Introduction to Supervised Learning

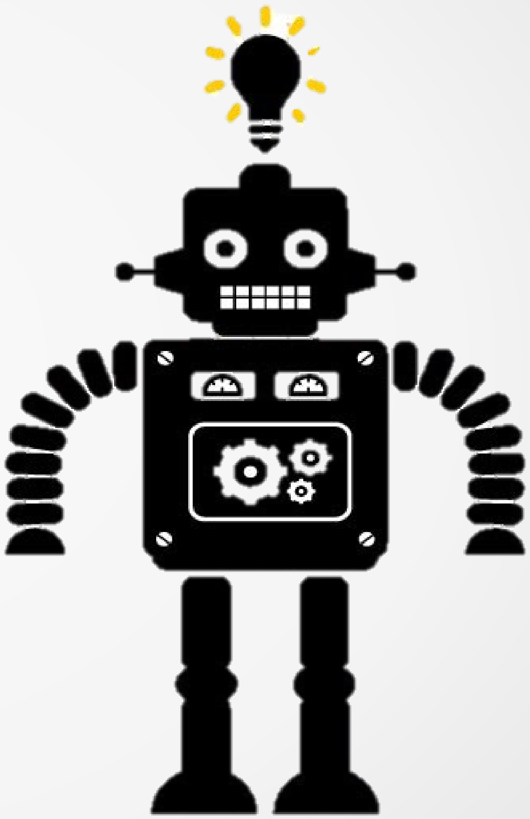
Learning Objectives

After completing this lecture, you will be able to:-

- Explain the difference between supervised and unsupervised learning
- Explain the difference between between regression and classification
- Describe the decisions to be made in order to formulate a Supervised Learning problem
- Implement and use K-nearest-neighbours
- Implement and use Linear Regression

What is Machine Learning?

- Machine Learning allows computers to learn and infer from data
- This already affects our daily lives, through things like spam filtering, web search, movie/purchase recommendations, web advertising, social networks, fraud detection, and many more



Types of Machine Learning

Differing methodology

- Supervised (data points have known outcome)
- Unsupervised (data points have unknown outcome)

Types of Supervised Learning

- Regression (outcome is continuous/numerical)
- Classification (outcome is a category)

Machine Learning Vocabulary

- Target: predicted category or data value (column)
- Feature: properties of data used for prediction (column)
- Example: a single data-point within the data (row)
- Label: the target value for a single data-point

Machine Learning Vocabulary

Features

Example

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

Target

Label

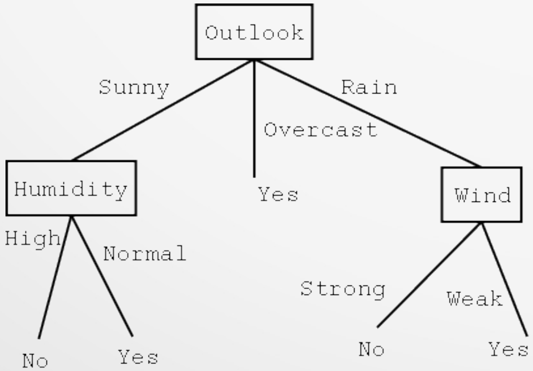
Formulating a Supervised Learning Problem

- Collect a labeled dataset (features and target labels)
- Choose the model
- Choose an evaluation metric
  - What we use to evaluate the performance
- Choose an optimization method*
  - What we use to find the model configuration the gives the best performance

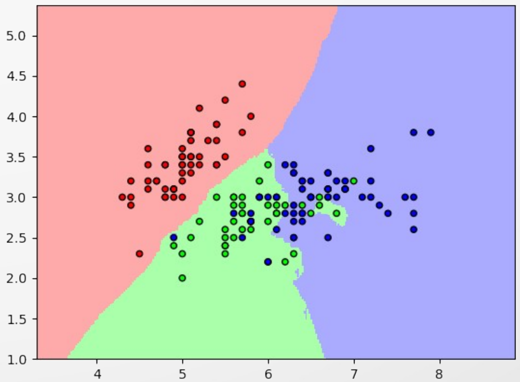
* - There are almost always standard methods for different models/metrics

Which Model?

- There are many models that represent the problem and make decisions in different ways each with their own advantages and disadvantages.
- A decision tree makes predictions by asking a series of yes/no questions.
- Nearest neighbor makes predictions by having the most similar examples vote.



Decision tree



Nearest neighbors

Which Model?

Some considerations for choosing a model

- Time needed for training
- Speed in making predictions
- Amount of data needed
- Type of data
- Problem complexity
- Ability to solve a complex problem
- Tendency to overcomplicate a simple one

What Evaluation Metric?

There are many metrics available* to measure performance, such as:-

- **Accuracy**: how well predictions match true values
- **Mean Squared Error**: Average squared distance between prediction and true value
- **AUC**: Area under ROC (receiver operating characteristic) curve

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

* - The wrong metric can be very misleading

What Evaluation Metric?

Consider using **accuracy** to evaluate a spam filter

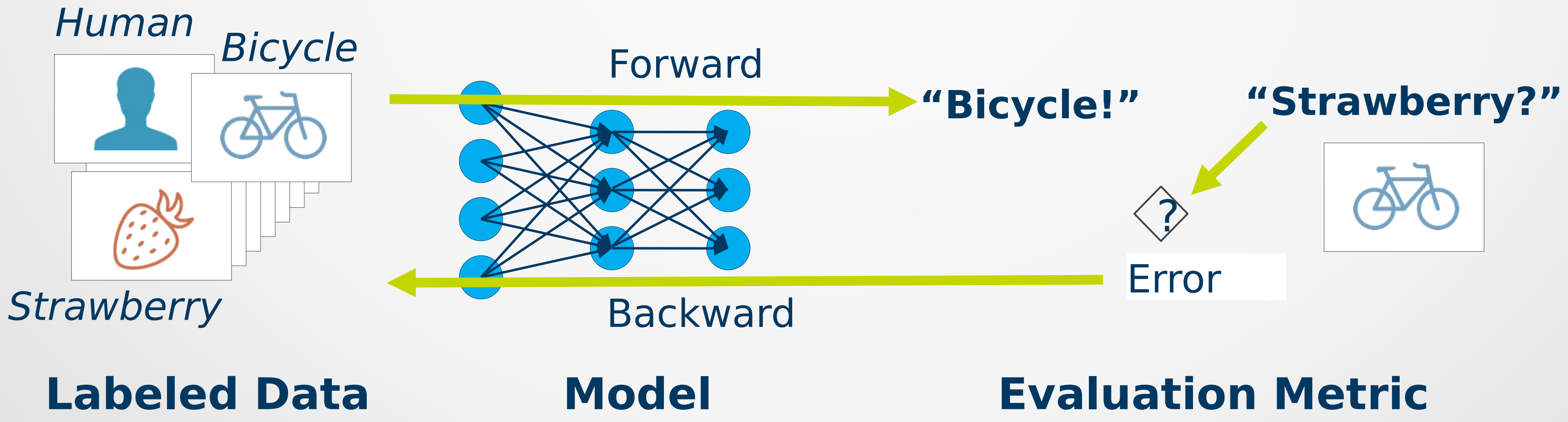
- If 99 out of every 100 emails are actually spam, then we can obtain 99% accuracy by simply classifying ALL emails as spam
- This would also classify important real email as spam, something which the accuracy metric obscures

Accuracy target

What Optimization Method?

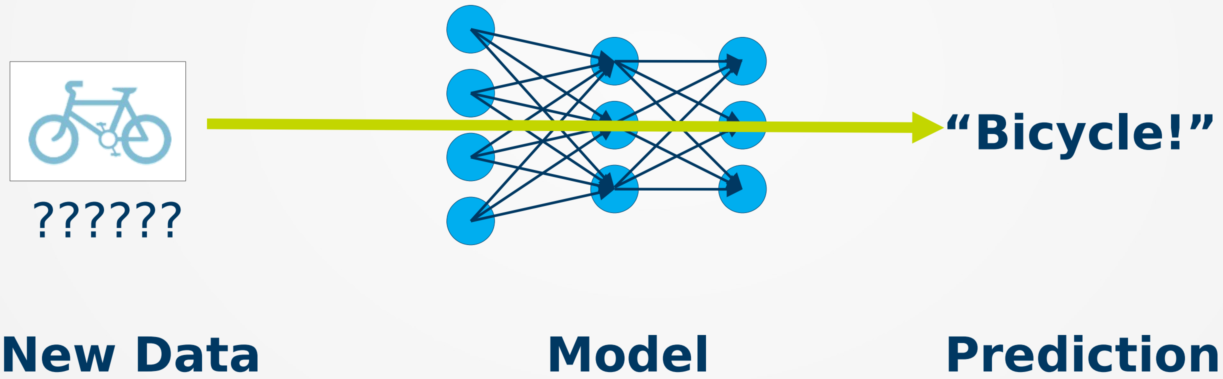
The process of optimizing a model such that it correctly predicts (models) the given data is called **training**.

- **Training Data**: The dataset used to train the model
- **Optimization**: Configuring the model to maximize performance



How To Deploy?

Once a model is properly trained, we can then provide new examples for predictions – this process is called **inference**.



Overview of Supervised Learning

- **Training:** Train a model to fit known data

Data with answers (features and labels) + Model $\xrightarrow{\text{Fit}}$ Trained Model

- **Inference:** Feed unseen/new data into trained model to make predictions

Data without answers (features only) + Trained Model $\xrightarrow{\text{Predict}}$ Predicted Answer

Classification Example

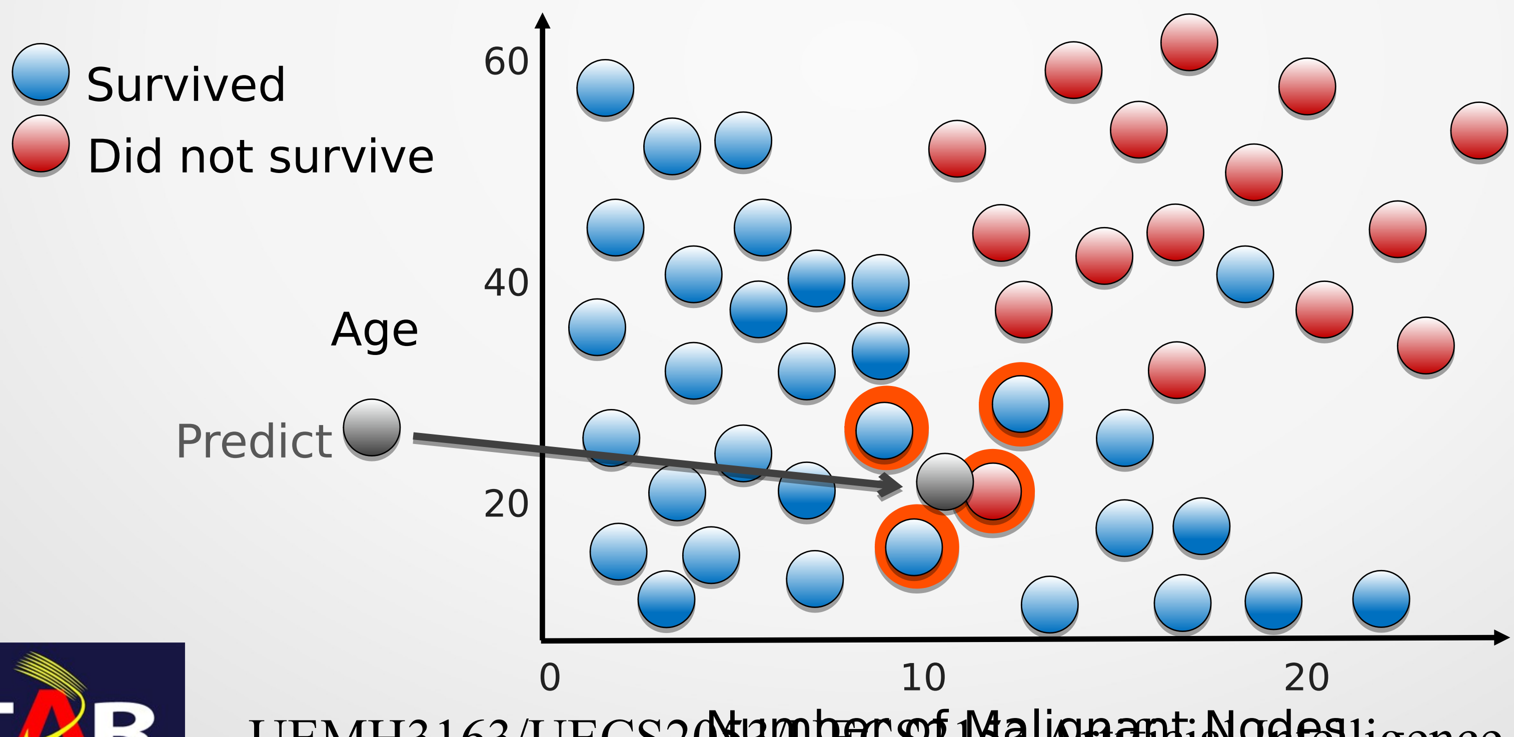
Suppose a flower shop wants to guess a customer's preferred purchase based on similarity to most recent purchase….

What Is Needed For Classification?

- Model data
  - Contains features which can be objectively quantified
  - Known labels (target)
- Method to measure similarity

K Nearest Neighbors Classification

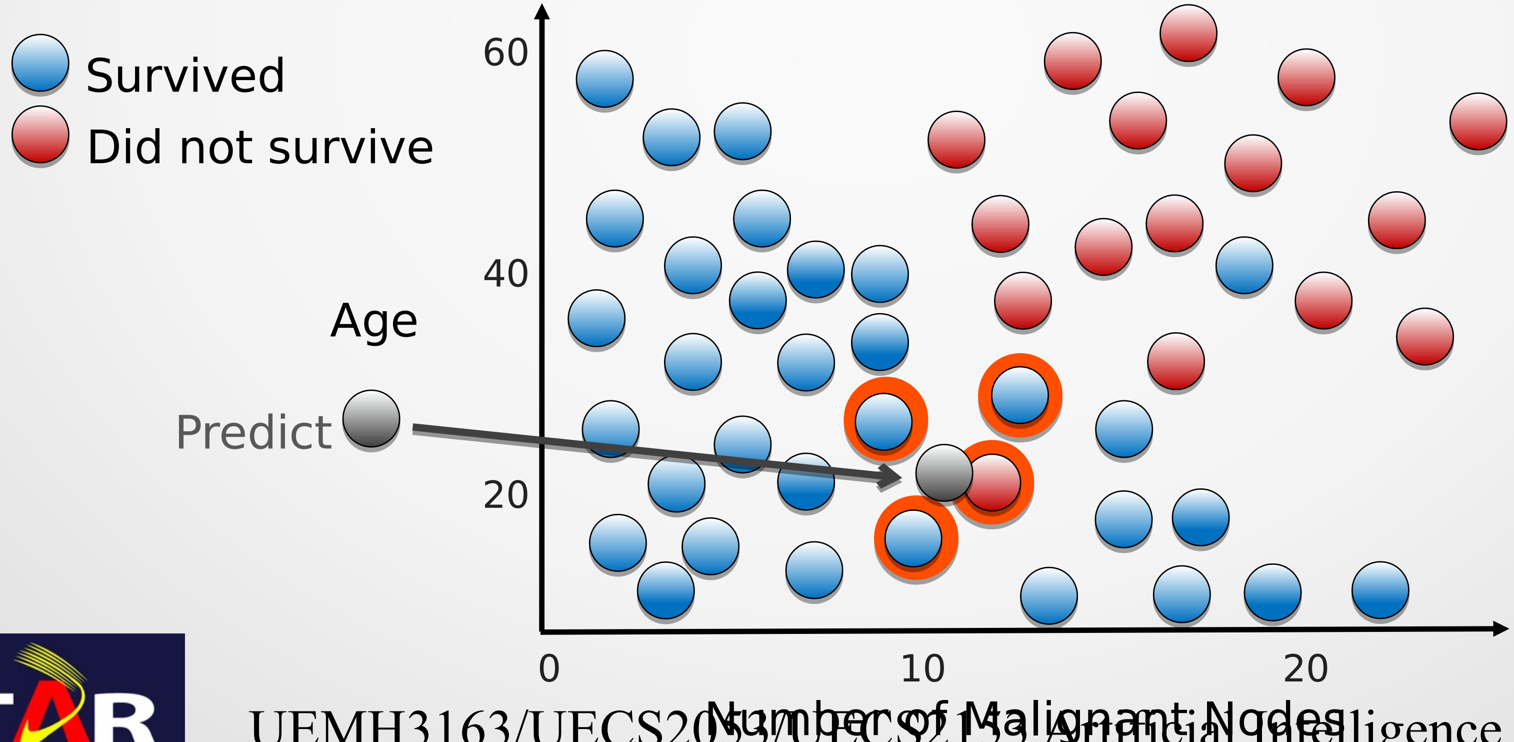
Neighbor Count (K):	1	2	3	4
	0	1	2	3
	1	1	1	1



K Nearest Neighbors Classification

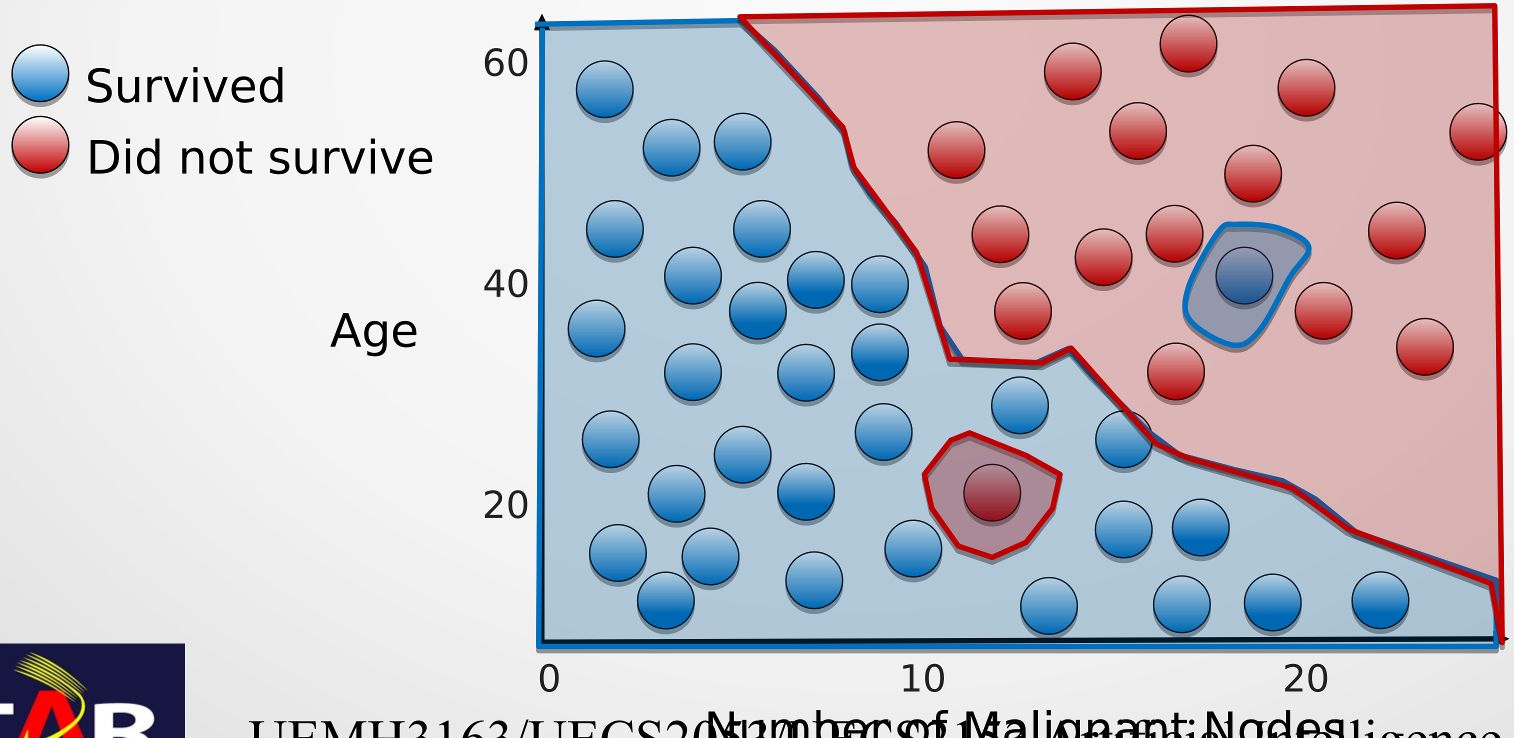
In order to optimize a KNN model we need

- Correct value for ‘K’
- Metric to measure closeness of neighbors



K Nearest Neighbors Classification

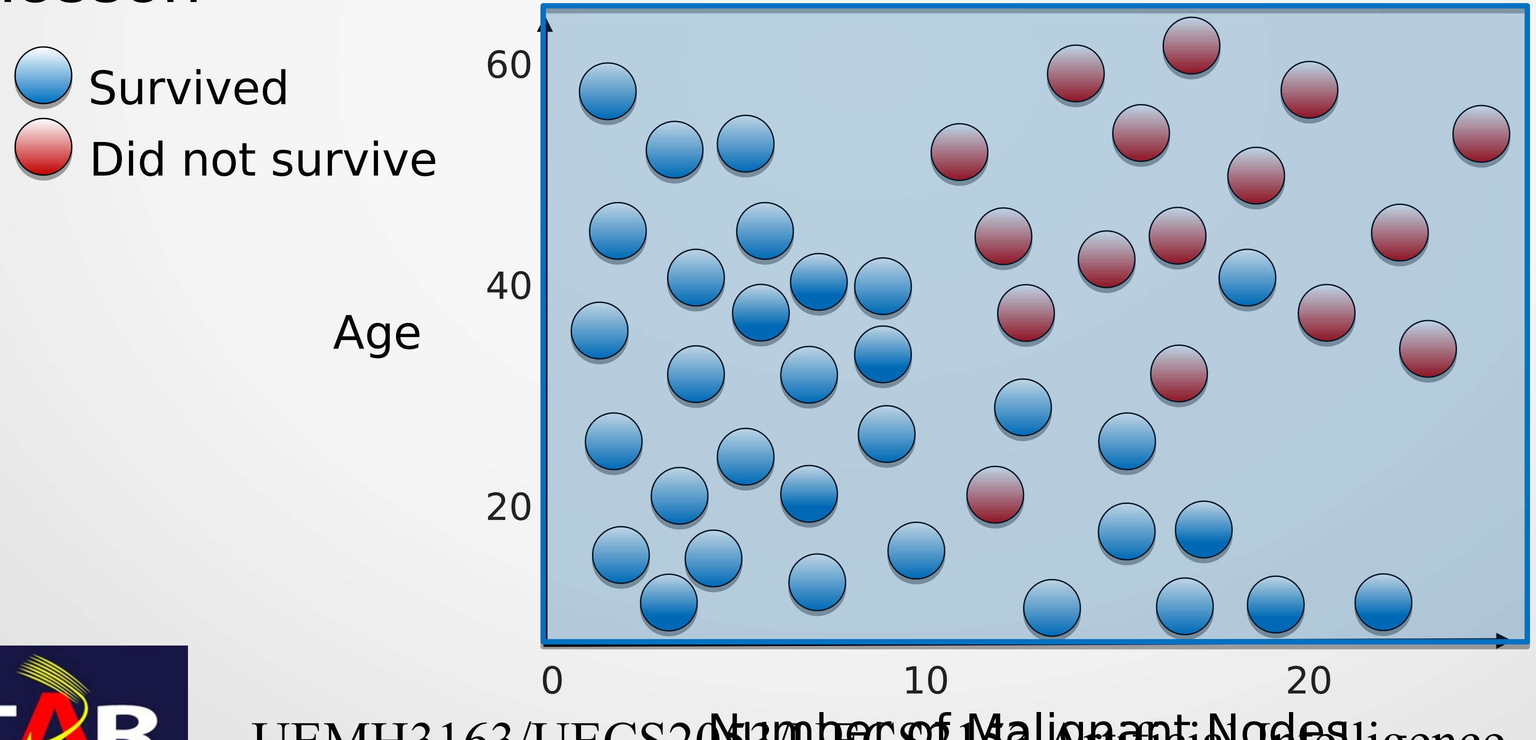
This is the K Nearest Neighbors decision boundary when K = 1



K Nearest Neighbors Classification

This is the K Nearest Neighbors decision boundary when K = all

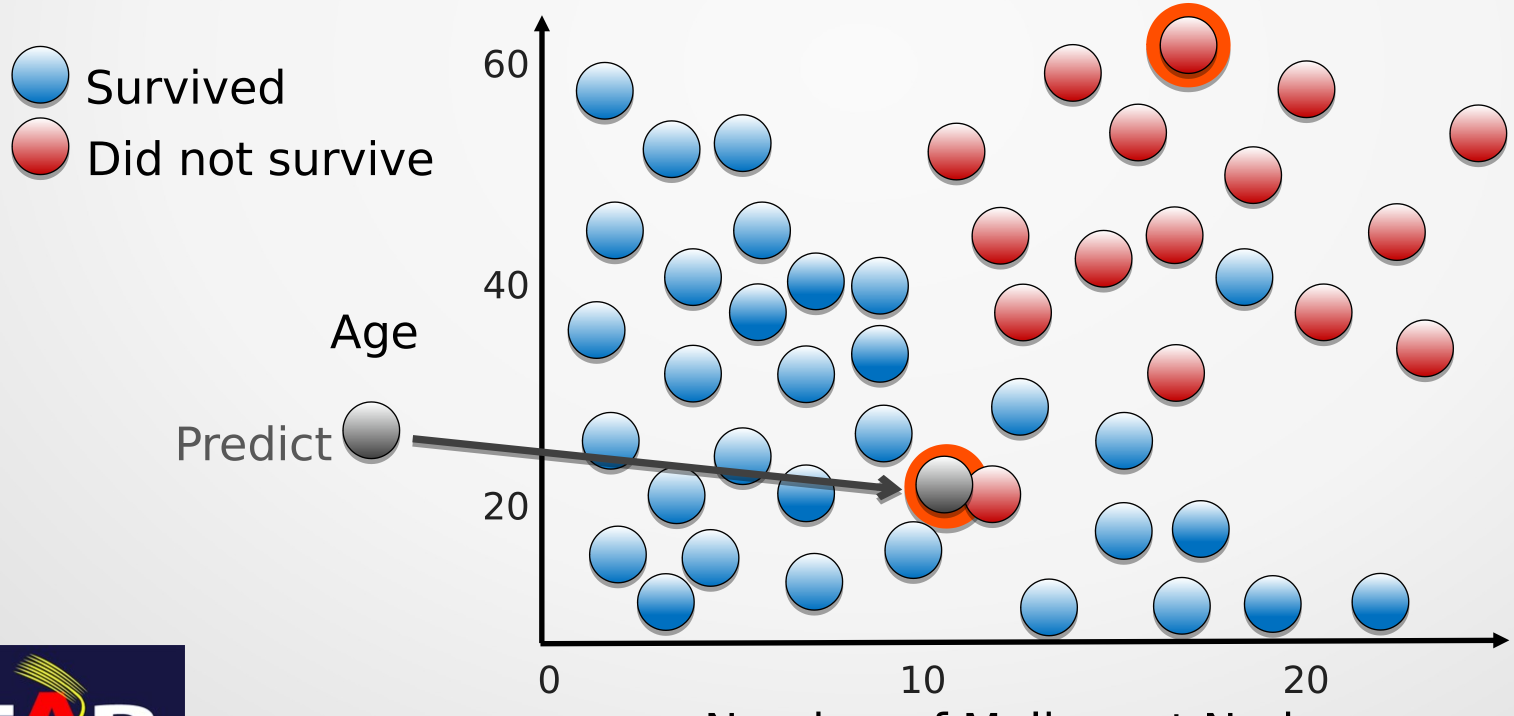
We will discuss methods for determining ‘K’ in a future lesson



KNN Distance Metric

How do we know which is the closest neighbor?

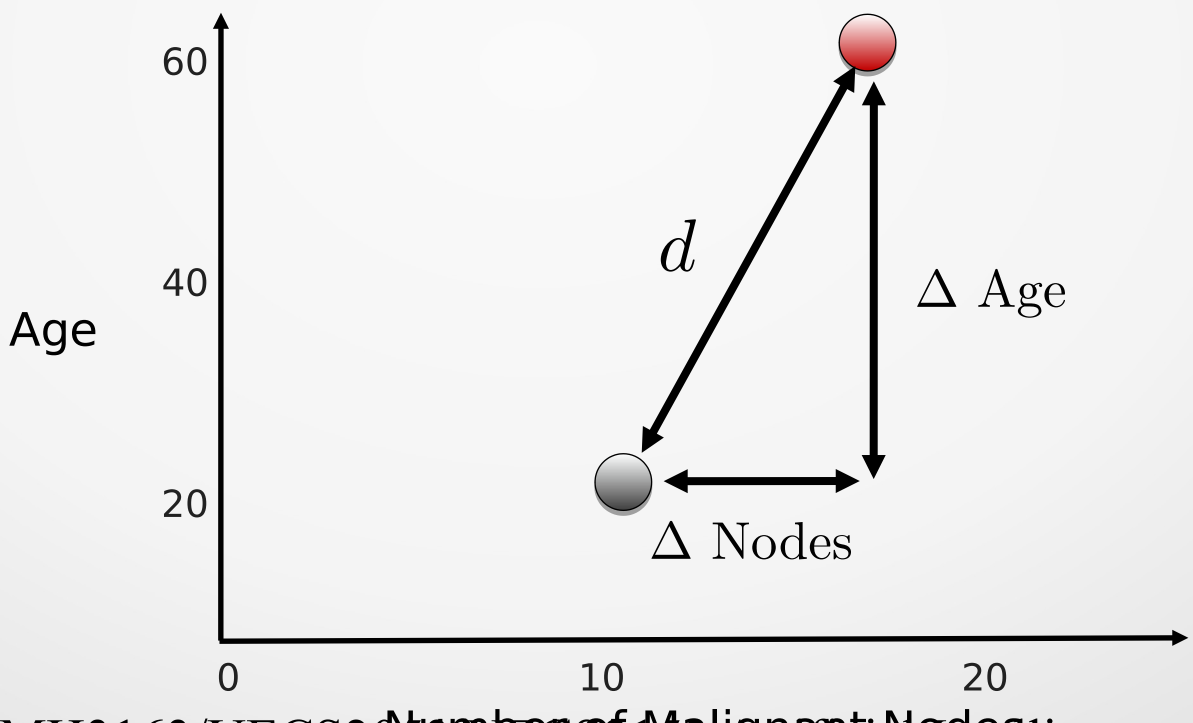
‘Closest’ implies evaluating distance



KNN Distance Metric

The most obvious measure is Euclidean Distance

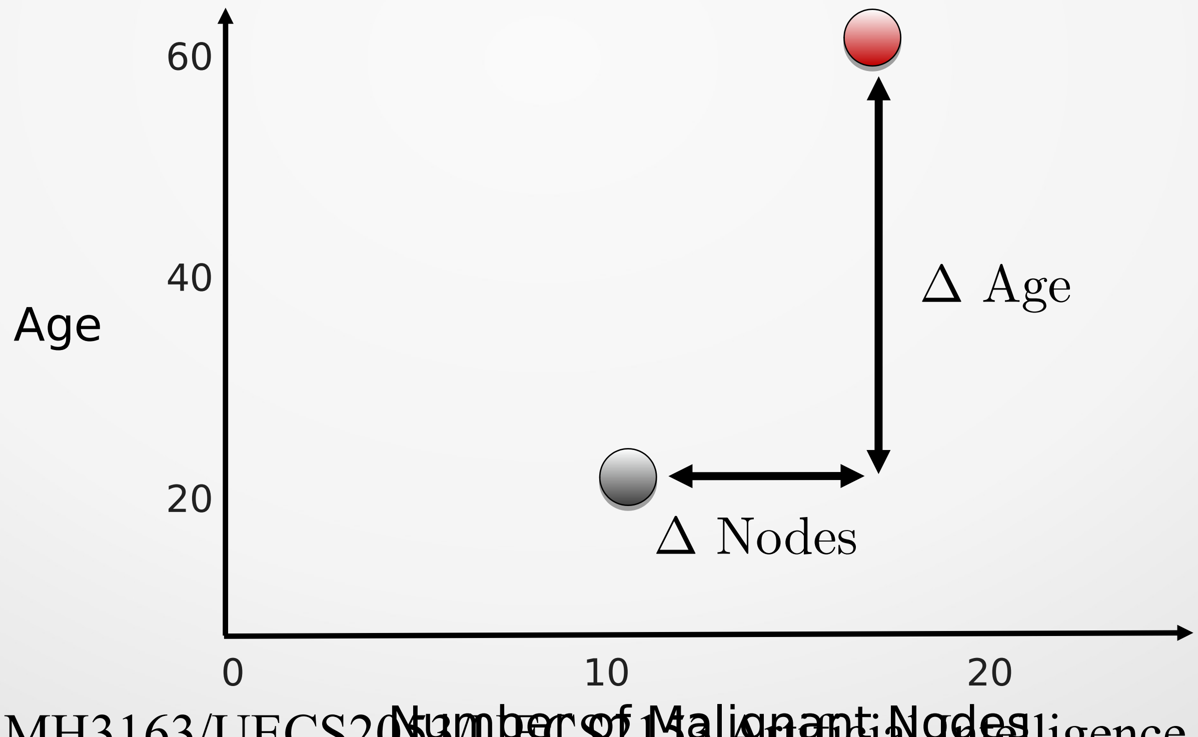
This is also known as L2 Distance $d = \sqrt{\Delta\ \mathrm{Nodes}^2 + \Delta\ \mathrm{Age}^2}$



KNN Distance Metric

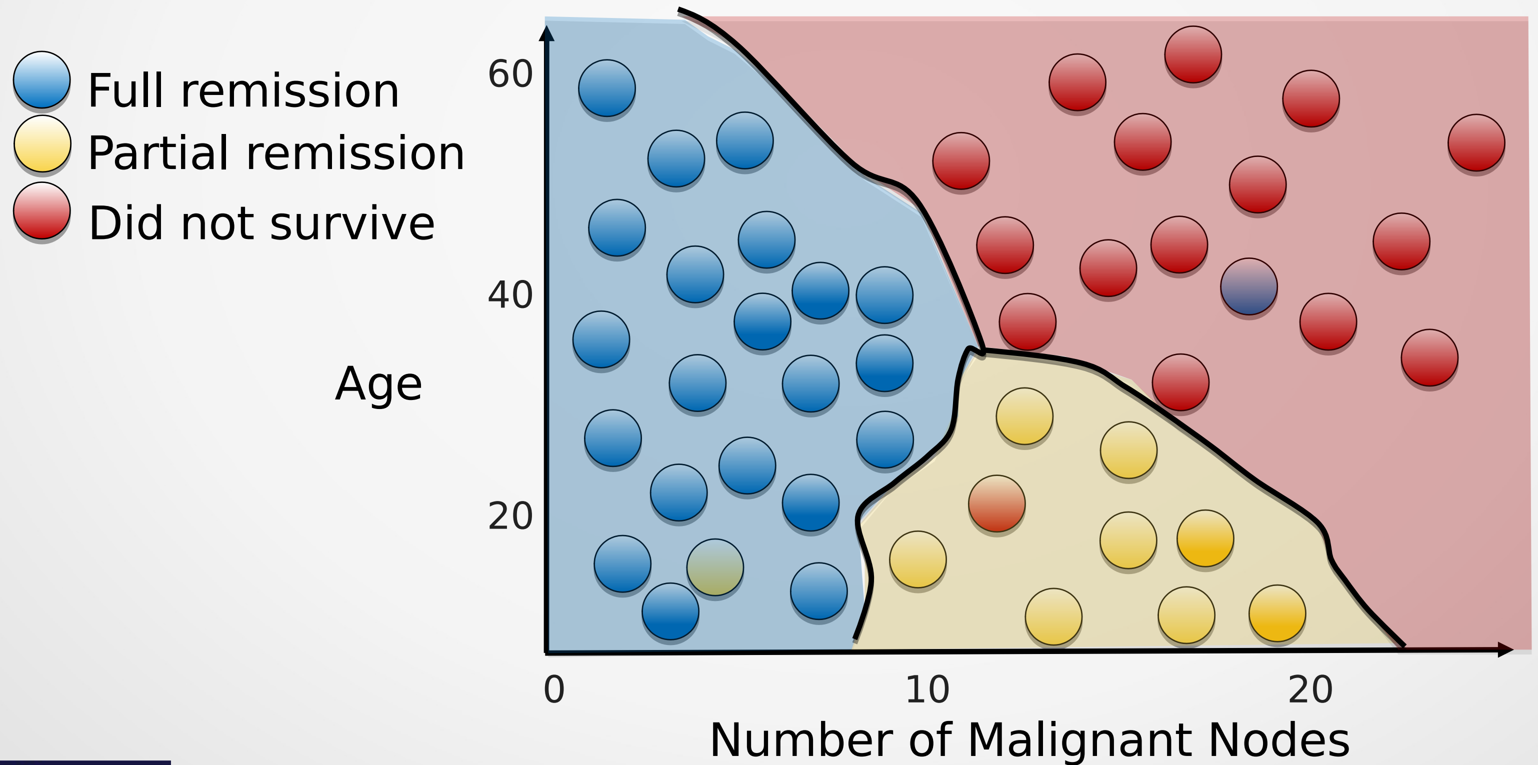
Another option is the Manhattan (City Block) Distance

This is also known as L1 Distance $d = |\Delta\ \text{Nodes}| + |\Delta\ \text{Age}|$



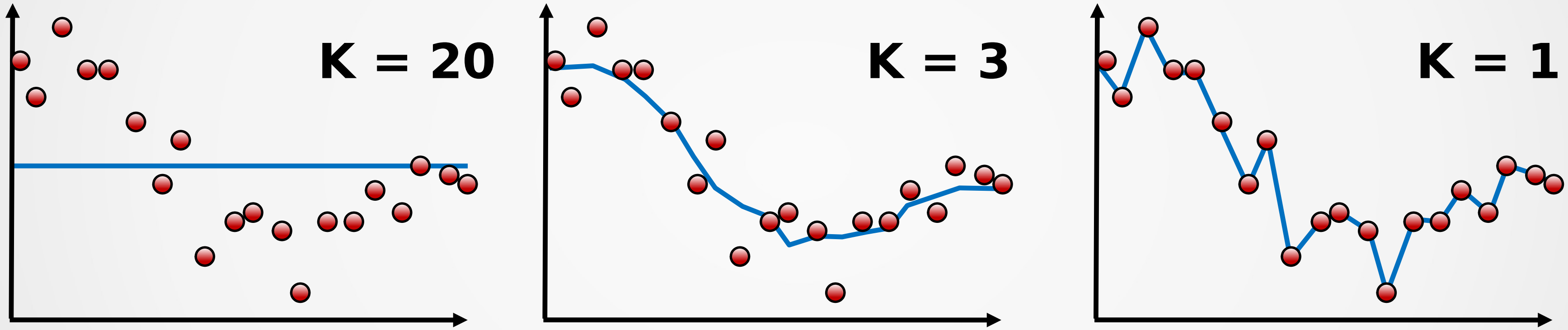
Multiclass KNN Decision Boundary

KNN is not limited to two-class problems. Consider the example below for K = 5



Regression with KNN

The Nearest Neighbors model can also be used for regression problems, as with this example below



Characteristics of a KNN Model

- Fast to create model because it simply stores data (no training process unless using CNN etc.)
- Slow to predict because many distance calculations are required
- Can require a lot of memory if data set is large
- Heavily affected by curse of dimensionality

Basic KNN Syntax

Import the class containing the classification method

```
from sklearn.neighbors import KNeighborsClassifier
```

Create an instance of the class

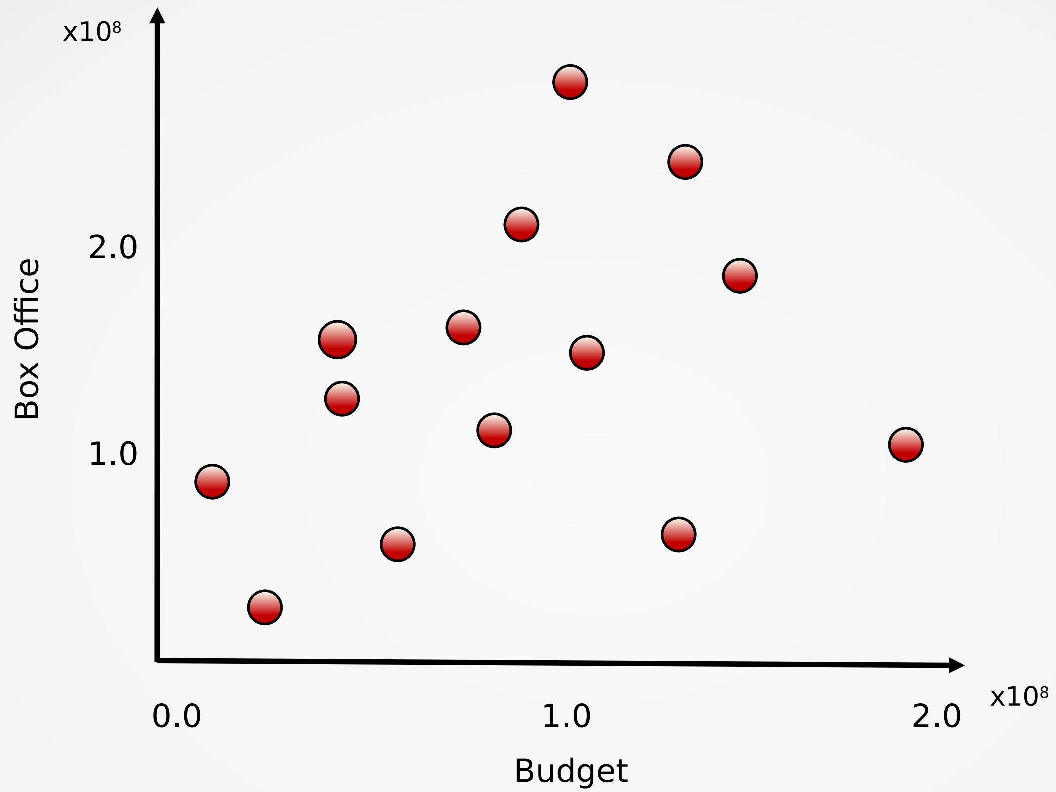
```
KNN = KNeighborsClassifier(n_neighbors=3)
```

Fit the instance on the data and then predict the expected value

```
KNN = KNN.fit(x_data, y_data)
y_predict = KNN.predict(x_data)
```

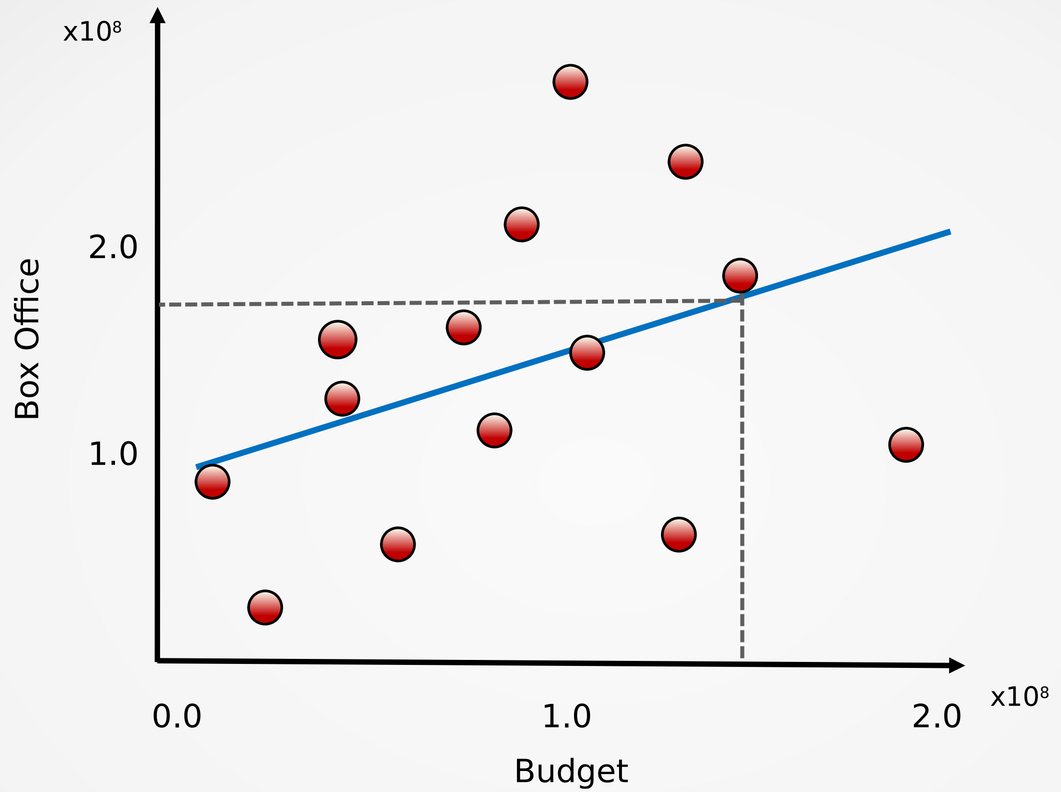
Regression can be done using KNeighborsRegressor

Introduction to Linear Regression



$$y_\beta(x) = \beta_0 + \beta_1 x$$

Predicting from Linear Regression

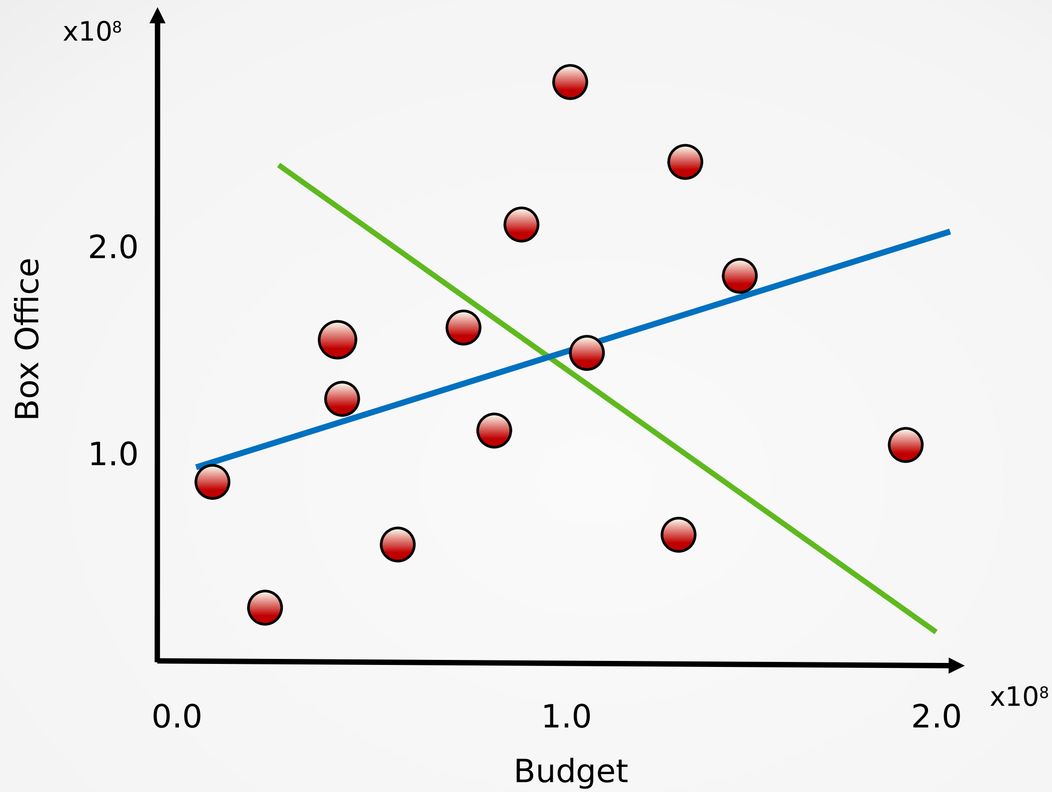


$$y_\beta(x) = \beta_0 + \beta_1 x$$

$$\beta_0 = 80 \text{ mill}, \beta_1 = 0.6$$

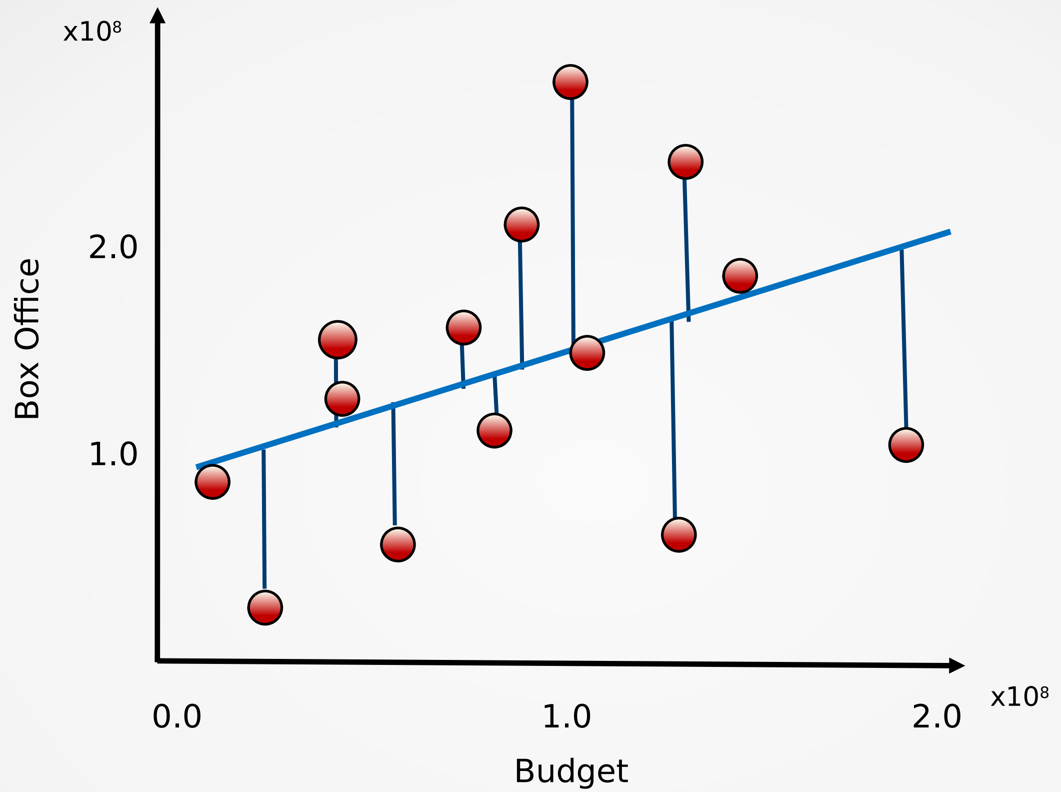
Predicts 175 million gross box office for 160 million budget

Best Linear Regression Model?



$$y_\beta(x) = \beta_0 + \beta_1 x$$

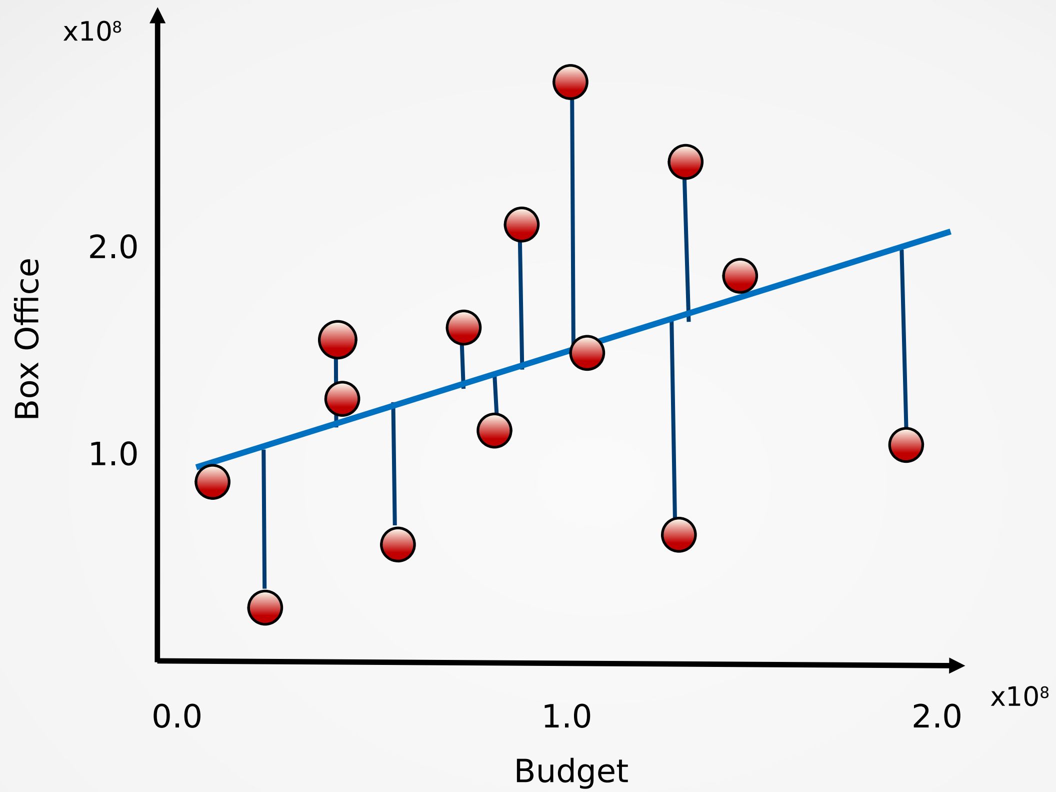
Calculating the Residuals



$$y_\beta(x^i_{obs}) - y^i_{obs}$$

$$\left(\beta_0 + \beta_1 x^i_{obs}\right) - y^i_{obs}$$

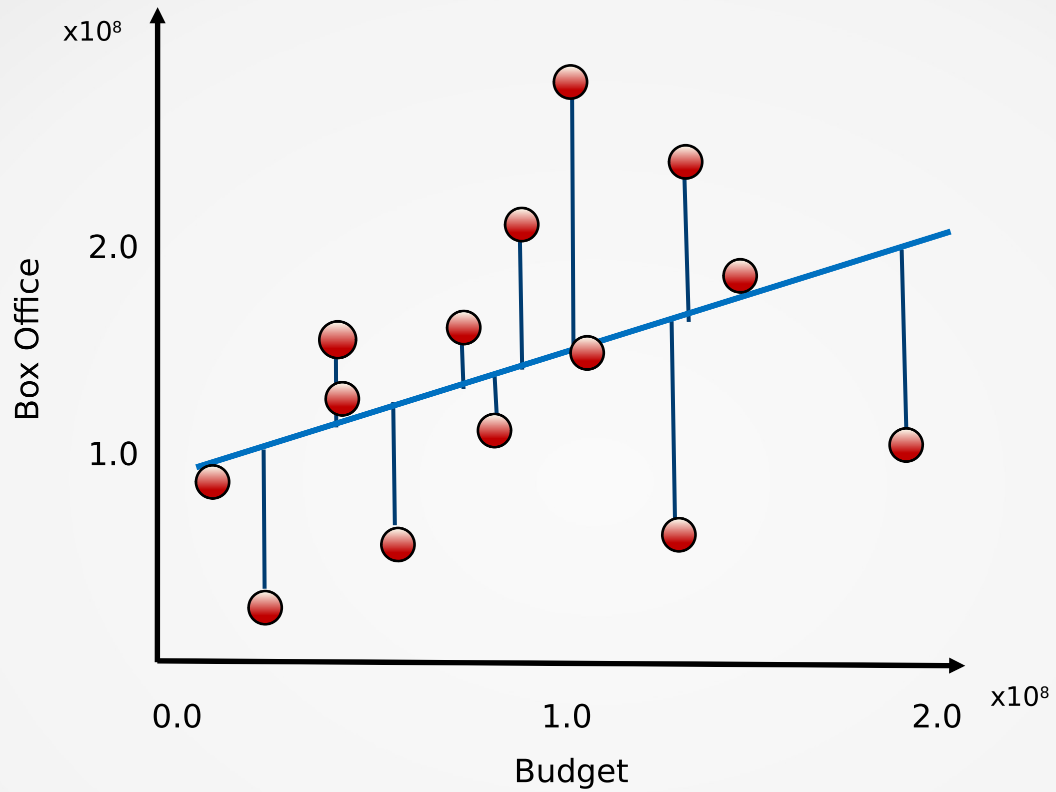
Calculating the Residuals



$$\frac{1}{m}\sum_{i=1}^{m} left(\left(\beta_0 + \beta_1 x^i_{obs}\right) - y^i_{obs}{}^2$$

Mean Squared Error

Calculating the Residuals



$$\min_{\beta_0, \beta_1} \frac{1}{m} \sum_{i=1}^{m} left((\beta_0 + \beta_1 x^i_{obs}) - y^i_{obs}{}^2$$

Minimum Mean Squared Error

Other Model Metrics

- Sum of Squared Error (SSE) $$\sum_{i=1}^{m}\left(y_\beta(x^i) - y_{obs}^i\right)^2$$
- Total Sum of Squares (TSS) $$\sum_{i=1}^{m}\left(\overline{y_{obs}} - y_{obs}^i\right)^2$$
- Correlation Coefficient (R^2) $$1 - \frac{SSE}{TSS}$$

Comparing Linear Regression/KNN

Linear Regression

- Fitting involves minimizing cost function (*slow*)
- Model has few parameters (*memory efficient*)
- Prediction involves calculation (*fast*)

K Nearest Neighbors

- Fitting involves storing training data (*fast*)
- Model has many parameters (*memory intensive*)
- Prediction involves finding closest neighbors (*slow*)

Linear Regression Syntax

- Import the class containing the regression method

```
from sklearn.linear_model import LinearRegression
```

- Create an instance of the class

```
LR = LinearRegression()
```

- Fit the instance on the data and then predict the expected value

```
LR = LR.fit(x_train, y_train)
y_predict = LR.predict(x_test)
```

End of Lecture

Many thanks to Intel Software for providing a variety of resources for this lecture series