

---

# **Data Science and Machine Learning basic course with Python**

***Release 0.0.3***

**Volodymyr Kovenko, Vitalii Shevchuk**

**Apr 04, 2020**



## CONTENT:

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Notes about the course</b>	<b>3</b>
<b>3</b>	<b>Basic knowledge requirements</b>	<b>5</b>
<b>4</b>	<b>Support</b>	<b>7</b>
4.1	How to start with Google Colaboratory . . . . .	7
4.1.1	Loading from colab . . . . .	8
4.2	Description . . . . .	9
4.2.1	Data science . . . . .	9
4.2.2	Machine learning . . . . .	10
4.2.3	Python . . . . .	14
4.3	Data Science with Python . . . . .	15
4.3.1	Getting and cleaning data . . . . .	15
4.3.2	Exploratory data analysis . . . . .	17
4.3.3	Reproducible research . . . . .	18
4.4	Machine Learning with Python . . . . .	20
4.4.1	Supervised learning . . . . .	20
4.4.2	Unsupervised learning . . . . .	44
4.5	FAQ . . . . .	49
4.5.1	How can I run assignments on my own machine? . . . . .	49
4.5.2	How can I contribute to the course to make it better? . . . . .	49
4.5.3	What is the exact list of courses which this course was built on? . . . . .	50





## INTRODUCTION

Welcome to Machine Learning & Data Science with Python basic course. The main purpose of this lookout course is to encourage students to apply basic knowledge of statistics, mathematics and Python to start solving real world problems using open source Machine Learning tools. The course is divided in two different parts.

Firstly, the basics of Data Science with Python are observed, as general data analysis, its processing and understanding along with the knowledge of how to represent information effectively is crucial for any machine learning practitioner. What is more, sometimes its beneficial to firstly try some statistical methods of solving the particular problem, before diving into more complicated ML stuff. In addition some open source packages for working with data and scientific calculus are introduced:

- [Numpy](#)
- [Pandas](#)
- [Matplotlib](#)
- [Seaborn](#)

As the second part of course, students will have a hands on introduction to basic machine learning algorithms and core machine learning concepts. The course won't dig in depth of the algorithms work, but we strongly encourage you to checkout useful references that can be found throughout this course. Finally, the students will have a chance to work with such packages:

- [Scikit learn](#)
- [Keras](#)
- [Tensorflow](#)

---

**Important:** Currently the course provides machine learning assignments based only on scikit learn package.

---



## NOTES ABOUT THE COURSE

In this course we will use Python 3.7 and Jupyter Notebook as main tools for conducting exercises. All the work will be conducted on Google Colaboratory, as it's free to use powerful tool based on Jupyter Notebook, thus you will need a google account to work on this platform. If you're not familiar with Jupyter Notebook, feel free to checkout this [tutorial](#). The course is VNTU students' oriented, but everyone can access the notebooks in Colaboratory and work through the course, if you want to do exercises on your local machine you can find the related information in [FAQ](#).

---

**Important:** Currently the course doesn't provide any testing of the exercises.

---



## **BASIC KNOWLEDGE REQUIREMENTS**

The course requires basic knowledge of Python along with its core concepts. In addition an intermediate knowledge of English or at least understanding of technical vocabulary is mandatory.



## SUPPORT

As the project is an open source, anyone can contribute to it, so feel free to join the development discussion :

- On github page of the [project](#) .

### 4.1 How to start with Google Colaboratory

For all the laboratories and assignments in this course, the google colaboratory is used. In order to understand key features of it, please follow [this](#) tutorial.

At the end of each lesson you will see the following button :

By pressing on it you'll be redirected to the related assignment :

The screenshot shows the Google Colaboratory interface for a notebook titled 'assignment\_1.ipynb'. The top bar includes the Google Colab logo, the notebook title, and a menu with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link 'Last edited on Dec 19, 2019'. Below the menu are buttons for '+ Code', '+ Text', and 'Copy to Drive'. The main content area is divided into sections: 'ASSIGNMENT 1', 'PREDICTING HOUSING PRICES USING LINEAR REGRESSION WITH 2 FEATURES', and 'PART 1'. The 'ASSIGNMENT 1' section contains a detailed introduction to the project, mentioning the Boston housing dataset and the goal of predicting housing prices using linear regression. The 'PART 1' section is titled 'Loading and processing the data' and begins with the instruction 'Firstly let's import necessary packages.'

In the google colab you can change the code as you want, you can add cells, write your own functions, etc, as all the notebooks for assignments are in playground mode. You can also choose the runtime type (it's beneficial while working with deep learning models) :

### Notebook settings

Runtime type  
**Python 3** ▼

Hardware accelerator  
**None** ▼ ⓘ

☐ Omit code cell output when saving this notebook

**CANCEL** **SAVE**

Note, that you don't need to install any packages, as they are already installed in colab environment. If you want to run all the code on your own machine, please go to this [section](#).

When you start running the cells, you will probably see this message :

#### Warning: This notebook was not authored by Google.

This notebook is being loaded from [GitHub](#). It may request access to your data stored with Google, or read data and credentials from other sessions. Please review the source code before executing this notebook.

**CANCEL** **RUN ANYWAY**

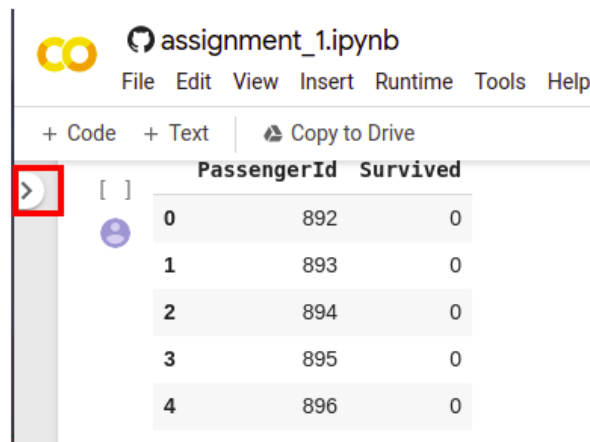
Just press *RUN ANYWAY* and go on with a notebook.

### 4.1.1 Loading from colab

In classification module you will try to solve the challenge from [kaggle](#), as the result the file with submission will be saved locally in colab. In order to load it to your machine and then submit on the page of competition, do the following :

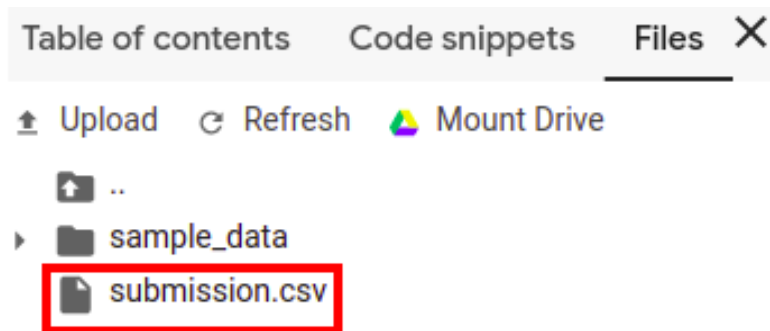
- Open the dropout menu :





	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	0

- Choose Files and download a submission.csv file :



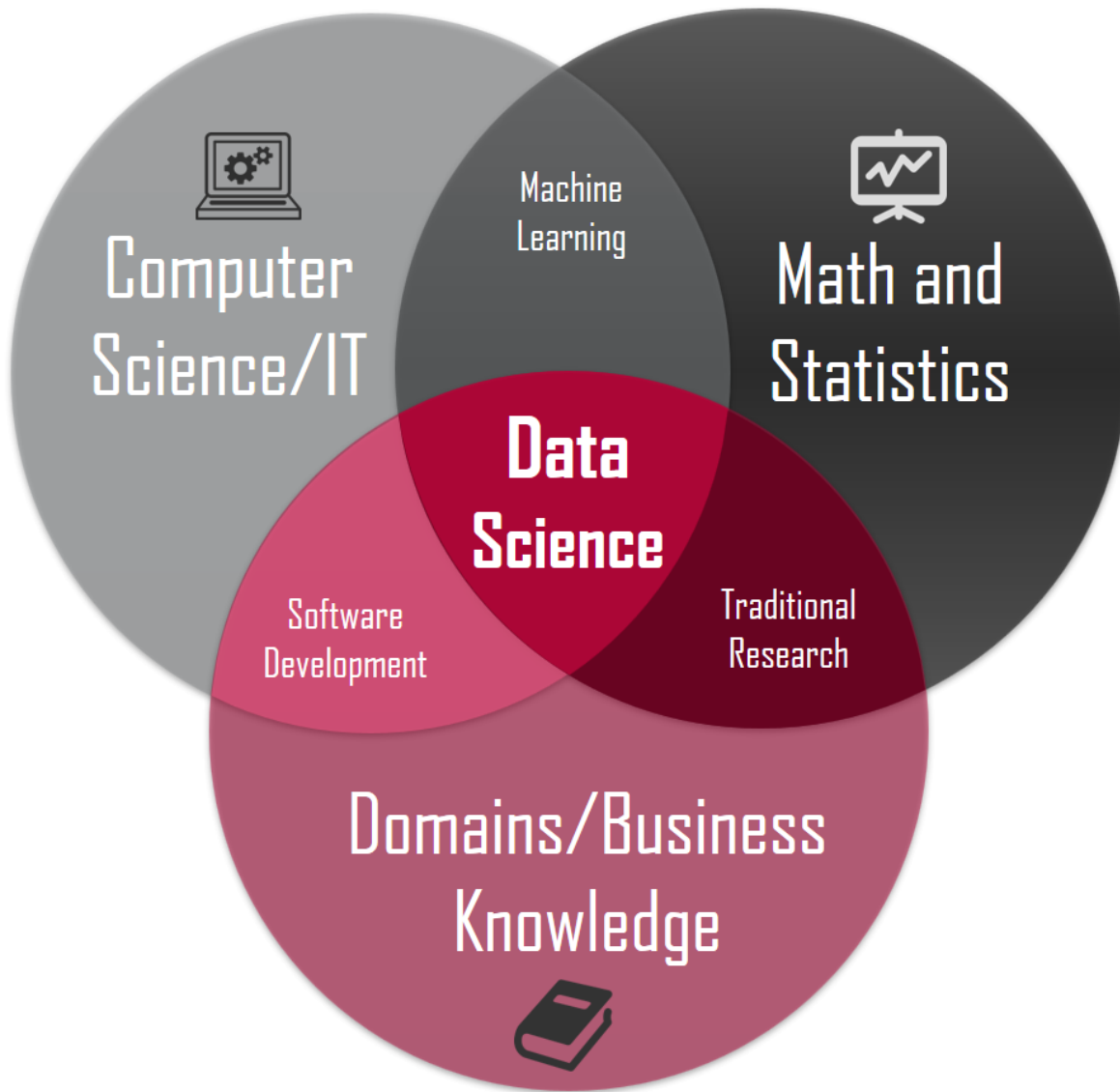
Colab is a powerful tool for working with notebooks, making research and analysis of data, because of that it will be used heavily through the course.

## 4.2 Description

In this section you will get to know the key notions that will be used through the course.

### 4.2.1 Data science

Data science - is a field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data. Data science is a field that combines statistics, data analysis and machine learning to understand and analyse the data. Nowadays data science is one of the most demanding professions in sphere of IT, as the contemporary world is a data driven one. The main duties of every data scientists include : analysis of data, it's preprocessing, cleaning and transformation and finally extracting sense from it using statistical and machine learning techniques along with presentation of performed results to non technical people. As the core of data science is statistics and mathematics, the related part of the course is focused on teaching how to apply the techniques of highlighted spheres to solve real world problems.

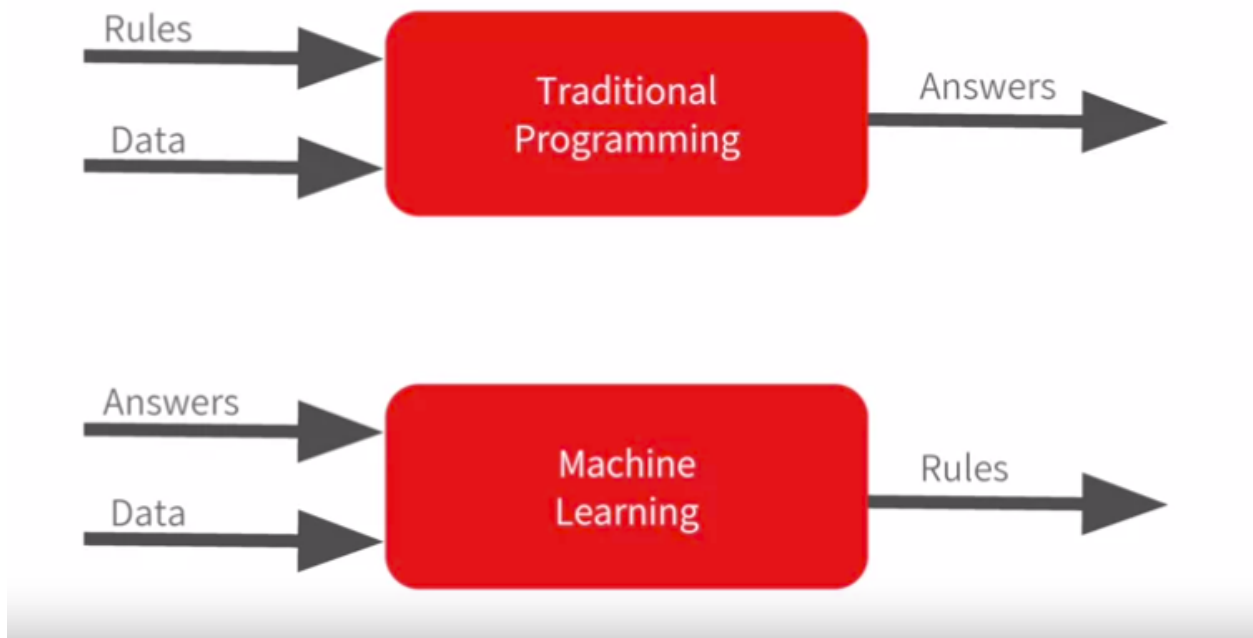


### 4.2.2 Machine learning

**Machine learning** - is a branch of AI (*Artificial Intelligence*) in programming the main goal of which is to make the machine “learn” how to solve the tasks which can’t be programmed explicitly. Nowadays, Machine learning is a really hot topic, as the applications of this sphere extended broadly. For instance let’s consider the contemporary smartphone : such functions as voice assistant, fingerprint and face verification, handwritten recognition are included by default, and all of the highlighted functions are machine learning driven. Actually, to understand the main difference between machine learning and ordinary programming we can expose the following example :

Let’s say we need to make a program that should predict the price of the house depending on the number of its rooms (*the example is simplified*). We can try to solve this program without using any machine learning techniques, just by writing some if else rules. But as the data grows it will become harder and harder to code all the rules to obtain an expected results. A much smarter solution is to train a machine learning algorithm that will find the dependency between price and number of rooms and then create the rules itself (*the current example is related to supervised learning*)

The picture describing this main difference is pinned below.



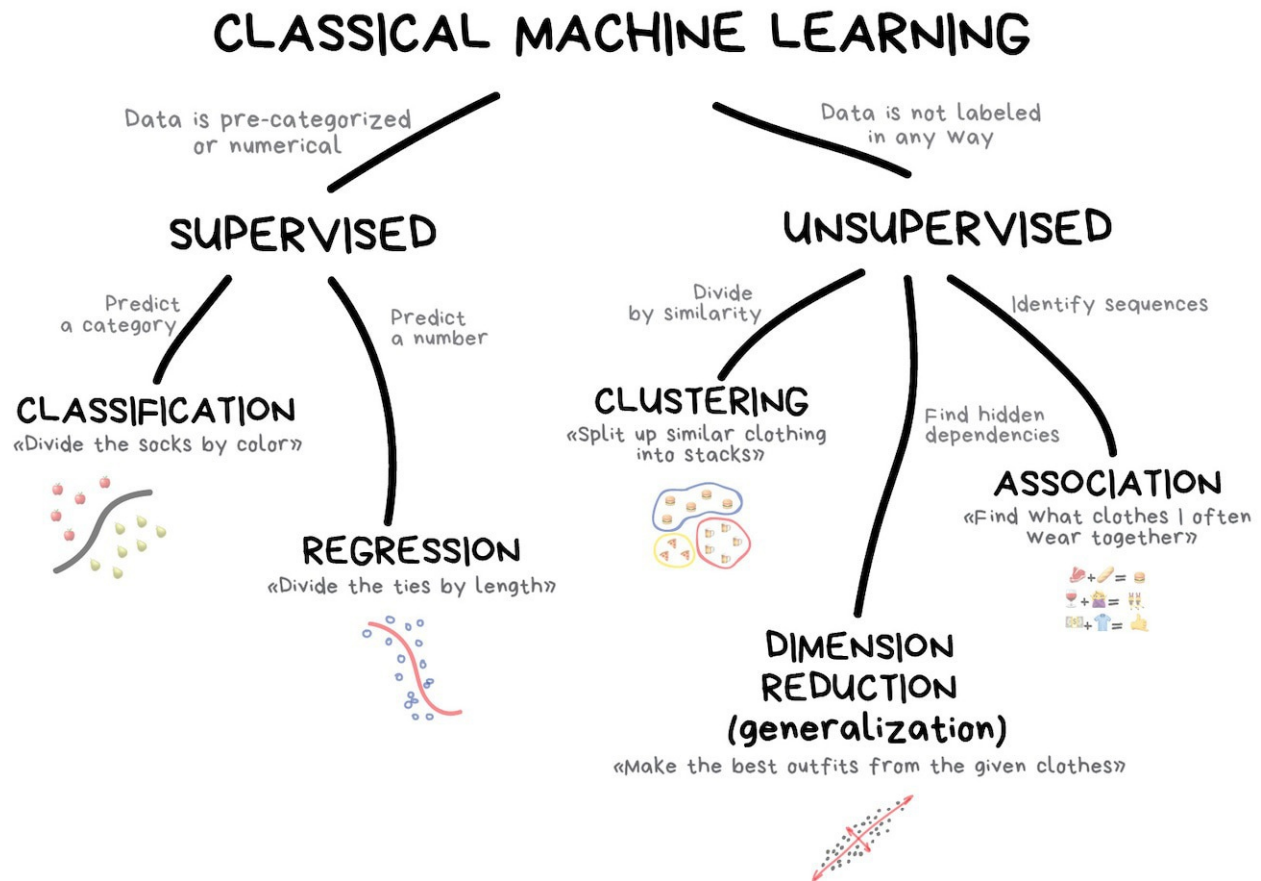
There are three main types of machine learning :

- *Supervised learning*
- *Unsupervised learning*
- *Reinforcement learning*

---

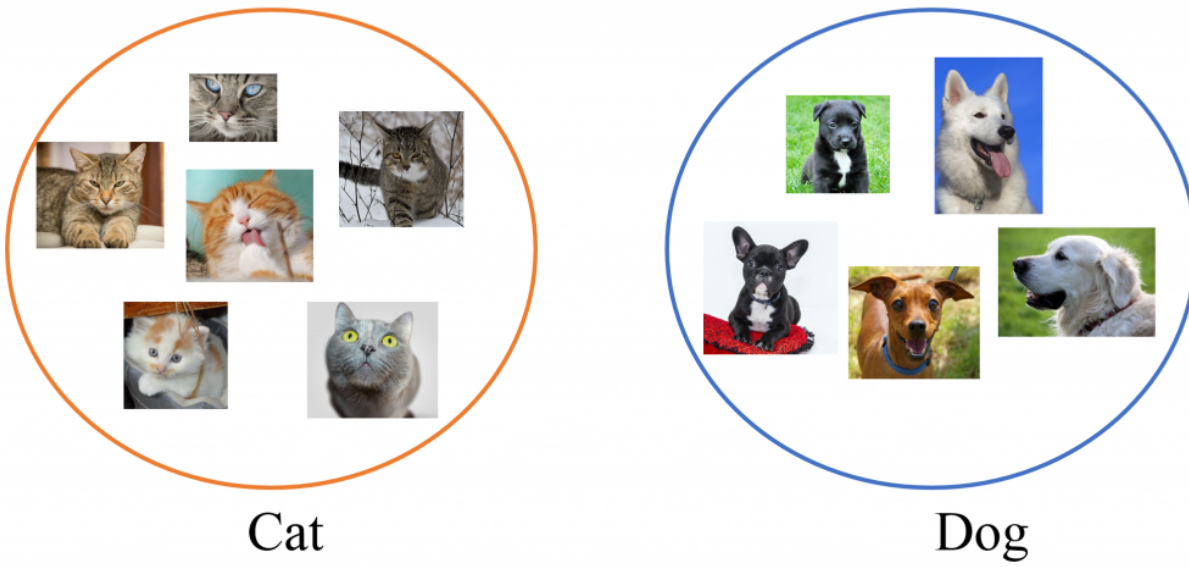
**Note:** We won't consider the Reinforcement learning in this course, but you can find additional information about it [here](#). Instead we will mainly focus on "classical machine learning".

---

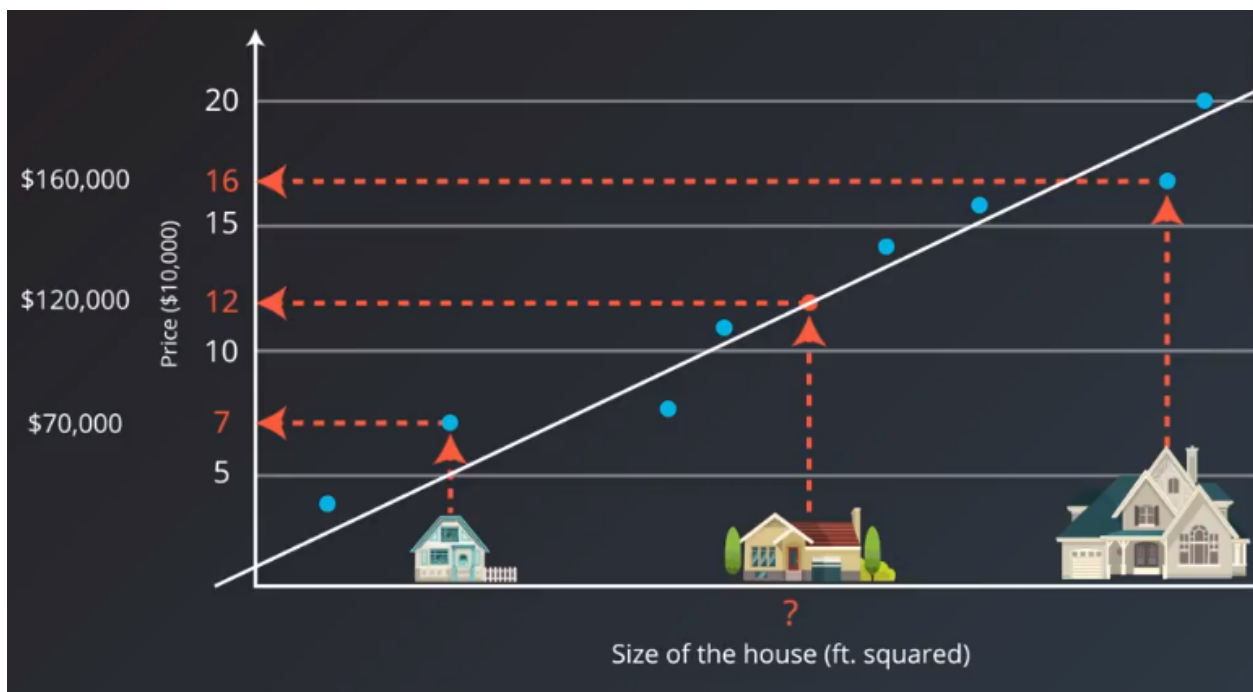


*Supervised learning* - is a type of machine learning, when given a data/features (by notation  $X$ ) and corresponding answers/labels (by notation  $Y$ ) an algorithm learns a complex function to map data/features to answers/labels. There are lots of useful application concerning supervised learning, for instance : image classification, fraud detection, object recognition, face verification, weather forecast, etc. The supervised learning is divided into two types of problems : *regression* and *classification*.

In *classification* problem the answer (*sometimes called the target, we will use this name further*) is a categorical label/class and the task of the algorithm (*sometimes called the classifier or the model*) is to classify the sample/object depending on the features of relevant data. For better understanding, let's say you encountered the following problem : you have pictures of cats and dogs and you should determine whether the picture contains dog or cat. That's a routine task of image classification.



On contrary the problem of *regression* exists, in which the target is a discrete continuous number and the task of a model is to learn the dependencies between features and the target to output the value that is the nearest to the target one. Simple example of regression is the task of predicting houses' prices depending on the number of rooms/size of the house. Instead of classifying houses into different groups, what we really want is to predict its exact price, that is actually a continuous number.



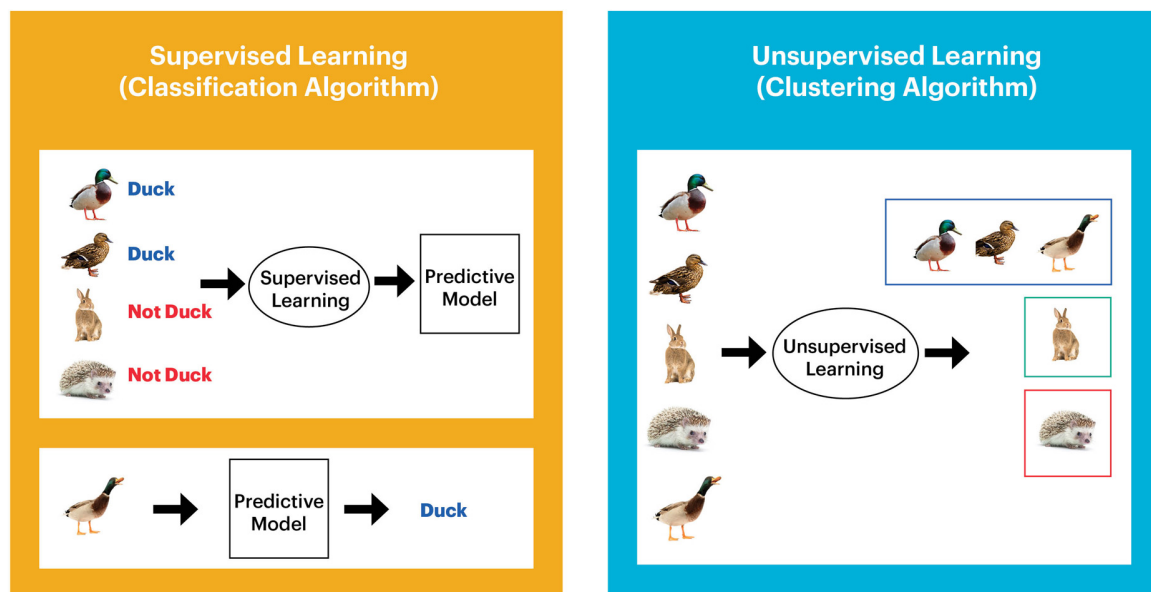
*Unsupervised learning* - is a type of learning when algorithm is given only data/features without any answers/labels. The purpose of unsupervised learning algorithms is to find the similarities between data samples and based on this

similarities perform some actions. The unsupervised learning is divided into three types of problems : *clustering* , *dimension reduction* , *association*.

**Note:** We will focus only on *clustering* as the other algorithms are out of the scope in this course, but we encourage you to visit this [page](#) to get more information.

In *clustering* problem the goal of the algorithm is to cluster the data into different groups based on the similarities between samples. To understand the gist of clustering let's define the following problem : you need to make a system that identifies spam messages and sends them to spam folder. What you can do is gather the information like words from the messages and then break these information into two different groups, after that you can determine yourself which group contains spam and which - not.

To understand the difference between supervised and unsupervised learning let's consider the following pictures which shows the difference between classification (supervised learning) and clustering (unsupervised learning) :



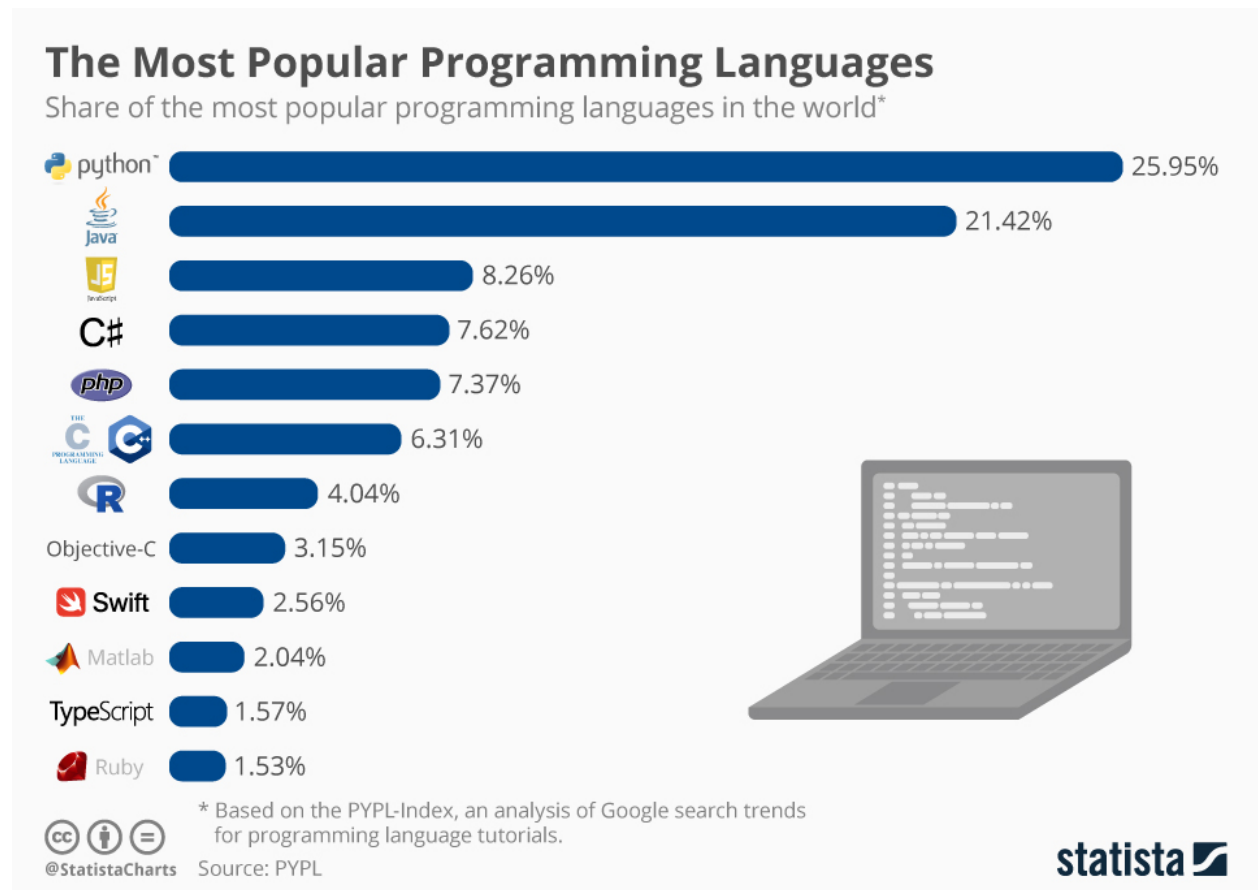
Western Digital.

You will have a chance to work with both supervised and unsupervised learning techniques and dig deeper into core concepts of machine learning further in this course.

### 4.2.3 Python

Why use Python for machine learning and data science? The answer is pretty obvious, because it's much simpler, much faster and finally much more efficient to do this heavy job using the exposed programming language. Python scientific packages such as scipy, numpy, pandas and others allow conducting complex mathematics computations and statistics calculus in few lines of code giving analysts and researchers a possibility to easily make analysis and developing new algorithms. What is more, Python is usually used in production solutions, thus you can easily refactor your draft code for (let's say) processing of the data and then scale it up to production system.

Despite the fact that in this course you won't write the production ready code, you will get to know how to use Python for basic analysis and machine learning that will give you the mandatory skills to continue learning and developing in data science area. Finally, to persuade you in the fact that Python is the language you should really use, let's look at the chart showing the popularity of languages for the current year :



Based on the diagram shown above, Python is the most popular language at the moment, just analytics, nothing personal.

## 4.3 Data Science with Python

*“Big data is not about the data.”* - Peter Sondergaard, Senior Vice President at Gartner 8

Welcome to the data science part of this course in which one should get in touch with essential skills used by DS/ML professionals day by day. We will review 3 starting points that every data scientist should know, each of topics will be represented by simplified problem. For first 2 examples additional task with a star is provided.

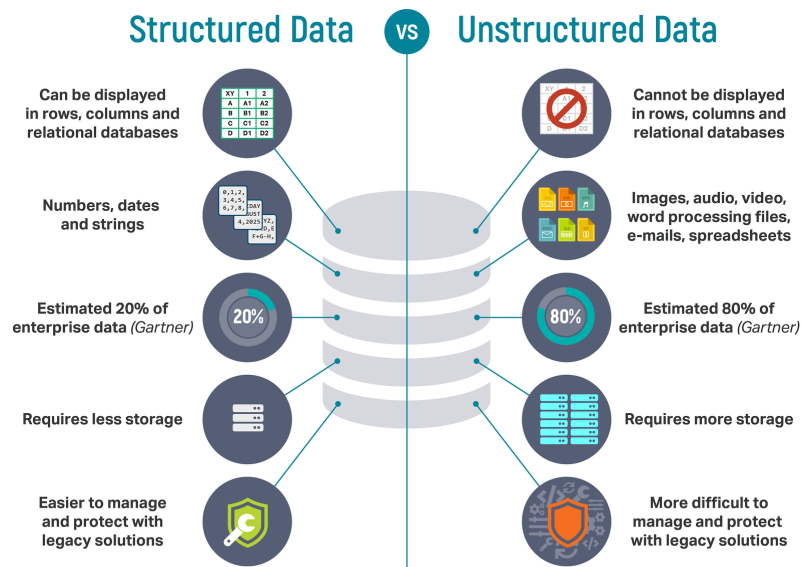
### 4.3.1 Getting and cleaning data

One can admit that term Data Science consist of 2 words: “Data” and “Science”. It’s not a coincidence, thus in this part we will focus on the first part - “Data”.

In the present times we can find data literally everywhere: from text messages on our phones to whether predictions on TV in different forms and presentations (video, audio, text, images, etc. . . ). Data can be divided into two subcategories

shown below.

Comparison of structured and unstructured data :



## Tidy Data

Up to 80% of the time devoted to data analysis is spent on data cleaning and data preparation. And here for the rescue comes a term *Tidy Data*. Idea behind tidy data is taken from relational databases and database normalization from computer science. Beside rows and columns in tables, additional point should be considered:

- The data is a collection of values of a given type
- Every value belongs to a variable
- Every variable belongs to an observation
- Observations are variables for a unit (like an object or an event).

The objective of tidy data is to map the meaning of the data (semantics) onto the structure of the data.

However, many real world data sets violate the principle of tidy data. Here is 5 common problems:

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Variables are stored in both rows and columns
- Multiple types of observational units are stored in the same table
- A single observational unit is stored in multiple tables



It is only after data is tidy that it is useful for data analysis.

Other areas outside of tidy data include parsing variable types (dates and numbers), dealing with missing values, characters encoding, typos and outliers.

### Description of assignment

In the first assignment we will work through the process of dates parsing using the dataset from [kaggle](#). Basically there are 2 datasets containing issues concerning dates columns and in order to produce meaningful results and parse the data, additional transformations (like converting strings to data) should be done. The one will learn how to work with data frames and visualize the data to check if applied functions are correct. In terms of example, one dataset is already processed, and yet other one is expected to be processed by students.

---

**Note:** Some information is taken from [this](#) article.

---

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



### 4.3.2 Exploratory data analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

*“Asking the right questions is what separate Data Scientists that know ‘why’ from folks that only know what (tools and technologies).” - Kayode Ayankoya*

To be effective Exploratory Data Analysis should target specific aims in form of questions / hypothesis raised by Data Scientists.

### Description of assignment

In this section we will work through crime dataset taken from [kaggle](#). Along the way many questions find their answers with help of visualization tools such as:

- seaborn
- matplotlib
- wordcloud

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



### 4.3.3 Reproducible research

We made a lot of work till now. But does it matter for other people if they did not understand what was done and how it was done?

*“An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.” - D. Donoho*

Consider the situation when you made a great research regarding crimes in USA, published the article and then get tons of questions from followers regarding your deeds. On the other side, if the one provided data, code, experiment flow and other information that is relevant to one's research, there will be much less questions for sure.

#### Top reasons for making your research reproducible:

- *Ethics – we have a responsibility to show our work to move science forward.*
- *Funding requirements – many funding agencies require the storage of data and the steps performed in the analysis as part of the effort to increase rigor and reproducibility.*
- *Catch your mistakes – when you generate a reproducible data document, you are more likely to catch mistakes. Also, your documented steps allow you to trace back to where you went wrong.*
- *Others can catch mistakes – it is far better that others, like the reviewers for your manuscript or your mentors, find the flaws in your analysis than for them to be buried while countless other students try to repeat your work.*

- *Others can learn how to perform the analysis – a reproducible research document can be a powerful teaching tool for future lab members or others around the globe.*
- *Better study design – when you write down why you performed a test, you realize the rationale, because ‘that’s how we always do this’ doesn’t quite cut it.*
- *Other people who want to do research in the field can really start from the current state of the art, instead of spending months trying to figure out what was exactly done in a certain paper. It is much easier to take up someone else’s work if documented code is also available.*
- *It highly simplifies the task of comparing a new method to the existing methods. Results can be compared more easily, and one is also sure that the implementation is the correct one.*

We encourage you to take a look on few articles that are publicly available on [rpubs](#). All of them are using R as main data processing language, but for now we are interested only in the structure of research papers.

- [article1](#)
- [article2](#)
- [article3](#)
- [article4](#)

Listed links are only for educational purposes.

Here is common structural elements in most of shown articles:

- **Title**
- **Initials of the author**
- **Introduction/Synopsis**
- **Data processing/Environment preparation**
- **Results**
- **Summary**

### Description of assignment

We hope that you remember the data used for previous assignment. Despite the fact that all the assignments in this course follow reproducibility rules, we are going to prepare a .pdf article, that is prepared from a changed notebook of previous lesson, to get better sense of theory in this section. The skills you gained in this module of course are mandatory for any machine learning practitioner. What is really important is to understand the data you are working with and be able to use it effectively for your purposes. Good luck.

[Pdf regarding research.](#)

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



## 4.4 Machine Learning with Python

*“Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don’t think AI will transform in the next several years.” - Andrew Ng*

Welcome to the machine learning part of this course in which you will learn how to build benchmark classifiers for solving problems related to both supervised and unsupervised learning using Scikit-learn package. This part of course is divided into two subparts : supervised and unsupervised learning. In the sub part dedicated to supervised learning you will work with regression and classification, whereas in the unsupervised learning part you will have to encounter the problem of clustering. Through the exercises and different tasks you will master the core concepts of machine learning, that will help you in digging deeper into the field.

### 4.4.1 Supervised learning

In this passage you will have a chance to work with Polynomial, Linear and Logistic regressions, Decision Trees and Support Vector Machines. The section is divided into two subcategories : regression and classification. We strongly recommend you to start with regression category, as main concepts of machine learning are exposed there.

#### Regression

In this section you will work with regression models in order to solve the specified tasks. Please move one to the lessons.

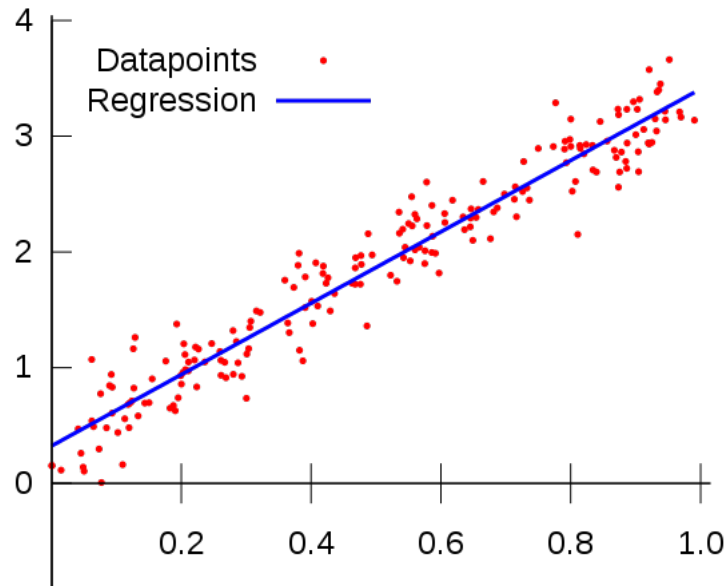
#### Linear regression and core concepts of machine learning

##### Basics of linear regression and loss functions

Speaking about machine learning, the best practice is to start with a basic simple algorithm and master core ideas using it. Before speaking about the particular model we are going to consider, you should understand that the key purpose of every machine learning model is to find a complex function that will do the complex stuff implied by the data. The classifier we will speak about is called *linear regression*. As the name implies this model is used for regression and the function it tries to learn is a linear one. The mathematical representation of this function is the following :

$$y = Wx + b$$

The graphical representation of this function is the following :



Here  $y$  - is a predicted discrete number,  $x$  - is a feature,  $W$  and  $b$  - parameters which the model tries to learn (they actually represent the templates learned by model to make different decisions). If we have more than one feature, for example to predict the house's price we have not just the number of rooms, but also the distance to the center of the city, the equation above will transform to this view :

$$y = W_1x_1 + W_2x_2 + b$$

We now understand where  $x$  (feature) is used, but we also have a real  $y$ , what is an aim of this thing? Actually to understand which parameters to choose our model needs to somehow measure if the predicted  $y$  is close to the real one, it needs to learn if it's wrong or right, and if it's wrong the model should continue learning to understand the gist better and calculate new parameters. That's the moment where such notation as *loss function* comes into the game. *The purpose of loss functions is to measure the performance of the model and help it understand made mistakes, and the purpose of the algorithm is to find such  $W$  and  $b$  parameters that minimize the loss function.* - this a core concept of machine learning that is used across the field. Actually the package we use (scikit-learn) doesn't use more sophisticated methods of optimizing the loss functions like gradient descent, instead it uses least-squares estimation that is much simpler and can give bad results.

---

**Note:** We won't dig into math of loss functions and optimization algorithms, but we hardly encourage you to take a look on [this](#) course made by Andrew Ng.

---

How does a loss function look like? Well, loss functions can differ depending on the problem the classifier should solve, but for regression tasks we will consider the following.

### MSE (Mean Squared Error) :

$$\frac{1}{n} \sum_{i=1}^n (y_{pred,i} - y_{true,i})^2$$

---

**Note:** There are other loss functions you can use for regression tasks as MAE (Mean Absolute Error), but throughout this part of course we will use MSE to validate our model.

---

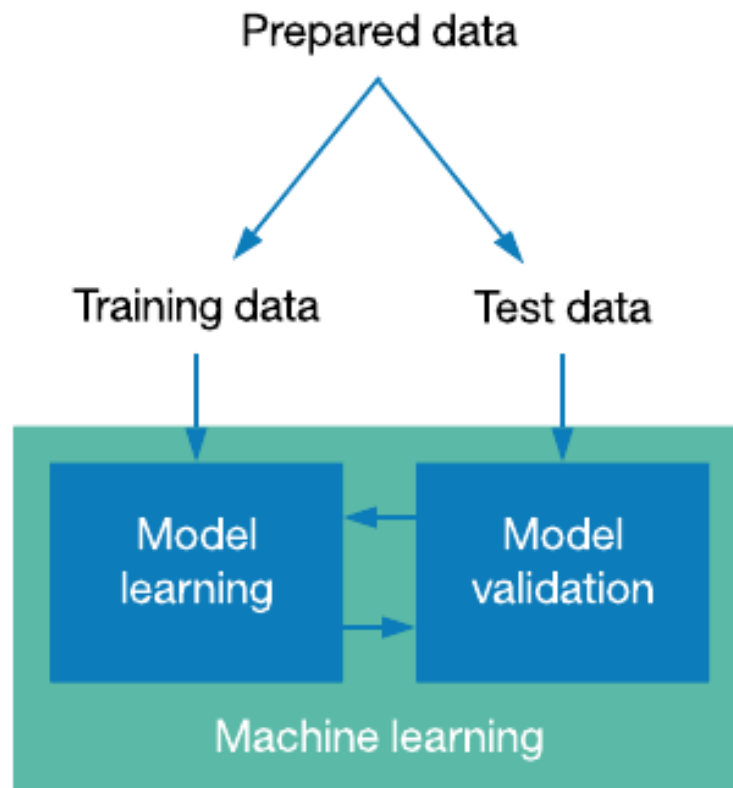
The intuition behind this loss function is in the fact that it makes big mistakes of our model bigger while small ones - smaller. We will use this particular loss function as our metric, in order to measure the performance of the model.

### Transferring and processing of data

In most of cases the performance of the algorithm depends on the quality of data, thus it's mandatory to examine and process it. First thing to check is if some data records are missing, if yes you should drop them from our data. The second thing is to verify that your feature is of the same type for all the data samples, if not - you should make some processing of it in order to cast everything to one type. In scenario of regression people encounter numeric data and sometimes it's really useful to scale it down. Let's say you have a price (the target value we try to predict) presented as thousands of dollars : 40000\$, 20000\$ and so on; then if you calculate MSE on it, you will see a huge score that isn't representative (we want MSE score to be as small as possible). One option to solve this problem is to scale down the price by dividing it by 1000, then instead of 40000 we will have 4000 and so on. Moreover features should be on the same small scale as it has a big effect on training of the algorithm.

### Train/Test split and model validation

Let's say we defined our classifier and prepared the data, but what should we do after? What data should we train the model on? Should we take all the data and just feed it to the classifier or there is some other process behind it? Actually, training the classifier on all the data can be useful in some custom cases, but frankly speaking that's not a great idea. *In machine learning workflow you need somehow understand if your model is good or bad. If you train your model on all the data you will see its performance that is related to learning only, but the fact we are really interested in is the performance on unseen real world data, as we train our algorithm to then use it for predictions.* Thus we need to somehow split our data into two parts (there is also a practice to split data into three parts train/validation/test, but for now we will use only train/test split) train data - the data we will use to train our model and test data - the one we will use to validate our model. While working with regression we will use only one metric named MSE (Mean Squared Error) to validate the performance of our algorithm. There is also a practice of comparing performance on train data with performance on test one. If the performance on train data is better than on test, then it is a signal of model overfitting on train data. If the performance on train data is bad, it means that model is underfitting. We will examine both cases later in the assignments, but a key thing to understand is the fact that you need to evaluate your model on an unseen data to estimate if it's ready for real usage.



When we use train/test split the good practice is to push 75-80% in train set and 25-20% to test set.

### Description of assignment

---

**Note:** Currently assignments are available only in the interactive mode, but you can change the notebook whatever you want.

---

In this assignment you will work with boston housing prices dataset that is available via `sklearn.datasets` package. As the data features and targets are already scaled and the data is cleaned, minimum efforts are required to process it. As it's your first assignment you will use only two features (number of rooms and average distance to center) to train a model and make predictions. After processing, you will visualize the dependency between two highlighted features in order to get some insights about the data. Then you will split the data into train and test subsets. Finally, you will create a linear regression model, train it on train data and evaluate on test one. With all this said, let's get started.

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



## Polynomial regression

### Key concepts of polynomial regression

Recall that in the previous lesson we introduced you to linear regression classifier and used only two features to train the algorithm. Unfortunately, sometimes it's not enough to use only two features and the relationship between features and target can be not linear. In order to determine this relationship and understand the correlation between variables, people often use Pearson correlation coefficient which we have already introduced you in the data science section. Previously we used Pearson correlation only for two features (average number of rooms and weighted distances to five Boston employment centres) and we saw a positive score, meaning that those features have a forward correlation with target (price). But some features can also have the inverse correlation, meaning that when the feature value decreases, the target value - increases and vice-versa. Sometimes due to complex relationships our features aren't linearly separable, thus we need a more complex function to fit the data. In this case the first thing to consider is polynomial regression. Polynomial regression is more about a sort of feature engineering than algorithm itself. Suppose that we have two features, as represented on the inset below:

$$\mathbf{x} = (x_0, x_1)$$

What we can do is transform this features to polynomial ones:

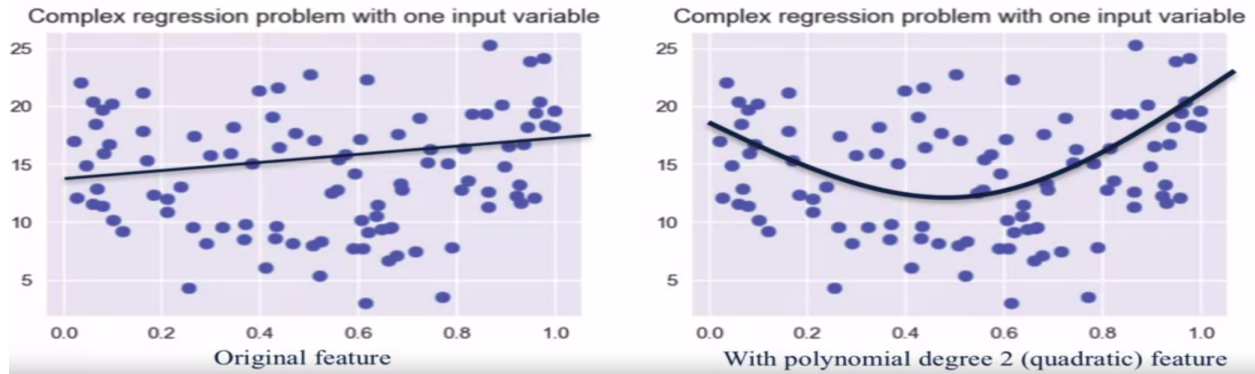
$$\mathbf{x} = (x_0, x_1) \longrightarrow \mathbf{x}' = (x_0, x_1, x_0^2, x_0x_1, x_1^2)$$

Now we still need to predict a target, but instead of 2 features we will use 5. The critical insight here is that it's still a weighted linear combination of features, so it's still a linear model, that can use same least-squares estimation method for  $w$  and  $b$  (our parameters). The degree of polynomial specifies how many variables participate at time in each new feature (below example : degree 2).

$$\hat{y} = \hat{w}_0x_0 + \hat{w}_1x_1 + \hat{w}_{00}x_0^2 + \hat{w}_{01}x_0x_1 + \hat{w}_{11}x_1^2 + b$$

So what really using polynomial features do is transforming our problem to a higher dimensional regression space. In effect adding these extra polynomial features allows us a much richer set of complex function that we can use to fit to the data. By using polynomial features we simply allow polynomials to be fit to the training data instead of a simply straight line, but using the same optimization algorithm that minimizes mse score.





The one downside of this polynomial transformation is that polynomial feature expansion with high degree can lead to complex models that overfit, thus the polynomial regression is often combined with regularization methods like ridge regression (we will talk more about it in the third lesson).

---

**Note:** We badly encourage you to take a look on the [following](#) course about using scikit-learn and other tools for applied machine learning.

---

### Description of assignment

In this assignment you will continue working with linear regression, but using all the features. First of all you will make a more complex data analysis to understand the dependencies between different features and target. Then you will fit a linear regression on all the features + transform them to polynomials in order to use polynomial regression. Finally, you will compare the results with the ones you obtained previously.

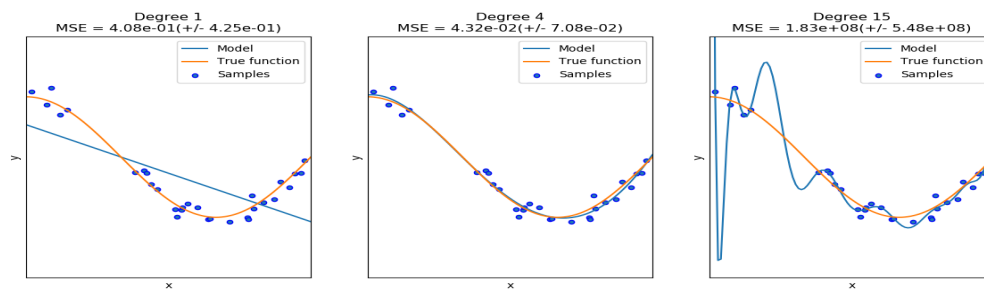
In order to check the gained knowledge, please carry on with the quiz related to this lesson.



### L1 and L2 regularization

## Overfitting problem

In the previous lesson you were introduced to polynomial regression model. Despite the fact that this classifier did much better on both train and test data than the linear regression, we see a gap between its performance on train and test data. Actually, the mse score on train data was lower than on test one, which is a sign of *overfitting* to train data. Overfitting is a most frequently encountered problem in the supervised learning, that means that the model has a poor generalization towards new data. Overfitting occurs when our model learns very complex function in order to exactly *fit* all the data in training subset. Let's consider the following example : You want to train your model to classify if the object is shoes, but the only images you have in our training data - are the photos of sneakers. Thus, when your model sees boots, it will think that it's not shoes which is actually incorrect. One problem that can lead to overfitting, as defined in the highlighted example, is the imbalanced dataset (when there are more samples in one class than in the other) or different distribution of data. Coming back to our problem, we experience overfitting because of the complexity of the decision boundary that was learned by our algorithm. The chart shown below describes the decision boundary of polynomial regression depending on different degree of polynomial.

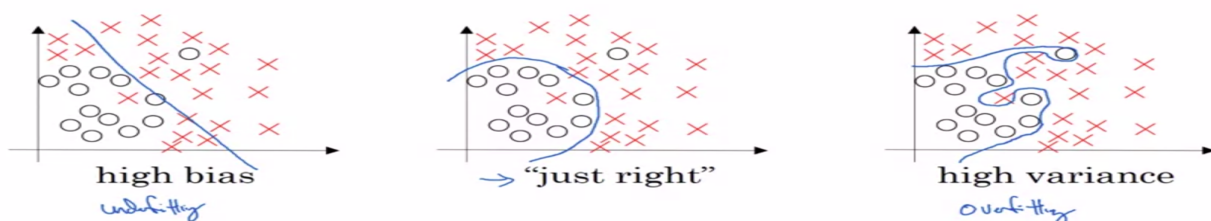


Note how the decision boundary changes with complexity of the model (with increasing the degree of polynomial). Overfitting is a fundamental problem, as the only thing we are interested in - performance of the model on unseen data. Overfitting is often referred to as the problem of high variance.

## Underfitting problem

Along with overfitting exists the problem of *underfitting*. Underfitting refers to the problem of a *bad fit to the data* often referred to as high bias, that implies that our classifier is too simple to solve our task. In order to understand the trade off between overfitting and underfitting let's consider the chart shown below.

## Bias and Variance



Actually, your classifier can encounter both problems at the same time.

**Note:** We badly encourage you to take a look on [this](#) course by Andrew Ng to get more information about bias-

variance trade off.

## Regularization

One solution to overfitting is called regularization. There are many types of regularization, but today we gonna focus on *l1* and *l2* regularization techniques. Actually *l1* and *l2* are the norms of matrices. In our case they are norms of weights' matrix that are added to our loss function, like on the inset below.

**L1 Regularization**

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

**L2 Regularization**

$$\text{Cost} = \underbrace{\sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^M W_j^2}_{\text{Regularization Term}}$$

*L1* and *L2* are often referred to as penalty that is applied to loss function. What the regularization does is making our classifier simpler to increase the generalization ability. Parameter  $\alpha$  in the chart above is hyper parameter which is set manually, the gist of which is the power of regularization, the bigger  $\alpha$  is - the more regularization will be applied and vice-versa. Regression that uses *L1* regularization is called Lasso regression and the one that uses *L2* - Ridge. Regularization is a key technique to solve overfitting.

## Feature normalization

We have already introduced you to data scaling in the description of the first assignment, but so far we haven't used it at all. In this assignment you will perform feature normalization in order to speed up the training of your classifier and get much better results, but before let's define some rules of using feature scaling and normalization.

- Fit the scaler using the training set, then apply the same scaler to transform the test set.
- Do not scale the training and test sets using different scalers : this could lead to random skew in data.
- Do not fit the scaler using any part of the test data: referencing the test data can lead to a form of data leakage.

Feature normalization is mandatory preprocessing step and we will use it further in this course.

### Description of assignment

In today's assignment you will use l1 and l2 regularization to solve the problem of overfitting. You will firstly scale you data using MinMaxScaler, then train linear regression with both l1 and l2 regularization on the scaled data and finally perform regularization on the polynomial regression.

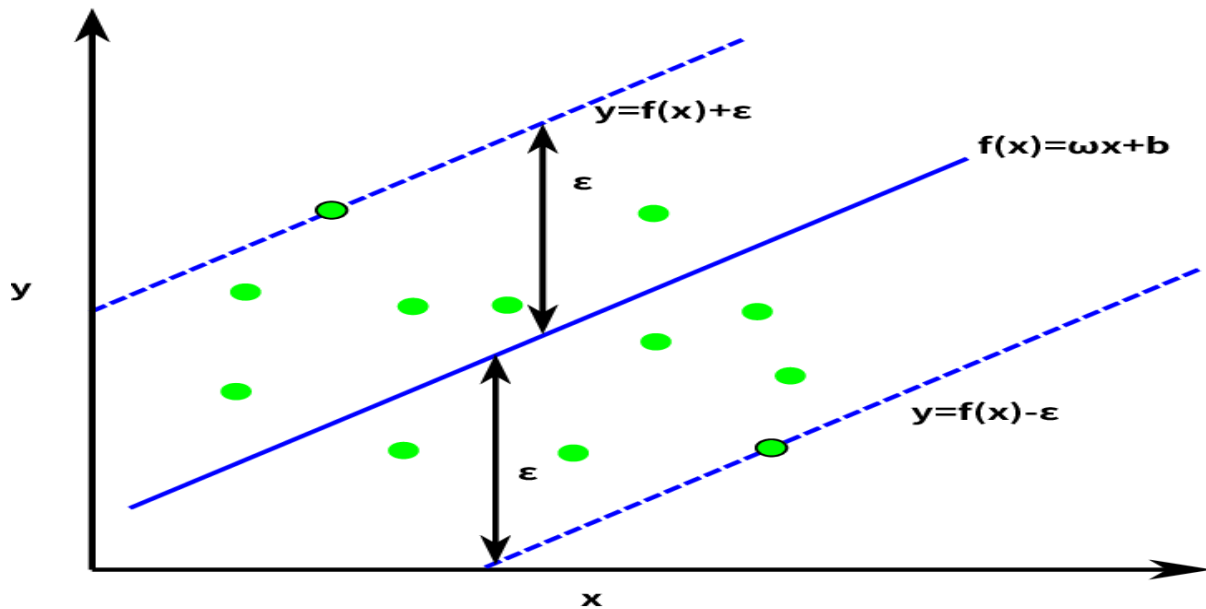
In order to check the gained knowledge, please carry on with the quiz related to this lesson.



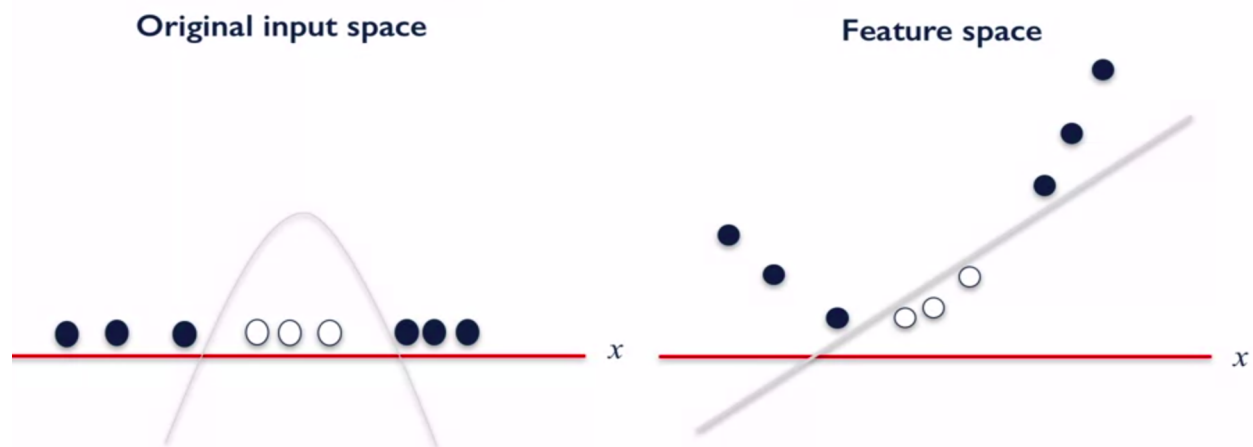
### SVM (Support Vector Machines)

#### The gist of SVM

Today we will introduce you to Support Vector Machines classifier. SVM is often referred to as maximum margin classifier. What does the classifier margin mean? Margin is defined as the maximum width the decision boundary area can be increased before hitting a data point. The linear classifier with maximum margin is a linear Support Vector Machine (LSVM). The formula that describes the decision boundary of linear SVM regression is the following (where epsilon denotes the width of margin) :



There is another type of SVM algorithm known as Kernelized SVM. Kernel is just a similarity measure (modified dot product) between data points. What it really does is transforming features to the other space in order for them to be easily fit by a linear classifier (see the example shown below).

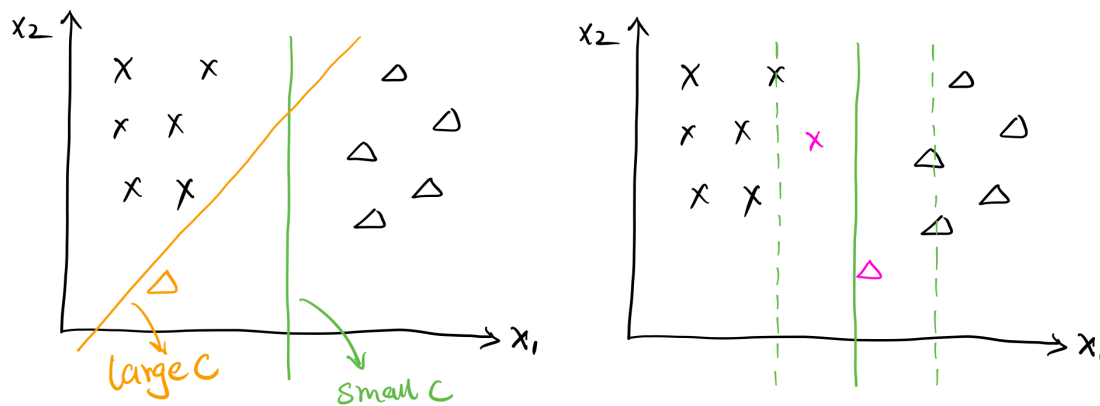


This kind of transformation resembles the polynomial transformation that was used earlier for linear regression, and frankly speaking the polynomial kernel function for SVM exists. There are different kernel functions, but the main thing about them is that they can transform the original input space to the feature space in which features are linearly separable.

**Note:** We won't dig deep into the math of SVM, but we badly encourage you to take a look on [this course](#) made by Andrew Ng.

## Handling Overfitting

The strength of regularization in SVM is determined by  $C$  parameter. Larger values of  $C$  - less regularization, smaller - more. There is also the parameter named gamma which is applied in kernel function and is responsible for the smoothness of decision boundaries. Smaller gamma results in more points grouped together and smoother decision boundaries, larger values of gamma results in more complex decision boundary. Both parameters affect the regularization and should be chosen correctly.



## Description of assignment

In today's assignment you will work with SVM regressor. You will have a chance to try different kinds of kernel functions, values of  $C$  and gamma and compare the results with the previous ones.

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



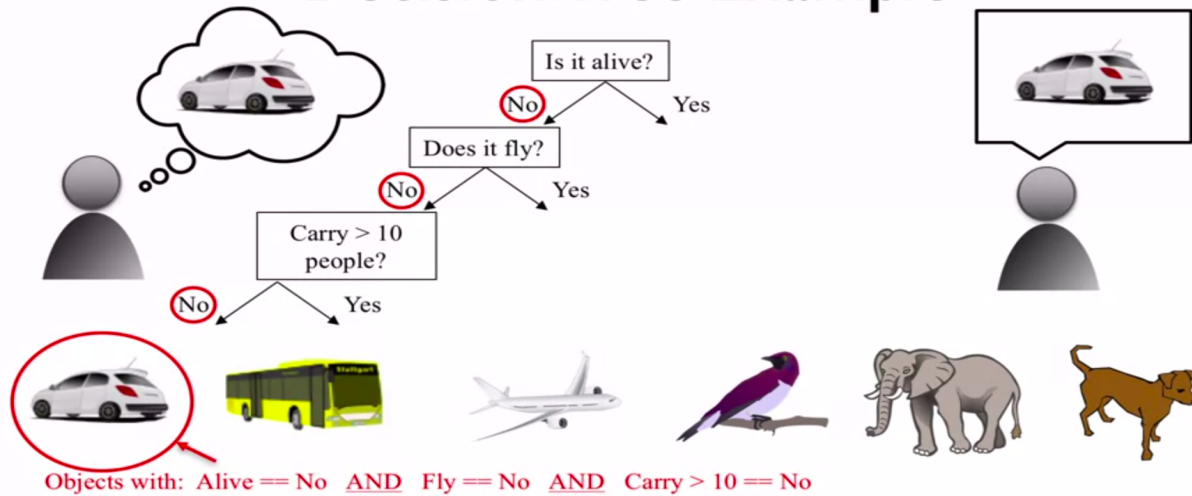
## Decision trees family

### Decision trees

Decision trees are a popular supervised machine learning method that can be used for both regression and classification. Decision trees are easy to use and interpret, thus this method can help to understand what features are the most

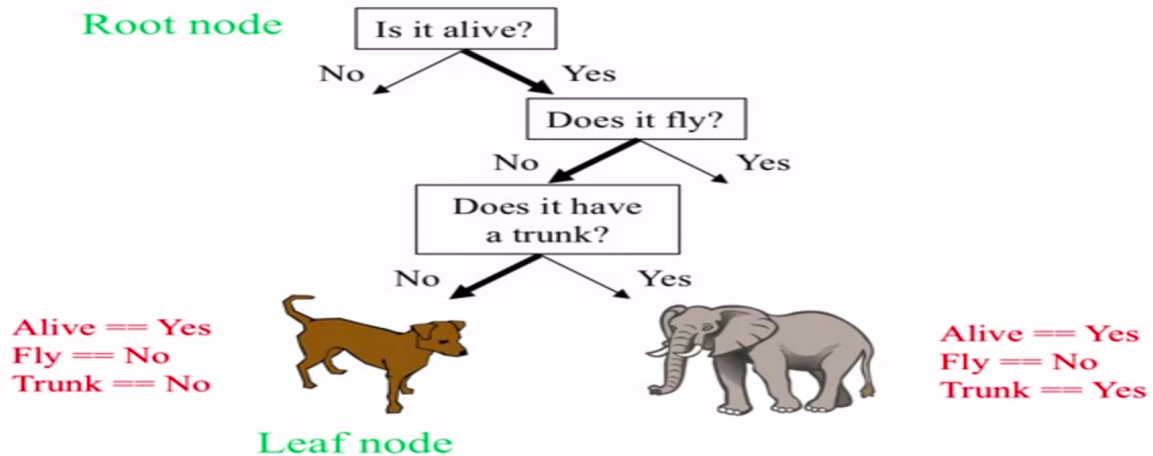
influential in the dataset. Basically, decision trees learn a series of explicit if then rules on feature values that result in a decision that predicts the target value. The simple example is shown below.

## Decision Tree Example



The inset above shows the example of rules (questions) the decision tree learned to classify the objects based on their features. The questions learned by the algorithm produces a tree with different branches based on the related answers (yes/no).

## Decision Tree Example

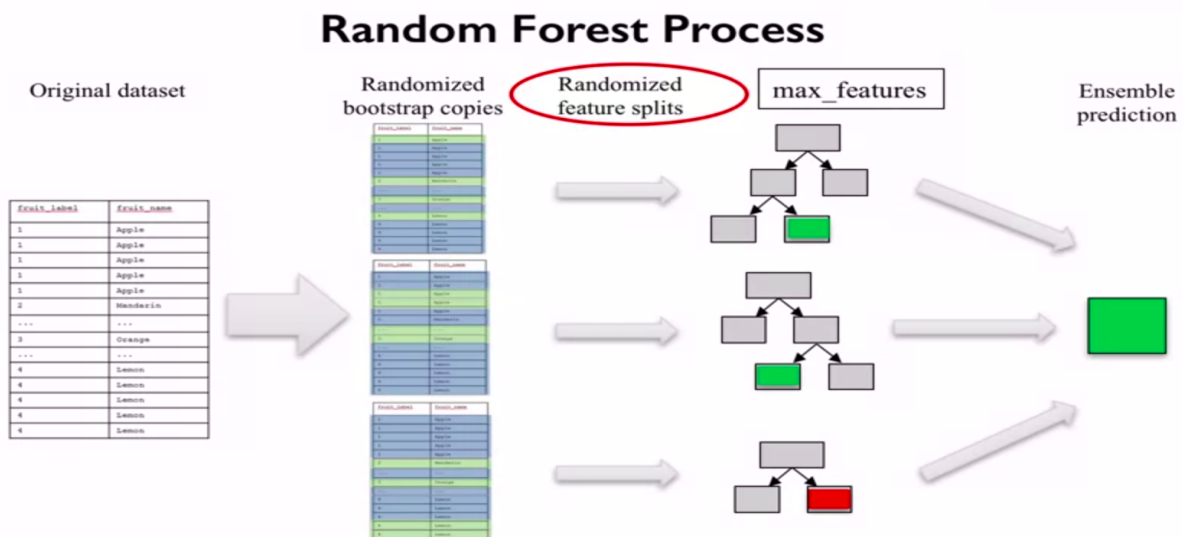


There are different parameters that are used to control the complexity of tree like max\_depth, max\_leaf\_nodes, min\_samples\_leaf and so on (you will see the effect of these parameters in the today's assignment). Thus by tuning them the one can reduce overfitting. The one key feature of decision trees is that they don't require specific scaling and normalization of data. The key parameters of decision tree is shown below.

- **max\_depth:** controls maximum depth (number of split points). Most common way to reduce tree complexity and overfitting.
- **min\_samples\_leaf:** threshold for the minimum # of data instances a leaf can have to avoid further splitting.
- **max\_leaf\_nodes:** limits total number of leaves in the tree.
- In practice, adjusting only one of these (e.g. max\_depth) is enough to reduce overfitting.

## Random forest

If we think about decision tree as about human, then the forest is a big amount of people with different opinions. In real life if the opinion of a big part of people about some thing agree, then there is a big probability that this opinion is a true one, that's the key idea behind random forest. Random forest is an ensemble of learning models. An ensemble takes multiple individual learning models and combines them to produce an aggregate model that is more powerful than any of its individual learning models alone. By combining different individual models into an ensemble, we can average out their individual mistakes to reduce the risk of overfitting while maintaining strong prediction performance. Each tree in random forest is built from the random sample of the data. There are some hyperparameters we need to choose for random forest classifier as number of trees, the number of features to consider when looking for the best split, etc.



The prediction using random forest is conducted in the following way:

- Make prediction for every tree in the forest.
- Combine individual predictions (for regression : mean of individual tree predictions)

Unlike the decision tree, random forest is hard to interpret. The key parameters of random forest are shown below.

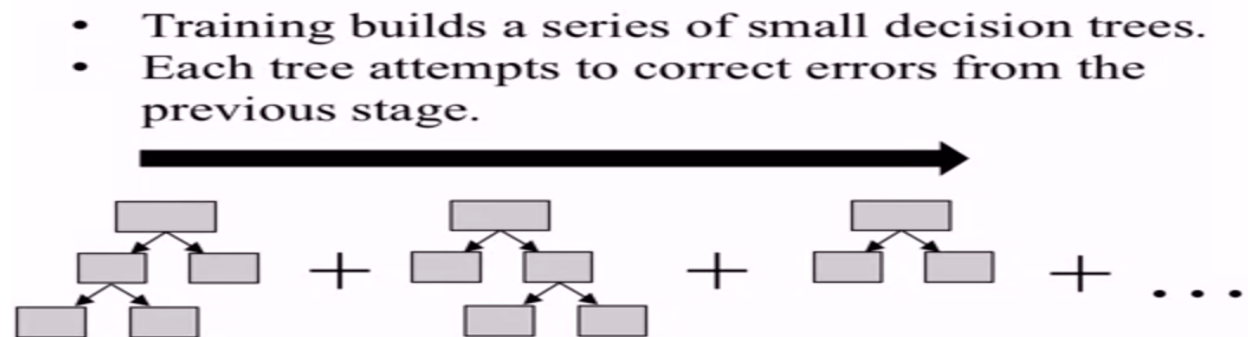


## Key Parameters

- **n\_estimators**: number of trees to use in ensemble (default: 10).
  - Should be larger for larger datasets to reduce overfitting (but uses more computation).
- **max\_features**: has a strong effect on performance. Influences the diversity of trees in the forest.
  - Default works well in practice, but adjusting may lead to some further gains.
- **max\_depth**: controls the depth of each tree (default: None. Splits until all leaves are pure).
- **n\_jobs**: How many cores to use in parallel during training.
- Choose a fixed setting for the **random\_state** parameter if you need reproducible results.

### GBDT (Gradient Boosted Decision Trees)

Like a random forest, GBDT uses an ensemble of decision trees to solve both regression and classification models. The key idea of GBDT is that it creates a series of trees, where each tree is trained that it attempts to correct the mistakes of the previous tree in the series.



- The learning rate controls how hard each new tree tries to correct remaining mistakes from previous round.
  - High learning rate: more complex trees
  - Low learning rate: simpler trees

GBDT often have a much higher performance than other decision tree based algorithms. The key parameters of GBDT are shown below.

### Key Parameters

- **n\_estimators**: sets # of small decision trees to use (weak learners) in the ensemble.
- **learning\_rate**: controls emphasis on fixing errors from previous iteration.
- The above two are typically tuned together.
- **n\_estimators** is adjusted first, to best exploit memory and CPUs during training, then other parameters.
- **max\_depth** is typically set to a small value (e.g. 3-5) for most applications.

---

**Note:** Many ideas represented on this page were taken from [this](#) course.

---

### Description of assignment

In today's assignment you will work with the models from decision trees family. You will use all the highlighted algorithms to solve the problem of predicting housing prices. Note that this is the last assignment in regression module. We hope that this section was useful to you and you will continue exploring machine learning sphere further.

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



### Classification

In this section you will work with classification models in order to solve the specified tasks. Please move on to the lessons.

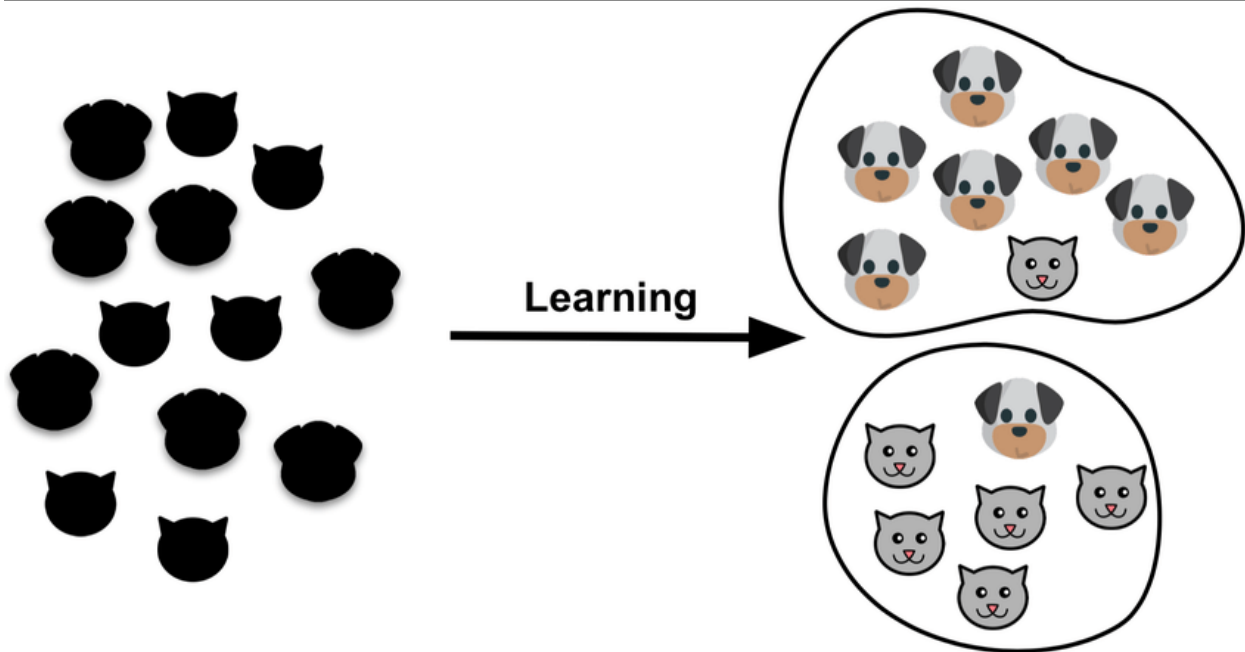
### Binary classification and core concepts of it

The problem of classification is fundamental for the field of machine learning, as many amazing applications based on the concepts of it were constructed. As the regression, classification is a part of supervised learning. Unlike the regression models, classification ones output the probability of a sample's belong to one of N classes. Thus, we should know the exact number of classes before training the model. Some applications that rely on the classification techniques are

the following : spam filters, fraud detection, object recognition, image classification, medical diagnostics, etc. Today we will speak about the binary classification, meaning that  $N=2$  (where  $N$  is a number of classes). One thing to notice is that in classification, the classes' names are mapped to integers, thus if we have two classes, the first one will be mapped to 0, and the second one to 1.

One of the simplest examples for understanding classification and not so easy to solve is to classify whether a dog or a cat is on image..

---

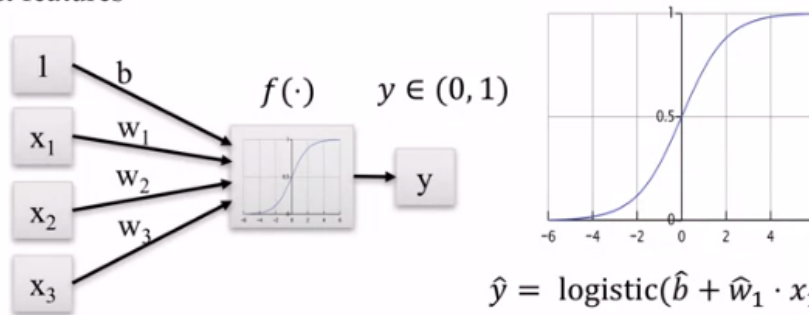


### Differences between classification and regression

The main thing is that the algorithms you already acquainted with, can be used for both regression and classification. However to use them for classification, some changes are needed. Firstly, instead of producing the discrete number our algorithms have to produce some sign/probability that the data point belongs to one of the predefined classes. This can be achieved by converting the linear output into the probability which varies in the interval from 0 to 1 (*sigmoid* function) and then use some threshold function to decide which class it corresponds to (that is how the *logistic regression* works).

## Linear models for classification: Logistic Regression

Input features



$$\hat{y} = \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \dots \hat{w}_n \cdot x_n)$$

$$= \frac{1}{1 + \exp[-(\hat{b} + \hat{w}_1 \cdot x_1 + \dots \hat{w}_n \cdot x_n)]}$$

The other option is to use a sign function on the linear output which is an analog of threshold function (that is how the SVM for classification works).

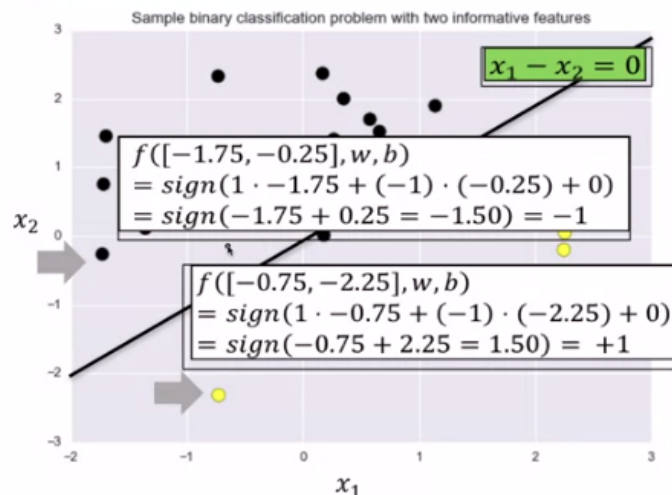
**Linear classifiers: how would you separate these two groups of training examples with a line?**

Feature vector                      Class value



$$f(x, w, b) = \text{sign}(w \cdot x + b)$$

$$\begin{aligned} x_1 - x_2 &= 0 \\ w &= [1, -1] \\ b &= 0 \end{aligned}$$



The second thing we need to change is the loss function. The usual choice for binary classification is *binary cross entropy* or *hinge* (binary cross entropy is usually used for logistic regression, whereas hinge - for SVM) loss.

**Note:** Understanding loss functions is out of scope of our introductory course, but we encourage you to visit [this](#) resource to get more intuition about this theme.

We won't consider changes in other algorithms as decision trees, random forest and so on, as these changes are minor. The main takeaway from it is that classification models produce a class label/probability.

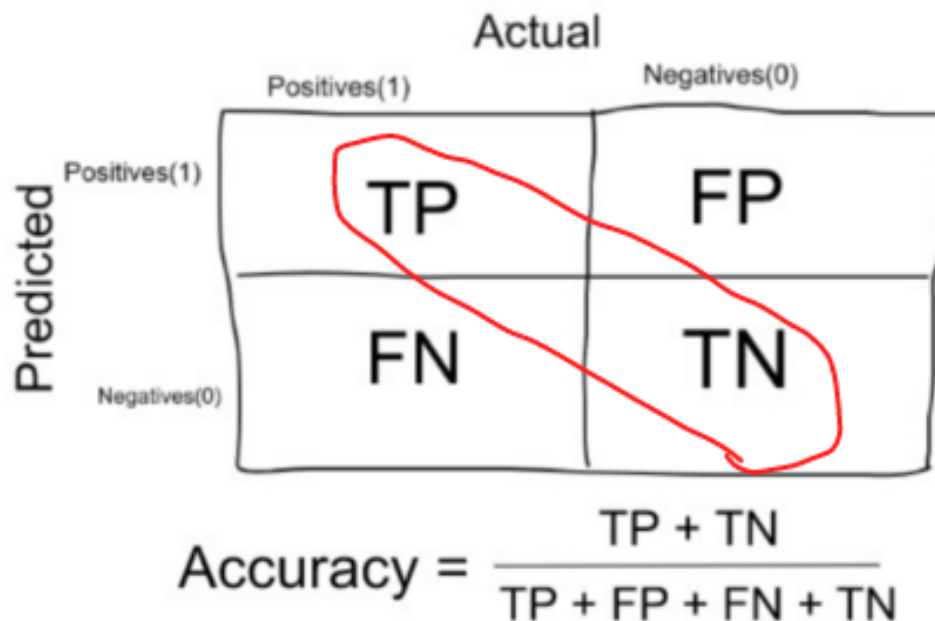
## Classification metrics

Speaking about classification it's mandatory to understand the data and pick the right tool for performance measurement. First of all, let's speak about such metric as *accuracy*. Accuracy is usually used in classification models when we deal with balanced dataset, meaning that each class has the same number of data points. To understand the metrics we will define next notations (all the metrics scores are in the interval from 0 to 1):

- TP - true positives, number of samples related to the first (positive) class that were predicted correctly.
- FP - false positives, number of samples related to the first (positive) class that were predicted incorrectly.
- TN - true negatives, number of samples related to the second (negative) class that were predicted correctly.
- FN - false negatives, number of samples related to the second (negative) class that were predicted incorrectly.

	Actual = Yes	Actual = No
Predicted = Yes	TP	FP
Predicted = No	FN	TN

Accuracy is calculated in the following manner (number of all the correct predictions divided by the number of overall samples in the dataset):



It's preferred to use accuracy when dealing with balanced dataset, as if your dataset is imbalanced, the accuracy won't capture the real performance of the classifier. Suppose that you are to solve the next problem: There is some data corresponding to user activity, the abnormal user activity as watching videos 24/7 - corresponds to the fraud one. You for sure won't have balanced classes, as much more people will have non fraud activity. Thus, using accuracy as the main metric you might be reassured by its score, whereas in fact you classifier performs bad.

One solution to the highlighted problem is to use such metrics as *precision*, *recall* and *f1 score*.

Precision is calculated in the following manner (number of correct predictions for the first (positive) class divided by total number of data points corresponding to first (positive) class) :

		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall is calculated in the following manner (number of correct predictions for the first (positive) class divided by the sum of the previous value and number of incorrect predictions for the second (negative) class):

		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

$$\text{Recall} = \frac{TP}{TP + FN}$$

It is clear that recall gives us information about a classifier's performance with respect to false negatives (how many did we miss), while precision gives us information about its performance with respect to false positives (how many did we caught). Thus, there is a kind of a trade off between precision and recall. As the result it's recommended to use the f1-score which is a harmonic mean of precision and recall.

F1 score is calculated in the following manner (harmonic mean of precision and recall) :

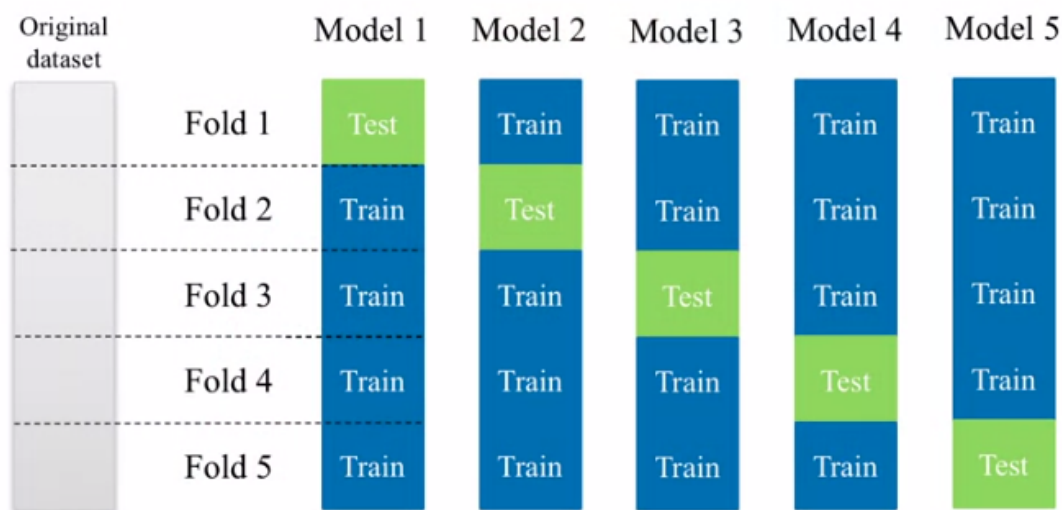
$$F_1 = \left( \frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

**Note:** In order to get more information and construct a much more robust intuition concerning metrics for classification please visit [this](#) resource and [this](#) course.

### Cross validation and greedy search

The last topic for today is about a new method of validating and evaluating the performance of our classifier. So far we validated our algorithm only on a test set, which is actually just the portion of data that is randomly cut off our overall dataset. Actually this approach is bad as it relies on only one random subset of data, and it isn't enough to state that the algorithm is good or bad. Thus, for not very complicated classifiers and small-medium datasets the other approach named *cross validation* is used. The basic idea of cross validation is to train and test classifier on different subsets of data and then receive the predefined score (f1, accuracy, precision, etc.) for each subset. After that the score can be averaged across subsets to get an adequate estimate of the algorithm's performance.

## Cross-validation Example (5-fold)



Having a chance to get an adequate estimate of each classifier, the one can then choose the best with respect to the averaged metric across subsets (f1/accuracy/etc). This strategy is called greedy search.



## Description of assignment

Today, you will work hard to solve the classification problem from [kaggle](#), as it's really useful to workout real problems and check for strength your knowledge, in addition it's super fun. In this first assignment, you will firstly make exploratory data analysis, feature engineering and then use cross validation along with greedy search to find the best classifier.

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



## Hyperparameters tuning

Tuning model's parameters is a very tedious and fragile work. Different algorithms have different sensitivity for different hyperparameters, that's why in order to get pick nice tunable parameters, the one need to investigate the "black box" of the algorithm, meaning it's implementation. But sometimes we just don't have enough time to make it. Suppose that you started working with some algorithm for the first time and based on it you should provide a nice solution in two days. Two days is not an enough term to understand all the subtleties of a new algorithm. What should the one do in this situation? Fortunately, the one can just try the variety of different hyperparameters and then choose the best ones.

## Grid search

One option is to use an exhaustive grid search which generates candidates (hyperparameters) from a grid of parameter values specified beforehand. It's very easy to understand it, but it's not very useful for a large amount of data with lots of features.

## Randomized parameter optimization

Instead of iterating through the entire parameters' space, randomized search samples candidates from a distribution over possible parameter values. Two main benefits of it are the following :

- A budget (by budget we mean computational time and power) can be chosen independent of the number of parameters and possible values.
- Adding parameters that do not influence the performance does not decrease efficiency.

### Tuning through minimization

The other interesting idea is to think about hyperparameters tuning as about the other optimization problem. By stating the problem that way, we can apply another algorithm (for example forest minimize) on top of ours to find the best hyperparameters by minimizing loss or maximizing accuracy/recall/etc. This solution is known to be much more precise than the two exposed before. The main benefits of this approach are the following :

- It's computationally efficient, as the search of hyperparameters is done by the learning algorithm, and thus nice values can be found much faster.
- With each iteration, the algorithm “on top” is supposed to find the hyperparameters which make the performance of the algorithm better, whereas random an exhaustive search can never find the best one.

---

**Note:** We won't consider running through minimization and randomize parameter optimization, but if you are interested you can learn about them [here](#) and [here](#).

---

Anyway it's beneficial to have an idea about in which space to seek for best hyperparameters, as it can speed up the process of hyperparameters tuning.

### Description of assignment

For today, you will continue working with the dataset from kaggle. You will have a change to observe the lift in kaggle score after applying hyperparameters tuning and making a new feature. Good luck!

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



### Multiclassification

In a real world, not all the problems are binary and the target is a categorical one. Thus, we need to understand what changes are needed to make our model appropriate for multiclassification. What we can do is convert our multiclassification problem into the chain of binary ones, by training a bunch of classifiers that predict one class against all the other classes, where number of classifiers equals number of classes.

# Multi-class Classification with Linear Models

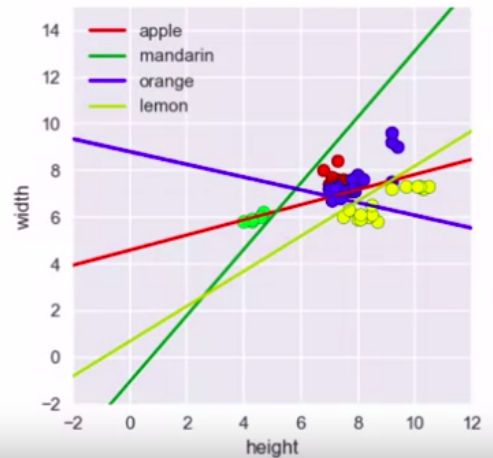
```
clf = LinearSVC(C=5, random_state = 67)
clf.fit(X_train, y_train)

print(clf.coef_)

[[-0.23401135  0.72246132]
 [-1.63231901  1.15222281]
 [ 0.0849835   0.31186707]
 [ 1.26189663 -1.68097   ]]

print(clf.intercept_)

[-3.31753728  1.19645936 -2.7468353  1.16107418]
```



Actually, that's not the best solution for big complex datasets and sophisticated models. However there is a solution of using a *softmax* function with loss function known as *categorical cross entropy*. It's out of scope of our course, but you can learn about it in this [course](#).

In order to calculate metrics for multiclassification we need to apply the next changes :

## Multi-Class Evaluation

- **Multi-class evaluation is an extension of the binary case.**
  - A collection of true vs predicted binary outcomes, one per class
  - Confusion matrices are especially useful
  - Classification report
- **Overall evaluation metrics are averages across classes**
  - But there are different ways to average multi-class results: we will cover these shortly.
  - The support (number of instances) for each class is important to consider, e.g. in case of imbalanced classes
- **Multi-label classification: each instance can have multiple labels (not covered here)**

While working with multiclassification, sometimes it's useful to examine the decision boundaries of the classifier, as it gives an idea of how our model separates different data instances. But if we work with high dimensional data, we can't simply visualize it in a human friendly manner. Thus, we need to reduce the dimension of our data to 2d or 3d. For this purpose *unsupervised learning* methods are used. However, *unsupervised learning* is the other part of the course, we will use its algorithm called *PCA* to achieve the exposed goal.

### Description of assignment

In your last assignment in classification section you will explore multiclassification with NIST handwritten digit dataset and visualize the decision boundaries of logistic regression model. We hope that this section was useful to

you and you will continue exploring machine learning sphere further.

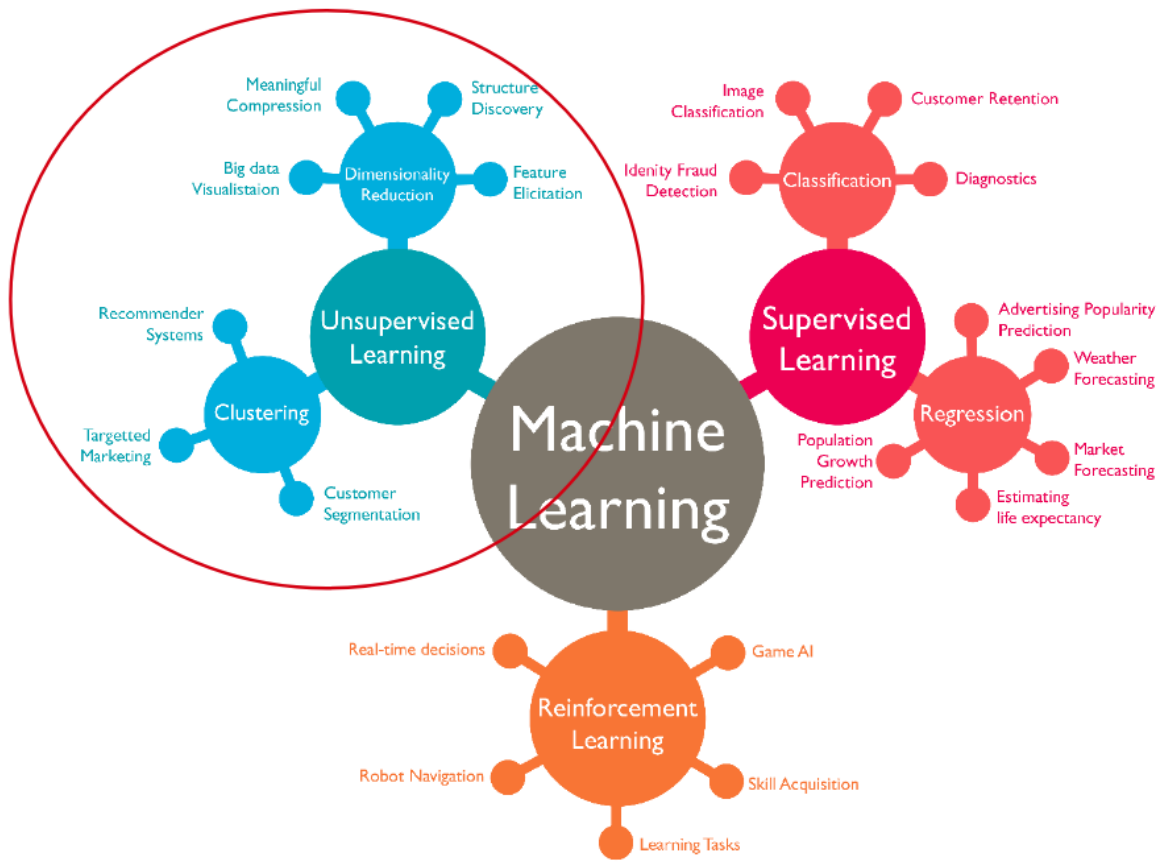
In order to check the gained knowledge, please carry on with the quiz related to this lesson.



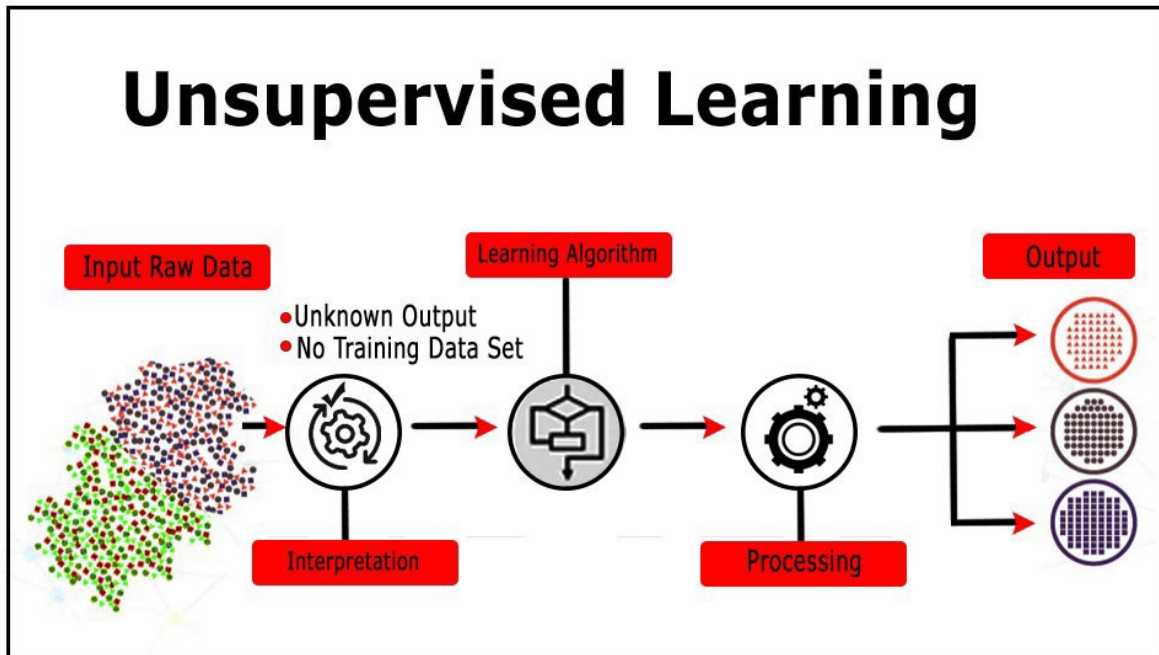
## **4.4.2 Unsupervised learning**

### **Clustering with unsupervised learning and key concepts of it**

Unsupervised learning techniques have many applications and in general can be divided into two groups by its usage : clustering and dimensionality reduction.



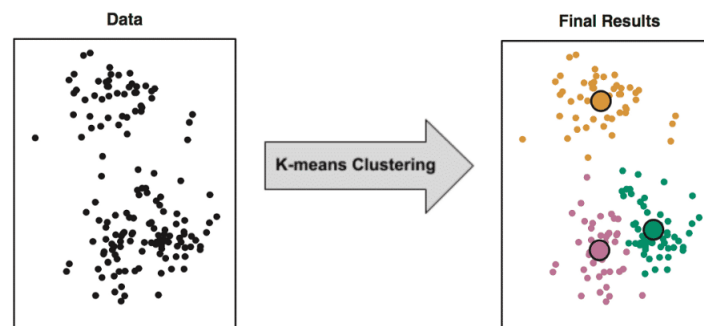
Some examples of applications based on unsupervised learning include clustering and grouping of users based on their profiles, data compression, visualization of high dimensional data and identifying withhold patterns in data. Unsupervised learning algorithms need only  $X$  (features) without  $y$  (labels) to work, as they tend to find similarities in data and based on them conduct clustering.



One of the most simple and yet powerful algorithms of unsupervised learning is KMeans.

## Understanding KMeans

Objective of K-means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number ( $k$ ) of clusters in a dataset. The one should understand that *cluster* is just a collection of data points aggregated together because of certain similarities.  $k$  is a hyperparameter that should be set manually.  $k$  refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster. Thus  $k$  is the explicit number of clusters you want KMeans to group your data into. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the KMeans algorithm identifies  $k$  number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.



A disadvantage of the algorithm is that the one needs to set the number of clusters ( $k$ ) manually. However there is a technique called “elbow” method which can give an insight about the best number of clusters for your data.

---

**Note:** You will get to know about “elbow” method in today’s assignment.

---

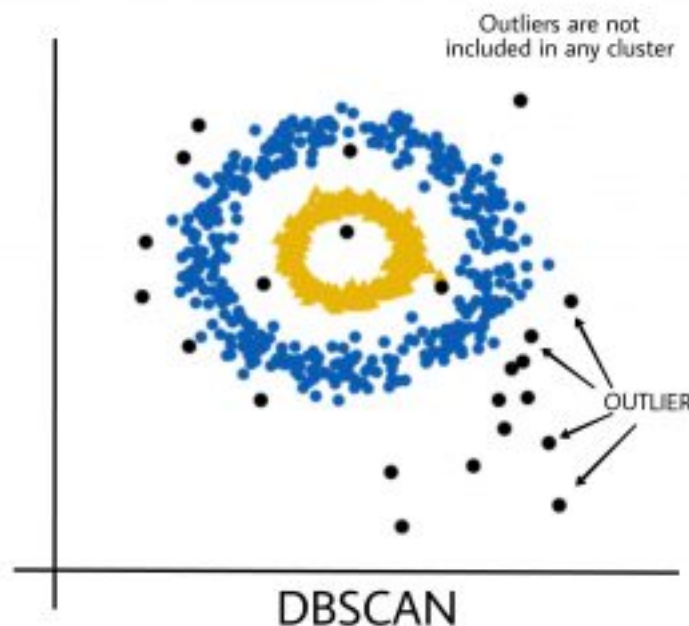
## Understanding DBSCAN

DBSCAN is a clustering method that is used in machine learning to separate clusters of high density from clusters of low density. Given that DBSCAN is a density based clustering algorithm, it does a great job of seeking areas in the data that have a high density of observations, versus areas of the data that are not very dense with observations. DBSCAN can sort data into clusters of varying shapes and determine best number of clusters itself, which eliminates a need of setting it explicitly. DBSCAN firstly divides the dataset into  $n$  dimensions, then for each point in the dataset, DBSCAN forms an  $n$  dimensional shape around that data point, and counts how many data points fall within that shape. Finally, DBSCAN iteratively expands the cluster, by going through each individual point within the cluster, and counting the number of other data points nearby.

---

**Note:** Many ideas were taken from [this](#) article.

---



However, DBSCAN has some disadvantages :

- DBSCAN does not work well when dealing with clusters of varying densities.
- DBSCAN does not work well with high dimensional data.

## Description of assignment

In today's assignment you will have a hands on introduction to clustering with KMeans and DBSCAN. You will learn how to apply the highlighted methods to group data into different clusters, understand "elbow" method and learn how to use it on practice, and grasp the advantages of DBSCAN. Have fun!

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



### Dimensionality reduction

The other important application of unsupervised learning is dimensionality reduction. It's mainly used for two purposes : data visualization and data compression.

Imagine, that you are working with high dimensional data and you want to somehow visualize data samples. As you have lots of features, your plot won't be very comprehensive and you won't get any useful insights from it. What you can do is to compress your features to just 2 or 3, thus you can easily plot them using 2D or 3D plot (that's also known as *feature projection*).

Now imagine that you have 30 features in your dataset and you need to train a model to classify something based on this features. First of all, using all the features available could slow down the training part a lot. Secondly, after a certain point, the performance of the model will decrease with the increasing number of elements. This phenomenon is often referred to as "The Curse of Dimensionality". One solution, is to use unsupervised learning algorithms to reduce number of features, aggregate them.

---

**Note:** We haven't included feature selection techniques in this lesson, but we encourage you to read [the article](#) regarding it.

---

### Understanding PCA

PCA is a technique for reducing the number of dimensions in a dataset whilst retaining most information. It is using the correlation between some dimensions and tries to provide a minimum number of variables that keeps the maximum amount of variation or information about how the original data is distributed. PCA uses mathematical techniques to reduce number of dimensions, mainly something known as the eigenvalues and eigenvectors of the data-matrix.

### Understanding t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is another technique for dimensionality reduction and is particularly well suited for the visualization of high-dimensional datasets. Contrary to PCA it is not a mathematical technique but a probabilistic one. t-SNE looks at the original data that is entered into the algorithm and seeks how to best represent this data using less dimensions by matching both distributions. The way it does this is computationally



heavy and therefore there are some limitations to the use of this technique. For example one of the recommendations is that, in case of very high dimensional data, you may need to apply another dimensionality reduction technique before using t-SNE.

### Description of assignment

In your last assignment in unsupervised learning section you will work with both t-SNE and PCA to reduce the dimensions of your data and visualize it effectively. Also, you will explore a fun application of photo compression using KMeans. We hope, that you liked our course and good luck!

In order to check the gained knowledge, please carry on with the quiz related to this lesson.



## 4.5 FAQ

In this section you can find frequently asked questions and answers to them. If your question isn't included, you can ask it in the issues of official [github](#) repository of the course.

---

**Note:** With time new frequently asked questions will be added to this section.

---

### 4.5.1 How can I run assignments on my own machine?

In order to run assignments on your own machine, you should do the following steps :

- 1) Clone the official github repository *git clone https://github.com/HikkaV/DS-ML-Courses/*.
- 2) Create virtual environment (either using using virtualenv or anaconda env) and activate it (see [this](#)).
- 3) In the course folder go to assignments and install the requirements : *pip install -r requirements.txt*.
- 4) Use jupyter notebook or other tool to work with assignments (you can run jupyter notebook by executing this command : *jupyter notebook*).

### 4.5.2 How can I contribute to the course to make it better?

We value every input that will make the course better, that's why everyone is encouraged to follow the discussion in the issues of official [github](#) repository and make your merge requests.

### 4.5.3 What is the exact list of courses which this course was built on?

- 1) Applied Data Science with Python specialization from MIT university .
- 2) Deep Learning specialization .
- 3) Machine Learning course from Andrew Ng .