# 

# Chart Description automatically generated

Data science programming

Final report

Sara Aljamal

Section 1

Contents

[1](#_Toc126198872)

[Technical Documentation 3](#_Toc126198873)

[1 Introduction 3](#_Toc126198874)

[1.1 Types of data structures 3](#_Toc126198875)

[1.2 Common libraries 5](#_Toc126198876)

[1.3 Plotting and visualization libraries 6](#_Toc126198877)

[2 Experiments 6](#_Toc126198878)

[2.1 Programming languages and tool 6](#_Toc126198879)

[2.2 Load Data and Prepare Data (Pre-processing) 7](#_Toc126198880)

[2.3 Approaches 10](#_Toc126198881)

[3 Results 11](#_Toc126198882)

[3.1 Compare the different models 11](#_Toc126198883)

[3.2 Charts 11](#_Toc126198884)

[3.3 Analysis of the results 12](#_Toc126198885)

[4 Evaluation 13](#_Toc126198886)

[4.1 The choice of data structures 13](#_Toc126198887)

[4.2 Selection of the appropriate libraries 14](#_Toc126198888)

[4.3 The effectiveness of different models 14](#_Toc126198889)

[4.4 Recommendations 15](#_Toc126198890)

[5 References 16](#_Toc126198891)

# Technical Documentation

# Introduction

## Types of data structures

Data structures are tools that helps in organizing, controlling, and storing data, this allows us to access the data and use it efficiently. In python we have several data structures divided to two main categories which are user defined data structures, and built-in structures. This part will only focus on built- in data structures which are split into two major groups, mutable, which means that we can modify on its content, and immutable, which means that we cannot change the content after we create the data structure. Having a lot of options for organizing and managing data in python indicates and explains why python may be considered as the main language used in data science, AI, and machine learning. When talking about data structures in python, we can’t say that we have one data structure that is better than others, all of them are good and each one of them can be used in an efficient way, so choosing the right data structure to use depends on the case and the operations we want to make on the data.

1. **Lists:**

* **Definition:**

**A picture containing chart

Description automatically generated**A list is a simple and useful data structure. A list content data be a single data type such as an int, float, string, or boolean, or multiple data types in one list. As other data structures, list indices start with 0, and we can control the length of the it as we want. We can understand that we are dealing with a list from these brackets **[]**. In addition, we can transform other data structures to a list using the command list ().

* **Defining a list in python:**
* **Operations we can do on lists:**

1. We can add elements on a defined list using:

* **Append**: which adds the element as the last element in a list.
* **Insert**: Which allows us to add an element in a specific location.
* **Extend**: Which works like append but with several elements.

1. We can remove an element using **pop.**
2. To check if an element exists in a list we can use **in** and **not in** functions.
3. We can combine and concatenate two lists using (+).
4. We can sort lists according to a specific key like sorting a string list according to the length of each element.
5. Also we can slice the list and take only part of it using slicing method.
6. **Tuples:**
   * **Definition:**

A tuple is the only immutable data structure in python, this means the assigned values can’t be changed. So, if want to set unchangeable values, using tuples would be a good choice. Also, tuples can take any data type, for example, int, float, and a lot more. We can identify a tuple by these brackets **()**. We can convert other data **A picture containing text

Description automatically generated**structure to a tuple using the command tuple ().

* + **Defining a tuple in python:**
  + **Operations we can do on tuples:**

1. We can access tuple elements using **[]**.
2. We can concat tuples together.
3. We can multiply a tuple by an integer, so every element will be multiplied by the same value of the integer.
4. We can count the number of an element in a tuple using **count** function.
5. If we want to sort a tuple, we can use **sorted** function.
6. Although we can’t change the values as it will show a type error message, but we may extract tuple values using unpacking.
7. **Dictionary:**

* **Definition:**

A dictionary is like a map, it consists of key – value pairs. Every pair contains a key, and a value for that key, this value can be anything, maybe a list. And the key of a dictionary may be a string, float, int, or boolean. A dictionary is a way of defining related things in a clear and clever way. We can identify dictionaries using **{}**.

* **Logo

  Description automatically generated with low confidenceDefining a dictionary in python:**

****

* **Operations we can do on dictionaries:**

1. We can access a dictionary values by passing the key in the [].
2. If we want to access the keys of a dictionary we may use **.keys** function, if we want the values then we use **.values**, and if we want both we can use **.items**.
3. To check if a key exists, we can use **in** and **not in**, also we can use them to check the values but with **.values**.
4. We can delete a key using **del** or **pop**.
5. We can add an item to the dictionary by passing the name of the new key after the dictionary name = its value.
6. **SETS:**

* **Definition:**

The special thing about sets is that it only contains unique values. A set is a mutable and unordered data structure. It can take several data types as elements but it can’t contain other data structures such as a list or dictionary, but we can include their elements in the set or change a list to a set using **set ()**. In sets, we cannot access elements using indexing or slicing as it’s an unordered data structure. We can identify a set by **{}**.

* **Defining a set in python:**
* **Operations we can do on sets:**

1. To add elements to a set, we can use add to **add** one element, and **update** to add several elements.
2. We use **clear** to clear the whole set or **remove** to remove one element.
3. There are some operations we can do on two sets, such as union (A | B), intersection (A & B), difference (A- B), and symmetric difference (A ^ B).

## Common libraries

As mentioned before, python is a language that is used in data science, that is because it contains libraries that helps data scientist in dealing with data, understand, and analyse it. Importing a library means getting access to use the pre-defined functions this library contains, functions benefit us a lot as programmers, as it makes things easier; libraries allow us to do operations to our code and to our data using these functions. Examples for common libraries in python are:

1. **NumPy:**

Which stands for Numerical python, is a library that focuses on multidimensional arrays. In addition, it includes some functions related to some math sections such as matrices, linear algebra, and statistics (Mean, Standard deviation, min, max, and a lot more). NumPy provides us with a collection of functions that we can use with arrays, these functions allow use to organize and explore our data, for example, **where** function which can be considered as a strong tool for selecting parts of data and modifying on them according to some conditions.

1. **Pandas:**

It is a library that is designed to provide us with various data structures and tools for data manipulation in order to understand, study, and prepare data. Pandas includes a variety of functions for cleaning and exploring data, which means that pandas library functions can be used during the pre-processing stage. Furthermore, pandas works with two main data structures: series, which is similar to a one-dimensional array but with a sequence, and data frames, which is similar to a table.

1. **Scikit learn:**

It’s a library that was built on other libraries such as NumPy, SciPy, and matplotlib. Scikit learn can be considered as a machine learning library as it has important functions that we use after the pre-processing stage to do the final preparations for the data before entering the machine learning model, these functions are label encoder which convert categorical values to numeric, and the function used to split the data to test and train. Also, it has the functions that we import to use the machine learning models, for example, the KNN, decision tree classifiers. In addition, this library is used to import the evaluation measures for the machine learning models such as the accuracy and R^2. So basically, scikit learn is the library we use to do everything related to the machine learning after the pre- processing stage and before the visualization stage.

## Plotting and visualization libraries

Visualization is considered the last part of using a machine learning model, it’s the stage where we present the results from the machine learning model or models, transform them to understandable form by normal people, and being able to compare between the models if we used more than one model. To achieve all these things, python has a popular and widely used plotting library which is matplotlib.

Matplotlib is a simple but strong library which contains functions that we use to visualize the results by different visual aids, for example, bar charts, box plots, histograms, and a lot more.

Each one of the four libraries explained above is important in a stage or multiple stages in machine learning, all these libraries works together to help us in producing the perfect machine learning model. Without these libraries everything would be harder, this indicates the importance of having such libraries in python and why python can be the appropriate language to use for data science and machine learning.

# Experiments

## Programming languages and tool

In this project scenario we were required to proof our abilities in data science by generating different machine learning models from A to Z. To do that, I had to use a programming language of course, and the most appropriate one was python. Python is powerful, popular and widely used programming language created in 1991. Since it’s a flexible, open source, and easy to write and learn language. Python is used in variety fields and tasks including data analytics, AI, and machine learning. As mentioned earlier in this report, python has may libraries and tools that makes things easier. These tools include anaconda, Jupyter notebook, and google colaboratory which was the tool I used in this project. Colab is a cloud-based service by google that allows us to write, and run programs, also it’s integrated with google drive which gives us storage and the chance to share our programs. Colab doesn’t require any configuration, and it gives us access to GPUs. Colab provides each user with a free12 GB RAM, to increase the amount provided RAM we can pay for the amount of RAM we want according to some subscriptions defined by google colab.

Logo, company name

Description automatically generated

## Load Data and Prepare Data (Pre-processing)

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Column/step Name** | **Description** | **Justification** |
|  | Importing drive from google colab library | Connecting my google drive account to google colab, after uploading the dataset to my drive. | To have permanent access to the dataset instead of uploading it every time I want to work on the project. |
|  | Importing important libraries | Such as NumPy, pandas, and random | These libraries are the main libraries I used in the pre- processing, as every library helped in a specific way through its functions. |
|  | Creating the data frame | Transforming the data set to a readable form in python which is a data frame | In order to clean and deal with data in python, we must transform it to an accepted form in python. |
|  | Viewing the dimensions of the data frame | Using **.shape** function | To know the size of the data frame we’re working on. |
|  | Checking the number of null values | I checked the number of null values each column has using **isnull (). sum ()** function | This step is one of the most important steps to start the pre-processing with, as we know what columns have null values we need to handle and the number of them, so can decide the right handling method. |
|  | Checking for duplicated | Using **duplicated (). sum ()** function | It’s important to remove any duplicated rows, we can check if our dataset contains any duplicated records using this function. |
|  | Dropping unwanted columns | I dropped 5 columns from the data frame:   1. Close group 2. Resolution 3. Resolution\_describtion 4. Case desc 5. Open gr | 1. Close group column contained a lot of null values (4890) that I couldn’t find a way to handle, as I couldn’t find relation between it and other columns 2. Resolution and Resolution\_describtion columns contained (10336), and (10376) null values, which means almost all values in both columns are null. 3. Case desc column had (9243) null values, which is a huge number. Also, this column was about describing if the case was solved or not, and in the dataset, we had another column called current status which described if the case is resolved or active, and it had 0 null values. 4. Open gr column had (4561) and I couldn’t find relations with other columns |
|  | Checking the data type for each column | Using **.info ()** function | To View the data type for each column |
|  | Filling null values in **OFFER NAME** column | I filled the null values in this column using **np. Where ()** function, which allowed me to fill the null values according to some condition, these conditions represented the relations I found between data in excel. | It was the best method I found, filling the null values according to some relations with other columns. |
|  | Filling null values in **CUSTOMER GROUP** column | I filled the null values in this column using **np. Where ()** function, which allowed me to fill the null values according to some condition, these conditions represented the relations I found between data in excel. | It was the best method I found, filling the null values according to some relations with other columns. |
|  | Filling null values in **ESCALATED** **GROUP** column | I used **np. Where (),** in addition to filling the null values after where function with the mode | After finding some relations, I still have some null values that are less than 100, so I filled them with the mode |
|  | Filling null values in **OPEN USER** column | For this columns I took a sample from the same column without null values, and filled the null values with random values from that sample | Thins column contained names, so I think this is the appropriate method to fill null values |
|  | Filling null values in **CLOSE USER** column | I filled some null values using np. Where (), and for the rest null values I did the same as in the open user; I filled them randomly. | Before going to randomly filling the null values, I preferred to find some relations, so it would be better. |
|  | Filling null values in **CLOSE DATE** column | Close date null values are all when the case is active, so there is no close date. | This was a good way because of course when the case is still active then there will be no close date. |
|  | Filling null values in **AGE BRACKET** column | Age bracket null values are all when the case is active, so there is no age bracket because there is no close date till now. | This was a good way because of course when the case is still active then there will be no close date and will result in having no age bracket. |
|  | Filling null values in **CALLBACK MECHANISM** column | I used a method that fills the null values with two random values form the same columns which are phone, and SMS. | The two values that were repeated the most were Phone and SMS, so I filled the null values with them. |
|  | Checking null values after handling all missing values | I checked the number of null values each column has using **isnull (). sum ()** function. | To make sure all null values were handled. |
|  | Printing the head of the data frame | Using **.head ()** function. | To see the data frame shape after the pre- processing stage |
|  | Downloading the data frame after finishing the pre- processing stage. | Using **.to\_csv()** function | This downloads a copy of the new data frame, it’s downloaded to our devices as an excel sheet, this is important to see the final result after the previous stage, so we can see the dataset that will be the input for the machine learning model. |
|  | Encoding | After the pre- processing step, but before starting with ML model, we need to transform all categorical columns to numeric using label encoder. | This is because most of the ML models don’t take categorical values as input, so transforming them is a must. |
|  | Calculating the correlation coefficient | Using the heat map from matplotlib. | This helped me a lot in viewing the relations between columns, as well as helping me decide which columns to drop. |

## Approaches

I used four different classification machine learning models to predict if the product is internet or mobile (product column)

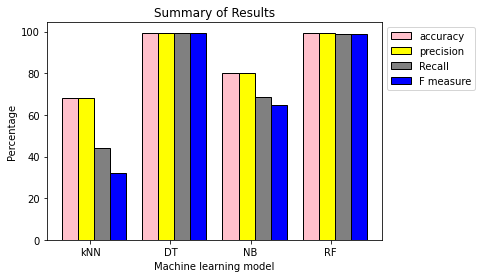
|  |  |  |
| --- | --- | --- |
| **Approach no.** | **Name** | **Description** |
|  | KNN | KNN or K Nearest Neighbors, is a supervised ML algorithm that classifies unclassified data according to the similarity between it and other classified data (based on the distance). KNN is a lazy algorithm, this means it has zero training time and it takes the time to classify data. The main parameter used in KNN is the value of K which is the number of neighbors taken to make predictions. In my project, I set the value of K=3. |
|  | Decision Tree | It can build different classification or regression models in a tree structure; this means that decision tree divides the dataset to smaller sets by the time the tree is getting bigger. A leaf node in a decision tree represents a classification. |
|  | Gaussian Naïve Bayes | It’s a probabilistic classification algorithm that uses bayes theorem (In probability) with independent assumptions. When talking about classification, independence means that a feature doesn’t affect other features. Naïve bayes classifiers are flexible, expressive, and accurate. Furthermore, naïve bayes performance may decrease when the training set grows. However, the number of features affects the classifies positively. |
|  | Random Forest | To make predictions, random forest combines multiple decision trees. That is why random forests can be used for regression ad classification. A random forest trains multiple decision trees on different sets, then takes the predictions from all decision trees and calculates the final prediction based on the largest prediction of all predictions. |

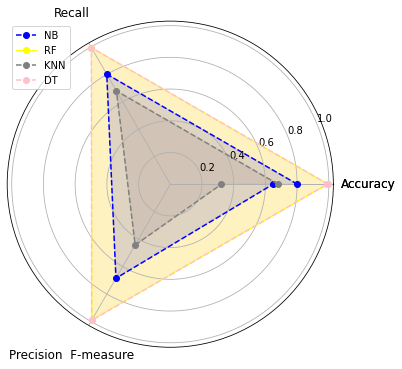
# Results

## Compare the different models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Approach no.** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **KNN** | 0.6794879255164387 | 0.4427107911192079 | 0.2541473147303551 | 0.322758516557035 |
| **Decision tree** | 0.9948631781084976 | 0.9920235427543106 | 0.990917878390151 | 0.9914642929430724 |
| **Naïve bayes** | 0.8012802048327732 | 0.6851667796523632 | 0.6161966380104704 | 0.6487056364508408 |
| **Random forest** | 0.9944311089774364 | 0.9935466229737194 | 0.9879196697055852 | 0.9907168782807109 |

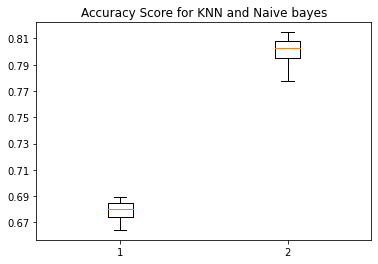
## Charts

* **Bar chart:** This chart shows the evaluation measures (Accuracy, precision, recall, and F-1 score) for the four machine learning models. Each evaluation measure value is the average for the 30 values that came from the 30 iterations for each model.
* **Radar chart:**

****This chart displays the evaluation measures for the four machine learning models, and every measure is represented in an axis.

* **Box plots:**

Chart, box and whisker chart

Description automatically generatedI have four box plots, each one of them represents the 30 accuracy values for each model.

## Analysis of the results

After building our machine learning models and trying them, now it’s the time to look at the results, understand them, and analyse them, in order to be able to explain them to other people in the company. We can analyse the results in a clear and organized manner by using the two points mentioned above, the table and the visual aids. The first thing we notice is that not all models produced the same results; in contrast, some models performed effectively and produced excellent results, while others did not. When the performance of the models was compared, the decision tree and random forest performed well and their results were very close to each other's, because, as explained previously, a random forest is a collection of decision trees. Both models' accuracy, as well as other accuracy measures, reached 99%. The accuracy of the Nave Bayes model was 80%, which was not the best. Moving on to the KNN, it was the worst of them all, even though I tried different K values and tried to improve it, it remained the same. All these things can be noticed and understood through the visual plots. For example, looking at the bar chart, we can see that decision tree and random forest models were the best of all, naive bayes was good, and KNN model bars were the shortest of all, indicating that its performance was poor. Through the radar chart, we can observe that F-1 score, and the precision looks like they are on the same axis, that is because the values of the precision and the F-1 score were close in all models. Another example of data visualization is the box plot, which has four boxes, each of which represents a model. The purpose of the boxes is to show different values of accuracy for each model in the 30 iterations. The plots show that some models, such as the KNN, differ in the accuracy score for each iteration, which is clear because the range for the entire KNN box was (0.65 - 0.69), which makes the largest difference between the value = 0.02. However, when we look at the decision tree box, we can see that the largest difference between the values was much smaller than the difference in KNN, as it was 0.007, proving that decision tree outperforms KNN because the values are stable and don't change much. We can see the value of visualization at this stage because it makes everything easier to notice and understand for the data scientist and others.

# Evaluation

## The choice of data structures

We know that python provides us with different options of dealing and storing data through its built -in data structures (List, tuple, dictionary, and set). Everyone can use the data structure they find appropriate for their program and problem, in my project, I used lists and dictionaries, as they had everything I could need in my code.

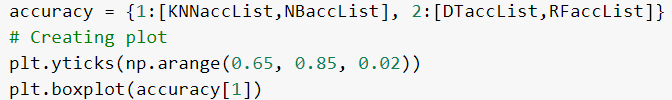
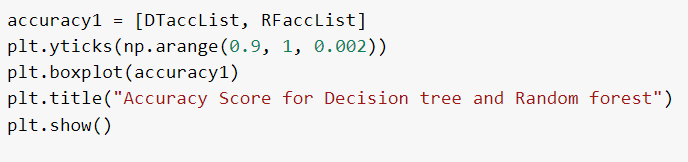
**Lists:**

As mentioned earlier, a list is a mutable data structure that we can edit, change, or remove its elements easily using some functions. Also, lists can take any data type as element. In the following points, I will clarify, explain, and evaluate through the following example of the use of lists in my code:

****I defined four lists for the accuracy of each model, every list saves the accuracy from every iteration in that model, taking naïve bayes as an example, I defined a list before the for loop of the iterations, them inside the for loop and after calculating the accuracy I used append to add the new accuracy to the list, by this, after the iteration and the for loop the list will have 30 generated values for the accuracy. I needed these lists to draw the box plot, so I used these lists to plot 4 boxes that describes the accuracy for each measure. Lists here were better that sets, for the reason that sets don’t duplicate the values, so if two iterations had similar accuracy, then we would only have one, and the set may have less than 30 values.

**Dictionaries:**

The special thing about a dictionary is that it’s the only data structure that contains pairs that consists of keys and values. In the following example, I used a dictionary that has lists as its values to separate four box plots into two graphs. Using a dictionary here was more efficient and takes less time due to the fact that we create a dictionary instead of two lists, so we write one line of code instead of two.

-Dictionary: -Lists:

## Selection of the appropriate libraries

I used various libraries in my code, including NumPy, Pandas, Scikit Learn, and matplotlib; each library added something to my code and assisted me in achieving certain goals. But if I had to pick one library to discuss, it would be Pandas.

Pandas is a library that I used frequently in my code, especially in data loading and pre-processing stages. Firstly, data loading is considered the starting point for my project. To load the dataset and transform it to an acceptable form in python, I used Pandas, without using it, this step will be hard and maybe impossible.

As I transformed the dataset to a data frame, which is a data structure provided by pandas, I found a lot of functions that helped me in the pre processing stage, while using these functions, I discovered how powerful and important Pandas is as a library. Some of these functions were:

1. . isnull (). sum ()
2. . info ()
3. . head ()
4. . fillna ()
5. . replace ()
6. . loc ()

## The effectiveness of different models

After building the machine learning models, testing them, and analysing their results, it’s time to talk about why the models preformed in that way, and what affected every model predictions.

To start with, the aim of building the models was to classify the product column in our dataset to internet and mobile. Although all models worked on the same problem, the models differ in their results, such as accuracy, precision, recall, and F1 score.

Decision tree and random forest performed better than KNN and Naïve Bayes, but why? The first thing to take into consideration is that a random forest is built by combining many decision trees, this clarifies why their results were very close to each other. But this doesn’t mean that both models can perform equally always, because in some cases, decision tree may be better than random forests and vice versa. As a random forest consists of many trees, it needs big amount of processing power, also, if the dataset is huge it may take a lot of time. Decision trees are fast in exploring data and finding relations, this explains why it outperformed other models. Another thing is that decision trees requires less data preparations, so if there are some mistakes in the pre-processing, if we have some null values or outliers the model may not be affected. In addition, decision trees can take the feature as it is, without the need of transforming the feature to numeric values.

Naïve Bayes was the third model, it performed moderately comparing it to other models’ performance. Naïve Bayes is a fast algorithm that doesn’t require much training data, that could be the reason why it performed in a good way; as our dataset is considered a small dataset comparing it to other datasets. But the thing that lowered the accuracy for this algorithm was that Naïve Bayes assumes that all features in the dataset are independent, which is a wrong assumption about our dataset, because there is a lot of related features.

While Decision tree, random forest, and Naïve bayes shows good results, the KNN didn’t. KNN is an lazy algorithm that has zero training time, as all the time goes on classification, as this can be taken as an advantage, we can also think about it as a limitation, because spending the whole time on classification won’t be a good idea when we deal with large datasets as it would take a lot of time trying to calculate the distance between the features. Another thing that affects KNN performance and effectiveness is K value, choosing incorrect K value will result in false predictions.

## Recommendations

After all, the process from loading the data all the way to the effectiveness of the models was good, organized, and planned. But of course everything could be improved, starting from the pre- processing, using machine learning models to impute the missing values would be a good choice, or maybe trying harder to find relation between some columns instead of dropping them. Moving to the machine learning models and starting with the KNN, I think we can improve the model by searching for other parameters as changing K value haven’t contributed a lot. Decision tree and random forest performed really good, but we can also try to find some ways to improve the accuracy of Naïve Bayes model. For visualization, I think the results where visualized in a clear way that everyone can understand. Lastly, the whole process was fine, but it would be better if we apply the recommendations above.

# References

Anon., 2021. *Decision Tree Advantages and Disadvantages: Decision Tree Regressor.* [Online]   
Available at: https://www.educba.com/decision-tree-advantages-and-disadvantages/  
[Accessed 1 2 2023].

Anon., 2021. *Understanding Random Forest.* [Online]   
Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2  
[Accessed 1 2 2023].

Anon., 2022. *Alternative to Colab Pro: Comparing Google's Jupyter Notebooks to Gradient Notebooks (Updated!).* [Online]   
Available at: https://blog.paperspace.com/alternative-to-google-colab-pro/#:~:text=Free%2Dtier%20Colab%20will%20almost,VMs%20to%2052%20GB%20RAM.  
[Accessed 1 2 2023].

Anon., 2022. *Gaussian Naive Bayes: What You Need to Know?.* [Online]   
Available at: https://www.upgrad.com/blog/gaussian-naive-bayes/#:~:text=features%20are%20independent.-,What%20is%20Gaussian%20Na%C3%AFve%20Bayes%20algorithm%3F,theorem%20with%20strong%20independence%20assumptions.  
[Accessed 1 2 2023].

Anon., 2022. *K Nearest Neighbours Explained.* [Online]   
Available at: https://towardsdatascience.com/k-nearest-neighbours-explained-7c49853633b6#:~:text=It's%20main%20disadvantages%20are%20that,it's%20a%20memory%20based%20approach.  
[Accessed 1 2 2023].

Anon., n.d. *Naïve Bayes Algorithm's Advantages and Disadvantages: Data Science and Machine Learning.* [Online]   
Available at: https://www.kaggle.com/getting-started/225022  
[Accessed 1 2 2023].

Anon., n.d. *Python Courses & Tutorials.* [Online]   
Available at: https://www.codecademy.com/catalog/language/python  
[Accessed 1 2 2023].