Session 1- Introduction

12 January 2022 08:38

“Machine learning is the field of study that gives computers the *ability to learn* without being explicitly programmed to.” (the algorithm is a piece of code is that tries to give computers the ability to learn).

A computer program is said to learn from **experience E** with respect to some **task T** and some performance **measure P**, if its performance on T, as measured by P, improves with experience E.

How to make a computer program learn?

- Experience E
- With respect to some task T
- And some performance measure P

If its performance on T, as measured by P, improves with experience E.

ML systems automatically learn programs from data.

➔ What is learning? Three different steps and we focus only on representation.

LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION.

1. **Representation:** represent your problem. It is choosing your type of solution/classifier based on the assumptions e.g. If your data is linear, we use linear regression.

   "Choosing the set of classifiers that it can possibly learn. This set is called the *hypothesis space* of the learner.”

2. **Evaluation:** you have your model, but you need to know if it's solving the problem correctly. Model is improving evaluation metric if you pick wrong evaluation metric (wrong).
   “An evaluation function (also called objective function or scoring function) is needed to distinguish good classifiers from bad ones. What are you trying to improve?”

3. **Optimization:** (machine learning III). It tells your model how to improve the evaluation metric e.g., the Size of the house in order to understand the price. If the ML algorithm identifies the name of the owner explains the price, the optimization metric tells you if it explains it correctly or not.
   “Needing a method to search among the classifiers in the language for the highest-scoring one. The choice of optimization technique is key to the efficiency of the learner.”

   Space of solutions
   Evaluation metric
   Optimization metric

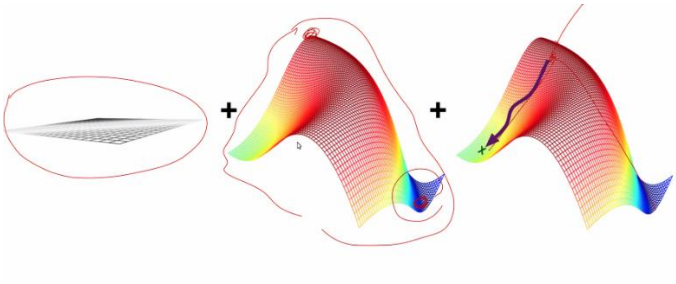


Table 1: The three components of learning algorithms.

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

INTRODUCTION

WHAT IS MACHINE LEARNING?

↳ It is the field of study that gives computers the ability to learn without being explicitly programmed.

↳ A computer is said to learn from Experience E with respect to some task T and some peformace measure P, if its peformace on T (as measured by P) improves with Experience E.

(1) • For e.g. your spam filter is a ML program that, given examples of spam emails (e.g. flagged by users) and regular emails can learn to spam them.

(2) The examples that the system uses to learn are called training set and each training example is called a training instance (or sample)

(3) In this case, the task T is to flag spam for new emails, the Experience E is the training data and the peformance P needs to be defined e.g. ratio of correctly classified emails - this peformance measure is called accuracy (and its often used in classification tasks).

↳ If you download a copy of wikipedia, your comp has a lot more data but it is not suddenly better at any task ∴ not ML.

∴ ML systems automatically learn programs from data

=> What is learning? (steps to allow computer to learn)

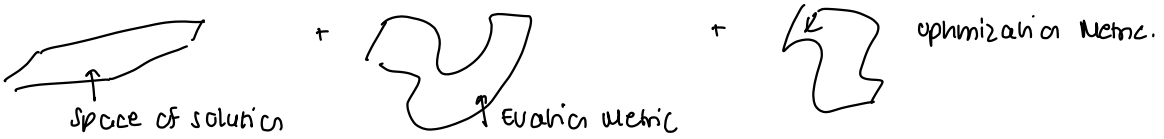
LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION
(focus)

① Representation: choosing your type of classifier / solution based on the assumptions
↳ e.g. if your data linear, we use linear regression.

∴ choosing set of classifiers that it can possibly learn. This set is called the hypothesis space of the learner".

② Evaluation: you have your model but you have to tell your model if its solving the problem correctly.
↳ An evaluation function / objective function / scoring function is needed to distinguish good classifiers from bad ones; what are you trying to improve?

③ Optimization: (ML III) tells your model how to improve your evaluation metric.
↳



What you care about is how your algorithm works with NEW data - you already know what happened with training data; you care about learning from that to predict new data.

First IDEA:

ITS GENERALIZATION THAT COUNTS

E.g., create a model to predict churn, 99% accuracy is it something you want? You want the algorithm to work well with new data. In your training data, you already know what happened- care about learning to predict new data.

- The fundamental goal of machine learning is to generalize beyond the examples in the training set. *Avoid* **overfitting!**
- **Generalization:** how good your model behaves with new data. Difficult because you only have past data.
- **Cross-validation** is an evaluation technique to see how good your model is when put to production.

"Randomly dividing your training data into (say) ten subsets, holding out each one while training on the rest, testing each learned classifier on the examples it did not see, and averaging the results"

➔ Overfitting- you do not generalize. Model is too rigid and learning only on training data.

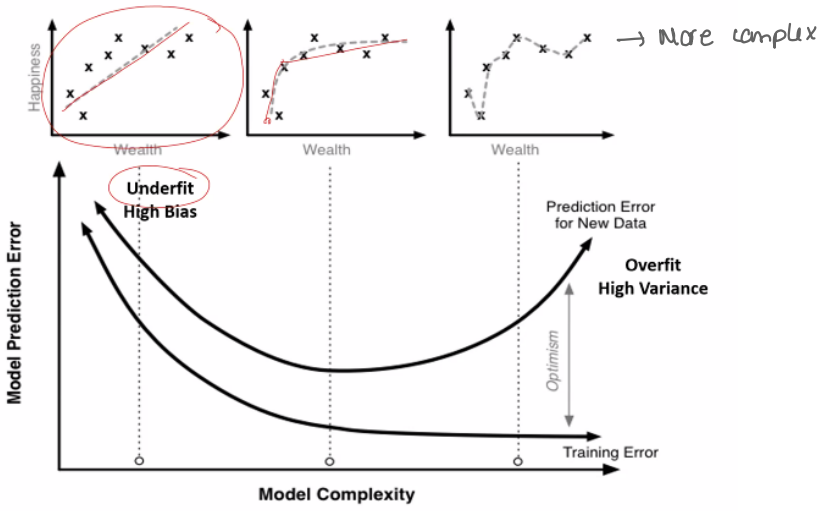
For example, with a known wealth, predict its happiness.

↗ the relationship is more complex;

- You suspect linear relationship and create linear model (we can see captures but it not good, underfitting: model too simple, the relationship is too simple).
- Quadratic model: Reduce train and test error
- Even more complex model like neural network predicts training perfectly and error is going to be 0. perfect on data but not on any new data.
- Overfitting: 0 error in training and huge in test. Learning too much from data and not generalization. Cross-validation is the way to evaluation ML model.
- The lager dataset and the more intelligent, the care you less about over fitting.

not learning so much -

↗ you are not learning enough (relationship more complex)



OVERFITTING HAS MANY FACES

- A **linear learner** has a **high bias**, because when the frontier between two classes is not a hyperplane the learner is unable to induce it.
- **Decision trees** don't have this problem because they can represent any Boolean function, but on the other hand they can suffer from **high variance**: decision trees learned on different training sets generated by the same phenomenon are often very different, when in fact they should be the same.
- Thus, contrary to intuition, **a more powerful learner is not necessarily better than a less powerful one**.

x m

$x \times x \times x$

x^3

wt $6\text{ kg} / x^3$

$2x \times 2x \times 2x = 8x^3 \times 6x^3$

$\frac{41}{78}$ → Spanish

$\frac{22}{78}$ → Italian $\quad P(A \cup B) = P(A) + P(B) - P(A \cap B)$

$41 + 22 \cdot 9$

9 $\quad P(S) = \frac{41}{78}$

$P(I) = \frac{22}{78} \cdot 54$

$12 + 5$

$12x + 5 = 17$

1

$12x + 5$

$4x = 30$

$x = 7.5$

5×7.5

4 machines → 30 hrs

0.6 $\quad$ 5 machines

$(\sqrt{2} - \sqrt{3})^2$

$(\sqrt{2} - \sqrt{3})(\sqrt{2} - \sqrt{3})$

$2 - \sqrt{2}\sqrt{3}$

$2 - \sqrt{2}\sqrt{3} - \sqrt{2}\sqrt{3} + 3$

$5 - 2(\sqrt{6})$

$10 + 14 = 24$

$10 + 14 + 24$

$4(2^{30})$

CURSE OF DIMENSIONALITY

For example:

A. 10,000 INSTANCES and 100 features
B. 10,000 and 10,000 features

In which more information? If the descriptors are informative enough and depend on each other - less information. It depends on the features.

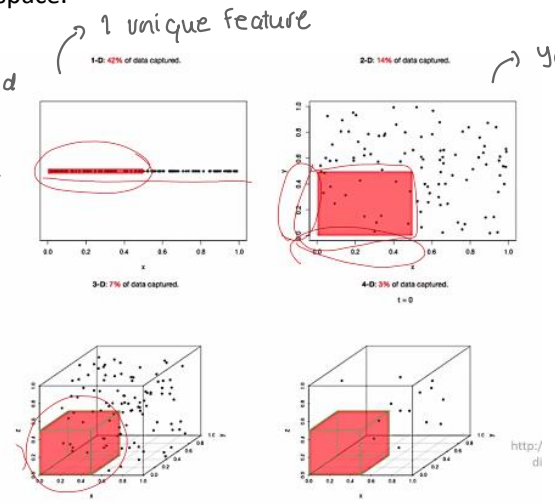
➔ If you are adding features to your data that are not relevant, you have LESS information because of the CURSE of DIMENSIONALITY.

E.g., If you have unique features like the size of the house to predict the value. When you add more features, you are forcing the algorithm to look for a solution on a higher dimension (with the same data points), less information.

As you add a feature, you add a new dimension and space is bigger and the ML algorithm is forced to search in this space.

Adding dimension increases your search space.

=> As you add a feature, you add a dimension and your search space is bigger and bigger :. forcing your ML model to search in a bigger space



Therefore, a key part of the ML algorithm is finding relevant features- important because your try to reduce the CURSE of DIMENSIONALITY.

If able to do feature engineering right and select the right features and data is going to be together in search space.

E.g., Label whether faces are sad or happy. By using the right features, you can cluster/put together similar data points and not randomly distributed them into space.

- Data not randomly distributed into space.
- Remove irrelevant and add relevant.

INTUITION FAILS IN HIGH DIMENSIONS

➔ **Curse of dimensionality:** many algorithms that work fine in low dimensions become intractable when the input is high-dimensional.

➔ There is an effect that partly counteracts the curse, which might be called the "**blessing of non-uniformity**". In some applications, examples are not spread uniformly throughout the instance space but are concentrated on or near a lower-dimensional manifold.

Data is not randomly distributed into the space.
-> If you select right features, data is going to be close together in search space.

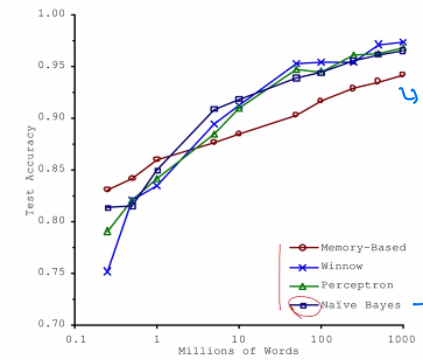
-> Decision trees are robust but still important.

FEATURE ENGINEERING IS THE KEY

- Some machine learning projects succeed and some fail. What makes the difference? the most important factor is the **features** used.
- Often, the raw data is not in a form that is amenable to learning, but you can construct features from it.
- Machine learning is not a one-shot process of building a data set and running a learner, but rather an **iterative process** of running the learner, analysing the results, modifying the data and/or the learner, and repeating.

MORE DATA BEATS A CLEVERER ALGORITHM.

If repeated, discard it.

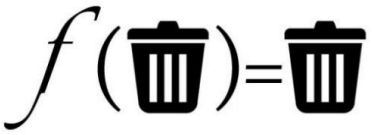


→ With few data points, a cleverer algorithm is going to be better
↳ Neural net (more complex)
∴ collect more data.
→ simple one

4 algorithms. Few data points, cleverer model (neural network-memory based) will be more accurate. If you add data points, all reach a performance of 0.95% of accuracy.

DATA ALONE IS NOT ENOUGH GIGO PRINCIPLE

More data is better unless data is rubbish.
↓
No errors, relevant, novel data.



↳ If you feed your ML with rubbish, you are going to get rubbish.

More data good solution, unless data is rubbish. Data without errors, that is novel and relevant. "Garbage in Garbage out" principle.

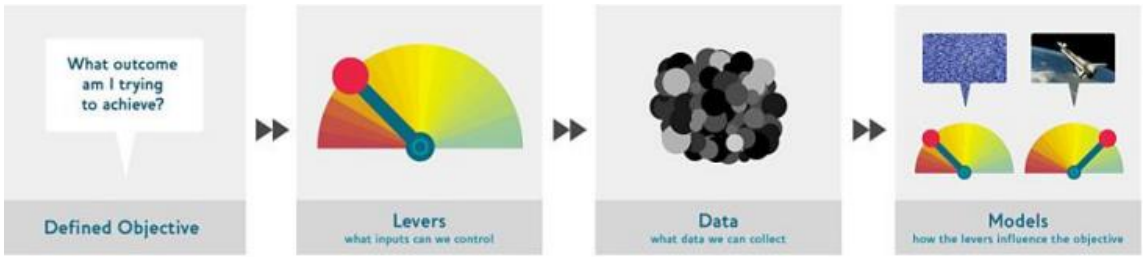
You want training data to be as representative as possible. Bigger data set less worrying about overfitting.

↑ The problem of overfitting is that your training data is not representative.

MORE DATA IS GOOD BUT DATA ALONE IS NOT ENOUGH.

To create an ML model, you need to understand the context of the model.

- Defined objective: what outcome am I trying to achieve?
- Levers: what inputs can we control
- Data: what data can we collect
- Models: how the levers influence the objective



One of the key criteria for choosing representation are which kinds of knowledge are easily expressed in it.

Every learner must *embody some knowledge* or assumptions beyond the data it's given: Try to remove irrelevant and add relevant (therefore reducing dimension).

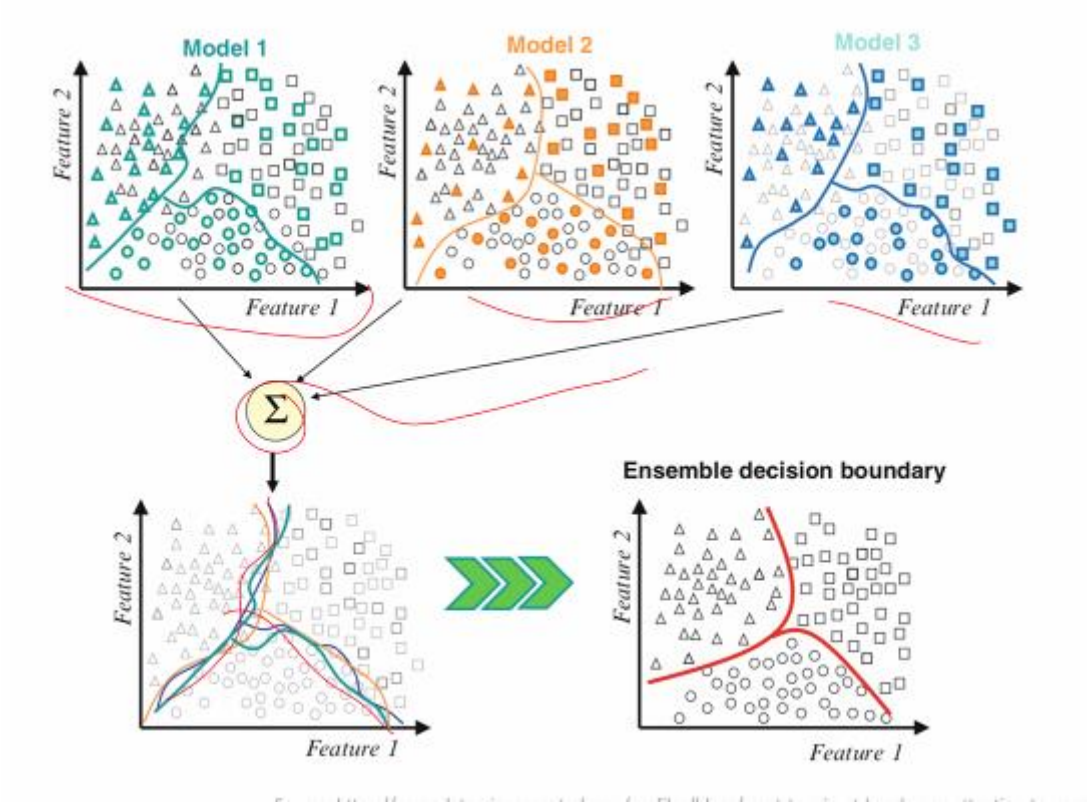
- if we have knowledge about what makes examples similar in a domain, instance based methods may be a good choice.
- if knowledge about probabilistic dependencies → graphical models
- and if knowledge about what kinds of preconditions are required by each class "IF... THEN..." rules may be the best option.

LEARN MANY MODELS, NOT JUST ONE

- Before everyone had their favourite learner, with some reasons to believe in its superiority. The most effort went into trying many variations of it and selecting the best one.
- **No free lunch theorem:** there is no single best model that works best for all problems.
- **Ensemble**: if instead of selecting the best variation found, we combine many variations, the result is better.

➔ **Concept of Ensemble Decision Boundary**

Instead of creating a unique model, create several models that focus on different aspects of the data. By averaging everything together, you expect that combined result is able to capture all aspects.



Another way of avoiding overfitting (random forest).

TAKE HOME POINTS

1. Be aware of overfitting: your ML should work well on new data.
2. Feature engineer the curse of dimensionality: too many features- feature engineering (look in the room vs apartment)
3. More features are not always a good idea, but more data is always better
4. Combine data with expertise (knowledge of stakeholders)
5. Ensemble many different ML models.

TYPES OF ML Learning Systems

- Whether or not they are trained with human supervision
  1. **Supervised**: the training set you feed to algorithm includes desired solutions called **labels**.
     - A typical supervised learning task is **classification** e.g. spam filter. It is trained with many example of emails along with their **class** (spam or ham) and it must learn how to classify new emails.
     - **Regression**: predict a target numeric value e.g. price of house given a set of **features** called **predictors**. => to train system you need to give it many examples including both their predictors and labels (i.e. their prices).
  2. **Unsupervised**: training data is unlabeled
     - Clustering / hierchical clustering
     - Visualization algorithms
     - Association rule learning
  3. **Semi-supervised**
  4. **Reinforcement learning**: based on rewarding desired behaviours and / or punishing undesired ones -> the learning system called an **agent**, can observe environment, select and peform actions and get **rewards** or **penalties** in return
     - It must learn by itself what is the best strategy called **policy**, to get the most reward over time.

- Whether or not they can learn incrementally on the fly → online
  → **Batch learning**: system incapable of learning incrementally → must be trained using all available data (**offline learning**)

- Whether or not they work by comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model.
  ∴ how they **generalize**

  1. **Instance-based learning**: system learns the examples by heart, then generalizes to new cases using a similarity measure to compare them
  2. **Model-based learning**: build a model of these examples and use them to make predictions

MAIN CHALLENGES OF MACHINE LEARNING

1. **Insufficient quantity of data**

2. **Non-representative training data**

   ↳ By using a non-representative training set, we trained a model that is unlikely to make accurate predictions, especially for very poor or very rich countries.

   ↳ Crucial to use training set that is representative of the cases you want to generalize to.

   ↳ If sample is too small; you will have **sampling noise** (i.e. non representative sample is a result of chance), but even large data sets can be non-representative if the sampling method is flawed: **sampling bias**.

3. **Poor quality data**

   ↳ If training data full of **errors**, **outliers** and **noise** (e.g. due to poor quality measurements) → harder to detect patters ∴ less likely to perform well

4. **Irrelevant Features** → Feature Engineering involves the following steps: 1) Feature selection
   2) " " extraction
   3) creating new fea.

5. **Overfitting the training data** → it means the model performs well on the training data, but it does not generalize well

   ↳ Happens when model is too complex relative to the amount and noisiness of the training data.
   Solutions:
   1) Simplify model by selecting one with fewer features by reducing nº attributes in training data or by constraining model.
   2) Gather more training data.
   3) Reduce noise in training data (e.g. fix data errors and remove outliers)

   ↳ Constraing a model to make it simpler and reduce the risk of overfitting → is called **regularization**.

   ↳ The amount of regularization to apply is controlled with a **hyperparameter** → a parameter of a learning algorithm (not the model) → it must be set prior to train and remains constant during training.

6. **Underfitting training data**: occurs when your model is to simple to learn the underlying structure of of your data. Solutions:
   1) Select more powerful model with ↑ nº parameters.
   2) Feed better features
   3) Reduce the constraints on the model (e.g. reduce the regularization hyperparameter)

↳ Once you train model, to know how it will generalize to new cases is to actually try with new cases: split your data into **training** and **test** set. Error rate in new cases is called the is called the **generalization error** (or **out-of sample error**)

↑ tells you how well your model will perform on instances it has never seen before.

→ If training error low (i.e few mistakes on train set) but generalization error high → model overfitting train data

Test Sit

Data acquisition → Data cleaning → Traing Data → Model training → Model testing → Model Deployment

Adjust hyperparameter

IS it fair to use our single split to evaluate our model performace?
↳ we were given the chance to update the model parameters again and again
↳ To fix this, split in three sets:

1) Training data: used to train model parameters.
2) Validation data: used to determine what model parameters to adjust.
3.) Test data: to get some final peformence metric.

Not allowed to go back to step 1 or 2. This step is so that you test your data with unseen data.

Evaluating Peformance

1. Classification Metrics → Accuracy, Recall, Precision, Recall
↳ Model can either be correct or incorrect in its prediction.

↳ For simplification: binay classification → cat or dog

1. Accuracy: $\frac{n^{o} \text{ of correct predictions}}{\text{total } n^{o} \text{ predictions}}$ ↳ useful when target class is well balanced

2. Recall: ability of a model to find all the relevent cases in a data set.

$$= \frac{n^{o} \text{ of true positives}}{n^{o} TP + n^{o} FN}$$

3. Precision: ability of a model to identify only the relevent data points

$$= \frac{n^{o} \text{ of } TP}{n^{o} \text{ of } TP + n^{o} FP}$$

↳ Trade off between both Precision and Recall

4. F1-Score - combination. Used to achieve optimal blend of both Precision and Recall.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$ → harmonic mean of both (not the average) so that it punishes extreme values.

TRUE Condition	P PREDICTED CONDITION P	N
P	TP	False Negative (Type II error)
N	False positive (Type I error)	TN

Scikit learn → Every algorithm is exposed via an "Estimator". General form:

```
from sklearn.family import Model

from sklearn.linear_model import LinearRegression
```

↓

2) Set hyperparameters → 3) cross_val_score impor train test split

↓

4) Once you split data, we can train/fit our model on the training data

" model.fit [X_train, y_train)

→ Model has been fit and trained on the training data ∴ ready to predict labels or values on test data

5. We get the predicted values using the predict method.

predictions = models.predict [X_test)

6. Evaluate method comping test and train sets.

Hyperparameter Tuning and Model Selection

=> If hesitating between two models, it is common to train both and compare how well they generalize using the test set.

↳ Imagine linear model generalizes better but you want to apply regularization to avoid overfitting

How do you choose value of hyperparameter?

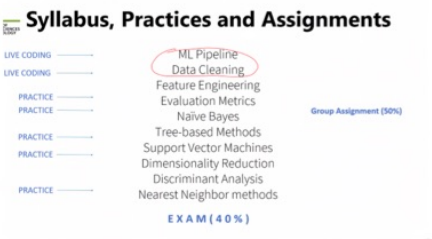
- One option is to train 100 diff. models with 100 diff. values and select one with lowest generalization error (5%.) → you launch into production and produces 15% error
- Problem is that you measured generalization error multiple times on test set and you adapted model and hyperparameters, for that particular set ∴ unlikely to perform well on new data.

↳ Solution: hold out validation: hold out part of training set to evaluate several candidate models (validation set).

↳ Solution: cross validation uses many small validation sets

↓

Each model is evaluated once per validation set after it is trained on the rest of the data. By averaging out all the evaluations of the model, you get a much more accurate measure of its performance



Hyperparameter Tuning and Model Selection

=> If hesitating between two models, it is common to train both and compare how well they generalize using the test set.

↳ Imagine linear model generalizes better but you want to apply regularization to avoid overfitting. How do you choose value of hyperparameter?

- One option is to train 100 diff. models with 100 diff. values and select one with lowest generalization error (5%.) → you launch into production and produces 15% error
- Problem is that you measured generalization error multiple times on test set and you adapted model and hyperparameters, for that particular set ∴ unlikely to perform well on new data.

↳ Solution: hold out validation: hold out part of training set to evaluate several candidate models (validation set).

↳ Solution: cross validation uses many small validation sets

Each model is evaluated once per validation set after it is trained on the rest of the data. By averaging out all the evaluations of the model, you get a much more accurate measure of its performance