

Data Preprocessing and ETL



Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- Split data to Train and Test Sets
- Deal with Imbalanced classes
- Feature Scaling

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

What is Data

We are living in a data-driven economy. It's a world where having tons of data, understanding it and knowing what to do with data is power.

Understanding your data is not one of the most difficult things in data science, but it is time-consuming.
Interpretation of data is effective when we know about the source of data.



What is Data (Types of Data)

Categorical

This represents qualitative data with no apparent inherent mathematical meaning.

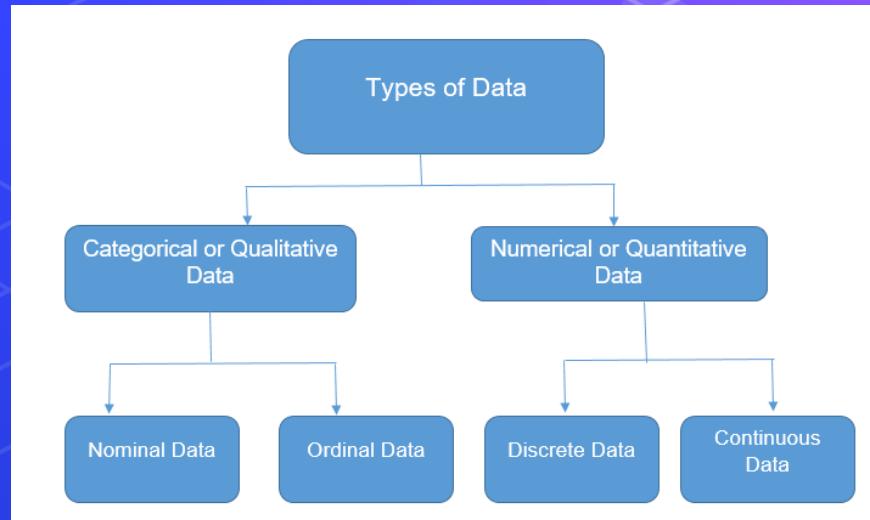
Example: Yes or No, Sex, Race, Marital Status etc.

These can be assigned numbers like Yes(0) and No(1), but numbers have no mathematical meaning.

Numerical

This represents some sort of quantitative measurement.

Example: height of people, stock price, page load time etc



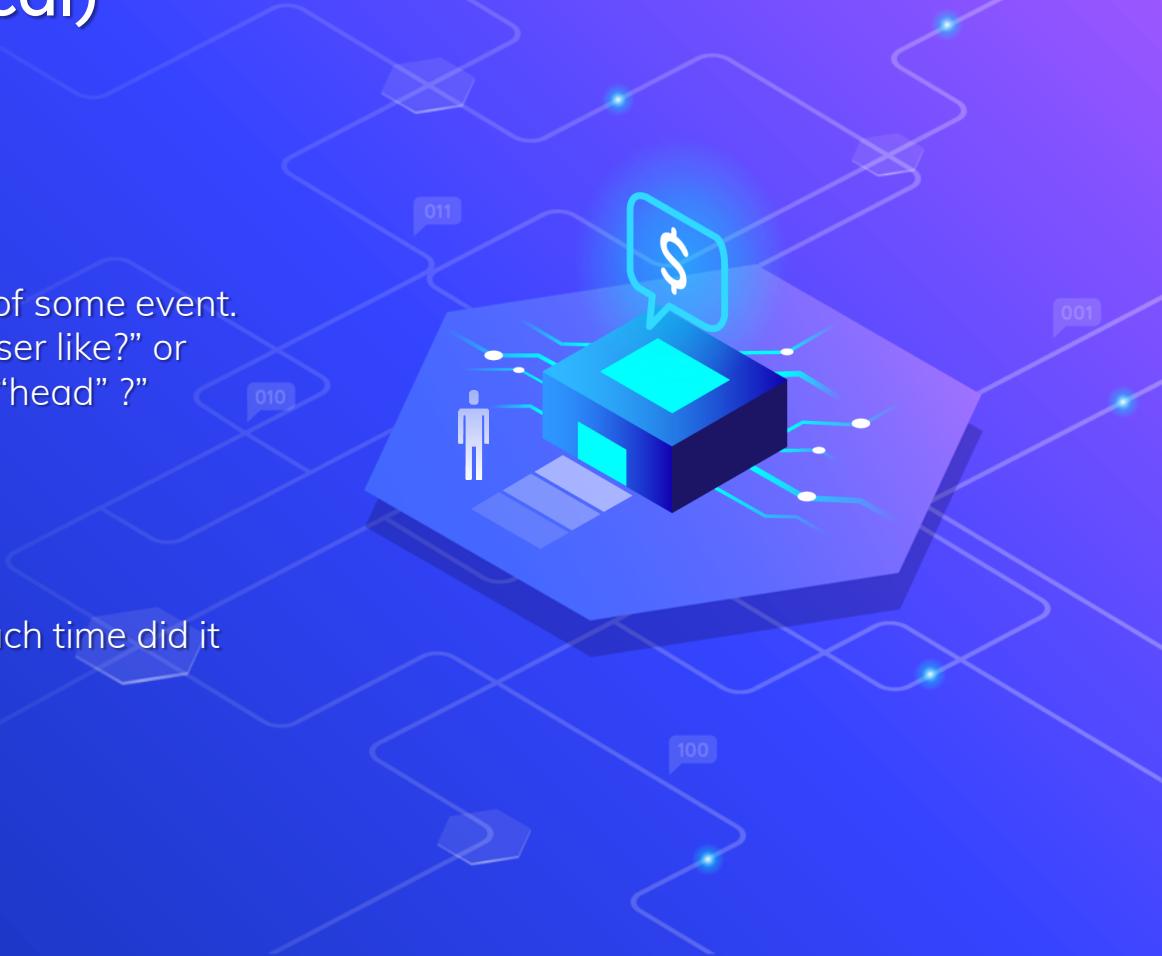
What is Data (Numerical)

Discrete data

This is integer based, often counts of some event.
Example: "How many songs do a user like?" or
"How many times a coin flipped to "head" ?"

Continuous data

It has an infinite number of
possible values. Example: "How much time did it
take for a user to buy?"



What is Data (Categorical)

Nominal or unordered data

we assign individual items to named categories that do not have an implicit or natural value or rank.

Example: Animals can be cats or dogs or horses etc.. that would be nominal data.

Ordinal or ordered data

items are assigned to categories that do have some kind of implicit or natural order.

Example: "Small, Medium, or Large." Another example is a survey question that asks us to rate an item on a 1 to 10 scale, with 10 being the best. This implies that 10 is better than 9, which is better than 8, and so on.



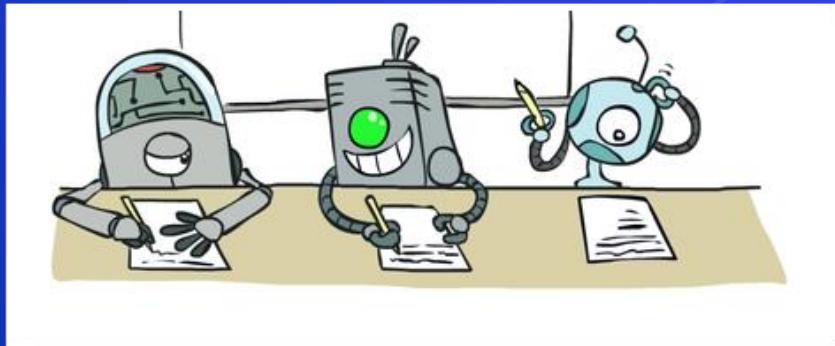
Agenda

- hexagon icon What is Data
- hexagon icon **Machine Learning**
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

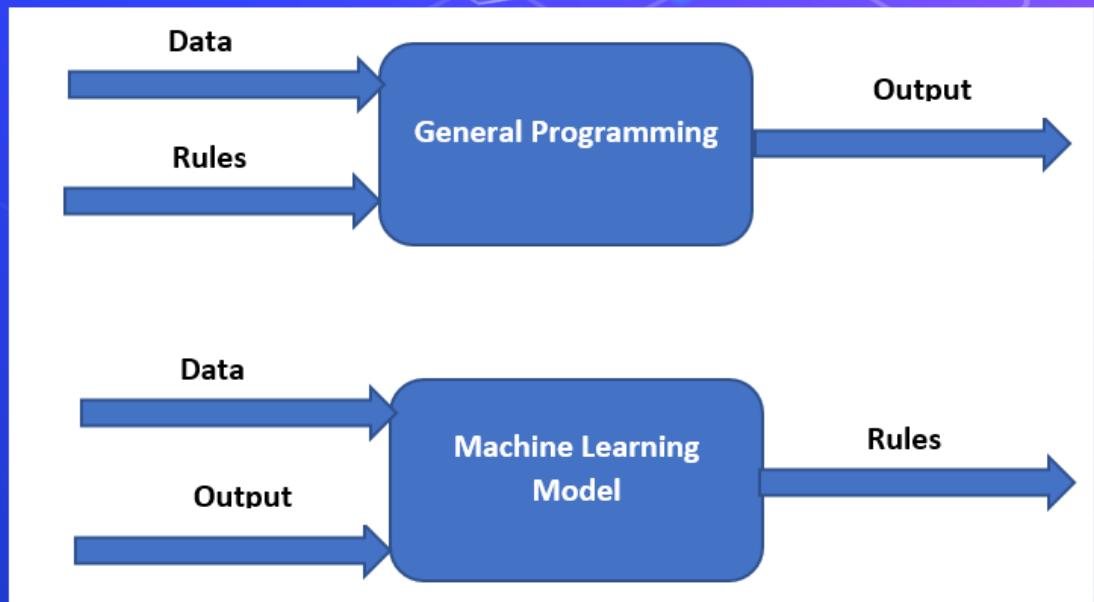
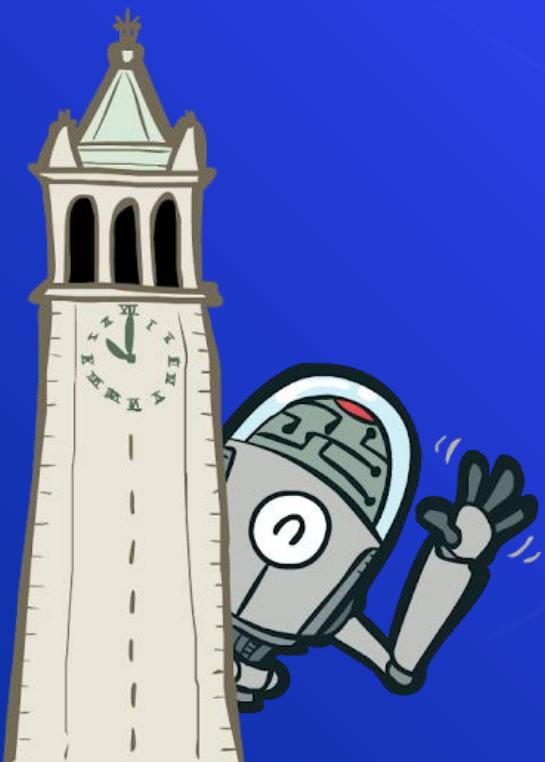
- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Machine Learning

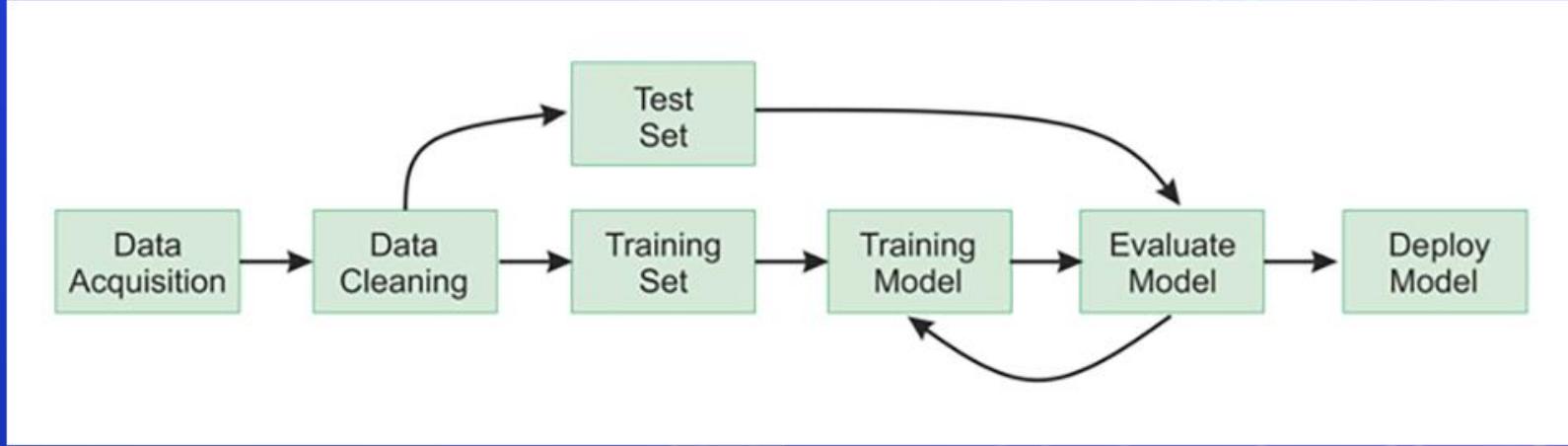
Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks.



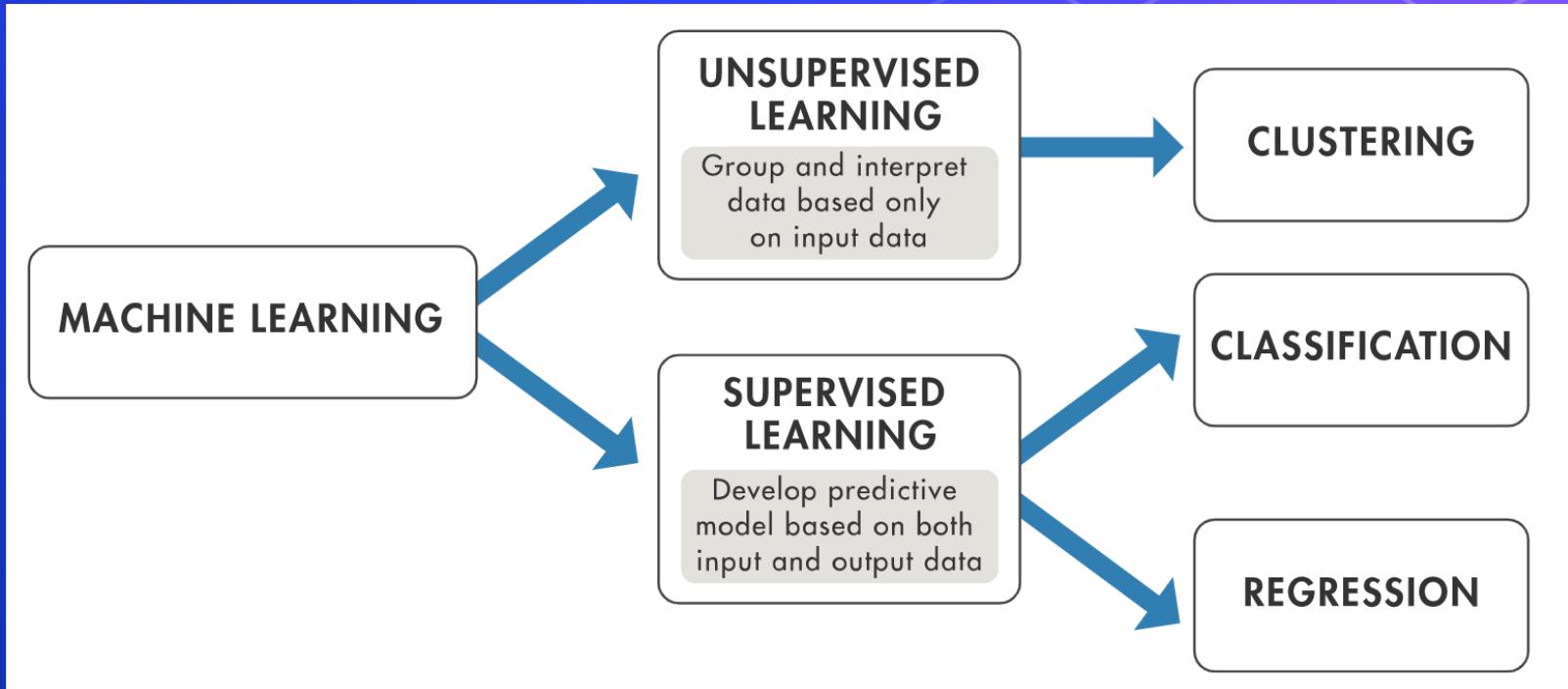
Machine Learning



Machine Learning



Machine Learning



Agenda

- What is Data
- Machine Learning
- Data Preprocessing
- Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

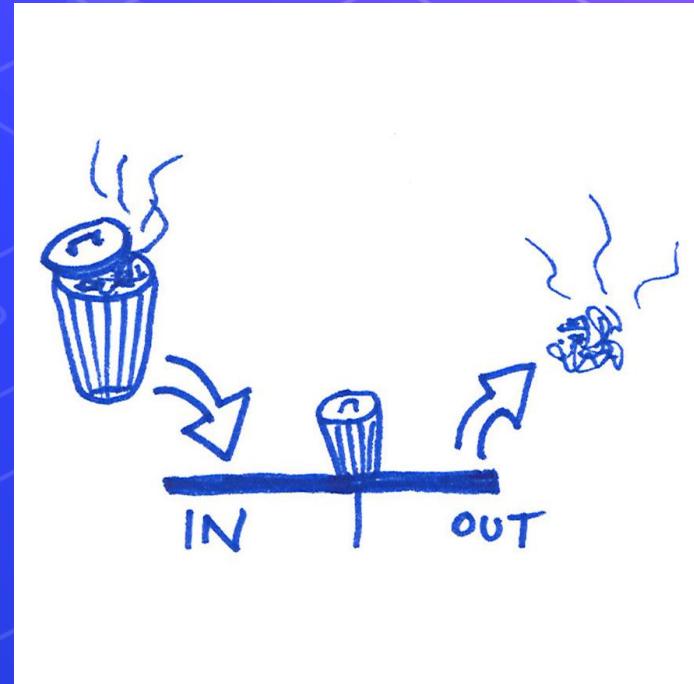
- Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Data Preprocessing

The phrase "garbage in, garbage out" is the best description for the use of data preprocessing.

Data gathering methods are often loosely controlled, resulting in out-of-range values, useless features, outliers, missing values, etc.

Garbage data produce misleading results. Thus, data preprocessing is the most important phase of a machine learning project.



ETL

E T L

Extract

Transform

Load



Data Preprocessing

Feature Engineering and Extraction

Extract and Create useful features from raw data into features suitable for modeling.

Feature Transformation

Transformation of data to improve the accuracy of the algorithm.



Agenda

- ❖ What is Data
- ❖ Machine Learning
- ❖ Data Preprocessing
- ❖ **Feature Engineering and Extraction**
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

- ❖ Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Domain knowledge features

It's the process of creating new useful features from current features based on our understanding of the problem domain knowledge.



```
1 # example for delivery system we calculate speed in Km/m  
2 df['speed'] = df['Distance (KM)'] / (df['Time from Pickup to Arrival'] / 60)
```

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon **Feature Engineering and Extraction**
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Date and Time features

Date columns usually provide valuable information about the problem, they are neglected as an input for the machine learning algorithms. It might be the reason for this, that dates can be present in numerous formats, which make it hard to understand by algorithms.



```
1 df['Time'] = pd.to_datetime(df['Time'],format='%d/%m/%Y %H:%M')
```

Date and Time features

```
1 df['Year'] = df['Time'].dt.year  
2 df['Month'] = df['Time'].dt.month  
3 df['Month_Name'] = df['Time'].dt.month_name()  
4 df['Week'] = df['Time'].dt.week  
5 df['Day'] = df['Time'].dt.day  
6 df['Week_Day'] = df['Time'].dt.weekday  
7 df['Day_Name'] = df['Time'].dt.day_name()  
8 df['Hour'] = df['Time'].dt.hour  
9 df['Minute'] = df['Time'].dt.minute
```

- Extract day, hour, minute, seconds, quarter, month, year, etc.
- Extract time-based features like evenings, noon, night time etc.
- Extract seasonal features like winter, summer, autumn.
- Calculate time elapsed between two related Date features.



Date and Time features

```
1 def map_hours(x):
2     if x in [0,1,2,3,4,5,6,7,8,9,10,11,12]:
3         return 'morning'
4     elif x in [13,14,15,16]:
5         return 'afternoon'
6     else:
7         return 'evening'
8
9 df['Period'] = df['Hour'].apply(map_hours)
```

- Extract day, hour, minute, seconds, quarter, month, year, etc.
- **Extract time-based features like evenings, noon, night time etc.**
- Extract seasonal features like winter, summer, autumn.
- Calculate time elapsed between two related Date features.



Date and Time features

```
1 def map_months(x):
2     if x in [12, 1, 2]:
3         return 'Winter'
4     elif x in [3, 4, 5]:
5         return 'Spring'
6     elif x in [6, 7, 8]:
7         return 'Summer'
8     elif x in [9, 10, 11]:
9         return 'Autumn'
10
11 df['Season'] = df['Month'].apply(map_months)
```

- Extract day, hour, minute, seconds, quarter, month, year, etc.
- Extract time-based features like evenings, noon, night time etc.
- **Extract seasonal features like winter, summer, autumn.**
- Calculate time elapsed between two related Date features.

Date and Time features



```
1 df['Elapsed_Years'] = (datetime.now() - df['Time']) / np.timedelta64(1, 'Y')
```

- Extract day, hour, minute, seconds, quarter, month, year, etc.
- Extract time-based features like evenings, noon, night time etc.
- Extract seasonal features like winter, summer, autumn.
- Calculate time elapsed between two related Date features.

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon **Feature Engineering and Extraction**
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

String operations

There is a lot of data hidden within strings, you can use a lot of string operations techniques to extract those data.

```
1 def extract_email_provider(email):
2     return email.split('@')[1]
3
4 df['Email_Provider'] = df['Email'].apply(extract_email_provider)
```

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon **Feature Engineering and Extraction**
 - Domain knowledge features
 - Date and Time features
 - String operations
 - **Web Data**
 - Geospatial features
 - Work with Text

- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Web Data

There are a lot of web applications data logs like user agents that contain a lot of info about used browser, OS and device.

```
1 import user_agents
2
3 ua = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
      Chrome/83.0.4103.116 Safari/537.36'
4 ua = user_agents.parse(ua)
5
6 print('Is a bot? ', ua.is_bot)
7 print('Is mobile? ', ua.is_mobile)
8 print('Is PC? ', ua.is_pc)
9 print('OS Family: ', ua.os.family)
10 print('OS Version: ', ua.os.version)
11 print('Browser Family: ', ua.browser.family)
12 print('Browser Version: ', ua.browser.version)
13 print('Device Family: ', ua.device.family)
14 print('Device Brand: ', ua.device.brand)
15 print('Device Model: ', ua.device.model)
```

Web Data

IP address is a great way to extract geographical data from it.

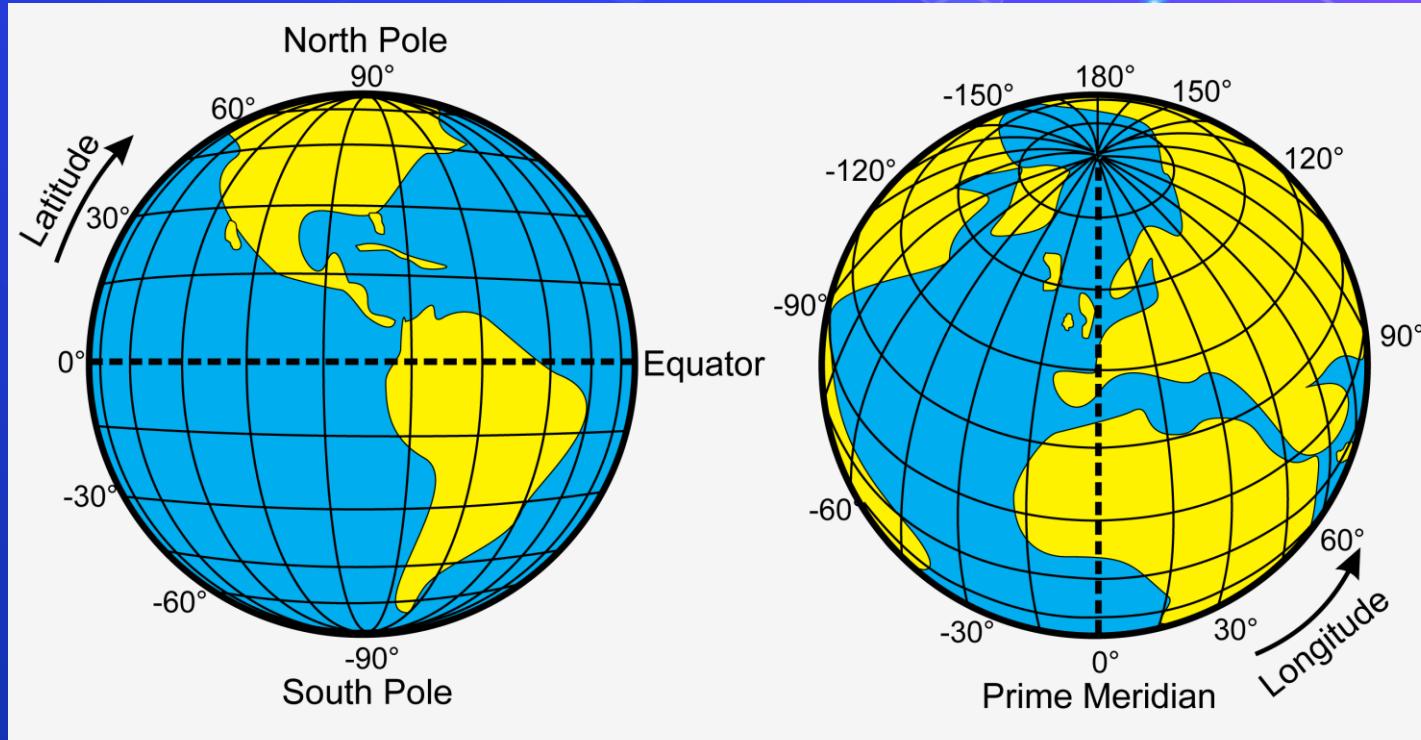
```
1 from ip2geotools.databases.noncommercial import DbIpCity as ip2geo
2
3
4 response = ip2geo.get('45.243.72.231', api_key='free')
5
6 print(response.ip_address)
7 print(response.city)
8 print(response.region)
9 print(response.country)
10 print(response.latitude)
11 print(response.longitude)
```

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon **Feature Engineering and Extraction**
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Geospatial features

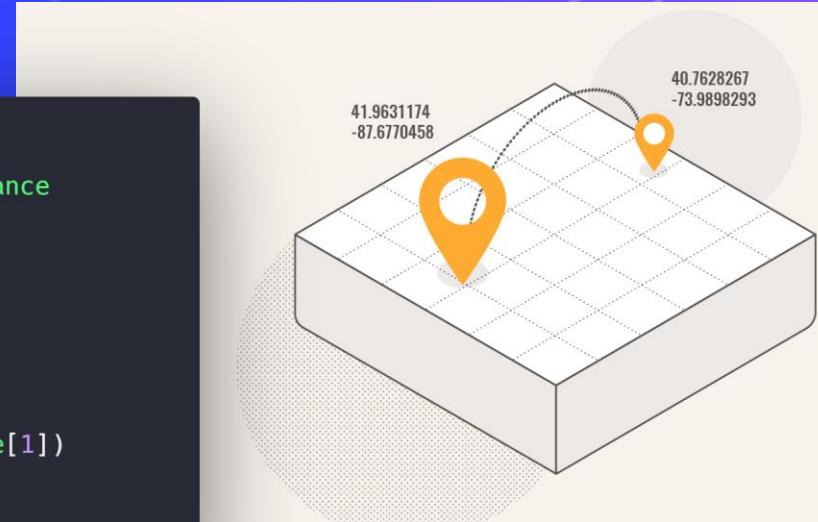


Geospatial features

Measure distance with `haversine_distance`

<https://www.movable-type.co.uk/scripts/latlong.html>

```
1 from datasist.feature_engineering import haversine_distance
2
3
4 # lat, long
5 my_home = [30.109919, 31.308797]
6 cafe = [30.120982, 31.322026]
7
8
9 haversine_distance(my_home[0], my_home[1], cafe[0], cafe[1])
10 """
11 1.769848
12 """
```



Geospatial features

You can also use Geocoding features to geolocate an address to coordinates, or you can get the reverse address city and country from coordinates.

<https://pypi.org/project/geopy>

<https://github.com/thampiman/reverse-geocoder>

```
1 from geopy.geocoders import Nominatim
2
3 geolocator = Nominatim(user_agent="specify_your_app_name_here")
4
5
6 location = geolocator.geocode("175 5th Avenue NYC")
7 print((location.latitude, location.longitude))
8 # (40.7410861, -73.9896297241625)
9
10 location = geolocator.reverse("52.509669, 13.376294")
11 print(location.address)
12 # Flatiron Building, 175, New York, NYC, New York, ...
13
```

Geospatial features

You can also use Zip Codes to extract valuable information about Locations, check this library.

<https://pypi.org/project/zipcodes/>

```
1 import zipcodes
2
3 print(zipcodes.matching('77429'))
4
5 """
6 {'acceptable_cities': [],
7  'active': True,
8  'area_codes': ['281', '832'],
9  'city': 'Cypress',
10 'country': 'US',
11 'county': 'Harris County',
12 'lat': '29.9857',
13 'Long': '-95.6548',
14 'state': 'TX',
15 'timezone': 'America/Chicago',
16 'unacceptable_cities': [],
17 'world_region': 'NA',
18 'zip_code': '77429',
19 'zip_code_type': 'STANDARD'}]
20 """
```

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon **Feature Engineering and Extraction**
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

- hexagon icon Feature Transformations
 - Data Cleaning or Cleansing
 - Work with Missing data
 - Work with Categorical data
 - Detect and Handle Outliers
 - Split data to Train and Test Sets
 - Deal with Imbalanced classes
 - Feature Scaling

Work with Text (Bag Of Words)

Bag of Words

Converts text to a matrix where every row is an observation and every feature is a unique word. The value of each element in the matrix is either a binary indicator marking the presence of that word or an integer of the number of times that word appears.

ChrisAlbon



```
1 from sklearn.feature_extraction.text import CountVectorizer  
2  
3 vectorizer = CountVectorizer(stop_words='english')  
4 txt_feats = vectorizer.fit_transform(text)  
5 txt_feats.to_array()
```

Work with Text (Bag Of Words)

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!
---------------------------	---	--

CountVectorizer



hi	every one	I'm	so	excited	about	this	course
1	1			1			
		1	1	1	1	1	1
		1	2	3			

Work with Text (TF-IDF)

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

Term frequency
Number of times term t appears in a doc, d

$$\log \frac{1 + n}{1 + df(d, t)}$$

Document frequency of the term t

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 vectorizer = TfidfVectorizer(stop_words='english')
4 txt_feats = vectorizer.fit_transform(text)
5 txt_feats.toarray()
```

Work with Text (TF-IDF)

TF (Term Frequency)

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!
---------------------------	---	--



hi	every one	I'm	so	excited	about	this	course
0.33	0.33			0.33			
		0.16	0.16	0.16	0.16	0.16	0.16
		0.16	0.33	0.5			

IDF (Inverse Document Frequency)

(excited) Hi everyone!	I'm so excited about this course!	So excited. SO EXCITED. EXCITED, I AM!
---------------------------	---	--



hi	every one	I'm	so	excited	about	this	course
0.36	0.36			0			
		0.06	0.06	0	0.18	0.18	0.18
		0.06	0.13	0			

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- Split data to Train and Test Sets
- Deal with Imbalanced classes
- Feature Scaling

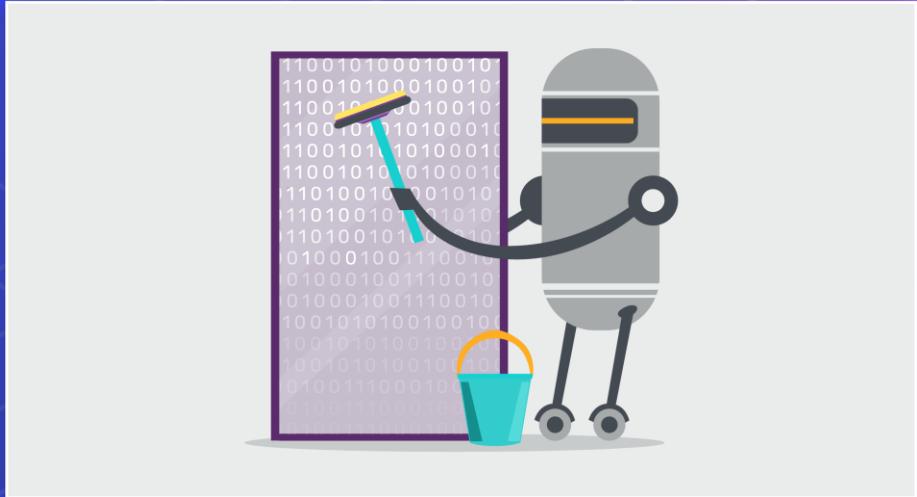
Data Cleaning or Cleansing

Cleaning data is the process of preparing the dataset for analysis. It is very important because the accuracy of machine learning or data mining models are affected because of poor quality of data.

So, data scientists spend a large amount of their time cleaning the dataset and transform them into a format with which they can work with. In fact, data scientists spend 80% of their time cleaning the data.

Example of problems

- Columns can have missing data indicators like xx and ?.
- Height column can have value of 0.
- Weight column can have negative values.
- ...



Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- Split data to Train and Test Sets
- Deal with Imbalanced classes
- Feature Scaling

Work with Missing data



```
1 df.isnull().sum()
```

- [Check Missing Data](#)
- [Drop Missing Data](#)
- [Fill Missing data with Pandas](#)
- [Fill Missing data with Sklearn SimpleImputer](#)
- [Fill Missing data with Sklearn KNNImputer](#)



Work with Missing data



```
1 df.dropna(axis = 0)  
2  
3 df.dropna(axis = 1)
```

- Check Missing Data
- [Drop Missing Data](#)
- Fill Missing data with Pandas
- Fill Missing data with Sklearn `SimpleImputer`
- Fill Missing data with Sklearn `KNNImputer`



Work with Missing data

```
1 # fill with mean  
2 df['BuildingArea'].fillna(df['BuildingArea'].mean(), inplace=True)  
3  
4  
5 # fill with median  
6 df['YearBuilt'].fillna(df['YearBuilt'].median(), inplace=True)  
7  
8  
9 # fill with mode (most frequent)  
10 df['Car'].fillna(df['Car'].mode()[0], inplace=True)
```

- Check Missing Data
- Drop Missing Data
- [Fill Missing data with Pandas](#)
- Fill Missing data with Sklearn SimpleImputer
- Fill Missing data with Sklearn KNNImputer



Work with Missing data

```
1 from sklearn.impute import SimpleImputer  
2  
3 # define the Imputer properties  
4 imputer = SimpleImputer(strategy='mean')      # can use 'median' or 'most_frequent'  
5  
6 # impute  
7 df['BuildingArea'] = imputer.fit_transform(df[['BuildingArea']])  
8  
9 # get the filled value  
10 imputer.statistics_
```

- Check Missing Data
- Drop Missing Data
- Fill Missing data with Pandas
- Fill Missing data with Sklearn SimpleImputer**
- Fill Missing data with Sklearn KNNImputer

Work with Missing data

```
1 from sklearn.impute import KNNImputer  
2  
3 imputer = KNNImputer()  
4  
5 df['BuildingArea'] = imputer.fit_transform(df[['BuildingArea']])
```

- Check Missing Data
- Drop Missing Data
- Fill Missing data with Pandas
- Fill Missing data with Sklearn **SimpleImputer**
- Fill Missing data with Sklearn KNNImputer**

Agenda

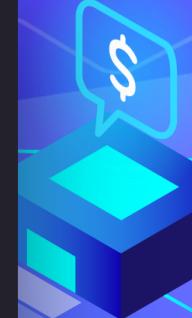
- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- Split data to Train and Test Sets
- Deal with Imbalanced classes
- Feature Scaling

Work with Categorical data

```
1 size_dict = {'XS':1,  
2      'S':2,  
3      'M':3,  
4      'L':4,  
5      'XL':5,  
6      'XXL':6}  
7  
8 # apply using map  
9 df['Size'] = df['Size'].map(size_dict)
```



- Work with Ordinal Features with pandas `map` method.
- Work with Nominal Features with pandas `get_dummies` method.

Work with Categorical data

ONE-HOT ENCODING

Feature	Apple	Pear
Apple	1	0
Pear	0	1
Apple	1	0
Pear	0	1
Apple	1	0

One-hot encoding allows us to turn nominal categorical data into features with numerical values, while not mathematically implying any ordinal relationship between the classes.

ChrisAlbon

```
1 df = pd.get_dummies(df, columns=['Color', 'Brand'], drop_first=True)
```

- Work with Ordinal Features with pandas `map` method.
- Work with Nominal Features with pandas `get_dummies` method.

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

hexagon icon Feature Transformations

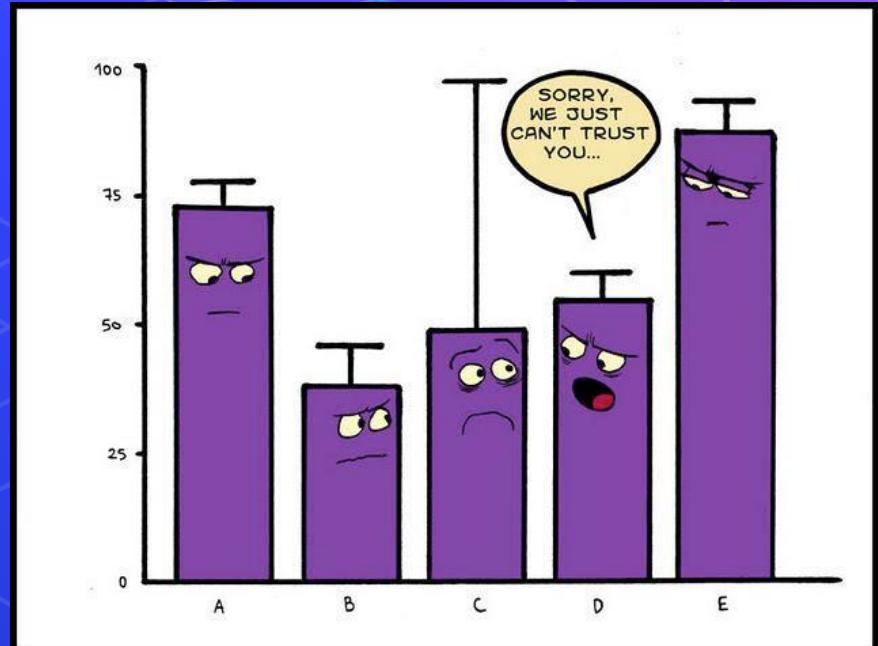
- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- **Detect and Handle Outliers**
- Split data to Train and Test Sets
- Deal with Imbalanced classes
- Feature Scaling

Detect and Handle Outliers

In statistics, an outlier is a data point that differs significantly from other observations.

An outlier may be due to variability in the measurement or it may indicate experimental error, the latter are sometimes excluded from the data set.

An outlier can cause serious problems in statistical analyses.



Detect and Handle Outliers



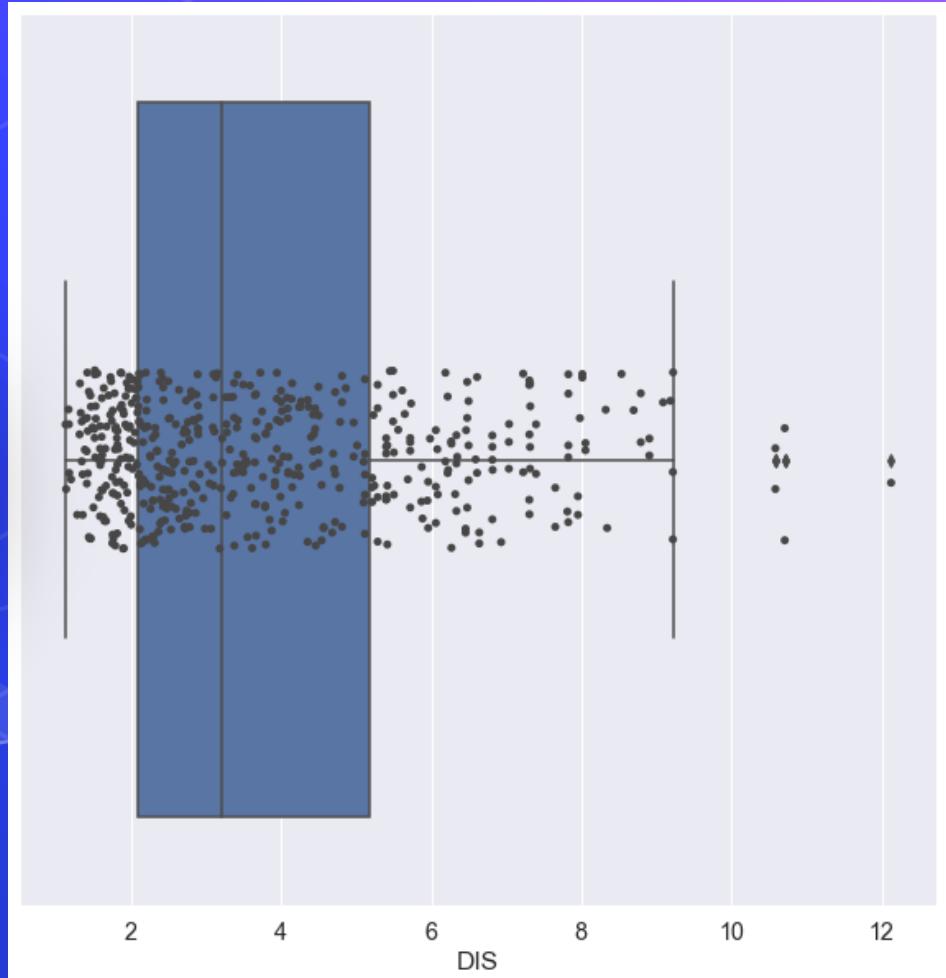
```
1 sns.boxplot(x='DIS', data=df)
2 sns.stripplot(x='DIS', data=df)
```



[Detect Outliers with Visualization.](#)



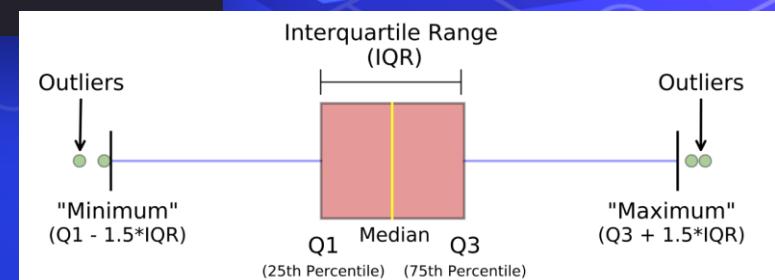
Detect and Handle Outliers with IQR.



Detect and Handle Outliers

```
1 from datasist.structdata import detect_outliers  
2  
3 # remove outliers  
4 outliers_indices = detect_outliers(df, 0, df.columns)  
5 df.drop(outliers_indices, inplace=True)  
6  
7  
8 # replace outliers with median value for each column  
9 for col in df.columns:  
10     outliers_indices = detect_outliers(df, 0, [col])  
11     col_median = df[col].median()  
12     df[col].iloc[outliers_indices] = col_median
```

- Detect Outliers with Visualization.
- Detect and Handle Outliers with IQR.



Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- **Split data to Train and Test Sets**
- Deal with Imbalanced classes
- Feature Scaling

Split data to Train and Test Sets

When you're working on a model and want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset.

the obvious solution is to split the dataset you have into two sets, one for training and the other for testing; and you do this before you start training your model.

	weight	height	drinks alcohol	healthy
0	112	181	0	0
1	123	165	1	1
2	176	167	1	1
3	145	154	X_train	1
4	198	181	0	0
5	211	202	1	0
6	145	201	1	1
7	181	153	1	1
8	90	142	0	1
9	101	169	X_test	1

Y_train

Y_test

```
1 from sklearn.model_selection import train_test_split  
2  
3 x = df.drop('healthy', axis=1)  
4 y = df['healthy']  
5  
6 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

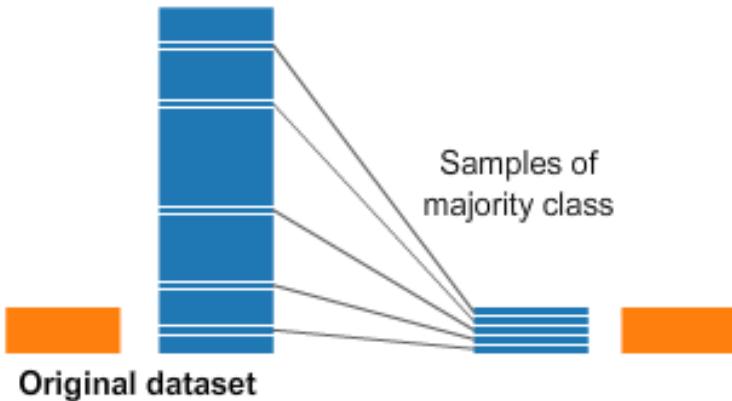
hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- Split data to Train and Test Sets
- **Deal with Imbalanced classes**
- Feature Scaling

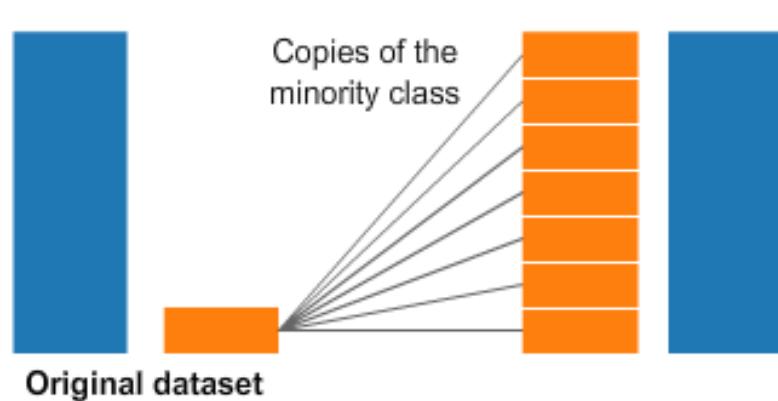
Deal with Imbalanced classes

Imbalanced classes are a common problem in machine learning classification where there are an unbalanced ratio of observations in each class. Class imbalance can be found in many different areas including medical diagnosis, spam filtering, and fraud detection.

Undersampling



Oversampling



Deal with Imbalanced classes

...

```
1 from sklearn.utils import resample  
2  
3 not_fraud = X[X['Class'] == 0]  
4 fraud = X[X['Class'] == 1]  
5  
6 not_fraud_downsampled = resample(not_fraud, replace=False  
7                                     , n_samples = len(fraud))  
8  
9 downsampled = pd.concat([not_fraud_downsampled, fraud])
```

Downsampling

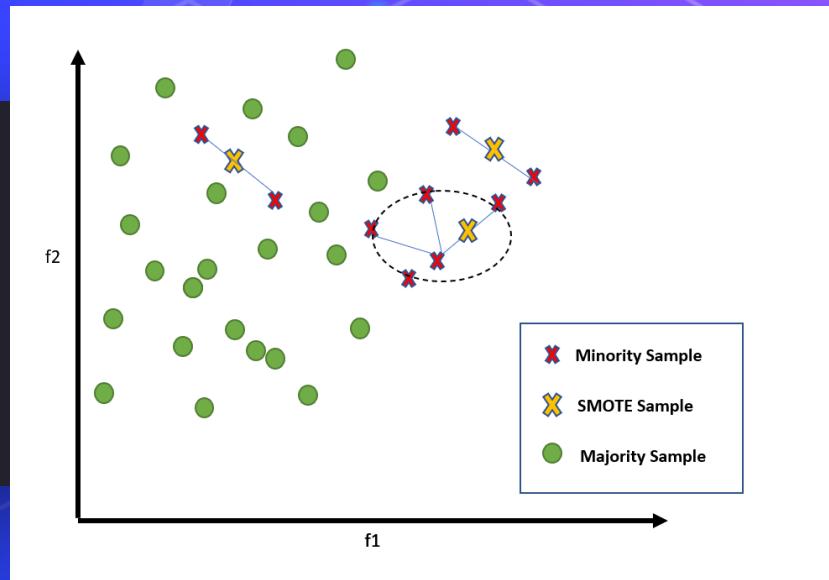
A strategy to handle imbalanced classes by creating a random subset of the majority of equal size to the minority class.

ChrisAlbon

- ▷ Down-sampling or Under-sampling Majority Class
- ▷ Generate Synthetic Samples with SMOTE

Deal with Imbalanced classes

```
● ● ●  
1 from imblearn.over_sampling import SMOTE  
2  
3 x = df.drop('Class', axis=1)  
4 y = df['Class']  
5  
6 smote = SMOTE()  
7  
8 x_train, y_train = smote.fit_sample(x_train, y_train)
```



- ▷ Down-sampling or Under-sampling Majority Class
- ▷ Generate Synthetic Samples with SMOTE

Agenda

- hexagon icon What is Data
- hexagon icon Machine Learning
- hexagon icon Data Preprocessing
- hexagon icon Feature Engineering and Extraction
 - Domain knowledge features
 - Date and Time features
 - String operations
 - Web Data
 - Geospatial features
 - Work with Text

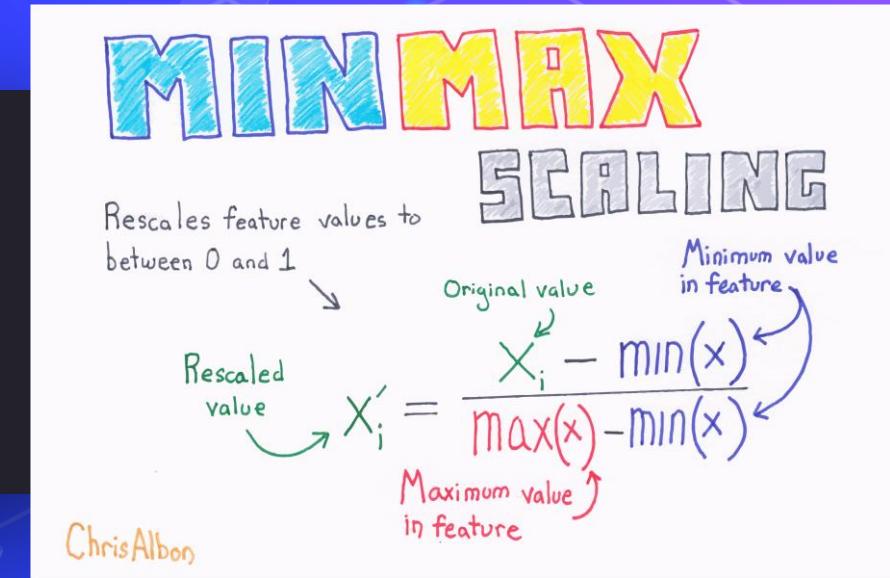
hexagon icon Feature Transformations

- Data Cleaning or Cleansing
- Work with Missing data
- Work with Categorical data
- Detect and Handle Outliers
- Split data to Train and Test Sets
- Deal with Imbalanced classes
- Feature Scaling

Feature Scaling

```
● ● ●  
1 from sklearn.preprocessing import MinMaxScaler  
2  
3 scaler = MinMaxScaler()  
4  
5 scaler.fit(x_train)  
6  
7 scaled_x_train = scaler.transform(x_train)  
8 scaled_x_test = scaler.transform(x_test)
```

- Normalization with Sklearn MinMaxScaler.
- Standardizing data with StandardScaler.



Feature Scaling

```
● ● ●  
1 from sklearn.preprocessing import StandardScaler  
2  
3 scaler = StandardScaler()  
4  
5 scaler.fit(x_train)  
6  
7 scaled_x_train = scaler.transform(x_train)  
8 scaled_x_test = scaler.transform(x_test)
```

- Normalization with Sklearn MinMaxScaler.
- Standardizing data with StandardScaler.

STANDARDIZATION

$$\tilde{x}_i = \frac{x_i - \bar{x}}{\sigma}$$

Standardized feature value
 \tilde{x}_i ← Value of the i th observation
 x_i ← Mean of the feature vector
 σ ← Standard deviation of the feature vector

Standardization is a common scaling method. \tilde{x}_i represents the number of standard deviations each value is from the mean value. It rescales a feature to have a mean of 0 and unit variance.

Chris Albon

Questions ?!



Thanks!

>_ Live long and prosper

