Start: July 4th, 2022 End: August 18th, 2022

Module 1: Introducing Machine Learning

Section 1: The Motivations Surrounding ML

Examples of ML:
- Automated customer support
    - Chatbots
    - Predictive Maintenance: monitor equipment, anticipate failures and proactively fix them
    - Recommendation System
    - Customer Churn Prediction
- Voice recognition
- Assisted driving technologies

Machine Learning: algorithms that use statistical reasoning to find patterns within massive amounts of data

AI vs ML vs Deep Learning (DL):
- AI: grand goal of developing machines that can mimic human behaviour and perform tasks that typically require human intelligence
- ML: small task of using algorithms to enable a machine to learn from data and make predictions
- DL: specific advanced ML technique that uses artificial neural networks with attempt to mimic how a human brain learns

Failures of ML:
- IBM Watson Cancer Research: doctors identified several examples of incorrect and sometimes lethal treatment recommendations
- Microsoft's Chatbot Tay: trolls corrupted the ML algorithm to spew inflammatory and offensive tweets

What ML is NOT:
- Rule-Based Methods: if it is a yes-no, then you'll be using an algorithm to *automate*.
    - No learning is being done so this is NOT ML.
- Descriptive Data Analysis: it uses hindsight only - data to describe the past ie. tracking largest buyers, ranking reliability of suppliers, etc.
    - Hindsight → Insight (making sense of data) → Foresight (dynamically preparing for future based events) : ML = Insight + Foresight
- ML is not used for: schedule-creation, ethical rationing,
    - ML is used for: prediction whether an mv on Youtube will hit 1 mill views, whether business plan is successful

Section 2: The Fundamental Concepts That Underlie ML

Note: ML is applicable as long as a "machine" is necessary to parse large amounts of data and as long as the relationships between the various data values are "learned" algorithmically rather than manually.

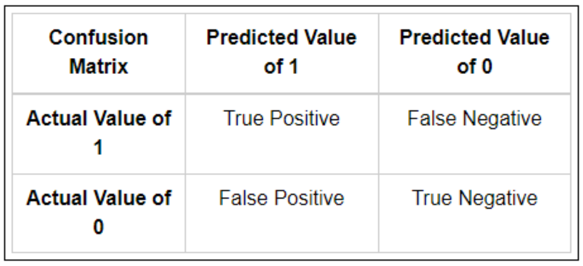
ML Paradigms:

1. Supervised Learning:
    - Definition: linking inputs with their corresponding outputs or labels based on historical data
    - "Learn by example"
    - Examples:
        - Image Recognition Classifier: classifies images of fruit as either apples or not-apples
        - Amazon's Recruitment Tool: tool trained on past hiring data to automate hiring decisions - on resumes of past job applicants (input) and whether they ended up getting hired (output)
2. Unsupervised Learning
    - Definition: infers some natural structure present within the data
    - Only uses input data, so no output data or labels
    - Examples:
        - Image Recognition Clustering: groups similar images of fruit
        - Retail Store Targeted Offers: uses customer sales data to cluster them with similar purchasing patterns ie regular frequent shoppers, discount and bulk shoppers (allows business to provide targeted coupons or offers)
3. Hybrid: Semi-Supervised Learning
    - Definition: makes use of unlabelled data by clustering after the supervised learning process
    - Useful when there are very few labelled examples and a large number of unlabelled examples
    - Example:
        - Classification of Photographs: requires a dataset of photographs that have already been labelled by human operators
4. Reinforcement Learning
    - Definition: the training of ML models to make sequence of decisions in an uncertain and potentially complex environment
    - "Learning by feedback" - performs action, received reward or punishment, uses the feedback to take future actions
    - "Trial and error learning" - selects it's actions based of its past experiences (exploitation) and also by new choices (exploration)
    - Examples:

        - DeepMind’s AlphaGo: program that plays Go using reinforcement learning
        - Personalized Advertising: reinforcement learning along with A/B testing is used to optimize click-through rates

Model Evaluation Metrics: used to evaluate a ML model’s performance

- Confusion Matrix:
    - Depicts whether model is correctly labelling the observations or wrongly predicting observations belonging to one class as belonging to another



Confusion Matrix	Predicted Value of 1	Predicted Value of 0
Actual Value of 1	True Positive	False Negative
Actual Value of 0	False Positive	True Negative

- Accuracy Score:
    - Identifies how often the ML model makes the right predictions
    - Accuracy = Number of correct predictions/Total number of predictions
    - Accuracy = (True Positive + True Negative) / (True Positive + True Negative + False Positive + False Negative)
- Type I and Type II Errors:
    - Type I: false positive
    - Type II: false negative
- Other classification Metrics: [the closer the value is to 1 the better]
    - Recall
        - How many positives were correctly classified as positive
        - Recall = True Positive / (True Positive + False Negative)
    - Precision
        - How many positives classified by the algorithm are really positives
        - Precision = True Positive / (True Positive + False Positive)
    - F1-score
        - The harmonic mean of Precision and Recall
        - F1-score = 2 * Precision * Recall / (Precision + Recall)
- Mean Square Error:
    - MSE is the average of the square of errors
    - The error denotes the difference between the actual target values (y) and the predicted ones - the larger the number, the larger the error
    - MSE must be compared with another model and whoever’s value is lower, that model is better

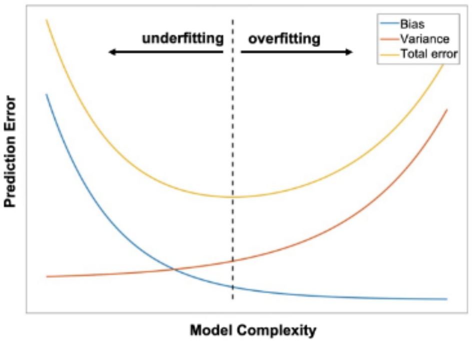
Note: Confusion matrix, accuracy score, and the classification metrics are usually used to evaluate the performance of classification models. While MSE is used for evaluating regression models' performance.

Model Validation:

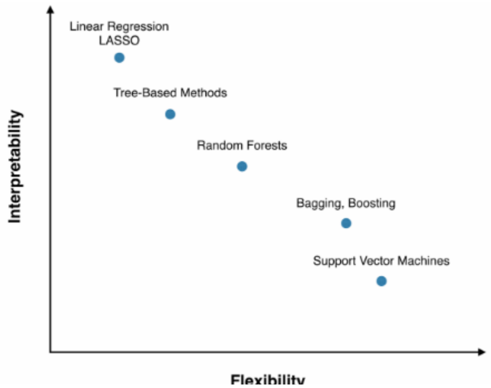
1. Cross-Validation
    - Definition: how accurate is the model "out of sample" (when the model is making predictions for data it has never seen before)
    - Can be simulated by randomly splitting data into 3 folds (parts) and in each round combine 2 of the folds to form the training data and record the average
    - *Caution: If you fail to keep a clear partition between training data and the data on which you test your model's performance, you will overestimate how good your model actually is*
2. Confidence Interval
    - If one model has a more narrow confidence interval than another, then that model will be more reliable
3. Bootstrapping
    - Randomly sample data with replacement and re-run the model on the sampled data to build confidence intervals around our model outcomes

Trade-offs in Model Selection:

1. Bias vs Variance
    - Bias: how closely the model matches the data
    - Variance: the sensitivity of the ML model to the data it is trained on ie. if the model changes its predictions drastically each time it sees new data, then the model is said to exhibit high variance
    - Both higher variance or higher bias leads to higher prediction error and poorer model performance

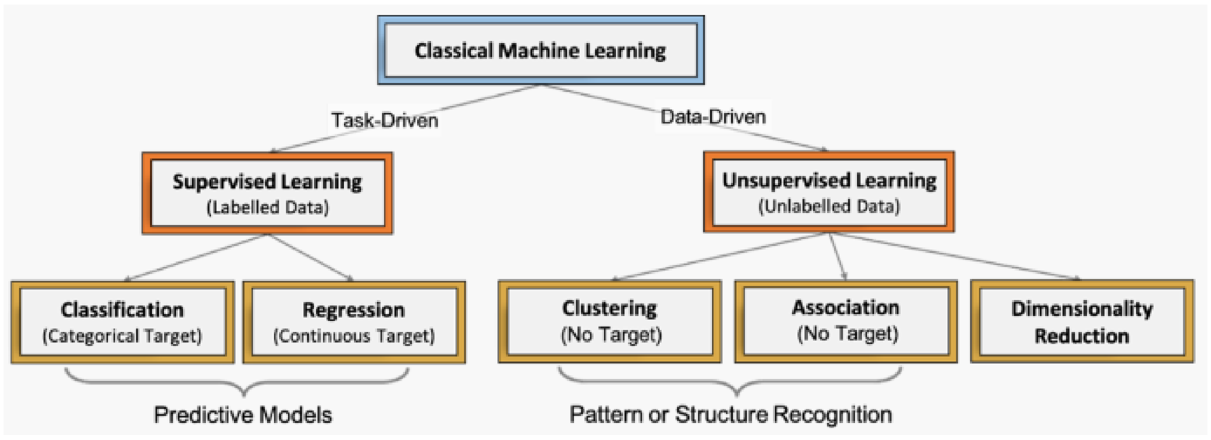


    - Complex ML models (e.g higher order regression polynomials): capture training data but are at a risk of overfitting to noisy or underepresentative training data → high variance when tested on unseen data
    - Simple ML models (e.g low-order or even linear regression polynomials): may induce bias for failing to capture the underlying ground truth fully and underfit their training data
    - Models with high variance: usually complex and represent the training set more accurately but may also represent a large noise component in the training set → predictions less accurate despite added complexity
    - Models with higher bias: tend to be simple but may produce lower variance predictions when applied beyond training set
2. Model Flexibility vs Interpretability
    - Models are usually either simple and non-flexible or complex and flexible
    - Being more flexible implies being more accurate



3. Model Accuracy vs Computational Ease
    - More accurate models are more complex and thus often require more resource-intensive training in terms of computation time or computation space

Section 3: Supervised and Unsupervised Learning



Classical Machine Learning
Task-Driven
Data-Driven
Supervised Learning (Labelled Data)
Unsupervised Learning (Unlabelled Data)
Classification (Categorical Target)
Regression (Continuous Target)
Clustering (No Target)
Association (No Target)
Dimensionality Reduction
Predictive Models
Pattern or Structure Recognition

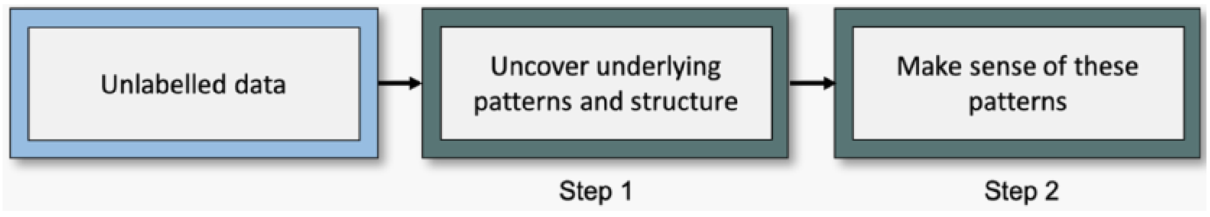
Supervised Learning:

- Regression: when the output variable is in the form of a real value
    - E.g size, weight, height, economic value
- Classification: when the output variable is in the form of a category or class
    - E.g 'pink' or 'white', 'tall' or 'short'
    - Binary classification: labelling input data into exactly 2 categories
    - Multiclass classification: labelling input data into more than 2 categories
- Main objective: leverage previous experiences to produce or collect data ~ predictive analysis
- Applications:
    - Marketing and sales: customer lifetime value, churn rate, sentiment analysis
    - Security: spam filtration, fraud detection

Unsupervised Learning:

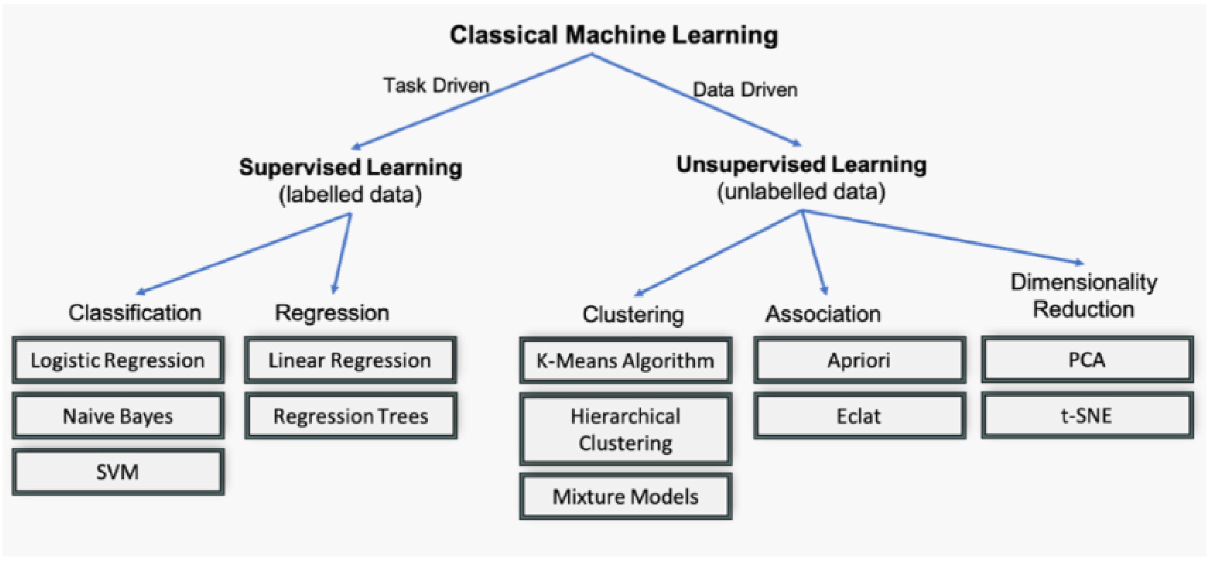
- Clustering: identify patterns or structure within a group of uncategorized data
- Association: identify associations between different data objects amidst heavy databases
- Dimensionality Reduction: reduces the number of input features to a more manageable number
    - Why:
        - The more the number of input features, the more observations will be needed to train accurate models
        - The more features a machine learning model uses, the greater the chances of overfitting
        - A large number of features will make the machine learning model more complex. Thus, it will require longer computation times and higher computation space

- Main objective: identifying unknown patterns and structures that can come in handy for categorization
- Applications:
    - Targeted marketing and customer marketing
    - Visual recognition



Unlabelled data
Uncover underlying patterns and structure
Make sense of these patterns
Step 1
Step 2

Algorithms:



Section 4: Current Use Cases

External Use Cases: provide or directly improve the service to end users

- Recommendation Systems:
    - Definition: it is a composite term used to describe:
        - Historical data of the individual user (to learn about their preferences)
        - Data of other users (to learn about patterns in preferences)
        - Features of the products offered
    - Usually used by companies associated with media content (Netflix, Amazon, Youtube)
- Biometric Identification:
    - Types:
        - Identification: finds a similarity index of the input data with all database entities and finds the most similar individual and assigns it as recognized as long as the similarity index clears a tolerance value [multi-class classification]

        - Verification: input data is then compared to the specific pattern of only a specific individual [binary classification]
- Smart Home Heating and Cooling:
    - ML algorithms used to optimally control the heating and cooling cycles to keep your home at the right temperature for when you return home from work
    - E.g Nest Thermostat

Internal Use Cases: used internally by the company to gain insights that improve its internal business operations

- Demand Forecasting:
    - Models from neural networks called Long Short Term Memory (LSTM) networks to linear regression based Auto Regressive Integrated Moving Average (ARIMA) models, are used to predict the value of a variable at a future point in time
    - Primarily used to predict and forecast the future demand for their product or service
        - Can help the company plan their supply chain operations for sourcing raw materials, retail and warehousing logistics, as well as create projections for profits and losses for investors
- Loan Default Detection
    - data about life events and financial history is collected and used to predict the financial ability to repay loan sums, or to create a metric that quantifies the risk associated with an individual for the purpose of offering insurance products
- Predictive Maintenance with IoT
    - “learn” the normal operating conditions of these complex systems and used to predict when it starts to malfunction
- Fraud Detection
    - Use very large volumes of data in a very short period of time have been developed to earmark and stop illegal transactions

Section 5: Strengths and Limitations of ML

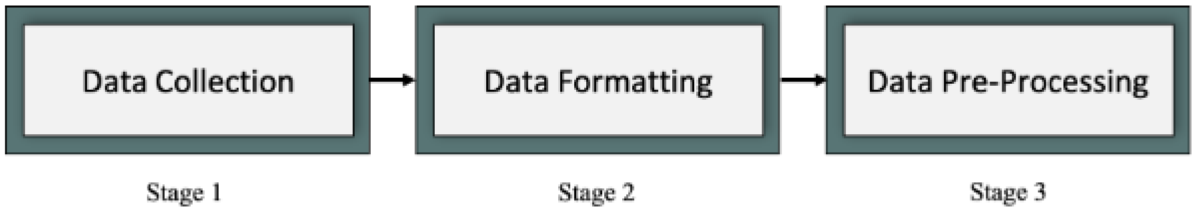
Strengths:

- Automation:
    - Examples: self-driving cars, fraud detection, loan application decisions, etc.
- Continuous Improvement:
    - Examples: recommendation systems
- Data-driven Policies:
    - Traditionally based on ‘a priori’ (before the fact), but ML requires to learn from ‘a posteriori’ (a data-driven approach to solving problems)
    - Examples: spam detection

Limitations:
- Biases:
    - unintentionally letting biases and prejudices creep in the decision-making process
    - 3 key reasons:
        - Biases in the input training data
        - Improper or incomplete model development without explicitly paying attention to bias avoidance
        - Lack of awareness about the social context the machine learning algorithm operates in
- Bad/Limited Data:
    - GIGO: garbage in, garbage out
    - If the data provided is incorrect, the model will learn incorrect relationships
    - Plus bias in input data by past biased human decision-makers will lead to the machine learning model also learning these biases
- Stochastic Models:
    - Deterministic models: where output is fully determining by the inputs to the model and the model parameters
    - Stochastic models: carry inherent randomness and thus, for the same input values, may generate different outputs
        - Most ML models are stochastic in nature
            - Makes models less reliable and harder to interpret

Section 6: Handling and Cleaning Databases

Data preparation pipeline:



Data Collection:
- As the cost of digital storage has cheapened, it has become more economically viable to store these vast and diverse data generated
- Typically a good idea to keep the data raw without too much pre-processing

Data Formatting:
- Standardization of how and where data is stored
    - enhancing interoperability and allowing data to be transferred and used seamlessly

- Represent temporal data in an efficient way to prevent the problem of target leakage or data leakage while building forecasting models

Data Pre-Processing:

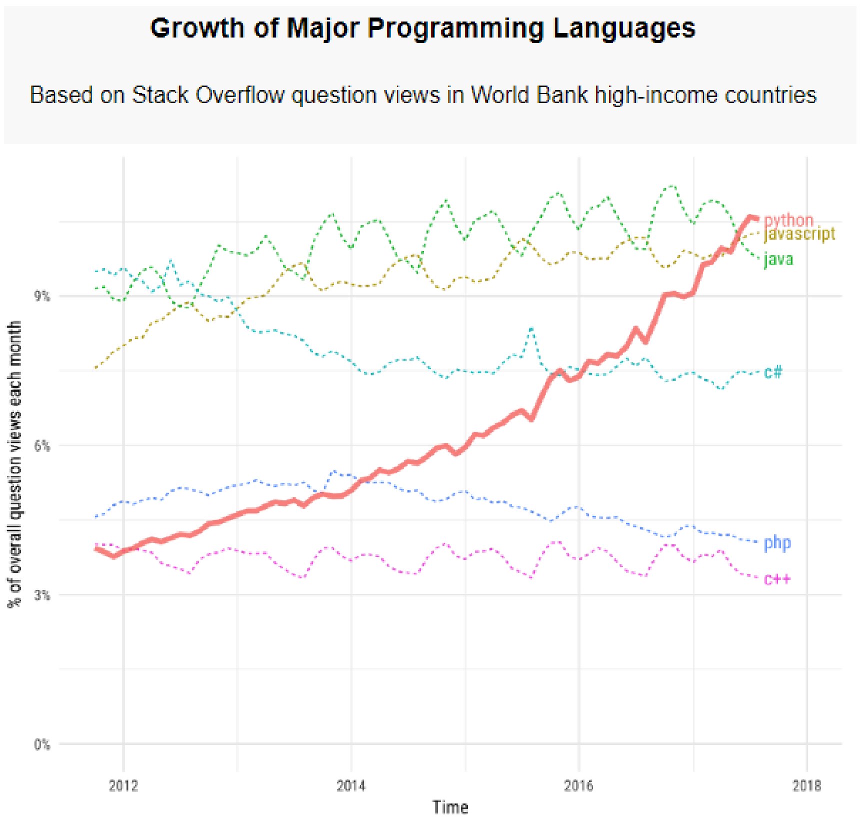
- Made up of 4 steps:
    1. Data Cleaning
        - Carefully dealing with observations with missing or incorrect data either by removing them or using data imputation techniques
        - Removing outliers and anomalies
    2. Data Scaling
        - Normalizing the range of independent variables or features of data
    3. Data Transformations
        - Converting numerical variables (for example, age of a person) to a categorical variable (is the person < 45?)
        - Transforming categorical variables into numerical inputs via one-hot encoding
            - One-hot encoding: A vector in which one element is 1 and all the other elements are 0 and can be used to numerically represent categories present in data
        - Reshaping data
        - Converting units (for example, inch to cm, lb to kg, etc.)
        - Converting units (for example, inch to cm, lb to kg, etc.)
    4. Feature Engineering
        - employing domain-specific knowledge and combining it with available raw data to come up with “new” features that can help us solve the problem

Missing data may be handled in 2 ways:

1. Replacing with the mean value
2. Removing the point all together

Module 2: Python for ML

Section 1: Why Python?



The Popularity of Python - Reasons:

- Simplicity of language allows ML engineers to focus on the challenges of the problem at hand without worrying about the programming aspects of it
- Support for language: has an extensive collection of open source libraries

R vs Python:

- Python is a general-purpose language that offers flexibility for many applications while R is primarily created to cater to statisticians
- Python is well-suited for ML with its extensive libraries while R is well-suited for exploratory data analysis
- Python is more production-friendly can be more easily integrated into the product workflow
- Python's simplicity and 'cleanliness' allows for easier debugging

MATLAB vs Python:

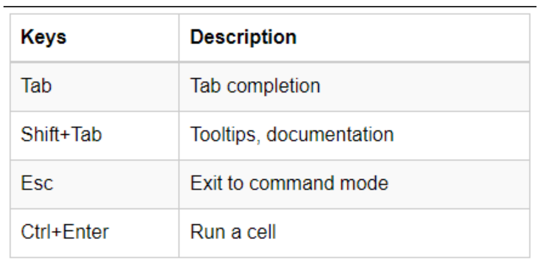
- Python is free and open-source while MATLAB is closed-source and proprietary
- MATLAB may also suffer from higher computation times for certain applications and is harder to integrate into the development workflow

C++ vs Python:

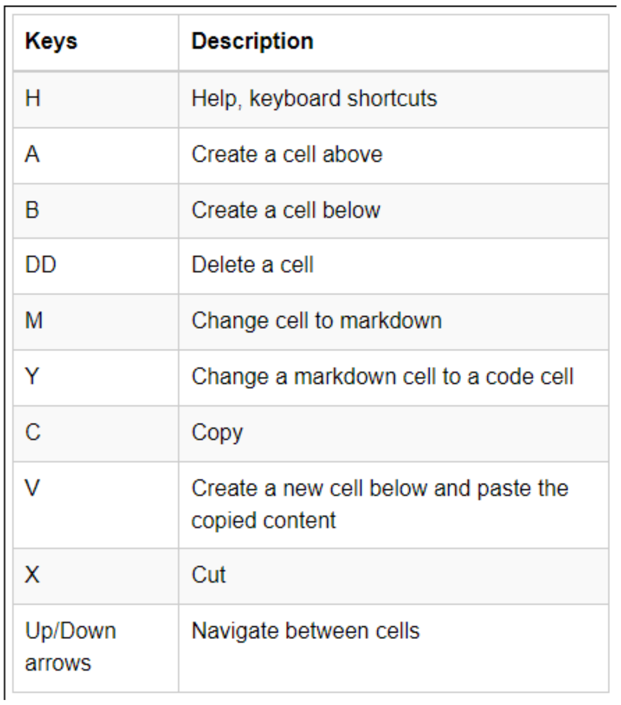
- Python is much more user-friendly and specifically, beginner-friendly
- Python is aimed at simplicity following intuitive conventions, while C++ has more syntax rules and programming conventions

- Since Python is an interpreted language while C++ is a compiled language, and interpretation takes more time, Python can be slower for some large applications

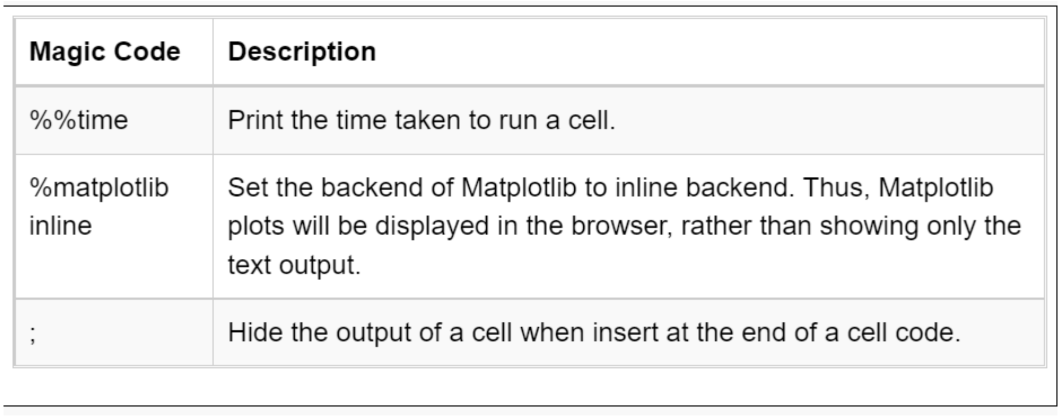
Jupyter Notebook:



Keys	Description
Tab	Tab completion
Shift+Tab	Tooltips, documentation
Esc	Exit to command mode
Ctrl+Enter	Run a cell



Keys	Description
H	Help, keyboard shortcuts
A	Create a cell above
B	Create a cell below
DD	Delete a cell
M	Change cell to markdown
Y	Change a markdown cell to a code cell
C	Copy
V	Create a new cell below and paste the copied content
X	Cut
Up/Down arrows	Navigate between cells



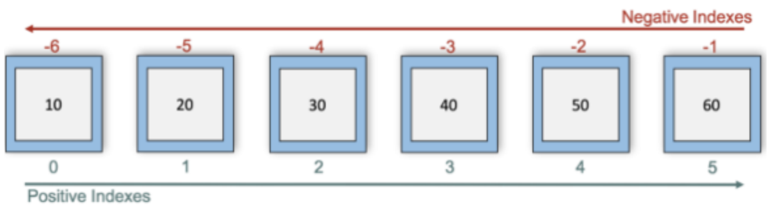
Magic Code	Description
%%time	Print the time taken to run a cell.
%matplotlib inline	Set the backend of Matplotlib to inline backend. Thus, Matplotlib plots will be displayed in the browser, rather than showing only the text output.
;	Hide the output of a cell when insert at the end of a cell code.

Section 2: Variables, Collections, and Control Flow

Control flow: the order in which individual statements of a program are executed or evaluated

Data Collection Types:

- Lists: variable_name = ["some element", 1, 2, 3]



- Sets: set_name = {2, 1, 4, 3} OR set_name = set (list_name)
    - Are unordered and contain only unique items
- Tuples: tuple_name = (1, 2, 3)
    - Are immutable
- Dictionary

Section 3: Creating and Manipulating Lists and Functions

Specify *args and **kwargs is the function parameters to specify arbitrary unnamed and named arguments respectively.

- Kwargs stores data in dictionaries

To check is something is not null:

```
# if not args is None:
```

Lambda functions:

```
# lambda x: 2*(x+1)**2
```

Exceptions:

```
# try:
#         print (x / y)
# except ZeroDivisionError:
#         print("Cannot divide by zero.")
```

Same list, different pointers:

```
# x = [1,2 ,3]
# copy_of_x = x
```

BUT! Slices are stored in different memory; so for a true copy:

```
# true_copy = x[:]
```

List comprehensions:

- # squares_numbers = [x**2 for x in list_of_numbers]
- # even_numbers = [x for x in list_of_numbers if x % 2 == 0]
- # even_squares = [x**2 for x in list_of_numbers if x % 2 == 0]
- # even_number_pairs = [(x,y)
- #   for x in list_of_numbers if x % 2 == 0
- #   for y in list_of_numbers if y % 2 == 0
- #   if x > y]

Enumerating lists:

```
# directory = ["Abdul", "Carl", "Davis", "Julia", "Ning", "Roberta"]
# list(enumerate(directory))
```

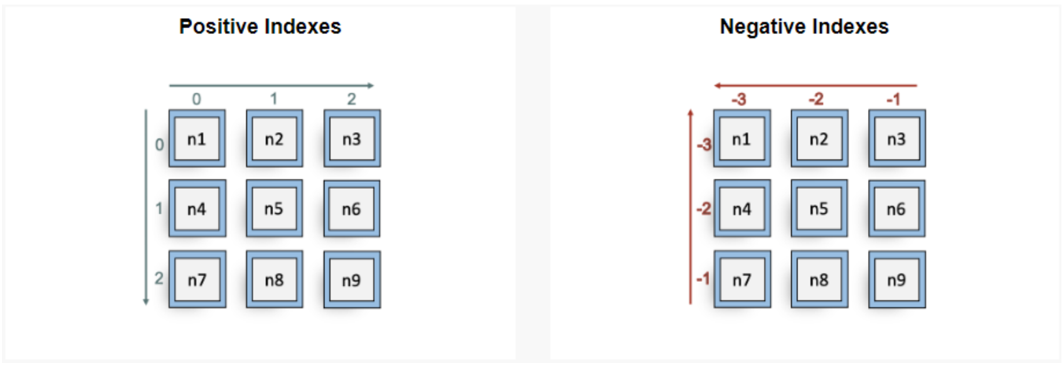
Returns: [(0, 'Abdul'), (1, 'Carl'), (2, 'Davis'), (3, 'Julia'), (4, 'Ning'), (5, 'Roberta')]

Manipulation of Sets:
- Union: # A|B
- Intersection: # A&B
- Difference: # A-B
  # B-A
- Symmetric Difference: # A^B
  # B^A

Section 4: Popular Libraries and Functions

Python Libraries for ML:
# import random as rnd
# print(rnd.random())

Popular Libraries for ML:
- NumPy: large and multi-dimensional arrays and matrices along with a large collection of mathematical constants, functions, and operators
  - Using pi: # numpy.pi
  - Linear arrays: arange(start point, end point, stepsize) # numpy.arange(0, 10, 1)
    - linspace(start point, endpoint, No of points) # np.linspace(0, 1, 9)
  - Standard arrays:
    - Array of 1s: # np.ones(5)
    - Array of 0s: # np.zeros(3)
    - Array of random numbers: # np.random.random((3))
    - 2D Arrays:
      - Array of 1s: # np.ones([3,3])
      - Array of 0s: # np.zeros([2,5])
      - Array of random numbers: # np.random.random((3,4))



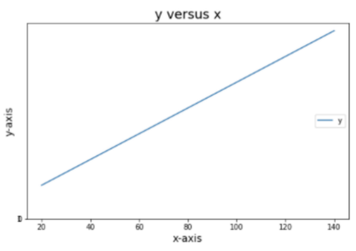
  - Slicing Arrays:
    - x[0:2] prints the first 2 elements of the 1D array
    - z[:,0] prints the first column of the 2D array
    - z[0,:] prints the first row of the 2D array

    - z[1:,1]) prints the second element onwards from the second column
  - Formatting data:
    - zip() function takes iterable objects like lists or tuples as arguments, and returns a series (1-D array) with elements from each iterable object
    - np.array(list(zip(list(np.arange(0,10,1)),list(np.arange(0,20,2))))) →
    - x1.shape → dimensions in a tuple
    -

```
List x1:
 [[ 0  0]
 [ 1  2]
 [ 2  4]
 [ 3  6]
 [ 4  8]
 [ 5 10]
 [ 6 12]
 [ 7 14]
 [ 8 16]
 [ 9 18]]
```

- Matplotlib: creating plots

Simple Plot

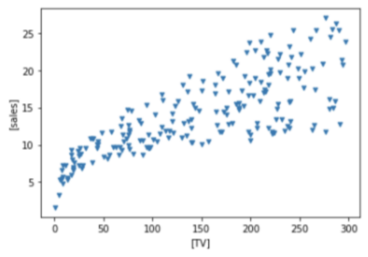


Description: A simple plot creates a graph showing a line that connects all data points. It is useful to visualize simple and multi-linear equations.

Method: plot()

Syntax: plot(x,y); x and y are the most common attributes

Scatter Plot



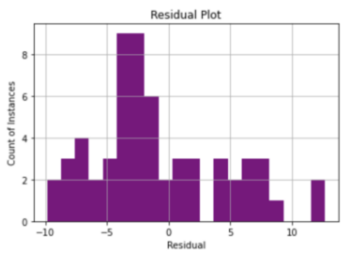
Description: A scatter plot creates a graph showing the distribution of the datapoints across the x-axis.

Method: scatter()

Syntax: scatter(x, y, c, s, marker, camp, alpha)

- s: marker size
- c: list of colours
- marker: Specify the shape of the marker
- cmap: colormap
- alpha: set transparency of markers, 0 (transparent) and 1 (opaque)

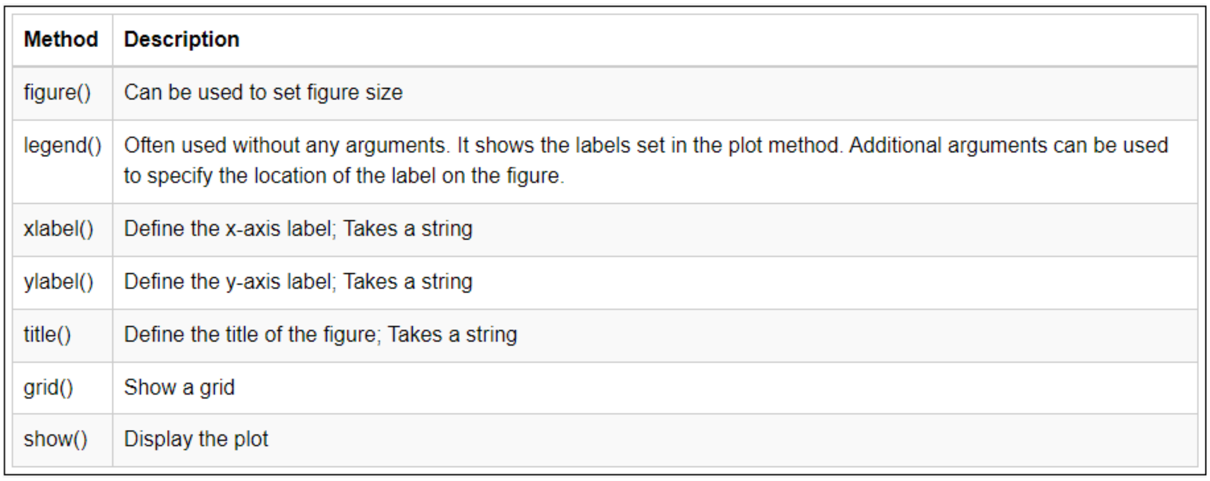
Histogram



Description: A histogram plot creates a column-based histogram graph that shows the distribution of a variable across a dataset. The x-axis shows the range of values that the variable can take while the y-axis shows the count.

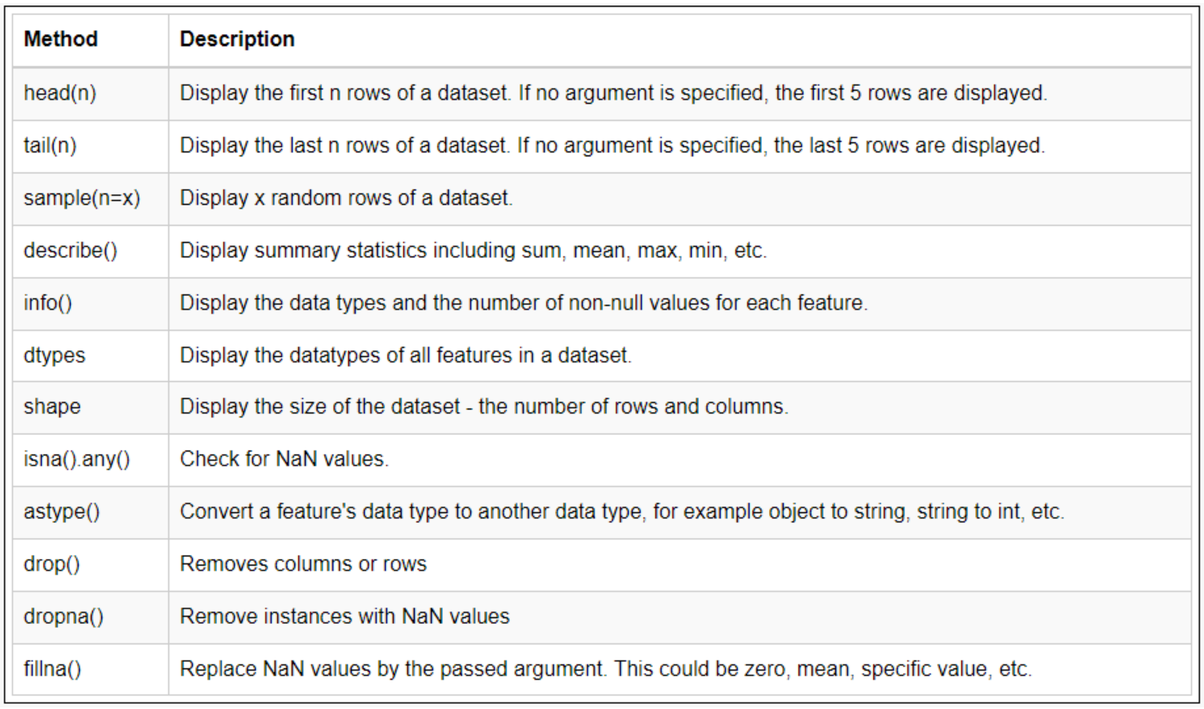
Method: hist()

Syntax: hist(x, bins, color, label)
Note that for the label to show, you need to include the legend() method



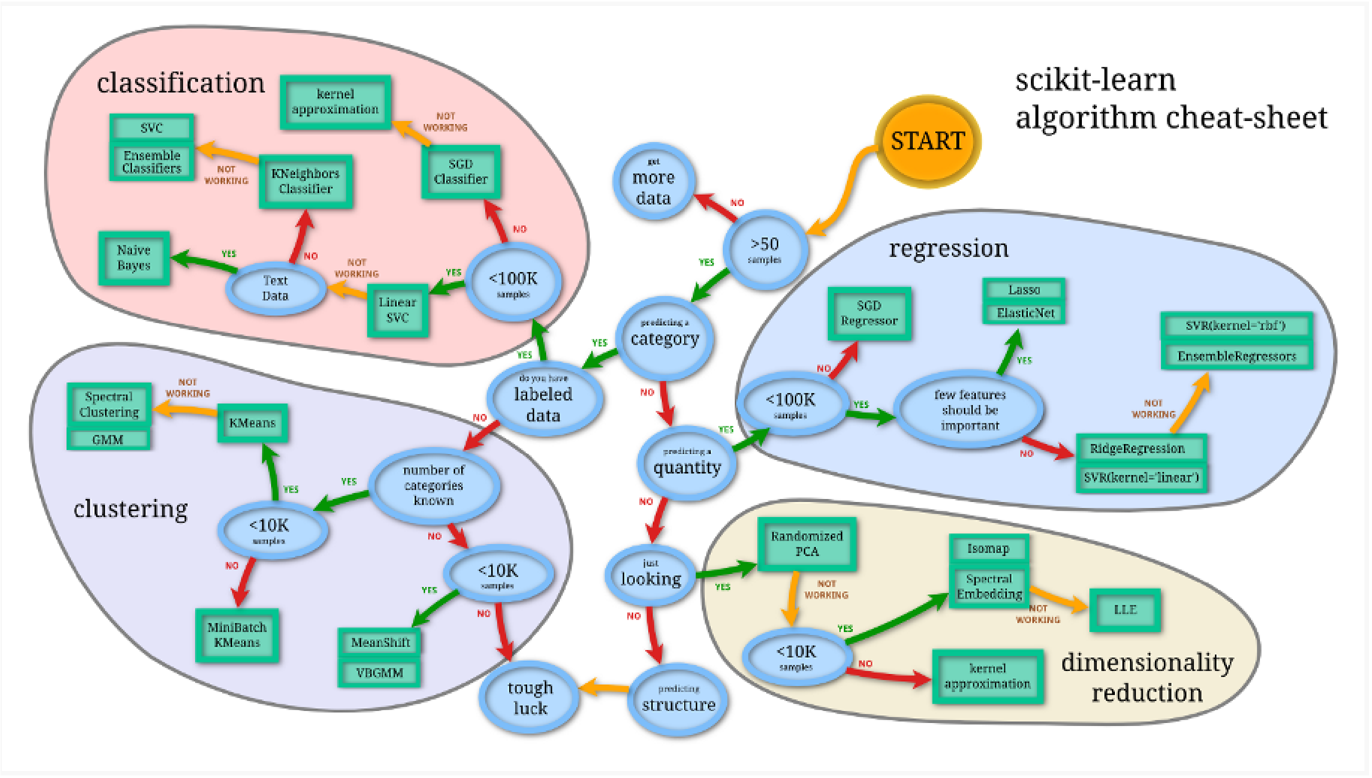
Method	Description
figure()	Can be used to set figure size
legend()	Often used without any arguments. It shows the labels set in the plot method. Additional arguments can be used to specify the location of the label on the figure.
xlabel()	Define the x-axis label; Takes a string
ylabel()	Define the y-axis label; Takes a string
title()	Define the title of the figure; Takes a string
grid()	Show a grid
show()	Display the plot

    - plt.plot(x,y,'^-',label='y') # '^' Use a triangle as a marker for the plot
      # '-' Connect the points
    -
- Pandas: cleaning, extraction, preparation, grouping, and filtering data and helps handle different formats including CSV, Excel, and HTML files
    - 2 Main Data Structures:
        - Series: 1D data structure; a single column of data where the data is of the same type
            - pd.Series([25,31,45]);
        - Dataframe: 2D labelled data structure with multiple columns corresponding to different features; collection of series
            - pd.DataFrame({'Employee_Name':employee_name,'Age':age})
                - Where employee_name and age are series
    - Reading csv
        - pd.read_csv('advertising.csv')
            - data.head() gives you the first 5 lines
            - data.drop(['radio'],axis=1) deleted the radio column
            - data_TV_high = data[data.TV > 200] select rows such that TV ad spend is larger than 200



Method	Description
head(n)	Display the first n rows of a dataset. If no argument is specified, the first 5 rows are displayed.
tail(n)	Display the last n rows of a dataset. If no argument is specified, the last 5 rows are displayed.
sample(n=x)	Display x random rows of a dataset.
describe()	Display summary statistics including sum, mean, max, min, etc.
info()	Display the data types and the number of non-null values for each feature.
dtypes	Display the datatypes of all features in a dataset.
shape	Display the size of the dataset - the number of rows and columns.
isna().any()	Check for NaN values.
astype()	Convert a feature's data type to another data type, for example object to string, string to int, etc.
drop()	Removes columns or rows
dropna()	Remove instances with NaN values
fillna()	Replace NaN values by the passed argument. This could be zero, mean, specific value, etc.

- Scikit-learn (Sklearn): offers pre-built implementations of supervised and unsupervised machine learning algorithms for the construction of an entire pipeline within a few minutes



- Example
  - # Import the class LinearRegression from sklearn.linear_model

    from sklearn.linear_model import LinearRegression
    model = LinearRegression(fit_intercept=True)

```
# Fit a linear regression model to (X,y) data generated above
model.fit(X[:, np.newaxis], y)

# yfit denotes our linear model's predictions over Xfit, over the range 0
to 1000
# in steps of 10. We will use (Xfit, yfit) to plot the predicted linear
model.

Xfit = np.linspace(0, 10, 1000)
yfit = model.predict(Xfit[:, np.newaxis])

plt.scatter(X, y)
plt.plot(Xfit, yfit);
```

- model.coef_[0] → get slope
- model.intercept_ → geet y-intercept

Module 3: Supervised Learning

Section 1: Why Supervised Learning?

Supervised learning involves inferring the mapping between inputs and outputs based on example input-output pairs

Types of Supervised Learning Tasks:

- Classification
    - A set of already classified and labelled data is used to construct a machine learning model that can then classify unknown entities into defined classes
    - E.g the image recognition algorithm with “apples” and “not apples”
- Regression
    - identifying the influence or effect of a set of input features on an output feature
    - E.g predicting the wages of individuals based on features such as their age, education, experience, and so forth, is a regression task

<u>Successful Use Cases</u>

Detecting Malicious Links:

- identifying malicious links in emails is critical for IT security departments across organizations

Algorithmic Trading:

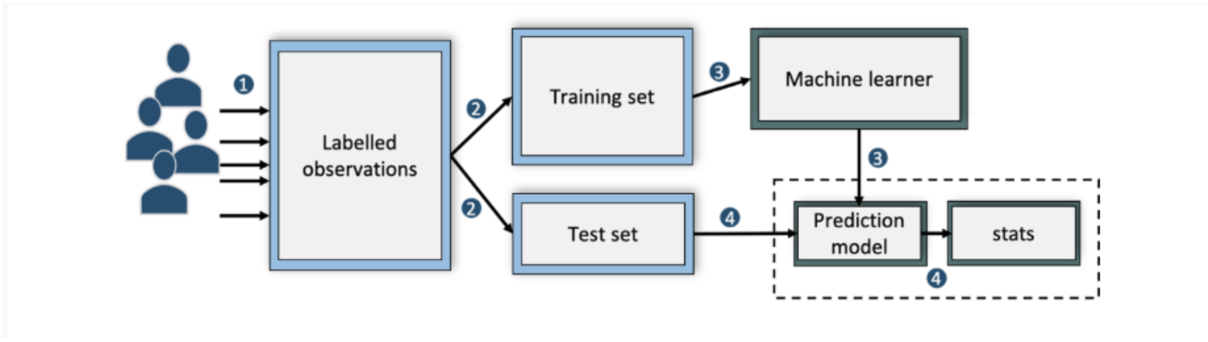
- a machine learning algorithm that has been trained on past data to predict and forecast the movement of stocks and options

Smart Farming:
- Work with IoT and historical data collected from these sensors to optimally fine tune various operational decisions undertaken by farmers, such as deciding the level of soil moisture, time, and depth of seed planting, and so forth, to achieve high crop yields

Internet of Trains:
- Embedded ML in sensors in trains and tracks to predict wear, tear, and failure on trains and tracks
  - to ensure minimization of train delays due to failures by preemptively fixing what is predicted to fail by their machine learning models

Predictive Maintenance:
- Leverage data from IoT sensors to monitor and maintain their elevators and aircraft engines, respectively

A Deeper Dive into Supervised Learning



Step 1: Attain historical data of inputs and outputs (labelled data) to train the algorithm
Step 2: Separate the training and test data
Step 3: Use the training data to train the appropriately chosen ML model
Step 4: After model has been trained and the parameters fine-tuned, evaluate the accuracy and performance of the model

Executive and Managerial Considerations

Data Availability:
- Garbage in, garbage out
- Is there labelled data available? If not labelled, can it be labelled now? Is it worth investing the effort to manually label data? Should the data labelling be carried out in-house or should it be outsourced?

- 70% of a data scientist's time is spent in procuring and preparing data
- often leads to cost and time overruns and an ‘overpromise-underdeliver’ vicious cycle

Accuracy:
- What is not measured cannot be managed
- Is the accuracy of the ML model good enough?
    - Consider when the cost of a false negative far outweighs the cost of a false positive
- Clear thresholds for accuracy and precision need to be laid out against which model performance will be evaluated

Successful supervised learning projects typically have:
- Plenty of available labelled data, or data that can accurately and cheaply be labelled by humans.
- Quantifiable objectives and targets set at the beginning against which model performance can be compared

Section 2: Regression Models

Regression Models: supervised learning techniques typically used to identify relationships between quantitative data typically used to predict or forecast an outcome variable in terms of a set of input variables

2 Methods: Linear Regression and Regression Trees

Linear Regression:
- Uses linear equations to predict the outcome (dependent/predicted) variable
- Objective: to identify the influence of each of the explanatory variables on the dependent variable and their relative effects
- Types of Linear Regression:
    - Simple: uses a single explanatory variable and the outcome variable
    - Multiple: natural generalization of multiple explanatory variables and the outcome variable

Regression Trees:
- Decision tree learning is a predictive method commonly used in data mining to find patterns in data using a top-down approach (a variable is chosen at each step that best splits the set of items)
- Decision tree models are simple to understand and interpret and require little data preparation
- 2 Types:
    - Regression Trees: perform predictions based on many numeric inputs and give a single numeric output

        - Classification Trees

<u>Simple Linear Regression</u>
Equation: y = a + bX

- Where X is the explanatory variable and y is the outcome variable, slope is b and a is the y-intercept
- "*Regressing y on X*"
- Using the training data we identify the line of best fit to find the slope b and the intercept a

Estimating the coefficients:

- Most common method is using least-squares to compute the slope and intercept
    - Calculates the best-fitting line for the observed data by minimizing the sum of squares of the vertical deviations from each data point to the line
        - Minimizes the residual sum of squares (RSS)
    - Since the deviations are first squared and then summed, there are no cancellations between positive and negative errors
- Vertical deviations are also referred to as errors or residuals which are calculated as follows:

  Residual = actual value - predicted value
- Points are defined as y_i and X_i where i can be 1, … , n where n is the total number of points

Optimization Problem:
Find a and b such that RSS is minimized where:

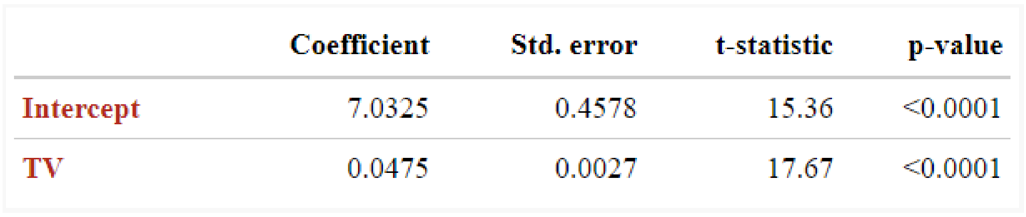
$$\text{RSS} = \sum_{i=1}^{n} [y_i - (a + bX_i)]^2$$

- We do not need to worry about the manual approach, this problem is solved by an algorithm that outputs the values of a and b

Note: The value of b is particularly important - it tells us the influence of X on y:
If b > 0, then there is a positive effect
If b < 0, then there is a negative effect

<u>Linear Regression Results Interpretation:</u>
E.g X is the ad spend on TV and y is the number of units sold

By running the ordinary least squares (OLS), we obtain the following:



	Coefficient	Std. error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	<0.0001
TV	0.0475	0.0027	17.67	<0.0001

We know that the above corresponds to the following linear model:

$$Y = 7.0325 + 0.0475X$$

Therefore, since b (the coefficient of the predictor variable) is positive, the ad spend on TV has a positive effect on sales

- For an increase in ad expenditure on TV by $1000, we predict an increase in sales by 0.0475 * $1000 = $47.50

The p-value tells us whether the relationship between the predictor (TV ad spend) and the predicted (Sales) is statistically significant and reliable.

- Low p-value ($p < 0.05$): predictor variable is likely to carry a meaningful and statistically significant relationship with the predicted variable
- If $p > 0.05$, we cannot be confident in the $47.50 estimate

<u>Linear Regression Assumptions and Diagnosis</u>

Outliers:

- Good practice to remove outliers before fitting a linear regression model

Extrapolation:

- Making predictions outside the range of the data is inappropriate and will almost certainly yield inaccurate answers

Non-linearities:

- If the true relationship is in fact non-linear, the model tends to underfit, and thus the conclusions that we draw from the linear regression model will naturally be inaccurate
- If trend is exponential, can apply log onto data and the trend will appear linear
- A linear regression model also assumes that the errors or residuals of the model are normally distributed with a mean of zero. Ideally, some of the residuals will be positive, others negative, averaging to zero, and showing no discernible pattern - the presence of a pattern may indicate a problem with some aspect of the linear model

Collinearity:

- model is less reliable when two or more of the predictor variables are collinear (closely related or correlated to each other)
    - collinearity reduces the accuracy of the estimates of the regression coefficients
- When faced with collinearity, a simple solution is to drop one of the problematic variables from the regression

- Does not compromise the regression fit much as the presence of collinearity implies that the information that this variable provides about the response is redundant in the presence of the other variables

<u>Multiple Linear Regression</u>

Equation: $y = a + b_1X_1 + b_2X_2 + b_3X_3 \ldots$

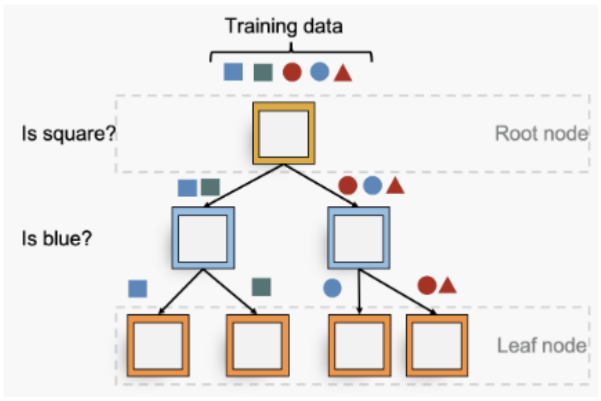
- Each predictor variable will be associated with its own coefficient

<u>Regression Trees</u>

- A good way to handle a more complicated non-linear relationship

Theory:

- Data is evaluated and split up multiple times according to certain cutoff values in the predictor variables to generate a traditional decision tree
- Decision trees can get very complicated and tend to overfit



Optimal Feature Space Split:

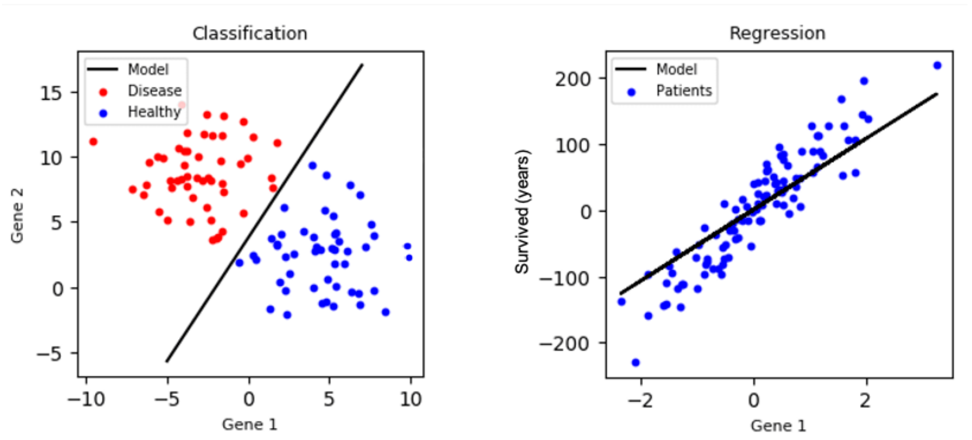
- Objective: identify the "right" splits of the data which will allow for an accurate prediction
- Need to choose the most optimal split that will ensure the highest prediction accuracy by minimizing the sum of squared errors as in the case with linear regression

Section 3: Classification Models

Instead of a quantitative predicted value, classifications evaluate a qualitative/categorical predicted value

2 Types:

- Binary Classification: the response variable can only be positive or negative
- Multi-class Classification: more than 2 possible outcomes, eg grades A, B, C, D

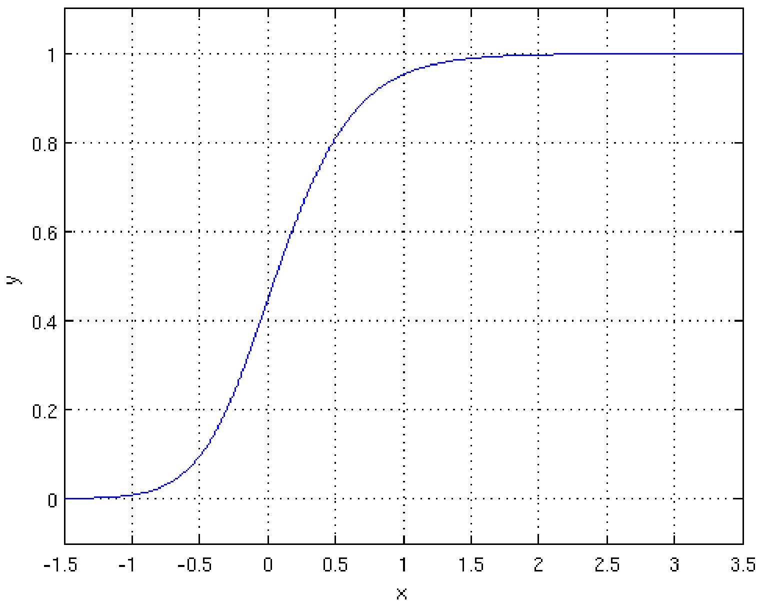


Logistic Regression:

- Used for binary classification
- Equation:

$$p(X) = \frac{e^{a+bX}}{1 + e^{a+bX}}$$

  - retains the linear model $a+bX$ but applies a transformation to squeeze the output of the linear model to lie between 0 and 1



- Uses maximum likelihood estimation (MLE)
- the coefficient b tells us how the odds, that is the ratio between the probability of default to the probability of no-default, change with respect to the balance
  - precisely, for a unit change in X, the odds change by e^b where e is approximately equal to 2.71828 and is the base of the natural logarithm
- E.g

	Weight	Odd Ratio	Std. Error
Intercept	-2.91	0.05	0.32
Hormonal contraceptives y/n	-0.12	0.89	0.30
Smokes y/n	0.26	1.29	0.37
Num. of pregnancies		1.04	0.10
Num. of diagnosed STDs	0.82	2.26	0.33
Intrauterine device y/n		1.85	0.40

- An increase in the number of diagnosed STDs (sexually transmitted diseases) changes (increases) the odds of cancer vs. no cancer by a factor of $e^{0.82}$=2.26, when all other features remain the same

Naive Bayes:

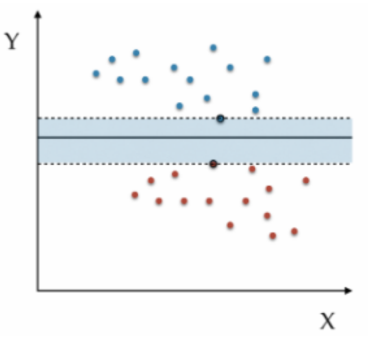
$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

- Based on the Bayes theorem
    - B could be an observation with feature (X)
    - A could be the the probability that this observation belongs to class A (C)

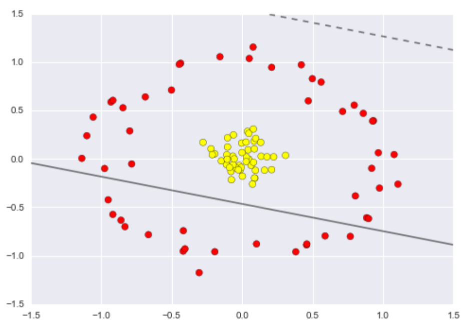
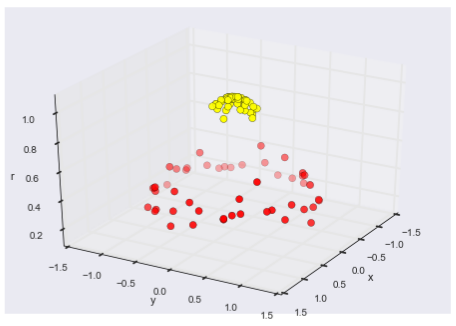
$$P(C/X_1,X_2) = \frac{P(X_1/C)P(X_2/C)P(C)}{P(X_1)P(X_2)}$$

- Is considered a straightforward and fast classification algorithm; suitable for large data; commonly used in various applications such as spam filtering, text classification, sentiment analysis, and recommender systems

Support Vector Machines:

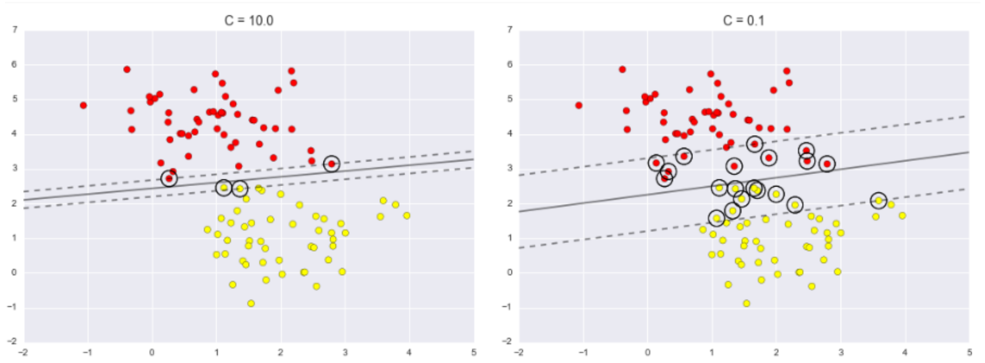


- Discriminative classifiers: learn how to discriminate between the various classes based on available data and features present in the dataset
- Base assumptions less on the underlying data distribution and more on the quality of observed data
- Finds a line or curve (in two dimensions) or manifold (in multiple dimensions) that divides the classes from each other
- Non-linear discriminants:

    - Elevate data by adding an extra dimension and using the radial basis function

- Overlapping data:
    - SVMs soften the margin using a fudge-factor, allowing some points to creep into the margin if that allows a better fit
- Hardness of margin is controlled by a tuning parameter C
    - A large C → margin is hard → points cannot lie in it

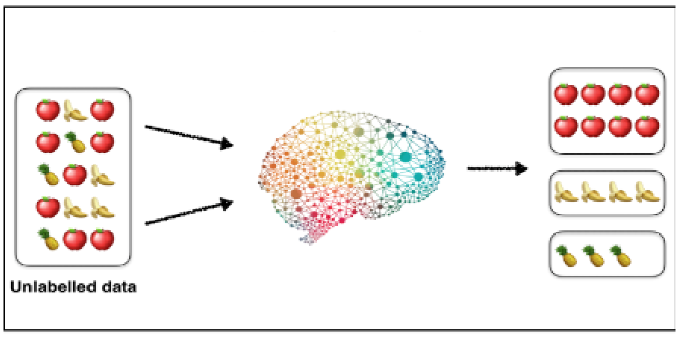


    - Optimal value of the C parameter will depend on your dataset, and should be tuned using cross-validation or a similar procedure
- Pros:
    - Low resource consumption: Their dependence on relatively few support vectors means that they are very compact models, and take up very little memory.
    - Fast predictions: Once the model is trained, the prediction phase is very fast.
    - Work with high-dimensional data: Because they are affected only by points near the margin, they work well with high-dimensional data — even data with more dimensions than samples, which is a challenging regime for other algorithms.
    - Versatile: Their integration with kernel methods makes them very versatile, able to adapt to many types of data
        - Kernels are functions that are used to transform data belonging to different classes to the required form e.g non-linear to linear.
- Cons:
    - For large numbers of training samples, the computational cost can be prohibitive.
    - The results are strongly dependent on a suitable choice for the hyperparameter C. This must be carefully chosen via cross-validation, which can be expensive as datasets grow in size.
    - The results do not have a direct probabilistic interpretation.
- SVMs are used only after the other classification methods fail to provide accurate answers

Module 4: Unsupervised Learning

Section 1: Why Unsupervised Learning?

Unsupervised learning deals with the task of learning similarities and associations in unlabelled data by searching for patterns and infers underlying associations in datasets without labels



It is NOT possible to easily check the performance of an unsupervised learning algorithm as we do not know the “right answer”/“true label” for any observation

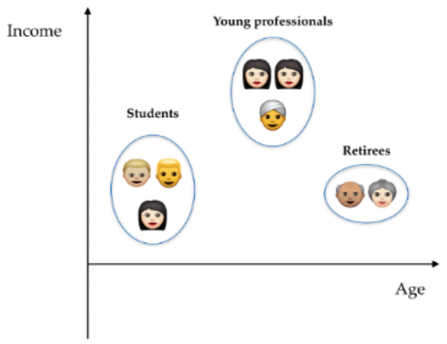
Types of Unsupervised Learning Tasks:

- Clustering: performs data analysis to find natural groups or clusters in a given data and does not do predictions - formed clusters have similar observations
- Association Learning: used to discover associations and relationships between features in a given dataset by looking for co-occurrences, and specific patterns, using metrics like support and confidence
    - E.g future marketing or advertisement
- Dimensionality Reduction: used to reduce the number of input features in big datasets that has a lot of features, which allows the performance meaningful analysis and reduces computation time and resources

<u>Successful Use Cases</u>

Customer Segmentation:

- identify groups or clusters of shoppers with similar purchasing patterns to help online retailers tailor marketing campaigns



Targeted Advertising:

- a search engine might selectively display targeted ads or highlighted search results to individuals based on the browsing patterns of similar individuals

Anomaly Detection:

- identifying malicious actors
- anomaly detection, also sometimes known as outlier detection, is useful in identifying anomalous data or observation that differs substantially from other routine non-malicious observations. This is useful in identifying and flagging unusual usage patterns by hackers

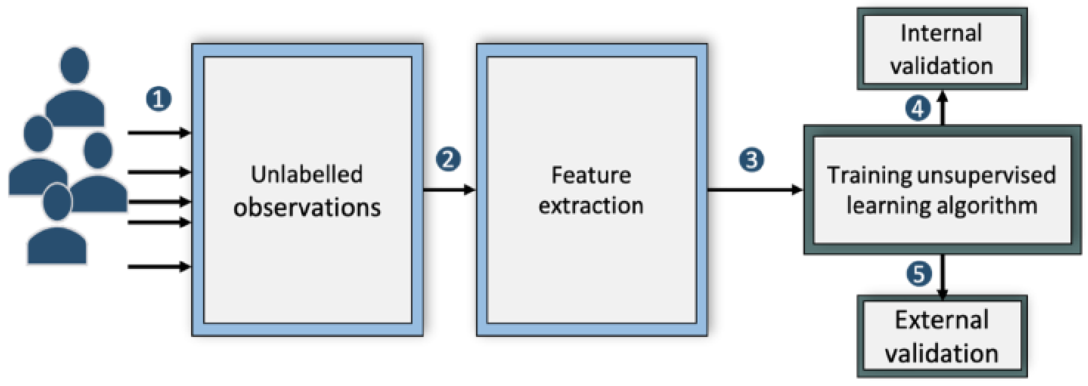
Cybersecurity:

- Flag fraudulent transactions or identify the click patterns of web crawlers to stop them from harvesting confidential or proprietary data

Medical Research:

- search for patterns among the genetic sequences of patients to identify sub-types of a particular disease to obtain a better understanding

Project Pipeline:

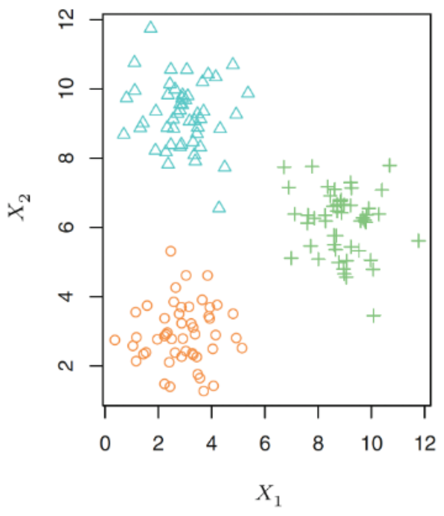


Step 1: Collect data
Step 2: Focus the features to match topic
Step 3: Train the algorithm
Step 4 & 5: Extract clusters/associations and check for internal and external validity

External and Internal Validation:

- Internal: evaluating the goodness of model outputs in terms of whether observations identified as belonging to the same cluster are closer to one another than to observations in other clusters
    - verifies that there is within-cluster similarity and across-cluster dissimilarity
    - Metrics: compactness, connectivity, and separation
- External: manual analysis of the unsupervised learning outputs and identification of whether the uncovered clusters or associations are meaningful
- the algorithm will be internally valid but the clusters uncovered by the algorithm may not correspond to meaningful groups and therefore it is said to fail external validity - the model results will be of limited practical utility.

Section 2: Clustering Models



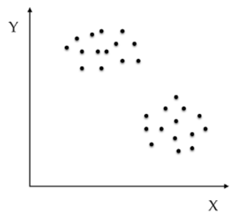
A dataset with observations that can be clustered into three groups based on two features X1 and X2. So as observations are well-separated, even a simple clustering algorithm such as the K-Means algorithm will perform well on such a dataset.

<u>K-Means Algorithm:</u>

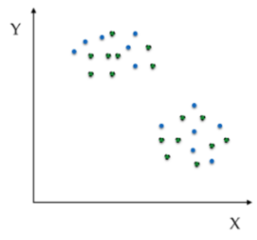
Overview:

- a simple approach for partitioning a dataset into K distinct clusters where each data point belongs to only one cluster
    - K is specified by data scientist = a pre-specified number of clusters
- Central idea is to minimize within-cluster variation
    - The within-cluster variation is a measure of the amount by which the observations within a cluster differ from each other
        - E.g the sum of the pairwise squared deviations of observations belonging to the same cluster

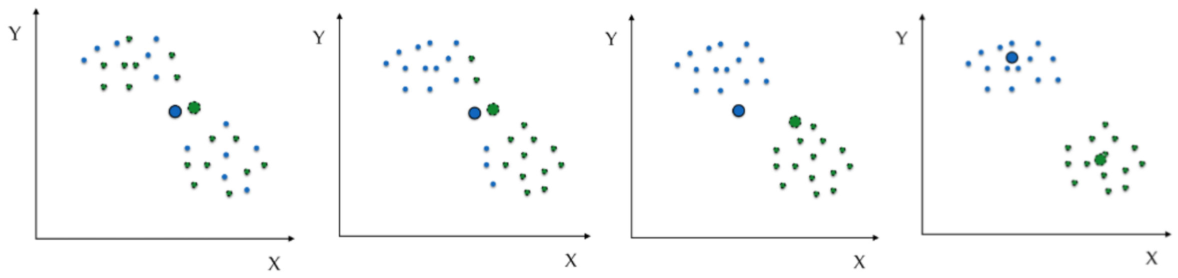
Inner Workings:



1. Specification Step
    - Specify K number of clusters
    - E.g K = 2
2. Initialization Step
    - Randomly assign 1 to K to each observation



3. Iteration Step
    - For each cluster, compute the average of all observations belonging to that cluster = compute the cluster centroid
    - Reassign each observation to the cluster whose centroid is closest
    - Iterate the previous 2 steps until the cluster assignment stop changing
    - Compute the new cluster centroid, reassign observations to closest centroid until the cluster assignment stops changing



Pros and Cons:

- Main pro → simplicity: easy to implement and interpret
- Main con → pre-specified K clusters, how do we know how many?

<u>Hierarchical Clustering:</u>

Overview:

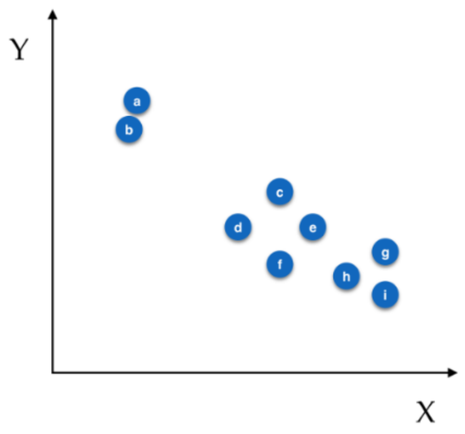
- Defines a dissimilarity measure between each pair of observations
    - observations which are sufficiently far apart or have feature values that are substantially different from one another are highly dissimilar whereas observations with feature values close to each other are not too dissimilar

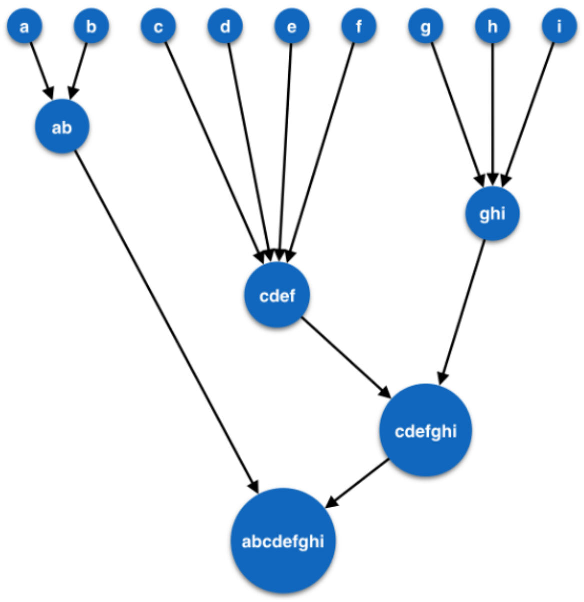
Types:

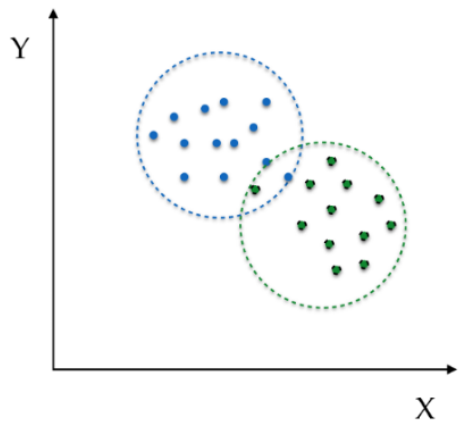
- Agglomerative clustering: works in a bottom-up manner.
- Divisive hierarchical clustering: works in a top-down manner.

Example:

- First algorithm assumes every observation belongs to its own cluster, so we'd have N clusters
- Two most similar clusters are fused
    - If more than two clusters are equally similar to one another, then all these clusters are fused
- process is repeated iteratively until all observations are part of a single cluster
- The result is a dendrogram: an interpretable tree-based representation of the observations







<u>Soft Clustering and Mixture Models:</u>

- When some observations are a close tie between belonging to two clusters

Soft Clustering:

- identifies clusters and also provides a probabilistic assessment of the likelihood of each observation belonging to a specific cluster
- prioritizes observations based on the confidence we have in them belonging to a particular cluster or another
    - E.g in anomaly detection, it is of practical significance whether we are 95% certain a particular user belongs to a fraudulent cluster versus whether we are 55% certain

Mixture Models:

- Suppose there are K clusters and N observations
- Procedure:
    - Each of the K clusters is associated with a unique probability distribution of feature values with a corresponding mean and variance.
    - Each of the N observations has an associated probability of belonging to each of the K clusters. These are known as mixture probabilities.
    - A mixture model is therefore specified by three ingredients: the number of clusters K, the mean and variance of feature values in each cluster, and the mixture probabilities for each observation

Section 3: Lab

Section 4: Association Learning

Eg the "frequently bought together" feature in Amazon

Useful Metrics:

Support: measures how frequently the set of items appears in the data with respect to the total number of transactions

$$\text{Support}(X) = \frac{\text{Number of transactions that include item X}}{\text{Total number of transactions}}$$

Confidence: a conditional probability that represents the number of times a rule is said to be true. To find the likelihood of item Y being purchased when item X is purchased, you can find the P(Y/X) or use:

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X, Y)}{\text{Support}(X)}$$

Apriori Algorithm:

- Uses a building-up or a bottom-up approach to identifying associations

Transactions
{apple, bread, cheese, milk}
{apple, bread, milk}
{apple, bread}
{bread, cheese, milk}
{bread, cheese}
{cheese, milk}
{bread, milk}

Example:

- Objective is to identify sets of items frequently purchased together
- "Frequently purchased": we would need a support threshold, let's make it 3 transactions
- Procedure:

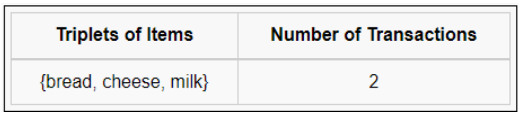
1. Consider the number of transactions each individual item appears in

Items	Number of Transactions
Apple	3
Bread	6
Cheese	4
Milk	5

2. Checks whether each individual item crosses the support threshold; if not, it is removed from further analysis
3. Complete step 1 using pairs of items →

Pairs of Items	Number of Transactions
{apple, bread}	3
{apple, cheese}	1
{apple, milk}	2
{bread, cheese}	3
{bread, milk}	4
{cheese, milk}	3

4. Repeat step 2: {apple, cheese} and {apple, milk} are removed or "pruned"
5. Consider Step 1 with triplets



Triplets of Items	Number of Transactions
{bread, cheese, milk}	2

6. Repeat Step 2: The triplet does not surpass the threshold
7. The algorithm terminates

- Results of the items frequently purchased together: {apple, bread}, {bread, cheese}, {bread, milk}, {cheese, milk}.

Pros and Cons:

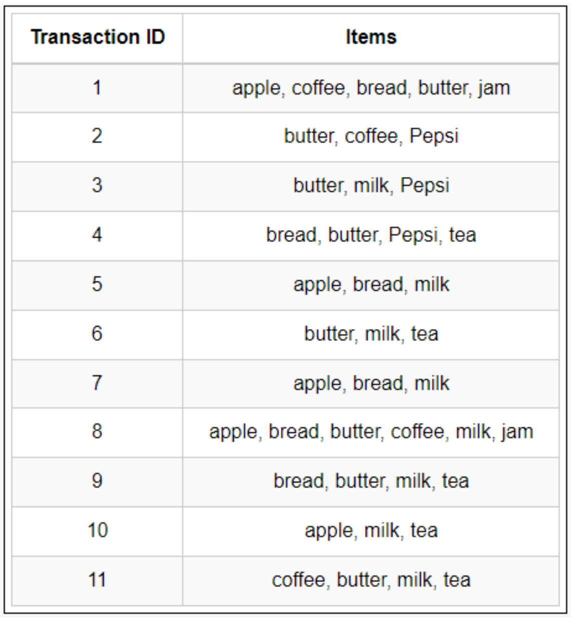
- Main pro: simplicity; easy to implement and interpret
- Main Con: it is computationally inefficient. It involves generating and checking the appearance of a large number of subsets and therefore is not practical for datasets with a large number of items and transactions

<u>ECLAT Algorithm:</u>

- Equivalence Class Clustering and Bottom-Up Lattice Traversal algorithm

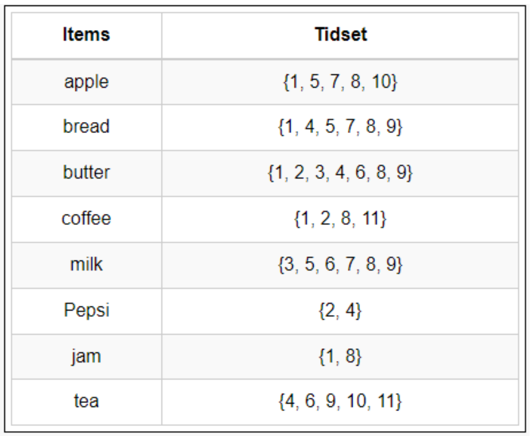
Example:

- Objective is to identify sets of items frequently purchased together
- Procedure:



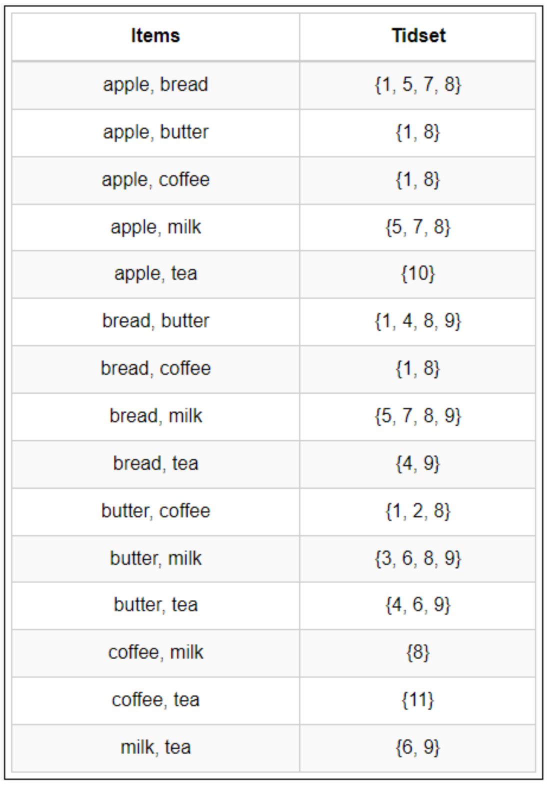
Transaction ID	Items
1	apple, coffee, bread, butter, jam
2	butter, coffee, Pepsi
3	butter, milk, Pepsi
4	bread, butter, Pepsi, tea
5	apple, bread, milk
6	butter, milk, tea
7	apple, bread, milk
8	apple, bread, butter, coffee, milk, jam
9	bread, butter, milk, tea
10	apple, milk, tea
11	coffee, butter, milk, tea

1. Performs vertical transformation of data to identify for each item the set of transactions in which it was purchased: tidset (Transaction ID Sets)

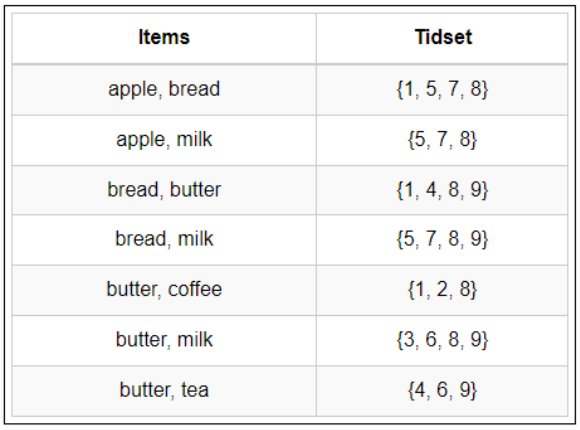


Items	Tidset
apple	{1, 5, 7, 8, 10}
bread	{1, 4, 5, 7, 8, 9}
butter	{1, 2, 3, 4, 6, 8, 9}
coffee	{1, 2, 8, 11}
milk	{3, 5, 6, 7, 8, 9}
Pepsi	{2, 4}
jam	{1, 8}
tea	{4, 6, 9, 10, 11}

2. Checks whether each individual item crosses the support threshold: Pepsi and Jam are pruned
3. Consider pairs of items: retain the intersecting IDs of each individual set - equivalence relation

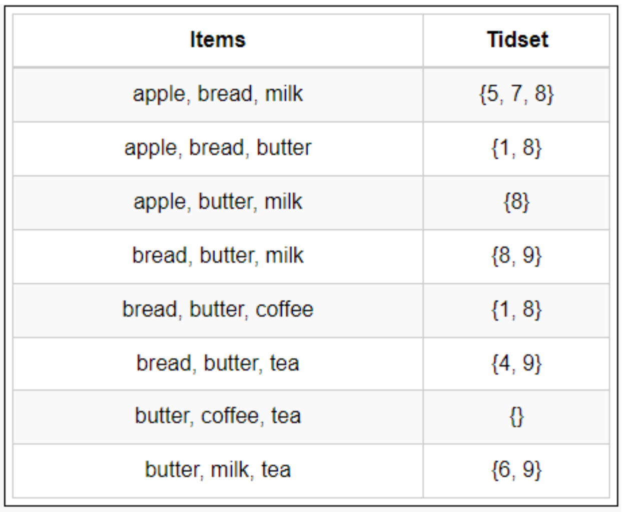


Items	Tidset
apple, bread	{1, 5, 7, 8}
apple, butter	{1, 8}
apple, coffee	{1, 8}
apple, milk	{5, 7, 8}
apple, tea	{10}
bread, butter	{1, 4, 8, 9}
bread, coffee	{1, 8}
bread, milk	{5, 7, 8, 9}
bread, tea	{4, 9}
butter, coffee	{1, 2, 8}
butter, milk	{3, 6, 8, 9}
butter, tea	{4, 6, 9}
coffee, milk	{8}
coffee, tea	{11}
milk, tea	{6, 9}



Items	Tidset
apple, bread	{1, 5, 7, 8}
apple, milk	{5, 7, 8}
bread, butter	{1, 4, 8, 9}
bread, milk	{5, 7, 8, 9}
butter, coffee	{1, 2, 8}
butter, milk	{3, 6, 8, 9}
butter, tea	{4, 6, 9}

4. Consider triplets



Items	Tidset
apple, bread, milk	{5, 7, 8}
apple, bread, butter	{1, 8}
apple, butter, milk	{8}
bread, butter, milk	{8, 9}
bread, butter, coffee	{1, 8}
bread, butter, tea	{4, 9}
butter, coffee, tea	{}
butter, milk, tea	{6, 9}

5. Check threshold; the only triplet that meets the threshold is {apple, bread, milk}
6. The algorithm terminates

- Results: {bread, butter}, {bread, milk}, {butter, milk}, {apple, bread, milk}

ECLAT vs Apriori:

- The ECLAT algorithm is faster and can handle larger datasets more easily than the Apriori algorithm
- The Apriori algorithm works in a horizontal manner over transactions, and gradually builds larger sets of items and checks how frequently these sets appear in transactions
    - The ECLAT algorithm works with a vertical transformation of the transaction data. For each item in the retailer's inventory, its tidset is first identified. Then,

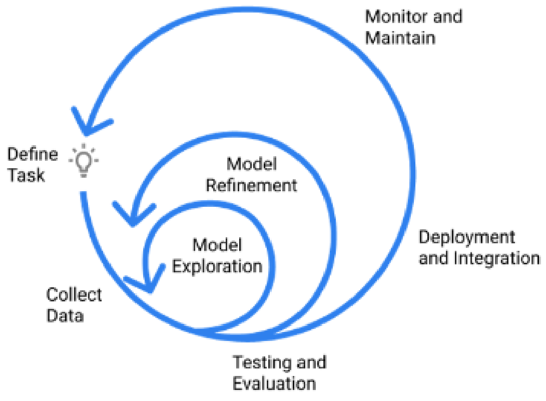
tidsets of larger sets of items are constructed by intersecting the tidsets of smaller sets. This also makes the ECLAT algorithm computationally more efficient than the Apriori algorithm

Metrics:

- Support: how frequently the set of items appears in the data with respect to the total number of transactions in the data
    - E.g the support of {bread, butter} is 4/11 = 36.36%, whereas the support of {bread, butter, milk} = 2/11 = 18.18%
- Confidence: a measure of how often the rule is found to be true; for an association to suggest Y whenever X is purchased, the confidence is the proportion of transactions that contain X which also contain Y.
    - E.g the confidence of the rule: suggest 'butter' whenever 'bread' is purchased is 4/6 = 66.66% because among the six transactions when bread was purchased, in four of them, the customers also purchased butter.

Module 5: Machine Learning Best Practices

Section 1: Machine Learning Best Practices



ML Development Life Cycle:

1. Task Definition Stage
    - Is the problem well-defined?
    - Is there sufficient data to address the question?
        - Is there a need for ML or will a heuristic solution (rule-based vs data-driven) suffice?
2. Data Collection Stage
    - Are you tracking and storing all the relevant data?
    - What are the limitations with the data?
3. Model Exploration and Refinement Stages
    - Has a baseline performance for the solution been established?
    - Is the problem well-studied?

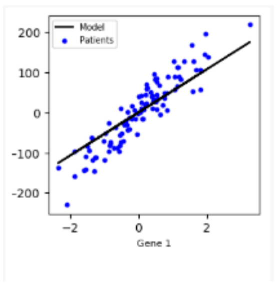
Guidelines for Algorithmic Selection - Task Identification:

- labelled data or a target for the algorithm to optimize for? → supervised learning
    - does the task involve predicting categories? → classification
        - E.g logistic, naive Bayes, support vector machine
    - the task involves predicting a quantitative response? → regression
        - E.g linear regression, regression trees

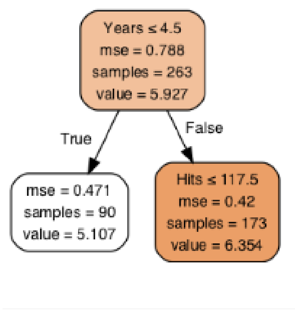
- the data does not have labels, and no clear target for the algorithm to optimize for? → unsupervised learning
    - the task involves identifying groups of observations that are similar to one another? → clustering
        - E.g K-means algorithm, hierarchical clustering, mixture models
    - the task involves identifying items occurring together, or learning interesting connections and links between data points based on co-occurrence? → association learning
        - E.g Apriori, ECLAT

Strengths and Weaknesses of Each Algorithm:

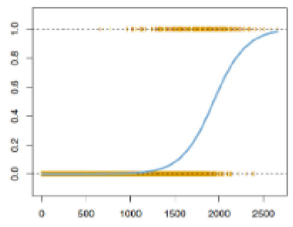
- Linear Regression:
    - Strengths: quick to implement, outputs are easy to comprehend and interpret
    - Weaknesses: often too simplistic as it assumes linear relationship
        - Fix: appropriate transformation of variables e.g log



- Regression Trees:
    - Strengths: simple to understand and interpret
    - Weaknesses: not robust models (small change in data can lead to large change in structure), outputs can be relatively inaccurate
        - Fix: advanced generalization as random forests but they're not easily interpretable



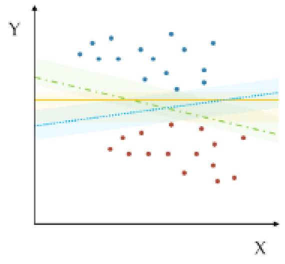
- Logistic Regression:
    - Strengths: nice probabilistic interpretation and can easily be tuned to avoid the problem of overfitting using regularization techniques, handles new data easily
    - Weaknesses: tend to underperform when there are multiple or non-linear decision boundaries, not flexible enough to naturally capture more complex relationships



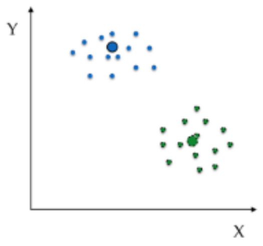
- Naive Bayes:
    - Strengths: simple but powerful algorithm, often works well in practice
    - Weaknesses: when it fails (for data of sufficient complexity), they tend to fail spectacularly

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

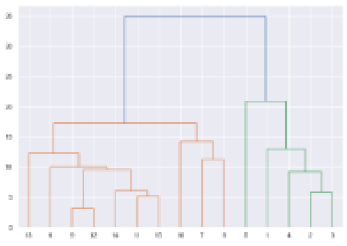
- Support Vector Machines:
    - Strengths: can easily handle multi-class problems and also work well on data of high dimension (ie data with several features), can handle non-linear classification tasks with an advanced variant known as the kernel method



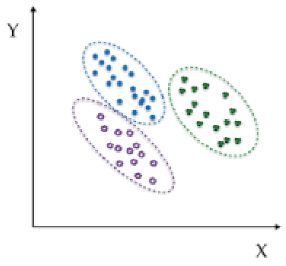
        - Weaknesses: do not perform well with very large or noisy data sets because the required training time is too high
- K-Means Algorithm:
    - Strengths: relatively simple to implement, scales well to large datasets, converges easily, can accommodate new data
    - Weaknesses: requires pre-specification of the number of clusters in the dataset, is very sensitive to outliers



- Hierarchical Clustering:
    - Strengths: simple as K-Means but with no pre-specification, generates a series of clusters in increasing levels of granularity which provides richer information on the structural features in the data
    - Weaknesses: computationally expensive, with large datasets the dendograms grow unwieldy because at the bottom-most level making it harder to interpret and more error-prone in the absence of expertise



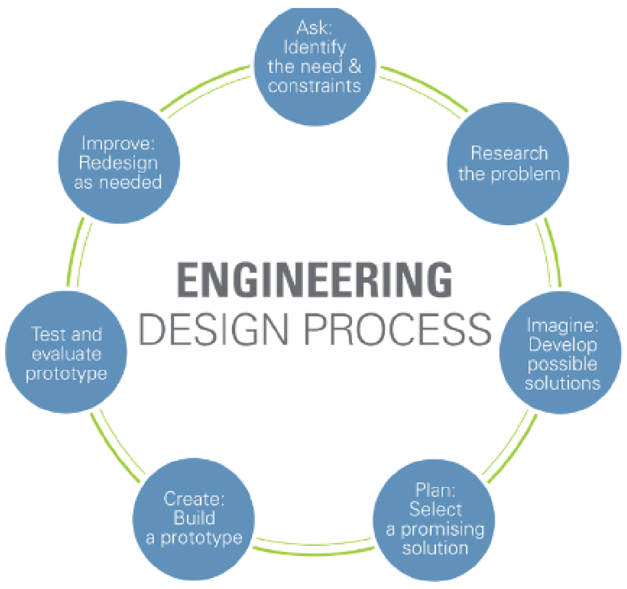
- Mixture Models:
    - Strengths: allow for soft clustering, work better than K-Means when the clusters are more elliptical than circular
    - Weaknesses: requires the pre-specification of the number of clusters present in the data, computationally expensive in situations with large data



- Association Algorithms:
    - Apriori:
        - Strengths: easy to implement and intuitive
        - Weaknesses: can be computationally expensive since it needs to repeatedly scan the entire database
    - ECLAT:
        - Strengths: more efficient than Apriori
        - Weaknesses: implementation is more challenging

Section 2: Poorly Defined Problems Under the Best ML Type

Iterative Engineering Design Process:


ENGINEERING
DESIGN PROCESS
Ask: Identify the need & constraints
Research the problem
Imagine: Develop possible solutions
Plan: Select a promising solution
Create: Build a prototype
Test and evaluate prototype
Improve: Redesign as needed

The Iterative Machine Learning Project Plan:

1. Understanding the Problem
2. Shaping Client Requirements
3. Asking Specific Questions

Section 3: Exploratory Data Analysis on a Real-World Dataset

Exploratory Data Analysis (EDA): an approach that is meant to inspect the datasets to uncover underlying patterns, detect outliers and anomalies, extract important features

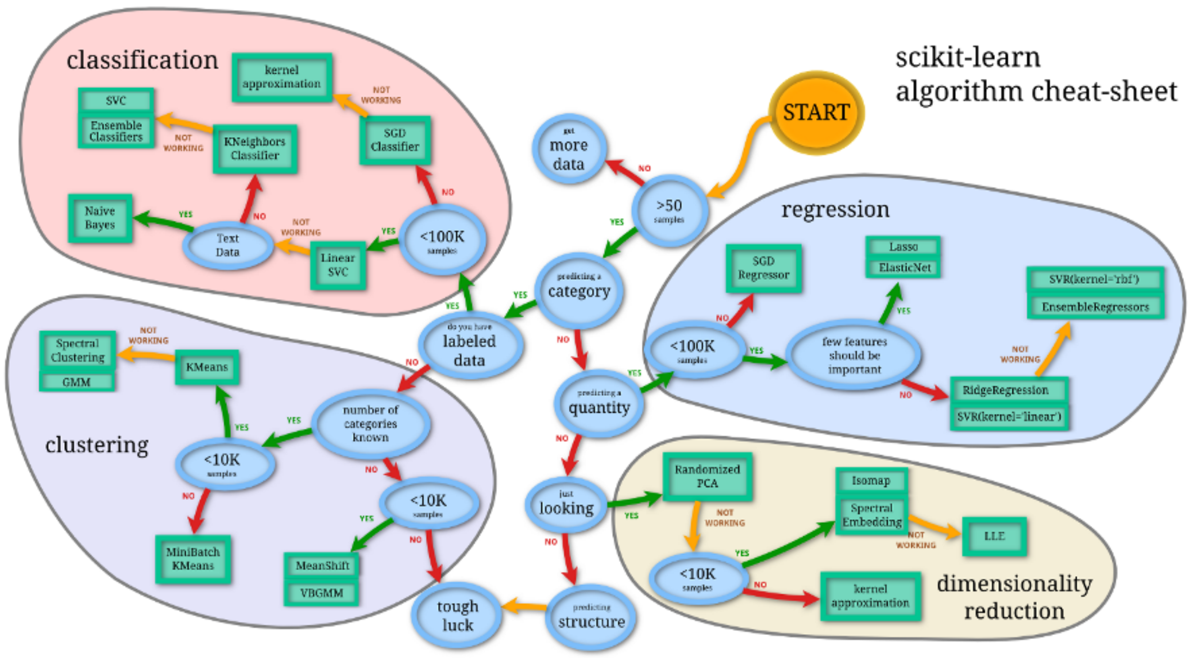
- includes data visualization (histograms, scatter plots, etc), dimensionality reduction (reducing the number of features under consideration)

Dimensionality Reduction:

- Manual Dimensionality Reduction Techniques:
    - Simple way is to remove column that is highly correlated with another column, or remove features that have very small variance across observations
- Principal Component Analysis (PCA):
    - Popular for deriving a low-dimensional set of features from a large set of variables
    - transform the existing variables to a new set of variables, the principal components, which retain the variation present in the existing dataset to a large extent
    - principal components are ordered such that the retention of variation present in the original variables decreases down the order
        - first principal component retains maximum variation present in the data
- T-distributed Stochastic Neighbour Embedding (t-SNE):

- Is fairly slow when there is a large number of features so it is a standard approach is to employ another dimension reduction technique prior to employing the t-SNE

Section 4: Additional Topics Relating to ML Applications


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
MeanShift
VBGMM
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
<100K samples
few features should be important
RidgeRegression
SVR(kernel='linear')
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
<10K samples
kernel approximation
YES
NO
NOT WORKING