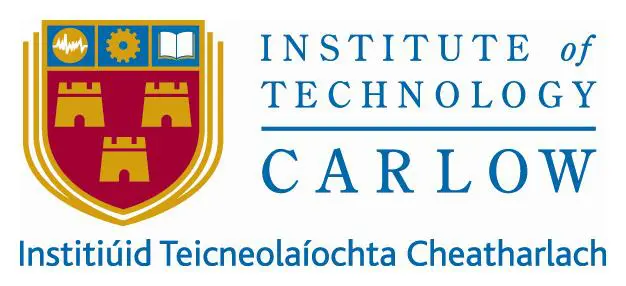
**Tutorial**



**Data Science Tutorial Project**

Name: Daniel Polak

Date: 23/04/2021

Supervisor: Greg Doyle

[**Introduction**](#_mvgkt4suwk7o) **1**

[**Install Instructions**](#_mvue233q0un7) **2**

[**Polynomial Regression Tutorial**](#_wkc5y6kstge4) **3**

[Introduction](#_hfg8g0s78197) 3

[Resources](#_abwaaeq00568) 3

[Tutorial](#_rdwsktnknt3v) 3

[Conclusion](#_j4btyk5vepbd) 9

[**Decision Tree**](#_qg7d6cgxkisz) **10**

[Introduction](#_bf0gl4u6dyln) 10

[Resources](#_3zu38pelbflm) 10

[Tutorial](#_5nee65k6uyks) 10

[Conclusion](#_2i94rn9q14ip) 14

[**K Means Clustering**](#_7pucr3waa6lx) **15**

[Introduction](#_chl3sgkzr97y) 15

[Resources](#_ke8h7u7y118z) 15

[Tutorial](#_t06ladbi8rm4) 15

[Conclusion](#_69l4n1pgp5j6) 18

[**Neural Networks**](#_r3flr7ki8bto) **19**

[Introduction](#_nw6o1upex0bu) 19

[Resources](#_aca4jqm0497m) 19

[Tutorial](#_s8kcqbtyw7ml) 19

[Conclusion](#_ygu3yq5lkell) 22

[**Conclusion**](#_rdffab1xjey0) **23**

[**References**](#_hx13jar4mjd) **24**

# **Introduction**

This document contains four tutorials which detail how some machine learning algorithms work. These algorithms are Polynomial Regression, Decision Trees, Clustering and Neural Networks. Each tutorial consists of resources that are helpful in learning about each algorithm in more depth as well as a dataset that is used for that tutorial. All of the following tutorials are done in jupyter notebook, however with a few changes you can make it run as a normal python file.

# **Install Instructions**

All necessary modules that need to be installed on your machine:

**Matplotlib**

**Sklearn**

**Pandas**

**Numpy**

You can install these modules by inputting this command in cmd/terminal

Pip install ‘name of module’

# **Polynomial Regression Tutorial**

## **Introduction**

In this tutorial I will show an example of Polynomial Regression using python and jupyter notebook.

**What is Polynomial Regression?** It is a form of regression analysis in which the relationship between variables x and y is modelled as nth degree polynomial. It fits the non-linear relationship between the value of x and y. While it fits non-linear model to the data, it is considered linear in the sense that the regression function E(*y* | *x*) is linear in the unknown parameters that are estimated from the data. Because of this, it is considered a special case of multiple linear regression [1].

**Why use polynomial regression?** It is best used when linear regression doesn’t capture patterns in data. This is called under-fitting.

In this tutorial, you will learn how to apply Polynomial Regression Algorithm to the data, process data. At the end I will show you how this algorithm can be changed into Linear Regression by changing two lines of code.

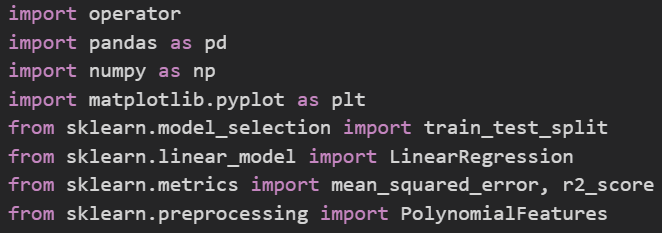
## **Resources**

Data for this tutorial: [Kaggle.com](https://www.kaggle.com/iqbalrony/polynomial-regression)

[Understanding Polynomial Regression!!!](https://medium.com/analytics-vidhya/understanding-polynomial-regression-5ac25b970e18)

## **Tutorial**

First we will need these import at the top of the file:



Pandas is used to read in our .csv file which contains data.

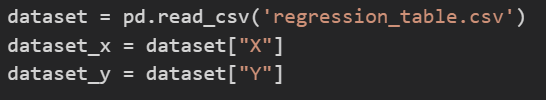
Numpy library will allow us to reshape our data.

Matplotlib is a library that will allow us to plot diagrams.

Sklearn library contains various Machine Learning algorithms which we will be using.

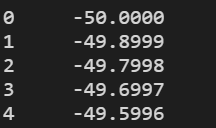
Except for the operator library, which is part of python, you will have to install all the libraries (Pandas, Numpy, Matplotlib, Sklearn) by using the pip command e.g. *pip install pandas*.

Next, we will read in our data from .csv file (link to the file in the resources section) and split it into two arrays, each will contain one column from the file:

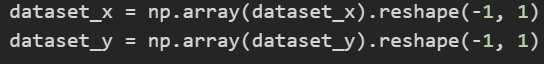


Be sure to put the .csv file into the same folder as your python file, otherwise the program will not be able to locate it. You can also change the path to the file instead.

Data will look like this:

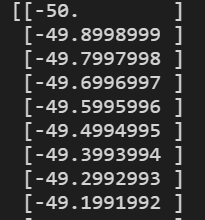


Now that we split the data into two columns, we will reshape them to form a 2D array:



We have to reshape our data because sklearn is not able to decide whether data that we passed is one row of data with multiple columns or multiple sets of data with one column. First, we will create an array from the columns by using np.array() method. Now we can reshape the array from 1D to 2D which will allow sklearn to use the data. We provide reshape with two arguments -1 and 1. -1 means that there is an unknown amount of rows and we want numpy to figure that amount out and 1 means that there is only one column.

Now data looks like this:

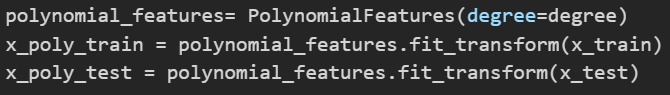


Now we can split each data set into training and testing sets:



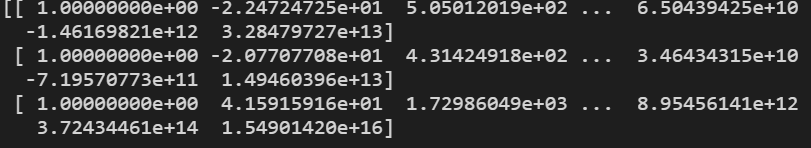
Test\_size means how much data we want to be put into testing sets. In the above you can see that in this case it is 0.33, this means that the split between training and testing sets is 67% and 33% or 670 and 330 data points.

Now we fit the data:



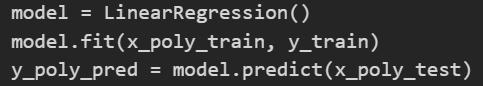
Here we supply degree n to polynomial features which best describes our data. We also fit and transform x\_train and x\_test.

It transforms data like so:



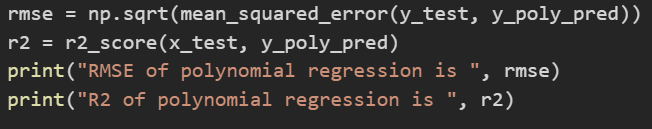
To learn more please visit <https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe>

Here we use training sets and fit them into linear regression:

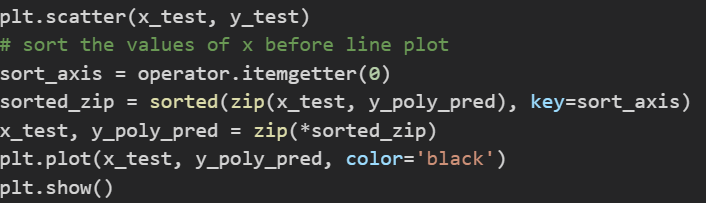


After we fit the training sets we use the x\_poly\_test to make a prediction.

Then we calculate R2(coefficient of determination) and RMSE (Root Mean Square Error):



Now we can finally plot the diagram and show the polynomial regression:



First we use matplotlib to scatter the test data on the diagram.

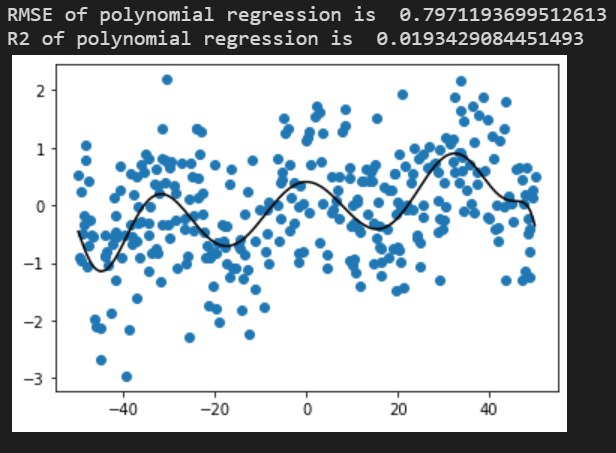
We then sort the data of x\_test and y\_poly\_pred.

Afterwards we plot the line and show the diagram.

We can run this in a jupyter notebook by typing polynomial\_regression(10).

This is due to us making it as a method in which you can replace the 10 for any number to see the effect of degree on the line.

Here is the final result:



Full Code:

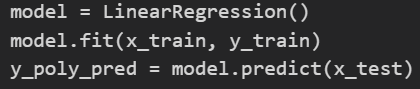


## **Conclusion**

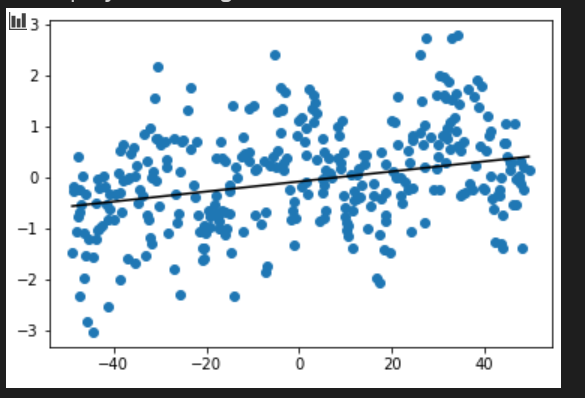
This concludes the tutorial to showcase polynomial regression. In this tutorial you hopefully learned how to read data from a file, separate the data, reshape the data and split the date into test and training sets. Also how to fit the data and predict the values by using training data.

To see different behavior of polynomial regression, you can use a dataset of different size.

By making a few changes to the code you can make this into linear regression.



Which will result in:



# **Decision Tree**

## **Introduction**

In this tutorial I will show you how to implement a decision tree in Python and Jupyter Notebook.

**What is a Decision Tree?** Decision tree(graph) is a structure which uses a tree-like model of decisions and their possible consequences [2]. It is used in machine learning, as it is one of the predictive modeling approaches. This algorithm uses a decision tree(graph) to go from observation to conclusion of an item/variables. The goal is to predict a value of a variable based on several input variables [3].

**Why use Decision Trees?** It is best used when you have more than two columns of data and you want to make a prediction on one column using the data in the rest of the dataset.

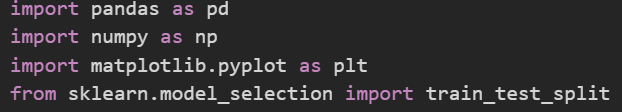
## **Resources**

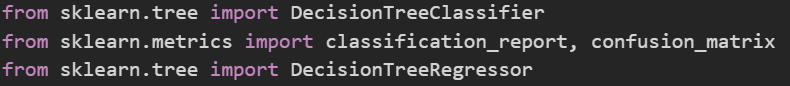
Data: [Kaggle.com](https://www.kaggle.com/raghupalem/bill_authentication)

[A Guide to Decision Trees for Machine Learning and Data Science](https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956)

## **Tutorial**

For this tutorial we will need following imports:





At this point you should have sklearn extension installed if you have followed from the beginning of this document. If you have not followed from the beginning please go to the installation guide at the start of the document.

Now, we will read in our data from the .csv file(link to the file in the resource section).



We split the dataset into two variables x and y. Variable x will contain data in all columns except one which will be assigned to variable y. This is done by using the drop() function which takes the name of the column and which axis that column can be found at.



As in the previous tutorial, we will split our data into training and testing sets. The reason for splitting the same dataset for training and testing is to ensure that the decision tree can derive rules that it can later use to predict the outcome. While not impossible to predict values when training the algorithm, it is more precise when used with the same dataset.



train\_test\_split () function takes three arguments in this case, X which is split into X\_train and X\_test, Y which is split into Y\_train and Y\_test. Test\_size specifies how much of the dataset is put into training sets, in this case 0.2(20%) of the dataset.

Now, we can fit our data



We supply the training sets to fit() function which will train the classifier so that we can make predictions.

We can now make a prediction by supplying the classifier with X\_test dataset



At this point we will print confusion matrix which is used to describe performance of the classifier.

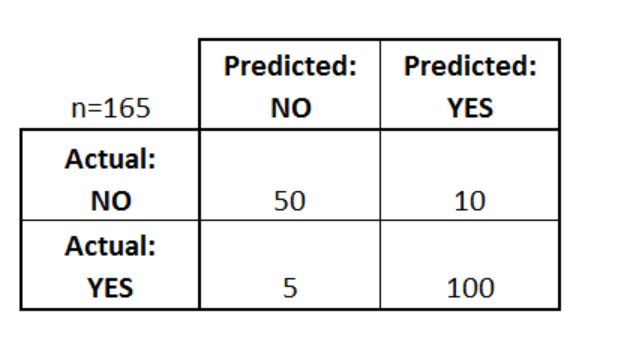
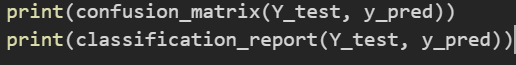
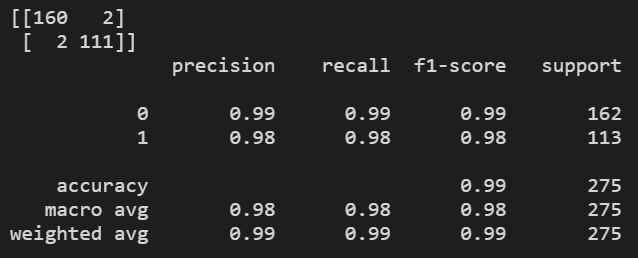


Fig. 1: Confusion Matrix Example [4]



We will also print classification report which will give us some values such as precision, recall, F1 and support score for the model.

It will look like this:

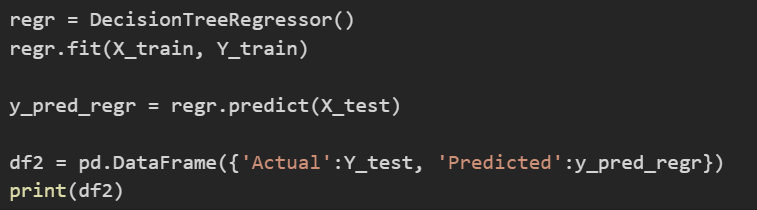


Visualisation of confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | Predicted: 0 | Predicted: 1 |
| Actual: 0 | 160 | 2 |
| Actual: 1 | 2 | 111 |

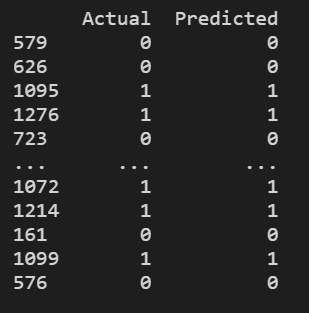
As you can see, in this case both Actual values and predicted values are the same, which means that our model was 100% accurate in its prediction.

This can change anytime you run the algorithm.

Now, we will try a regressor instead of a classifier. The difference between classifier and regressor is that classifier is for predicting a label while regressor is for predicting quantity.

We have used the exact same dataset used for the classifier. Based on the highlighted information above. Is this the right way to use a regressor?

Output:



## 

## **Conclusion**

This concludes the tutorial section on Decision Trees. Hopefully, you have learned more about them in the process as well as a way to implement them in python.

If you want to learn more about Decision Trees, please have a look at the Resources section of this tutorial section.

# **K Means Clustering**

## **Introduction**

In this tutorial I will show you how to implement a K Means Clustering in Python and Jupyter Notebook.

**What is a K Means Clustering?** It is a type of unsupervised learning which is used when you have data which is not defined by categories or groups. The aim of this algorithm is to partition n observations into k clusters with the nearest center(mean) [5].

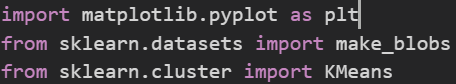
**Why use K Means Clustering?** It is the easiest clustering algorithm to implement as well as run as it is not memory intensive. In many instances this algorithm is used as pre-clustering for other, more intensive algorithms. However, many people only need the simplest solution as they don’t need more advanced clustering algorithms.

## **Resources**

[Understanding K-Means Clustering in Machine Learning](https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1)

## **Tutorial**

These are the modules that we will be importing for this tutorial:



In this tutorial, we will not be reading in a dataset from a file, instead we will use a built in function of sklearn to create a dataset that will be used for this algorithm.



N\_samples is for the number of data points in a column.

N\_features is the number of columns.

Centers is the number of centers around which the data will gather.

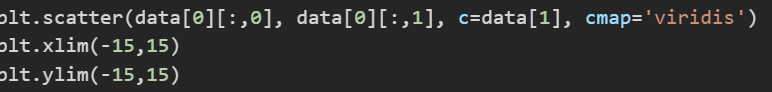
Clusters\_std is a standard deviation of clusters.

Random\_state determines a random number generation for a dataset.

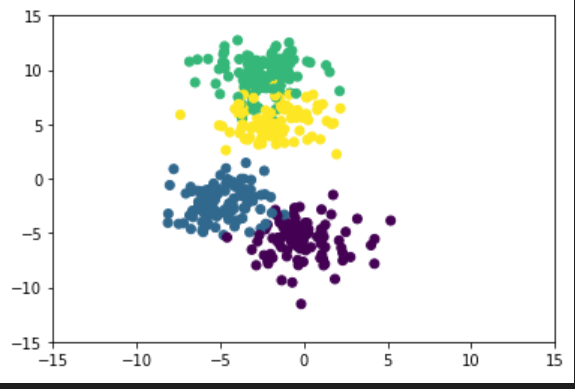
Now, we will assign the first column of the data to variable points.



We will visualize the data that is in the dataset:



Which will have output as:



We will use a built in function of sklearn to generate K-means object:



N\_clusters is to declare how many clusters we want and how many centers we want as well. Default value is 8.

We will now fit the data into those 4 clusters(kmeans object):

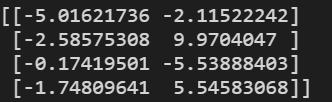


Now, we will find out where our centers are so that we know around where our data is being concentrated.



This prints out a location of clusters learned by fitting the data into kmeans object.

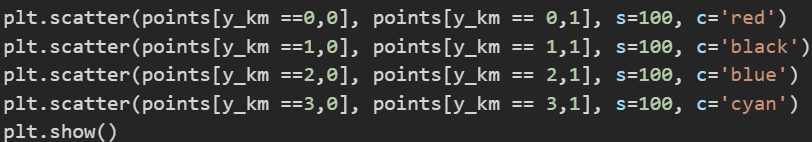
This gives us our four centers on x and y axis:



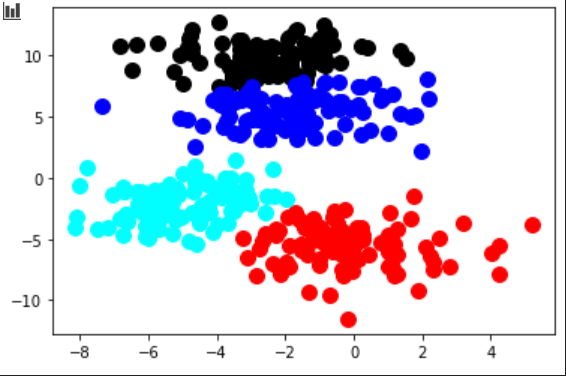
We will now make a prediction using the points therefore creating a new clusters:



Now all that is left to do is to plot the new clusters:



Output of this is



This gives us a better visualisation of clusters compared to the previous diagram.

## **Conclusion**

This concludes the tutorial about K Means Clusters. Hopefully, you have learned something new today about clustering algorithms. You should check the resource section to read more about K Means Clustering. You can play around with the algorithm by changing the number of clusters that are produced (don’t forget to add or delete plt.scatter() function at the bottom as even if increase the number of clusters they will not show on diagram) as well as number of centers (don’t forget to change the number when creating KMeans object).

# **Neural Networks**

## **Introduction**

In this tutorial I will show you how to implement a simple neural network in Python and Jupyter Notebook.

**What is a Neural Network?** It is a series of algorithms that tries to recognize relationships in a set of data.This is done through a process that tries to mimic the way the human brain works. It is based on a collection of connected nodes which loosely model the neurons in the human brain [6].

**Why use a Neural Network?** It is used to find hidden patterns in a set of data, cluster and classify the data. Over time it can learn and improve the process. Today, many business problems are solved by using neural networks. It is also used in self driving cars.

## **Resources**

Data:

[Kaggle.com](https://www.kaggle.com/arshid/iris-flower-dataset)

Read:

[Neural Networks for Beginners: Popular Types and Applications](https://blog.statsbot.co/neural-networks-for-beginners-d99f2235efca)

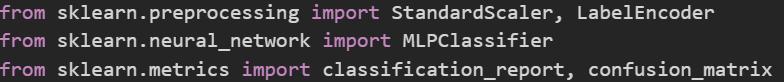
[Neural Networks from Scratch with Python Code and Math in Detail— I](https://pub.towardsai.net/building-neural-networks-from-scratch-with-python-code-and-math-in-detail-i-536fae5d7bbf)

Youtube Videos:

1. [What is a Neural Network?|Deep Learning, Chapter 1](https://www.youtube.com/watch?v=aircAruvnKk&ab_channel=3Blue1Brown)
2. [Neural Network In 5 Minutes | What Is A Neural Network? | How Neural Networks Work | Simplilearn](https://www.youtube.com/watch?v=bfmFfD2RIcg&ab_channel=Simplilearn)

## Tutorial

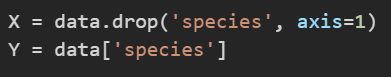
For this tutorial we will need the following imports:



We will now read in our data from .csv file which can be found at resource section of this tutorial:



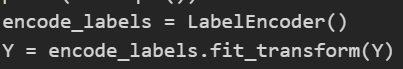
We will proceed to split the data into two. One will contain all the columns except the last one called ‘species’ and the other will contain ‘species’ column. We specify which column to drop when assigning the data to a variable X and on which axis(row) does this particular label reside, in most cases that would be axis 1 or first row. As we are assigning data to variable Y, we just need to specify which column the data resides in.



Our data at Y contains following unique values:



As you can see they are categorical data, however neural networks work best with numerical data. Therefore we will convert them from categorical to numerical by using sklearn LabelEncoder.



This will give us a following unique values:

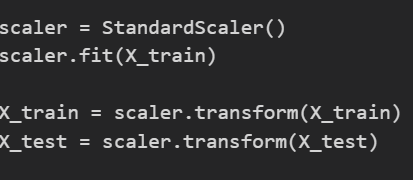


Now that our data is converted, we can split it into testing and training sets.



Training data will contain 80% of the overall data to give the algorithm a better chance to predict an output.

Before we get to prediction, we will feature scale our data so that they can be uniformly evaluated. We do this only to the training sets as data in the real world is not scaled and ultimately the neural network is to make predictions on real world data.



At this point we can finally train a neural network using a fit() function and make a prediction using predict() function.



Hidden\_layer\_sizes: what we have done here is created 3 hidden layers and each layer consists of 10 nodes.

Max\_iter: specifies number of maximum iterations.

.ravel() makes a multidimensional array into a flattened array.

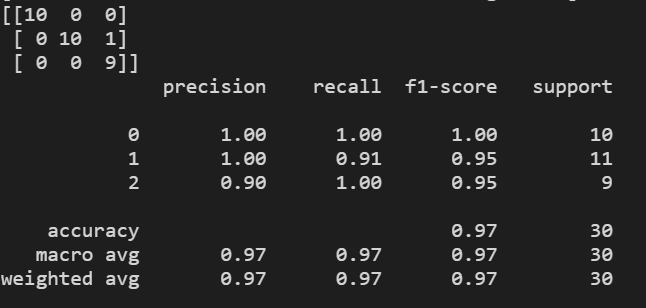
Now we can make a prediction, as our neural network is trained, using the test data.



Now is time to evaluate how the algorithm performed. This is done by using confusion matrix and classification report functions.



This will generate the following output:



Our prediction has misclassified one out of 30 tested. This can change as test\_train\_split splits the dataset randomly which means that there will not be the same result if you run it multiple times. If you have been following along and your results are different that is the reason. Accuracy should in all cases be greater than 90%.

## **Conclusion**

This concludes the tutorial about Neural Networks. Hopefully you have learned something new. I would recommend playing around with the numbers to see how it behaves and if you can achieve better results. You can easily change the algorithm from classifier to regressor by changing MLPClassifier() to MLPRegressor() function and instead of confusion matrix and classification report, you will have to calculate RMSE and R2 values. I would also recommend following a tutorial that will show you how to build neural networks from scratch without using sklearn (included in resource section).

# **Conclusion**

This is the end of the document that focused on showcasing some machine learning algorithms.

This tutorial can be found at Github, which contains all of the datasets and code.

Github Link: <https://github.com/DanielP1308/Data-Science-Tutorial>

# **References**

[1] 25th December 2020, “Polynomial Regression”, “Wikipedia.com”, <https://en.wikipedia.org/wiki/Polynomial_regression>, (Accessed on 3rd March 2021)

[2] 20th March 2021, “Decision Tree”, “Wikipedia.com”, <https://en.wikipedia.org/wiki/Decision_tree>, (Accessed on 28th March 2021)

[3] 20th February 2021, “Decision Tree Learning”, “Wikipedia.com”, <https://en.wikipedia.org/wiki/Decision_tree_learning>, (Accessed on 28th March 2021)

[4] 25th March 2014, “Simple Guide to Confusion Matrix Terminology”, dataschool.io, <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>, (Accessed on 2nd April 2021)

[5] 2nd February 2021, “K-Means Clustering”, Wikipedia.com, <https://en.wikipedia.org/wiki/K-means_clustering>, (Accessed on 5th April 2021)

