Lecture#4

Numerical Regression

Numerical Regression

- A common type of machine learning task is when we have numerical features…
- … and want to predict a continuous variable (number)
- Typical tasks are to estimate:
  - The price of a new product based on existing products
  - Stock market predictions
  - …

Price Models

- One algorithm that is very effective for numerical regression is *K-nearest neighbor*
- In K-nearest Neighbor (KNN), all attributes must be numeric
- Nominal attributes can be translated to numeric in a pre-processing step
- If we have two categories, *harddrive* and *SSD*, we can use 0 for *harddrive* and 1 for *SSD*

Important variables

- When building price models it is important to determine which variables that affect the price
- Example: “is processor speed a good indicator for laptop price?”
- Variables such as RAM and screen size likely has large impact on the price
- If the laptop comes bundled with free software or not probably have low impact on the price

Example dataset

- We will use a simple dataset where the task is to predict the price of a TV based on:
  - Screen size in inches
  - Screen type {Full HD, 4K, OLED}
  - Customer rating
- The screen type variable is nominal and must be translated to a numerical variable
- The following translation is used:
  - 1: Full HD
  - 2: 4K
  - 3: OLED

Example dataset

ScreenSize	ScreenType	Rating	Price
55	2	3.9	5495
55	2	3.8	5990
50	1	4.5	6495
55	2	4.4	6990
58	1	4.0	7490
50	2	4.4	7590
55	2	4.2	7990
55	1	4.5	7990
55	1	4.3	7990
55	2	4.1	7995
…	…	…	…
55	3	4.7	16790
55	3	4.8	23990
55	3	4.3	24790
55	3	4.0	29990

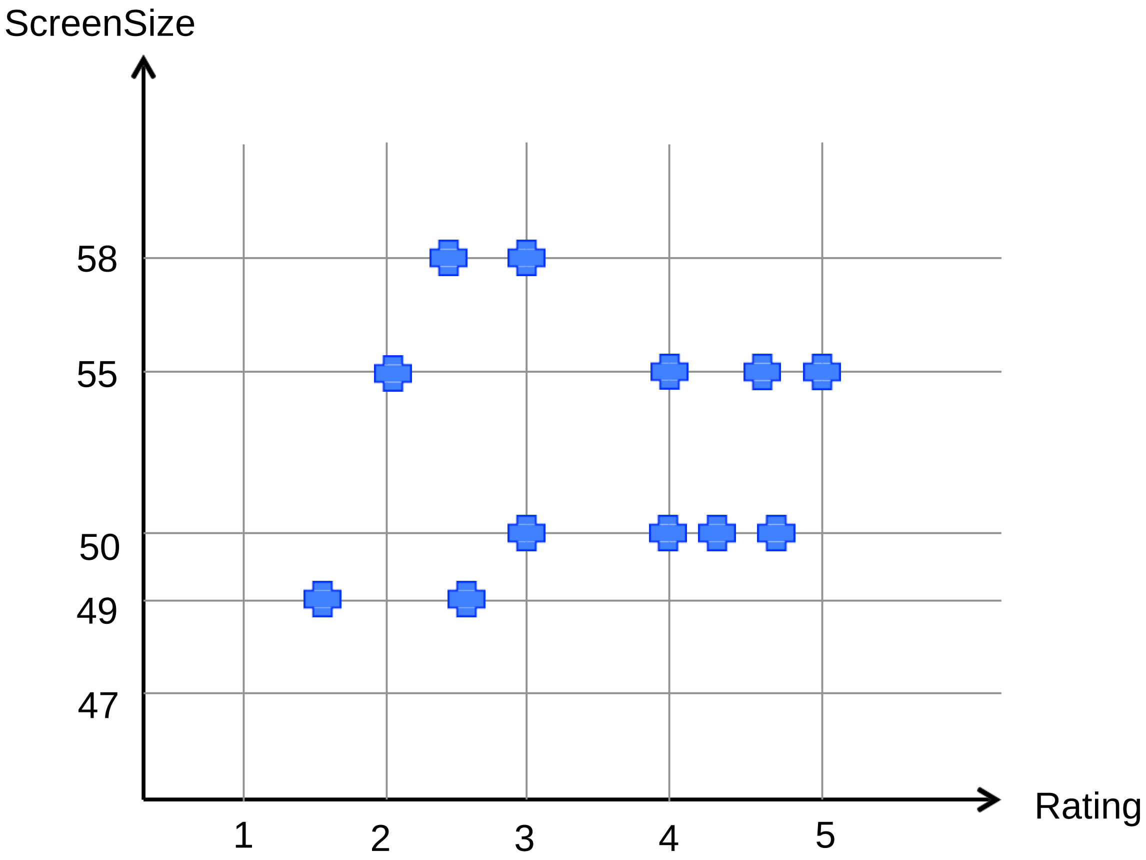
K-nearest neighbor

- What would you do if you manually shall guess the price of a new TV?
- A common approach is to find TVs similar to the one you shall guess the price for,
- , …then take an average of the prices of the similar TVs
- Now you have a reasonably good guess!
- This is exactly how KNN works!

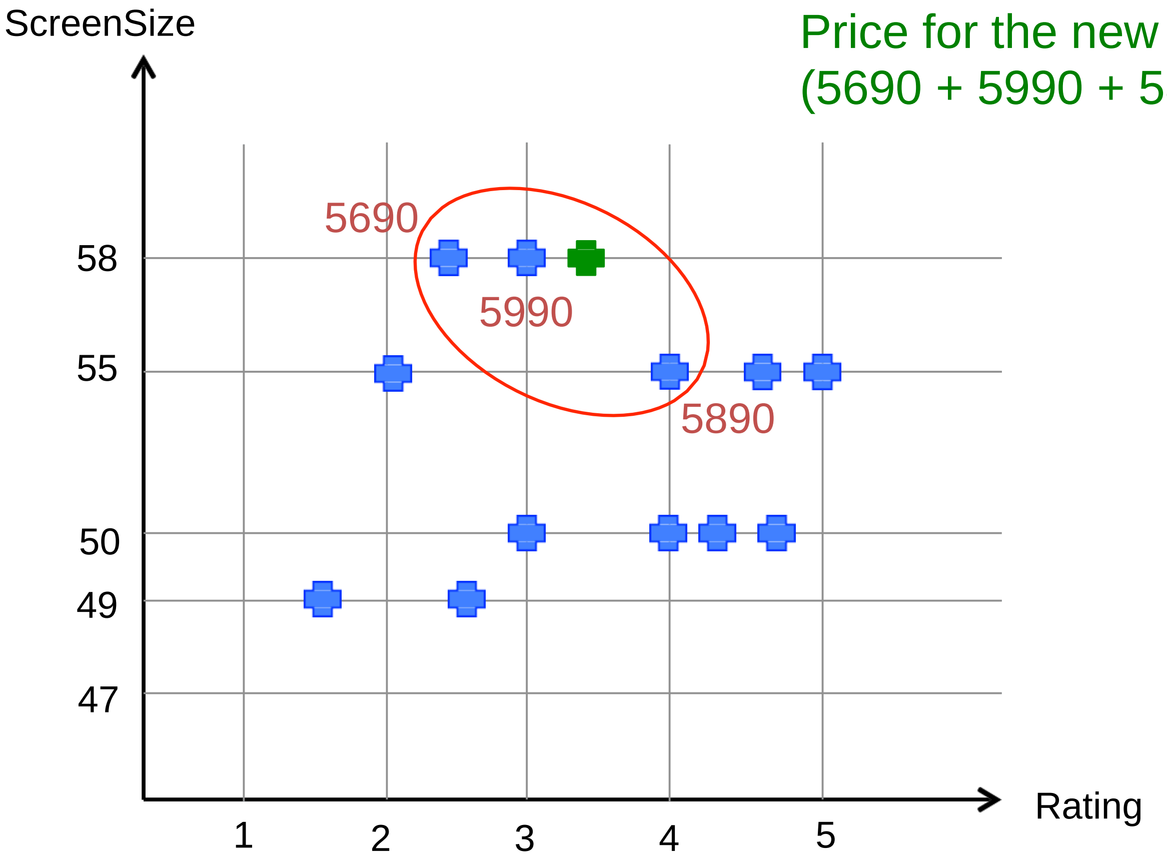
K-nearest neighbor

- In KNN, you search for the *k* most similar examples in the training dataset
- A k value between 3 and 5 is quite common, but you can experiment with other k values
- The predicted price is then the average price for the k examples
- Using k = 1 only picks the nearest example, but that is rarely useful
- Some items are cheaper or more expensive than they should be, and using a k larger than 1 reduces the effect of this

Example data



Example classification, k = 3



Price for the new example is:
(5690 + 5990 + 5890) / 3 = 5857

Similarity

- In KNN, Euclidean distance is commonly used to calculate the distance between examples:

$$distance = \sqrt{(a_0 - b_0)^2 + (a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2}$$

Training and Classification

- No actual training is done in KNN
- It simply stores the training data in the main memory or a data base
- All computation is done when classifying an example
- The drawback is that classification is slower compared to other algorithms that train a prediction model
- An advantage is that we can add more training data when the system is up and running, which many other algorithms don’t support

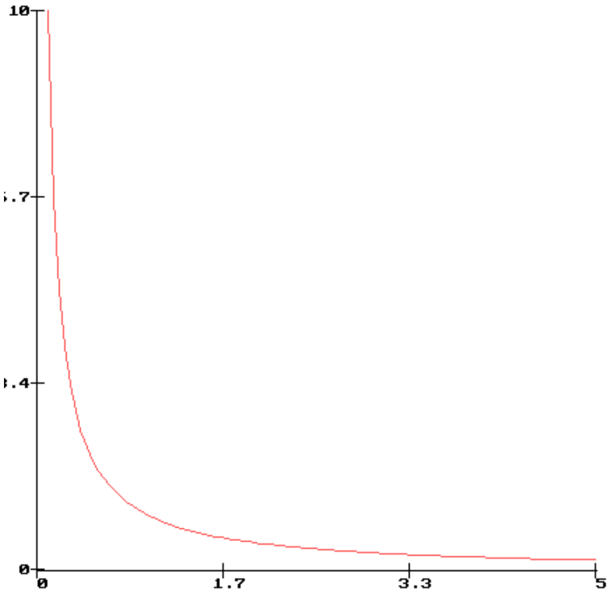
Introducing weights

- So far, the algorithm calculates the average of the k nearest neighbors
- Depending on how the data is distributed, some neighbors can be very far from the example while others are very close
- To account for this, we can introduce a weight for each neighbor based on the distance
- A weighted average price is then calculated
- We will look at two ways of weighting neighbors:

Inverse Function

- The inverse function returns a value of 1 divided by the distance
- The problem is that very small distances can lead to very high or infinite weights due to how the inverse function works
- To get around this, we add a small number to the distance before inverting it:

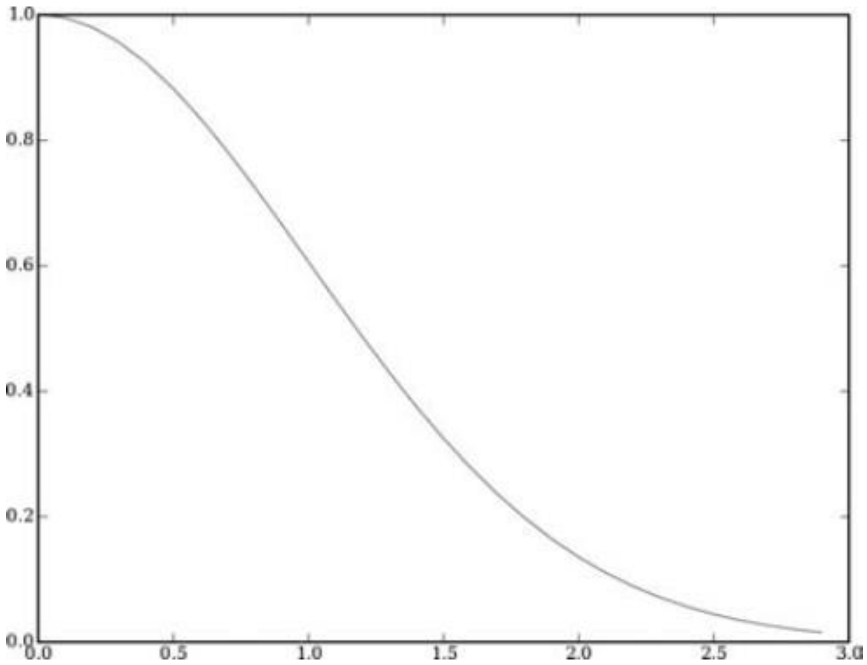
$$inverse = \frac{1}{distance + 0.1}$$



Gaussian Function

- The second way is to use the *Gaussian function*, also known as a *bell curve*
- The weight is 1 when the distance is 0, and gradually declines as the distance increased.
- The weight will however never fall to 0, so we will never have problems with 0 weights
- The Gaussian function looks like this:

Gaussian Function



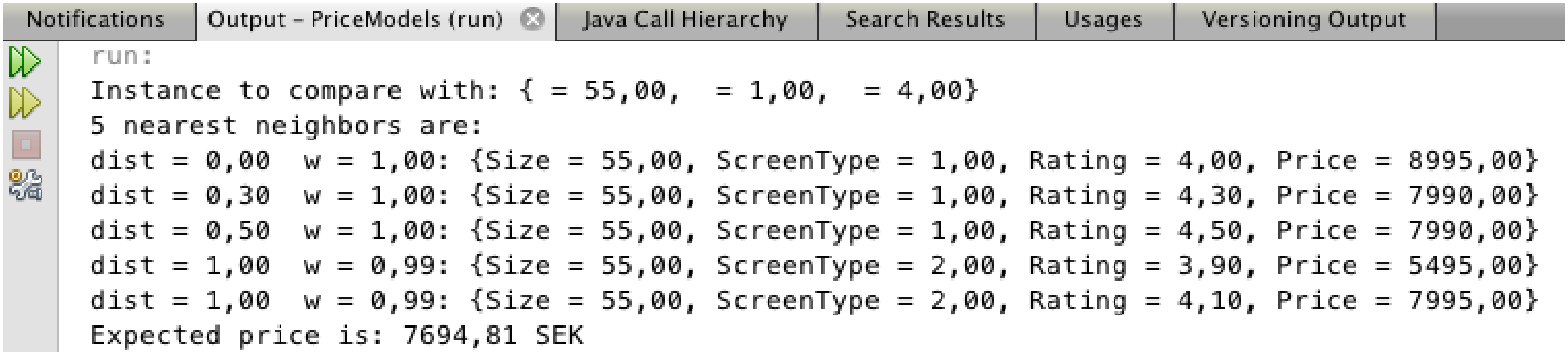
$$gaussianW = e^{-\frac{distance^2}{2 \cdot \sigma^2}}$$

Adding weights to KNN

- The price for each example is multiplied by the weight
  - Using the distance between the training examples and the example we want to classify
- The average price is then divided by the sum of the weights

Testing it

- We use a KNN with k = 5 and the TV dataset (attributes ScreenSize, ScreenType and Rating).
- The task is to find a price for a new TV where the attributes are known:
  - {ScreenSize = 55, ScreenType = 1, Rating = 4.0}
- Result:



```
run:
Instance to compare with: { = 55,00,  = 1,00,  = 4,00}
5 nearest neighbors are:
dist = 0,00  w = 1,00: {Size = 55,00, ScreenType = 1,00, Rating = 4,00, Price = 8995,00}
dist = 0,30  w = 1,00: {Size = 55,00, ScreenType = 1,00, Rating = 4,30, Price = 7990,00}
dist = 0,50  w = 1,00: {Size = 55,00, ScreenType = 1,00, Rating = 4,50, Price = 7990,00}
dist = 1,00  w = 0,99: {Size = 55,00, ScreenType = 2,00, Rating = 3,90, Price = 5495,00}
dist = 1,00  w = 0,99: {Size = 55,00, ScreenType = 2,00, Rating = 4,10, Price = 7995,00}
Expected price is: 7694,81 SEK
```

Laptop dataset

- A second dataset has been generated from laptops sold at Elgiganten.se
- The following attributes are used:
  - ScreenSize (13.3”)
  - ProcessorSpeed (2.10)
  - Cores (2)
  - RAM (4)
  - StorageType (0 for harddrive, 1 for SSD)
  - StorageSize (256)

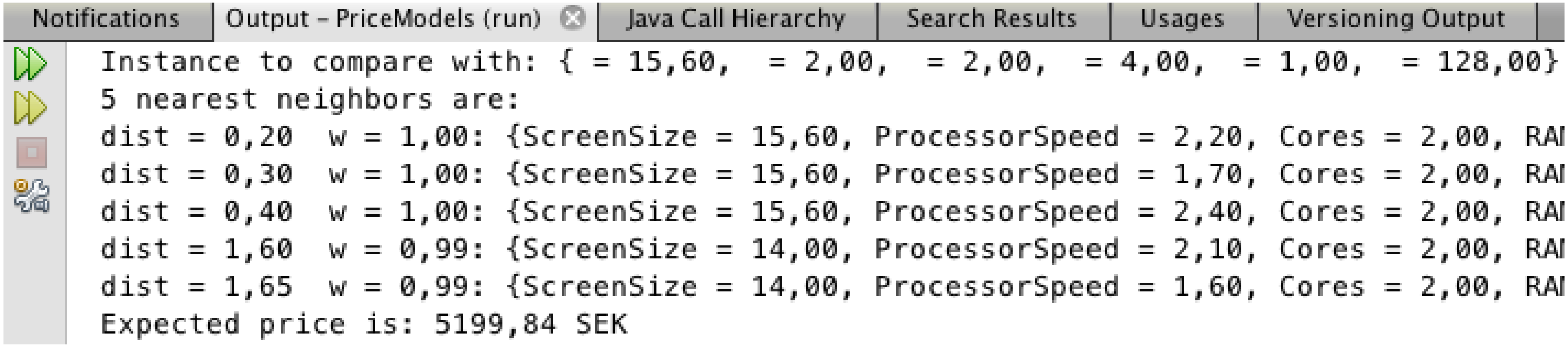
Laptop dataset

- A part of the laptop dataset looks like this:

Screen Size	Processor Speed	Cores	RAM	Storage Type	Storage Size	Price
14	1.60	2	4	1	128	3495
15.6	1.60	2	4	0	500	3495
10.1	1.44	4	2	1	64	3695
15.6	2.10	2	8	1	128	3695
15.6	1.70	2	4	1	128	3995
15.6	2.00	4	4	1	128	3995
15.6	2.40	2	8	1	256	4490
13.3	1.90	2	4	0	500	5495
14	2.40	2	4	1	356	5996

Testing it

- Task: find the price of the example:
  - {15.6, 2, 2, 4, 1, 128}
- Result:



```
Instance to compare with: { = 15,60,  = 2,00,  = 2,00,  = 4,00,  = 1,00,  = 128,00}
5 nearest neighbors are:
dist = 0,20  w = 1,00: {ScreenSize = 15,60, ProcessorSpeed = 2,20, Cores = 2,00, RAI
dist = 0,30  w = 1,00: {ScreenSize = 15,60, ProcessorSpeed = 1,70, Cores = 2,00, RAI
dist = 0,40  w = 1,00: {ScreenSize = 15,60, ProcessorSpeed = 2,40, Cores = 2,00, RAI
dist = 1,60  w = 0,99: {ScreenSize = 14,00, ProcessorSpeed = 2,10, Cores = 2,00, RAI
dist = 1,65  w = 0,99: {ScreenSize = 14,00, ProcessorSpeed = 1,60, Cores = 2,00, RAI
Expected price is: 5199,84 SEK
```

Heterogeneous Variables

- Consider the TV dataset
- The values for screen size (47-58”) are much higher than the values for rating (1-5)
- Therefore, the screen size attribute has much higher impact on the distance than the rating attribute
- To get around this we can normalize attributes to be of similar range, for example between 0 and 1

When to use KNN

- The major drawback of KNN is that classification is computationally expensive
- It also has high memory requirements since no model is built from the training data
- An advantage is that new training examples can be added after the initial training
- It is also easy to interpret how the algorithm makes its decisions
- KNNs are best used when you have numerical inputs and the slow classification time is not a problem

Classification Tasks

- KNN can also be used if we have categories
- We then return the most frequent category among the k nearest neighbors
- The attributes must be numeric, or translated to numbers
  - A = 1, B = 2, C = 3, …