
Course: CM-072

Presentation 1

Lists of topics

- What is machine learning?
 - Supervised Learning
 - Unsupervised Learning
 - Steps of Machine Learning
 - KDD process. Crisp DM
 - Libraries in Python
 - Scikit-learn API
 - Example:k-nearest neighbors algorithms
 - Generalization, overfitting, underfitting
 - Examples.
-

What is machine learning?


Frase de Arthur Samuel, AI pioneer.

"A field of study that gives computers the ability to learn without being explicitly programmed." (1959)

What is machine learning?

Definition de Yann Le Cunn :

 **MACHINELEARNING** **comments** related other discussions (4)

 **AMA: Yann LeCun** (self.MachineLearning)
submitted 5 months ago * by ylecun

My name is Yann LeCun. I am the Director of Facebook AI Research and a professor at New York University. Much of my research has been focused on deep learning, convolutional nets, and related topics.

Seriously, I don't like the phrase "Big Data". I prefer "**Data Science**", which is the **automatic (or semi-automatic) extraction of knowledge from data**. That is here to stay, it's not a fad. The amount of data generated by our digital world is growing exponentially with high rate (at the same rate our hard-drives and communication networks are increasing their capacity). But the amount of human brain power in the world is not increasing nearly as fast. This means that now or in the near future **most of the knowledge in the world will be extracted by machine and reside in machines**. It's inevitable. En entire industry is building itself around this, and a new academic discipline is emerging.

What is machine learning?

Tom Mitchell (1998)

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

Wikipedia (2016)

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.

What is machine learning?

One definition: “Machine learning is the semi-automatic extraction of knowledge from data.”

- **Knowledge from data:** Starts with a question that might be answerable using data.
- **Automatic extraction:** A computer provides the insight.
- **Semi-automatic:** Requires many smart decisions by a human.

Data Mining vs. Machine Learning

In Big Data Analytics, data mining and machine learning are the two most commonly used techniques. Learners get confused between the two but they are two different approaches used for two different purposes. Let us see the difference between data mining and machine learning.

- **Data mining** is the process of identifying patterns in large amounts of data to extract useful information from those patterns. It may include techniques of artificial intelligence, machine learning, neural networks, and statistics. The basis of data mining is real world data. It may have taken inspiration and techniques from machine learning and statistics but is put to different ends.
- **Machine learning** process is an approach to developing artificial intelligence. We use Machine Learning algorithm to develop the new algorithms and techniques. These allow the machine to learn from the analyzed data or with experience. Most tasks that need intelligence must have an ability to induce new knowledge from experiences. Thus, a large area within AI is machine learning.

Types of machine learning

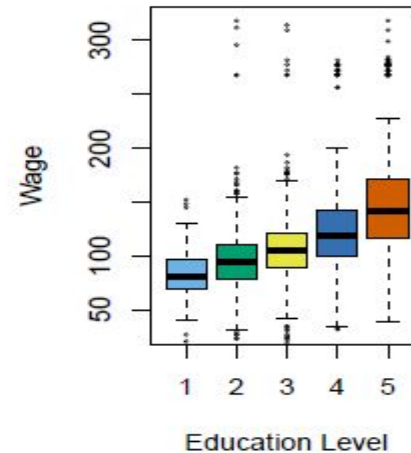
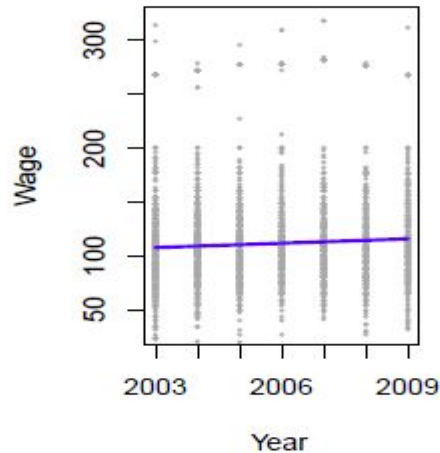
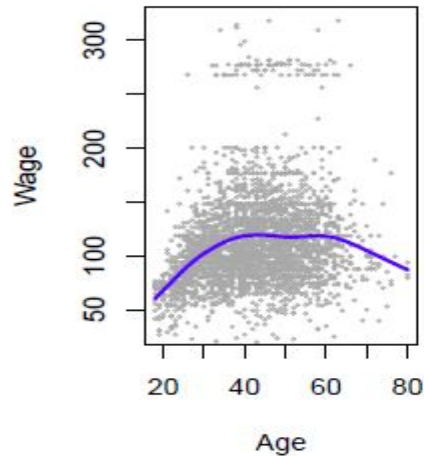
There are two main categories of machine learning: **supervised learning** and **unsupervised learning**.

Supervised learning (“predictive modeling”):

- Predict an outcome based on input data.
- Example: predict whether an email is spam or ham.
- **Goal is “generalization”.**

Supervised Learning

Predict salary using demographic data

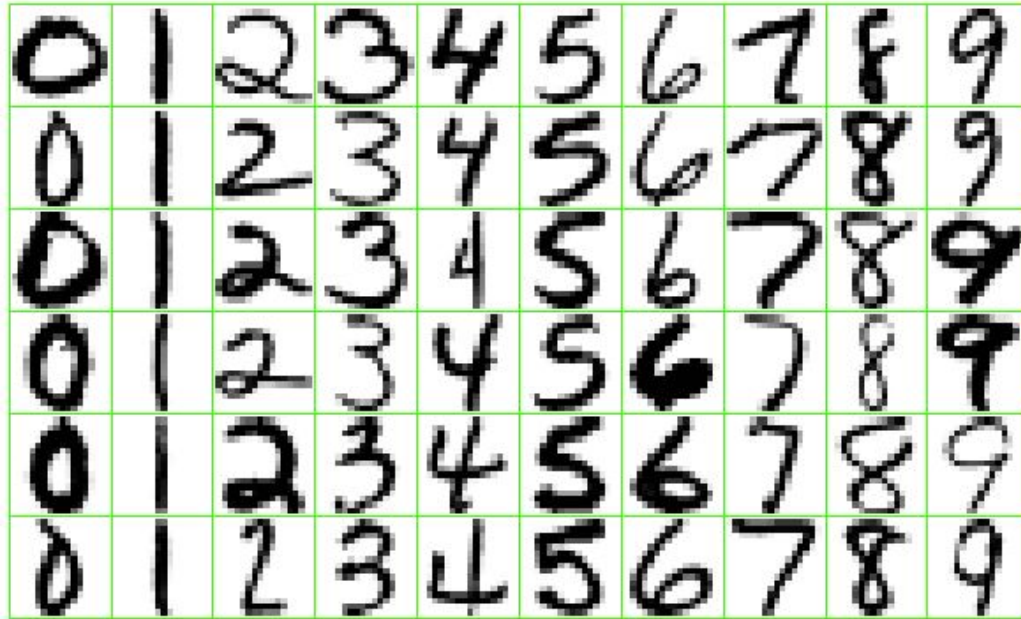


Income survey data for males from the central Atlantic Region of USA.

<https://class.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/introduction.pdf>

Supervised Learning

Identify the numbers in a handwritten zip code



<https://class.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/introduction.pdf>

Supervised Learning

There are two categories of supervised learning:

Regression

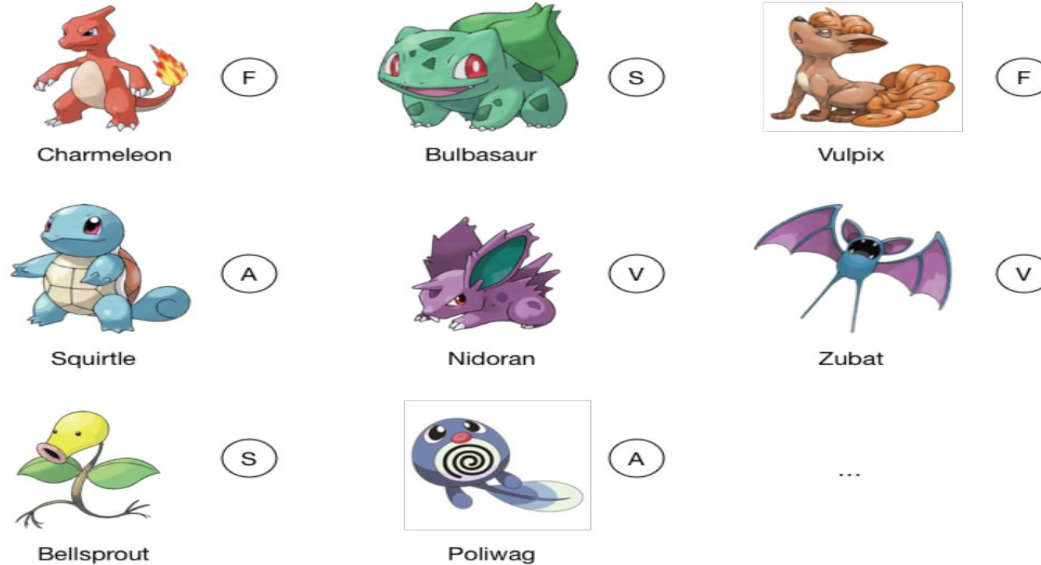
- Outcome we are trying to predict is continuous.
- Examples: price, blood pressure.

Classification

- Outcome we are trying to predict is categorical (values in a finite set).
- Examples: spam/ham, cancer class of tissue sample.

Supervised Learning: classification

In the following problem, the agent (apprentice) begins with observe a series of individuals classified as: water (A), seed (S), poison (V) or fire (F):



Supervised Learning: classification

After the phase of training, the agent's ability is measured by its ability to correctly classify new individuals (performance).



Diglett



Psyduck



Oddish



Regression or Classification?

Problem: Children born prematurely are at high risk of developing infections, many of which are not detected until after the baby is sick .

Goal: Detect subtle patterns in the data that predicts infection before it occurs.

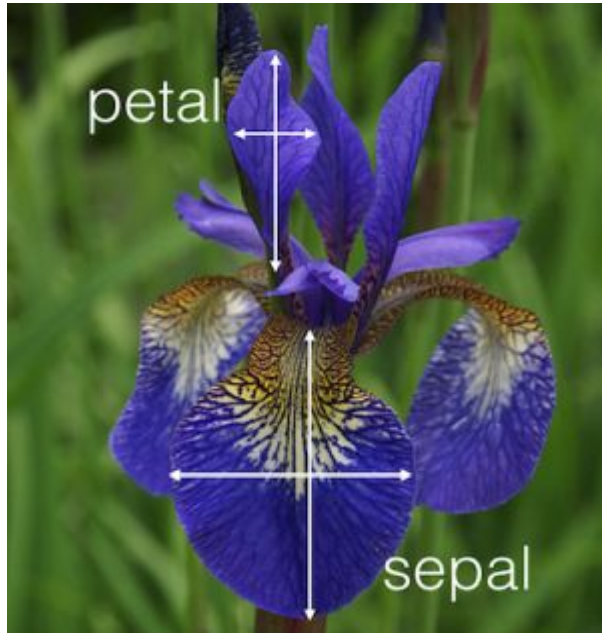
Data: 16 vital signs such as heart rate, respiration rate, blood pressure, etc...

Impact: Model is able to predict the onset of infection 24 hours before the traditional symptoms of infection appear.



<http://www.amazon.com/Big-Data-Revolution-Transform-Think/dp/0544002695>

Regression or Classification?



Fisher's *Iris* Data

Sepal length ↕	Sepal width ↕	Petal length ↕	Petal width ↕	Species ↕
5.1	3.5	1.4	0.2	<i>I. setosa</i>
4.9	3.0	1.4	0.2	<i>I. setosa</i>
4.7	3.2	1.3	0.2	<i>I. setosa</i>
4.6	3.1	1.5	0.2	<i>I. setosa</i>
5.0	3.6	1.4	0.2	<i>I. setosa</i>
5.4	3.9	1.7	0.4	<i>I. setosa</i>
4.6	3.4	1.4	0.3	<i>I. setosa</i>
5.0	3.4	1.5	0.2	<i>I. setosa</i>

<https://archive.ics.uci.edu/ml/datasets/iris>

Machine Learning Terminology

150 observations
($n = 150$)

Feature matrix “X” has
n rows and p columns

Response “y” is a
vector with length n

Fisher's *Iris* Data

Sepal length ⚡	Sepal width ⚡	Petal length ⚡	Petal width ⚡	Species ⚡
5.1	3.5	1.4	0.2	<i>I. setosa</i>
4.9	3.0	1.4	0.2	<i>I. setosa</i>
4.7	3.2	1.3	0.2	<i>I. setosa</i>
4.6	3.1	1.5	0.2	<i>I. setosa</i>
5.0	3.6	1.4	0.2	<i>I. setosa</i>
5.4	3.9	1.7	0.4	<i>I. setosa</i>
4.6	3.4	1.4	0.3	<i>I. setosa</i>
5.0	3.4	1.5	0.2	<i>I. setosa</i>

4 features ($p = 4$)

response

Machine Learning Terminology

- **Observations** are also known as: samples, examples, instances, records.
- **Features** are also known as: predictors, independent variables, inputs, regressors, covariates, attributes.
- **Response** is also known as: outcome, label, target, dependent variable.
- **Regression problems** have a continuous response.
- **Classification problems** have a categorical response. The type of supervised learning problem has nothing to do with the features!.

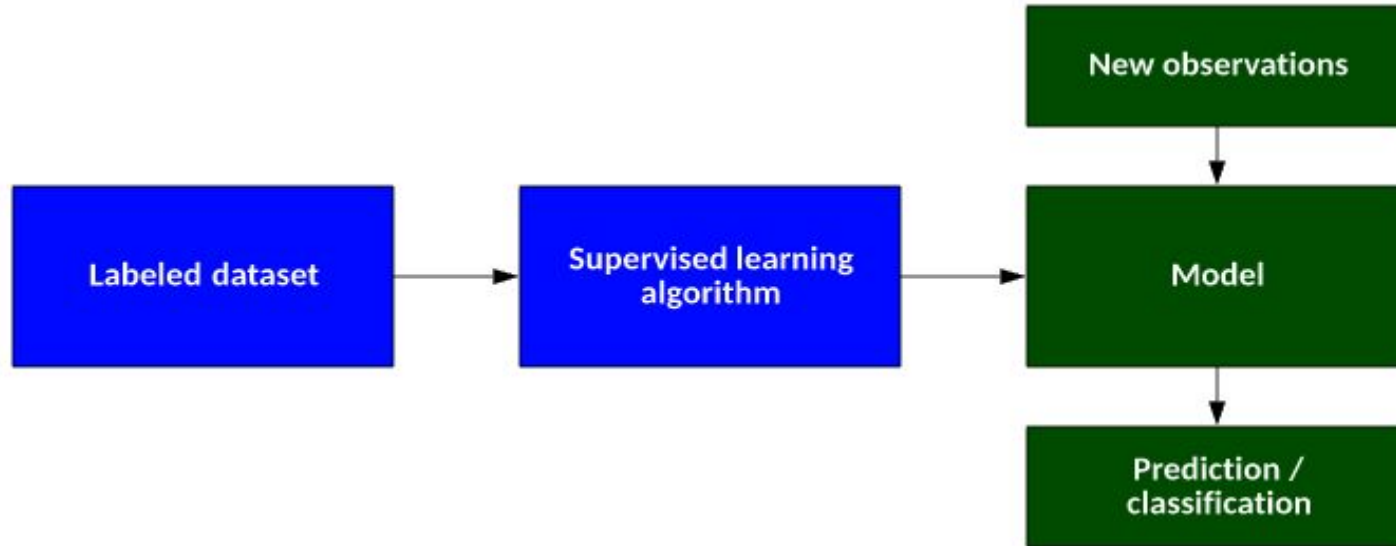
How does Supervised Learning work?

The steps that are carried out in supervised learning models are:

1. **Train a machine learning model using labeled data**
 - **“Labeled data” is data with a response variable.**
 - **“Machine learning model” learns the relationship between the features and the response.**
2. **Make predictions on new data for which the response is unknown.**
 - **The primary goal of supervised learning is to build a model that “generalizes”: It accurately predicts the future rather than the past!.**

How does Supervised Learning work?

The supervised learning algorithm uses a labeled data set to produce a model. You can then use this model with new data to validate the model's accuracy or in production with live data.



Example of Supervised Learning

Example: Coin classifier

- **Observations:** Coins.
- **Features:** Size and mass.
- **Response:** Coin type, hand-labeled.

1. **Train a machine learning model using labeled data**
 - Model learns the relationship between the features and the coin type.
2. **Make predictions on new data for which the response is unknown**
 - Give it a new coin, predicts the coin type automatically.

Unsupervised Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

The characteristics of unsupervised learning are:

- Extracting structure from data.
- Example: segment grocery store shoppers into “clusters” that exhibit similar behaviors.
- Goal is “representation”.

Unsupervised Learning

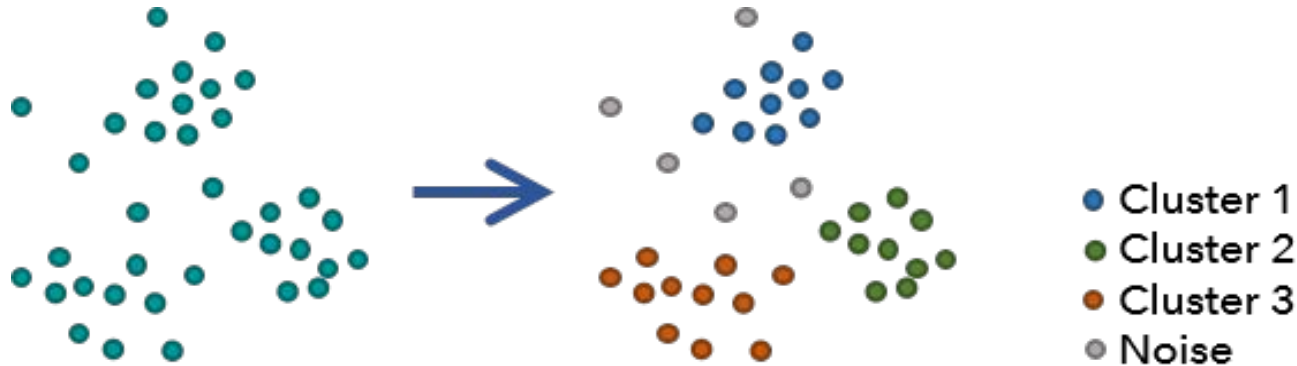
Common types of unsupervised learning:

- **Clustering:** group “similar” data points together.
- **Dimensionality Reduction:** reduce the dimensionality of a dataset by extracting features that capture most of the variance in the data.
- **NFE y manifold learning.**

Unsupervised Learning: Clustering

Clustering characteristics:

- Try to group individuals of a sample into groups.
- Unsupervised learning method (no information a-priori).
- Also know as: clustering, re-grouping, unsupervised classification.



Unsupervised Learning: Clustering

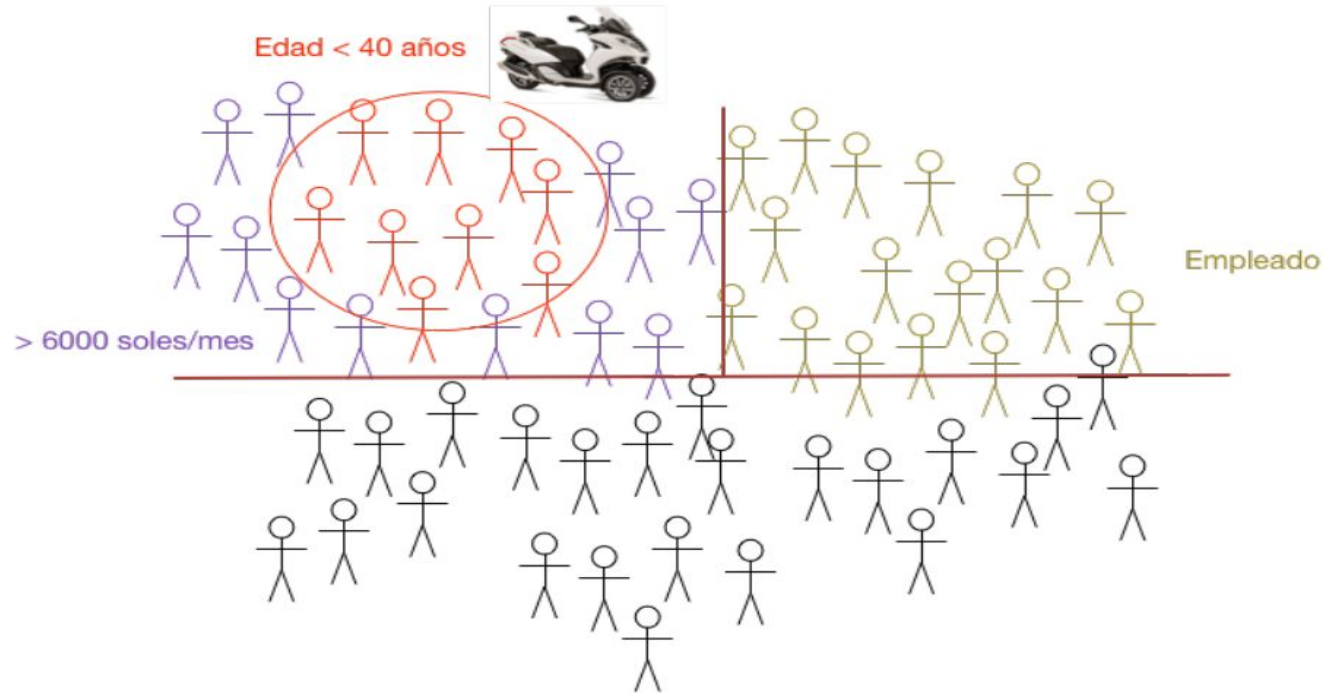
Example:



- Divide the sample into two groups: employed/unemployed.
- From employed, select those people who earn more than 6,000 USD/mes.
- We can also divide the sub-group by age, keeping only people who are less than 40 years old.

Unsupervised Learning: Clustering

Outcome:



Unsupervised Learning

Unsupervised learning has some clear differences from supervised learning. **With unsupervised learning:**

- **There is no clear objective.**
- **There is no “right answer” (hard to tell how well you are doing).**
- **There is no response variable, just observations with features.**
- **Labeled data is not required.**

Unsupervised Learning

Unsupervised learning example: Coin clustering

- **Observations:** Coins.
- **Features:** Size and mass.
- **Response:** There isn't one (no hand-labeling required!).

Perform unsupervised learning

- Cluster the coins based on “similarity”.
- You're done!.

Sometimes, unsupervised learning is used as a “preprocessing” step for supervised learning. (How?).

Step of Machine Learning

Example:

Let's pretend that we've been asked to create a system that answers the question of whether a drink is wine or beer. This question answering system that we build is called a **"model"**, and this model is created via a process called **"training"**. The goal of training is to create an accurate model that answers our questions correctly most of the time. But in order to train a model, we need to collect data to train on.

Our data will be collected from glasses of wine and beer. There are many aspects of the drinks that we could collect data on, everything from the amount of foam, to the shape of the glass.

For our purposes, we'll pick just two simple ones: The color (as a wavelength of light) and the alcohol content (as a percentage). The hope is that we can split our two types of drinks along these two factors alone. We'll call these our **"features"** from now on: color, and alcohol.

Step of Machine Learning

Gathering data:

Once we have our equipment and booze, it's time for our first real step of machine learning: gathering data. This step is very important because the quality and quantity of data that you gather will directly determine how good your predictive model can be. In this case, the data we collect will be the color and the alcohol content of each drink. This will yield a table of color, alcohol%, and whether it's beer or wine. This will be our training data.

Color (nm)	Alcohol %	Beer or Wine?
610	5	Beer
599	13	Wine
693	14	Wine

Step of Machine Learning

Data preparation:

Is where we load our data into a suitable place and prepare it for use in our machine learning training. We'll first put all our data together, and then randomize the ordering. We don't want the order of our data to affect what we learn, since that's not part of determining whether a drink is beer or wine.

We'll also need to split the data in two parts. The first part, used in training our model, will be the majority of the dataset. The second part will be used for evaluating our trained model's performance. We don't want to use the same data that the model was trained on for evaluation, since it could then just memorize the "questions".

Sometimes the data we collect needs other forms of adjusting and manipulation. Things like **duplication, normalization, error correction**, and more. These would all happen at the data preparation step.

Step of Machine Learning

Choosing a model:

The next step in our workflow is choosing a model. There are many models that researchers and data scientists have created over the years. Some are very well suited for image data, others for sequences (like text, or music), some for numerical data, others for text-based data.

- k-Nearest Neighbour (kNN)
- Linear Regression
- Naive Bayes
- Classification and Regression Tree (CART)

In our case, since we only have 2 features, color and alcohol%, we can use a small linear model, which is a fairly simple one that should get the job done.

Step of Machine Learning

Training:

In this step, we will use our data to incrementally improve our model's ability to predict whether a given drink is wine or beer. We will do this on a much smaller scale with our drinks. In particular, the formula for a straight line is $y=m*x+b$ and the values we have available to us for adjusting, or **"training"**, are m and b .

In machine learning, there are many m 's since there may be many features. The collection of these m values is usually formed into a matrix, that we will denote W , for the **"weights"** matrix. Similarly for b , we arrange them together and call that the biases.

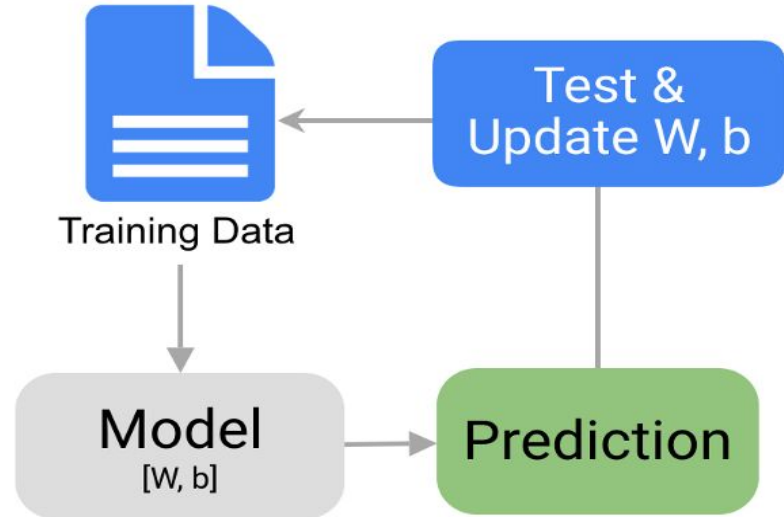
The **training process** involves initializing some random values for W and b and attempting to predict the output with those values. It does pretty poorly. But we can compare our model's predictions with the output that it should produced, and adjust the values in W and b such that we will have more correct predictions.

Step of Machine Learning

Training:

This process then repeats. Each iteration or cycle of updating the weights and biases is called one training “step”.

Let's look at what that means in this case, more concretely, for our dataset. When we first start the training, it's like we drew a random line through the data. Then as each step of the training progresses, the line moves, step by step, closer to an ideal separation of the wine and beer.



Step of Machine Learning

Evaluation:

Once training is complete, it's time to see if the model is any good, using **evaluation**. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

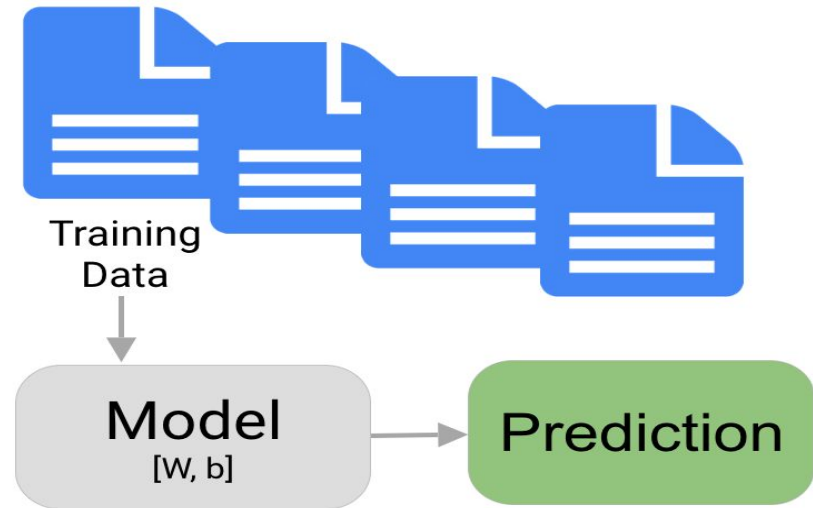
A good rule of thumb I use for a **training-evaluation** split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don't need as big of a fraction for the evaluation dataset.

Step of Machine Learning

Parameter tuning:

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

One example is how many times we run through the training dataset during training. What I mean by that is we can “show” the model our full dataset multiple times, rather than just once. This can sometimes lead to higher accuracies.



Step of Machine Learning

Parameter tuning:

For more complex models, initial conditions can play a significant role in determining the outcome of training. Differences can be seen depending on whether a model starts off training with values initialized to zeros versus some distribution of values, which leads to the question of which distribution to use.

There are many considerations at this phase of training, and it's important that you define what makes a model “good enough”, otherwise you might find yourself tweaking parameters for a very long time.

These parameters are typically referred to as “**hyperparameters**”. The tuning, of these hyperparameters, remains a bit of an art, and is more of an experimental process that heavily depends on the specifics of your dataset, model, and training process. Once you're happy with your training and hyperparameters, guided by the evaluation step, it's time to finally use your model to do something useful!.

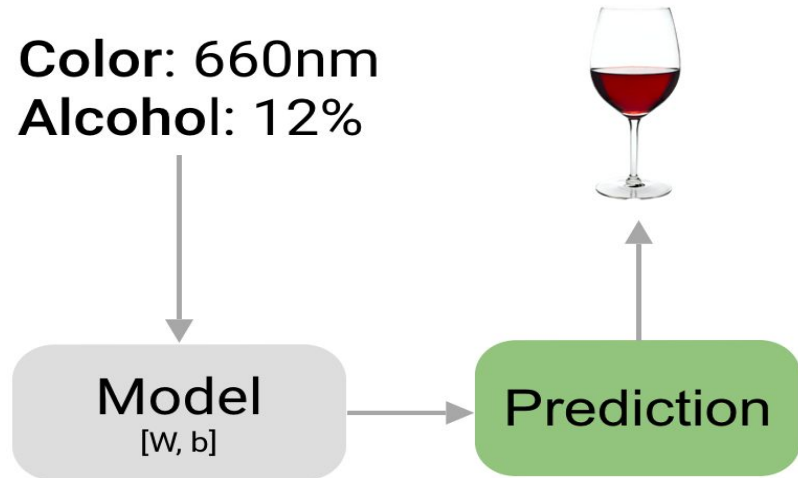
Step of Machine Learning

Prediction:

Machine learning is using data to answer questions. So Prediction, or inference, is the step where we get to answer some questions. This is the point of all this work, where the value of machine learning is realized.

We can finally use our model to predict whether a given drink is wine or beer, given its color and alcohol percentage.

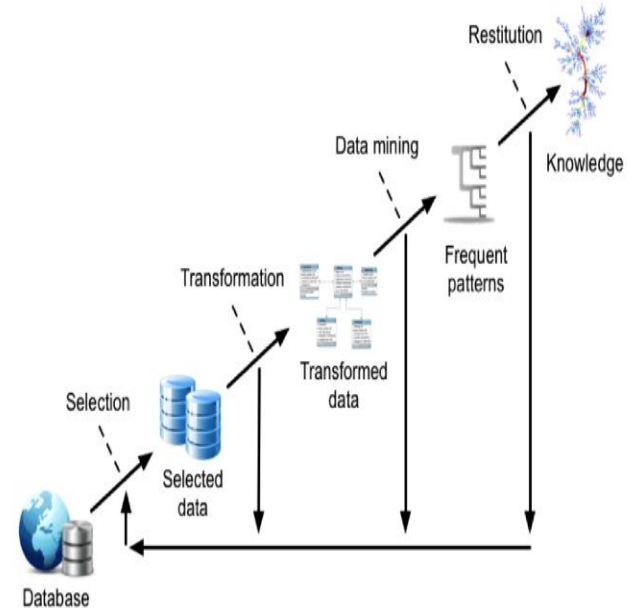
[Fuente: The 7 Steps of Machine Learning-Yufeng G](#)



KDD Process

Summary:

- Iterative and interactive multi-step process to transform huge databases into knowledge.
- Dataset. Heterogeneous: transactional databases, textual, images, complex structures, etc.
- Selecting data. Interaction with experts to select interesting data->domain.
- Pre-processing: Transform the data in order to apply some Data Mining technique.
- Data mining: Extract interesting/unusual/...from data.



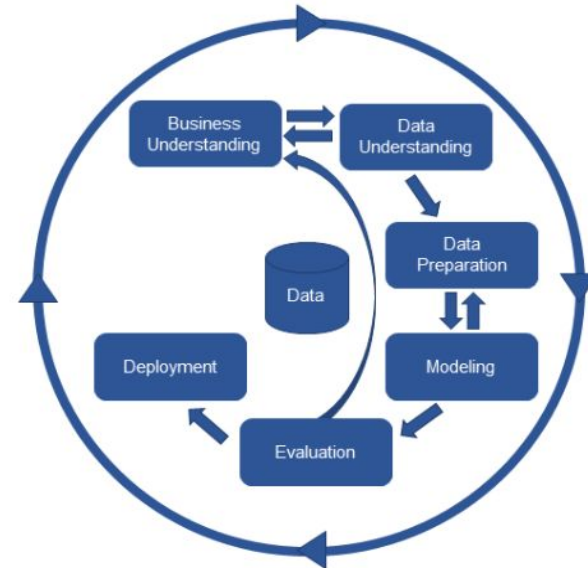
Crisp DM

De wikipedia:

Cross-industry standard process for data mining, known as CRISP-DM, is an open standard process model that describes common approaches used by data mining experts. It is the most widely-used analytics model.

This model is an idealised sequence of events. In practice many of the tasks can be performed in a different order and it will often be necessary to backtrack to previous tasks and repeat certain actions. The model does not try to capture all possible routes through the data mining process.

<https://www.kdnuggets.com/2017/01/four-problem-crisp-dm-fix.html>



Libraries in Python

Tools:

- **NumPy** facilitates easy and efficient numerical computation. It has many other libraries built on top of it. Make sure to learn NumPy arrays.
- **Pandas** One library built on top of NumPy is Pandas. It comes in handy with data structures and exploratory analysis. Another important feature it offers is DataFrame, a 2-dimensional data structure with columns of potentially different types.
- **Scikit-learn** is the primary library for machine learning. It has algorithms and modules for pre-processing, cross-validation, and other such purposes. Some of the algorithms deal with regression, decision trees, ensemble modeling, and non-supervised learning algorithms like clustering.
- **Matplotlib:** Is a flexible plotting and visualization library. However, it is cumbersome, so, you may go for Seaborn instead.
- **Seaborn:** With Seaborn, it is easier than ever to plot common data visualizations. It is built on top of Matplotlib and offers a more pleasant, high-level wrapper. You should learn effective data visualization.

Scikit-learn Estimator API

The Scikit-Learn API is designed with the following guiding principles in mind, as outlined in the [Scikit-Learn API paper](#):

- **Consistency:** All objects share a common interface drawn from a limited set of methods, with consistent documentation.
- **Inspection:** All specified parameter values are exposed as public attributes.
- **Limited object hierarchy:** Only algorithms are represented by Python classes; datasets are represented in standard formats (NumPy arrays, Pandas DataFrames, SciPy sparse matrices) and parameter names use standard Python strings.
- **Composition:** Many machine learning tasks can be expressed as sequences of more fundamental algorithms, and Scikit-Learn makes use of this wherever possible.
- **Sensible defaults:** When models require user-specified parameters, the library defines an appropriate default value.

Basic of API Scikit-Learn

Most commonly, the steps in using the Scikit-Learn estimator API are as follows:

1. Choose a class of model by importing the appropriate **estimator class** from Scikit-Learn.
2. Choose **model hyperparameters** by instantiating this class with desired values.
3. Arrange data into a features matrix and target vector following the discussion above.
4. Fit the model to your data by calling the **fit() method** of the model instance.
5. Apply the Model to new data:
 - For supervised learning, often we predict labels for unknown data using the **predict() method**.
 - For unsupervised learning, we often transform or infer properties of the data using the **transform()** or **predict() method**.

Data Representation in Scikit-learn

Machine learning is about creating models from data:

- **Data as table:** A basic table is a two-dimensional grid of data, in which the rows represent individual elements of the dataset, and the columns represent quantities related to each of these elements. we will refer to the columns of the matrix as *features*, and the number of columns as *n_features*.

```
In [1]: import seaborn as sns  
iris = sns.load_dataset('iris')  
iris.head()
```

Out[1]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Data Representation in Scikit-learn

Machine learning is about creating models from data:

- **Feature matrix:** The features matrix is assumed to be two-dimensional, with shape `[n_samples, n_features]`, and is most often contained in a NumPy array or a Pandas DataFrame, though some Scikit-Learn models also accept SciPy sparse matrices. Se almacena en una variable X.
 - The **samples** (i.e., rows) always refer to the individual objects described by the dataset. For example, the sample might be a flower, a person, a document, an image, a sound file, a video, an astronomical object, or anything else you can describe with a set of quantitative measurements.
 - The **features** (i.e., columns) always refer to the distinct observations that describe each sample in a quantitative manner. Features are generally real-valued, but may be Boolean or discrete-valued in some cases.

Data Representation in Scikit-learn

Machine learning is about creating models from data:

- **Target array:** is usually one dimensional, with length **n_samples**, and is generally contained in a NumPy array or Pandas Series. The target array may have continuous numerical values, or discrete classes/labels. Usualmente se llama **y**.
 - Often one point of confusion is how the target array differs from the other features columns. The distinguishing feature of the target array is that it is usually the quantity we want to **predict from the data**: in statistical terms, it is the **dependent variable**.

```
In [3]: X_iris = iris.drop('species', axis=1)
        X_iris.shape
```

```
Out[3]: (150, 4)
```

Resultado con el
conjunto de datos
Iris.

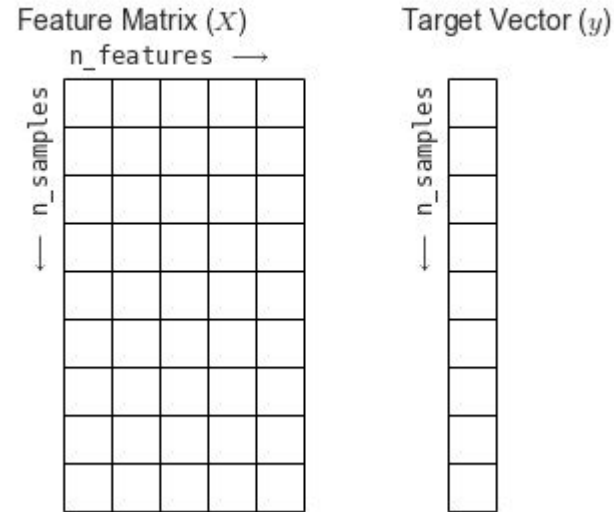
```
In [4]: y_iris = iris['species']
        y_iris.shape
```

```
Out[4]: (150,)
```

Data Representation in Scikit-learn

Machine learning is about creating models from data:

To summarize, the expected layout of features and target values is visualized in the following diagram-->



K-nearest neighbors (KNN) classification

Algorithm:

1. Pick a value for K.
2. Search for the K observations in the data that are "nearest" to the measurements of the unknown iris.
 - Euclidean distance is often used as the distance metric, but other metrics are allowed.
3. Use the most popular response value from the K "nearest neighbors" as the predicted response value for the unknown iris.

Question: What's the "best" value for K in this case?

Answer: The value which produces the most accurate predictions on **unseen data**. We want to create a model that generalizes!.

Example:k-nearest neighbors

Review of the Iris dataset:

```
In [1]: # read the iris data into a DataFrame
import pandas as pd
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
col_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
iris = pd.read_csv(url, header=None, names=col_names)
```

Feature matrix:

```
In [8]: # store feature matrix in "X"
feature_cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
X = iris[feature_cols]
```

Response vector:

```
In [10]: # store response vector in "y"
y = iris.species_num
```


Example:k-nearest neighbors

Scikit-learn's 4-step modeling pattern:

- Import the class you plan to use:

```
In [15]: from sklearn.neighbors import KNeighborsClassifier
```

- Instantiate the “estimator”
 - “Estimator” is scikit-learn term for “model”.
 - “Instantiate” means “make an instance of”.

```
In [16]: # make an instance of a KNeighborsClassifier object
knn = KNeighborsClassifier(n_neighbors=1)
type(knn)
```

```
Out[16]: sklearn.neighbors.classification.KNeighborsClassifier
```

Created an object that “knows” how to do K-nearest neighbors classification, and is just waiting for data.

Example:k-nearest neighbors

- **Fit the model with data (aka “model training”)**

- Model is "learning" the relationship between X and y in our "training data".
- Process through which learning occurs varies by model.
- Occurs in-place.
- Once a model has been fit with data, it's called a "fitted model".

```
In [18]: knn.fit(X, y)
```

```
Out[18]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                             metric_params=None, n_neighbors=1, p=2, weights='uniform')
```

- **Predict the response for a new observation**

- New observations are called “out-of-sample” data.
- Uses the information it learning during the model training process.
- Return a Numpy array, and keep track of what the numbers “mean”.
- Can predict for multiple observations at once.

```
In [20]: X_new = [[3, 5, 4, 2], [5, 4, 3, 2]]  
         knn.predict(X_new)
```

```
Out[20]: array([2, 1], dtype=int64)
```

Example:k-nearest neighbors

Tuning a KNN model:

```
In [21]: # instantiate the model (using the value K=5)
         knn = KNeighborsClassifier(n_neighbors=5)

         # fit the model with data
         knn.fit(X, y)

         # predict the response for new observations
         knn.predict(X_new)
```

```
Out[21]: array([1, 1], dtype=int64)
```

Question: Which model produced the correct predictions for the two unknown irises?

Answer: We don't know, because these are **out-of-sample observations**, meaning that we don't know the true response values. Our goal with supervised learning is to build models that generalize to out-of-sample data. However, we can't truly measure how well our models will perform on out-of-sample data.

```
In [22]: # calculate predicted probabilities of class membership
         knn.predict_proba(X_new)
```

```
Out[22]: array([[ 0. ,  0.8,  0.2],
                [ 0. ,  1. ,  0. ]])
```

Overfitting and Underfitting

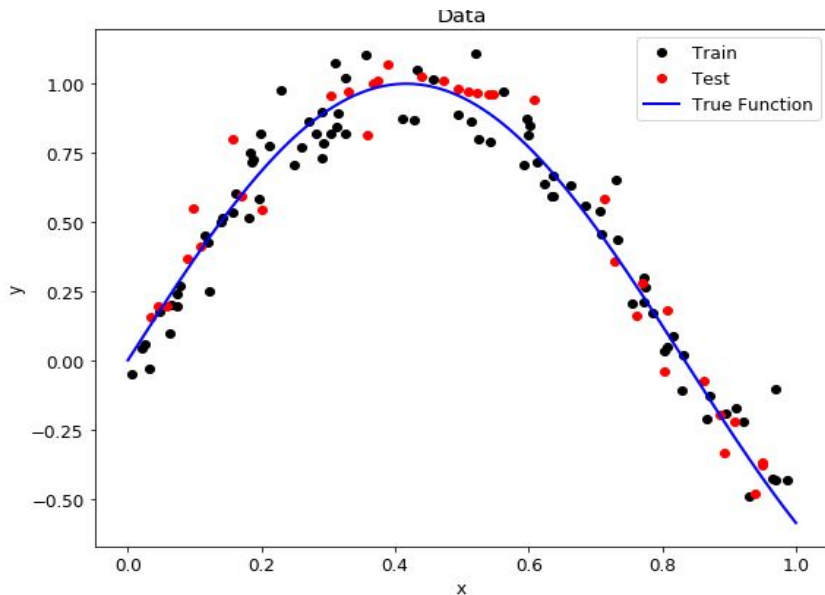
Concepts:

- **Model basic:** A model is simply a system for mapping inputs to outputs. A model learns relationships between the inputs, called **features**, and outputs, called **labels**, from a training dataset. During training the model is given both the features and the labels and learns how to map the former to the latter. A **trained model** is evaluated on a **testing set**, where we only give it the features and it makes **predictions**. We compare the predictions with the known labels for the testing set to calculate accuracy.
- **Training and Testing Data:** To make a model, we first need data that has an underlying relationship. For example, we will create our own simple dataset with X-values (features) and y-values (labels). An important part of our data generation is adding random noise to the labels. In any real-world process, whether natural or man-made, the data does not exactly fit to a trend. There is always noise or other variables in the relationship we cannot measure. In the example, the trend between area and price is linear, but the prices do not lie exactly on a line because of other factors influencing house prices.

Overfitting and Underfitting

The data similarly has a trend (which we call the **true function**) and random noise to make it more realistic. After creating the data, we split it into random training and testing sets. The model will attempt to learn the relationship on the training data and be evaluated on the test data. In this case, 70% of the data is used for training and 30% for testing.

We can see that our data are distributed with some variation around the true function (a partial sine wave) because of the random noise we added. During training, we want our model to learn the true function without being “distracted” by the noise.



Overfitting and Underfitting

- **Model Building:** Choosing a model can seem intimidating, but a good rule is to start simple and then build your way up. The simplest model is a linear regression, where the outputs are a linearly weighted combination of the inputs.

In our model, we will use polynomial regression to learn the relationship between x and y . Polynomial regression, where the inputs are raised to different powers, is still considered a form of “linear” regression even though the graph does not form a straight line .

The general equation for a polynomial is below.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots + \beta_n x^n + \varepsilon.$$

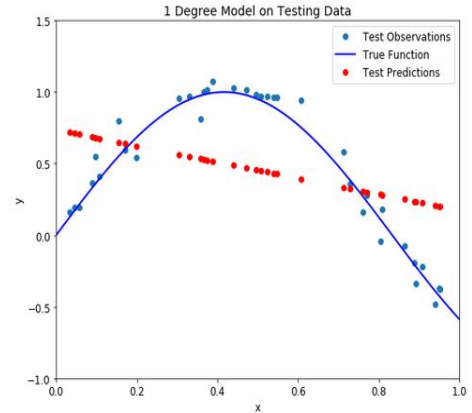
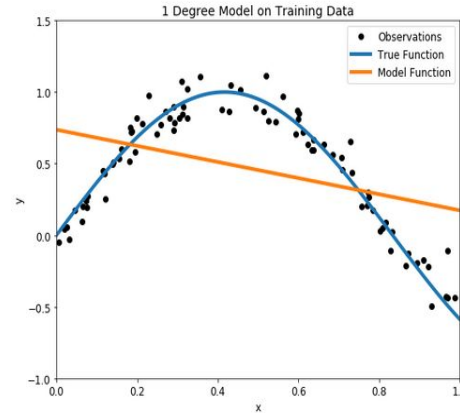
Here y represents the label and x is the feature. The beta terms are the model parameters which will be learned during training, and the epsilon is the error present in any model.

Once the model has learned the beta values, we can plug in any value for x and get a corresponding prediction for y .

Overfitting and Underfitting

The **problem of overfitting and underfitting**, finally appears when we talk about the polynomial degree. The degree represents how much flexibility is in the model, with a higher power allowing the model freedom to hit as many data points as possible. An **underfit model** will be less flexible and cannot account for the data.

First up is an underfit model with a 1 degree polynomial fit. In the image on the left, model function in orange is shown on top of the true function and the training observations. On the right, the model predictions for the testing data are shown compared to the true function and testing data points.



Overfitting and Underfitting

Conclusions:

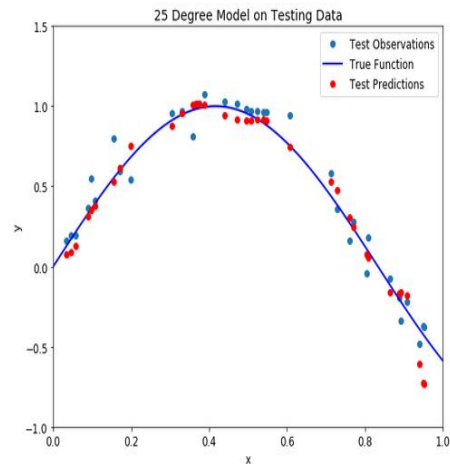
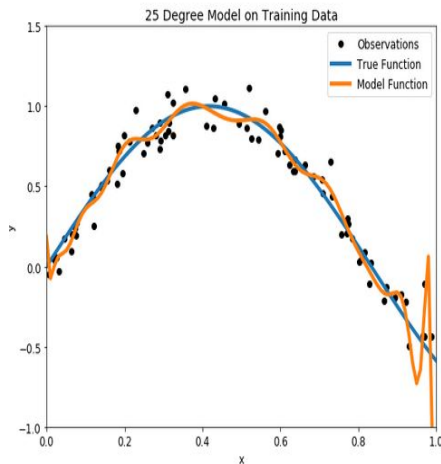
Our model passes straight through the training set with no regard for the data!. This is because an underfit model has **low variance** and **high bias**. **Variance refers to how much the model is dependent on the training data.** For the case of a 1 degree polynomial, the model depends very little on the training data because it barely pays any attention to the points! Instead, the model has **high bias, which means it makes a strong assumption about the data.** For this example, the assumption is that the data is linear, which is evidently quite wrong. When the model makes test predictions, the bias leads it to make inaccurate estimates. The model failed to learn the relationship between x and y because of this bias, a clear example of underfitting.

We saw a low degree leads to underfitting. A natural conclusion would be to learn the training data, we should just increase the degree of the model to capture every change in the data. This however is not the best decision!.

Overfitting and Underfitting

With such a high degree of flexibility, the model does its best to account for every single training point. This might seem like a good idea – don't we want to learn from the data? Further, the model has a great score on the training data because it gets close to all the points. While this would be acceptable if the training observations perfectly represented the true function, because there is noise in the data, our model ends up fitting the noise.

This is a model with a **high variance**, because it will change significantly depending on the training data. The predictions on the test set are better than the one degree model, but the twenty five degree model still does not learn the relationship because it **essentially memorizes the training data and the noise**.



Overfitting and Underfitting

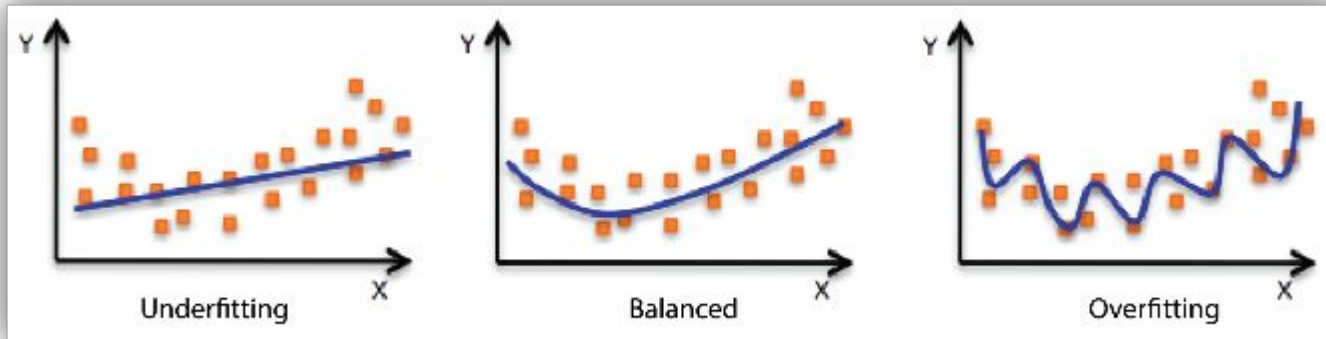
The problem is that we want a model that does not “**memorize**” the training data, but learns the actual relationship! How can we find a balanced model with the right polynomial degree? If we choose the model with the best score on the training set, we will just select the **overfitting model** but this cannot generalize well to testing data.

Fortunately, there is a well-established data science technique for developing the optimal model: **validation**.

Overfitting and Underfitting

Summary:

Understanding **model fit** is important for understanding the root cause for poor model accuracy. This understanding will guide you to take corrective steps. We can determine whether a predictive model is **underfitting** or **overfitting** the training data by looking at the prediction error on the training data and the evaluation data.



<https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html>

Overfitting and Underfitting

Summary:

- Your model is **underfitting** the training data when the model performs poorly on the training data. This is because the model is unable to capture the relationship between the input examples (often called X) and the target values (often called Y).
- Your model is **overfitting** your training data when you see that the model performs well on the training data but does not perform well on the evaluation data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples.

Poor performance on the training data could be because the model is too simple (the input features are not expressive enough) to describe the target well. Performance can be improved by increasing model flexibility. To increase model flexibility, try the following:

- Add new domain-specific features and change the types of feature processing used (e.g., increasing n-grams size).
- Decrease the amount of regularization used.

Overfitting and Underfitting

Summary:

If your model is **overfitting** the training data, it makes sense to take actions that reduce model flexibility. To reduce model flexibility, try the following:

- **Feature selection:** consider using fewer feature combinations, decrease n-grams size, and decrease the number of numeric attribute bins.
- Increase the amount of regularization used.

Accuracy on training and test data could be poor because the learning algorithm did not have enough data to learn from. You could improve performance by doing the following:

- Increase the amount of training data examples.
- Increase the number of passes on the existing training data.

Example

Supervised learning example: Simple linear regression....

This is the end of class!

¿Questions?