Logistic Regression is used for classification tasks. The two classes are usually 0 and 1 (false and true) but there are more extensions if there are more than 2 classes. The regression model fits the data into a logistic (sigmoidal curve) instead of fitting it into a straight line. The logistic regression assumes that the relationship between the feature and the class variable is non-linear.
Ridge Regression is a regularized version of Linear Regression and is also known as the Tikhonov Regularization. It uses the same equation as the Linear Regression Model to make predictions however the regression coefficients are chosen in such a way that not only does it fit the training data well, but it also makes sure that the magnitude of coefficients is as small as possible. Small values of coefficients means that each feature will have little effect on the outcome and the regression line will have a small slope. This makes it in such a way that the linear regression model is less complex and more restricted thus making it less likely to overfit. It uses the so called L2 regularization method. The main goal of this algorithm is to have high accuracy on the training data and be a low complexity model. The parameter α controls the trade-off between the performance on the training set and the model complexity.
Lasso Regression is another type of regularized version of the standard Linear Regression Model. LASSO stands for Least Absolute Shrinkage and Selection Operator Regression. As the Ridge regression, this regression model adds a regularization term to the cost function, but it uses the L1 norm of the regression coefficient vector unlike the Ridge regression which uses L2 norm. The main goal of this regression model is to get high accuracy on training data while have a low complexity model. The consequence of using the L1 norm is that some regression coefficients can become exactly 0. That means that certain features would be completely ignored by the model, which makes it kind of like an automatic feature selection model. As the features would be less, the model itself will be less complex.
Naïve Bayes Algorithm uses the Bayes theorem to solve classification tasks. It takes into consideration 2 assumptions, these are: The attributes are conditionally independent of each other for each class value, all attributes are equally important. Considering that these assumptions are almost never correct, the algorithm is called Naïve Bayes theorem. These assumptions, however, leads to a simple and easy to implement algorithm which works well in practice. Given below are the steps to use Bayes Theorem: Calculate P(H | E) for each H (class). ~ P(yes | E) and P(no | E). Compare them and assign E to the class with the highest probability. For P(H | E) we need to calculate P€, P(H) and P(E | H) from the given training data.
Evaluating Machine Learning Algorithms: Holdout Method, Holdout Method with parameter tuning, Holdout Method with stratification, Repeated Holdout Method, Cross Validation, Cross Validation with stratification, Leave-one-out Validation., Cross validation for parameter tuning.
Entropy measures the homogeneity of a set with respect to the class. The smaller the entropy, the greater the purity of the data set. **Information Gain** measures the reduction in entropy caused by using an attribute to partition the set of training examples. The best attribute is the one with the highest information gain (biggest reduction in entropy). Information gain is the difference between 2 values of entropies.
Ensemble Methods: Bagging (also called as bootstrap aggregation. Given a dataset 'D' with 'n' examples. Bootstrap examples 'D-'contains 'n' examples as well chosen randomly from 'D' where some examples will appear more than once or some not at all.), **Ada Boosting** (This algorithm uses a weighted training set meaning that each training example has an associated weight to it which is greater or equal to 0. The higher the weight, the more difficult the example was to classify by the previous classifier and hence have a higher chance to be selected for the training set of the next classifier.).
Kernel trick is a method for computing the dot product of a pair of vectors in the new space without first computing the transformation of each vector from the original to the new space. First, we compute the dot product of the original features and use it in a 'kernel function' to determine the dot product of the transformed features. The kernel function specifies the relationship between the dot products in the original and transformed space.
Principal Component Analysis is the most popular dimensionality reduction method. It is often called a feature projection method. The main idea of this method is to find a new set of dimensions and project data into it. The dimensionality of the new space is smaller than the original space and the new axes captures the essence of the data. The resulting dataset (projection) can be used as an input to train a ML algorithm. In short, the PCA method deals with the construction of new features which are smaller than the number of original features.
Back propagation Algorithm is used to train a multi-layer perceptron neural network. The main idea of this algorithm is that for each training algorithm, an input is propagated through the network and an output is calculated. This is compared with the target output and error is calculated. The weights are then updated to reduce the error until the error for overall examples is less than the threshold value. The weights are updated backwards, from the output to the input neurons by propagating the weight change to minimize the error and hence is called a back propagation algorithm.
Vanishing Gradient Problem: when there are more layers in the network, the value of the product of derivative decreases until at some point the partial derivative of the loss function approaches a value close to zero, and the partial derivative vanishes. We call this the vanishing gradient problem.
Dropout: This is a method to prevent overfitting. The main idea is to avoid learning spurious features at the hidden nodes through intuition. Relevant features are more resilient to the removal of neurons, and they perform well for different combinations of neurons. While spurious features depend on certain neurons. Dropout forces the Neural Network to be less dependent on certain neurons to collect more evidence from the other neurons and to be more robust to noise. During training, at each iteration of the backpropagation, random neurons are selected in each layer and their values are set to 0. This results in a thinned sub-network of a smaller size. During training, the weights and biases are updated using back propagation algorithm and then new weight values are added to the original network. During testing, no neurons are dropped out and the network is scaled down based on the dropout rate.
SoftMax: The neural network outputs are processed to turn them into probabilities. The main motivation for this is to interpret the outputs as probabilities that sum up to 1.
GMM stands for **Gaussian Mixture Model** clustering which is a probabilistic clustering technique. It assumes that the data is generated by a mixture of normal distributions. In this algorithm we assume that the data is generated by a mixture of k Gaussian distributions and each distribution has 2 parameters, mean and standard deviation. One distribution corresponds to one cluster and starting from the initial values, the parameters are estimated iteratively. After each estimation, probability for each example is computed and then parameters are recomputed until they don't change. K-Means uses hard assignment whereas GMM uses probabilistic assignment. GMM can be seen as a generalization of K-Means. It is more flexible and it allows for elliptical clusters rather than circular for probabilistic assignment to each cluster rather than crisp.
Agglomerative clustering (Hierarchical) also called as the bottom-up clustering wherein the clusters merges iteratively. The algorithm starts with each item in its own cluster and iteratively merge until all items belong in one big cluster. The key operation in this type of clustering algorithm is computing the distance between 2 clusters. There are different versions of how the clusters are merged at each step.
Divisive clustering (Hierarchical) also called the top-down clustering wherein the clusters split into two until all items are in their own clusters. This can be implemented based on computing the minimum spanning tree.
DBSCAN stands for Density Based Spatial Clustering of Applications with Noise. The clusters created are regions of high density, separated from one another by regions with low density. In contrast to K-Means, DBSCAN can find clusters with arbitrary and complex shape. The main idea of the algorithm was that in a cluster the density of the points around it should be higher than the threshold. We need to define the density and the neighbourhood of a point. The neighbourhood of a point is the area within a radius of the point. Density of a point is the number of points in the neighbourhood of a point including that point. And Density threshold MinPts is the minimum number of points in the EPS neighbourhood of a point. The density of a point depends on the neighbourhood Eps (Epsilon). If the epsilon is too big, the density will be less and if the Eps is too small, all the points will have density as 1.
Grid based clustering is a density-based clustering algorithm which utilizes grids. The main idea is to break the data points into grid cells and then form clusters from the cells that are dense enough. The values of each attribute are split into intervals which are called grid cells. The common approach to do this is by splitting the values in equal width intervals. This leads the grid cells to have the same volume and the number of points will give the density of the cell. There are more sophisticated approaches as well such as breaking the values of an attribute in intervals such that there are equal data points in each grid, using clustering to determine intervals, break the initial values into large number of equal width intervals and then combine intervals with similar densities.
Clustering algorithms that use a few attributes from a pool of attributes is called **subspace clustering algorithms. CLIQUE** is called as the Clustering in Quest algorithm. It is a dimension growth-based clustering algorithm. The main idea of the algorithm is to find subspaces of high dimensionality where high density clusters exist. At each subspace, the data is partitioned into rectangular cells and the dense cells are identified based on a threshold. To efficiently find the dense cells at k-dim subspace, it uses the Apriori property from association rule mining and only considers the dense cells from the previous lower dimensional subspace (k-1). Apriori principle states that if a set of points forms a density-based cluster in k dimensional space, then the same set of points are part of the density-based cluster in all possible subsets of those 3 dimensions.
Markov Model was proposed by Andrey Markov. A Markov chain (Markov process) is a model describing a sequence of transitions from one state to another, in which the probability of each state depends on the previous state. The main assumption of Markov model is that the probability of any concrete state depends only on the previous state and not the older history.
Hidden Markov Models: Markov Models are useful when probability of directly observable states is to be computed. Hidden Markov Models are used when are not observed directly or are hidden but can develop a judgement based on indirect observations. A Hidden Markov model is a probabilistic model that allow us to predict a sequence of hidden events from a set of observed events.
Markov's Decision Process is a discrete-time stochastic control process. It provides a mathematical framework for modelling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

Question 5. 1R

Given the training data in the table below where ***credit history, debt, deposit*** and ***income*** are attributes and ***risk*** is the class, predict the class of the following new example using the 1R algorithm: *credit history=unknown, debt=low, deposit=none, income=average.* If needed, settle ties by random selection. Show your calculations.

credit history	debt	deposit	income	risk
bad	high	none	low	high
unknown	high	none	average	high
unknown	low	none	average	moderate
unknown	low	none	low	high
unknown	low	none	high	low
unknown	low	adequate	high	low
bad	low	none	low	high
bad	low	adequate	high	moderate
good	low	none	high	low
good	high	adequate	high	low
good	high	none	low	high
good	high	none	average	moderate
good	high	none	high	low
bad	high	none	average	high

Solution:
1. Attribute ***credit history***
bad: 0 low, 1 moderate, 3 high => risk=high, errors: 1/4
unknown: 2 low, 1 moderate, 2 high => risk=low, errors: 3/5
good: 3 low, 1 moderate, 1 high => risk=low, errors: 2/5
total errors: 6/14

2. Attribute ***debt***
high: 2 low, 1 moderate, 4 high => risk=high, errors: 3/7
low: 3 low, 2 moderate, 2 high => risk=low, errors: 4/7
total errors: 7/14

3. Attribute ***deposit***
none: 3 low, 2 moderate, 6 high => risk=high, errors: 5/11
adequate: 2 low, 1 moderate, 0 high => risk=low, errors: 1/3
total errors: 6/14

4. Attribute ***income***

Question 6. Perceptron

Given is the following training set:

	input	output
ex. 1:	1 0 0	1
ex. 2:	0 1 1	0
ex. 3:	1 1 0	1
ex. 4:	1 1 1	0
ex. 5:	0 0 1	0

a) Train a perceptron **with a bias** on this training set. Assume that all initial weights (including the bias of the neuron) are 0. Show the set of weights (including the bias) at the end of the first epoch. Apply the examples in the given order.

Recall that the perceptron uses a step function defined as:
step(n) = 1, if n >= 0
= 0, otherwise.

Solution:
a) starting point: w=[0 0 0], b=0

applying ex.1: p_1= [1 0 0], t_1=1
a=step([0 0 0][1 0 0]+0)=step(0)=1
e=t_1-a=1-1=0
w^{new}=[0 0 0]+0[1 0 0] =[0 0 0]
b^{new}= 0+0=0
i.e. no change in w and b as e=0

applying ex.2: p_2= [0 1 1], t_2=0
a=step([0 0 0][0 1 1]+0)=step(0)=1
e=t_2-a=0-1=-1
w^{new}=[0 0 0]+(-1)[0 1 1] =[0 -1 -1
b^{new}= 0+(-1)=-1
i.e. w an b have been updated
Similarly:
After applying ex. 3: w^{new}=[1 0 −1], b^{new}=0
After applying ex. 4: w^{new} =[0 -1 -2], b^{new} =-1

After applying ex. 5: no change
end of epoch 1
Weight vector and bias at the end of epoch 1: w =[0 -1 -2], b =-1

Mean Absolute Error (MAE)

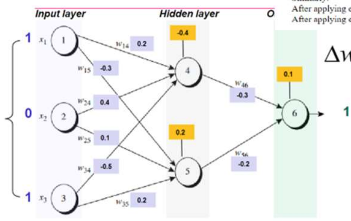
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$



$$\Delta w_{pq} = \eta \cdot \delta_q \cdot o_p \quad \text{- weight change}$$

$$\delta_q = (t_q - o_q)o_q(1 - o_q)$$

input vector x			*initial weights w*								*initial biases θ*		
x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

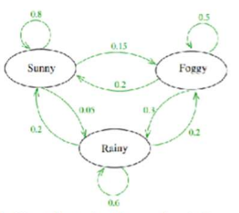
Input to neuron 4: z_4=1*0.2+0*0.4+1*(-0.5)-0.4=-0.7, output of neuron 4: o_4=1/(1+$e^{0.7}$)=0.332
Input to neuron 5: z_5=1*(-0.3)+0*0.1+1*0.2+0.2=0.1, output of neuron 5: o_5=1/(1+$e^{-0.1}$)=0.525 *NN output*
Input to neuron 6: z_6=0.332*(-0.3)+0.525(-0.2) +0.1=-0.105, output of neuron 6: o_6=1/(1+$e^{0.105}$)= **0.474**

δ_6= (t_6-o_6)*o_6·(1-o_6)=(1-0.474)*0.474*(1-0.474)=0.1131
Δw_{46}=η* δ_6*o_4= 0.9*0.1311*0.332=0.039, w_{46}new=w_{46}old+ Δw_{46}=-0.3+0.039=-0.261
Δw_{56}=η* δ_6*o_5= 0.9*0.1311*0.525=0.0619, w_{56}new=w_{56}old+ Δw_{56}=-0.2+0.0619=-0.138
$\theta_6$new= $\theta_6$old + $\Delta\theta_6$ = $\theta_6$old +η* δ_6*1=0.1+0.9*0.1311*1=0.218

δ_4=o_4*(1- o_4)*w_{46}* δ_6=0.332*(1-0.332)*(-0.3)*0.1311=-0.0087
Δw_{14}=η* δ_4*o_1= 0.9*(-0.0087) *1=-0.0079, w_{14}new=w_{14}old+ Δw_{14}=0.2-0.0079=0.1921
Δw_{24}=η* δ_4*o_2= 0.9* *(-0.0087) *0=0, w_{24}new=w_{24}old+ Δw_{24}=0.4+0=0.4
Δw_{34}=η* δ_4*o_3= 0.9* *(-0.0087) *1=-0.0079, w_{34}new=w_{34}old+ Δw_{34}=-0.5-0.0079=-0.5079
$\theta_4$new= $\theta_4$old + $\Delta\theta_4$= $\theta_4$old +η* δ_4*1=-0.4+0.9*(-0.0087)*1=-0.4078

Question 8. Markov models

Given is the following Markov model for the weather in Sydney:



a) Given that today the weather is *Sunny*, what is the probability that it will be *Sunny* tomorrow and *Rainy* the day after tomorrow, i.e. what is the probability *P(*π_3 *= Rainy,* π_2 *= Sunny|* π_1 *= Sunny)?*

Hint: P(A,B|C) = P(A|B,C) P (B|C)

b) If the weather yesterday was *Rainy*, and today is *Foggy*, what is the probability that tomorrow it will be *Sunny*?

For both questions, briefly show your calculations.

Solution:
a)
$P(\pi_3 = Rainy,\ \pi_2 = Sunny|\ \pi_1 = Sunny) =$
$= P(\pi_3 = Rainy\ |\ \pi_2 = Sunny,\ \pi_1 = Sunny) * P\ (\pi_2 = Sunny\ |\ \pi_1 = Sunny)=$ (rule from hint)
$= P(\pi_3 = Rainy\ |\ \pi_2 = Sunny) * P\ (\pi_2 = Sunny\ |\ q\pi_1 = Sunny)=$ (Markov assumption)
$= 0.05 * 0.8 = 0.04$

b)
$P(\pi_3 = Sunny\ |\pi_2 = Foggy,\ \pi_1 = Rainy) =$
$= P(\pi_3 = Sunny\ |\ \pi_2 = Foggy)=$ (Markov assumption)
$= 0.2$

Exercise 1. ***Nearest Neighbor (to do in class)***
The dataset below consists of 4 examples described with 3 numeric features (a1, a2 and a3); the class has 2 values: yes and no.

What will be the prediction of 1-Nearest Neighbor (1-NN) and 3-Nearest Neighbor (3-NN) with Euclidian distance for the following new example: a1=2, a2=4, a3=2?

Assume that all attributes are measured on the same scale – no need for normalization.

	a1	a2	a3	class
1	1	3	1	yes
2	3	5	2	yes
3	3	2	2	no
4	5	2	3	no

Exercise adapted from M. Kubat, Introduction to Machine Learning, Springer, 2017

Solution:

D(new, ex1) = sqrt[$(2-1)^2+(4-3)^2+(2-1)^2$]=sqrt(3) yes
D(new, ex2) = sqrt[$(2-3)^2+(4-5)^2+(2-2)^2$]=sqrt(2) yes
D(new, ex3) = sqrt[$(2-3)^2+(4-2)^2+(2-2)^2$]=sqrt(5) no
D(new, ex4) = sqrt[$(2-5)^2+(4-2)^2+(2-3)^2$]=sqrt(14) no

The closest nearest neighbor is ex. 2, hence 1-NN predicts class=yes
The closest 3 nearest neighbors are ex.2 (yes), ex.1 (yes) and ex.3 (no); the majority class is yes. Hence, 3-NN predicts class =yes

Exercise 2. ***Nearest neighbor with nominal features (to do at your own time)***
Consider the *iPhone* dataset given below. There are 4 nominal attributes (age, income, student, and credit_rating) and the class is buys_iPhone with 2 values: yes and no.

What would be the prediction of 1-NN and 3-NN for the following new example:
age<=30, income=medium, student=yes, credit-rating=fair

If there are ties, make random selection.

Tip: As the examples are described with nominal attributes, when calculating the distance use the following rule:

difference=1 between 2 values that are not the same
difference=0 between 2 values that are the same
e.g. D(1, new)=sqrt(0+1+1+0=)=sqrt(2)

	age	income	student	credit rating	buy iPhone
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	[31,40]	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	excellent	no
6	[31,40]	low	yes	excellent	yes
7	<=30	medium	no	fair	no
8	[31,40]	medium	no	excellent	yes
9	>40	medium	no	excellent	no

Dataset adapted from J. Han and M. Kamber, Data Mining, Concepts and Techniques, Morgan Kaufmann.

Solution:

new example: age<=30, income=medium, student=yes, credit-rating=fair
D(1, new) = sqrt(0+1+1+0) = sqrt(2) no
D(2, new) = sqrt(0+1+1+1) = sqrt(3)
D(3, new) = sqrt(1+1+1+0) = sqrt(3)
D(4, new) = sqrt(1+0+1+0) = sqrt(2) yes
D(5, new) = sqrt(1+1+0+1) = sqrt(3)
D(6, new) = sqrt(1+1+0+1) = sqrt(3)
D(7, new) = sqrt(0+0+1+0) = sqrt(1)=1 no
D(8, new) = sqrt(1+0+1+1) = sqrt(3)
D(9, new) = sqrt(1+0+1+1) = sqrt(3)

- 1-NN: ex. 7 (D=1) is the closest neighbor, hence 1-NN predicts buy_iPhone=no
- 3-NN: the 3 closest neighbors are: ex.7 (D=1), ex.1 and ex.4 (D=sqrt(2)); 2 no and 1 yes => the majority class is no. Hence, 3-NN predicts buy_iPhone=no

Exercise 3. ***PRISM (to do at your own time)***
Given the training data in the table below, generate the PRISM rules for class=no. In case of ties, make random selection.

Weather data with nominal attributes:

	outlook	temperature	humidity	windy	play
1.	sunny	hot	high	false	no
2.	sunny	hot	high	true	no
3.	overcast	hot	high	false	yes
4.	rainy	mild	high	false	yes
5.	rainy	cool	normal	false	yes
6.	rainy	cool	normal	true	no
7.	overcast	cool	normal	true	yes
8.	sunny	cool	high	false	no
9.	sunny	mild	normal	false	yes
10.	rainy	cool	normal	false	yes
11.	sunny	mild	normal	true	yes
12.	overcast	mild	high	true	yes
13.	overcast	hot	normal	false	yes
14.	rainy	mild	high	true	no

Solution:

Let's start with generating rules for class `no`:

`if ? than class=no`

10 possible tests with their corresponding accuracy p/t:

outlook=sunny	3/5
outlook=overcast	0/4
outlook=rainy	2/5
temperature=hot	2/4
temperature=cool	2/5
temperature=mild	1/5
humidity=high	4/7
humidity=normal	1/7
windy=true	3/6
windy=false	2/8

Best test: `outlook=sunny`
Rule: `if outlook=sunny then class=no`
Examples covered by the current rule:

	outlook	temperature	humidity	windy	play
1.	sunny	hot	high	false	no
2.	sunny	hot	high	true	no
8.	sunny	cool	high	false	no
9.	sunny	mild	normal	false	yes
11.	sunny	mild	normal	true	yes

Not a perfect rule as it covers also 2 examples from class yes => add other tests to this rule
`if outlook=sunny and ? then class=no`

Possible tests with the corresponding accuracy p/t:

temperature=hot	2/2
temperature=cool	1/1
temperature=mild	0/2
humidity=high	3/3
humidity=normal	0/2
windy=true	1/2
windy=false	2/3

Best test: `humidity=high` (bigger coverage than `temperature=hot` and `temperature=cool`)
Rule: `if outlook=sunny and humidity=high then class=no`, perfect rule as it covers only examples from class `no` => stop adding other tests to this rule; delete the examples covered by the rule; there are still uncovered examples from class no, so generate another rule for class `no`.

	outlook	temperature	humidity	windy	play
3.	overcast	hot	high	false	yes
4.	rainy	mild	high	false	yes
5.	rainy	cool	normal	false	yes
6.	rainy	cool	normal	true	no
7.	overcast	cool	normal	true	yes
9.	sunny	mild	normal	false	yes
10.	rainy	cool	normal	false	yes
11.	sunny	mild	normal	true	yes
12.	overcast	mild	high	true	yes
13.	overcast	hot	normal	false	yes
14.	rainy	mild	high	true	no

`if ? than class=no`
10 possible tests with their corresponding accuracy p/t:

outlook=sunny	0/2
outlook=overcast	0/4
outlook=rainy	2/5
temperature=hot	0/2
temperature=cool	1/4
temperature=mild	1/5
humidity=high	1/4
humidity=normal	1/7
windy=true	2/5
windy=false	0/6

Best test: `outlook=rainy` (draw with `windy=true`, random selection)
Rule: `if outlook=rainy then class=no`
Examples covered by the current rule:

	outlook	temperature	humidity	windy	play
4.	rainy	mild	high	false	yes
5.	rainy	cool	normal	false	yes
6.	rainy	cool	normal	true	no
10.	rainy	cool	normal	false	yes
14.	rainy	mild	high	true	no

Not a perfect rule as it covers also 3 examples from class yes => add other tests to this rule
`if outlook=rainy and ? then class=no`
Possible tests with the corresponding accuracy p/t:

temperature=cool	1/3
temperature=mild	1/2
humidity=high	1/2
humidity=normal	1/3
windy=true	2/2
windy=false	0/3

Best test: `windy=true`
Rule: `if outlook=rainy and windy=true then class=no`, perfect rule as it covers only examples from class `no` => stop adding other tests to this rule; delete the examples covered by the rule; there are no more uncovered examples from class no, so the rules for class `no` are:

```
if outlook=rainy and windy=true then class=no
if outlook=sunny and humidity=high then class=no
```

Repeat the procedure for class `yes`. PRISM will generate the following rules:

```
if outlook=overcast than class=yes
if humidity=normal and windy=false than class=yes
if temperature=mild and humidity=normal than class=yes
if temperature=mild and windy=false than class=yes
```

$$Weighted\ D(A,B) = \sqrt{w_1|a_1-b_1|^2 + w_2|a_2-b_2|^2 + \dots + w_n|a_n-b_n|^2}$$
$$Manhattan\ D(A,B) = |a_1-b_1| + |a_2-b_2| + \dots + |a_n-b_n|$$
$$Euclidean\ D(A,B) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \dots + (a_n-b_n)^2}$$

Exercise 1. ***Naïve Bayes for data with nominal features (to do in class)***
Given is the following dataset where *loan default* is the class. Predict the class of the following new example using Naïve Bayes:
home owner = no, marital status = married, annual income=very high

	home owner	marital status	income	loan default
1	yes	single	very high	yes
2	no	married	high	yes
3	no	single	medium	no
4	yes	married	very high	no
5	yes	divorced	high	yes
6	no	married	low	no
7	yes	divorced	very high	no
8	no	single	high	yes
9	no	married	medium	no
10	no	single	low	yes

Dataset adapted from, Tan, Steinbach, Karpatne and Kumar, Introduction to Data Mining, Pearson, 2019

Solution:
E= home owner = no, marital status = married, annual income=very high
E1 is home owner = no, E2 is marital status = married, E3 is annual income=very high

We need to compute P(yes|E) and P(no|E) and compare them.

$$P(yes|E) = \frac{P(E_1|yes)P(E_2|yes)P(E_3|yes)P(yes)}{P(E)}$$

$$P(no|E) = \frac{P(E_1|no)P(E_2|no)P(E_3|no)P(no)}{P(E)}$$

P(yes)=5/10 P(no)=5/10

P(E1|yes)=P(home owner=no|yes)=3/5
P(E2|yes)=P(marital status=married|yes)=1/5
P(E3|yes)=P(annual income=very high|yes)=1/5

P(E1|no)= P(home owner=no|no)=3/5
P(E2|no)=P(marital status=married|no)=3/5
P(E3|no)=P(annual income=very high|no)=2/5

$$P(yes|E) = \frac{\frac{3}{5}\cdot\frac{1}{5}\cdot\frac{1}{5}\cdot\frac{5}{10}}{P(E)} = \frac{\frac{3}{250}}{P(E)} = \frac{0.012}{P(E)}$$

$$P(no|E) = \frac{\frac{3}{5}\cdot\frac{3}{5}\cdot\frac{2}{5}\cdot\frac{5}{10}}{P(E)} = \frac{\frac{9}{125}}{P(E)} = \frac{0.072}{P(E)}$$

P(no|E)>P(yes|E) => Naïve Bayes predicts **loan default = no** for the new example.

Exercise 2. ***Naïve Bayes for data with numeric features (to do in class)***
The same task as in the previous exercise but now *annual income* is a numeric feature:

	home owner	marital status	income (in K)	loan default
1	yes	single	125	yes
2	no	married	100	yes
3	no	single	70	no
4	yes	married	120	no
5	yes	divorced	95	yes
6	no	married	60	no
7	yes	divorced	220	no
8	no	single	85	yes
9	no	married	75	no
10	no	single	90	yes

Use Naïve Bayes to predict the class of the following new example:

home owner = no, marital status = married, annual income=120

Solution:

1) Calculate the mean μ and standard deviation σ values for the numeric feature *income*:

$$\mu = \frac{\sum_{i=1}^{n} X_i}{n} \qquad \sigma^2 = \frac{\sum_{i=1}^{n}(X_i-\mu)^2}{n-1}$$

where X_i, *i*=1..*n* – the *i*-th measurement, *n*-number of measurements

We need to calculate the mean and standard deviation separately for each class (yes and no) – separate the values of income:

class **yes**	class **no**
income	income
125	70
100	120
95	60
85	220
90	75
μ_income_yes=99	μ_income_no=109
σ_income_yes=15.57	σ_income_no=66.18

2) Calculate P(income=120|yes) and P(income=120|no) using the probability density function for normal distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$f(income=120|yes) = \frac{1}{15.57\sqrt{2\pi}}e^{-\frac{(120-99)^2}{2*15.57^2}} = 0.01032$$

$$f(income=120|no) = \frac{1}{66.18\sqrt{2\pi}}e^{-\frac{(120-109)^2}{2*66.18^2}} = 0.00595$$

3) Calculating the probabilities P(yes|E) and P(no|E) using the Bayes Theorem; we already have the probabilities for the nominal attributes from the previous exercise:

$$P(yes|E) = \frac{\frac{3}{5}\cdot\frac{1}{5}\cdot 0.01032\cdot\frac{5}{10}}{P(E)} = \frac{0.000619}{P(E)}$$

$$P(no|E) = \frac{\frac{3}{5}\cdot\frac{3}{5}\cdot 0.00595\cdot\frac{5}{10}}{P(E)} = \frac{0.001071}{P(E)}$$

P(no|E)>P(yes|E) => Naïve Bayes predicts **loan default = no** for the new example.

Exercise 1. ***Decision trees and information gain (parts a) and b) – done in class; the rest in your own time)***
Consider the following set of training examples:

shape	color	class
circle	blue	+
circle	blue	+
square	blue	-
triangle	blue	-
square	red	+
square	blue	-
square	red	+
circle	red	+

Adapted from M. Kubat, Introduction to Machine Learning, Springer, 2021

a) What is the entropy of this collection of training examples with respect to the class?
b) What is the information gain of the attribute *shape*?
c) Which attribute will be selected as root of the tree based on information gain?
d) Build the whole decision tree. Draw the tree after each selected attribute.

You may use this table to calculate information gain:

x	y	-(x/y)* log2(x/y)	x	y	-(x/y)* log2(x/y	x	y	-(x/y)* log2(x/y	x	y	-(x/y)* log2(x/y
1	2	0.50	4	5	0.26	6	7	0.19	5	9	0.47
1	3	0.53	1	6	0.43	1	8	0.38	7	9	0.28
2	3	0.39	5	6	0.22	3	8	0.53	8	9	0.15
1	4	0.5	1	7	0.40	5	8	0.42	1	10	0.33
3	4	0.31	2	7	0.52	7	8	0.17	3	10	0.52
1	5	0.46	3	7	0.52	1	9	0.35	7	10	0.36
2	5	0.53	4	7	0.46	2	9	0.48	9	10	0.14
3	5	0.44	5	7	0.35	4	9	0.52			

Solution:

a) H(S)=I(5/8, 3/8) = -5/8 log(5/8)-3/8 log(3/8) = 0.42 + 0.53 = 0.95 bits

b) Split on *shape*:
$H(S_{circle})$=I(3/3, 0/3) = -3/3 log(3/3)-0/3 log(0/3) = 0 + 0 = 0 bits
$H(S_{square})$=I(2/4, 2/4) = -2/4 log(2/4)-2/4 log(2/4) = 0.5 + 0.5 = 1 bit
$H(S_{triangle})$=I(1/1, 0/1) = -1/1 log(1/1)-0/1 log(0/1) = 0 + 0 = 0 bits

H(S|shape)=3/8*0 + 4/8*1 + 1/8*0 = 0.5 bits
gain(shape)=0.95 - 0.5 = 0.45 bits

c) To answer this question we need to calculate the information gain of all attributes. The attribute with the highest information gain will be selected.

There are 2 attributes – *shape* and *color*. We already calculate the information gain for shape. Let's do this for *color*.

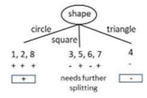
Split on *color*:
$H(S_{blue})$ = I(2/5, 3/5) = -2/5 log(2/5)-3/5 log(3/5) = 0.53 + 0.44 = 0.97 bits
$H(S_{red})$ = I(3/3, 0/3) = -3/3 log(3/3)-0/3 log(0/3)= 0 + 0 = 1 bit

H(S|color) = 5/8*0.97 + 3/8*0 = 0.61 bits
gain(color)=0.95 - 0.61 = 0.34 bits

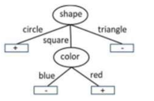
gain(shape) > gain(color) = > shape will be selected as the root of the DT (the first attribi to split on)

d) Building the decision tree:

After selecting *shape*:



We need to repeat the procedure for the examples in the middle branch. The final decision tree i



Exercise 1. ***K-means clustering***
Use the K-means algorithm to cluster them into 2 clusters. The initial centroids are A ... the clusters after the first epoch.

	A	B	C	D	E
A	0	2	7	10	1
B	2	0	3	4	6
C	7	3	0	5	9
D	10	4	5	0	1
E	1	6	9	1	0

Solution:

Initial centroids: A and B
cluster1 – the cluster of A
cluster2 – the cluster of B

Epoch 1 – **start:**
A is assigned to cluster1 (centroid)
B is assigned to cluster2 (centroid)

C:
d(C, A)=7, d(C, B)=3
=> C is assigned to cluster2

D:
d(D, A)=10, d(D, B)=4
=> D is assigned to cluster2

E:
d(E, A)=1, d(E, B)=6
=> E is assigned to cluster1

End of epoch 1, the clusters are: {A, E} and {B, C, D}

Exercise 2. ***Hierarchical clustering***
Use the single link agglomerative clustering from the lecture (given below) to group the dat described by the following distance matrix:

	A	B	C	D	E
A	0	2	7	10	1
B	2	0	3	4	6
C	7	3	0	5	9
D	10	4	5	0	8
E	1	6	9	8	0

Hierarchical clustering algorithm:

```
1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4.         Merge the two closest clusters
5.         Update the proximity matrix
6. Until only a single cluster remains
```

Solution:

1) Each point is in a cluster of itself – 5 clusters: {A}, {B}, {C}, {I
2) The 2 closest clusters are A and E (distance=1), merge them and

	A,E	B	C	D
A,E	0	2	7	8
B	2	0	3	4
C	7	3	0	5
D	8	4	5	0

There are 4 clusters: {A,E}, {B}, {C}, {D}

3) The 2 closest clusters are {A,E} and {B} (distance=2),), merge distance matrix:

	A,B,E	C	D
A,B,E	0	3	4
C	3	0	5
D	4	5	0

There are 3 clusters: {A,B,E}, {C}, {D}

4) The 2 closest clusters are: {A,B,E} and {C} (distance=3), merge distance matrix:

	A,B,C,E	D
A,B,C,E	0	4
D	4	0

There are 2 clusters: {A,B,C,E} and {D}

5) The 2 closest clusters are {A,B,C,E} and {D} (distance=4), merg points now belong to a single cluster: {A,B,C,D,E}

Exercise 1. ***DBSCAN clustering***
Use the DBSCAN algorithm to cluster the items A1, A2, …, A8. The dist
Assume that Eps=2 and MinPts=2.

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	6	3.6	7	7.2	8	2.2
A2		0	6.1	4.2	5	4.1	3.2	4.5
A3			0	5	1.5	1.5	7.5	6.5
A4				0	3.6	4.1	7.2	1.5
A5					0	1.4	6.7	5
A6						0	5.4	5.5
A7							0	7.5
A8								0

Solution:
1) Find the number of points in the neighborhood of each point based on Eps<=2:

N(A1)={A1} as the distance from A1 to all the other points is >2
N(A2)= {A2}
N(A3)= {A3, A5, A6}
N(A4)= {A4, A8}
N(A5)= {A5, A3, A6}
N(A6)= {A6, A3, A5}
N(A7)= {A7}
N(A8)= {A8, A4}

2) Label each example as core, border or noise using MinPts>=2
- A3, A4, A5, A6 and A8 are core as their neighborhood contains 2 or 3 points which is =>2
- A1, A2 and A7 are noise as their neighborhood contains only 1 point (themselves) which is <2 and they are not in the neighborhood of a core point

3) Find the clusters
-A3 is core, and its neighborhood contains A5 and A6 which are also core => A3, A5 and A6 form a cluster
-A4 is core and its neighborhood contains A8 which is core => A4 and A8 form a cluster
Final clustering: K1={A3, A5, A6}, K2={A4, A8}

$$Correlation\ r = \frac{n\sum xy - \sum x \sum y}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

$$corr(x,y) = \frac{covar(x,y)}{std(x) * std(y)}$$

$$mean(x) = \frac{\sum_{k=1}^{n} x_k}{n} \qquad std(x) = \sqrt{\frac{\sum_{k=1}^{n}(x_k - mean(x))^2}{n-1}}$$

$$covar(x,y) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - mean(x))(y_k - mean(y))$$

Exercise 2. ***Evaluating clustering quality using the silhouette coefficient***
Given are 4 items P1, P2, P3 and P4. They were clustered using a clustering algorithm. The cluster labels and the distance matrix are shown below. Evaluate the quality of the clustering by computing the silhouette coefficient for each point, each of the 2 clusters and the overall clustering.

Distance matrix:

	P1	P2	P3	P4
P1	0	0.1	0.65	0.55
P2		0	0.7	0.6
P3			0	0.3
P4				0

Cluster labels:

point	cluster label
P1	1
P2	1
P3	2
P4	2

Solution:
The algorithm has found 2 clusters K1={P1, P2} and K2={P3, P4}. The silhouette coefficient measures both cluster cohesion and separation.

a_i – the average distance from example *i* to all examples in the same cluster
b_i – the minimum of the average distance of *i* to all examples in other clusters
Silhouette coefficient $s_i=(b_i-a_i)/\max(a_i,b_i)$

For each point:
For P1: a_1=0.1, b_1=(0.65+0.55)/2, s_1=0.833
For P2: a_2=0.1, b_2=(0.7+0.6)/2, s_2=0.846
For P3: a_3=0.3, b_3=(0.65+0.7)/2, s_3=0.556
For P4: a_4=0.3, b_4=(0.55+0.6)/2, s_4=0.478

For each cluster:
For cluster K1, the averaged silhouette coefficient s_1=(0.833+0.846)/2=0.84
For K2, s_2=(0.556+0.478)/2=0.52

Overall for the clustering:
s=(0.84+0.52)/2=0.68, relatively good (positive and close to 1)

Exercise 3. ***Evaluating clustering quality using correlation***
For the data from the previous exercise, evaluate the clustering quality using the correlation between the similarity matrix derived from the distance matrix (given below) and the similarity matrix derived from the clustering results (i.e. the matrix whose ij entry is 1 if two objects belong to the same cluster and 0 otherwise).

The similarity matrix derived from the distance matrix is given below. It was computed from the distance matrix as *s =1 - (d - dmin)/(dmax - dmin)*, where *dmin* and *dmax* are the minimum and maximum distances in the matrix: *dmin*=0.1 and *dmax*=0.7).
Similarity matrix:

	P1	P2	P3	P4
P1	1	1	0.08	0.25
P2		1	0	0.17
P3			1	0.67
P4				1

Solution:
Given that there are two clusters K1={P1, P2} and K2={P3, P4}, the ideal similarity matrix is:

	P1	P2	P3	P4
P1	1	1	0	0
P2		1	0	0
P3			1	1
P4				1

Thus, we need to compute the correlation between the vectors
x=[1, 0.08, 0.25, 0, 0.17, 0.67] and
y=[1,0,0,0,0,1]

Recall that:

$$corr(\mathbf{x},\mathbf{y}) = \frac{covar(\mathbf{x},\mathbf{y})}{std(\mathbf{x})std(\mathbf{y})} \qquad std(\mathbf{x}) = \sqrt{\frac{\sum_{k=1}^{n}(x_k - mean(\mathbf{x}))^2}{n-1}} \qquad mean(\mathbf{x}) = \frac{\sum_{k=1}^{n} x_k}{n}$$

$$covar(\mathbf{x},\mathbf{y}) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - mean(x))(y_k - mean(y))$$

mean(x)=0.36, std(x)=0.39
mean(y)=0.33, std(y)=0.52
covar(x,y)=0.16

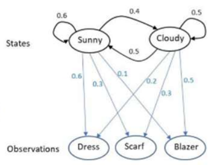
covar(x,y)=[(1-0.36)(1-0.33)+(0.08-0.36)(-0.33)+(0.25-0.36)(-0.33)+(-0.36)(-0.33)+(0.17-0.36)(-0.33)+(0.67-0.36)(1-0.33)] / (6-1) = 0.189

=>corr(x,y)=0.189/(0.39*0.52)=0.93 => high correlation (close to 1) => items that are close to each other are in the same cluster => good clustering

. Hidden Markov models

positive to COVID and had to quarantine at home for several days. Her friend Nicole came to od every day. We don't know what the weather was on the quarantine days but we know the type Nicole wore and it provides evidence about the weather.

ng Hidden Markov Model models the situation. The initial state probabilities are: A_0(Sunny)=0.5 udy)=0.5.



Suppose that on the first quarantine day Nicole wore a dress and on the second she wore a blazer.

a) What is the probability of the observation sequence?
b) What is the most likely sequence of hidden states?

Briefly show your calculations.

Solution:

States: Sunny (S) and Cloudy (C)
Observations: Dress (D), Scarf (Sc) and Blazer (B)
Observation sequence: X=Dress (D), Blazer (B)

a) This is HMM problem 1. We can use the Forward algorithm to solve it.

Step 1: Initialization

Day 1: D observed

$V_S(1) = A_0(S)e_S(D)$= *0.5*0.6=0.3*
$V_C(1) = A_0(C)e_C(D)$= *0.5*0.2=0.1*

Step 2: Iteration

Day 2: B observed

$V_S(2) = e_S(B)*(V_S(1)a_{SS} + V_C(1)a_{CS})$=
= 0.1*(0.3*0.6+ 0.1*0.5)=0.1*(0.18+0.05)=0.023

$V_C(2) = e_C(B)*(V_S(1)a_{SC} + V_C(1)a_{CC})$=
= 0.5*(0.3*0.4 + 0.1*0.5)=0.5*(0.12+0.05)=0.085

Step 3: Termination

P(X)=P(D,B)=0.023+0.085=0.108

The probability of the observation sequence Dress, Blazer is 0.108.

b) This is HMM problem 2. We can use the Viterbi algorithm to solve it.

Step 1: Initialization

Day 1: D observed

$V_S(1) = A_0(S)e_S(D)$= *0.5*0.6=0.3*
$V_C(1) = A_0(C)e_C(D)$= *0.5*0.2=0.1*

Step 2: Iteration

Day 2: B observed

$V_S(2) = e_S(B)*\max(V_S(1)a_{SS}, V_C(1)a_{CS})$=
= 0.1*max(**0.3*0.6,** 0.1*0.5)=0.1*0.3*0.6=0.018
$Ptr_S(2)$ = argmax(**0.3 ∙ 0.6**, 0.1 ∙ 0.5) = 1, i.e. S

$V_C(2) = e_C(B)*\max(V_S(1)a_{SC}, V_C(1)a_{CC})$=
= 0.5*max(**0.3*0.4,** 0.1*0.5)=0.5*0.3*0.4=0.06
$Ptr_C(2)$ = argmax(**0.3 ∙ 0.4**, 0.1 ∙ 0.5) = 1, i.e. S

Step 3: Termination and trace-back

Final state = argmax($V_S(2), V_C(2)$)=argmax(0.018, **0.06**)=2, i.e. C

Trace-back through pointers:
$Ptr_C(2)$ =S

Hence, the most likely sequence of hidden states is S, C (Sunny, Cloudy)