




# M01: Introduction to machine learning

➤ Codes	 <a href="#">C1: ML spec</a>
📅 Created	@Jan 12, 2021
➤ Related to Neuroscience Notes (Property)	
▼ Source	Hand-on ML BOOK
☰ Tags	

## Contents ✚

[The Advent of  
Machine Learning  
Wave](#)

[Machine  
Learning  
Problems where  
Machine  
learning can  
be applied](#)  
[Applications  
and Examples  
in Machine  
Learning](#)  
[Types of  
Machine  
Learning  
Systems](#)  
[Small  
Points  
related to  
each  
algorithm](#)

[Main Challenges  
of Machine  
Learning](#)  
[Bad Data](#)

## The Advent of Machine Learning Wave

In 2006, [Geoffrey Hilton et al.](#) published a [paper](#) showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision which was greater than 98 per cent. They had branded this technique as deep learning.

Deep learning neural networks are simplified models of our cerebral cortex, composed of a stack of layers of artificial neurons. Training these networks was considered impossible at that time and most of the researchers ended up abandoning this by the end of 1990s, Although Yann LeCun's deep CNN had worked well for image recognition since that time.

## Machine Learning

Insufficient  
Quantity  
of Training  
Data →  
Non-  
Representative  
Training  
Data  
Poor  
Quality  
Data  
Irrelevant  
Features  
Overfitting  
the  
Training  
data  
Underfitting  
the  
Training  
Data  
Evaluating a  
Machine  
Learning model  
Testing and  
validating  
Hyperparameter  
Tuning and  
model  
selection  
Data  
mismatch  
Takeaway  
Questions

- It is the field of study that gives computers the ability to learn without being explicitly programmed.
  - **Arthur Samuel, 1959**
- 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  $E$ .
  - A more formal engineering-oriented term by **Tom Mitchell, 1997**

Common Machine learning example - Email-Spam filter



Machine learning shines in areas that are either too complex for traditional approaches or have no algorithms. For example, Speech recognition

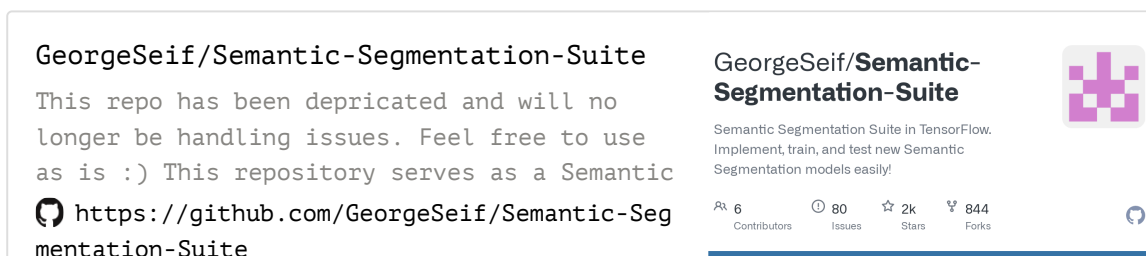
Note → Applying ML techniques to dig into large data can help discover patterns that were not immediately apparent. This is called **Data Mining**.

## Problems where Machine learning can be applied

- Problems for which existing solutions require a lot of fine-tuning or a long list of rules.
- Complex problems for which using a traditional approach yields no good solution
- Fluctuating environments because ML can easily adapt to new data
- Getting insights about complex problems and huge datasets.

## Applications and Examples in Machine Learning

- Analyzing the images of a production line to classify them.
- Tumour Detection in brain scans
  - **Semantic Segmentation** - Semantic segmentation is the task of assigning a class to every pixel in a given image. Note here that this is significantly different from classification. Classification assigns a single class to the whole image whereas semantic segmentation classifies every pixel of the image to one of the classes. This is done to determine the exact location and shape of the tumour.



- Automatic article classification of new articles, this is NLP or Natural Language Processing
- Automatically flagging offensive comments on discussion forums
- Summarizing long docs
- Creating Chatbots - NLP + NLU(Natural language understanding)
- Forecasting revenues and weather etc, using various performance matrices
- Making voice reactive software.
- Credit card frauds
- Recommender Systems
- Intelligent Bots

## Types of Machine Learning Systems

Broad categories of machine learning systems are as follows:

- Based on Human Supervision- **Supervised, Unsupervised, Semi-supervised, Reinforcement Learning.**
- Based on Learning Incrementally - **Online and Batch Learning.**
- Based on detecting data points or data patterns - **instance-based learning, model-based learning.**



Note → These criteria are not exclusive and can be mixed up to develop models. For more information refer to these articles [中 General Concepts of Machine Learning](#) and [article by Jason Brownlee.](#)

## Small Points related to each algorithm

### ▼ Supervised Learning

In Supervised Learning, the training set you feed to the algorithm includes the desired solutions, called labels. This dataset is called the labelled training dataset.

The two most common tasks in supervised learning are - classification and regression.

### ▼ Unsupervised Learning

Most common Unsupervised Learning tasks

#### 1. Clustering

- K-Means
- DBSCAN
- Hierarchical Cluster Analysis(HCA)

#### 2. Anomaly and Novelty detection

- One-class SVM
- Isolation Forest

#### 3. Visualization and dimensionality reduction

- Principal Component Analysis(PCA)
- Kernel PCA
- Locally Linear Embedding (LLE)

- t-Distributed Stochastic Neighbor Embedding(t-SNE)

#### 4. Association rule Learning

- Apriori
- Eclat

Visualization algorithms are also good examples of unsupervised learning.



Zero-Shot Learning Through Cross-Modal Transfer,  
 Proceedings of the 26th International Conference on  
 Neural Information Processing Systems 1(2013): 935-  
 943

**Dimensionality Reduction** - the goal is to simplify data without losing too much information.

**Feature Extraction** - is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process.

#### ▼ Semi-Supervised

Since labelling data is usually time-consuming and costly, we often have unlabeled instances and few labelled instances. Some algorithms are capable of dealing with such data and are called semi-supervised learning algorithms.

Most of these are a combination of supervised and unsupervised learning algorithms. For example, deep belief networks(DBNs) are based on unsupervised algorithms called restricted Boltzmann machines(RBMs) stacked over one another.

#### ▼ Reinforcement learning

In this a learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return(or penalties as negative rewards). It must learn by itself what is the best strategy, called a

policy, to get the most reward over time. A policy defines what action an agent should choose when given a situation.

#### ▼ Online Learning

In Online learning, we train models incrementally by feeding them data instances sequentially, either individually or in small groups, called mini-batches.

This lets the system learn on the fly as the data comes.

### Challenges of Online Learning

- If bad data is fed into the system, the system's performance will gradually decline.

#### ▼ Batch Learning

This system is incapable of learning incrementally: it must be trained using all the available data. This takes lots of computational resources and time and because of that, it is mostly done offline. First, it gets trained then launched into production and runs without learning anymore and applies what it has learned. This is also called **offline learning**.

#### ▼ Instance-Based Learning

The system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples.

**measure of similarity** - By the example of emails, this could be the count of the number of words in common(I know it's a basic example).

#### ▼ Model-Based learning

Another way to generalize a model is to make *predictions*. This is called model-based learning

## Main Challenges of Machine Learning

In machine learning, we have to select an algorithm and train it over some data. So only two things can go wrong:

1. Bad Data
2. Bad Algorithm

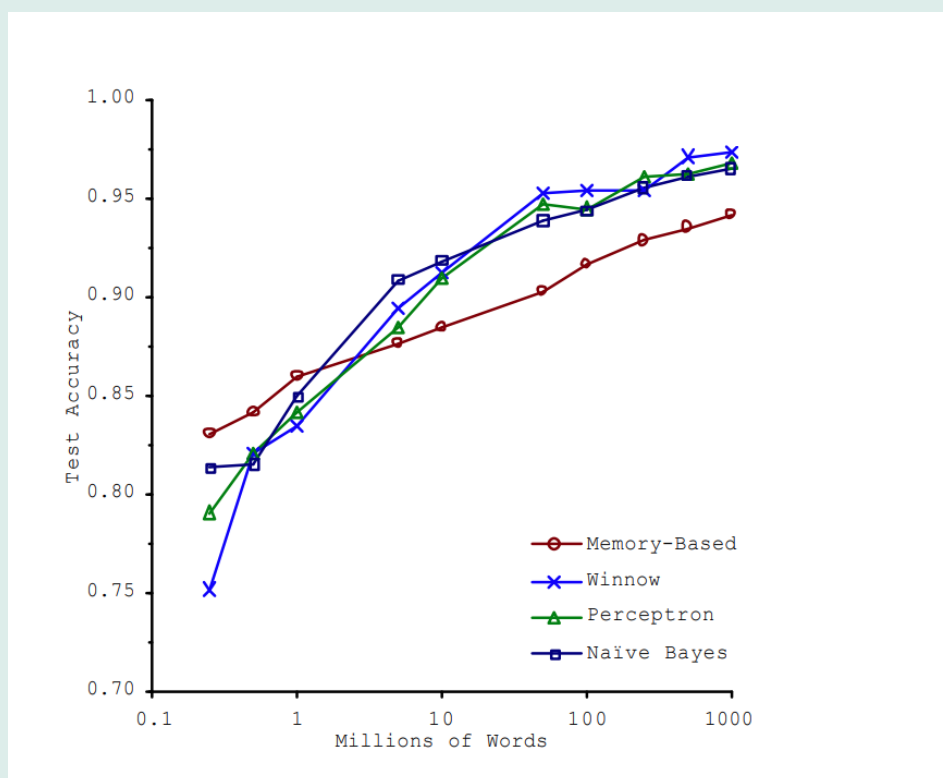
## Bad Data

### Insufficient Quantity of Training Data →

It takes a lot of data for most Machine Learning algorithms to work properly.

#### ▼ The Unreasonable Effectiveness of Data

In 2001, Microsoft researchers **Michele Banko** and **Eric Brill** Scaling to Very Very Large Corpora for Natural Language Disambiguation showed that different ML algorithms, including simple ones, performed almost identically well on a complex problem of natural language disambiguation (Knowing whether to write "to", "two", or "too") once they were given enough data.



The importance of data versus algorithms

These authors further suggested that with the development of algorithms we should also choose to develop the data corpus and spend time and money resources on it.

This idea of giving importance to data was further popularized by **Peter Norvig et al.** in "*The Unreasonable Effectiveness of Data*" published in 2009.

## Non-Representative Training Data

To generalize well, your training data must be representative of new cases you want to generalize to. This is true whether you use instance-based learning or model-based learning.

It is crucial to use a training set that is representative of the cases you want to generalize to. This is often harder than it sounds: if the sample is too small, you will have the sampling noise(that is non-representative data as a chance), but even very large samples can be non-representative of the sampling method is flawed. This is called **sampling bias**.

### ▼ Examples of Sampling Bias

The most famous sampling bias happened during the US presidential election in 1936, which pitted Landon against Roosevelt. For this election, the Literary Digest conducted a very large poll sending 10 mils. people invite and in turn receiving 2.4 mil answers predicting Landon's victory by 57% votes, Instead Roosevelt won with 62% votes (Oooh the burn at that time\*\*). The flaw in this was in the sampling method:

1. The Literary digest used magazines, club memberships lists for mailing lists and most of these are favoured by wealthier people.
2. The pole was answered by less than 25% population originally invited. This introduced a special type of sampling bias, by potentially ruling out the people uninterested in politics or who do not favour Literary digest. This special type of bias is called *nonresponse bias*.



## Poor Quality Data

The training data used for machine learning algorithms should be free of errors, outliers, and noise for the models to make meaningful predictions. For this purpose, most of the machine learning model development process takes significant time in data cleaning. Some of the situations where data cleaning is certainly necessary:

- If some instances are outliers, it may help to simply discard them or try to fix the errors manually
- If some instances are missing a few features. In this case, you can make multiple decisions like:
  - Whether to ignore this attribute altogether.
  - Ignore these instances.
  - Fill in the missing values.
  - Train one model with the features and another one without them.

## Irrelevant Features

A system can only learn about the context when there are sufficient relevant features present in the data and not too many of the features present are "irrelevant".

To counter this we have Feature Engineering, which is a crucial step for the success of any ML model and it involves the following steps:

- **Feature Selection** - Selecting the most useful features to train on among existing features.
- **Feature Extraction** - Combining existing features to produce a more useful one. Dimensionality reduction algorithms can help.
- **Creating new features by gathering new data.**

## Overfitting the Training data

**Overgeneralization** is the act of drawing conclusions that are too broad because they exceed what could be logically concluded

from the available information. This act happens more often than not and Machine Learning algorithms can also fall into the trap of overgeneralization if we are not careful with the data and logistics. This is called **overfitting**: *which means that the model performs well on training data, but it does not generalize well.*

**Overfitting** happens when the model is too complex to the relative amount and noisiness of data. Possible solutions to this are as follows 🙋



- Simplify the model - This can be done by simplifying the model by selecting one with fewer parameters, by reducing the number of attributes in training data through dimensionality reduction, or by constraining the model.
- Gather more training data.
- Reducing the noise in the data.

Constraining a model to make it simpler and reduce the risk of overfitting is called regularization. The amount of regularization can be controlled by a *hyperparameter*. A hyperparameter is a parameter of a learning algorithm(not of the model). As such, it is not affected by the learning algorithms and must be set before training so that it can remain constant during the training.



If you set the regularization parameter to a very large value, it will result in an almost flat model(slope  $\simeq 0$ )

## Underfitting the Training Data

Underfitting is the opposite of overfitting: it happens when the model is too simple to understand the underlying structure of the data.

Some of the solutions to this problem.

- Select a powerful model, with more parameters.
- Feed Better features to the learning algorithm.
- Reduce the constraints on the model(Reduce the regularization parameter).

## Evaluating a Machine Learning model

---

### Testing and validating

For knowing if your models generalize well and give good results there is only one way apart from launching it to production and get your users angry in case it's horrible, we need to test the model on new instances. This step can be done by splitting the data into 2 sets: the *training set* and the *test set*.

The model is trained on the training set and then evaluated over the test set. The error rate on the new cases is called *generalization error*(or *out-of-sample error*), and by evaluating the model on the test set, we get an estimate of this error. It tells us about the model's performance on instances that it has never seen before.



If the training error is low but the generalization error is high, it means that the model is overfitting on the training data.

### Hyperparameter Tuning and model selection

New to machine learning we measure the generalization error multiple times on the test set leading the model and hyperparameters to produce the best model for that particular set, meaning the model will not perform well on new data.

A common solution to this problem is *Holdout validation*: we hold a part of the training set to evaluate several candidate models and select the best one. This set is called a **validation set**. This usually works however if the validation set is too

small, then model evaluations will be imprecise and if the validation model is too large, then the remaining training set will be much smaller than the full training set. This is bad because we will be selecting the best candidate for small data which is similar to selecting a sprinter for a marathon race.

One way to solve this problem is by performing repeated **cross-validation**, using many small validation sets. Here each model is evaluated once per validation set after it is trained on the rest of the data. By averaging the evaluations we get a much more accurate performance measure. However, it comes with the drawback of taking more time as it is multiplied by the numbers of validation sets.

## Data mismatch

In production, there is a high possibility that the data can be different from the training data which can result in our model performing poorly. To counter this situation the validation and test sets must be representative of the data which our model will get in the production phase. After making sure that there are no duplicates or near-duplicates in the sets when splitting the data on validation and test set we can make it a little more representative.

**Note**→ After evaluating the model on the validation set if you get unsatisfactory performance, it will be difficult to know whether the model is overfitting or there is a data mismatch. One solution to this is to hold out some of the training pictures (from the used set) in another set, called the *train-dev set* and after the model is trained on the training set, we can evaluate it on the train-dev set. If it performs well, then it is not overfitting the training set. However, if it performs poorly on the validation set then the problem is coming from data mismatch and can be solved by transforming the data used to represent the real-world data more closely, and then train the model. Conversely, if the model is performing poorly on train-dev then it is overfitting and regularization should be done.

### ▼ No Free Lunch Theorem

A model is simplified version of the observations. The simplifications are meant to discard the superfluous details that are unlikely to generalize well to new instances, for this *assumptions* must be made.

In a famous paper David Wolpert demonstrated that if no assumptions are made then there is no reason for giving preference to one model over another. This is called **No Free Lunch(NFL)** theorem. For some datasets the best model is a linear model, while for other datasets it is a neural network. There is no model that is a *priori* guaranteed to work better. The only way to be sure is to evaluate the model over the data which is practically(in project-timeline and funding constraints) impossible, in practice we make some reasonable assumptions about the data and evaluate only a few reasonable models.

## Takeaway Questions

### ▼ How would you define machine learning?

- It is the field of study that gives computers the ability to learn without being explicitly programmed.
  - **Arthur Samuel, 1959**
- 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  $E$ .
  - A more formal engineering-oriented term by **Tom Mitchell, 1997**

## Appendix A Solution

Machine learning is about building systems that can learn from data. Learning means getting better at some task, given some performance measure.

### ▼ Name 4 types of problem where ML shines?

- Problems for which existing solutions require a lot of fine-tuning or a long list of rules.

- Complex problems for which using a traditional approach yields no good solution
- Fluctuating environments because ML can easily adapt to new data
- Getting insights about complex problems and huge datasets.

## Appendix A Solution

ML is great for complex problems for which we have no algorithmic solution, to replace long lists of hand-tuned rules, to build systems that adapt to fluctuating environments, and finally to help human learn. Example - data mining

### ▼ What is a labelled training set?

In Supervised Learning, the training set you feed to the algorithm includes the desired solutions, called labels. This dataset is called the labelled training dataset.

## Appendix A Solution

A labelled training set is a set that contains the desired solutions(a.k.a. a label) for each instance.

### ▼ What are the two most common supervised tasks?

They are classification and regression.

### ▼ Can you name 4 common Unsupervised learning tasks?

Most common Unsupervised Learning tasks

#### 1. Clustering

- K-Means
- DBSCAN
- Hierarchical Cluster Analysis(HCA)

#### 2. Anomaly and Novelty detection

- One-class SVM
- Isolation Forest

### 3. Visualization and dimensionality reduction

- Principal Component Analysis(PCA)
- Kernel PCA
- Locally Linear Embedding (LLE)
- t-Distributed Stochastic Neighbor Embedding(t-SNE)

### 4. Association rule Learning

- Apriori
- Eclat

## Appendix A Solution

Common tasks include Clustering Visualization, dimensionality reduction and association rule learning

▼ What type of Machine learning algorithm would you use to allow a robot to walk in various unknown terrains?

Many robots implement reinforcement learning algorithms to learn how to walk.

## Appendix A Solution

Reinforcement learning is likely to perform best if we want a robot to walk on various unknown terrains since this is typically the type of problem that RL tackles. It might be possible to express the problem supervised or unsupervised but it would be less natural.

▼ What type of algorithm would you use to segment your customers into multiple groups?

If you don't know the groups then clustering - unsupervised learning

if you know the groups then - multiclass classification

## Appendix A Solution

If you don't know the groups, then we can use a clustering algorithm to segment customers into clusters of similar customers. However, if you know the groups then we can feed

many examples of each group to a classification algorithm, and it will classify all your customers into these groups.

▼ Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?

I would frame it as supervised learning.

## Appendix A Solution

It is a typical supervised learning problem: the algorithm is fed by emails along with labels(spam or not spam)

▼ What is an online learning system?

In Online learning, we train models incrementally by feeding them data instances sequentially, either individually or in small groups, called mini-batches.

This lets the system learn on the fly as the data comes.

## Appendix A Solution

An online learning system can learn incrementally, as opposed to a batch learning system. This makes it capable of adapting rapidly to both changing data and autonomous systems, and of training on very large quantities of data.

▼ What is out-of-core learning?

Online learning algorithms can also be used to train systems on huge datasets that cannot fit in one machine's main memory and this is called out-of-core learning.

## Appendix A solution

Out-of-core algorithms can handle vast quantities of data that cannot fit in a computer's main memory. An out-of-core learning algorithm chops the data into mini-batches and uses a similarity measure to find the most similar learned instances and uses them to make predictions.

▼ What type of learning algorithm relies on a similarity measure and a learning algorithm's hyperparameters?



Instance-based learning algorithms rely on similarity measure and learning algorithm's hyperparameter

## Appendix A Solution

An instance-based learning system learns the training data by heart; then, when given a new instance, it uses a similarity measure to find the most similar learned instances and uses them to make predictions.

▼ What is the difference between a model parameter and a learning algorithm's hyperparameter?

### PARAMTERS

A model parameter is a configuration variable that is internal to the model and while value can be estimated from the data

- required by the model while making predictions
- Its value define skill of model on given data
- estimated or learned from the data
- not set manually
- saved as a part of the model.

### HYPERPARAMETERS

Model hyperparameter is a configuration that is, external to the model and whose value can not be estimated from the data.

- used in the process of estimation of model parameters
- specified by practitioner
- can often be set using heuristics
- often tuned for a given prediction modeling problem.

## Appendix A Solution

A model has one or more model parameter that determines what it will predict given a new instance(e.g., the slope of the linear model). A learning algorithm tries to find optimal values for these parameters such that the model generalizes well to new instances. A hyperparameter is a parameter of learning algorithm, not of the model.(e.g., the amount of regularization to apply.)

▼ What do model-based learning algorithms search for? what is the most common strategy they use to succeed? How do they make predictions?

## Appendix A Solution

Model-based learning algorithms search for an optimal value for the model parameters such that the model will generalize well to new instances. We usually train such systems by minimizing a cost function that measures how bad the systems is at making predictions on the training data, plus a penalty for model complexity if the model is regularized. To make predictions, we feed the new instance's features into the model's prediction function, using the parameter values found by the learning algorithm.

▼ Can you name four of the main challenges in machine learning?

### Challenges

- Lack of data
- poor data quality
- Non-representative data
- Uninformative data features
- Excessively simple models(underfit)
- Excessively complex models(overfit).

## Appendix A Solution

Some main challenges in ML are lack of data, poor data quality, non-representative data, uninformative features, excessively simple models that underfit or excessively complex data that overfit the data.

▼ If your model performs great on the training data but generalizes poorly to the new instances, what is happening? can you name three possible solutions?

### Overfitting 🙅

- Getting more data

- Regularizing the model
- Reducing noise in the data

## Appendix A Solution

If a model performs great on training data but generalizes poorly to new instances, the model is likely overfitting the training data (or we got extremely lucky on the training data). Possible solutions to overfitting are getting more data, simplifying the model (selecting a simpler algorithm, reducing the number of parameters or feature used, or regularizing the model), or reducing the noise on the training data.

### ▼ What is a test set, and why would you want to use it?

set used to estimate generalization error on new instances, before releasing the model for production

## Appendix A Soloution

A test set is used to estimate the generalization error that a model will make on new instances, before the model is launched in production.

### ▼ what is the purpose of a validation set?

A validation set is used to compare models or test models while training.

Mostly used for model selection and tuning hyperparameters.

## Appendix A Solution

A validation set is used to compare models . It makes it possible to select the best model and tune hyperparameters.

### ▼ What is the train-dev set, when do you need it, and how do you use it?

The train-dev set is used when there is a risk of data mismatch between training and validation set. The train-dev set is part of training set that's held out. The model performs well on training set but no on train-dev set, it is overfitting the training set. If it performs well on train-

dev and training but not on validation, then there is Significant data mismatch between training and validation + test set. Solution is to improve training data to look like validation + test dataset.

## Appendix Solution A

The train-dev set is used when there is a risk of mismatch between the training data and the data used in the validation and test datasets(which should always be as close as possible to the data used once the model is in production). The train-dev set is a part of the training set that's held out(the model is not trained on it). The model is trained on the rest of the training set, and evaluated on both the train-dev set and the validation set, then the model is likely overfitting the training set. If it performs well on both training set and the train-dev set, but not on the validation, then there is probably a significant data mismatch between the training data and the validation set + test data, and you should try to improve the training data to make it look more like the validation + test data.

### ▼ What can go wrong if you tune hyperparameters using the test set?

If we tune hyperparameters using test set, we risk overfitting test set which will give wrong generalization error.

## Appendix A solution

If you tune hyperparameters using the test set, you risk overfitting the test set, and the generalization error you measure will be optimistic (you may launch a model that performs worse than you expect)