PREDICTIVE ANALYTICS/MACHINE LEARNING

BAN400

Håkon Otneim

The plan for today

- This course mainly concerns the *programming* aspect of data science. In other words, we deal with a lot of the infrastructure that is necessary to perform data science in practice.
- Today, however, we will briefly visit a core topic: namely the statistical modeling that is used to produce *predictions* in data science. This is often labelled *machine learning* or *statistical learning* (and sometimes more vaguely as AI and other fancy terms...).
- We will only give a superficial treatment today (for completeness). The topic is covered in detail in BAN404 - Predictive Analytics with R, due to run in the spring.

Literature

- The standard basic reference to statistical learning (and textbook in BAN404) is ISL.
- ESL is a more advanced book.
- Both of these are freely available online.

The basic (generalized) linear model

Let us go back to basics and model the association between a set of explanatory variables (covariates, independent variables, right-hand-side-variables, X's, etc ...) and a response variable (dependent variable, left-hand-side-variable, or just simply ...) Y using the following linear relationship:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon,$$

where ϵ is a random variable typically assumed to be normally distributed. If Y is a continuous variable we call this a *regression problem*.

If Y is a binary (dummy, zero-one) variable, we use the logistic regression:

$$\text{logit}(P(Y = 1|X_1, \ldots, X_p)) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

We call this a *classification problem*.

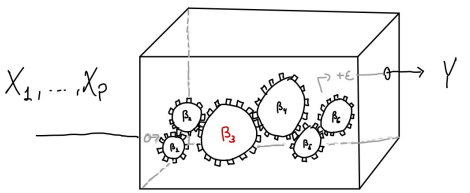
- In classical econometrics we are often interested in statistical and causal inference.
- In modern data science we are more interested in predictions, but this distinction should not be taken too far.
- We will focus on the classification problem in the examples today, because it is easier to visualize.

The rise of the black box analogy and why we don't like it

THE LINEAR MODEL

$$Y = \underbrace{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots + \beta_p X_p}_{\text{The systematic, possibly causal, relationship between } X_1, \dots, X_p \text{ and } Y, \text{ assumed to be linear}} + \varepsilon$$

Random, unpredictable noise



Is this a good illustration?

Not really...

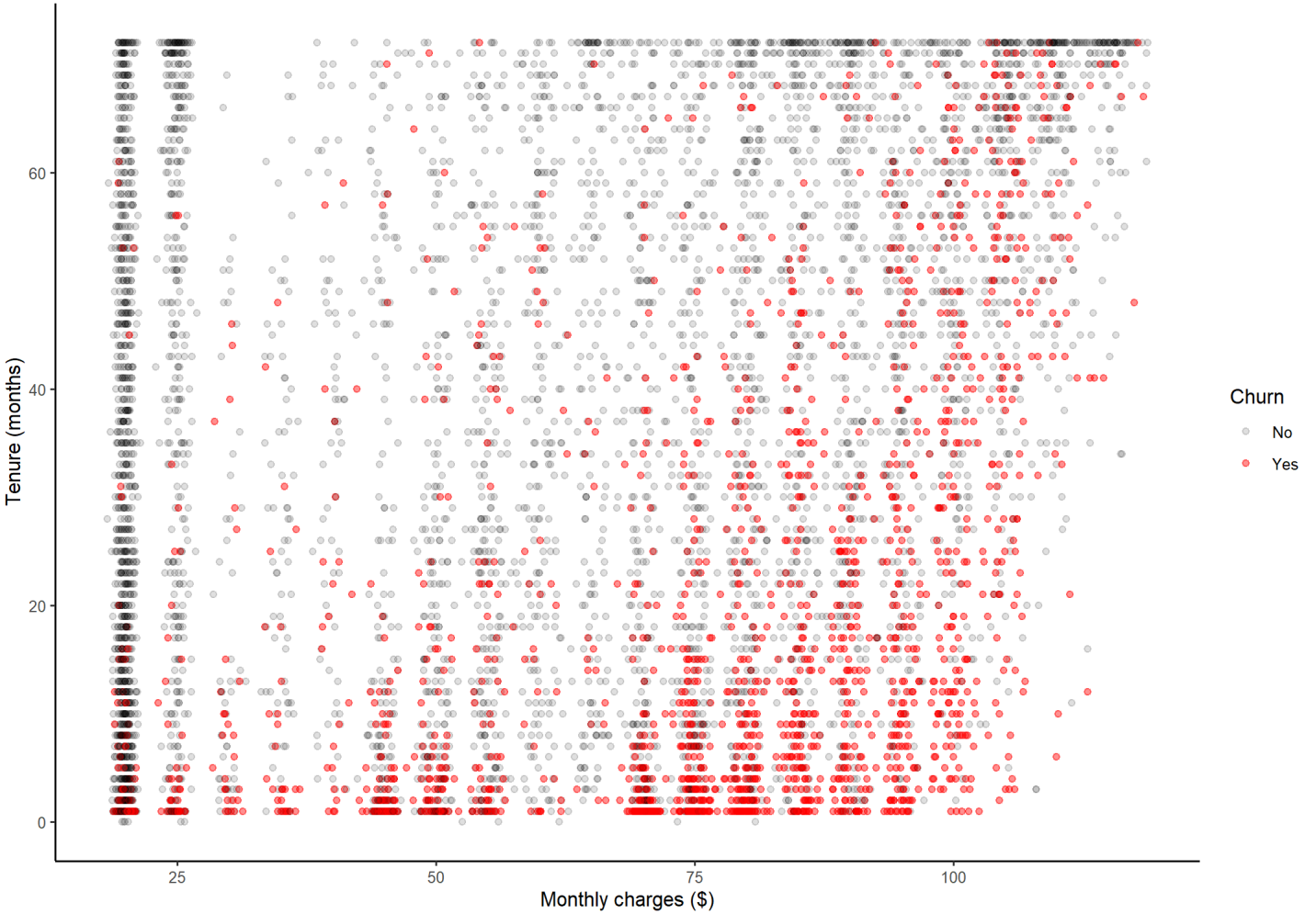
A GENERAL PREDICTION MODEL

$$Y = \underbrace{f(X_1, \dots, X_p, \varepsilon)}_{\text{The systematic relationship between } X_1, \dots, X_p \text{ and } Y, \text{ which we learn directly from the data, with few, if any, a priori assumptions.}}$$

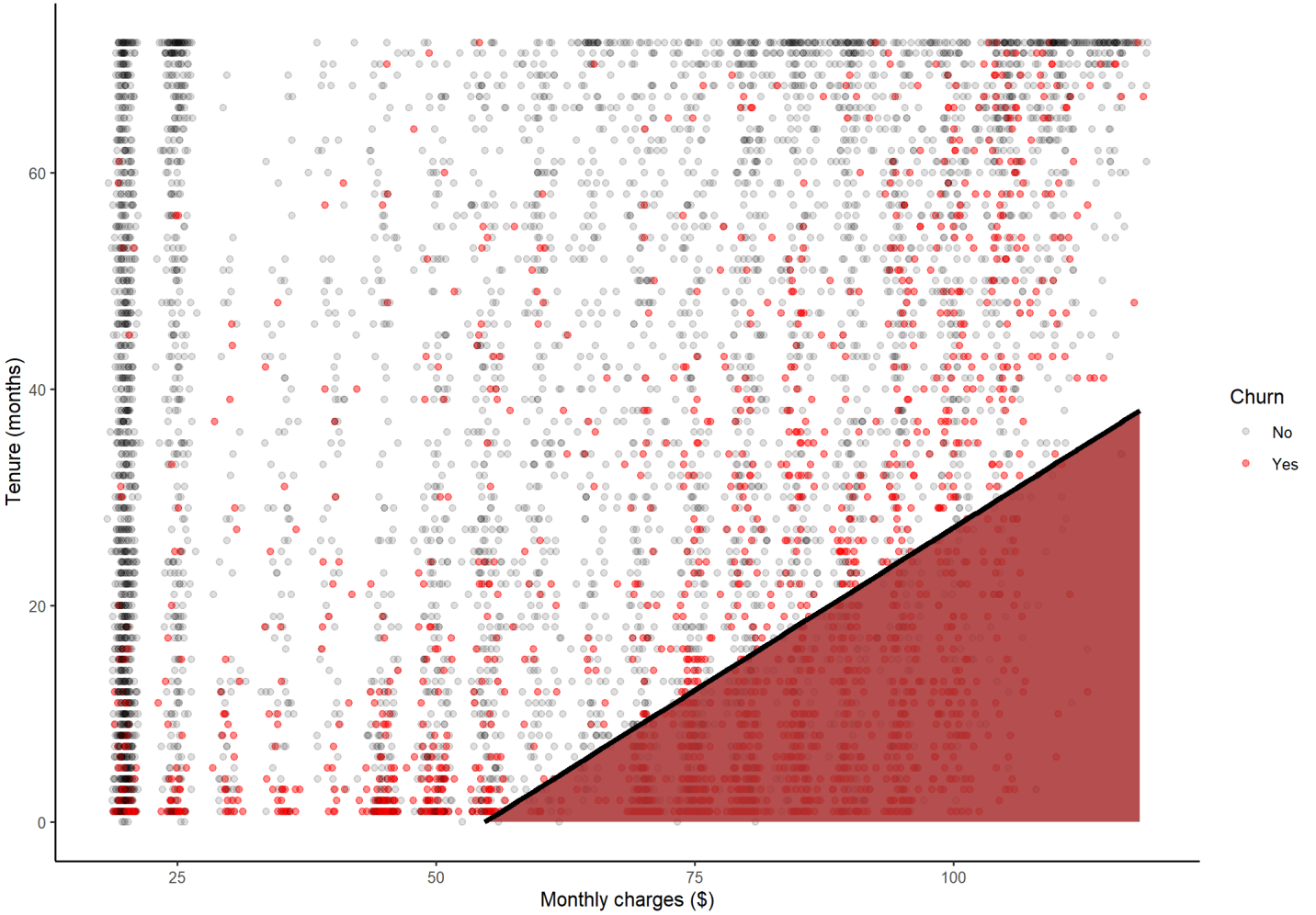
Classical example: Customer churn

- We have access to a database of current and previous subscribers at a large American cell phone company (Telco).
- Data source at Kaggle.
- Number of observations: $n = 7043$.
- We know the length of the customer relationship, how much the customer pays each month, and wheter the customer has ended the subscription (churn).
  - The classical inference question: to what degree do the two variables explain the probability of churn?
  - The ML-question: How can we classify current customers as either loyal or at risk of ending their subscription? Who do we pick out for targeted marketing?

The churn data: What do we see?



Logistic regression decision boundary



Pros and cons of using LR for classification

- The logistic regression is a classical econometric model designed for *statistical inference*, i.e. figuring out which explanatory variables contributes to variation in the "success probability" $P(Y = 1)$, and in which direction.
- The logistic regression defines a proper probability model:

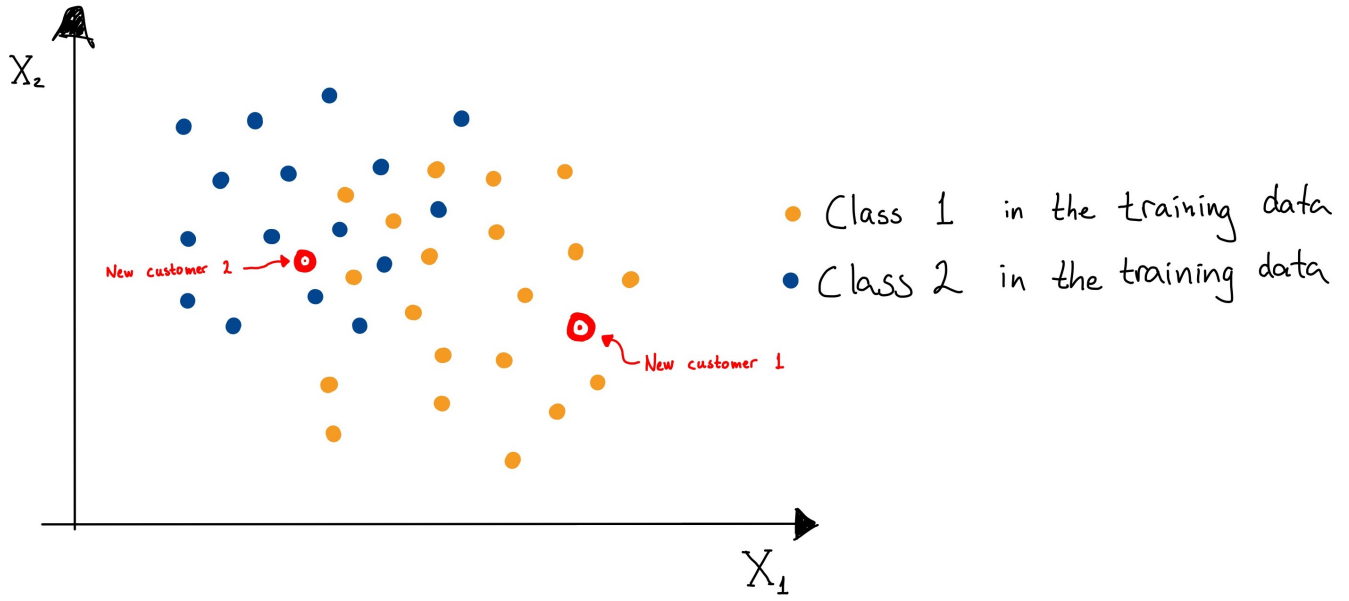
$$\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

- This is classical parametric statistics. Unless you introduce polynomial terms, the LR *always* leads to a straight line decision boundary.
- the LR is interpretable, requires less data (because we have decided the shape of the boundary even before we see any data).
- But if we
  1. don't want to assume that this probability model is (approximately) true, and
  2. are more interested in predicting the status of new customers than interpreting the estimated model,
  3. (and have a reasonable amount of data),
- we might want to do something else!

A simple, yet powerful idea

- Make predictions based on *similarity*.
- That is: when predicting the status of a new customer A, look at your data set of known outcomes, and specifically the customers the are most *similar* to A.
  - If the customers that are most similar to A tend to churn a lot, then predict A to be a churner as well.
  - If the customers that are most similar to A do not tend to churn, then predict A to be a safe customer.
- This is basically the starting point of the relatively recent popularity of (data science/predictive analytics/machine learning/deep learning/statistical learning/AI/...), choose your own poison.
- Today we will look at some basic principles of this idea, and point to some fallacies.

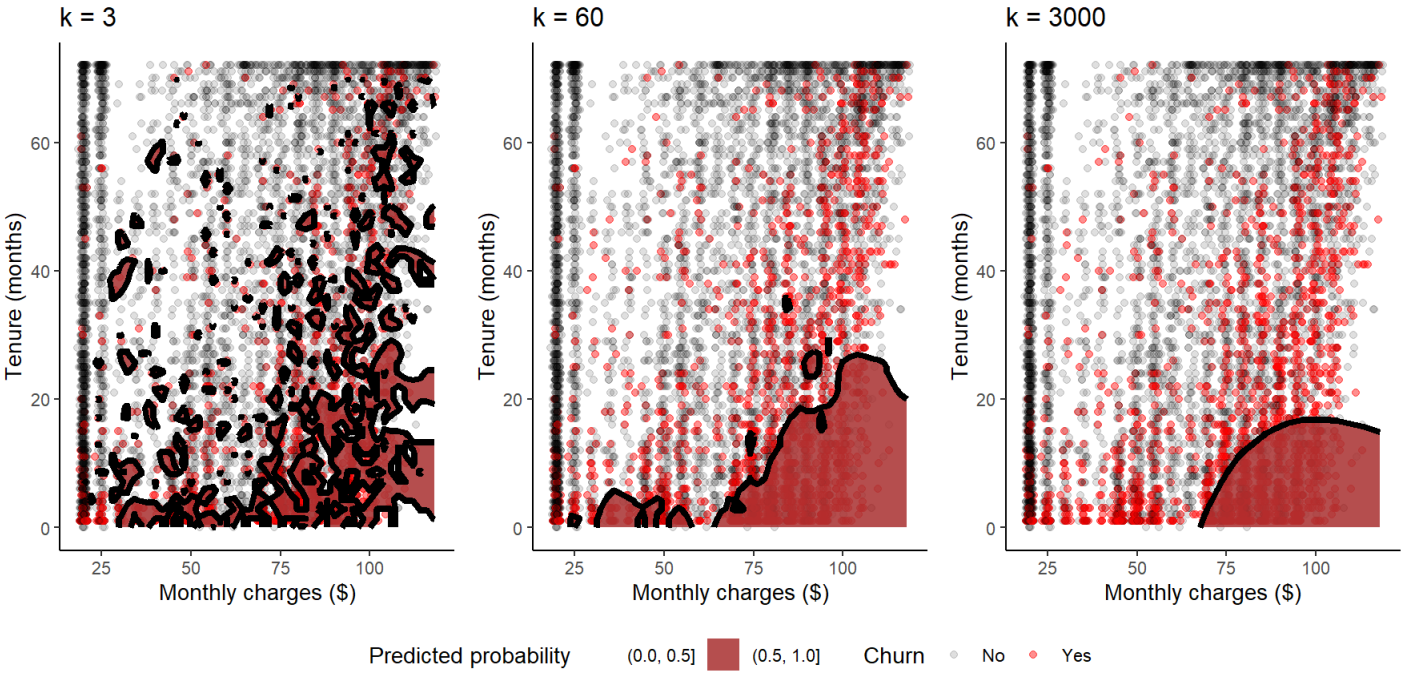
A simple algorithm: kNN, k nearest neighbours



k nearest neighbours algorithm for predicting the class of a new individual:

- Select an integer k.
- Identify the k individuals in the training data which is closest/most similar to the new individual.
- Perform a "majority vote" among these individuals in order to predict the status of the new individual.

It is crucial to select a good value for k



- $k = 1$ gives a *perfect* prediction on the training data, but horrible predictions -- the model is *overfitted*.
- $k = n$ results in total ignorance of the explanatory variables. You always predict into the majority class in the training data -- it is *underfitted*.
- The best value must lie somewhere in between, **but where, and how do we find it?**

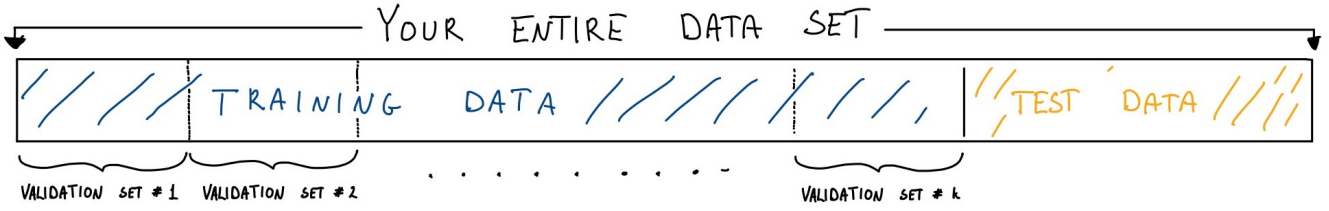
Choosing the right complexity

- If we use the same data for *estimating (training)* the model and *choosing the complexity* we make the classical overfitting mistake.
- In the kNN case, we will always end up with $k = 1$, because that gives perfect predictions on the training data.
- It will result in catastrophic predictions, however.
- We can fix this by splitting the data randomly into three parts:

- The three parts of the data set play different roles when estimating the model:
  - The **training data** is used to estimate/train the model.
  - We use the **validation data** to tune the model: Which value of k, when used to estimate the model using the training data, gives the best predictions on the validations set?
  - We use the **testing data** to test the model afterwards: how well does the model predict unseen data (out-of-sample)?.
- It is **abslolutely crucial** that the testing data is kept separate from model training and tuning. Checking predictions on data that was used to train a model is considered cheating!

Slightly more sophisticated: Cross-validation

- We can use the data more efficiently using cross-validation:



- Split your training data randomly into m random folds (splits).
- (Unfortunately, the symbol k is typically used for this number as well, but we will use m to separate it from the k in kNN).
- For each candidate values of k:
  - For each validation set 1:m:
    - Estimate the model on the remaining training data and record the predictions on the validation set.
  - Report the average prediction error over all m folds.
- Choose the value of k that gives the smallest "cross-validated" prediction error.
- In the end you test the model on the test data as usual.
- **Very precise, but also very computer intensive.**

Measuring performance

We can measure the performance of prediction models in many ways.

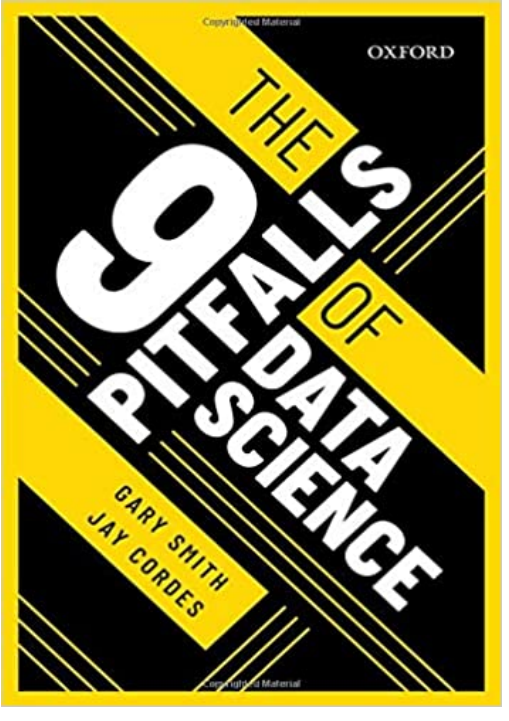
- Making a confusion matrix:

	True 0	True 1
Predicted 0	True negative (TN)	False negative (FN)
Predicted 1	False positive	True positive (TP)

- Drawing the reciever operating characteristic (ROC), takes all classification thresholds into account:

- A numeric measure of over-all predictive power is the AUC (Area under the curve).
  - AUC = 0.5 is random guessing
  - AUC = 1 is perfect prediction.

This is not magic.



- Data science is a young, but large field that covers parts of applied statistics, mathematics, computer science and informatics.
- Personal opinion: lots of the Kaggle-type "data science" predict-whatever-using-lots-of-variables is garbage.
- If you stop thinking and ignore the importance of data validity and subject field expertise, you always loose.
- Data mining is (still) a dodgy activity.