Data Preprocessing and ETL

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

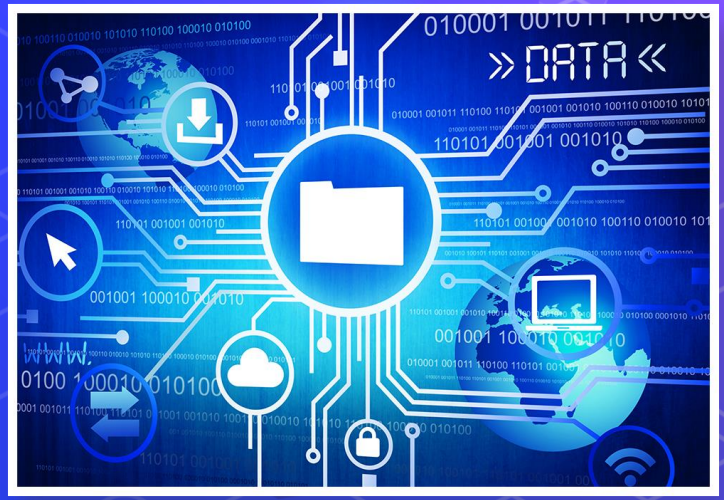
Agenda

- **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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

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.



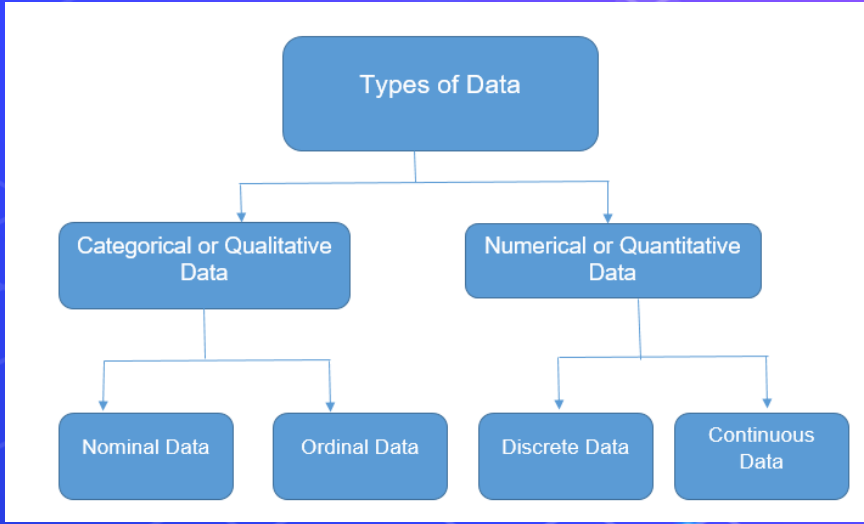
What is Data (Types of Data)

Categorical

This represents qualitative data with no apparent inherent mathematical meaning.
Example: Yes or No, Sex, Race, Marital Status etc.
These can take 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.

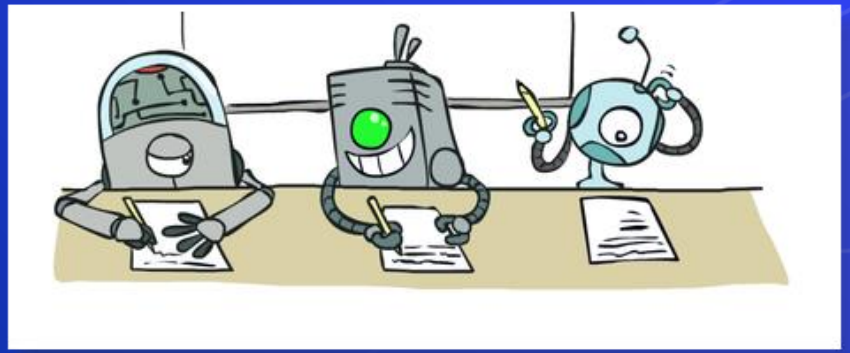
Ordinal or ordered data
items are assigned to categories that do have some kind of implicit or natural order.
Example: “Small, Medium, or Large." Another example is a survey question that asks us to rate an item on a 1 to 10 scale, with 10 being the best. This implies that 10 is better than 9, which is better than 8, and so on.

Agenda

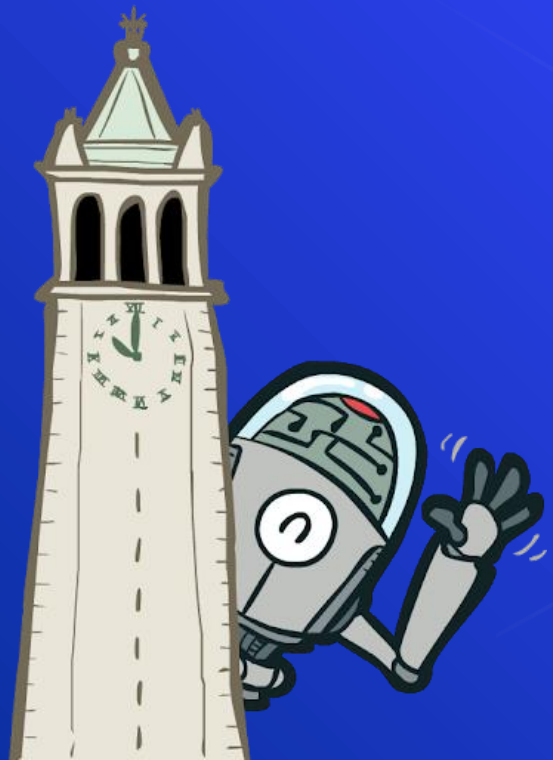
- What is Data
- **Machine Learning**
- Data Preprocessing
- Feature Engineering and Extraction
  - Domain knowledge features
  - Date and Time features
  - String operations
  - Web Data
  - Geospatial features
  - Work with Text
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

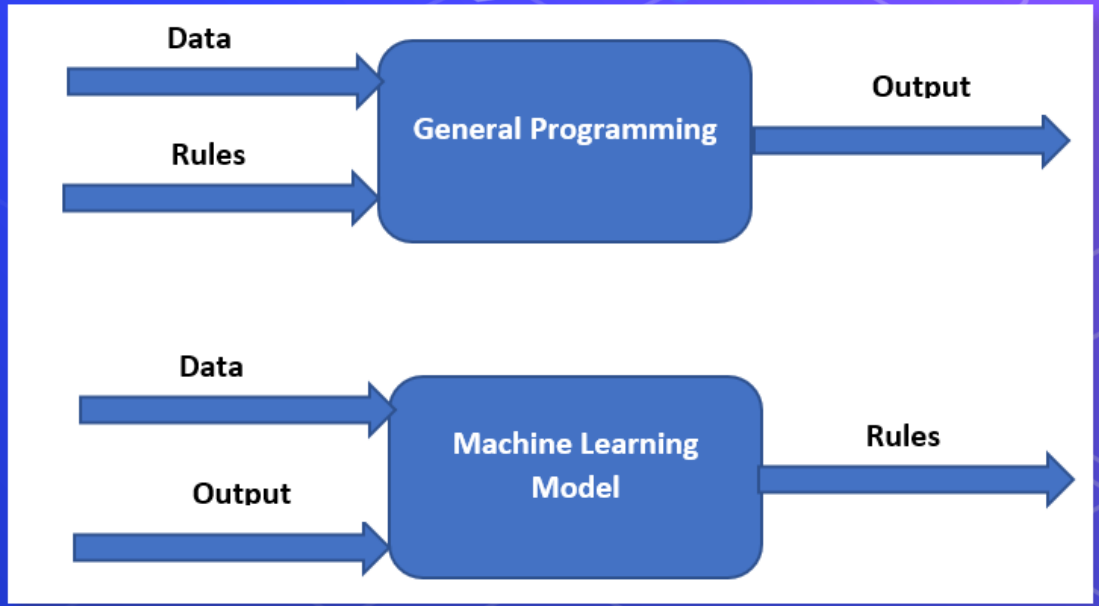
Machine Learning

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.

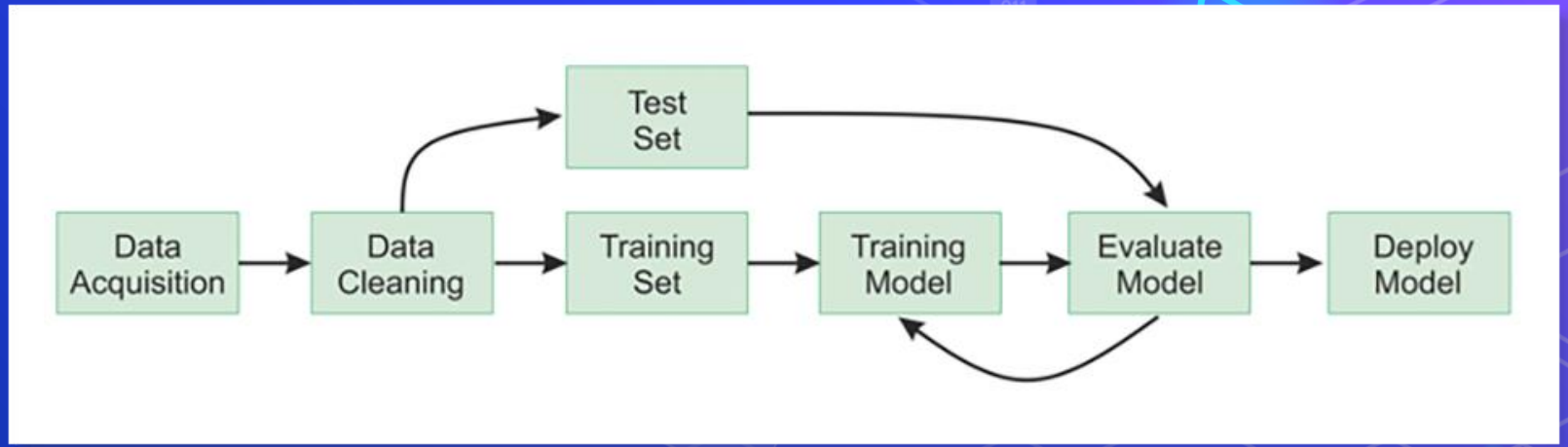


Machine Learning

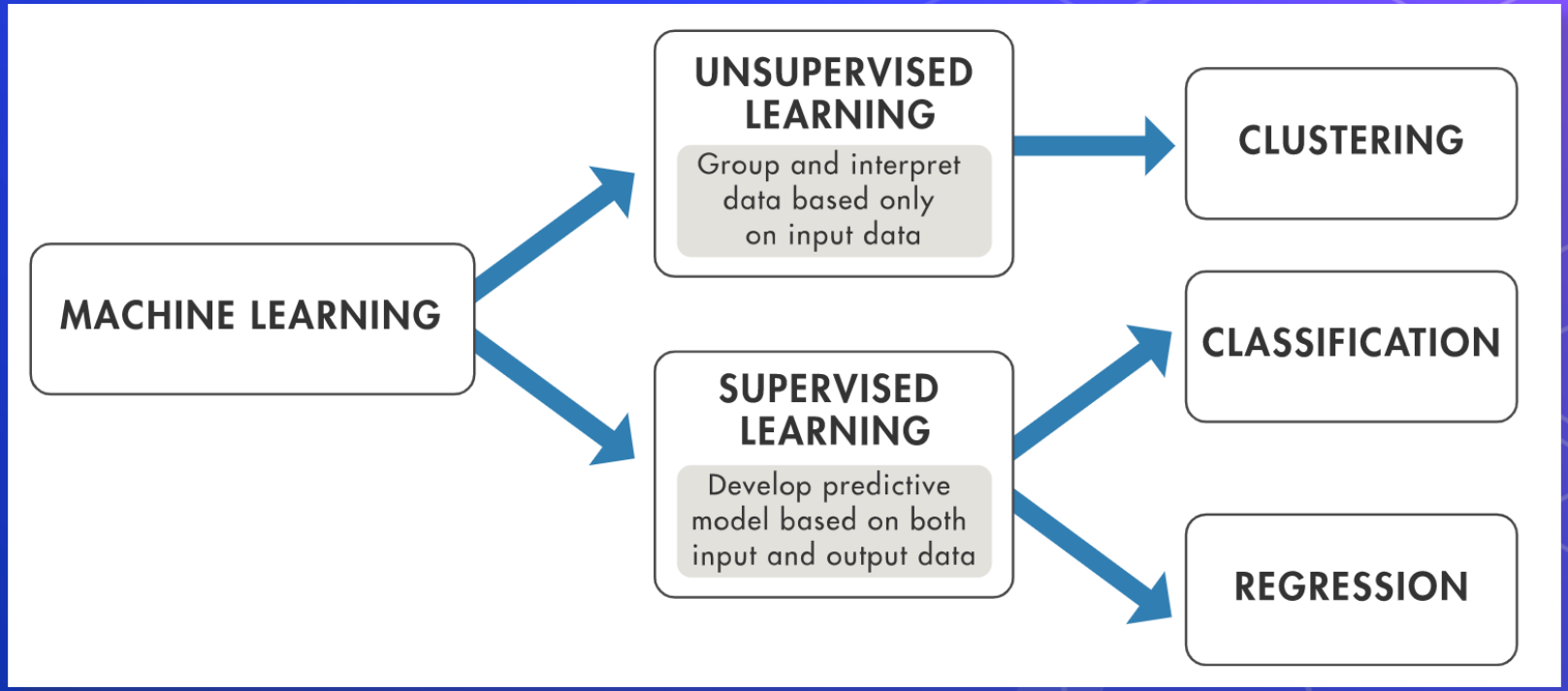




Machine Learning



Machine Learning



Agenda

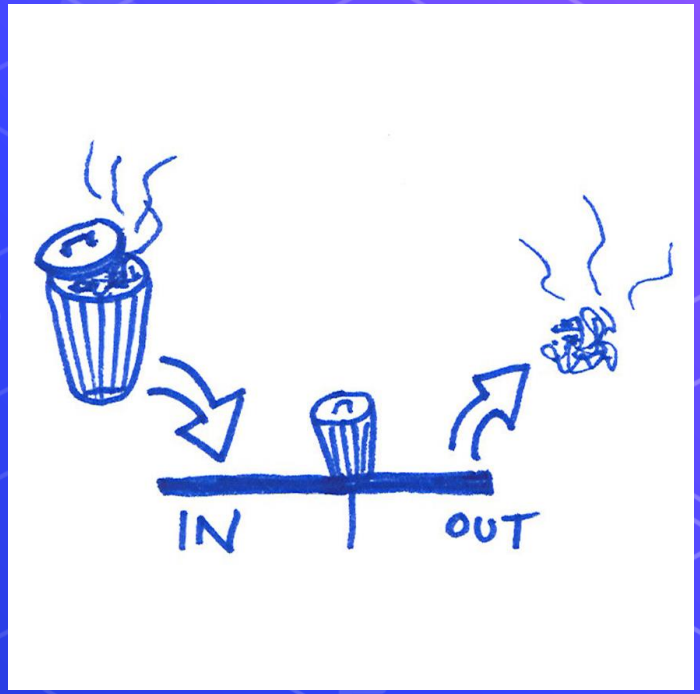
- What is Data
- Machine Learning
- **Data Preprocessing**
- Feature Engineering and Extraction
  - Domain knowledge features
  - Date and Time features
  - String operations
  - Web Data
  - Geospatial features
  - Work with Text
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

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.

Feature Selection
Removing unnecessary features.

Data Preprocessing

Install

```
>_ conda install scikit-learn
```

Use

```
# example for importing train_test_split function
from sklearn.model_selection import train_test_split
```

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

Domain knowledge features

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

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

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

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.

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

Date and Time features

```python
df['Year'] = df['Time'].dt.year
df['Month'] = df['Time'].dt.month
df['Month_Name'] = df['Time'].dt.month_name()
df['Week'] = df['Time'].dt.week
df['Day'] = df['Time'].dt.day
df['Week_Day'] = df['Time'].dt.weekday
df['Day_Name'] = df['Time'].dt.day_name()
df['Hour'] = df['Time'].dt.hour
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

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

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.

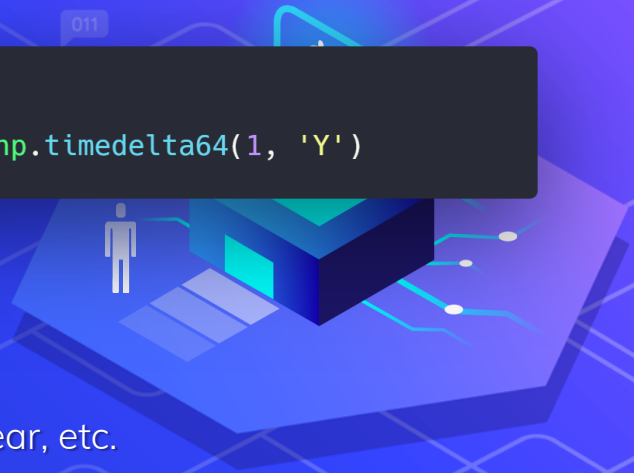
Date and Time features

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

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

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



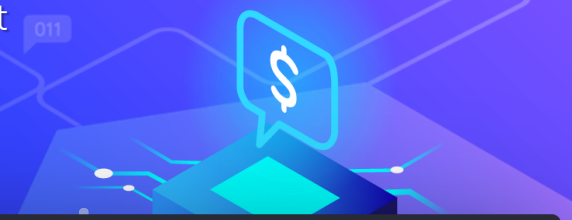
- Extract day, hour, minute, seconds, quarter, month, year, etc.
- Extract time-based features like evenings, noon, night time etc.
- Extract seasonal features like winter, summer, autumn.
- **Calculate time elapsed between two related Date features.**

Agenda

- 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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

String operations

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



```python
def extract_email_provider(email):
    return email.split('@')[1]

df['Email_Provider'] = df['Email'].apply(extract_email_provider)
```

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

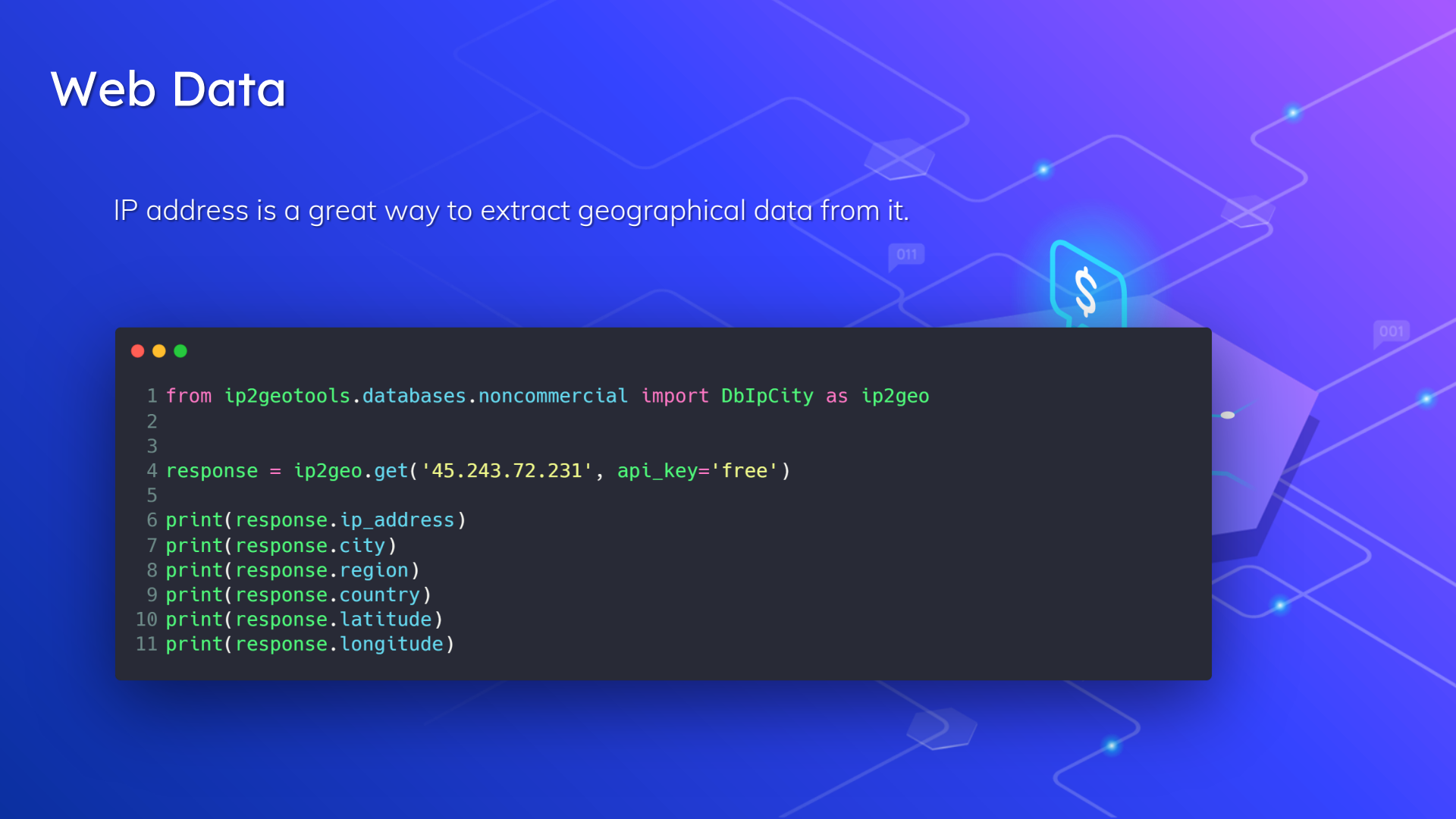
Web Data

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

```python
import user_agents

ua = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36'
ua = user_agents.parse(ua)

print('Is a bot? ', ua.is_bot)
print('Is mobile? ', ua.is_mobile)
print('Is PC? ',ua.is_pc)
print('OS Family: ',ua.os.family)
print('OS Version: ',ua.os.version)
print('Browser Family: ',ua.browser.family)
print('Browser Version: ',ua.browser.version)
print('Device Family: ',ua.device.family)
print('Device Brand: ',ua.device.brand)
print('Device Model: ',ua.device.model)
```

Web Data

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

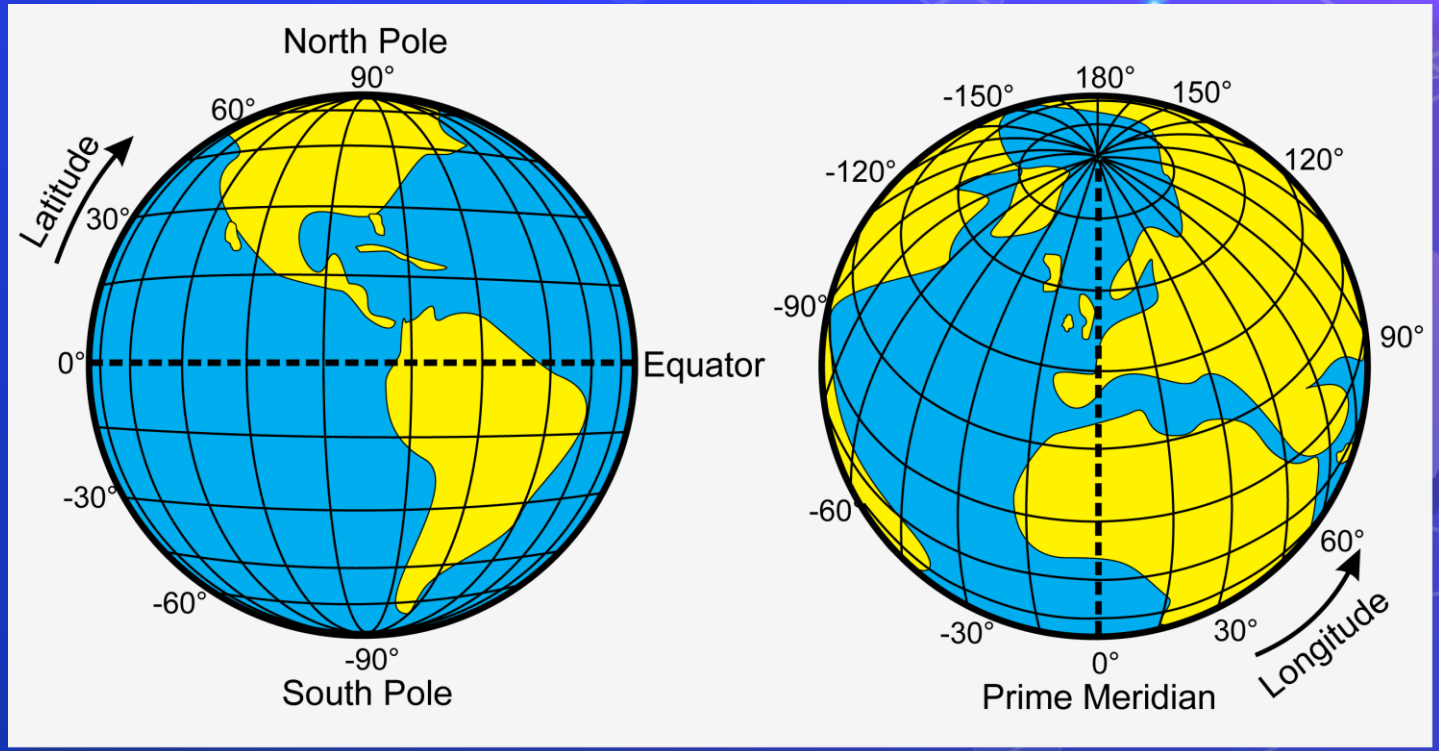


```python
from ip2geotools.databases.noncommercial import DbIpCity as ip2geo


response = ip2geo.get('45.243.72.231', api_key='free')

print(response.ip_address)
print(response.city)
print(response.region)
print(response.country)
print(response.latitude)
print(response.longitude)
```

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

Geospatial features

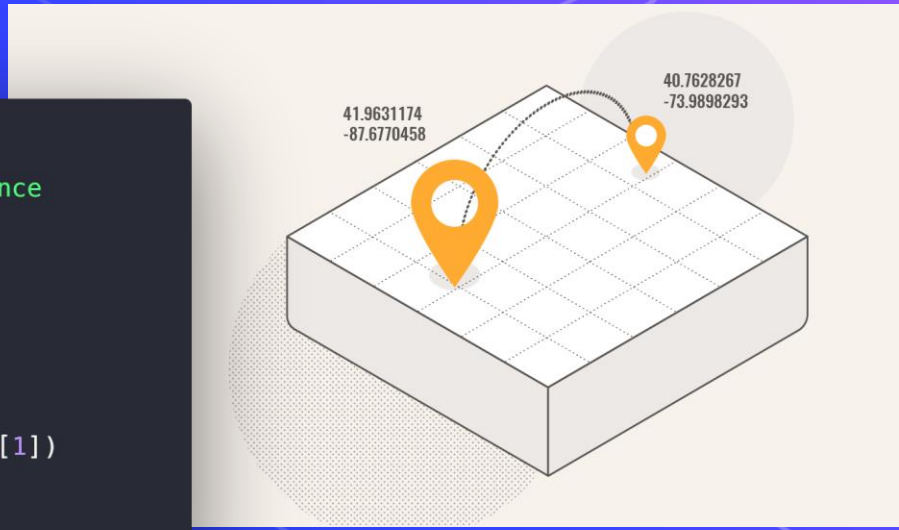


Geospatial features

Measure distance with `haversine_distance`

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

```python
from datasist.feature_engineering import haversine_distance


# lat, long
my_home = [30.109919, 31.308797]
cafe = [30.120982, 31.322026]


haversine_distance(my_home[0], my_home[1], cafe[0], cafe[1])
"""
1.769848
"""
```



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

```python
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="specify_your_app_name_here")


location = geolocator.geocode("175 5th Avenue NYC")
print((location.latitude, location.longitude))
# (40.7410861, -73.9896297241625)

location = geolocator.reverse("52.509669, 13.376294")
print(location.address)
# Flatiron Building, 175, New York, NYC, New York, ...

```

Geospatial features

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

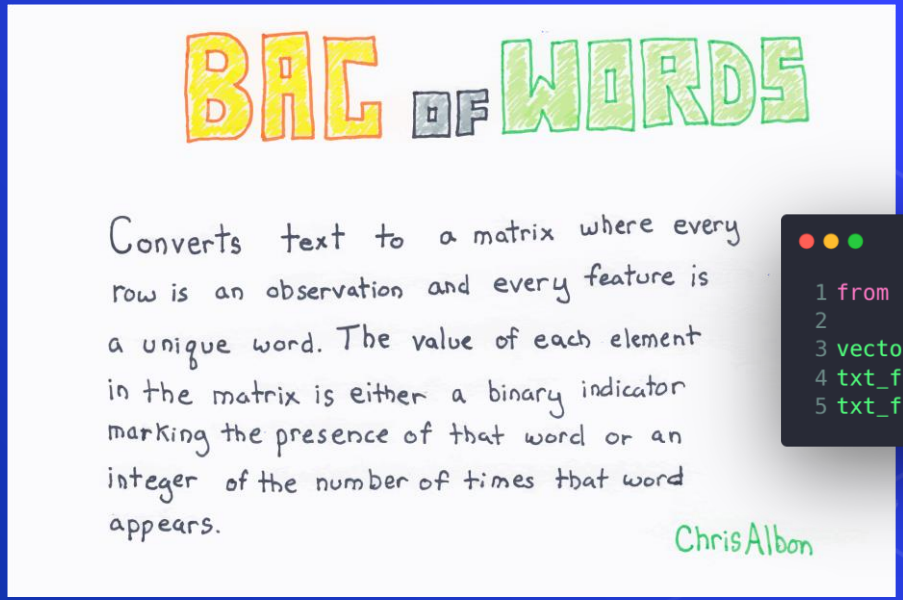
https://pypi.org/project/zipcodes/

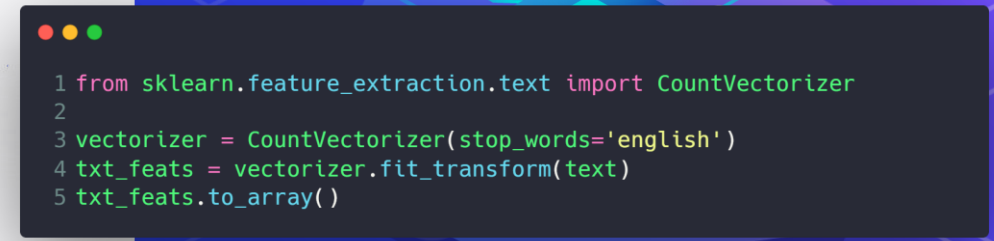
```
import zipcodes

print(zipcodes.matching('77429'))

"""
[{'acceptable_cities': [],
  'active': True,
  'area_codes': ['281', '832'],
  'city': 'Cypress',
  'country': 'US',
  'county': 'Harris County',
  'lat': '29.9857',
  'long': '-95.6548',
  'state': 'TX',
  'timezone': 'America/Chicago',
  'unacceptable_cities': [],
  'world_region': 'NA',
  'zip_code': '77429',
  'zip_code_type': 'STANDARD'}]
"""
```

Agenda

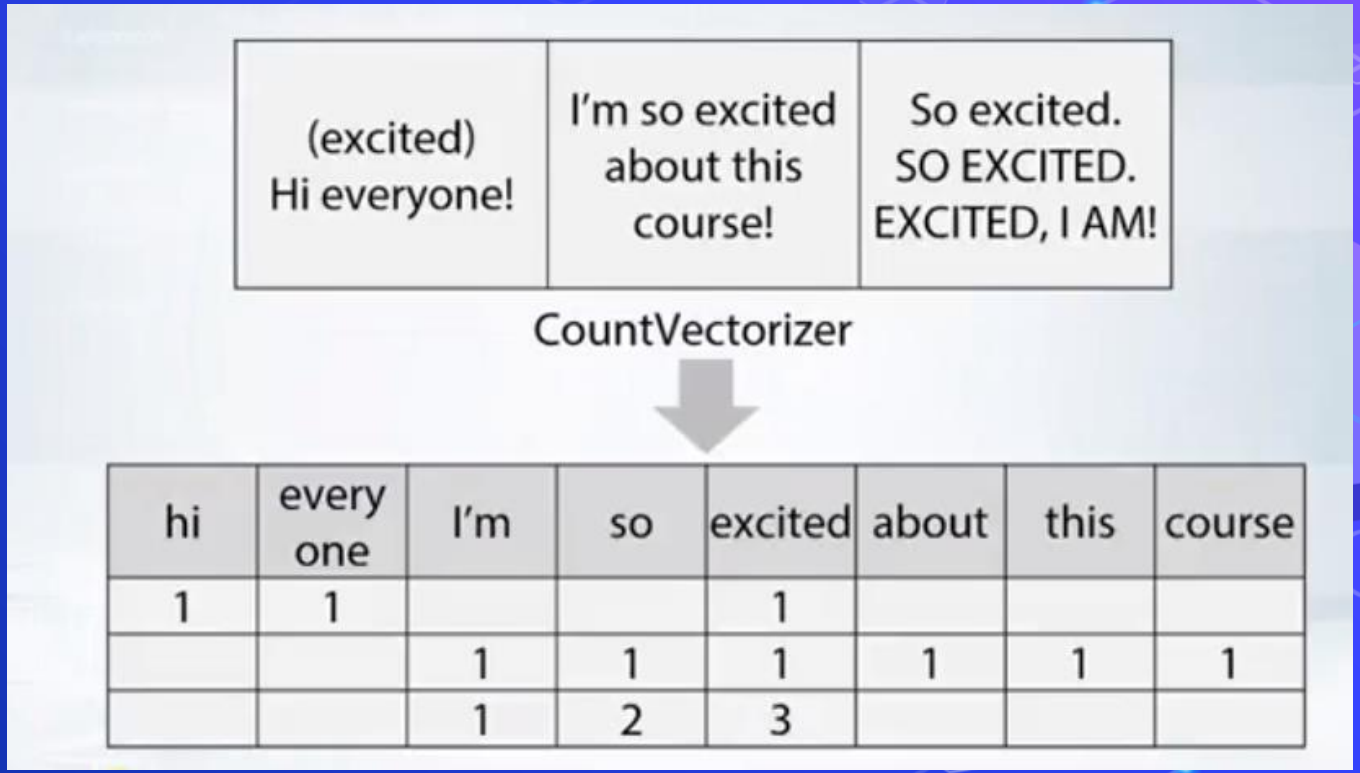
- What is Data
- Machine Learning
- Data Preprocessing
- **Feature Engineering and Extraction**
  - Domain knowledge features
  - Date and Time features
  - String operations
  - Web Data
  - Geospatial features
  - **Work with Text**
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

Work with Text (Bag Of Words)



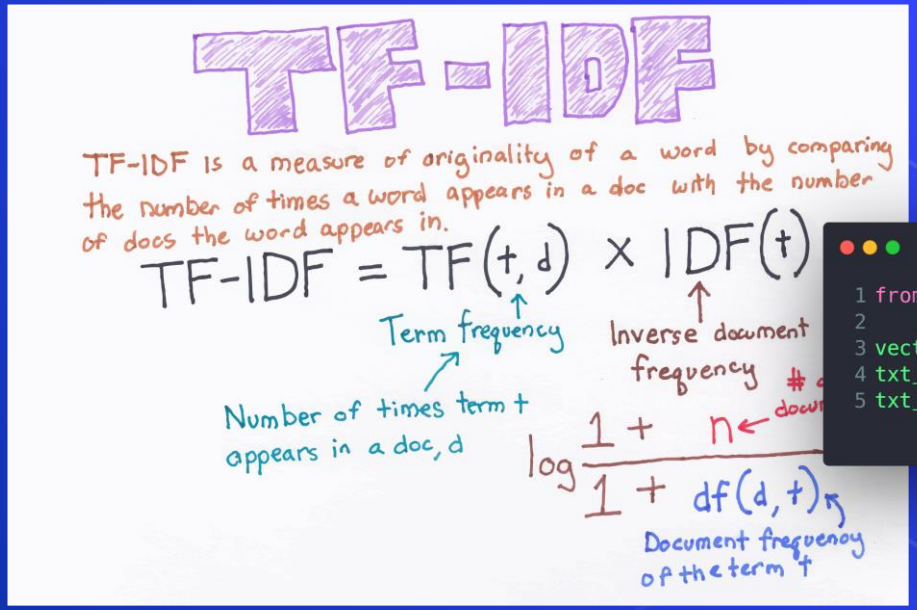


Work with Text (Bag Of Words)



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)



```python
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english')
txt_feats = vectorizer.fit_transform(text)
txt_feats.to_array()
```

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

- 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
  - **Work with Images**
  - Work with Audio
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

Work with Images

What We See

08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08 08 02 22 97
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00 49 49 99 40
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65 81 49 31 73
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91 52 70 95 23
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80 22 31 16 71
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50 24 47 32 60
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70 32 98 81 28
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21 67 26 20 68
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72 24 55 58 05
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95 21 36 23 09
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92 78 17 53 28
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57 16 39 05 42
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58 86 56 00 48
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40 19 80 81 68
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66 04 52 08 83
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69 88 36 68 87
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36 04 42 16 73
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16 20 69 36 41
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54 20 73 35 29
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48 01 70 54 71

What Computers See

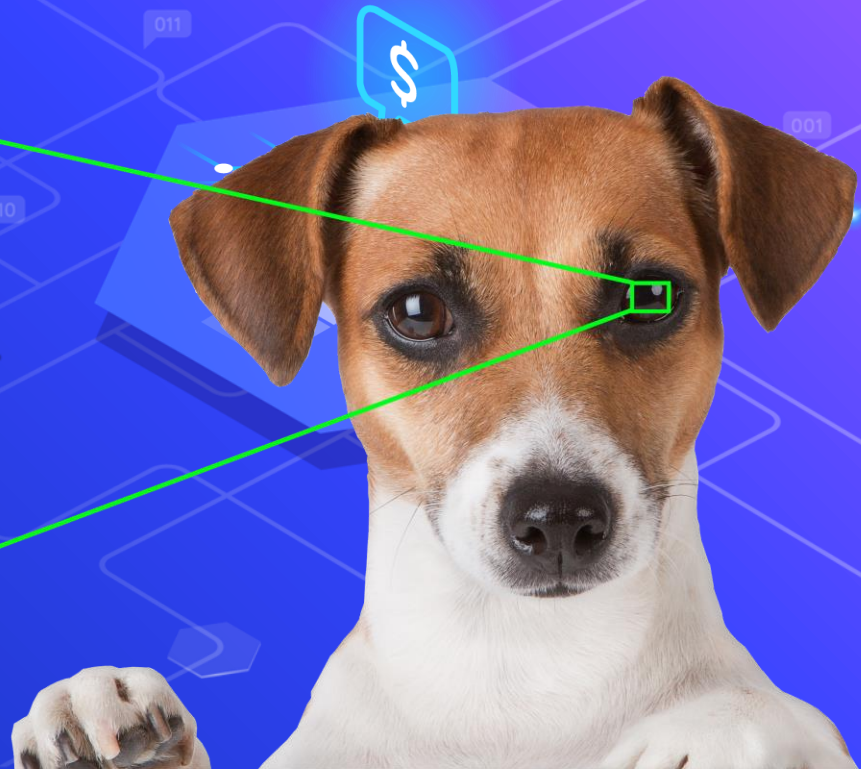
Work with Images

Pixels

Every pixel is one color

Height

Width

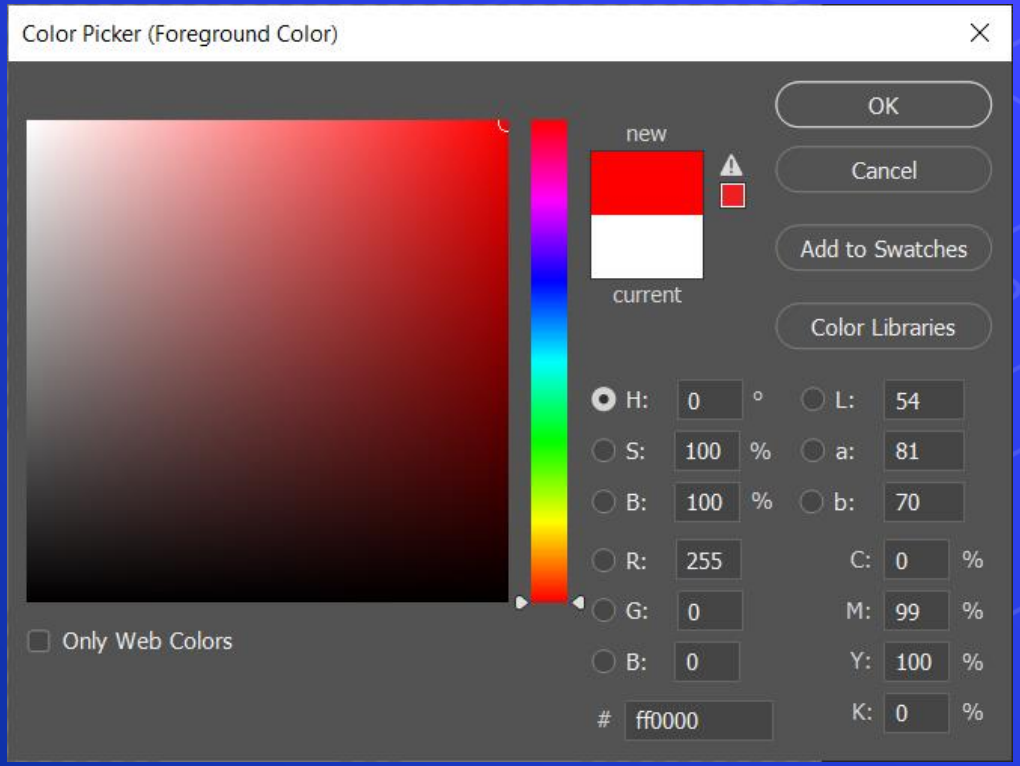


Work with Images

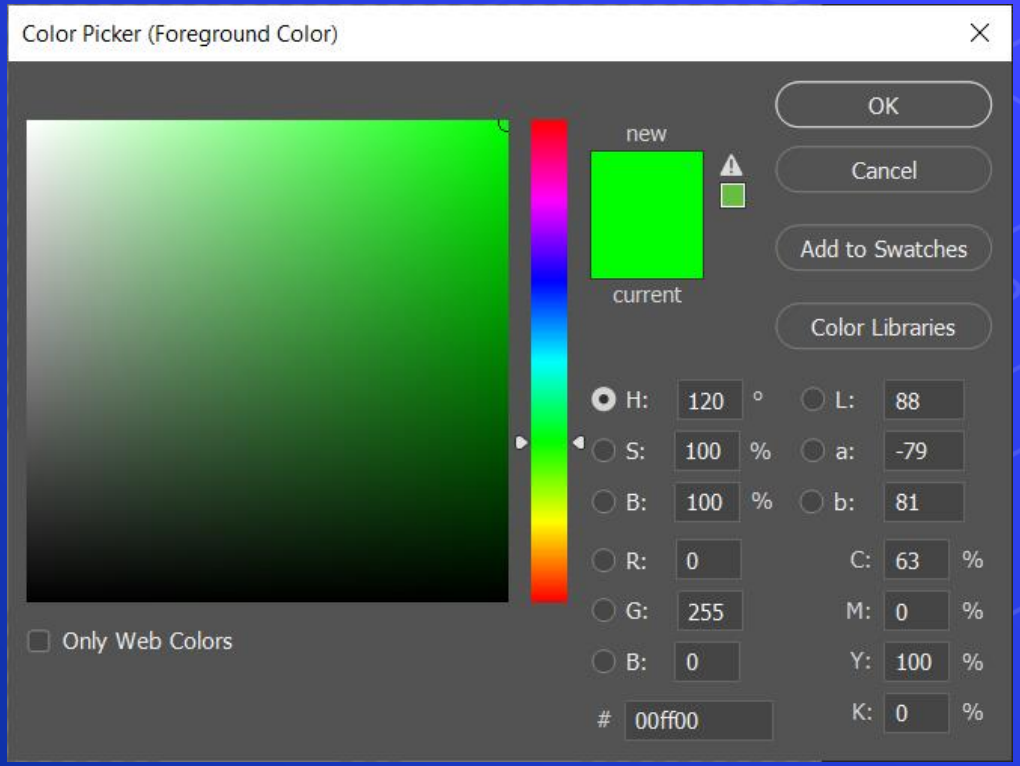
RGB Digital Images

We can generate any color by mixing
(red, green, blue) colors with different ratios.

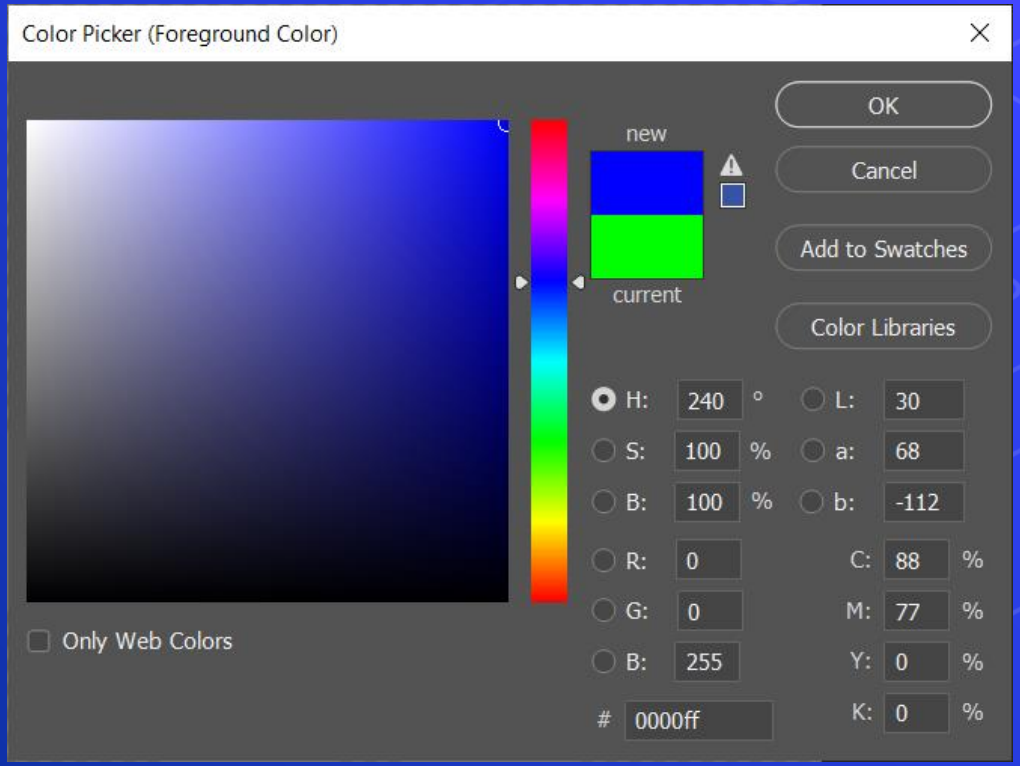
Work with Images (Red)



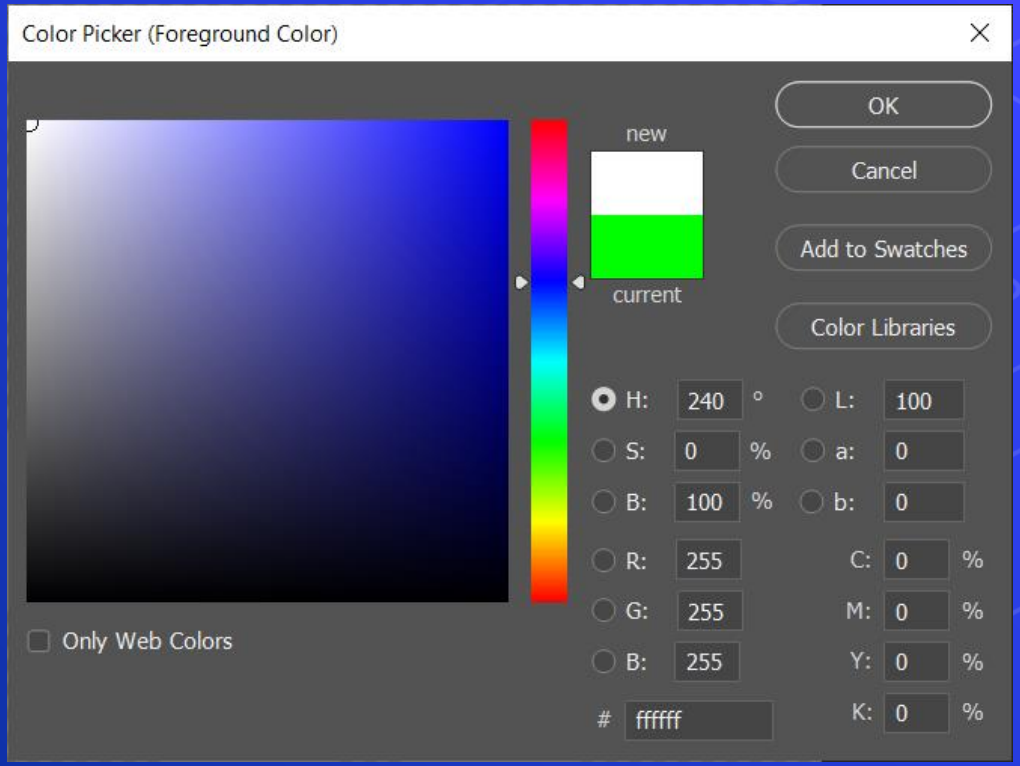
Work with Images (Green)



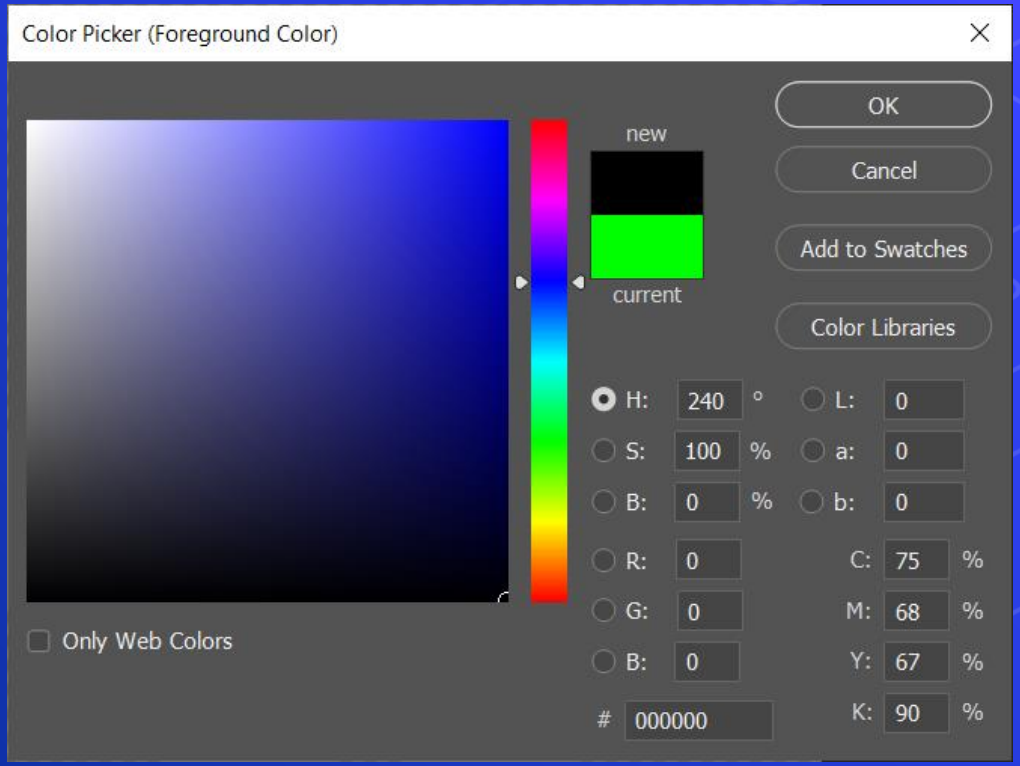
Work with Images (Blue)



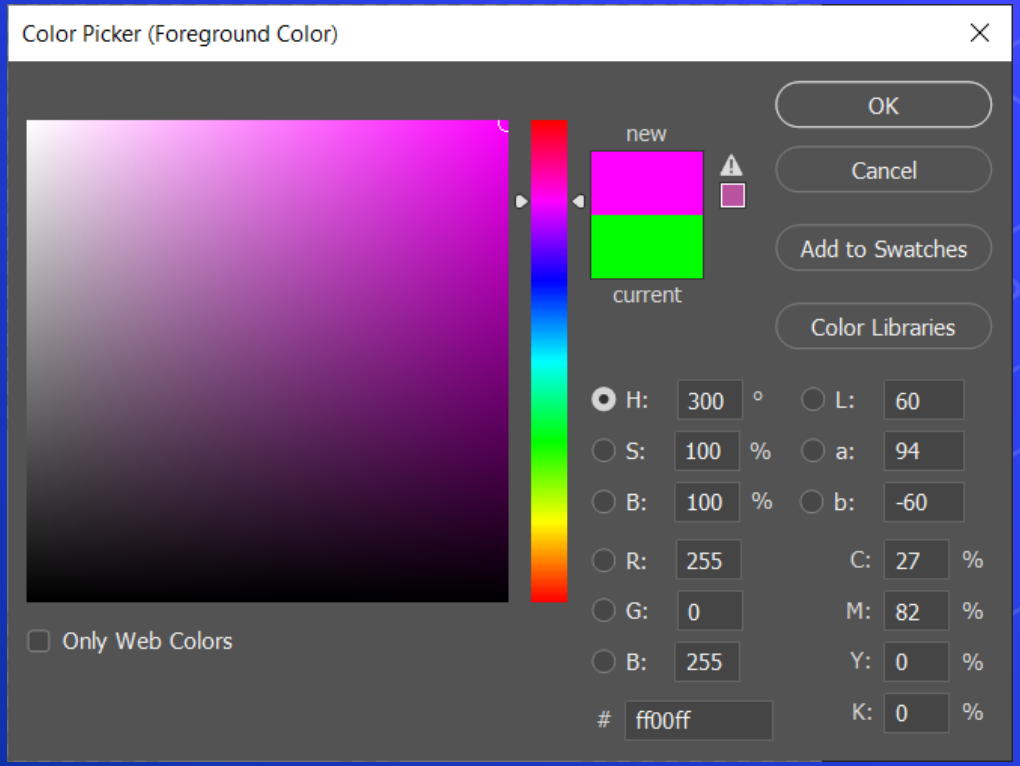
Work with Images (White)



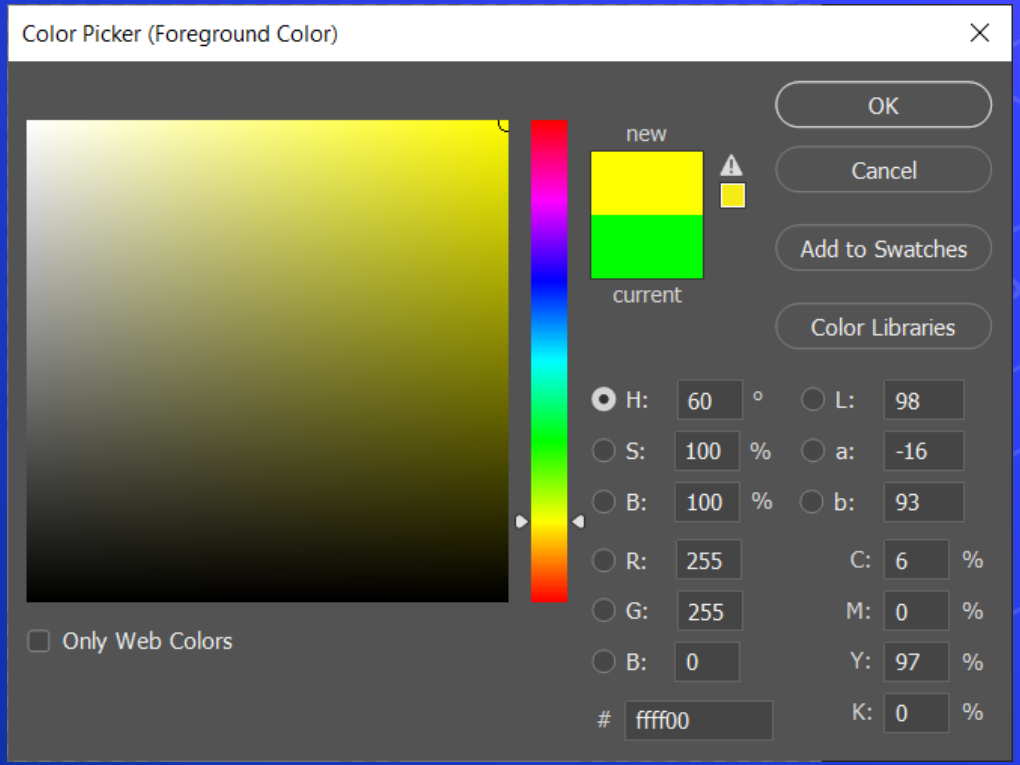
Work with Images (Black)



Work with Images (Pink)



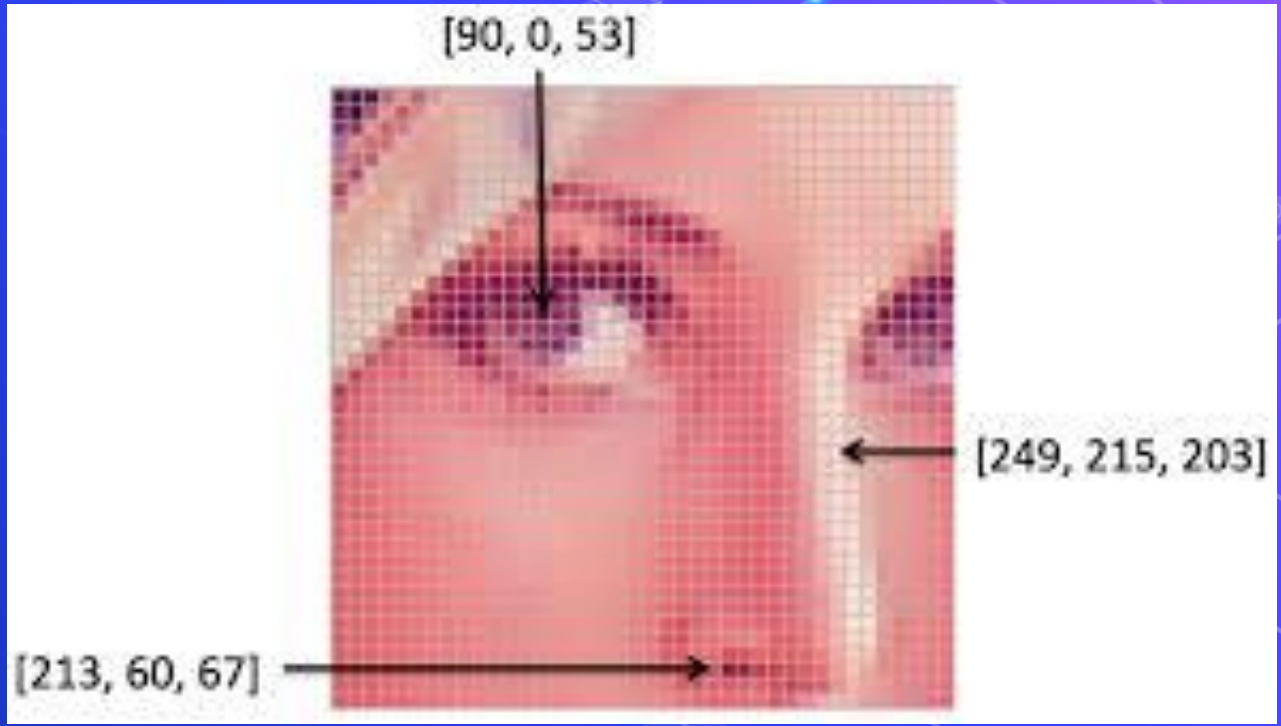
Work with Images (Yellow)



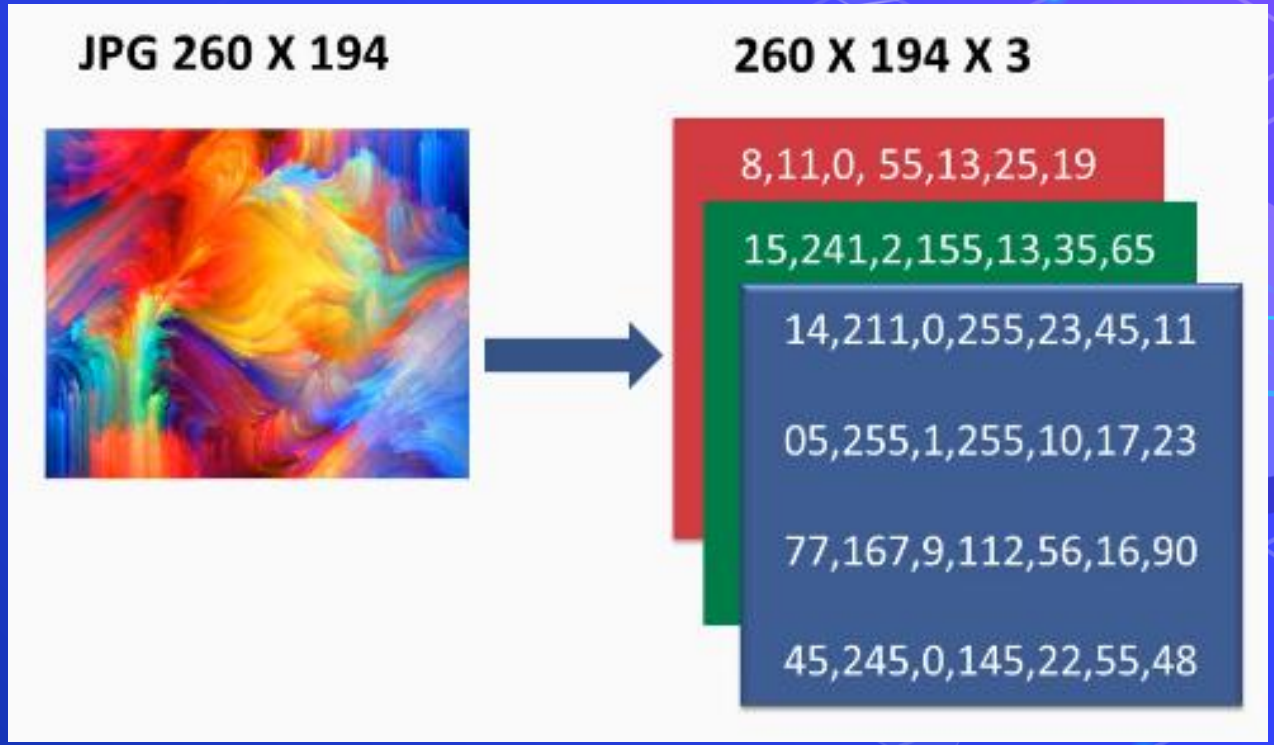
Work with Images

Resolution: 100x100

Width : 100 pixels
Height: 100 pixels



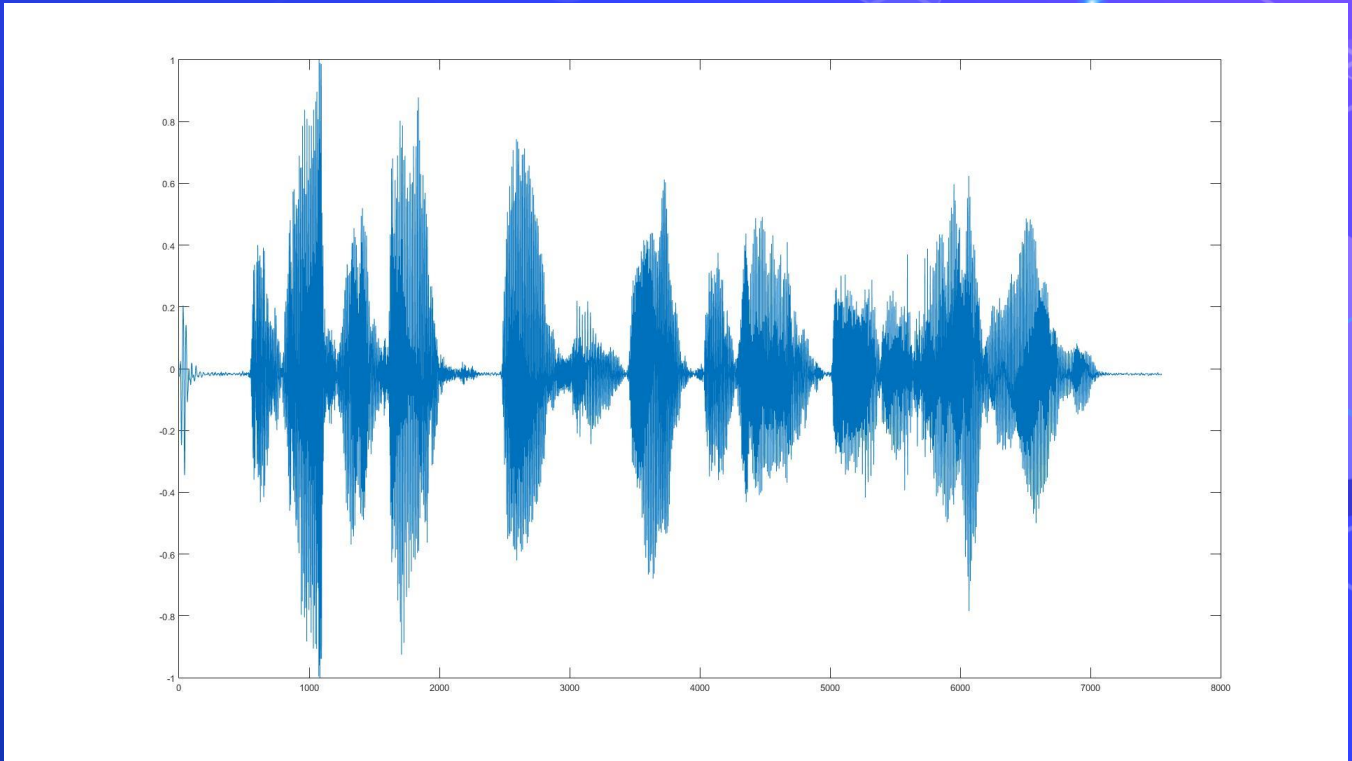
Work with Images



Agenda

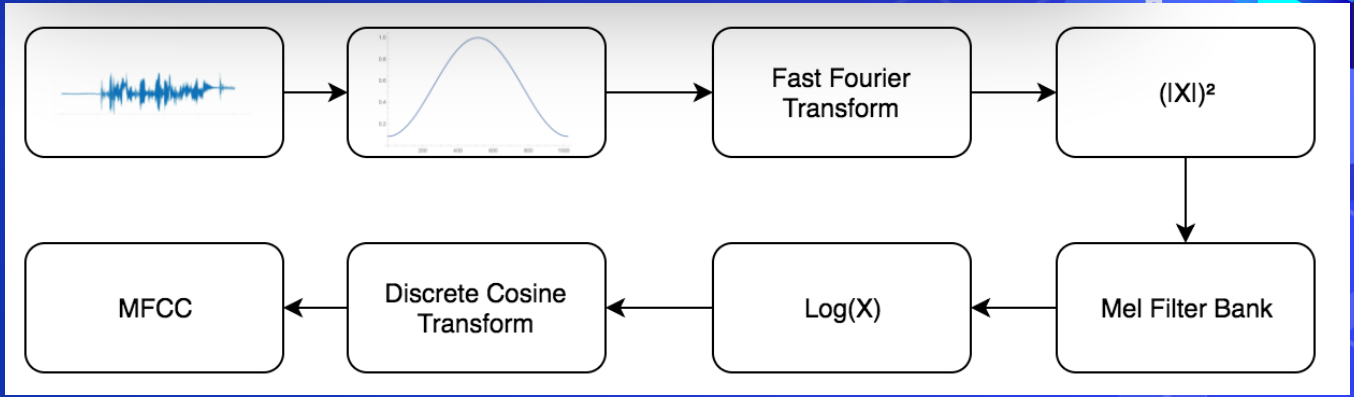
- What is Data
- Machine Learning
- Data Preprocessing
- **Feature Engineering and Extraction**
  - Domain knowledge features
  - Date and Time features
  - String operations
  - Web Data
  - Geospatial features
  - Work with Text
  - Work with Images
  - **Work with Audio**
  - Dimension Reduction with PCA
- 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
  - How to use Pipelines
- Feature Selection

Work with Audio



Work with Audio

```python
import librosa

x, fs = librosa.load('sound.wav')
mfccs = librosa.feature.mfcc(x, sr=fs)
```



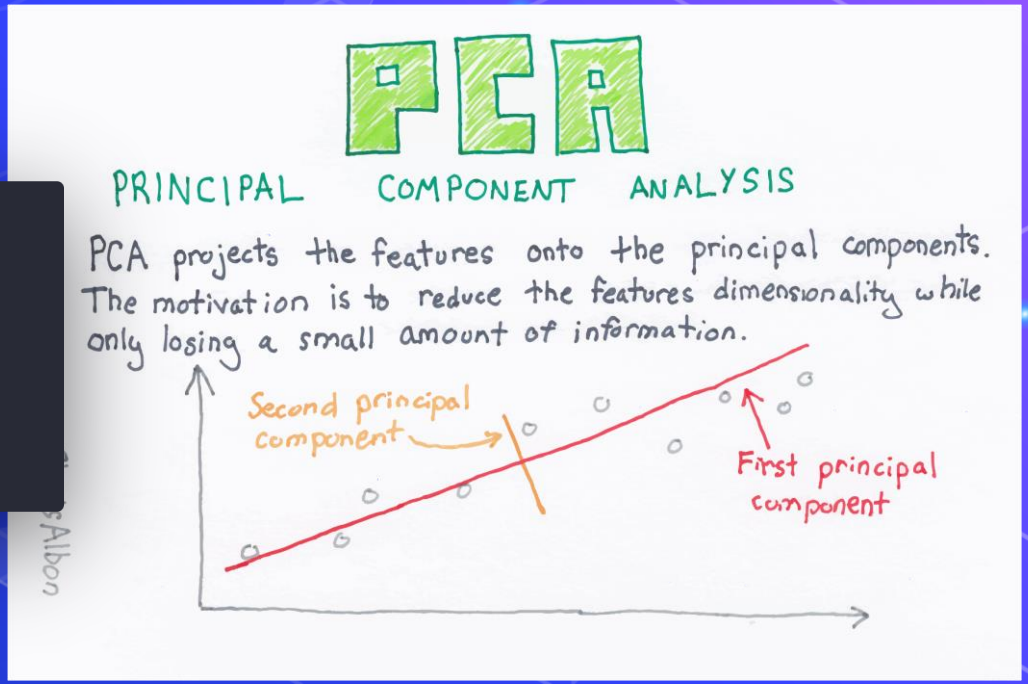
https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

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
  - Work with Images
  - Work with Audio
  - **Dimension Reduction with PCA**
- 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
  - How to use Pipelines
- Feature Selection

Dimension Reduction with PCA

```python
from sklearn.decomposition import PCA

X = # feature vector

pca = PCA(0.9)
X = pca.fit_transform(X)
pca.explained_variance_ratio_
```



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

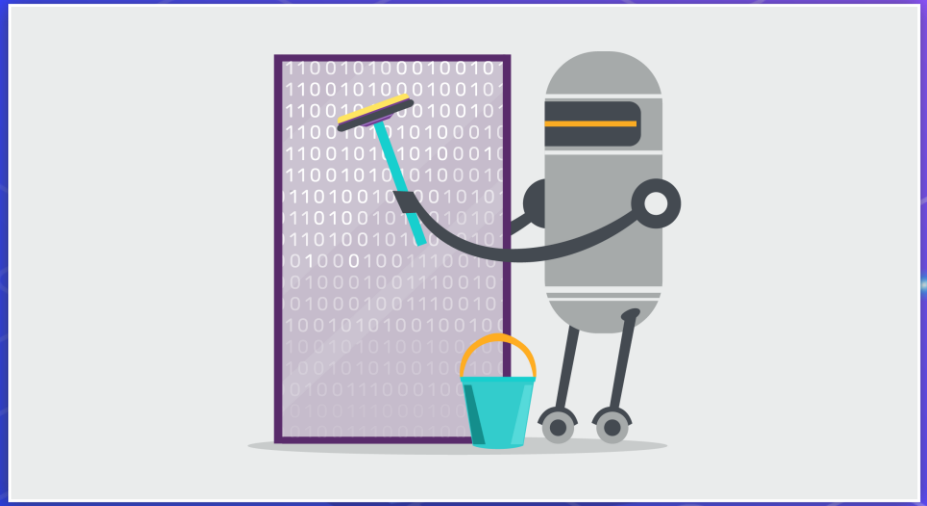
Agenda

- 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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- Feature Selection

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

- 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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- Feature Selection

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

```python
# fill with mean
df['BuildingArea'].fillna(df['BuildingArea'].mean(), inplace=True)


# fill with median
df['YearBuilt'].fillna(df['YearBuilt'].median(), inplace=True)


# fill with mode (most frequent)
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

```python
from sklearn.impute import SimpleImputer

# define the Imputer properties
imputer = SimpleImputer(strategy='mean')    # can use 'median' or 'most_frequent'

# impute
df['BuildingArea'] = imputer.fit_transform(df[['BuildingArea']])

# get the filled value
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

```python
from sklearn.impute import KNNImputer

imputer = KNNImputer()

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

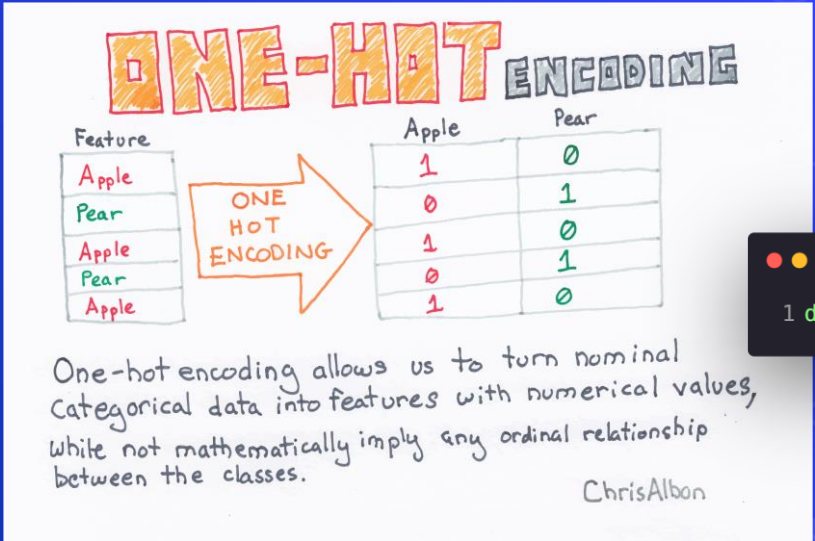
- What is Data
- Machine Learning
- Data Preprocessing
- Feature Engineering and Extraction
  - Domain knowledge features
  - Date and Time features
  - String operations
  - Web Data
  - Geospatial features
  - Work with Text
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- Feature Selection

Work with Categorical data

```python
size_dict = {'XS':1,
        'S':2,
        'M':3,
        'L':4,
        'XL':5,
        'XXL':6}

# apply using map
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




```
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

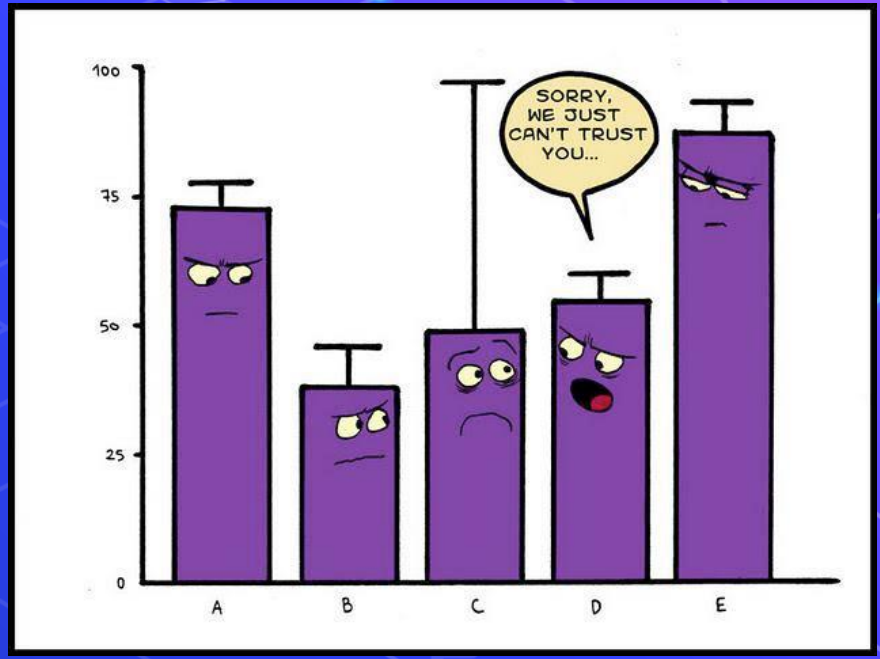
- What is Data
- Machine Learning
- Data Preprocessing
- Feature Engineering and Extraction
  - Domain knowledge features
  - Date and Time features
  - String operations
  - Web Data
  - Geospatial features
  - Work with Text
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- Feature Selection

Detect and Handle Outliers

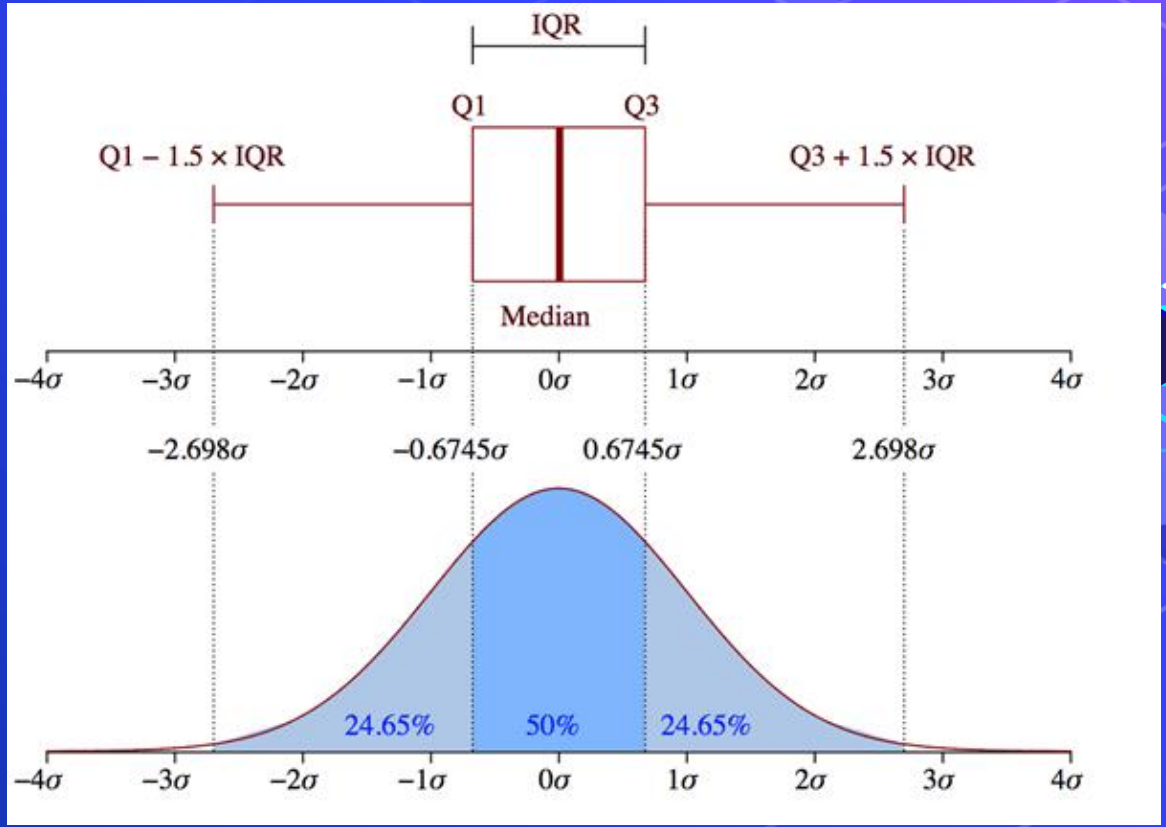
In statistics, an outlier is a data point that differs significantly from other observations.

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.

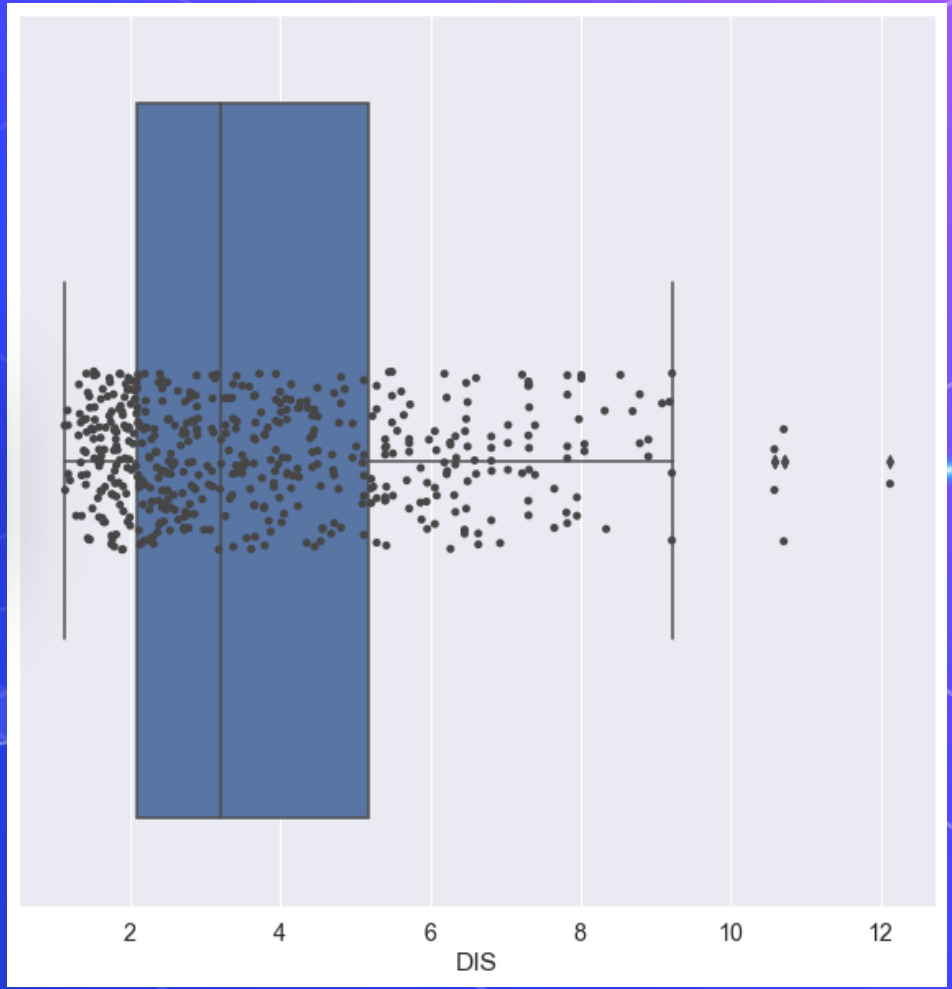


Detect and Handle Outliers



Detect and Handle Outliers

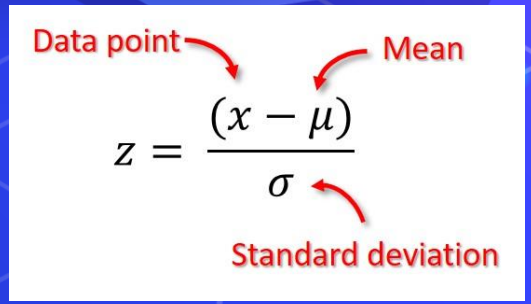
```
1 sns.boxplot(x='DIS', data=df)
2 sns.stripplot(x='DIS', data=df)
```



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

Detect and Handle Outliers

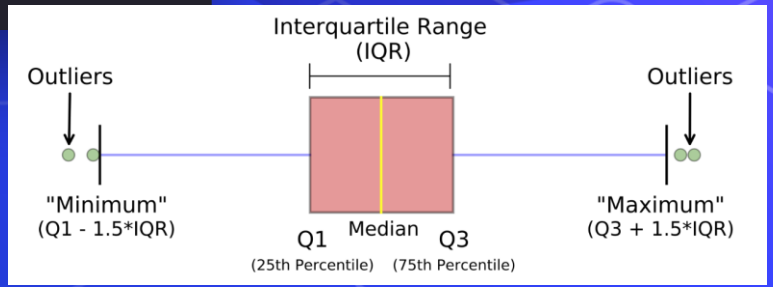
```python
# remove outliers that has absolute z-score > 3
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]


# replace outliers with median value for each column
for col in df.columns:
    df.loc[np.abs(stats.zscore(df[col])) > 3, col] = df[col].median()
```



- Detect Outliers with Visualization.
- **Detect and Handle Outliers with Z-Score.**
- Detect and Handle Outliers with IQR.

Detect and Handle Outliers

```python
from datasist.structdata import detect_outliers

# remove outliers
outliers_indices = detect_outliers(df, 0, df.columns)
df.drop(outliers_indices, inplace=True)


# replace outliers with median value for each column
for col in df.columns:
    outliers_indices = detect_outliers(df, 0, [col])
    col_median = df[col].median()
    df[col].iloc[outliers_indices] = col_median
```



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

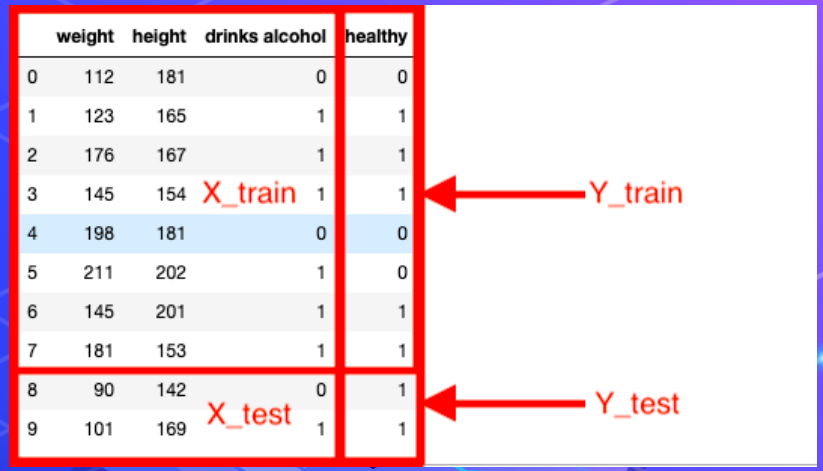
Agenda

- 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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- Feature Selection

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.



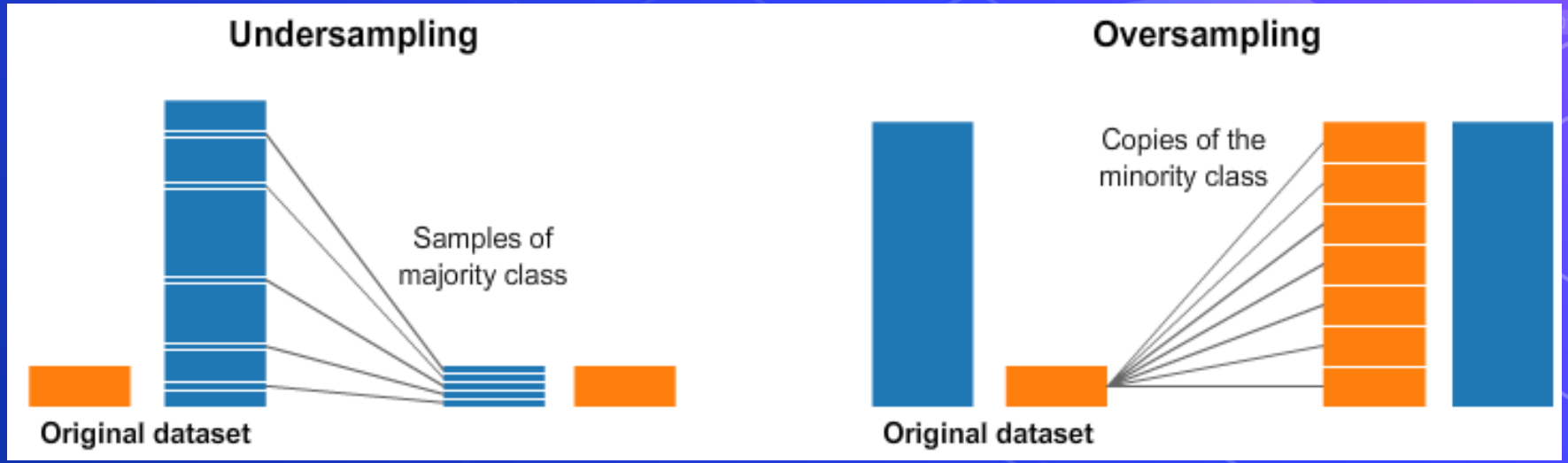

```python
from sklearn.model_selection import train_test_split

x = df.drop('healthy', axis=1)
y = df['healthy']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- Feature Selection

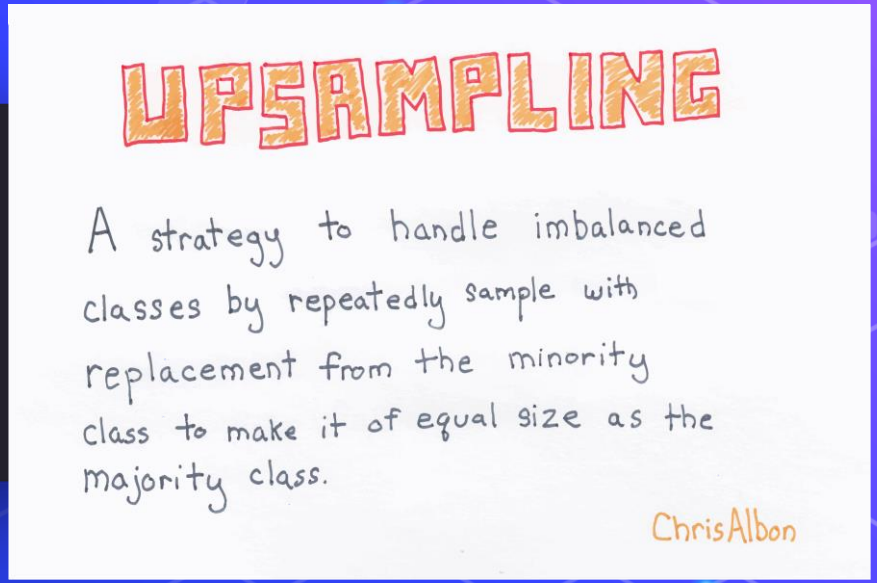
Deal with Imbalanced classes

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.



Deal with Imbalanced classes

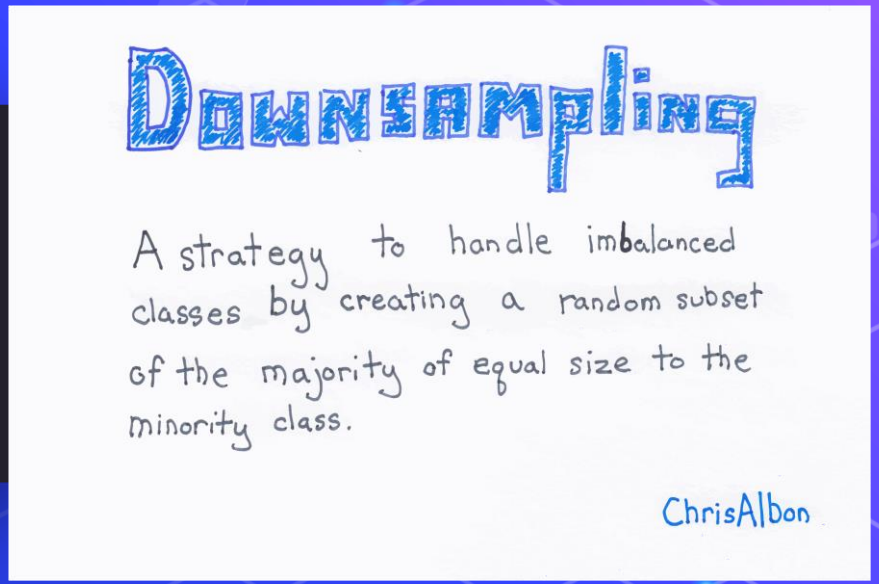
```python
from sklearn.utils import resample

not_fraud = X[X['Class'] == 0]
fraud = X[X['Class'] == 1]

fraud_upsampled = resample(fraud, replace=True
                           , n_samples=len(not_fraud))

upsampled = pd.concat([not_fraud, fraud_upsampled])
```



- Up-sampling or Over-sampling Minority Class
- Down-sampling or Under-sampling Majority Class
- Generate Synthetic Samples with SMOTE

Deal with Imbalanced classes

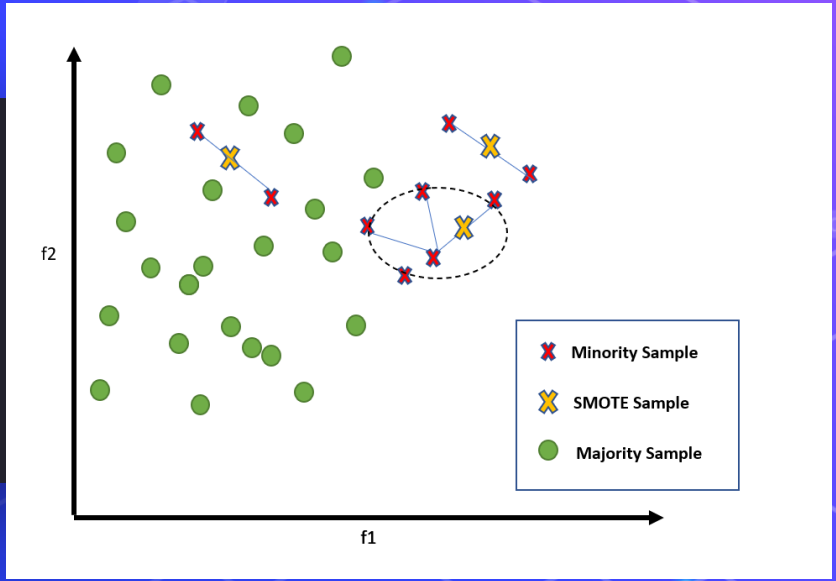
```python
from sklearn.utils import resample

not_fraud = X[X['Class'] == 0]
fraud = X[X['Class'] == 1]

not_fraud_downsampled = resample(not_fraud, replace=False
                                 , n_samples = len(fraud))

downsampled = pd.concat([not_fraud_downsampled, fraud])
```



- Up-sampling or Over-sampling Minority Class
- **Down-sampling or Under-sampling Majority Class**
- Generate Synthetic Samples with SMOTE

Deal with Imbalanced classes

```python
from imblearn.over_sampling import SMOTE

x = df.drop('Class', axis=1)
y = df['Class']

smote = SMOTE()

x_train, y_train = smote.fit_sample(x_train, y_train)
```



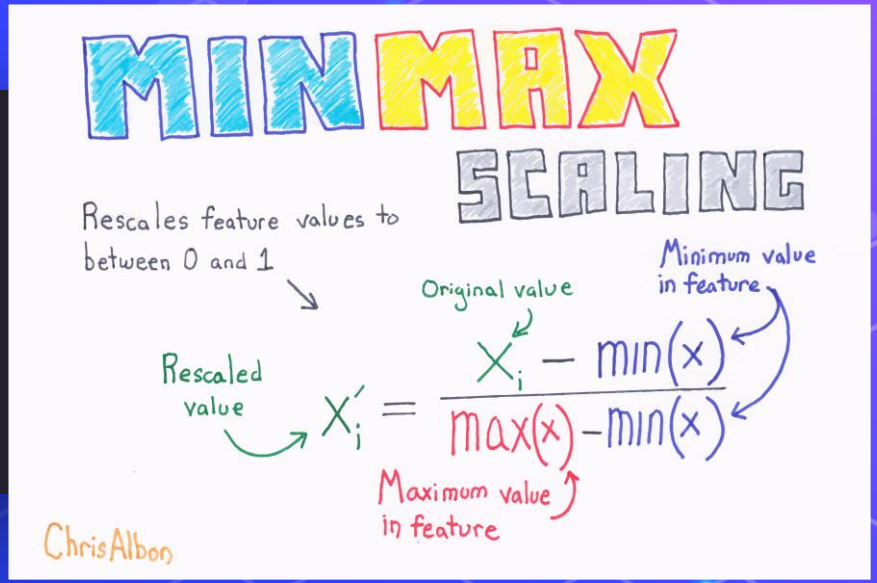
- Up-sampling or Over-sampling Minority Class
- Down-sampling or Under-sampling Majority Class
- Generate Synthetic Samples with SMOTE

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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**
  - How to use Pipelines
- Feature Selection

Feature Scaling

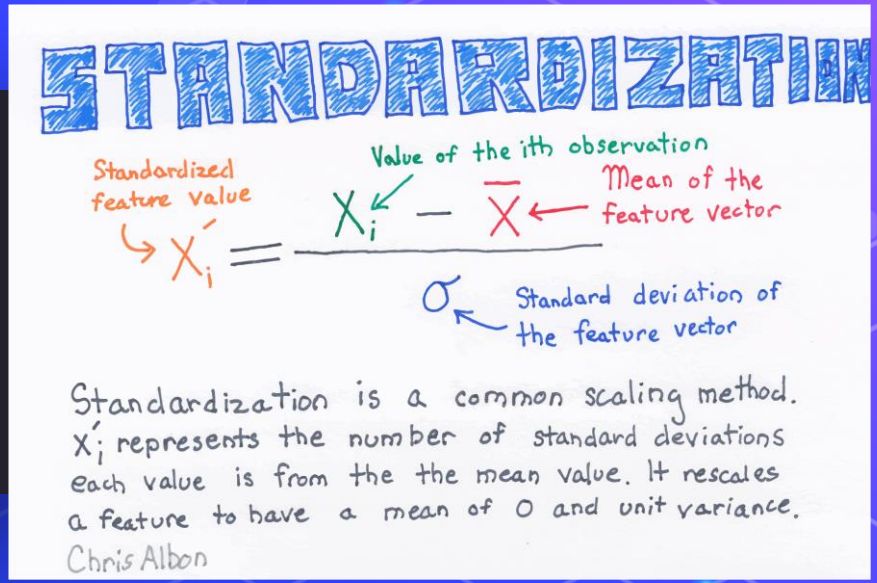
```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(x_train)

scaled_x_train = scaler.transform(x_train)
scaled_x_test = scaler.transform(x_test)
```



- Normalization with Sklearn `MinMaxScaler`.
- Standardizing data with `StandardScaler`.
- Rescaling using `RobustScaler`.

Feature Scaling

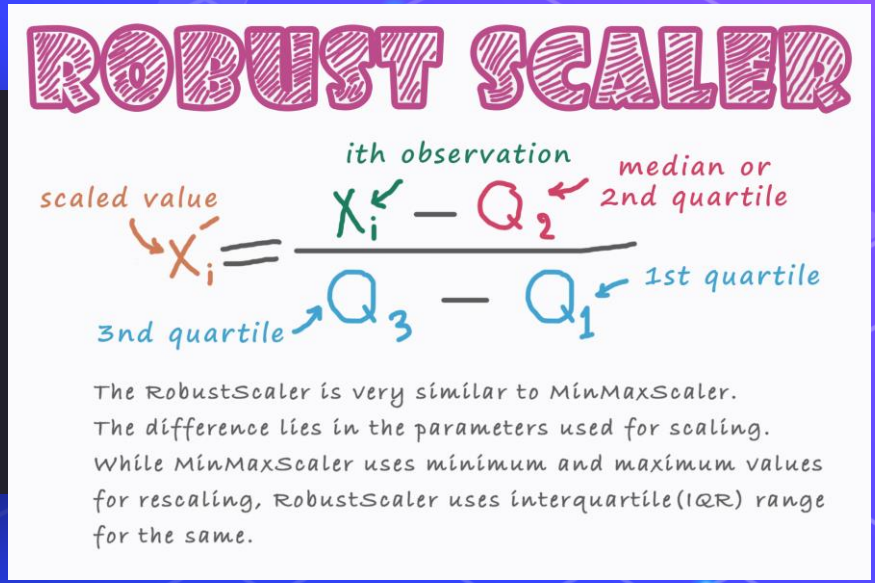
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(x_train)

scaled_x_train = scaler.transform(x_train)
scaled_x_test = scaler.transform(x_test)
```



- Normalization with Sklearn `MinMaxScaler`.
- **Standardizing data with `StandardScaler`.**
- Rescaling using `RobustScaler`.

Feature Scaling

```python
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

scaler.fit(x_train)

scaled_x_train = scaler.transform(x_train)
scaled_x_test = scaler.transform(x_test)
```



- Normalization with Sklearn `MinMaxScaler.`
- Standardizing data with `StandardScaler`.
- **Rescaling using `RobustScaler.`**

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - **How to use Pipelines**
- Feature Selection

How to use Pipelines

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

pipeline_lr = Pipeline([('scalar1',StandardScaler()),
                        ('pca1',PCA(n_components=2)),
                        ('lr_classifier',LogisticRegression())])

pipeline_lr.fit(X_train, y_train)
pipeline_lr.predict(X_test)
```

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
  - Work with Images
  - Work with Audio
  - Dimension Reduction with PCA
- **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
  - How to use Pipelines
- **Feature Selection**

Feature Selection

In machine learning, feature selection is the process of choosing a subset of input features that contribute the most to the output feature for use in model construction.

Feature selection is substantially important if we have datasets with high dimensionality (i.e., large number of features). High-dimensional datasets are not preferred because they have lengthy training time and have high risk of overfitting.

Feature selection helps to mitigate these problems by selecting features that have high importance to the model, such that the data dimensionality can be reduced without much loss of the total information.

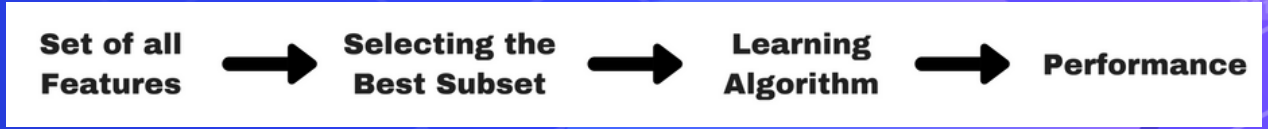
Some benefits of feature selection are:

1. Reduce training time
2. Reduce the risk of overfitting
3. Potentially increase model's performance
4. Reduce model's complexity such that interpretation becomes easier

Feature Selection

- Filter Methods
- Wrapper Methods
- Embedded Methods

In filter methods, features are selected independently from any machine algorithms. Filter methods generally use a specific criteria, such as scores in statistical test and variances, to rank the importance of individual features.

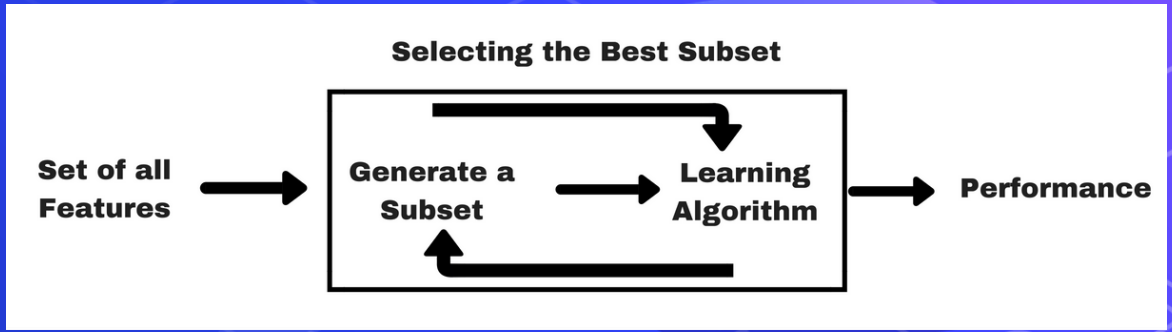


- Pearson's Correlation Coefficient with `f_regression()`
- Mutual Information for regression using `mutual_info_regression()`
- ANOVA for numeric feature using `f_classif()`
- Chi-squared for categorical feature using `chi2()`
- Mutual Information for classification using `mutual_info_classif()`

Feature Selection

- Filter Methods
- Wrapper Methods
- Embedded Methods

Wrapper methods try to find a subset of features that yield the best performance for a model by training, evaluating, and comparing the model with different combinations of features.

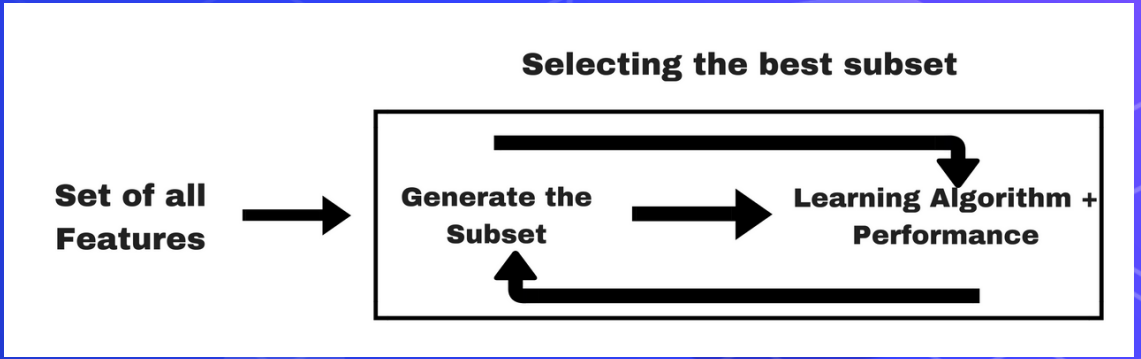


- Recursive Feature Elimination with `RFE()`

Feature Selection

- Filter Methods
- Wrapper Methods
- Embedded Methods

Embedded methods combine the strong points of filter and wrapper methods by taking advantage of machine algorithms that have their own built-in feature selection process. They integrate a feature selection step as a part of the training process



- LASSO & Ridge regression
- Decision trees & Random forests
- Support vector machines

Questions ?!

Thanks!

>_ Live long and prosper