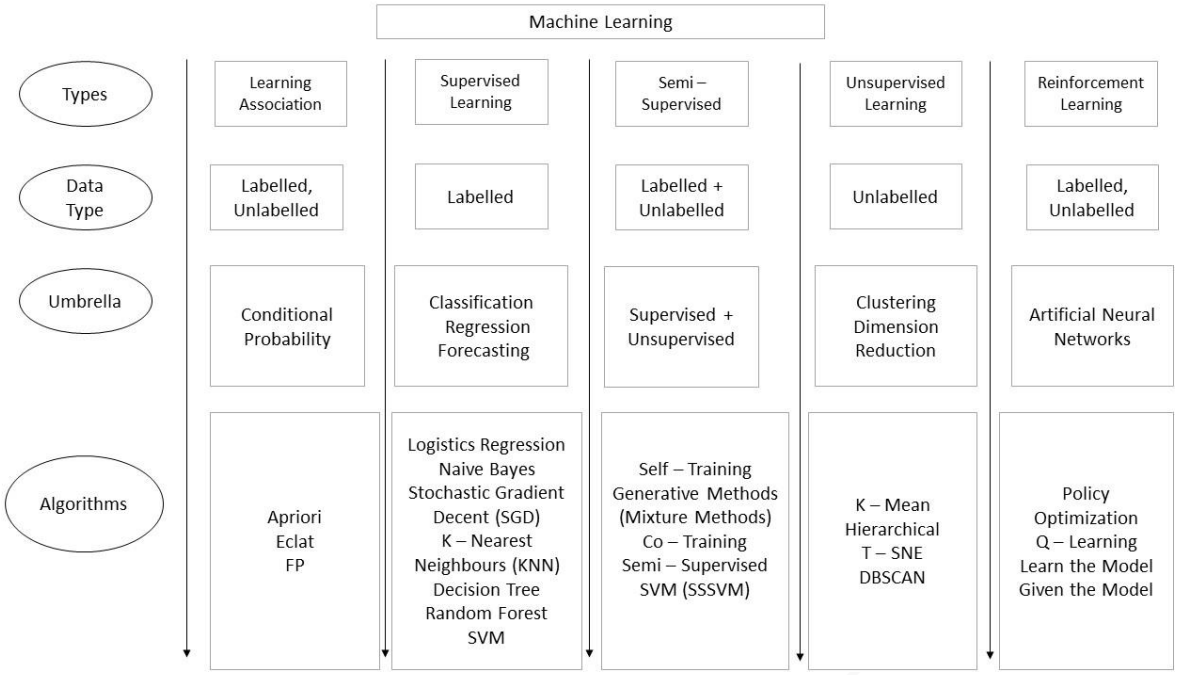
1. What is Machine Learning: -

1. Everything in today's world going online and everyone is sharing data's everywhere. We all are the producers and consumers of the data. Ex. Supermarkets are producing goods and millions of customers are purchasing. They want to know about the customer prediction. Everybody does not purchase everything random.
2. There are certain patterns in the data. To solve a problem on a computer, we need Algorithm. It is a sequence of instructions that should be carried out to transform input to output. Ex. One can device an algorithm for sorting the input is a set of number and output is their ordered list. For the same task there may be various algorithms and we ma be interest in finding the best one ~ **requiring least number of instructions or memory or both.**
3. For some task we are not having algorithm such as "Customer Behaviour", "Spam emails from legitimate one's".
4. Application of machine learning methods to large databases is called **data mining.** Large volume of data is processed to construct a simple model with valuable use.
5. Machine learning is not just a database problem it is also a part of artificial intelligence. To be intelligent aa system that in a changing environment should have an ability to learn.
6. Pattern recognition: every face is symmetrical based upon the persons face it is able to recognize.
7. Machine learning is the theory of statistics in building a mathematical model, because the core task is making **inferences from a sample.**

2. Types of machine learning algorithms.



3. Learning Associations

Learn the conditional probability of the form p(y|x), were y is the product we like condition on x. Ex. In order to distinguish customer p (y|x, D) were D is the customer attributes i.e., gender, age, marital status etc.

An association rule is a pattern that states when an event occurs, another event occurs with certain probability. It is **if/then statements** that helps discover relationships between unrelated data in a data repository i.e., to find the relationships between the objects which are frequently used together. It finds all set of items that have a support greater than the minimum support. Then using a larger itemset to generate the desired rules that have confidence greater than the minimum confidence. A typically and widely used examples of association rules application is market basket analysis. It needs to satisfy the **user specified minimum support and a user specified minimum confidence at the same time.**

Ex. If the customer buys milk, he may also buy cereal. If a customer buys a tablet (iPad, etc) he may also buy a cover (case).

Rule: x => y

1. Support = frequency (x, y) / N

2. Confidence = frequency (x, y) / frequency (x)

3. Lift = support / support (x). support (y)

4. Algorithms of Association Rules

1. **AIS**
2. **SETM**
3. **Apriori**
4. **AprioriTid**
5. **Apriori Hybrid**
6. **FP – Growth**
7. **Eclat**

4.1. AIS

It is the first algorithm proposed for learning associations. In this algorithm only one item consequent association rules are generated which means that the consequent of the rules **contains one item,** for example: rule like x ∩ y => z but not the rules like x => y ∩ z.

Steps:

1. The databases were scanned many times to get the frequent item set. To make this algorithm more efficient **estimation method was introduced.**

2. The support count of each individual item was accumulated during the first pass over the databases based on the minimal support count items were supposed to count less than its minimum values gets eliminated from the list of items.
3. Candidate 2 – item sets are generated by extending frequent 1 – item sets with other items in the transaction. During these operations, support count of those 2 – item sets are accumulated and checked against the support threshold.
4. Similarity the candidate (k+1) – items were generated by extending frequent k – item sets with items on the same transaction.
5. The candidate item sets generation and frequent item sets generation process until any one of them becomes empty.

Disadvantage

Too many candidate items set, so it requires more space and too many passes over the whole databases.

4.2. SETM

In SETM candidate item sets are generated on the fly as the databases is scanned but counted at the end of the pass. It follows the same procedure as AIS algorithm. But the transaction identifier (TID) of generating transaction is saved with candidate itemset in a sequential structure.

Disadvantage

SETM has a same disadvantage as AIS algorithm. Another disadvantage is that for each candidate itemset there are as many entries as it supports value.

4.3. APRIORI

It is used for frequent item set mining and association rule learning. The algorithm uses a **k – itemset** are used to explore (k+1) – itemset to mine frequent itemset from transaction databases for Boolean association rules.

In this algorithm frequent subsets are extended one item at a time and this step is known as **candidate generation process.** Then group of candidates are tested against the data. To count candidate item sets efficiently, **it uses breadth – first search method and a hash tree structure.**

It identifies the frequent individual items in the databases and extends them to larger and larger item sets as long as those item sets appeared sufficiently often in the databases.

It determines frequent item sets that can be used to determine association rules which highlight general trends in the databases.

Steps in APRIORI generation

1. CI_k = Candidate itemset having size k
2. FI_k = frequent itemset having size k
3. FI_1 = {frequent items};
4. FOR (k = 1; $FI_{k!}$ = null; k++) do begin
5. CI_{k+1} = candidates generated from $FI_{k;}$
6. FOR each transaction t in database D do
7. Increment the count value of all candidates in
8. CI_{k+1} that contained in t
9. FI_{k+1} = candidates in CI_{k+1} with min _ support
10. END
11. RETURN $FI_{k:}$

Disadvantages of APRIORI

The complex candidate generation process which uses most if the time, space, memory. It requires multiple scans of the databases.

4.4. APRIORITID

In this algorithm database is not used for counting the support of candidate item sets after the first pass. The process of candidate generation is same as APRIORI algorithm. Another set 'c' is generated of which each member has the TID of each transaction and large item sets present in this transaction. IT is used to count the support of each candidate itemset.

Advantage over APRIORI

The later passes the performance if APRIORITID is better than APRIORI.

4.5. APRIORI HYBRID

As APRIORI does better than APRIORITID in the early passes and APRIORITID does better than APRIORI in later passes. A new algorithm is designed on the combination the two to get the better performance in both the passes.

4.6. FP – Growth

To break the two drawbacks of APRIORI – algorithm, FP – growth algorithm is used it requires to construct FP – Tree for that it requires two passes.

Strategy: It uses divide and conquer strategy

1. It first computes a list of frequent items sorted by frequency in descending order (F-List) during its first databases scan
2. In the second scan databases is compressed into FP- tree recursively.

The frequent item sets are generated with only two passes over the databases and without any candidate generation process.

There are two sub process of FP generation

1. Construction of FP – tree
2. Generation of frequent patterns from the FP -tree

4.7. ECLAT

It stands for Equivalence Class Clustering and Bottom – up Lattice Traversal. It is most efficient and scalable version of APRIORI. It works in vertical sense the depth first search. The vertical approach makes it faster than APRIORI.

5. Conclusion

From the above 7 algorithms of Learning Associations the widely used one's are APRIORI, FP – Tree algorithm.

6. Grocery Dataset – A Market Basket Analysis

6.1. APRIORI Approach

Steps

1. **Importing and installing libraries**

```
import numpy as np # linear algebra
import pandas as pd # for pre - processing
import matplotlib.pyplot as plt # for Data - Visualization
from scipy.special import comb # The number of combinations of N things taken K at a time; "N choose K"
from itertools import combinations, permutations # to form a "iterator algebra"
import pyfpgrowth
import url
```

```
#! pip install apyori # Apriori algorith to find the Associations of the Grocery Data
#! pip install pyfpgrowth
#! pip install url
```

Import the necessary libraries and install apyori for association rule and pyfpgrowth.

2. Load File

Load the file. The dataset is taken from Kaggle – grocery market basket analysis.

	Item(s)	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	...	Item 23
0	4	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	...	NaN
1	3	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	1	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
3	4	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	...	NaN
4	4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	...	NaN

5 rows × 33 columns

3. APRIORI – Association Rule

```
def apyori(df, minimum_support=0.1, confidence=0.22):
    df_values = df.values.astype(str)
    index, counts = np.unique(df_values,return_counts=True)
    df_item = pd.DataFrame(zip(index, counts), columns = ['product', 'frequency'])
    df_item.drop(df_item[(df_item['product'] == 'nan' )|(df_item['product'] == 'None' )].index, inplace=True)
    df_item.sort_values(by='frequency', ascending=False, inplace=True)
    df_item.reset_index(drop=True, inplace=True)
    df_item_frequent = df_item[df_item['frequency']>= minimum_support*len(df)]
    df_itemset_frequency = pd.DataFrame(columns=['itemset', 'frequency'])
    for i in range(1, len(df_item_frequent)+1):
        comb = list(combinations(df_item_frequent['product'].values, i) )
        for w in comb:
            count = 0
            for instance in df_values:
                if all(elem in instance  for elem in w):
                    count = count +1
            if count >= (minimum_support*len(df)/2):#tirar /2
                df_itemset_frequency = df_itemset_frequency.append({'itemset':w, 'frequency':count}, ignore_index=True)
    df_itemset_frequency.sort_values(by='frequency', inplace=True, ascending=False)
    reliability = pd.DataFrame(columns=['rule', 'frequency', 'reliability'])
    for w in df_itemset_frequency['itemset'].values:
        w_p = list(permutations(w,len(w)))
        for j in w_p:
            #print (len(j[0]))
```

```
            p_uni = []
            for i in range(len(j)):

                count = 0
                for instance in df_values:
                    if all(elem in instance  for elem in j[i:]):
                        count = count +1
                p_uni.append(count/len(df))

            if len(j) != 1:
                a = p_uni[-2]/p_uni[-1]

                for i in range(len(p_uni)-2):
                    a = p_uni[-i-3]/a
                j = list(j)
                j.reverse()
                reliability = reliability.append({'rule':j, 'frequency':p_uni[0], 'reliability':a}, ignore_index=True)
            else:
                reliability = reliability.append({'rule':j, 'frequency':p_uni[0], 'reliability':p_uni[0]}, ignore_index=True)
    reliability.sort_values(by='frequency', ascending=False)
    return reliability[reliability['reliability']>=confidence]
apyori(df.drop(columns='Item(s)'))
```

	rule	frequency	reliability
0	(whole milk,)	0.255516	0.255516
8	[other vegetables, whole milk]	0.074835	0.386758
9	[whole milk, other vegetables]	0.074835	0.292877
10	[rolls/buns, whole milk]	0.056634	0.307905
11	[whole milk, rolls/buns]	0.056634	0.221647
12	[yogurt, whole milk]	0.056024	0.401603

From the analysis it is clearly states that p(y | x , D) is p(other items | whole milk, customer TID) i.e., other items such as other vegetables, rolls/buns, yogurt are association with whole milk. Which has been purchased by the customers having transaction ID 0, 8 to 12. Customer who are purchasing vegetables they are preferring milk, because it is the necessary item to be considered in every household which go along with everything.

6.2. FP – Growth Tree Approach

Steps

1. **Dataset**

```
dataset = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs'],]
```

2. **FP – Growth Association Rules**

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogurt
0	False	False	False	True	False	True	True	True	True	False	True
1	False	False	True	True	False	True	False	True	True	False	True
2	True	False	False	True	False	True	True	False	False	False	False
3	False	True	False	False	False	True	True	False	False	True	True
4	False	True	False	True	True	True	False	False	True	False	False

3. Converting to arrays

```
In [137]: te_ary.astype("int")
```

```
Out[137]: array([[0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1],
                 [0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1],
                 [1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1],
                 [0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0]])
```

```
In [138]: from mlxtend.frequent_patterns import fpgrowth

          fpgrowth(df, min_support=0.6)
```

Out[138]:

	support	itemsets
0	1.0	(5)
1	0.8	(3)
2	0.6	(10)
3	0.6	(8)
4	0.6	(6)
5	0.8	(3, 5)
6	0.6	(10, 5)
7	0.6	(8, 3)
8	0.6	(8, 5)
9	0.6	(8, 3, 5)
10	0.6	(5, 6)

```
In [139]: fpgrowth(df, min_support=0.6, use_colnames=True)
```

Out[139]:

	support	itemsets
0	1.0	(Kidney Beans)
1	0.8	(Eggs)
2	0.6	(Yogurt)
3	0.6	(Onion)
4	0.6	(Milk)
5	0.8	(Kidney Beans, Eggs)
6	0.6	(Kidney Beans, Yogurt)
7	0.6	(Onion, Eggs)
8	0.6	(Onion, Kidney Beans)
9	0.6	(Onion, Kidney Beans, Eggs)
10	0.6	(Milk, Kidney Beans)

The FP – Growth Tree is generated for the given transactions and datasets. In which mostly egg is associated with other items.

References

Alpaydin, E., 2014. Adaptive Computation and Machine Learning. In: T. Dietterich, ed. *Introduction to Machine Learning.* London: The MIT Press, pp. 1 - 20.