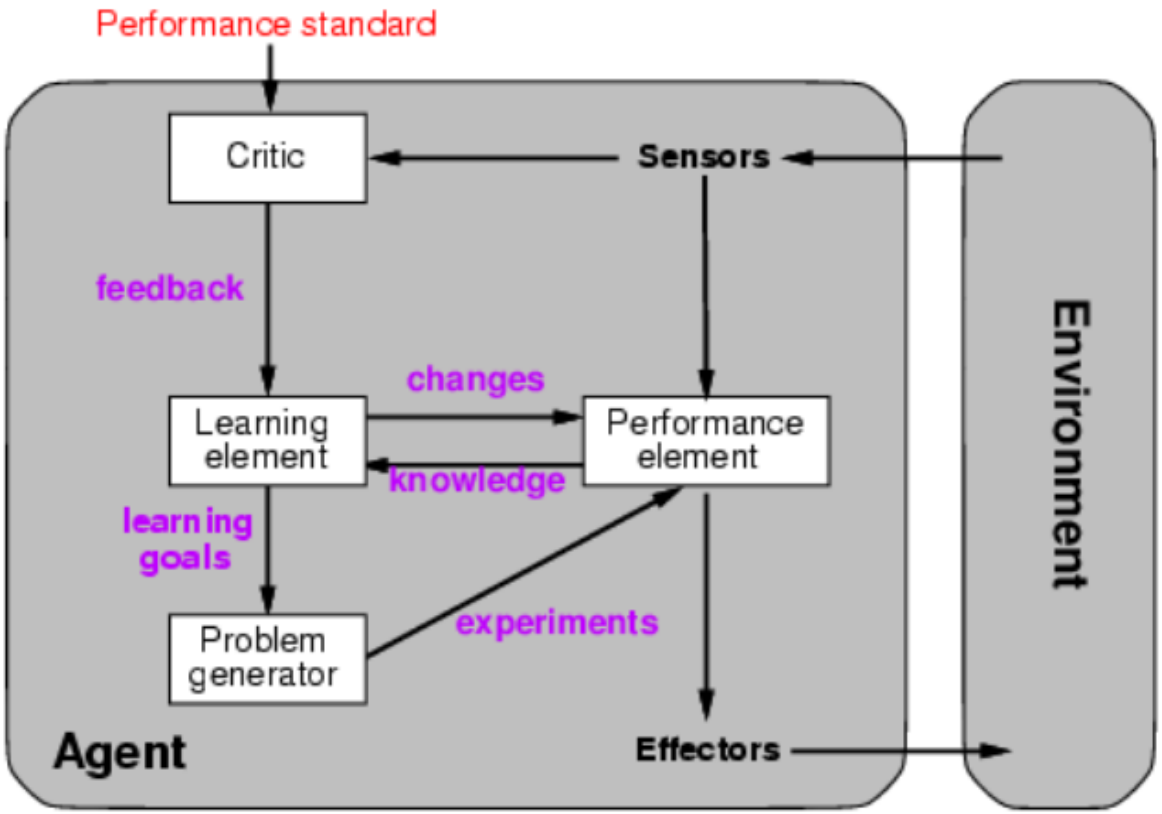
Trường Đại học Tôn Đức Thắng

K Nearest Neighbor (KNN)

Giảng viên: Tiến sĩ Bùi Thanh Hùng
Trưởng Lab Khoa học Phân tích dữ liệu và Trí tuệ nhân tạo
Giám đốc chương trình Hệ thống thông tin
Đại học Thủ Dầu Một
Email: tuhungphe@gmail.com
Website: https://sites.google.com/site/hungthanhbui1980/

Learning agents



are capable of self-improvement. They can become more competent than their initial knowledge alone might allow

Learning element

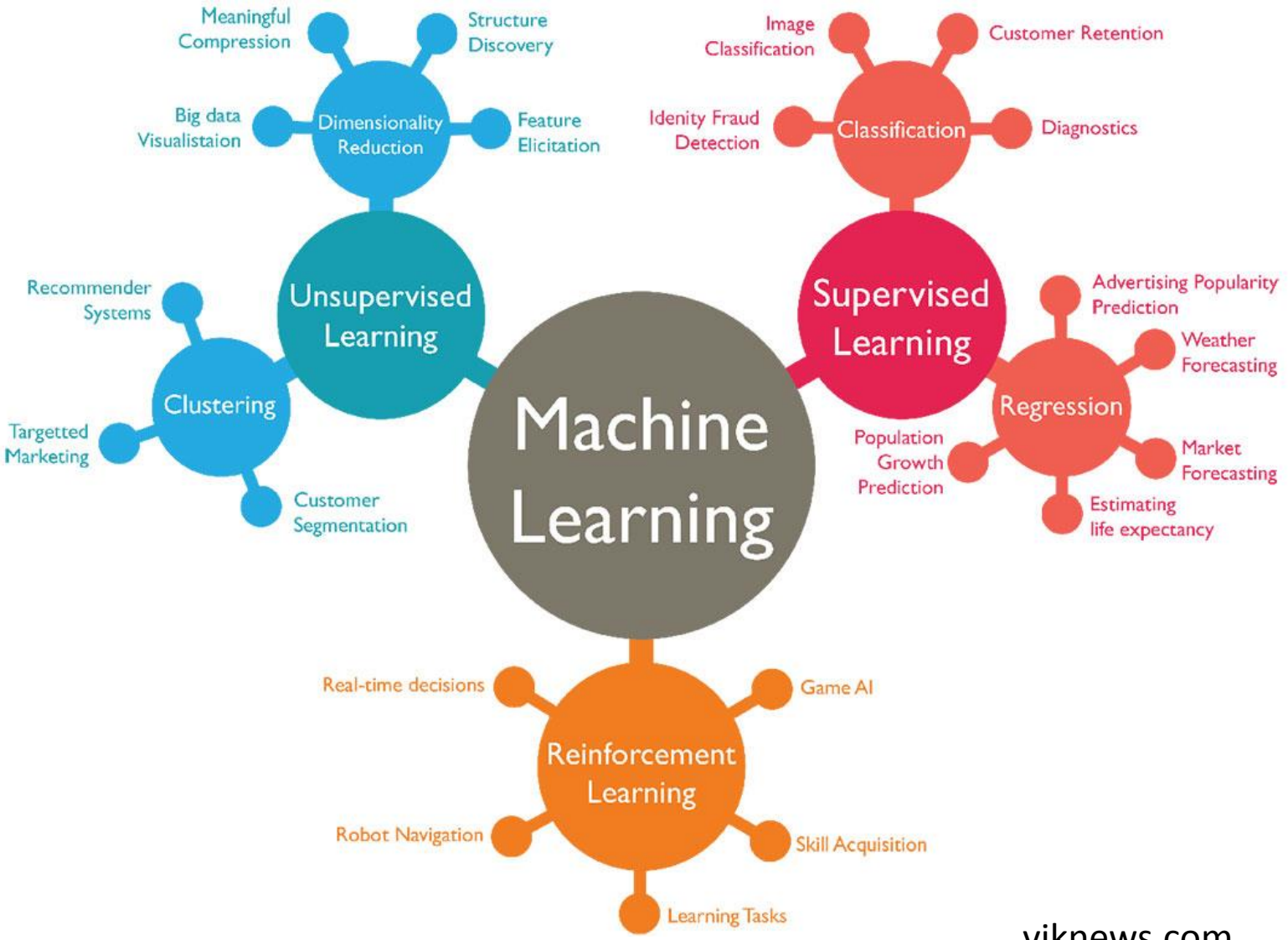
Design of a learning element is affected by

- ✓ Which components of the performance element are to be learned
- ✓ What feedback is available to learn these components
- ✓ What representation is used for the components

Type of feedback:

- ✓ Supervised learning: correct answers for each example
- ✓ Unsupervised learning: correct answers not given
- ✓ Reinforcement learning: occasional rewards

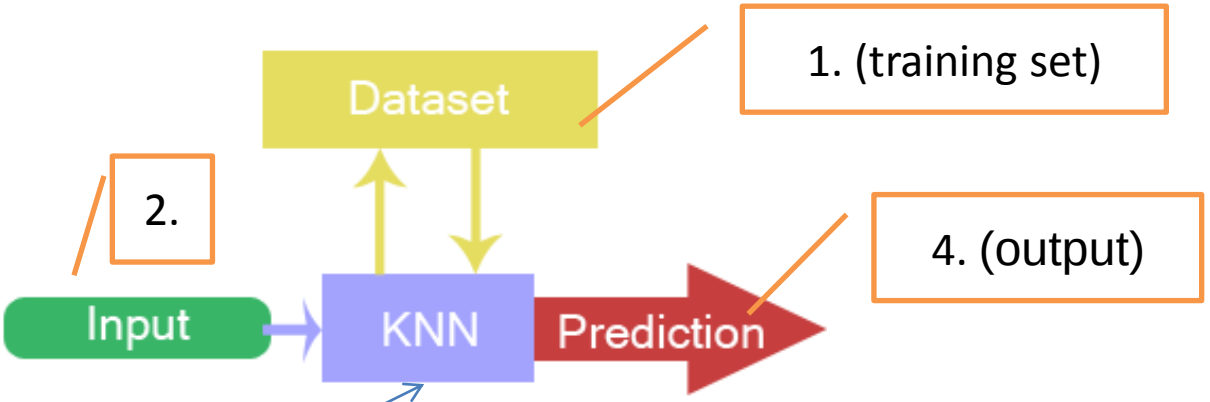
Machine Learning



viknews.com

I. KNN introduction

- a type of **supervised-learning**.
- k-NN is a type of **instance-based learning**, or **lazy learning.**
- **non-Parametric** used for classification.
- can be used for both classification and regression.



Algorithm:

1. Location (Tọa độ điểm) of input.
2. Calculate distance between input & all points in trainir set.
3. Choose **k** nearest points.
4. Design which class is the label (output) (classification) / the specific output value (regression)

3.

II. KNN in Classification

- **Important Features:**

- Data type: n-dimensional vector.
- Classes: to which one vector will be classified.
- Training Set: samples (vectors) each with input & ouput.
- Neighbors: closest points to the test data point.
- k.

- **Input (test data point):** a vector of n dimensions with property's values of an object.
- **Output:** classify test data point to 1 class from the classes in training set.

- **k=1**: assigned to the class of that single nearest neighbor
- **k>1**:

  + major voting: fair voting:
  + giving weight to the closer ones: voting w/ weight.

II. KNN in Classification

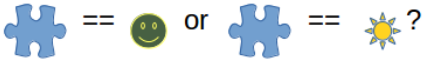
- **Calculating distance:**

- Euclidean Distance (commonly used, continuous variables);
- Hamming distance (discrete variables, such as for text classification);
- Minkowski; and many more...

- **Choose k.**
- **Weight:**

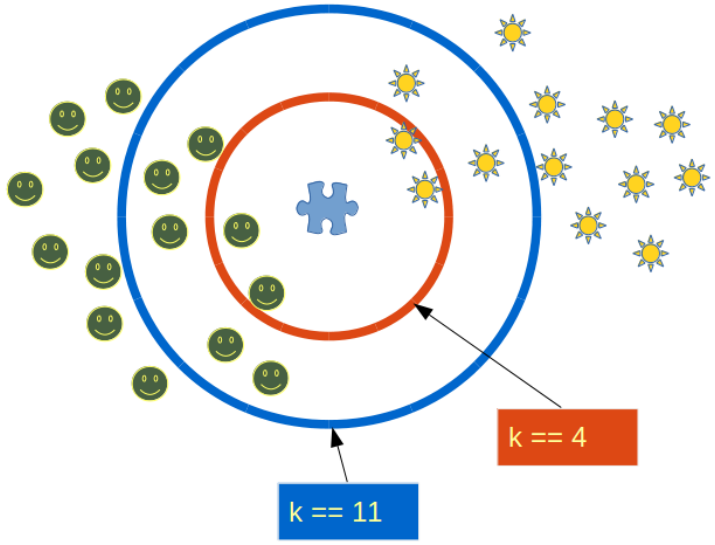
$$w_i = \exp\left(\frac{-||\mathbf{x} - \mathbf{x}_i||_2^2}{\sigma^2}\right)$$

$$W(x, p_i) = \frac{exp(-D(x, p_i))}{\sum_{i=1}^{k} exp(-D(x, p_i))}$$

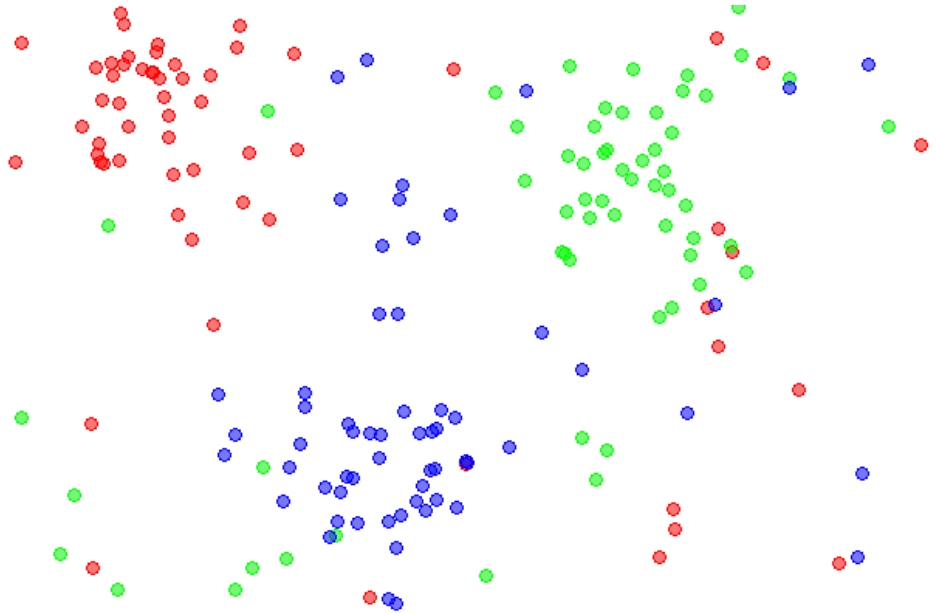


- **Applications:**

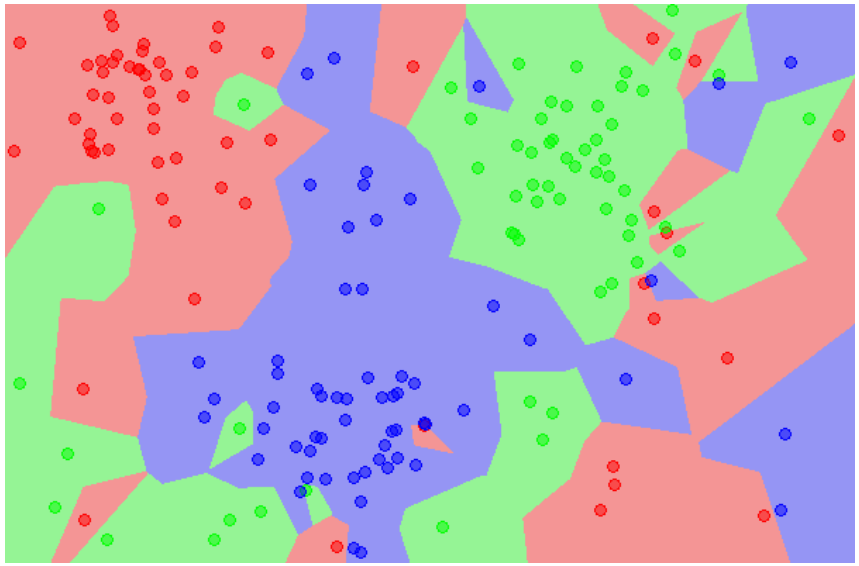
- Finance: customer classification, etc.
- Agriculture
- Text Mining
- Medicine



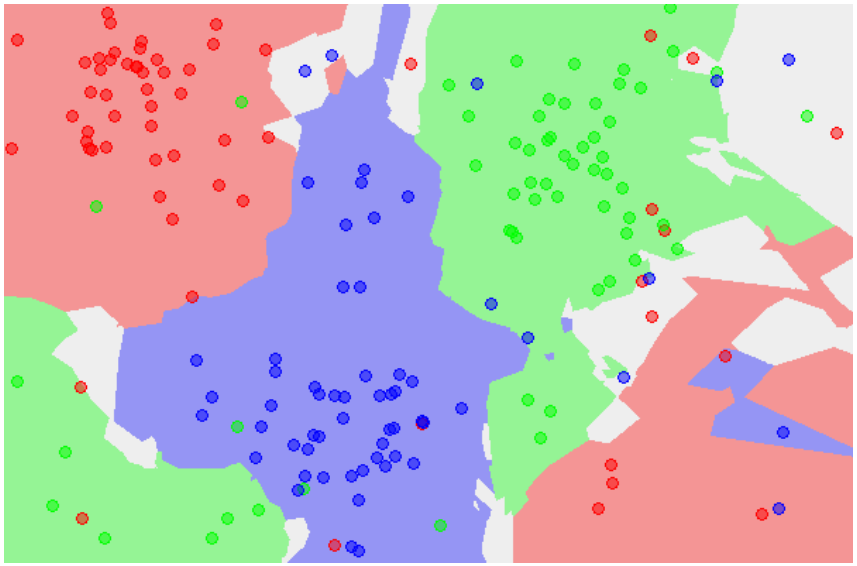
Voting example, can use fair/weight



The dataset



The 1NN classification map



The 5NN classification map

III. KNN in Regression

- **Input:** n-dimensional vector (like in classification)
- **Output:** a specific value calculated based on ***k nearest points***.

- k=1: take the output of the nearest point.

- k>1: the output are calculated based on the average of the labels (output) of the k nearest points.

- **Applications:**

- Finance: predict stock price, etc.

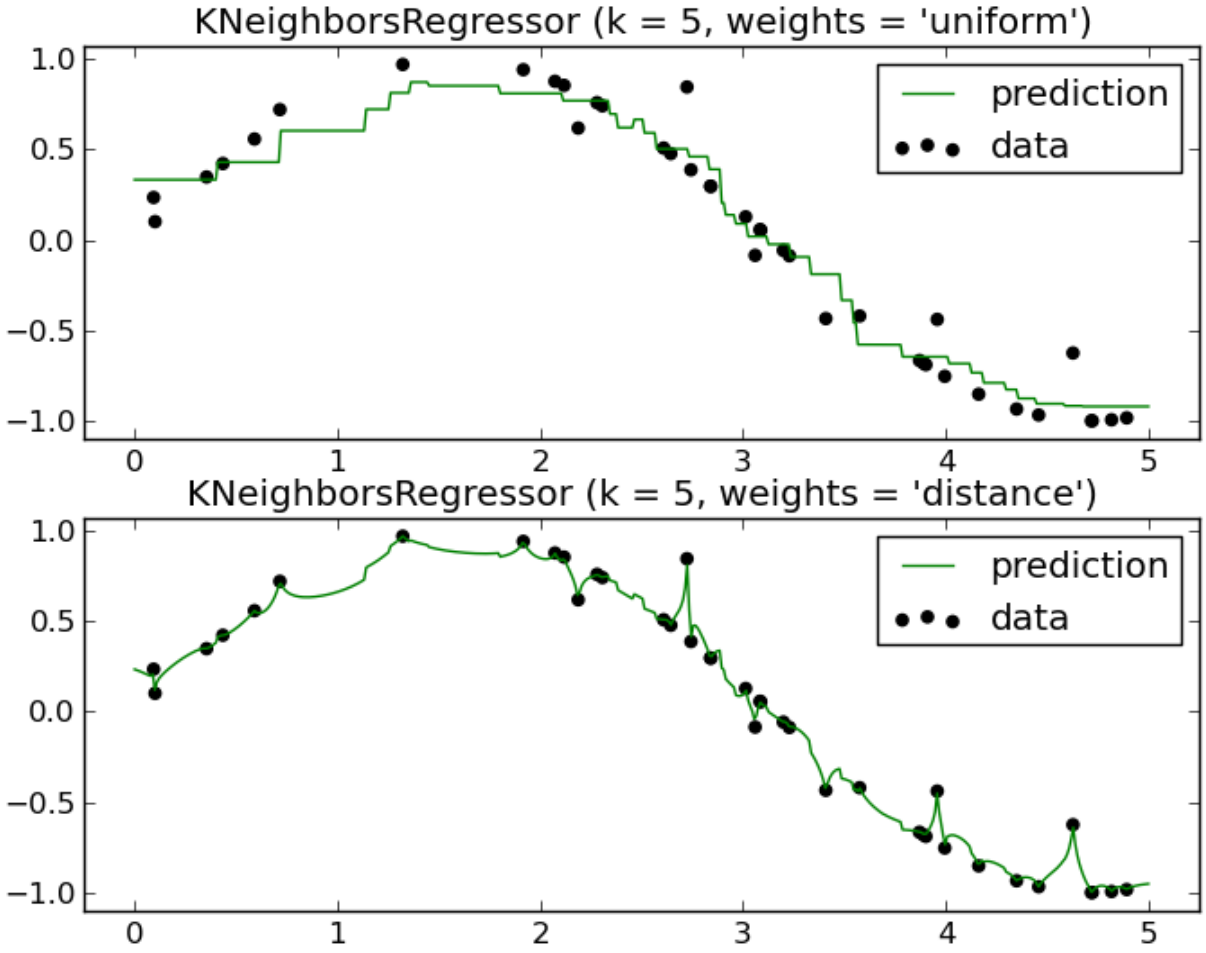
- Agriculture: weather forecast (temperature, etc).

III. KNN in Regression

- In *k*-NN regression, the *k*-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the *k* nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean or Mahalanobis distance from the query example to the labeled examples.
2. Order the labeled examples by increasing distance.
3. Find a heuristically optimal number *k* of nearest neighbors, based on RMSE. This is done using cross validation.
4. Calculate an inverse distance weighted average with the *k*-nearest multivariate neighbors.

III. KNN for Regression



Example:

Consider the following data concerning House Price Index or HPI. Age and Loan are two numerical variables (predictors) and HPI is the numerical target.

Age	Loan	House Price Index	Distance	
25	$40,000	135	102000	
35	$60,000	256	82000	
45	$80,000	231	62000	
20	$20,000	267	122000	
35	$120,000	139	22000	2
52	$18,000	150	124000	
23	$95,000	127	47000	
40	$62,000	216	80000	
60	$100,000	139	42000	3
48	$220,000	250	78000	
33	$150,000	264	8000	1
48	$142,000	?		

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

We can now use the training set to classify an unknown case (Age=33 and Loan=$150,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with HPI=264.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> HPI = 264

By having K=3, the prediction for HPI is equal to the average of HPI for the top three neighbors.

HPI = (264+139+139)/3 = 180.7

IV. Advantages & Disadantages

- **Advantages:**

- KNN does not learn any model -> training phase: O(1).

- Lazy classifiers are most useful for large datasets with few attributes.

- does not make any assumptions about the distribution of data (for training phase: O(1)).

- **Disadvantages:**

- space for storing training set.

- time to search (calculating distances) the k closest points (n comparisons, each take m steps (operations) for distance algorithm calculation).

- different distance measures, which is best?

- different weight methods, which to use?

- find the best value of k.

- noise -> effect the output (especially when k is small).

V. Improve K-Nearest Neighbor Algorithm

- A simple and effective way to remedy skewed class distributions is by implementing **weighed voting**. The class of each of the K neighbors is multiplied by a weight proportional to the inverse of the distance from that point to the given test point. This ensures that nearer neighbors contribute more to the final vote than the more distant ones.
- **Changing the distance metric** for different applications may help improve the accuracy of the algorithm. (i.e. Hamming distance for text classification)

V. Improve K-Nearest Neighbor Algorithm

- **Rescaling your data** makes the distance metric more meaningful. For instance, given 2 features `height`and `weight`, an observation such as x=[180,70]x=[180,70] will clearly skew the distance metric in favor of height. One way of fixing this is by column-wise subtracting the mean and dividing by the standard deviation. Scikit-learn's `normalize()` method can come in handy.
- **Dimensionality reduction** techniques like PCA should be executed prior to appplying KNN and help make the distance metric more meaningful.

Resources (information):

- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- https://machinelearningcoban.com
- https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/
- https://nlp.stanford.edu
- www.ijere.com
- https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

DATASET

pima-indians-diabetes.csv

pima-indians-diabetes.csv (768)

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Model Evaluation

Model Evaluation

Classification Metrics

Classification problems are perhaps the most common type of machine learning problem and as such there are a myriad of metrics that can be used to evaluate predictions for these problems.

1- Classification Accuracy.

2- Log Loss.

3- Area Under ROC Curve.

4- Confusion Matrix.

5- Classification Report.

1. Classification Accuracy

- Classification accuracy is the number of correct predictions made as a ratio of all predictions made.
- This is the most common evaluation metric for classification problems, it is also the most misused. It is really only suitable when there are an equal number of observations in each class (which is rarely the case) and that all predictions and prediction errors are equally important, which is often not the case.

2. Log Loss

- Logistic loss (or log loss) is a performance metric for evaluating the predictions of probabilities of membership to a given class.
- The scalar probability between 0 and 1 can be seen as a measure of confidence for a prediction by an algorithm. Predictions that are correct or incorrect are rewarded or punished proportionally to the confidence of the prediction.

3. Area Under ROC Curve

- Area Under ROC Curve (or ROC AUC for short) is a performance metric for binary classification problems.
- The AUC represents a model's ability to discriminate between positive and negative classes. An area of 1.0 represents a model that made all predictions perfectly. An area of 0.5 represents a model as good as random.
- A ROC Curve is a plot of the true positive rate and the false positive rate for a given set of probability predictions at different thresholds used to map the probabilities to class labels. The area under the curve is then the approximate integral under the ROC Curve.

4. Confusion Matrix

- The confusion matrix is a handy presentation of the accuracy of a model with two or more classes.
- The table presents predictions on the x-axis and accuracy outcomes on the y-axis. The cells of the table are the number of predictions made by a machine learning algorithm.
- For example, a machine learning algorithm can predict 0 or 1 and each prediction may actually have been a 0 or 1. Predictions for 0 that were actually 0 appear in the cell for prediction=0 and actual=0, whereas predictions for 0 that were actually 1 appear in the cell for prediction = 0 and actual=1. And so on.

Confusion matrix

- For each pair of classes $<c_1,c_2>$ how many documents from c_1 were incorrectly assigned to c_2?
  - $c_{3,2}$: 90 wheat documents incorrectly assigned to poultry

Docs in test set	Assigned UK	Assigned poultry	Assigned wheat	Assigned coffee	Assigned interest	Assigned trade
True UK	95	1	13	0	1	0
True poultry	0	1	0	0	0	0
True wheat	10	90	0	1	0	0
True coffee	0	0	0	34	3	7
True interest	-	1	2	13	26	5
True trade	0	0	2	14	5	10

Per class evaluation measures

Recall:

Fraction of docs in class *i* classified correctly:

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

Precision:

Fraction of docs assigned class *i* that are actually about class *i*:

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

Accuracy: (1 - error rate)

Fraction of docs classified correctly:

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- **Macroaveraging**: Compute performance for each class, then average.
- **Microaveraging**: Collect decisions for all classes, compute contingency table, evaluate.

Micro- vs. Macro-Averaging: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

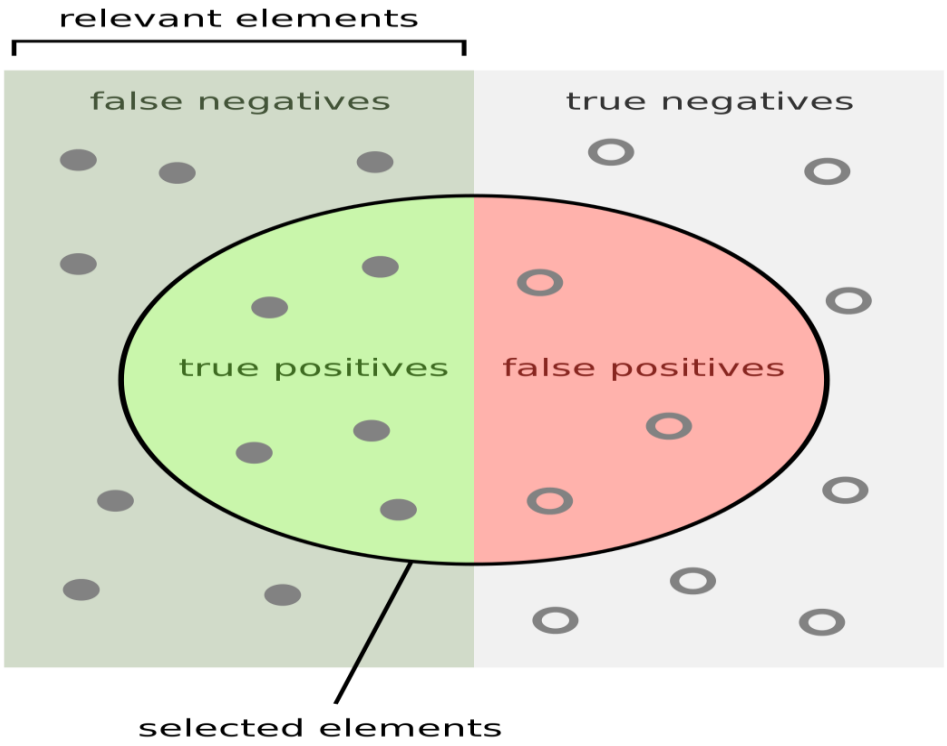
	Truth: yes	Truth : no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Microaveraged score is dominated by score on common classes

5. Classification Report



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

$$F = 2PR/(P+R)$$

Precision and recall

- **Precision**: % of selected items that are correct
  Recall: % of correct items that are selected

	correct	not correct
selected	tp	fp
not selected	fn	tn

A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- The harmonic mean is a very conservative average
- People usually use balanced F1 measure
  - i.e., with $\beta = 1$ (that is, $\alpha = ½$):
  - $F = 2PR/(P+R)$

Regression Metrics

- Mean Absolute Error.
- Mean Squared Error.
- R^2.

1. Mean Absolute Error

- The Mean Absolute Error (or MAE) is the average of the absolute differences between predictions and actual values. It gives an idea of how wrong the predictions were.
- The measure gives an idea of the magnitude of the error, but no idea of the direction (e.g. over or under predicting).

2. Mean Squared Error

- The Mean Squared Error (or MSE) is much like the mean absolute error in that it provides a gross idea of the magnitude of error.
- Taking the square root of the mean squared error converts the units back to the original units of the output variable and can be meaningful for description and presentation. This is called the Root Mean Squared Error (or RMSE).

3. R^2 Metric

- The R^2 (or R Squared) metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination.
- This is a value between 0 and 1 for no-fit and perfect fit respectively.

Practice

1- Using Pima dataset

2- Divide dataset to 7:3 ratio

3- KNN Algorithms

4- Evaluate by

- Accuracy
- Confusion Matrix
- F1 score
- 5- Folds