PhUSE EU Connect 2018

Paper ML01

Introduction to Machine Learning

Nicolas Dupuis, d-Wise, Basel, Switzerland
Kevin Lee, Company, City, Country

ABSTRACT

Machine Learning (ML) is a buzz word nowadays. It is changing many aspects of our lives, *e.g.* self-driving vehicles, online search and recommendations, fraud detection, image and video recognition, Natural Language Processing and many more. The pharmaceutical industry is in an interesting position as we also are experts in programming, statistics and data processing. This provides a great opportunity for us to use ML. PhUSE recently initiated a working group to gather and curate knowledge on ML for the benefit of our industry. The presentation will introduce the basic concepts of ML, supervised or unsupervised. We will take a concrete example with the famous KNN algorithm. The presentation will also discuss the current and future ML implementation in the pharmaceutical industry. Our goal is to help programmers and statisticians to lead this exciting and disruptive technology in the pharmaceutical industry.

INTRODUCTION

This paper serves as a basic introduction to Machine Learning. No existing knowledge is assumed.

BASIC DEFINITIONS

Machine learning is an ensemble of statistical techniques that give computers the ability to *learn* using large amount of data, without being explicitly programmed. Imagine you want to create a program that can read hand-written digits on a piece of paper. In traditional programming, you would have to write a lot of "if-then-else" code. Like "If there's a loop and a stick below on the right side, then it's a 9". Programming this would be challenging because we all have different hand writing. ML takes a different approach, if you provide enough pictures the digit 9 labelled as such, then the algorithm will learn from the data and make predictions on unseen data. The same algorithm could be used for a very different problem using very different data.

ML algorithms can make predictions or provide knowledge (*e.g.* "*Is this email a spam?*" or "*What can I learn from my customers behavior?*" *etc.*). Think about Gmail's way of moving your spam emails into the spam folder. Obviously, Google does not employ thousands of people to read your emails and make that kind of decision. So, it clearly can automate work, replace human labor and solve many complex business problems. It is disrupting many industries and every day we learn about a new application. For example, performing a medical diagnosis in the future will probably be done using ML. It's faster, more accurate, has less bias and can use the incredible amount of scientific literature available, which no man can absorb anymore.

The power of ML relies on data, famously called the new oil. Data is indeed crucial but only if you can turn it into knowledge and actions. It must be tidy and used appropriately. The real new oil is in fact labeled data, used for training the ML algorithms. For example, Google paid people around the world to label pictures, *e.g.* "this is a cat", "this is a house" etc., just to train their image recognition algorithm. A major part of the work when implementing ML is to have the right training data.

There are two major types of Machine Learning: **supervised** and **unsupervised**. Let's see the difference.

SUPERVISED MACHINE LEARNING

You can use the supervised algorithms when you already have **labeled** data. For example, if the goal is to decide whether an email is a spam or not, you need a collection of emails that a human has reviewed and labeled (spam or not spam). The algorithm you chose will digest the data and make predictions on unseen data.

In another words, Y = f(X) where Y and X are known in the training data. The goal is approximate the mapping function to be able to predict Y for unseen X.

There are many existing algorithms that you can choose. For example:

- K-Nearest Neighbors
- Linear and Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machine
- Artificial Neural Network
- Deep Neural Network

The choice depends on many factors: what data do you have (numeric, categorical) and how much, what outcome do you expect, what are your processing capabilities etc. This field is constantly evolving, and progress is made every year.

These algorithms can be used for **classifications** problems (the output variable is categorical, *e.g.* "cat" or "dog") and/or **regression** problems (the output variable is a numeric value, *e.g.* "cost in USD").

UNSUPERVISED MACHINE LEARNING

The unsupervised algorithms are used when the data is not labeled, and the correct answer is not known. The goal is to model the underlying structure or distribution and learn about the data.

These algorithms can be used for **clustering** problems (where you want to discover the inherent groupings in the data, *e.g.* grouping customers by purchasing behavior) or **association rule** problems (to discover rules in the data, *e.g.* people that buy X also tend to buy Y).

SHORTCOMINGS

Effective machine learning remains difficult and often fails to deliver the expected value. It can be caused by the lack of (suitable) data, data bias, badly chosen tasks and algorithms, wrong tools and people. The quality of the outcome will not be better than the input data. If you throw garbage in, you will get garbage out.

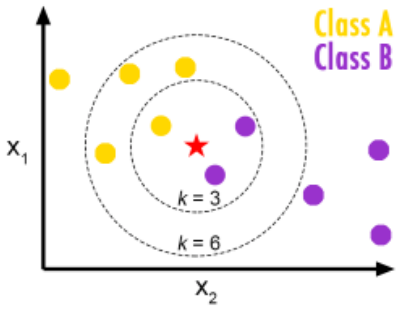
Even when the planets align, mistakes or failures are possible. Google experienced a bad one recently. Its image recognition algorithm mistakenly labeled people with black skin as gorillas. They had to remove gorillas' images/labels from their training data to prevent the algorithm from being confused, and Google to be embarrassed! I suppose they are working on improving the accuracy before trying again with gorillas…

A well-known issue is the **black box** effect. It's often difficult or sometimes impossible to understand why a prediction was made. Obviously explaining it to someone (*e.g.* Health Authorities) is even harder.

A SIMPLE EXAMPLE: THE K-NEAREST NEIGHBORS CLASSIFIER

KNN is a well-known, robust and versatile classifier that is often used as a benchmark for more complex classifiers (*e.g.* Neural Networks or SVM). It's simple to understand and yet powerful. It's therefore used in a variety of applications such as economic forecasting, data compression and genetics.

The idea is to plot 2 features of your labeled data (*e.g.* X: color of the hair, Y: color of the skin) for different categories in your data (e.g. country of origin). A new data point is classified by a majority vote of its nearest neighbors. The hyperparameter **k** represents how many neighbors we consider for the vote.



The assigned class for the new data point (red star) depends on k:

- k = 3. We see 2 B versus 1 A around it => class B
- k = 6. We see 4 A versus 2 B around it => class A

THE IRIS DATASET

Let's take a concrete example. We will use the famous Iris dataset. It contains 150 observations describing the characteristics (sepal/petal width and length) of 3 iris flower species (Versicolor, Setosa and Virginica).

Iris Versicolor **Iris Setosa** **Iris Virginica**

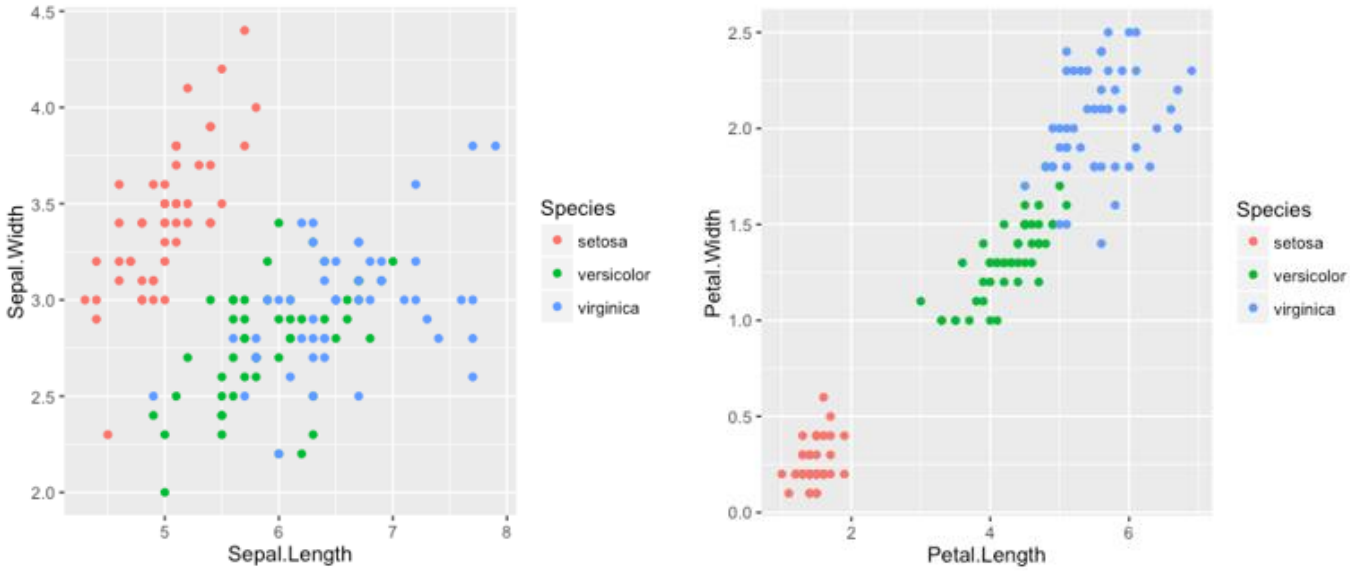
Using this dataset to discover Machine Learning is the equivalent of doing a "Hello World" in a programming language! Talking about language, I'll use Python in the Jupyter Notebook environment for these examples. You should download the Iris dataset and save it locally if you want to run the below code.

```
# Let's import all the libraries we need
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
%matplotlib inline

# define column names
names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

# load training data in a dataframe and print the first 5 records
df = pd.read_csv('iris.data', header=None, names=names)

# let's display the first 5 observations
df.head(5)
```

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

We have here input 4 features (the flower characteristics) and 1 output class (the species).

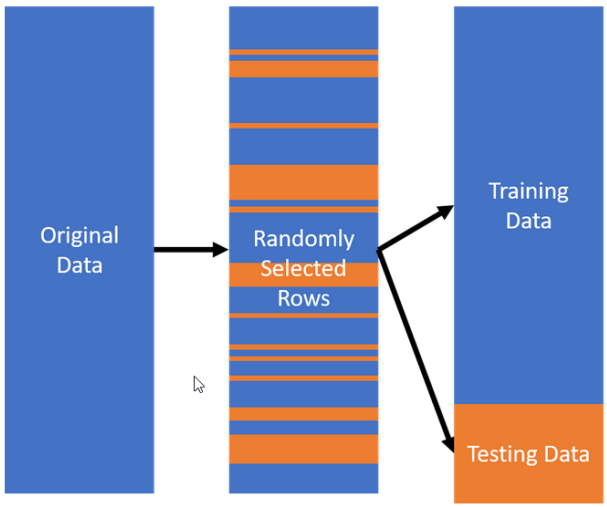
We can plot these (I have used R) and see some nice clusters. Learning from the data should be convenient with K-NN.



CLASSIC WORKFLOW

When facing such classification problem, first we choose a classifier. We saw earlier that there are many. In this example, we picked KNN.

The second task is to split the labeled data into two sets: a training and a test set. The first one will be used to teach the algorithm, the second one to measure the accuracy of the predictions. The split should be made picking records randomly to avoid bias. Fortunately, Python and Scikit-learn will do this easily for you.



Next, we train our classifier with the training data. After that, the classifier can make predictions on either the test data (which was already labeled) or unseen data (not labeled). Making predictions on the test set allows to measure lots of things: accuracy, error rate, false positive/negative etc. Maybe we will have to tweak the hyperparameters of our algorithm (k in our example), maybe we want to plot the error rate versus k and pick the value of k that seems optimal with this data. Once optimized, our classifier is ready for the big game: unseen data.

LET'S GIVE IT A TRY

```
# split data into training and test sets
X = np.array(df.iloc[:, 0:2])
y = np.array(df['class'])

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size    = 0.33,
                                                    random_state = 42)

# Instantiate the classifier into an object
knn = KNeighborsClassifier(n_neighbors = 1)

# Fit the model with training data
knn.fit(X_train, y_train)

# Now the classifier is ready to make predictions on the test data
y_pred = knn.predict(X_test)

# Print some predictions
Print(y_pred[:5])

['Iris-versicolor', 'Iris-setosa', 'Iris-virginica','Iris-versicolor', 'Iris-
virginica']
```

Let's see how many of the predictions on the test data were true or false:

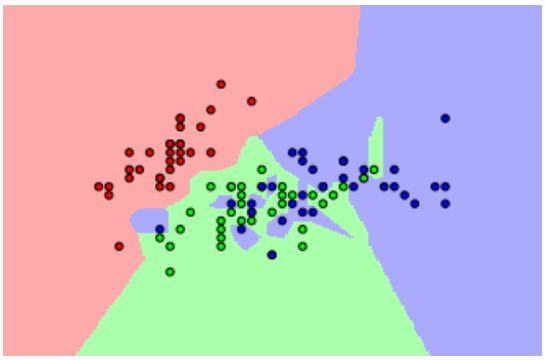
```
# How many predictions were correct?
print(y_pred == y_test)
```

```
[ True  True  True  True False  True  True  True  True  True  True  True
  True  True  True False  True  True  True  True  True False  True  True
  True  True False  True  True  True  True  True False  True  True False
 False  True  True  True False False False  True  True  True False  True
 False  True]
```

Finally, we can print the accuracy score using a function from scikit-learn.

```
accuracy_score(y_pred, y_test)
0.72
```

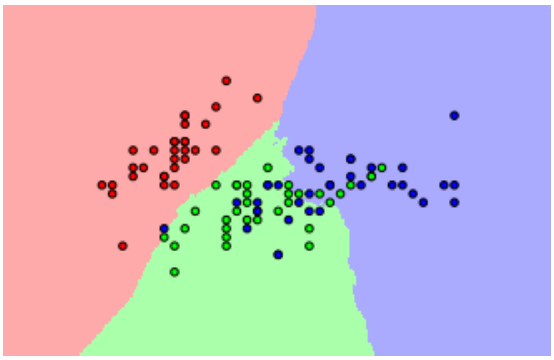
Our classifier seems doing good but it's not perfect. Let's plot the decision boundaries for 3 values of k:



With k = 1, we seem to be **overfitting** the data.

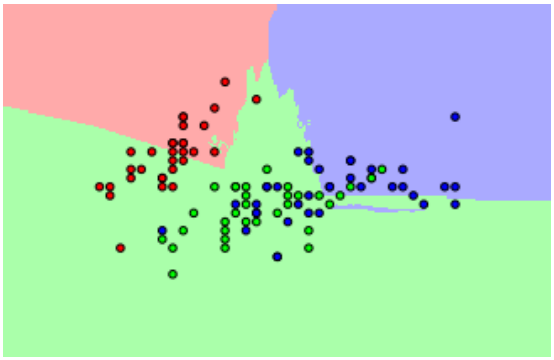
The boundaries are jagged, we are way too close to the training data and blind to the overall distribution.

It models too well the training data and will probably not generalize well to new data.



With k = 24, the boundaries are smoother. We still model the training data well, but we are now more resilient to outliers.

This is the **sweet spot**.

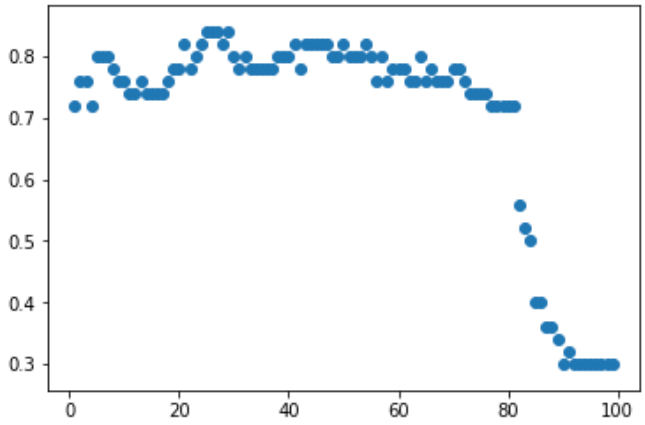


With k = 82, the model is clearly **underfitting**.

This does neither model the training data nor generalize to new data.

To find the sweet spot, we can calculate the accuracy for a wide range of k values. It's easy to find the highest accuracy and the therefore the best k (for this data). We can also plot it to make it visual.

```
# Measure accuracy for k from 1 to 100
all_k = range(1, 100, 1)
scores = []
for k in all_k:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    scores.append(accuracy_score(y_pred, y_test) )

# Plot accuracy versus k
matplotlib.pyplot.scatter(all_k,scores)
matplotlib.pyplot.show()
```



We used only 2 features in this example. Using the 4 available features in this Iris dataset allows to increase much more the accuracy (~0.98%).

THE ECOSYSTEMS

There are several ecosystems to implement ML, open source or commercials. Here are 4 interesting ones:

- The **Python** programing language and the **Scikit-Learn** library have become the most popular solution for ML implementation. Many books or MOOCs available for Python. Scikit-learn has plenty algorithms available: Classification, Regression, SVM, Clustering, Dimensionality Reduction, Decision Trees, Random Forests. It's usually the starting point when learning ML.
- The **R** programming language also offers some libraries for ML implementation. It seems that R is more used for data manipulation and wrapping Python libraries.
- TensorFlow is an open source Python framework developed by Google. It offers more recent and powerful techniques, especially neural networks.
- **SAS** has a module called "SAS Visual Data Mining and Machine Learning". It's not free and therefore has limited players. It offers several algorithms for regression and classification.

CONCRETE APPLICATIONS

ML is a transformative technology in many fields. The number of applications is exploding.

- Handwriting recognition, like reading cheques
- Interface with humans: chatbots, Siri, Alexa etc.
- SPAM detection
- Customers recommendation: Amazon, Netflix, Spotify etc.
- Sentiment analysis using Natural Language Processing: Amazon, Facebook
- Alpha Go was the first AI program to have beaten a Go world champion. There's an interesting movie about this story which I recommend.

There are also more and more applications in the pharmaceutical industry:

- Drug discovery and candidate selection
- Medical image recognition (Scan, MRI etc.)
- Medical diagnosis, sometimes saving lives where doctors couldn't reach a diagnosis
- Optimum site selection and recruitment
- Data anomality detection
- Personalized medicine
- Operational data about clinical trials
- And maybe one day soon, SDTM, ADaM and TFL?

The adoption is slow in Pharma compared to other industries. It's probably due to regulatory constraints. ML operates as a black box and we need to prove and validate models to explain the outcome. There are big investments in Healthcare and more and more AI startups are coming the Pharma industry. GAFA are also very interested to disrupt the way we work. The market is growing (40% annually) and is estimated to be at 10 billion USD in 2024.

There's a talent shortage, which is a great opportunity for us.

EDUCATING FOR THE FUTURE

This PhUSE initiative started in 2018 and is aiming to raise awareness and educate our industry on specific topics, like ML, Data Engineering or Design Thinking. The teams will gather and curate materials that any PhUSE member can use to ease the learning curve. White papers, online materials and PhUSE presentations will be provided to you. You are welcome to join the effort!

CONCLUSION

This revolution is coming, embrace it! Can you think of a useful application of Machine Learning in your daily work?

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Nicolas Dupuis
d-Wise
Basel, Switzerland
Nicolas.dupuis@d-wise.com