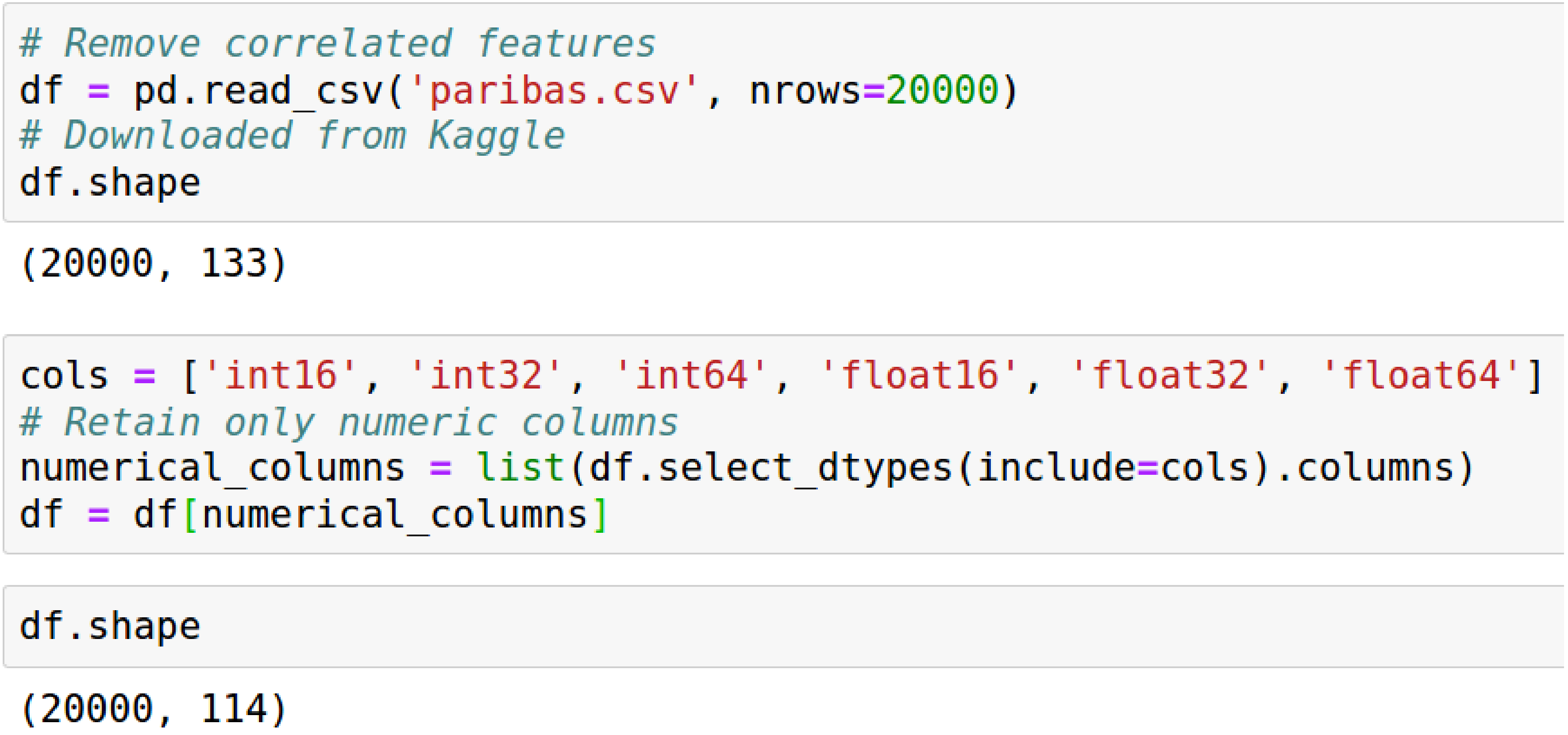
```
duplicated_features = [dup_col for dup_col in X_train.columns
                       if dup_col not in unique_features.columns]
duplicated_features
```

```
['ind_var2',
 'ind_var13_medio',
 'ind_var18_0',
 'ind_var18',
 'ind_var26',
 'ind_var25',
 'ind_var27_0',
 'ind_var28_0',
 'ind_var28',
 'ind_var27',
 'ind_var29_0',
 'ind_var29',
 'ind_var32',
 'ind_var34_0',
 'ind_var34',
```

Remove correlated features

- A dataset can also contain correlated features. Two or more than two features are correlated if they are close to each other in the linear space.
- Take the example of the feature set for a fruit basket, the weight of the fruit basket is normally correlated with the price. The more the weight, the higher the price.
- Correlation between the output observations and the input features is very important and such features should be retained.
- However, if two or more than two features are mutually correlated, they convey redundant information to the model and hence only one of the correlated features should be retained to reduce the number of features.

Let’s Code



```
# Remove correlated features
df = pd.read_csv('paribas.csv', nrows=20000)
# Downloaded from Kaggle
df.shape
```

```
(20000, 133)
```

```
cols = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
# Retain only numeric columns
numerical_columns = list(df.select_dtypes(include=cols).columns)
df = df[numerical_columns]
```

```
df.shape
```

```
(20000, 114)
```

Find Correlations

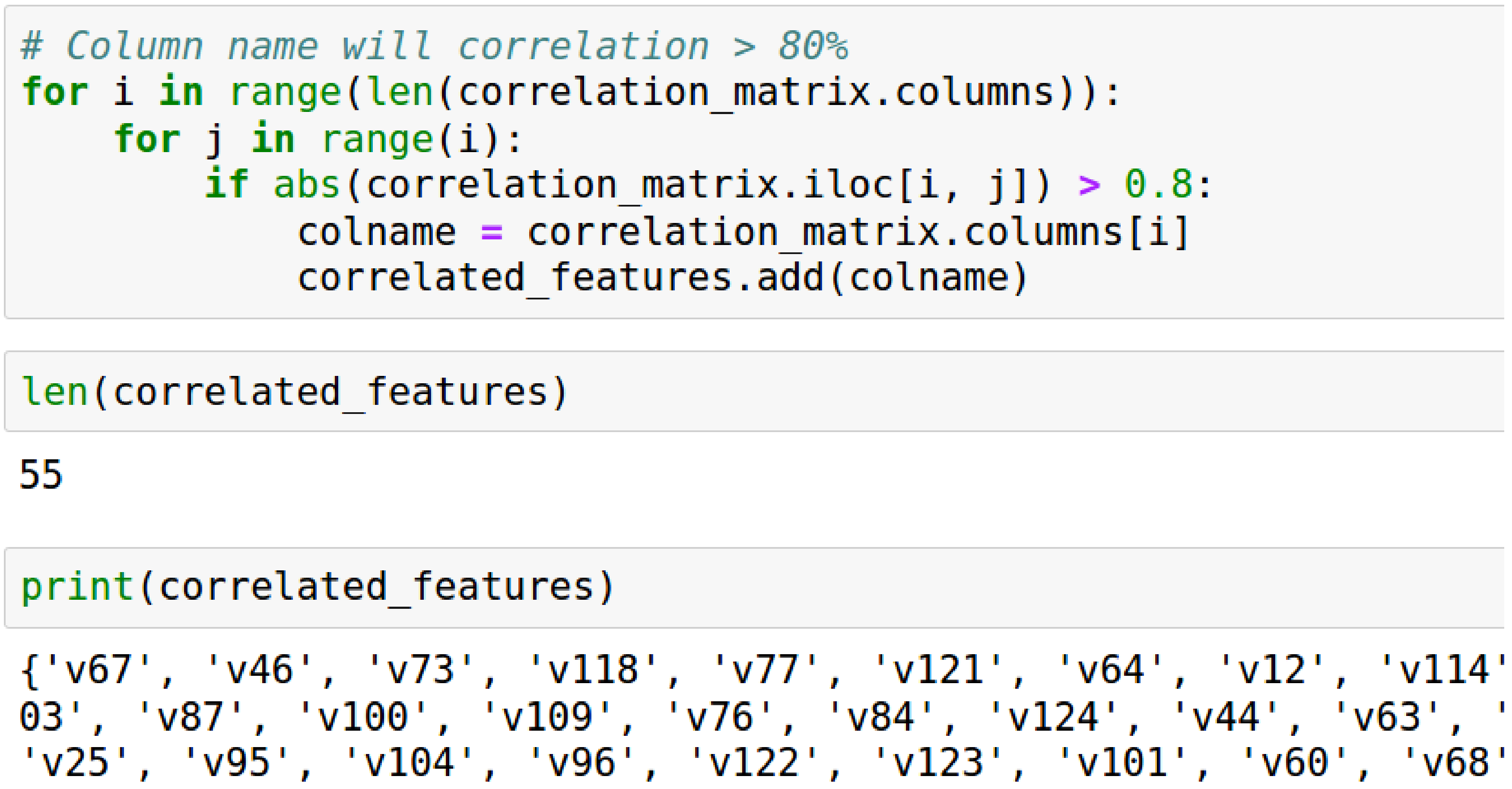
```
X_train, X_test, y_train, y_test =train_test_split(
    df.drop(['target','ID'], axis=1),
    df['target'],
    test_size=0.2,
    random_state=0)
```

```
correlated_features = set()
correlation_matrix = df.corr()
```

```
correlation_matrix.head()
```

	ID	target	v1	v2	v4	v5	v6
ID	1.000000	0.005830	0.004817	0.001134	-0.003499	0.002015	-0.010675
target	0.005830	1.000000	-0.016746	0.042634	0.054891	0.009453	0.043471
v1	0.004817	-0.016746	1.000000	-0.205826	-0.145037	-0.049337	-0.020251

Correlation above 80%



```python
# Column name will correlation > 80%
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.8:
            colname = correlation_matrix.columns[i]
            correlated_features.add(colname)
```

```python
len(correlated_features)
```

```
55
```

```python
print(correlated_features)
```

```
{'v67', 'v46', 'v73', 'v118', 'v77', 'v121', 'v64', 'v12', 'v114'
03', 'v87', 'v100', 'v109', 'v76', 'v84', 'v124', 'v44', 'v63', '
'v25', 'v95', 'v104', 'v96', 'v122', 'v123', 'v101', 'v60', 'v68'
```

Drop correlated features

```
X_train.drop(labels=correlated_features, axis=1, inplace=True)
X_test.drop(labels=correlated_features, axis=1, inplace=True)
```

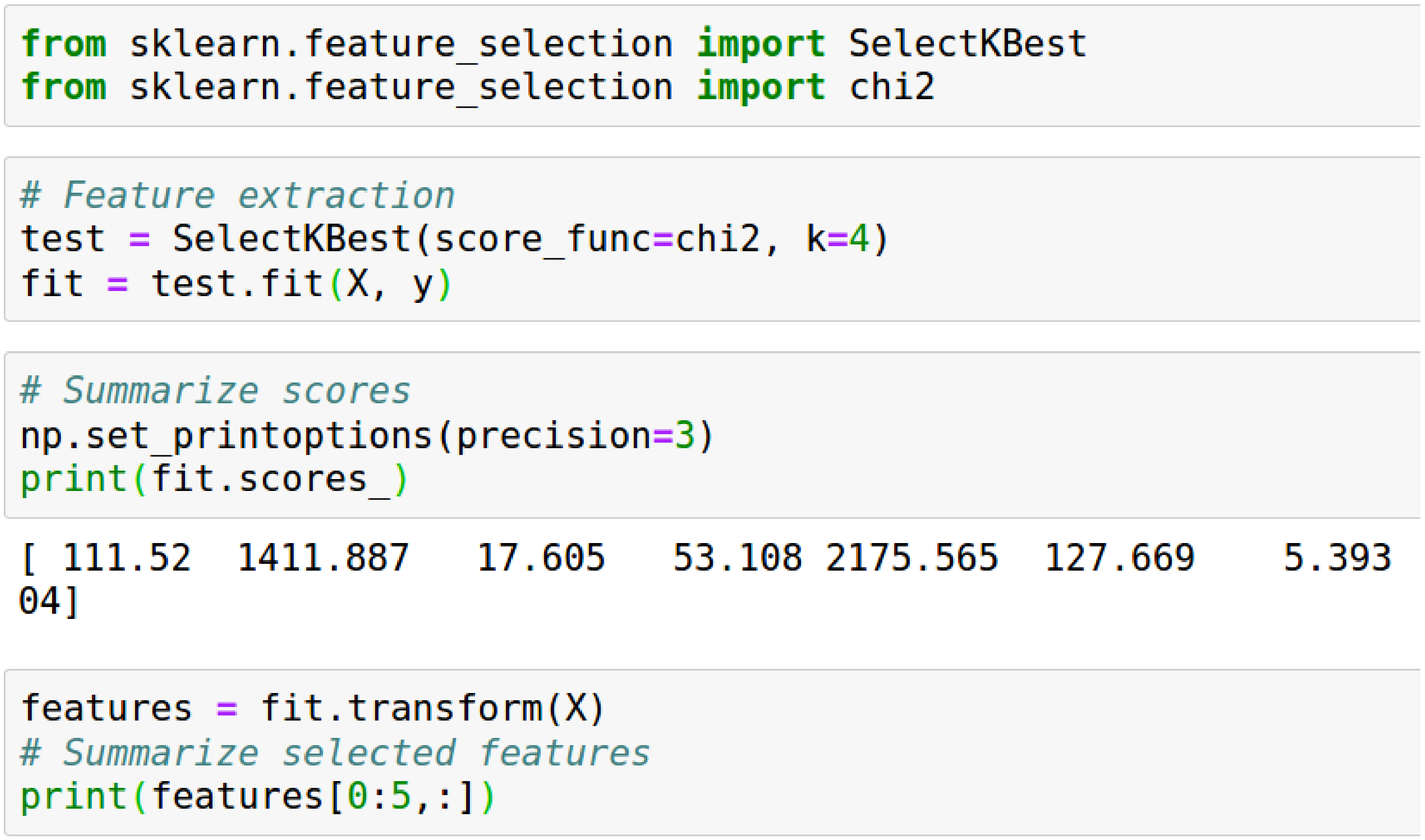
```
X_train.shape
```

```
(16000, 57)
```

```
X_train.head()
```

	v1	v2	v4	v5	v6	v7	v8	v9
17815	1.573723	4.744721	3.731608	10.831683	1.474480	1.720226	1.852197	8.846154
18370	0.896505	10.448345	6.167674	9.769273	2.146071	2.969885	1.450758	9.478909
1379	1.867116	10.866761	3.853017	9.191264	2.169892	3.061395	2.301629	7.830188

Select K best



```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

```
# Feature extraction
test = SelectKBest(score_func=chi2, k=4)
fit = test.fit(X, y)
```

```
# Summarize scores
np.set_printoptions(precision=3)
print(fit.scores_)
```

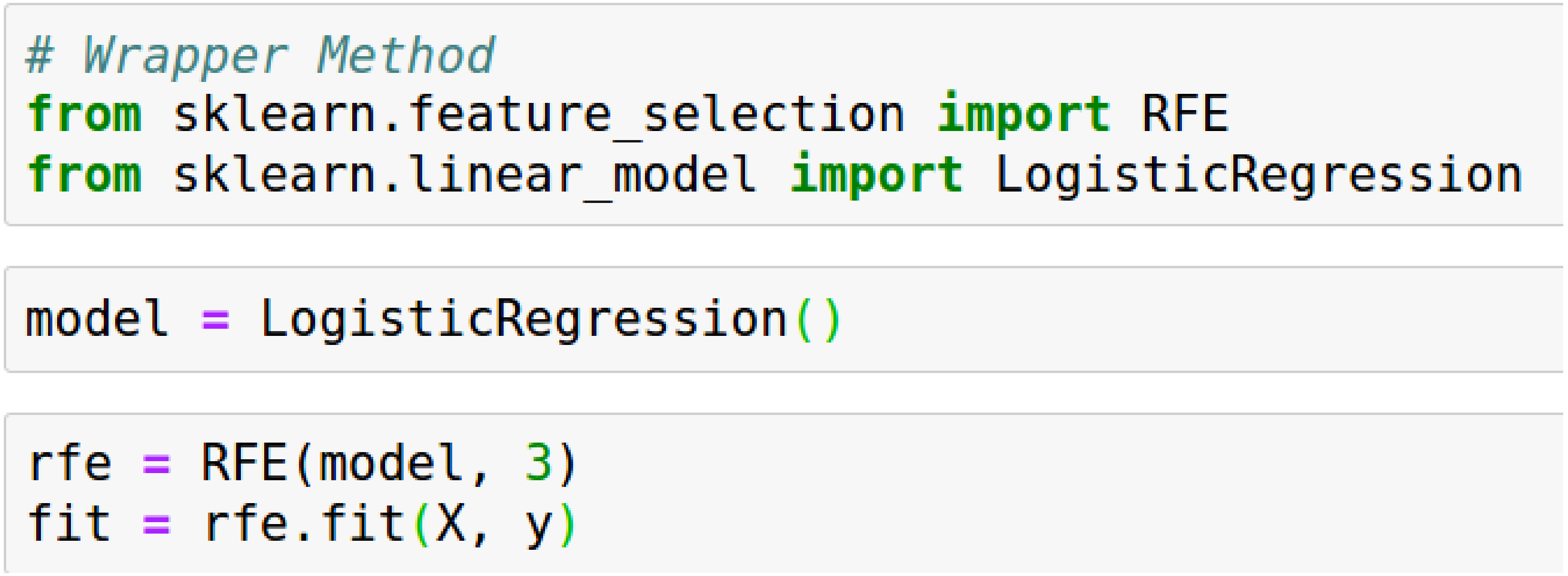
```
[ 111.52   1411.887   17.605    53.108 2175.565   127.669     5.393
04]
```

```
features = fit.transform(X)
# Summarize selected features
print(features[0:5,:])
```

Recursive Feature Elimination

- The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.
- It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

RFE



```
# Wrapper Method
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression()
```

```
rfe = RFE(model, 3)
fit = rfe.fit(X, y)
```

RFE

```
print("Num Features: %s" % (fit.n_features_))
print("Selected Features: %s" % (fit.support_))
print("Feature Ranking: %s" % (fit.ranking_))
```

```
Num Features: 3
Selected Features: [ True False False False False  True
Feature Ranking: [1 2 4 5 6 1 1 3]
```

```
X.iloc[:,fit.support_].head()
```

	TimesPregnant	BMI	DiabetesFunct
0	6	33.6	0.627
1	1	26.6	0.351

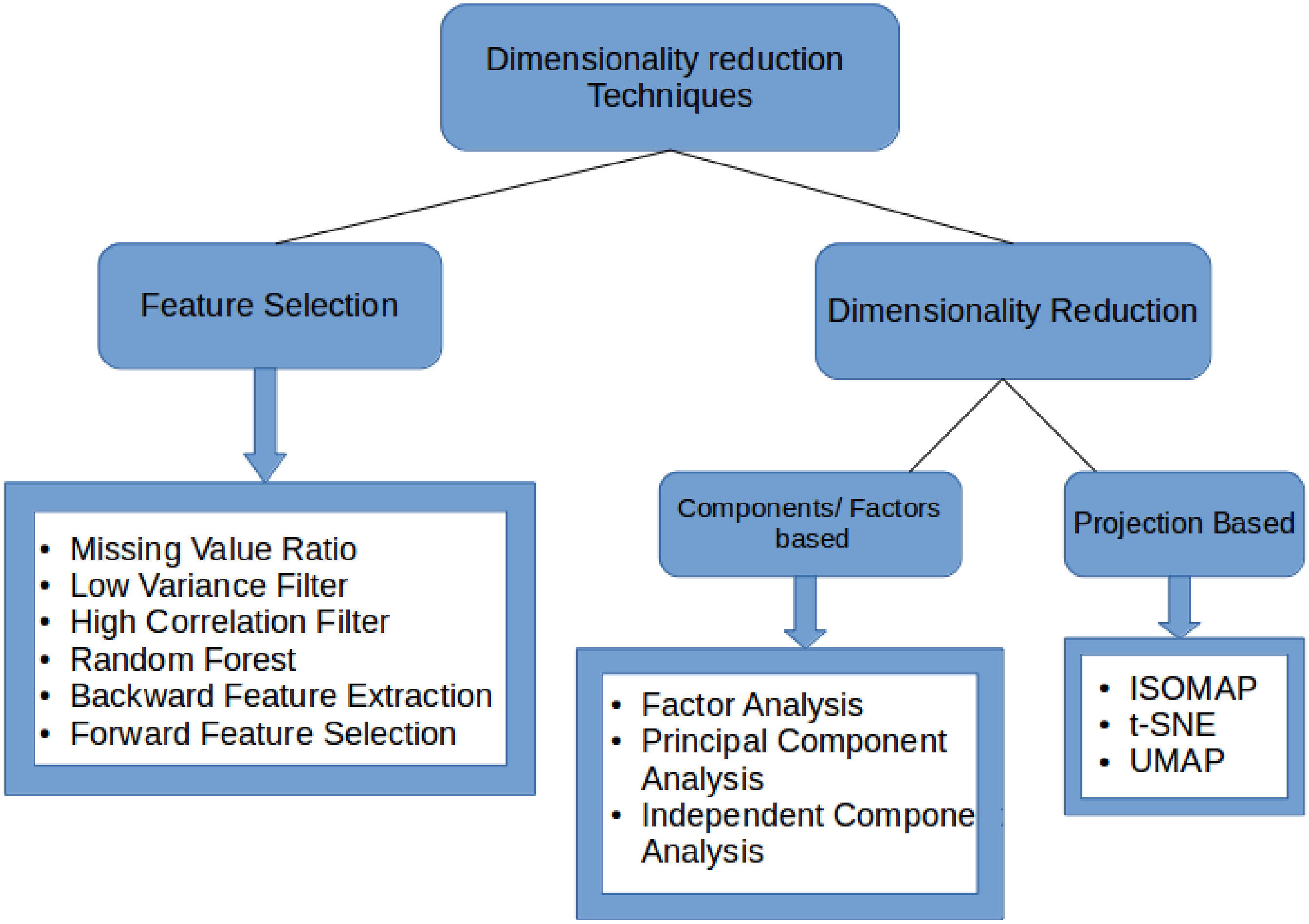
Dimensionality Reduction

- Dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables.
- It can be divided into feature selection and feature extraction.
  - Feature selection approaches try to find a subset of the original variables (also called features or attributes).
  - Feature projection or Feature extraction transforms the data in the high-dimensional space to a space of fewer dimensions.

Large Dimensions

- Large number of features in the dataset is one of the factors that affect both the training time as well as accuracy of machine learning models. You have different options to deal with huge number of features in a dataset.
  - Try to train the models on original number of features, which take days or weeks if the number of features is too high.
  - Reduce the number of variables by merging correlated variables.
  - Extract the most important features from the dataset that are responsible for maximum variance in the output.

Dimensionality Reduction Techniques


Dimensionality reduction Techniques
Feature Selection
Dimensionality Reduction
Components/ Factors based
Projection Based
Missing Value Ratio
Low Variance Filter
High Correlation Filter
Random Forest
Backward Feature Extraction
Forward Feature Selection
Factor Analysis
Principal Component Analysis
Independent Compone Analysis
ISOMAP
t-SNE
UMAP

Principal Component Analysis

- Principal component analysis, or PCA, is a statistical technique to convert high dimensional data to low dimensional data by selecting the most important features that capture maximum information about the dataset.
- The features are selected on the basis of variance that they cause in the output.
- The feature that causes highest variance is the first principal component. The feature that is responsible for second highest variance is considered the second principal component, and so on.
- It is important to mention that principal components do not have any correlation with each other.

Advantages of PCA

- The training time of the algorithms reduces significantly with less number of features.
- It is not always possible to analyze data in high dimensions. For instance if there are 100 features in a dataset. Total number of scatter plots required to visualize the data would be 100(100-1)2 = 4950. Practically it is not possible to analyze data this way.

Normalization of features

- It is imperative to mention that a feature set must be normalized before applying PCA. For instance if a feature set has data expressed in units of Kilograms, Light years, or Millions, the variance scale is huge in the training set. If PCA is applied on such a feature set, the resultant loadings for features with high variance will also be large. Hence, principal components will be biased towards features with high variance, leading to false results.
- Finally, the last point to remember before we start coding is that PCA is a statistical technique and can only be applied to numeric data. Therefore, categorical features are required to be converted into numerical features before PCA can be applied.

Factor Analysis

- These variables can be grouped by their correlations.. Here each group represents a single underlying construct or factor.
- These factors are small in number as compared to a large number of dimensions. However, these factors are difficult to observe.
- There are basically two methods of performing factor analysis:
  - EFA (Exploratory Factor Analysis)
  - CFA (Confirmatory Factor Analysis)

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com