DataScience®
foundation
MAKE BETTER DECISIONS

Hands-On with your First ML Model

Author, Mayank Tripathi

A Data Science Foundation White Paper

March 2020

----

www.datascience.foundation

Copyright 2016 - 2017 Data Science Foundation

Hope you were following along with the various posts on Data Science; Machine Learning. And by this time we now know that the objective of any data science project is to derive valuable knowledge for the business from data in order to make better decisions. It is the responsibility of data scientists to define the goals to be achieved for a project. When we mention data science, we usually think about machine learning, and at-time it gets mixed up with each other, and gets confused in both.

So in-short Machine learning is the field of building algorithms that can learn patterns by themselves without being programmed explicitly. Which I have tried to explain in my previous post.
Refer to https://datascience.foundation/datatalk/understanding-why-machine-learning.

So machine learning is a family of techniques that can be used at the modeling stage of a data science project.

Going deeper, lets understand what is a model, and then having a basic understanding of how Machine Learning can be done, i mean having the hands dirty.

What Is a Model?

A machine learning model learns patterns from data and creates a mathematical function to generate predictions.

A supervised learning algorithm will try to find the relationship between a response variable and the given features.

Refer to https://datascience.foundation/datatalk/machine-learning-algorithm to understand different types of ML Algoriths.

If you are from a mathematical background then you might be aware of the mathematical function, which can be represented as a function $f()$, that is applied to some input variables, X (which is composed of multiple features), and will calculate an output (or prediction) as $\hat{y}$.Typically the formula will be as

$$\hat{y} = f(x) = f(x1, x2, \ldots.., xn)$$

Probably i will not make this article more boring, probably if you need more details, please add a comment and will share the details. Not all audiences are interested in knowing the back-process.

Will directly jumping to make our hands dirty.

Here I will be using the scikit-learn (or sklearn) package, also once you have learned how to train one algorithm, it is extremely easy to train another one with very minimal code changes.With sklearn or any other ML package, there are four main steps to train a machine learning model:

1. Instantiate a model with specified hyperparameters (if any) → this will configure the machine learning model you want to train.
2. Train the model with training data → during this step, the model will learn the best parameters to get predictions as close as possible to the actual values of the target.
3. Predict the outcome from input data → using the learned parameter, the model will predict the outcome for new data.

4. Assess the performance of the model predictions → for checking whether the model learned the right patterns to get accurate predictions.

Please remember that in a real project or testing any model, there might be more steps depending on the situation, but for simplicity, we will stick with these four steps for now. I will try to share more posts / articles to cover the other steps. Above 4 are generic one.

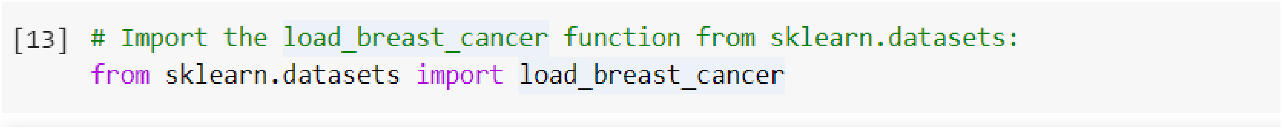
First we need to import the Data-Set. Here I am taking the example of Breast Cancer data-set, which is freely available with the sklearn package.

Also I am using google.colaboratory (it's free to use, one just needs to have a google drive account refer to https://colab.research.google.com/notebooks/welcome.ipynb),
you can also use Jupyter Notebook (https://cocalc.com/doc/jupyter-notebook.html).

I will attach the complete code for reference.

Assumption : Having a basic understanding of Python.

We will build a machine learning classifier using RandomForest from sklearn to predict whether the breast cancer of a patient is malignant (harmful) or benign (not harmful).Ignore the number in brackets. It is just the execution count of that cell.

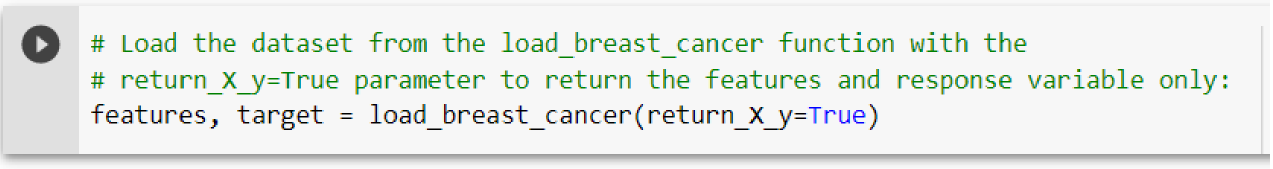


```
[13] # Import the load_breast_cancer function from sklearn.datasets:
     from sklearn.datasets import load_breast_cancer
```

In this example I am using a very basic method, thus will not go into more details, in-actual we may need to import various other packages, which will be used for Data-Cleaning; Data Visualization etc.

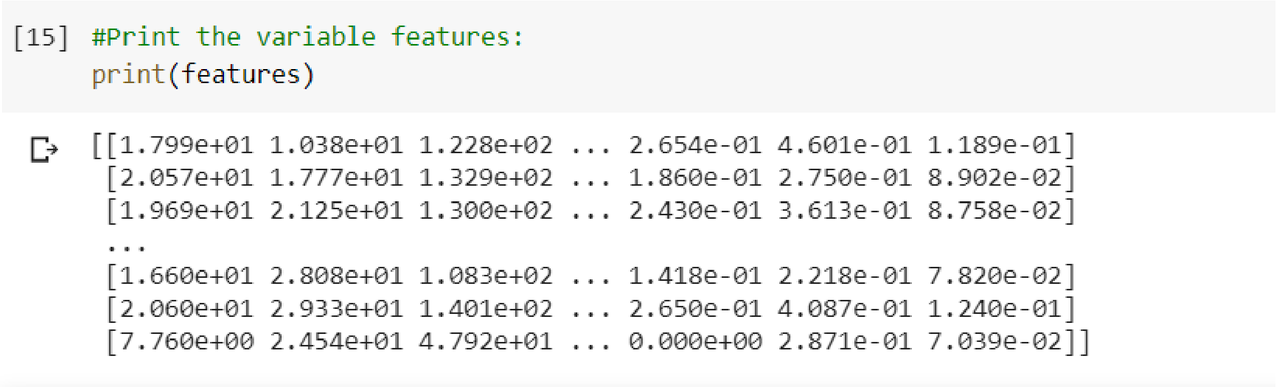
Sklearn has many other datasets which we can reference from scikit learn website as https://scikit-learn.org/stable/datasets/index.html.

Next we will load the data-set into two variables, say features, and target. Also sklearn will provide a parameter return_X_y which we need to set as True, so that we can have X which are features from the data-set, and y which is target from the data-set will be retrieved and captured in respective variables.



```
# Load the dataset from the load_breast_cancer function with the
# return_X_y=True parameter to return the features and response variable only:
features, target = load_breast_cancer(return_X_y=True)
```

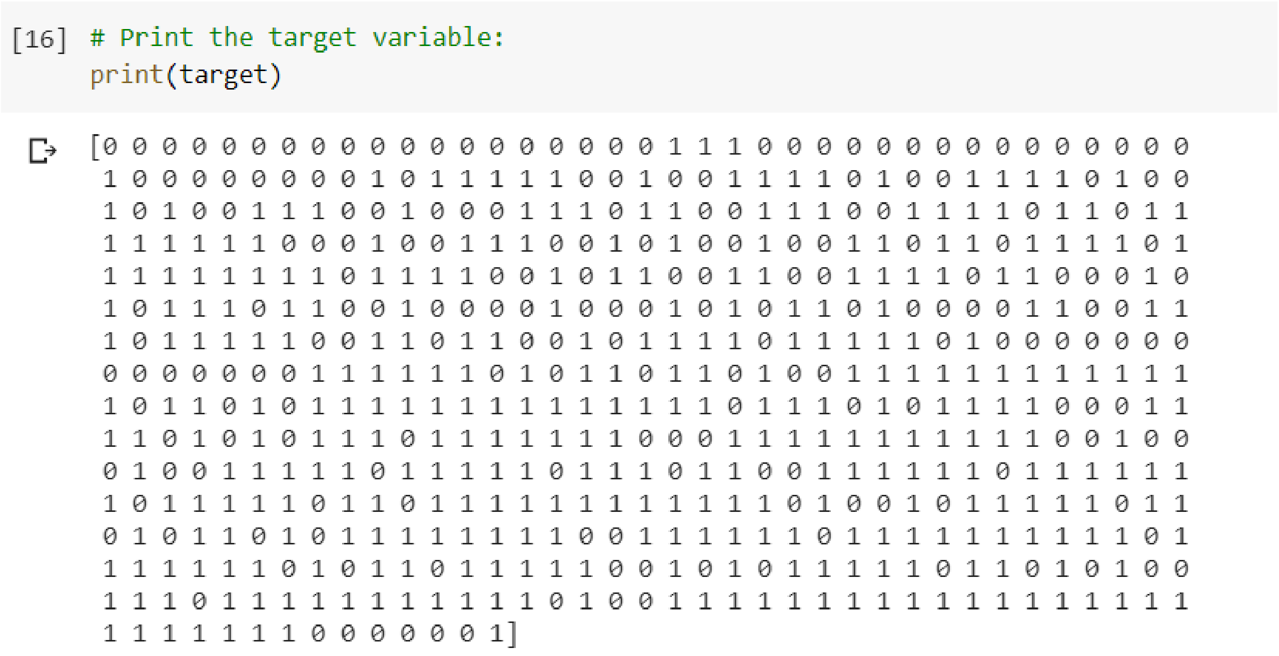
Now will see what values we do have in our feature variable.



```
[15] #Print the variable features:
     print(features)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

You should get the output similar to as shown in the above screenshot.
Similar to above, will see what values we do have in our target variable.

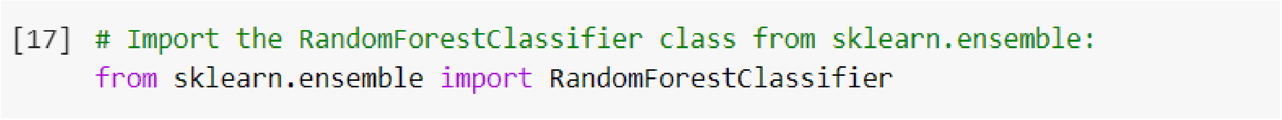


```
[16] # Print the target variable:
     print(target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

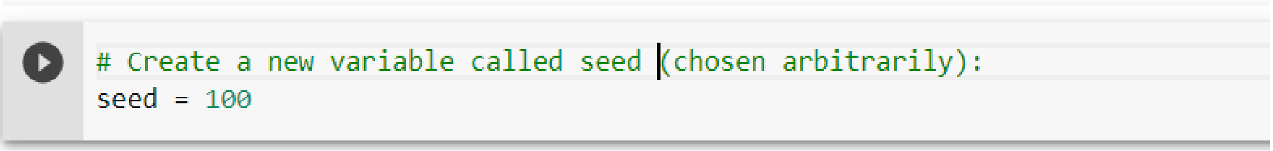
The above screenshot shows output of the target variable. There are two classes shown for each instance in the dataset. These classes are 0 and 1, representing whether the cancer is malignant or benign respectively.

Next will import the machine learning classifier. Generally it’s recommended to have all imports at the start of the program or coding, but you are free to import whenever it is required.



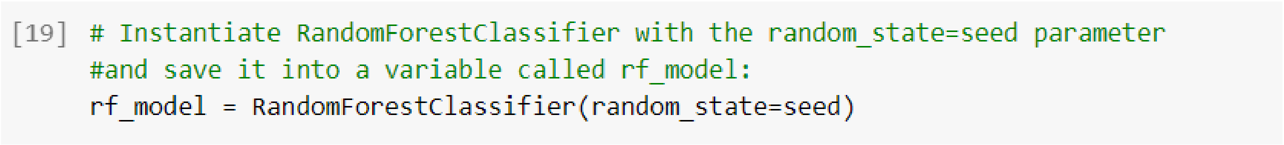
```
[17] # Import the RandomForestClassifier class from sklearn.ensemble:
     from sklearn.ensemble import RandomForestClassifier
```

Can take any random value. Will see later what impact it has. There are n number of parameters in each model, and each has its own significance, for details you can refer to the documentation https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier. for now I am using random_state and will set the seed value to it.



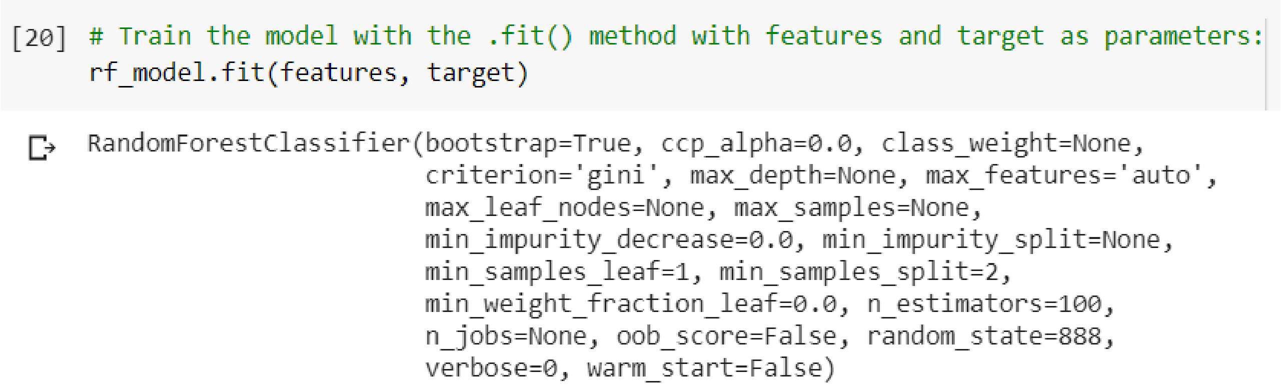
```
# Create a new variable called seed (chosen arbitrarily):
seed = 100
```

Instantiate Random forest Classifier with the above defined see value. I personally prefer to have a variable name which has some meaning, thus instantiating the model and assigning it to variable rf_model.



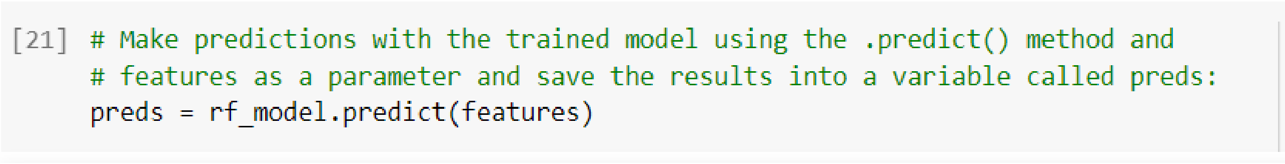
```
[19] # Instantiate RandomForestClassifier with the random_state=seed parameter
     #and save it into a variable called rf_model:
     rf_model = RandomForestClassifier(random_state=seed)
```

Now it's time to train the model using .fit() method.



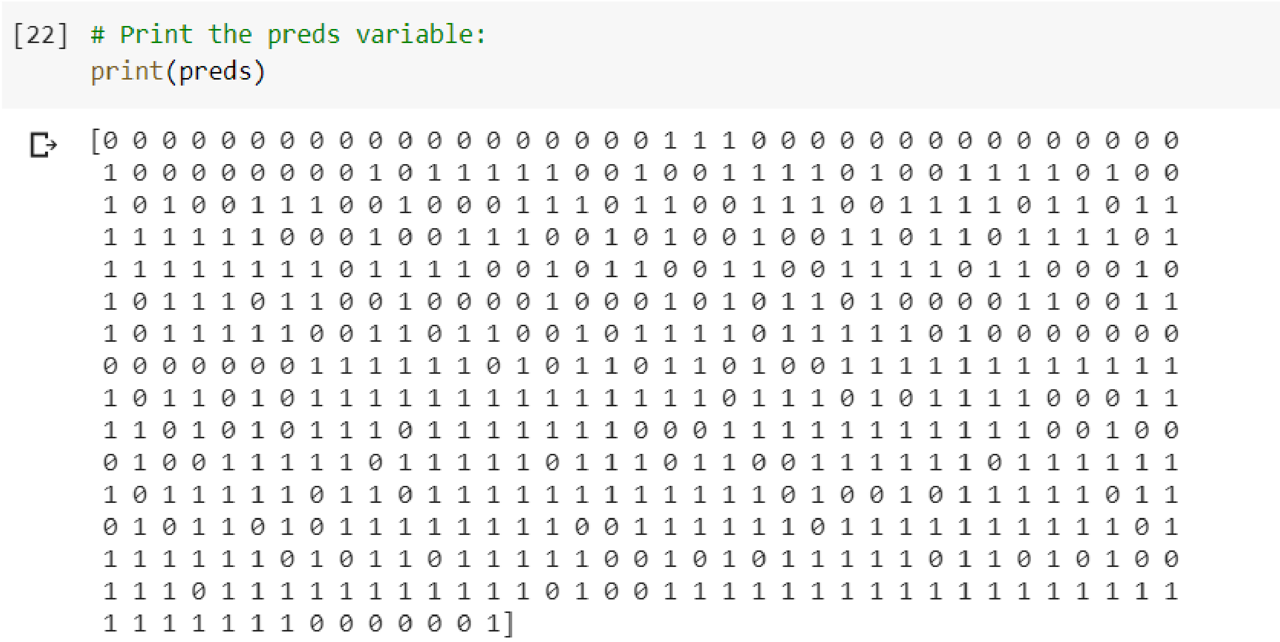
```
[20] # Train the model with the .fit() method with features and target as parameters:
     rf_model.fit(features, target)

     RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                            criterion='gini', max_depth=None, max_features='auto',
                            max_leaf_nodes=None, max_samples=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, n_estimators=100,
                            n_jobs=None, oob_score=False, random_state=888,
                            verbose=0, warm_start=False)
```

We are all set.
We have trained our model based on the data we had.
Now it's time to predict the data, for this will use the same features which we already have.



```
[21] # Make predictions with the trained model using the .predict() method and
     # features as a parameter and save the results into a variable called preds:
     preds = rf_model.predict(features)
```

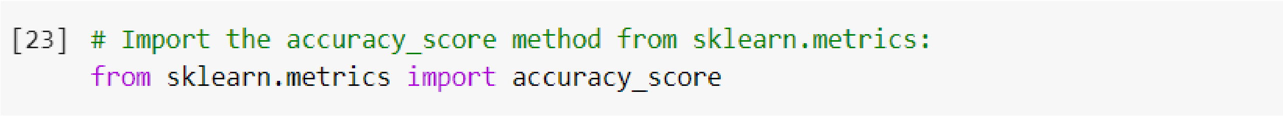
Checking the predicted values.



```
[22] # Print the preds variable:
     print(preds)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

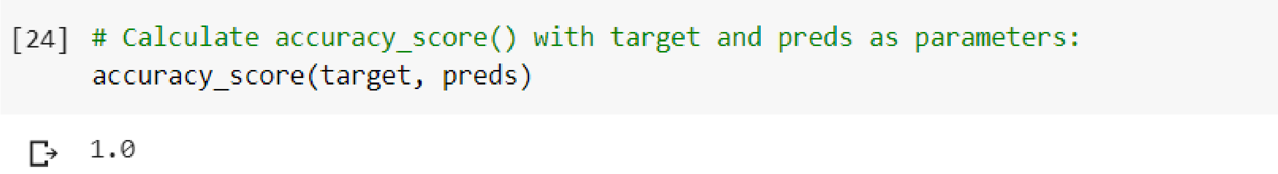
Once we have trained and predicted the values, we have to check how accurate our model is, for this will be using accuracy_score() method which is also provided by sklearn. For this we need to import it. There are various other metrics available to check the score and various other methods. The main goal is to check how accurate our model is.



```
[23] # Import the accuracy_score method from sklearn.metrics:
     from sklearn.metrics import accuracy_score
```

Here you go.. Our model says it's 100% accurate as we used the same data (features) on which we have trained our model.

Score lies between 0 to 1. Where 0 represents 0% very bad, and 1 represents 100% very very accurate, which is ideal.



```
[24] # Calculate accuracy_score() with target and preds as parameters:
     accuracy_score(target, preds)

     1.0
```

Excellent! Congrats, you just have trained a Random Forest model using sklearn and achieved an accuracy score of 1 in classifying breast cancer observations.

So in this simple way one can train a model, and with more and more effort into it, will make it a more robust and perfect model.

Hope you did get the basic idea of how to and what to do in Machine Learning / Data Science.

See you in the next article.

Code can be referenced from https://colab.research.google.com/drive/10Lq5YSmRglGQ-yNMBKMW3kGD9CcX3PB6.

About the Data Science Foundation

The Data Science Foundation is a professional body representing the interests of the Data Science Industry. Its membership consists of suppliers who offer a range of big data analytical and technical services and companies and individuals with an interest in the commercial advantages that can be gained from big data. The organisation aims to raise the profile of this developing industry, to educate people about the benefits of knowledge based decision making and to encourage firms to start using big data techniques.

Contact Data Science Foundation

Email:admin@datascience.foundation
Telephone: 0161 926 3641
Atlantic Business Centre
Atlantic Street
Altrincham
WA14 5NQ
web: www.datascience.foundation