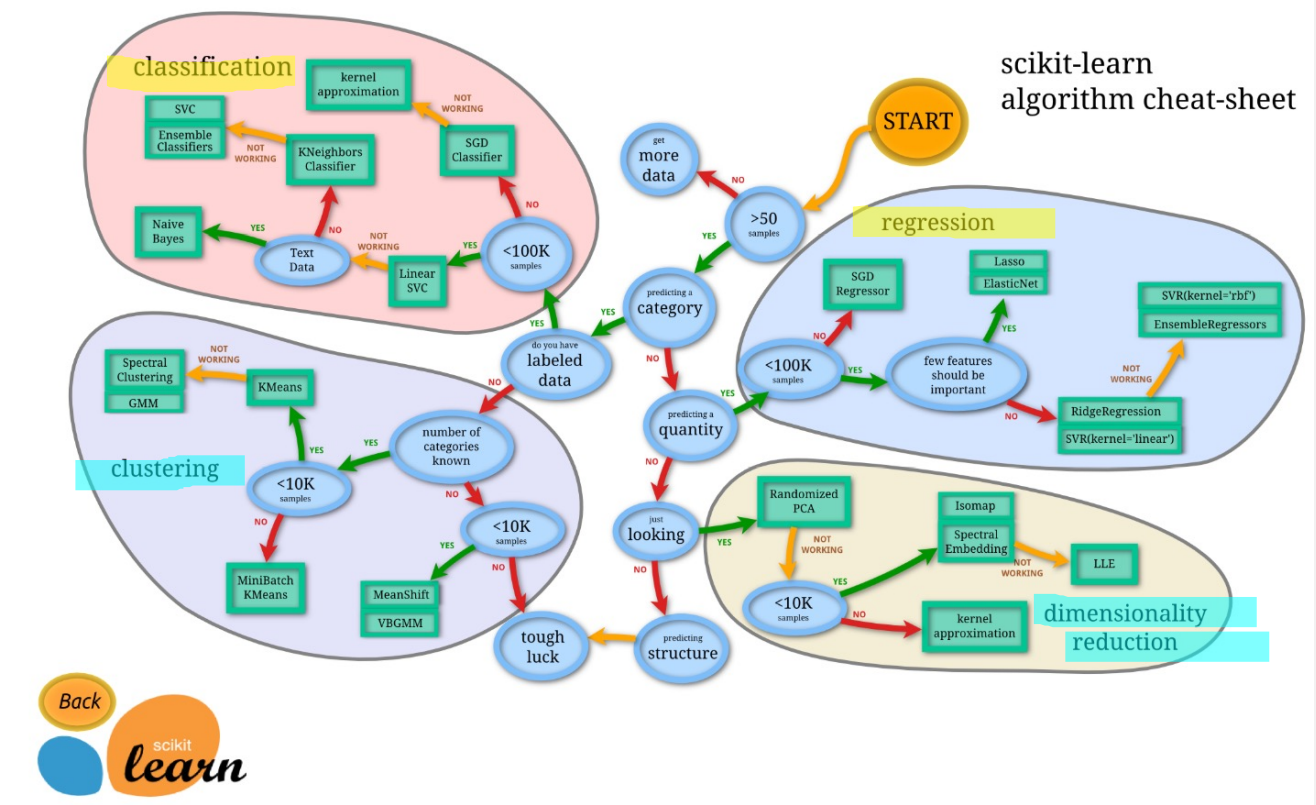
K-NEAREST NEIGHBOR

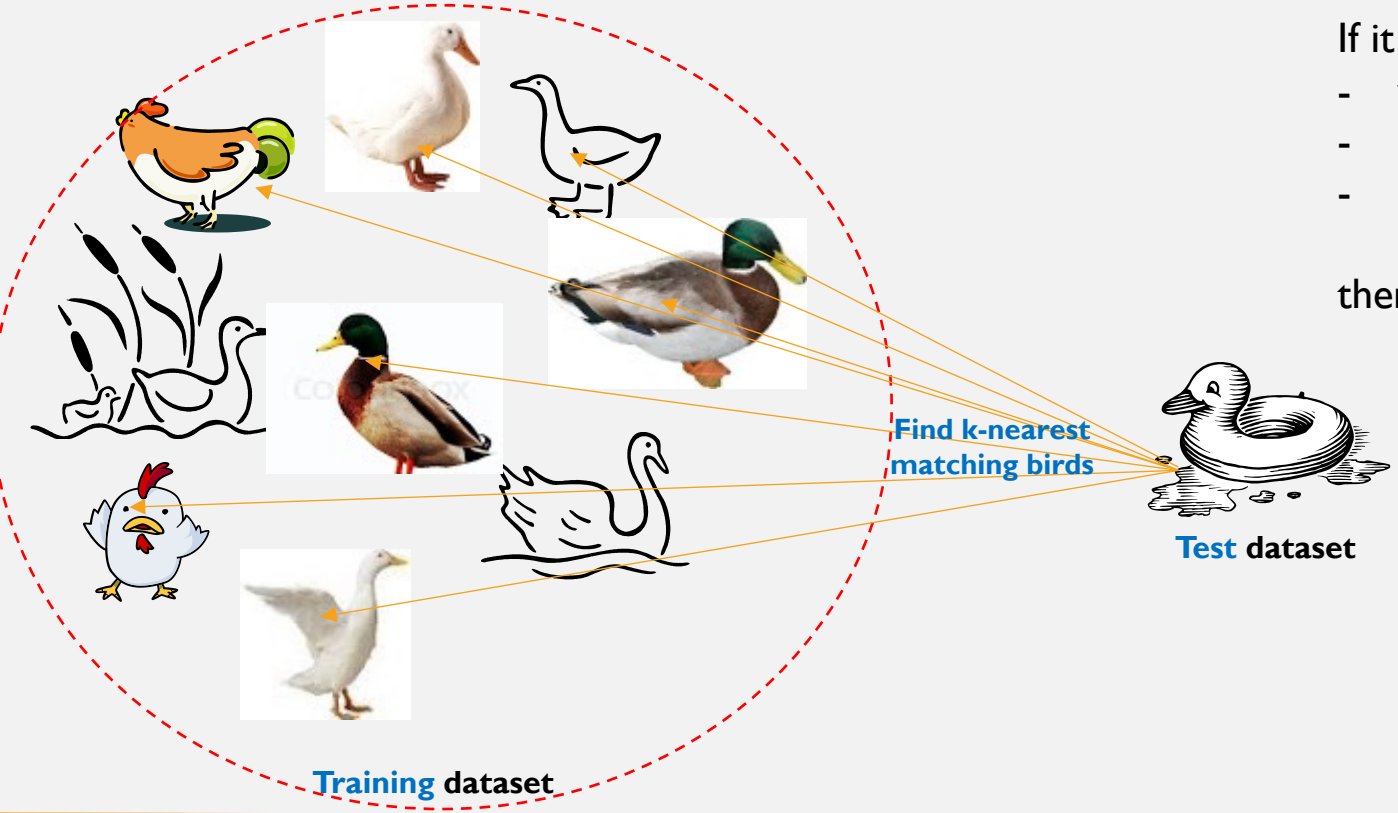
Data Science Overview

- Know what KNN is
- Algorithm - steps
- Process - body
- output
- Evaluation methods
- Evaluation measures
- Application

SCIKIT-LEARN – CHEAT SHEET



KNN – BASIC IDEA



If it

- **walks** like a duck,
- **quacks** like a duck,
- **Eats** like a duck

then it's probably a duck !!

KNN – WHAT IT IS

KNN can be used for both classification and regression predictive problems.

KNN falls in the supervised learning family of algorithms.

Informally, this means that we are given a labelled dataset consisting of training observations (x, y) and would like to capture the relationship between x and y.

More formally, our goal is to learn a function $h:X \rightarrow Y$ so that given an unseen observation x, $h(x)$ can confidently predict the corresponding output y.

KNN – WHAT IT IS

- The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM).

- Despite its simplicity, KNN can outperform more powerful classifiers and is used in a variety of applications such as economic forecasting, data compression and genetics.

- Lazy Learning Algorithm (instance based learning)
  - Defer the decision to generalize beyond the training examples till a new test is encountered
  - Whenever we have a new point to classify, we find its K nearest neighbors from the training data.

- The distance is calculated using one of the following measures
  - Euclidean Distance
  - Minkowski Distance
  - Mahalanobis Distance

ASSUMPTIONS

- Suitable for applications for which sufficient domain knowledge is available.
- This knowledge supports the selection of an appropriate measure of feature set

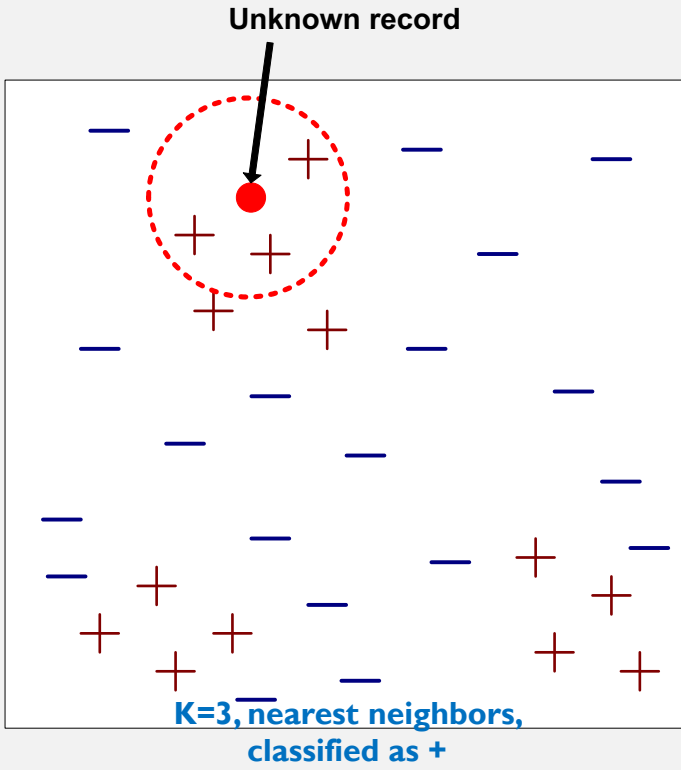
KNN – ALGORITHM (STEPS)

Requires three things

- The set of pre-classified training samples
- Distance Metric to compute distance between test and all the training samples
- The value of *k*, the number of nearest neighbors to retrieve

To classify an unknown record:

- Compute distance to other training samples
- Identify *k* nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking **majority** vote)



DISTANCE CALCULATION

- commonly based on the Euclidean distance between a test sample and the specified training samples.
- Let x_i be an input sample with p features $(x_{i1}, x_{i2}, ..., x_{ip})$, n be the total number of input samples $(I = 1, 2, ..., n)$.

- The Euclidean distance between sample x_i and x_l is defined as:

$$d(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \ldots + (x_{ip} - x_{lp})^2}$$

PROXIMITY MEASURES FOR HOMOGENEOUS DATA

- City-block or Manhattan distance function

$$d(x,y) = \sum_{a=1}^{m} |x_a - y_a|$$

- Minkowskian r-distance function

$$d(x,y) = \sqrt[r]{\sum_{a=1}^{m} (x_a - y_a)^r}$$

- Mahalanobis distance function

$$d(x,y) = [det\ V]^{1/m} (x - y)^T V^{-1} (x - y),$$

where V is a covariance matrix of A_1 to A_m

- Canberra distance function

$$d(x,y) = \sum_{a=1}^{m} \frac{|x_a - y_a|}{|x_a + y_a|}$$

- Chebychev distance function

$$d(x,y) = max_{a=1}^{m} |x_a - y_a|$$

- Quadratic distance function

$$d(x,y) = (x - y)^T Q(x - y) = \sum_{b=1}^{m} \{\sum_{a=1}^{m} (x_a - y_a) q_{ab}\} (x_b - y_b),$$

where Q is a problem-specific positive definite m × m weight matrix

- correlation distance function

$$d(x,y) = \frac{\sum_{a=1}^{m} (x_a - \overline{x}_a)(y_a - \overline{y}_a)}{\sqrt{\sum_{a=1}^{m} (x_a - \overline{x}_a)^2 \sum_{a=1}^{m} (y_a - \overline{y}_a)^2}}$$

- chi-square distance function

$$d(x,y) = \sum_{a=1}^{m} \frac{1}{sum_a} \left(\frac{x_a}{size_x} - \frac{y_a}{size_y} \right)^2$$

where sum_a is the sum of all values for attribute a occuring in the training set and $size_x$ and $size_y$ are the sums of all values in the instance x and y respectively.

The above mentioned distance functions work well for quantitative attributes, but they do not have the solution for nominal, ordinal or heterogeneous data.

OUTPUT

The output depends on whether you use the KNN algorithm for classification or regression.

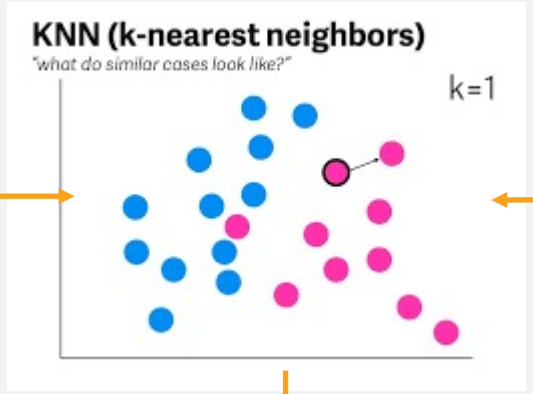
1. In KNN classification, the predicted class label is determined by the voting for the nearest neighbors, that is, the majority class label in the set of the selected k instances is returned.

2. In KNN regression, the average value of the target function values of the nearest neighbors is returned as the predicted value.

EVALUATION – 1 (RANDOM INPUT TEST SAMPLE)

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Training dataset



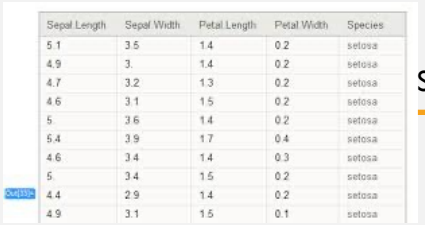
Test sample input

```
inputarray.append(input('first attribute : '))
inputarray.append(input('second attribute : '))
inputarray.append(input('third attribute : '))
inputarray.append(input('fourth attribute : '))
```

- Entire training set considered
- Test sample entered externally

	sepal_length	sepal_width	petal_length	petal_width	Class_name	distance
106	4.9	2.5	4.5	1.7	Iris-virginica	3.209361
114	5.8	2.8	5.1	2.4	Iris-virginica	3.416138
121	5.6	2.8	4.9	2.0	Iris-virginica	3.439477
59	5.2	2.7	3.9	1.4	Iris-versicolor	3.478505

EVALUATION – 2 (LEAVE-ONE-OUT CROSS-VALIDATION - LOOCV)



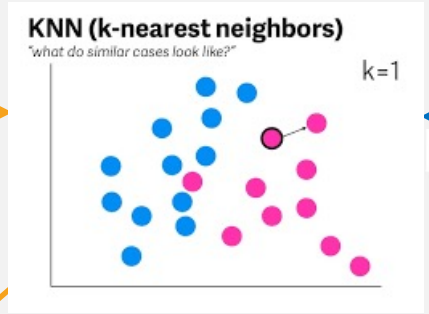
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Training dataset

Split into

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	NaN	setosa
2	4.7	NaN	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	NaN

Training



1 test sample

4	5.0	3.6	1.4	0.2	NaN

	sepal_length	sepal_width	petal_length	petal_width	Class_name	distance
106	4.9	2.5	4.5	1.7	Iris-virginica	3.209361
114	5.8	2.8	5.1	2.4	Iris-virginica	3.416138
121	5.6	2.8	4.9	2.0	Iris-virginica	3.439477
59	5.2	2.7	3.9	1.4	Iris-versicolor	3.478505

- is a special case of cross-validation method in which each instance is used once as the test case and all other instances are used as the training set.
- also called as n-fold cross validation.
- utilizes the utmost training instances
- but due to its expensive nature it is usually applied to small datasets.

EVALUATION – 3 (HOLDOUT METHOD)

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Training dataset

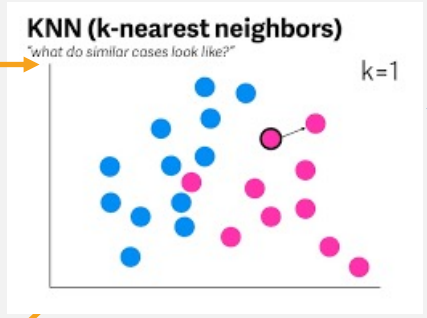
Split into

Training

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	NaN	setosa
2	4.7	NaN	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa

Test

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	NaN	setosa
2	4.7	NaN	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa



Test sample

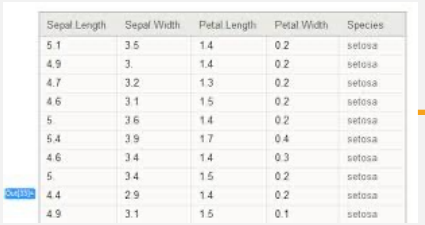
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	NaN	setosa
2	4.7	NaN	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa

	sepal_length	sepal_width	petal_length	petal_width	Class_name	distance
106	4.9	2.5	4.5	1.7	Iris-virginica	3.209361
114	5.8	2.8	5.1	2.4	Iris-virginica	3.416138
121	5.6	2.8	4.9	2.0	Iris-virginica	3.439477
59	5.2	2.7	3.9	1.4	Iris-versicolor	3.478505

Repeated for each test sample

- original data set is partitioned into two parts
- 50-50 or 70-30 or 60-40
- randomly divided into the training and test sets

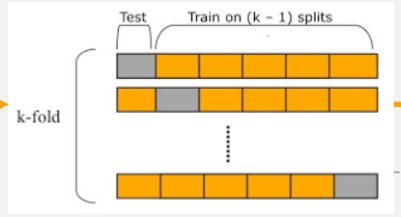
EVALUATION – 4 (K-FOLD CROSS-VALIDATION METHOD)

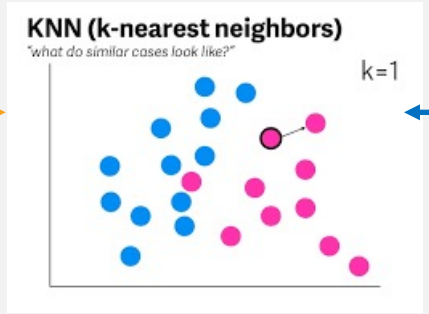


Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Training dataset

Split into





Test sample

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	NaN	setosa
2	4.7	NaN	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa

Repeated for each fold

	sepal_length	sepal_width	petal_length	petal_width	Class_name	distance
106	4.9	2.5	4.5	1.7	Iris-virginica	3.209361
114	5.8	2.8	5.1	2.4	Iris-virginica	3.416138
121	5.6	2.8	4.9	2.0	Iris-virginica	3.439477
59	5.2	2.7	3.9	1.4	Iris-versicolor	3.478505

Repeated for each test sample

- Popular choices for K are 3, 5 and 10 as they're manageable computationally
- The cross-validation method is used with moderate datasets having instances around hundreds or more.

SKLEARN - KNEIGHBORSCLASSIFIER

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5,
                                              weights='uniform',
                                              algorithm='auto',
                                              leaf_size=30,
                                              p=2,
                                              metric='minkowski',
                                              metric_params=None,
                                              n_jobs=1, **kwargs)
```

- 'n_neighbors' are the number of neighbors that will vote for the class of the target point
- default is 5.
- An odd number is preferred to avoid any tie.

SKLEARN - KNEIGHBORSCLASSIFIER

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

- 'weights' parameter has two choices: 'uniform' and 'distance'.
  - For the 'uniform' weight, each of the k neighbors has equal vote whatever its distance from the target point.
  - If the weight is 'distance' then voting weightage or importance varies by inverse of distance; those points who are nearest to the target point have greater influence than those who are farther away.
  - Default is 'uniform'

SKLEARN - KNEIGHBORSCLASSIFIER

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

- Parameter 'metric' decides how distances are calculated in space.
  - Euclidean distance
  - Manhattan distance are also used.
  - A general formulation of distance metric is 'minkowski' distance. (default, along with p=2)

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
"euclidean"	EuclideanDistance		sqrt(sum((x - y)^2))
"manhattan"	ManhattanDistance		sum(|x - y|)
"chebyshev"	ChebyshevDistance		max(|x - y|)
"minkowski"	MinkowskiDistance	p	sum(|x - y|^p)^(1/p)
"wminkowski"	WMinkowskiDistance	p, w	sum(w * |x - y|^p)^(1/p)
"seuclidean"	SEuclideanDistance	V	sqrt(sum((x - y)^2 / V))
"mahalanobis"	MahalanobisDistance	V or VI	sqrt((x - y)' V^-1 (x - y))

SKLEARN - KNEIGHBORSCLASSIFIER

Metrics intended for two-dimensional vector spaces:

Note that the haversine distance metric requires data in the form of [latitude, longitude] and both inputs and outputs are in units of radians.

identifier	class name	distance function
“haversine”	HaversineDistance	•2 arcsin(sqrt(sin^2(0.5*dx)cos(x1)cos(x2)sin^2(0.5*dy)))

SKLEARN - KNEIGHBORSCLASSIFIER

Metrics intended for integer-valued vector spaces: Though intended for integer-valued vectors, these are also valid metrics in the case of real-valued vectors.

identifier	class name	distance function
“hamming”	HammingDistance	N_unequal(x, y) / N_tot
“canberra”	CanberraDistance	sum(|x - y| / (|x| + |y|))
“braycurtis”	BrayCurtisDistance	sum(|x - y|) / (sum(|x|) + sum(|y|))

SKLEARN - KNEIGHBORSCLASSIFIER

Metrics intended for boolean-valued vector spaces: Any nonzero entry is evaluated to "True". In the listings below, the following abbreviations are used:

N : number of dimensions
NTT : number of dims in which both values are True
NTF : number of dims in which the first value is True, second is False
NFT : number of dims in which the first value is False, second is True
NFF : number of dims in which both values are False
NNEQ : number of non-equal dimensions, NNEQ = NTF + NFT
NNZ : number of nonzero dimensions, NNZ = NTF + NFT + NTT

identifier	class name	distance function
"jaccard"	JaccardDistance	NNEQ / NNZ
"matching"	MatchingDistance	NNEQ / N
"dice"	DiceDistance	NNEQ / (NTT + NNZ)
"kulsinski"	KulsinskiDistance	(NNEQ + N - NTT) / (NNEQ + N)
"rogerstanimoto"	RogersTanimotoDistance	2 * NNEQ / (N + NNEQ)
"russellrao"	RussellRaoDistance	NNZ / N
"sokalmichener"	SokalMichenerDistance	2 * NNEQ / (N + NNEQ)
"sokalsneath"	SokalSneathDistance	NNEQ / (NNEQ + 0.5 * NTT)

DISTANCE FUNCTION – WHEN TO USED WHAT

identifier	Purpose	
euclidean	• For numeric features • Symmetric, treats all dimensions equally. Sensitive to extreme values	sqrt(sum((x - y)^2))
Hamming	• For categorical features	N_unequal(x, y) / N_tot
Minkowski	Default (equivalent of Euclidean) If p =2 , Euclidean If p=1, Manhattan	sum(|x - y|^p)^(1/p)
Manhattan		sum(|x - y|)
mahalanobis		sqrt((x - y)' V^-1 (x - y))

SKLEARN - KNEIGHBORSCLASSIFIER

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

- Parameter 'n_jobs' - The number of parallel jobs to run for neighbors search.
- If -1, then the number of jobs is set to the number of CPU cores.
- Default is 1

SKLEARN - KNEIGHBORSCLASSIFIER

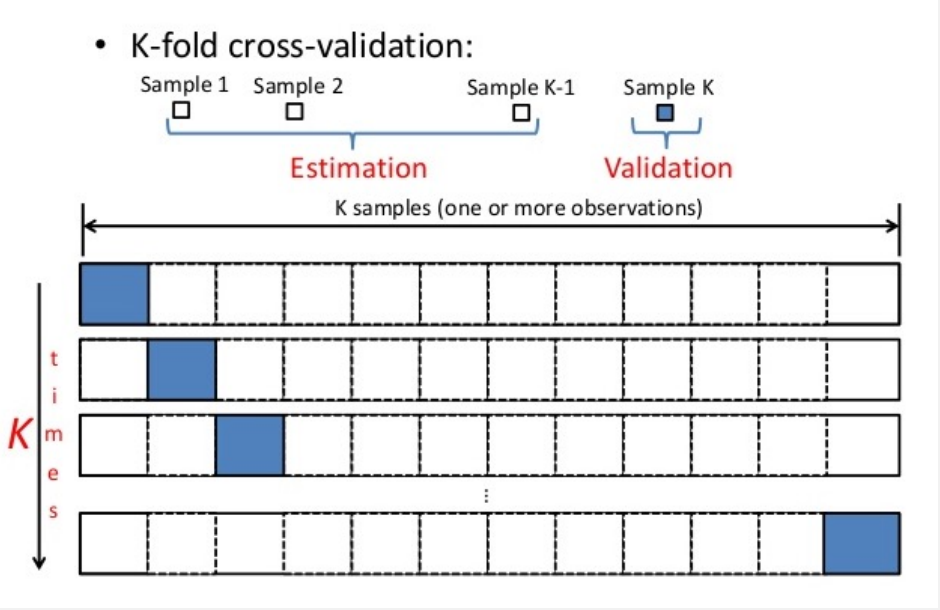
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

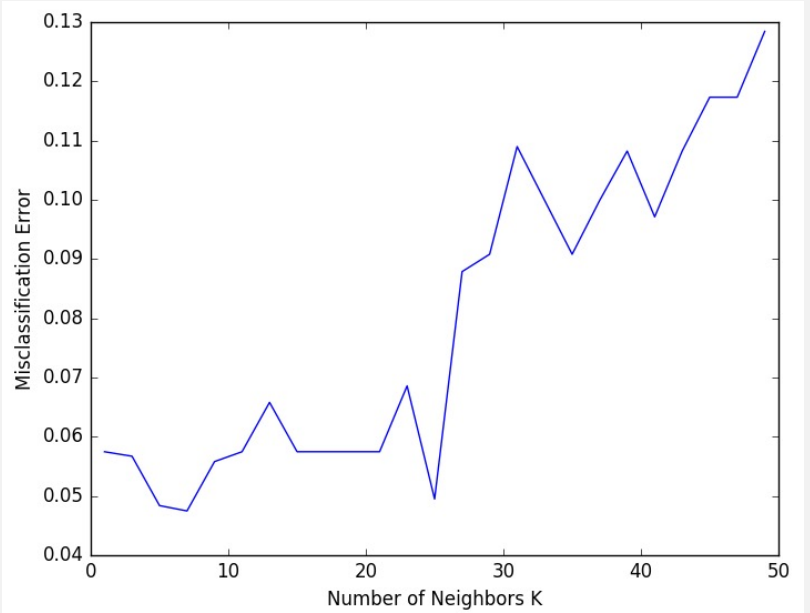
- 3 different nearest neighbors algorithms:
  - BallTree,
  - KDTree, and
  - a brute-force

- The choice of neighbors search algorithm is controlled through the keyword 'algorithm', which must be one of ['auto', 'ball_tree', 'kd_tree', 'brute'].

- When the default value 'auto' is passed, the algorithm attempts to determine the best approach from the training data.

BEST K (NEIGHBORS)





10-fold cross validation tells us that K=7 results in the lowest validation error.

K - VALUE

In general, the optimal value for K will depend on the bias-variance tradeoff.

- A small value for K provides the most flexible fit, which will have low bias but high variance.
    - This variance is due to the fact that the prediction in a given region is entirely dependent on just one observation.
- In contrast, larger values of K provide a smoother and less variable fit;
    - the prediction in a region is an average of several points, and so changing one observation has a smaller effect

CATEGORICAL VARIABLES

Ordinal (along with other continuous variables)	Nominal
Create dummy variables out of a categorical variable and include them instead of original categorical variable. For example, a categorical variable named "Department" has 5 unique levels / categories. So we will create 5 dummy variables. Each dummy variable has 1 against its department and else 0.	KNN does not do well in this case

PROS AND CONS

Pros:

- No assumptions about data—useful, for example, for nonlinear data
- Simple algorithm—to explain and understand/interpret
- High accuracy (relatively)—it is pretty high but not competitive in comparison to better supervised learning models
- Versatile—useful for classification or regression

Cons:

- Computationally expensive—because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data

SCIKIT - KNN – TREE TUNING

Algorithms	
Brute Force	• most naive neighbor search implementation involves the brute-force computation of distances between all pairs of points in the dataset • competitive for small data samples • as the number of samples grows, the brute-force approach quickly becomes infeasible • {algorithm = 'brute'}
K-D Tree	• these structures attempt to reduce the required number of distance calculations by efficiently encoding aggregate distance information for the sample • The basic idea is that if point A is very distant from point B and point B is very close to point C, then we know that points A and C are very distant,!!! • KD tree approach is very fast for low-dimensional neighbors searches, it becomes inefficient as grows very large {algorithm = 'kd_tree'}
Ball Tree	• Faster than K-D tree {algorithm = 'ball_tree'}

BRUTE FORCE

- The most naive neighbor search implementation involves the brute-force computation of distances between all pairs of points in the dataset

- Efficient brute-force neighbors searches can be very competitive for small data samples. However, as the number of samples grows, the brute-force approach quickly becomes infeasible

- algorithm = 'brute'

- 'Brute Force' implementation consists of 3 stages
  - 1st stage is to calculate all of the `distances' from each test point to every reference point in the training set
  - 2nd stage is to sort these distances and select the k objects that are the closest
  - 3rd stage is final classification

SPEEDING UP KNN – USING KD-TREE

- K-Dimensional Tree, Invented in 1970s by Jon Bentley
- Name originally meant "3d-trees, 4d-trees, etc" where k was the # of dimensions, A k-dimensional tree is a binary tree.
- In the generic k-NN model, each time a prediction is to be made for a test point, first this test point's distance from all other points is calculated and then only nearest k-points can be discovered for voting.
- This approach is also known as brute-force approach.
- With increasing data volume and dimensionality, this repeated distance calculations is COSTLY
- To speed up and to avoid measuring distances from all the points in the data set, some prepossessing of training data is done.

KD-TREE – EXAMPLE CONSTRUCTION

1. Consider a three dimensional (training) data set.
2. 3 attributes ‘a’, ‘b’ and ‘c’.
3. Among the three, attribute ‘b’ has the greatest variance.
4. We sort the data set on the attribute ‘b’
5. Divide it into 2 parts at the median.

Refer to the excel sheet

a	b	c
22	38	21
4	8	6
2	14	3
8	20	12
10	26	18
12	32	15
18	56	33
16	44	27
20	50	24
14	62	30
6	2	9

	a	b	c
Mean	12	32	18
Variance	44	396	99
SD	6.63	19.90	9.95

Column ‘b’ has max variance

a	b	c
6	2	9
4	8	6
2	14	3
8	20	12
10	26	18
12	32	15
22	38	21
16	44	27
20	50	24
18	56	33
14	62	30

	a	b	c
Median	12	32	18

median is at (12,32,15).

BALL TREE

- In computer science, a ball tree, or metric tree, is a space partitioning data structure for organizing points in a multi-dimensional space.
- The ball tree gets its name from the fact that it partitions data points into a nested set of hyperspheres known as "balls".
- The resulting data structure has characteristics that make it useful for a number of applications, most notably nearest neighbor search.

CHOICE OF TREE ALGORITHM

The optimal algorithm for a given dataset is a complicated choice and depends on a number of factors:

- number of samples (i.e. n_samples) and dimensionality (i.e. n_features).
  - For small data sets, brute force algorithms can be more efficient than a tree-based approach.
  - Both KDTree and BallTree address this through providing a leaf size parameter: this controls the number of samples at which a query switches to brute-force.
- number of neighbors
  - Brute force query time is largely unaffected by the value of k

KNN REGRESSOR

1. Assume a value for the number of nearest neighbors K and a prediction point x_o.
2. KNN identifies the training observations No closest to the prediction point x_o.
3. KNN estimates f (x_o) using the average of all the responses in No, i.e.

$$\hat{f}(x_o) = \frac{1}{K} \sum_{x_i \in N_o} y_i$$

KNN – SOME CONSIDERATIONS

considerations	Comments
Parametric or non parametric ?	KNN is a non-parametric machine learning algorithm
Features normality?	Non-parametric means that KNN does not make assumptions about the distribution of the data it is modeling
Categorical features?	For binary and ordinal, converting to numbers makes sense. *** But for nominal, it does not make sense.
Normalizing features?	Required
Affect of outliers?	Needs to be removed
All attributes are equally important	KNN assumes that all attributes are equally important. Feature Engg required
Affect of features collinearity	KNN makes no assumption about the data.

KNN – SOME CONSIDERATIONS

considerations	Comments
Noisy instances	• Noisy instances are instances with a bad target class. If the dataset is noisy , then by accident we might find an incorrectly classified training instance as the nearest one to our test instance. • Majority vote over K nearest neighbors instances . • Identification of reliable "prototypes" for each class
Identify noisy data	• By changing the k, if the accuracy changes a lot between various settings of k it's may be a noisy data set
Mixed data types (continuous and categorical)	• Use GOWER method for distance calculation
KNN for only categorical variables	

APPLICATION

- **Text mining**
  - k-NN is often used in search applications where we are looking for "similar" items;
    - when our task is some form of "find items similar to this one".

  - searching for semantically similar documents (i.e., documents containing similar topics), this is referred to as Concept Search

  - Recommender Systems - If you know a user likes a particular item, then you can recommend similar items for them.
    - To find similar items, compare the set of users who like each item—
    - recommending products, media to consume, or even 'recommending' advertisements to display to a user!

APPLICATION

- **Finance**
  - Stock market forecasting
  - Best time to purchase the stocks
  - what stocks to purchase.
  - Forecasting stock market: Predict the price of a
  - stock, on the basis of company performance
  - measures and economic data.
    - ☐ Currency exchange rate
    - ☐ Bank bankruptcies
    - ☐ Understanding and managing financial risk
    - ☐ Trading futures
    - ☐ Credit rating
    - ☐ Loan management
    - ☐ Bank customer profiling
    - ☐ Money laundering analyses

- **Medicine**
- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.
- Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person's blood.
- Identify the risk factors for prostate cancer, based on clinical and demographic variables.

WEIGHTED K NEAREST NEIGHBOR

Approach 1

- Associate weights with the attributes

- Assign weights according to the relevance of attributes
  - Assign random weights
  - Calculate the classification error
  - Adjust the weights according to the error
  - Repeat till acceptable level of accuracy is reached

WEIGHTED K NEAREST NEIGHBOR

Approach 2

- Backward Elimination
- Starts with the full set of features and greedily remove the one that most improves performance, or degrades performance slightly

WEIGHTED K NEAREST NEIGHBOR

Approach 3 (Instance Weighted)

- Gradient Descent
- Assign random weights to all the training instances
- Train the weights using Cross Validation

WEIGHTED K NEAREST NEIGHBOR

Approach 4 (Attribute Weighted)

- Gradient Descent
- Assign random weights to all the instances
- Train the weights using Cross Validation

DEALING WITH CATEGORICAL DATA

Not all data has numerical values. Here are examples of categorical data:

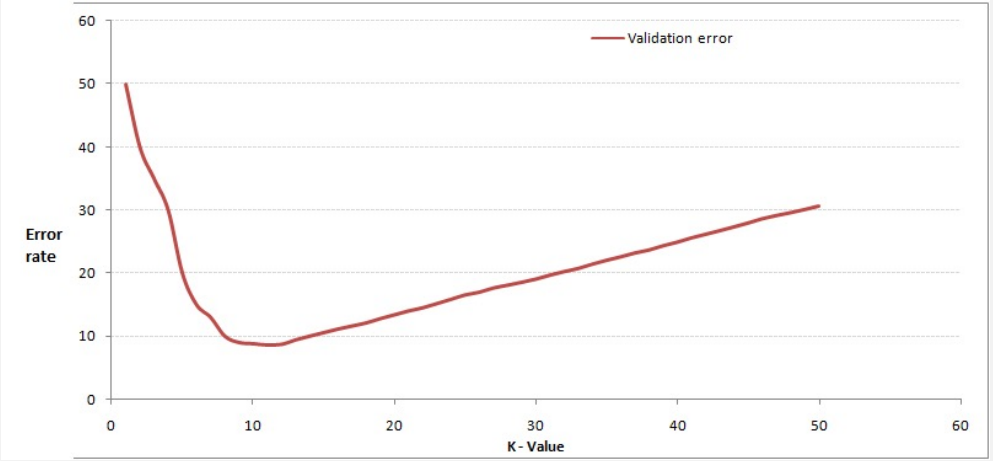
- The blood type of a person: A, B, AB or O.
- The state that a resident of the INDIA lives in.
- T-shirt size. XL > L > M
- T-shirt color
- Rating

QS

1. k-NN algorithm does more computation on test time rather than train time.

A) TRUE
B) FALSE

2. In the image below, which would be the best value for k assuming that the algorithm you are using is k-Nearest Neighbor.



QS

Which of the following distance metric can not be used in k-NN?

A) Manhattan
B) Minkowski
C) Tanimoto
D) Jaccard
E) Mahalanobis
F) All can be used

Which of the following option is true about k-NN algorithm?

A) It can be used for classification
B) It can be used for regression
C) It can be used in both classification and regression

QS

Which of the following statement is true about k-NN algorithm?

1. k-NN performs much better if all of the data have the same scale
2. k-NN works well with a small number of input variables (p), but struggles when the number of inputs is very large
3. k-NN makes no assumptions about the functional form of the problem being solved

A) 1 and 2
B) 1 and 3
C) Only 1
D) All of the above

QS

Which of the following is true about Manhattan distance?

A) It can be used for continuous variables
B) It can be used for categorical variables
C) It can be used for categorical as well as continuous
D) None of these

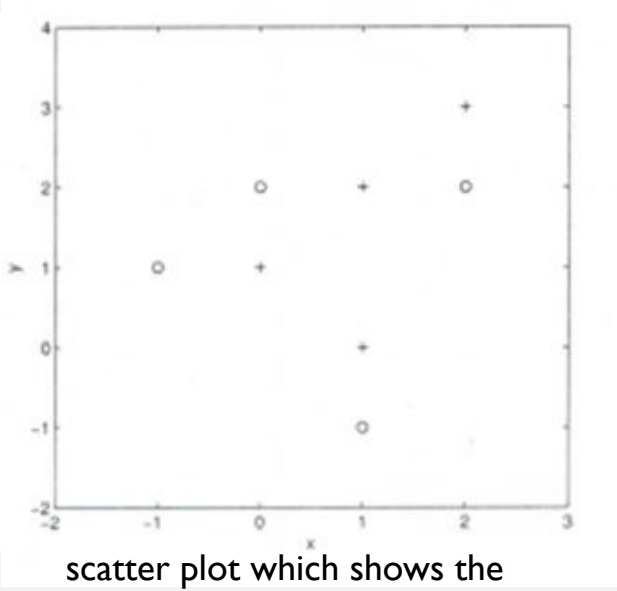
Which of the following distance measure do we use in case of categorical variables in k-NN?

1. Hamming Distance
2. Euclidean Distance
3. Manhattan Distance

QS

Suppose, you have given the following data where x and y are the 2 input variables and Class is the dependent variable.

x	y	Class
−1	1	−
0	1	+
0	2	−
1	−1	−
1	0	+
1	2	+
2	2	−
2	3	+



scatter plot which shows the above data in 2D space.

you want to predict the class of new data point x=1 and y=1 using Euclidian distance in 3-NN. In which class this data point belong to?

A) + Class
B) – Class
C) Can't say
D) None of these

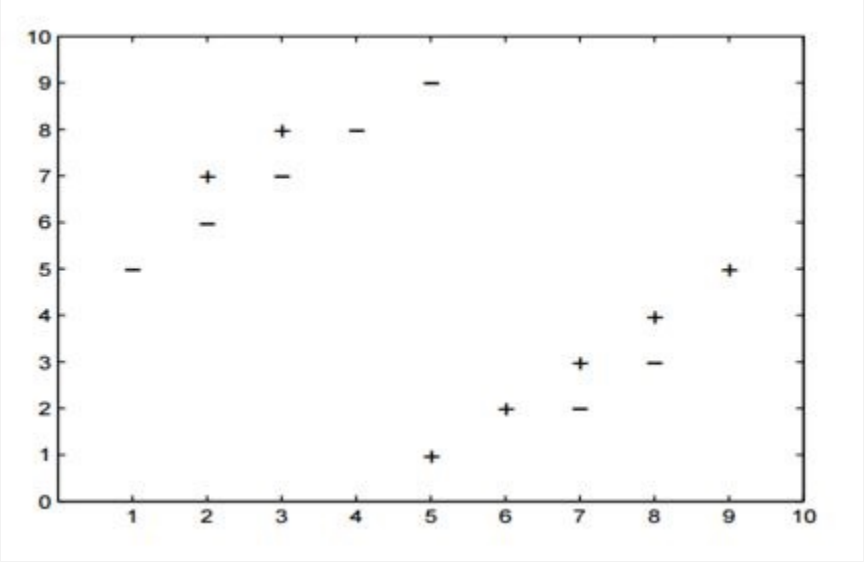
you are now want use 7-NN instead of 3-KNN which of the following x=1 and y=1 will belong to?

QS

Suppose you have given the following 2-class data where "+" represent a positive class and "-" is represent negative class.

Which of the following value of k in k-NN would minimize the leave one out cross validation accuracy?

A) 3
B) 5
C) Both have same
D) None of these



QS

Which of the following will be true about k in k-NN in terms of Bias?

A) When you increase the k the bias will be increases
B) When you decrease the k the bias will be increases
C) Can't say
D) None of these

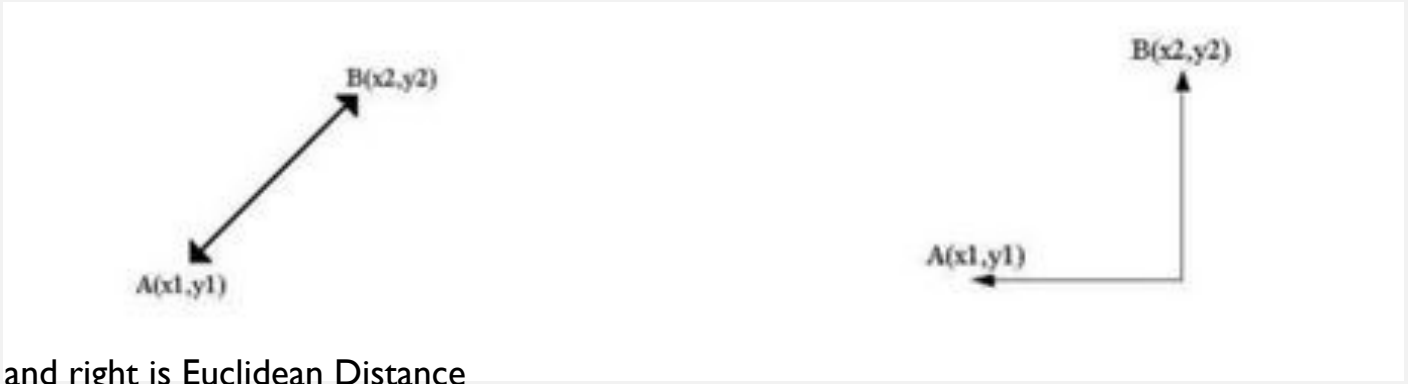
Which of the following will be true about k in k-NN in terms of variance?

A) When you increase the k the variance will increases
B) When you decrease the k the variance will increases
C) Can't say
D) None of these

QS

The following two distances(Euclidean Distance and Manhattan Distance) have been given to you which generally we used in K-NN algorithm. These distance are between two points A(x1,y1) and B(x2,Y2).

Your task is to tag the both distance by seeing the following two graphs. Which of the following option is true about below graph ?



A) Left is Manhattan Distance and right is Euclidean Distance
B) Left is Euclidean Distance and right is Manhattan Distance
C) Neither left or right are a Manhattan Distance
D) Neither left or right are a Euclidian Distance

QS

When you find noise in data which of the following option would you consider in k-NN?

A) I will increase the value of k
B) I will decrease the value of k
C) Noise can not be dependent on value of k
D) None of these

In k-NN it is very likely to overfit due to the curse of dimensionality. Which of the following option would you consider to handle such problem?

1. Dimensionality Reduction
2. Feature selection

A) 1
B) 2
C) 1 and 2
D) None of these

Q5

Below are two statements given. Which of the following will be true both statements?

1. k-NN is a memory-based approach is that the classifier immediately adapts as we collect new training data.
2. The computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario.

A company has build a kNN classifier that gets 100% accuracy on training data. When they deployed this model on client side it has been found that the model is not at all accurate. Which of the following thing might gone wrong?

Note: Model has successfully deployed and no technical issues are found at client side except the model performance

A) It is probably a overfitted model
B) It is probably a underfitted model
C) Can't say
D) None of these

QS

You have been given the following 2 statements, find which of these option is/are true in case of k-NN?

1. In case of very large value of k, we may include points from other classes into the neighborhood.
2. In case of too small value of k the algorithm is very sensitive to noise