Machine Learning with Python

22/4/2019

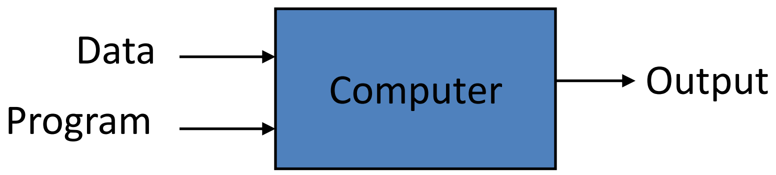
Machine learning

- A Machine learning model Derives the mathematical relationship between these input features and the output. Once we know the relationship between input and output, we can use that algorithm to predict the output for future inputs.
- We thereby made the model “intelligent” hence it is considered part of Artificial Intelligence at the most basic level.

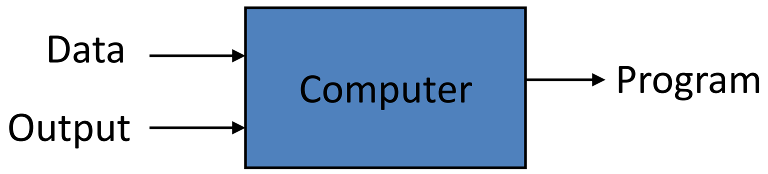
So What Is Machine Learning?

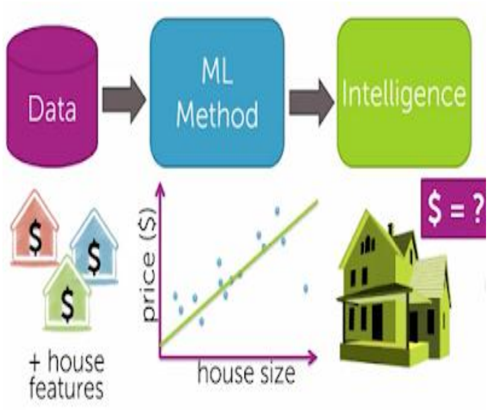
- Automating automation
- Getting computers to program themselves
- Writing software is the bottleneck
- Let the data do the work instead!

Traditional Programming



Machine Learning





Source: http://stem-2.blogspot.in/2016/03/coursera-machine-learning-course-notes-1.html

Different Modes of Machine Learning

- **Supervised:**
  - Ground truth is available for each data point
  - Map input (features) to output (labels)
  - Examples: find the energy consumption of next month given a database of labeled previous month energy consumption
- **Unsupervised:**
  - No ground truth is available
  - Map data points to groups (or clusters) in the data
  - Examples: categorizing household type based on the load/energy consumption

<ins>SUPERVISED LEARNING</ins>

- Predict the value of an outcome based on a number of input measures/features
- Target and predictors/features
- Classification and regression

<ins>UNSUPERVISED LEARNING</ins>

- Exploratory in nature
- Identify the associations and patterns among a set of input

Types of Supervised Learning

CLASSIFICATION

- Target variable is categorical / discrete eg. 1/0 or Yes/No
- Eg. Is the person a churner or a non-churner

REGRESSION

- Target variable is continuous/interval
- Eg. Time to equipment failure

Types of UNSUPERVISED LEARNING

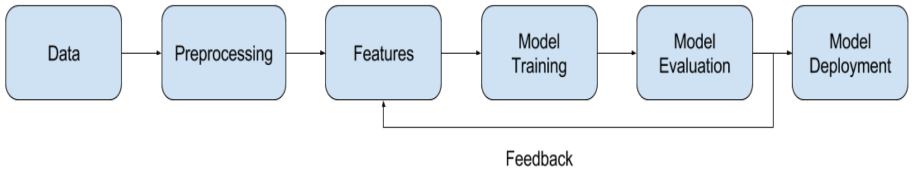
CLUSTERING

- Grouping of entities on the basis of common characteristics.
- K-Means, Hierarchical clustering, etc.

PATTERN MINING

- Mining associations in the data
- Apriori, Frequent Pattern Mining

Machine Learning Pipeline



Data Pre-processing

- Adapt and coalesce the data to a compatible format
- Cleaning the data
- Splitting the data into training/validation/testing sets

Data Pre-Processing

- Data in the Real World Is Dirty: Lots of potentially incorrect data, e.g., instrument faulty, human or computer error, transmission error
  - incomplete: lacking attribute values, lacking certain attribute s of interest, or containing only aggregate data
    - e.g., *Occupation*=“ ” (missing data)
  - noisy: containing noise, errors, or outliers
    - e.g., *Salary*=“−10” (an error)
  - inconsistent: containing discrepancies in codes or names, e.g.,
    - *Age*=“42”, *Birthday*=“03/07/2010”
    - Was rating “1, 2, 3”, now rating “A, B, C”
    - discrepancy between duplicate records
  - Intentional (e.g., *disguised missing* data)
    - Jan. 1 as everyone’s birthday?

Incomplete (Missing) Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (when doing classification)—not effective when the % of missing values per attribute varies considerably
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may be due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which require data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

How to Handle Noisy Data?

- Binning
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
  - smooth by fitting the data into regression functions
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Data Cleaning as a Process

- Data discrepancy detection
  - Use metadata (e.g., domain, range, dependency, distribution)
  - Check field overloading
  - Check uniqueness rule, consecutive rule and null rule
  - Use commercial tools
    - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
    - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)
- Data migration and integration
  - Data migration tools: allow transformations to be specified
  - ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface
- Integration of the two processes
  - Iterative and interactive (e.g., Potter's Wheels)

Features

- Extract representations that capture the "essence" of data
- For classification: Maximize the intra-class similarity and inter-class distance
- For regression: Maximize the explanation power
- Multiple methods
- No clear "right" or "wrong" answers

Feature scaling/Data preprocessing

- Feature scaling is a method used to standardize the range of independent variables or features of data.
- In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. (Source: Wikipedia)

To find the null values use isnull() function

```
data.isnull()
```

You can also replace also

```
data.replace([np.inf, -np.inf], np.nan, inplace = True)
```

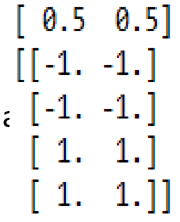
Also you can drop the rows/columns having ‘NaN’ values

```
data.dropna( inplace = True)
```

Standardization

- Standardization involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one.

```
from sklearn.preprocessing import StandardScaler
data = [[0, 0], [0, 0], [1, 1], [1, 1]]
scaler = StandardScaler()
scaler.fit(data)
  #Compute the mean and standard deviation to be used for
    later scaling.
print(scaler.mean_)
print(scaler.transform(data))
#Perform standardization by centering a
```



```
[ 0.5  0.5]
[[-1. -1.]
 [-1. -1.]
 [ 1.  1.]
 [ 1.  1.]]
```

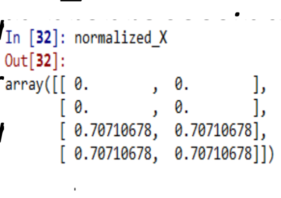
Normalize , Scale

1. normalize the data attributes
2. used for sparse data with varying scale attributes.

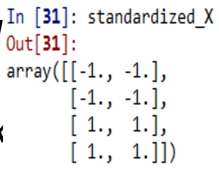
import sklearn

normalized_X =
sklear *norma*

standar
sklear *scale(x*



```
In [32]: normalized_X
Out[32]:
array([[ 0.        ,  0.        ],
       [ 0.        ,  0.        ],
       [ 0.70710678,  0.70710678],
       [ 0.70710678,  0.70710678]])
```



```
In [31]: standardized_X
Out[31]:
array([[-1., -1.],
       [-1., -1.],
       [ 1.,  1.],
       [ 1.,  1.]])
```

MinMax scaler

from sklearn.preprocessing import MinMaxScaler

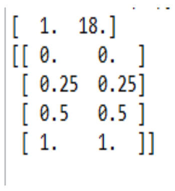
data = [[-1, 2], [-0.5, 6],[0, 10], [1, 18]]

scaler = MinMaxScaler()

print(scaler.fit(data))

print(scaler.data_max_)

print(scaler.transform(data))



```
[  1.  18.]
[[ 0.    0.  ]
 [ 0.25  0.25]
 [ 0.5   0.5 ]
 [ 1.    1.  ]]
```

Binarizer

Binarizer is used to convert the data or features into binary form using a binary threshold :

- all the values above threshold becomes 1
- all values below threshold become 0

from sklearn.preprocessing import Binarizer

x=[34,56,23,78,89,12,90,87,44]

b=Binarizer(threshold=60)

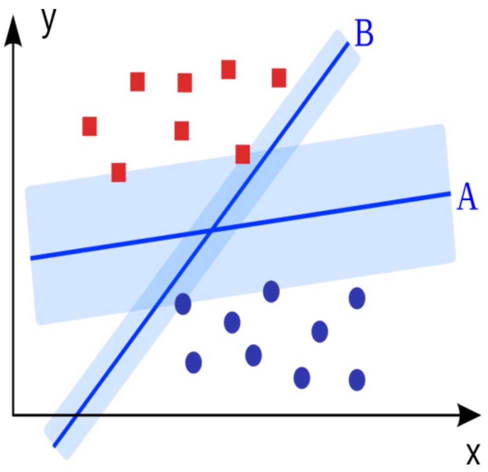
b.fit(x)

b.transform(x)

```
Out[34]: array([[0, 0, 0, 1, 1, 0, 1, 1, 0]])
```

Model Training

- Learn the mapping from features to output
- Ideal: 100% accuracy in differentiating features of different classes
- Practical: Maximize accuracy while maintaining "generalizability"
- Quality dependent on training data quality/quantity
- Choosing an ML algorithm for the task
- Choosing the labels



Model Evaluation

- Success Criterion:
  - Classification: accuracy

$$\frac{No.\,of\ correctly\ classified\ data\ points}{Total\ no.\,of\ data\ points}$$

  - Regression: residual error

$$\sum^{all\ data\ points} (actual - predicted)^2$$

- Preliminary evaluation: Validation
- Validation: “validate” the trained model on a set of unseen data
- Final evaluation: Testing
- Testing: “test” the estimated real-world performance of the model

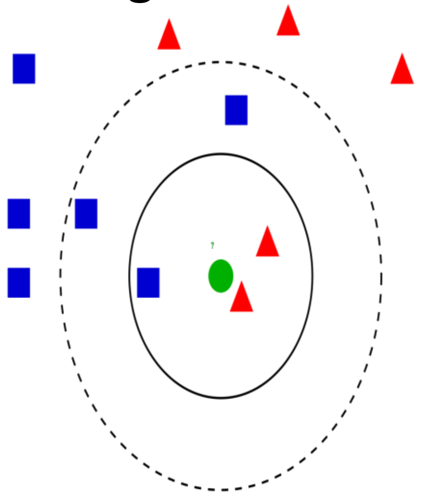
- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy…….

Feedback and Model Deployment

- Machine Learning is an iterative process
- The model is re-trained many times till validation performance is satisfactory
- What changes in future iterations?
    - Preprocessing
    - Features
    - Choice of algorithm
    - Choice of parameters
    - Training data
- Testing performance indicates estimated real-world performance of the model
- The testing performance is the final reported value

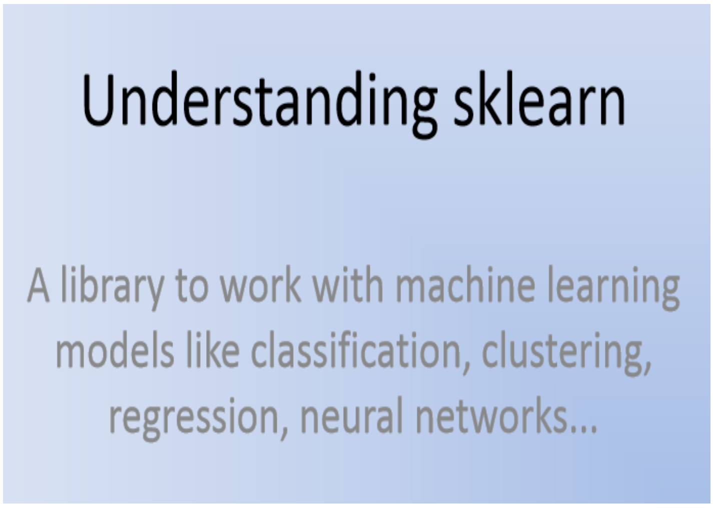
KNN: *K*-Nearest Neighbors

- **Classification:**
    - Assign the class label by majority voting among the labels of the k nearest neighbors
- **Regression:**
    - Assign the average value of the k nearest neighbors



Possible usecases in Smart Grid for Machine learning

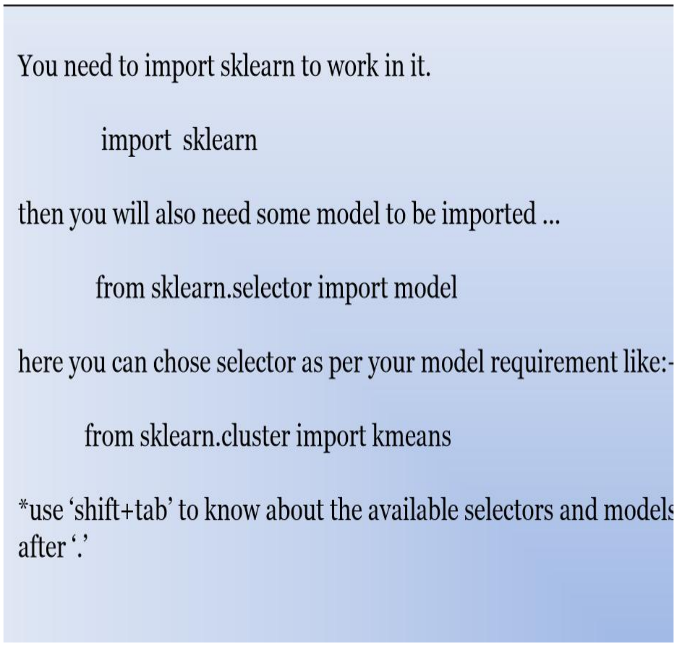
- ML methods have been widely used in smart grid for monitoring and control of power systems
- Can be employed to predict failures of system components
- For energy management of load and source in smart grid network
- For prediction of intrusion detection or malacious activities .



Understanding sklearn

A library to work with machine learning models like classification, clustering, regression, neural networks…

- *Machine Learning in Python*
- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib



You need to import sklearn to work in it.

```
import  sklearn
```

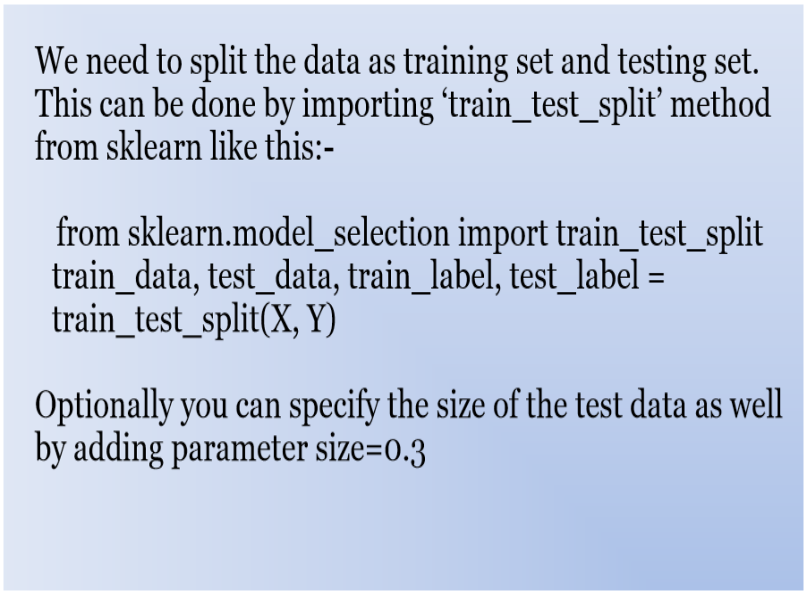
then you will also need some model to be imported ...

```
from sklearn.selector import model
```

here you can chose selector as per your model requirement like:-

```
from sklearn.cluster import kmeans
```

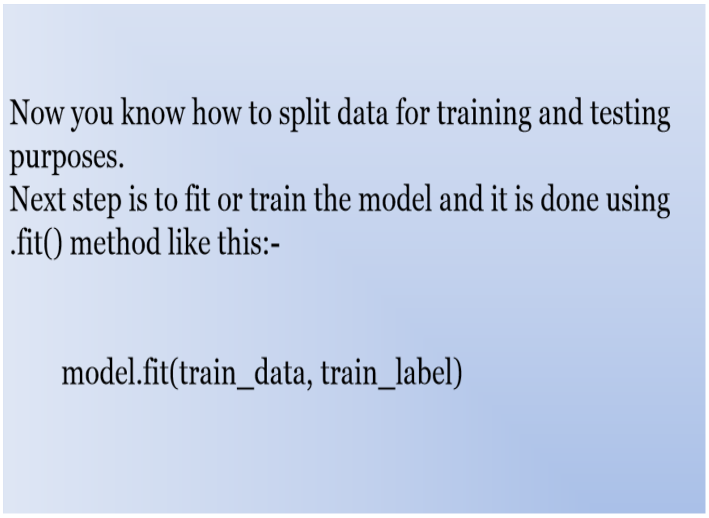
*use 'shift+tab' to know about the available selectors and models after '.'



We need to split the data as training set and testing set. This can be done by importing 'train_test_split' method from sklearn like this:-

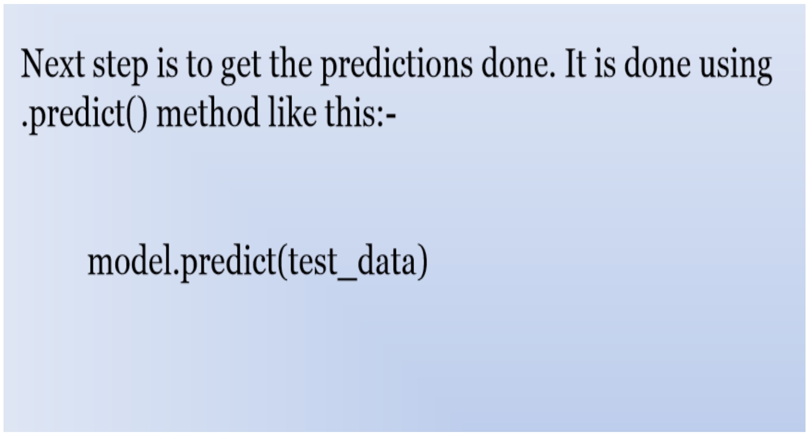
```
 from sklearn.model_selection import train_test_split
train_data, test_data, train_label, test_label =
train_test_split(X, Y)
```

Optionally you can specify the size of the test data as well by adding parameter size=0.3



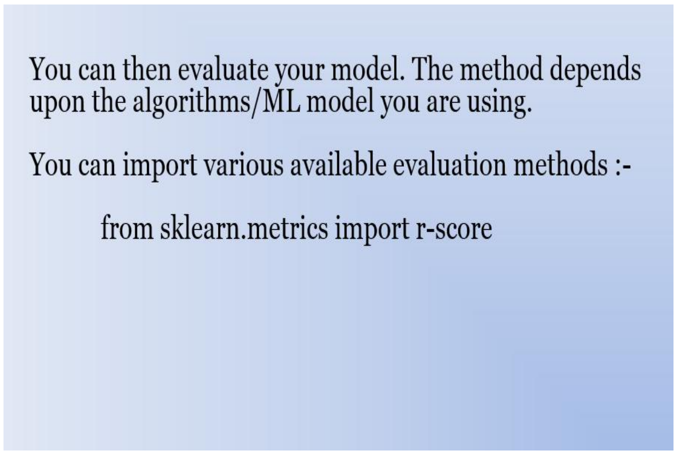
Now you know how to split data for training and testing purposes.
Next step is to fit or train the model and it is done using .fit() method like this:-

```
model.fit(train_data, train_label)
```



Next step is to get the predictions done. It is done using .predict() method like this:-

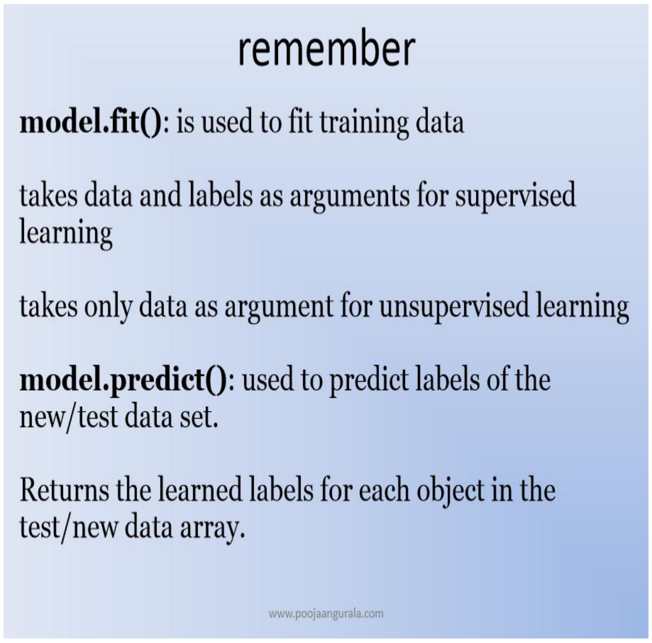
```
model.predict(test_data)
```



You can then evaluate your model. The method depends upon the algorithms/ML model you are using.

You can import various available evaluation methods :-

```
from sklearn.metrics import r-score
```



remember

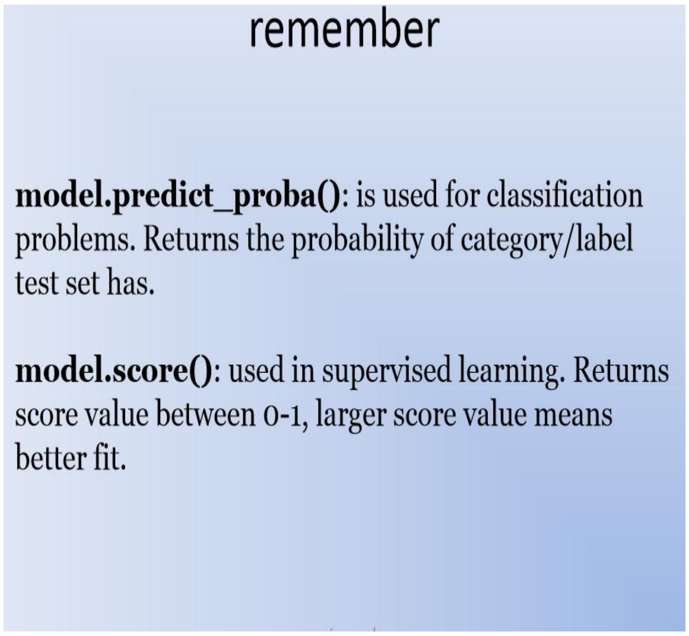
model.fit(): is used to fit training data

takes data and labels as arguments for supervised learning

takes only data as argument for unsupervised learning

model.predict(): used to predict labels of the new/test data set.

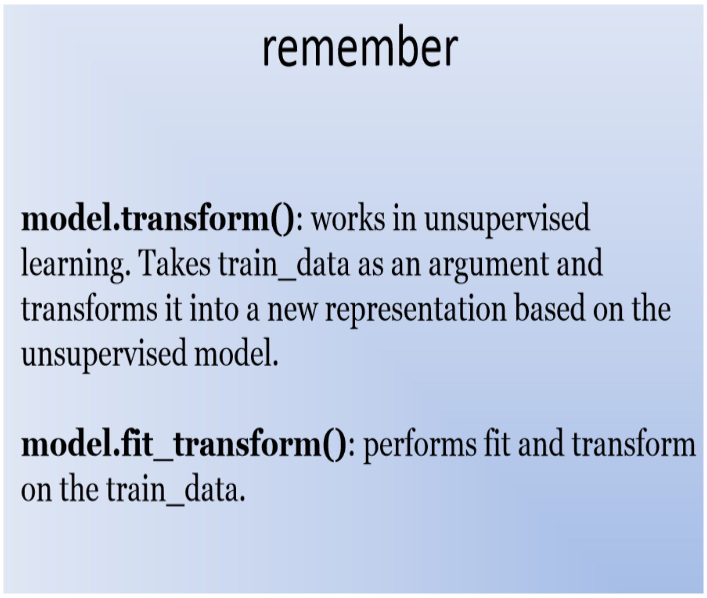
Returns the learned labels for each object in the test/new data array.



remember

model.predict_proba(): is used for classification problems. Returns the probability of category/label test set has.

model.score(): used in supervised learning. Returns score value between 0-1, larger score value means better fit.



remember

model.transform(): works in unsupervised learning. Takes train_data as an argument and transforms it into a new representation based on the unsupervised model.

model.fit_transform(): performs fit and transform on the train_data.