# Technical Documentation

# Introduction

## Types of data structures

Data structures are helpful in organizing, managing, and storing data in a specific way, so it can be easily accessed, modified, and processed. Each type of data structure has its own strengths and weaknesses, and the choice of data structure depends on the specific requirements of the project.

There are two types of data structures:

* Built-in data structures: such as list, tuple, dictionary, and set.
* User-defined data structures: such as stack, queue, tree, and linked list.

**Built-in data structures:**

There are two types of built-in data structures:

* Mutable: these data structures can be modified after they are created, items can be added, removed, or changed after they are created. Such as, lists, sets, and dictionaries. They are useful when the data need to be modified frequently.
* Immutable: this types cannot be modified after they are created. For example, tuples. They are useful when the data needs to be protected from being modified.

**Examples of built-in data structures:**

* Lists: one of the most fundamental and versatile data structures. They are ordered collection of data, which can contain different types. The lists are frequently used to materialize iterator or generator expressions in data processing. Lists are mutable data structures, so items can be added or removed. Also, we can combine concatenate and combine lists together, moreover, we can sort or slice lists.
* Tuples: are fixed length and immutable data structure and the data in the tuples can be any data type. They are useful for situations where data needs to be stored and protected from being modified. Tuples can be concatenated, sorted, and can be unpacked (extract the values back to variables)
* Dictionaries: are used to store key-value pairs, where each key and value associated with together, key and value can be any python data type. Dictionaries are used when data needs to be quickly retrieved by a specific key, in order to add, modify, or delete as its mutable data structure, also, it’s possible to merge dictionaries together.
* Sets: are unordered collection of unique elements. these elements can be any python data type, but the set can’t store a mutable elements such as list or dictionary. Sets are useful for storing unique elements and perform the set operations such as union, intersection, difference, and symmetric difference. We can’t access or change an element in sets using slicing and indexing.
* Arrays: they are not built-in data structure, but they can be used by third party library called numpy, they are efficient for mathematical operations and often used for numerical computations. Arrays can store any python data structure.

## Common libraries

Python is full of many libraries that is very helpful and time effective for every programmer, libraries are pre-written pieces of code which can be imported and used. These libraries are the libraries we used and also, they are ethe libraries that is most used in python:

## NumPy stands for numerical python, and it is a library the is used for numerical computing in python, such as dealing with arrays and matrix. moreover, it is also has many built-in functions that is used in working with linear algebra, matrices. I used it

## pandas is a library that is used for data analysis in python, it contains data structures and data manipulation tools that are made in order to analyse, explore, manipulate, and clean the data and analyse it easily and fast. It contains many data structures such as data frames and series. I used it in reading the csv file and creating a data frame and in dealing with the data frame.

## SciPy is a Python library that provides functionality for scientific computing and engineering. it include many algorithms and tools for statistical analysis, probability distributions, linear algebra, etc... making it a powerful tool for scientific data analysis and visualization. I used a function in it that calculate the mode.

## The datetime library contains tools to deal with dates and times. It offers functions for formatting and parsing dates and times, as well as classes to handle the dates, times. The datetime library can also be used to conduct time-related calculations. I used in converting a sting date into a datetime data type in order to do calculations on the date.

## Google collab library has many uses, but one that we used is that we needed it in order to mount the csv file from google drive.

## Plotting and visualization libraries

As mentioned above, libraries are so useful and essential nowadays. As there is libraries that deal with date or numerical data, also there is library that are used to produce a visual representation of the data that is used to understand the data better and analyse it, in my project I used matplotlib in order to create a visual representations.

Matplotlib is a Python plotting library that create and deal with 2D and 3D plotting. It is used for data visualization and analysis. I used it in plotting heatmaps, bar charts, boxplots, and radar charts.

# Experiments

## Programming languages and tool

Python is a very strong programming language because of many reasons such as its vast library of modules and functions that allows the developer to quickly and easily build complex applications without wasting the time in writing the code from scratch, also python is full of many data types and dynamic typing system that made the work easier to manipulate with data.

We also used google colab in order to write the python program on it without the need to set up a local development environment and made it easier for us to start with our projects, moreover, colab provides powerful GPUs and TPUs which allow us to run our code fastly.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Column/step Name** | **Description** | **Justification** |
|  | Check for the nulls and duplicated rows | I checked the number of nulls in each column and for duplicated rows | * The number of nulls in each column will lead me how to deal with it whether deleting the column or fill the nulls, etc. * There isn’t any duplicated rows, but if they exist, they should be dropped and removed from the data frame. As keeping them will mislead the model and decrease the accuracy of prediction |
|  | Drop columns | I dropped RESOLUTION, RESOLUTION\_DESCRIPTION, CASE\_DESC, and CASE\_ID columns using drop function | * I dropped these columns (RESOLUTION, RESOLUTION\_DESCRIPTION, CASE\_DESC) because they contains so many nulls that is above 9000 null in each column, so filling the nulls in them won’t be accurate and it can make the model inaccurate. * I dropped the case id column because it’s not necessary or needed in modelling and dropping it will make the data frame easier to deal with and faster in any computation or analysis performed. |
|  | Filling nulls in OFFER\_NAME, CUSTOMER\_GROUP, CLOSE\_GROUP, OPEN\_GR | I filled the nulls in these column by using group by function and take the mode after the grouping | I used the group by function to group the data frame base on specific columns and take the mode of each group and used transform function to fill it in all the nulls of each group. The columns which data frame were grouped on has been chosen based on the heatmap, so I was looking for the most three columns that has a high correlation value whether in positive or negative and use them in the group by function |
|  | Fill the nulls in OPEN\_USER and CLOSE\_USER | 1. I filled the nulls of open user from the close user and vice versa. 2. I filled the rest of nulls in close user using group by and the method in step 2. 3. I filled the rest of nulls in open user from the close user again | 1. I used eq function which shows how many values in open user equal to the values in close user in the same row and it was 7024 equal row, so I filled each one from the other 2. I used for the rest of the nulls in close user a group by function and as in step 2 above, and I chose the close user to be filled by group by because the sum of correlation values was higher comparing to open user column. 3. I filled the rest of nulls of open user from the close user again. |
|  | Fill the nulls in ESCALATED\_GROUP | 1. Filled the nulls by “no escalation group” when the escalation flag is no 2. Then fill the rest of nulls by group by function as in step 2. | 1. When the escalation flag value is no means that the complaint hasn’t been escalated which means there is no escalation group, so I filled “no escalation group” in nulls. 2. The rest of nulls has been filled using group by as in step 2. |
|  | Fill the nulls in CLOSE\_DATE and AGE\_BRACKET | 1. I converted the open date and close date values from string to datetime data type. 2. Fill the null in age bracket column with (-999) 3. Fill the null in close date by added the age bracket from the open date. 4. Drop the rested null rows in close date which was 19 row | 1. I converted the date in these column from their string to datetime data type in order to do subtraction. 2. I filled the age bracket nulls with illogical value which is (-999) that represent that the case isn’t not closed yet. 3. I added the age bracket to open date and filled it in the close date, so the close date will be before the open date and it’s illogical value but represents that the case isn’t closed yet. 4. I dropped the rested rows which was 19 row because there is now way to know the open date and close date without any additional information, and deleting these rows won’t affect the model performance so much. |
|  | Fill the nulls in the CALLBACK\_MECHANISM | 1. I filled “no callback” when the current status of the complaint is active. 2. I filled the nulls with “phone” when the difference between the open date and close date is less then 1800 seconds = 30 minutes 3. Fill the rest with “no preferred call-back mechanism’’. | 1. When the current status is active it means that the complaint isn’t closed yet, so there is no call-back needed. 2. When the difference between the open and close date is less or equal 30 minutes, it means that is probably the complaint was from phone call because if it was email or SMS, it would take longer to respond so the difference will be more than 30 minutes. 3. The rest of nulls I filled with no preferred call-back because it would be inaccurate to predict customer’s preference. |

## Approaches

|  |  |  |
| --- | --- | --- |
| **Approach no.** | **Name** | **Description** |
|  | K nearest neighbours (KNN) | a supervised machine learning approach for classification and regression. It works by locating the K nearest neighbours to a new data point and predicting the class of the new data point depending on the majority class of those neighbours. The number of neighbours, K, is a parameter that may be adjusted in order to give the best results. |
|  | Naive Bayes (NB) | Naive Bayes is a supervised machine learning algorithm that is used for classification. It is built on Bayes' theorem, which asserts that the likelihood of a class given an instance's characteristics is proportional to the probabilities of the features given the class multiplied by the class's prior probability. |
|  | Random forest (RF) | is a learning algorithm that predicts based on several decision trees. Each tree in the forest is generated from a random subset of the characteristics and data. then it merge the various tree predictions to get the final prediction. |
|  | Decision tree (DT) | a supervised machine learning approach used for classification and regression problems. The algorithm works by continuously separating the data into subsets depending on the characteristic that delivers the maximum information benefit, until the data is divided into subsamples. The resultant tree may be used in generating predictions, by tracking the splits to the leaf nodes. |

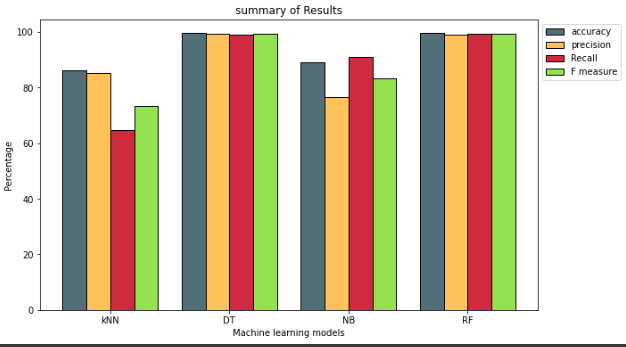
# Results

## Compare the different models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Approach no.** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
|  | 86.01% | 85.2% | 64.5% | 73.4% |
|  | 88.97% | 76.59% | 90.93% | 83.14% |
|  | 99.53% | 99.09% | 99.33% | 99.21% |
|  | 99.48% | 99.2% | 99.06% | 99.13% |

## Charts

**Bar chart:**



**Boxplot**:

Chart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

**Radar chart:**

Chart, radar chart

Description automatically generated

## Analysis of the results

Both Random Forest and Decision Tree got the highest accuracy with 99.53% and 99.48% respectively, indicating that both approaches are very effective in correctly classifying instances. Random Forest and Decision Tree both have high precision and recall values of 99.09% and 99.33% in recall, 99.2% and 99.06% in precision. Random Forest has the greatest F1-score of any technique, with 99.21%.

KNN, on the other hand, has poorer accuracy, precision, recall, and F1-Score. With 90.93% recall, Naive Bayes is in between of three other models in all measures. According to the data, Random Forest is the best approach out of the four, having the highest accuracy, precision, recall, and F1-Score.

Bar chart: it is very clear that the RF and DT are taking the lead based on the measures value above, but if looked clearly and go back to the table of measures value, RF is higher than the DT in all measures except the precision. While KNN was the least ranked out in all the measures expect the precision, where the NB was the least value in that measure.

Boxplots: The plots show that Random Forest and Decision Tree have higher accuracy compared to Naive Bayes and KNN. The median value of accuracy for Random Forest and Decision Tree is close to 99.5%, while the median value of Naive Bayes is around 89%, and around 86% for KNN. Moreover, the boxplot shows the five descriptive statistics (min,max,Q1,Q3,median).

Radar chart: just like the bar chart, the radar chart shows the performance of each model based on each measure, and the measure are divided to be on the 360 circle.

# Evaluation

## The choice of data structures

Firstly, I used the data frame data structure in order to deal with data and handle it, pre-processing, modelling and visualization. As the data frames from pandas offers a high level of flexibility and it integrates easily with other libraries such as NumPy and matplotlib. Also, its fast in processing.

Then I used lists when calculating the measures values of each model every single iteration, so I used lists because it was easily to use and to deal with, they are memory efficient because their size can be adjusted based on the number of elements they have, easily to store and manipulate the data inside of it, and can be NumPy functions can be performed on it. Moreover, you store any data type in the lists so it made it easily to deal with many problems using lists.

## Selection of the appropriate libraries

Firstly, I used NumPy library, because it afford high performance mathematical computations which means that it afford a variety of mathematical functions that optimized for fast computation. Moreover, it is easily integrated with other libraries such as pandas and matplotlib, and also it efficient in memory management which enable it to make mathematical computations on large arrays.

Secondly, I used the pandas library a lot, because it is a very powerful library in data manipulation and analysis because it provides a variety of functions for cleaning, transforming, and aggregating the data, moreover, it is fast in processing and it can e easily used with other libraries such as NumPy and matplotlib. Lastly, it afford a convenient data representation.

## The effectiveness of different models

Random Forest were the best model that achieved the highest values in measures because RF is known for its ability to handle large datasets and come up with high accuracy model, because of his low overfitting risk. Moreover, It is a powerful algorithm because it combines multiple decision trees to make predictions, which can lead to better performance than a single decision tree.

The second model that scored high was the decision tree because from its simplicity and efficiency as a model, also it can handle non-linear relationships, and it less likely to be affected to outliers.

## Recommendations

I would revise me pre-processing and try to choose different column in the group by and see how the results of the models are affected, or I can fill the date in different way which maybe can increase the accuracy of one model. Moreover, I would try to increase the iterations of each model so I can get the best results I could have and ensure that my results are accurate and true.

In knn, maybe I would try more k values and try to examine the best k to use as our data set is relatively big so we can put high k values that are above 11 or something like that, so we can try different k values in order to increase the accuracy and other measures of this model.

## References:

* Course slides
* *Pros and cons of using pandas* (no date) *Studytonight.com*. Available at: https://www.studytonight.com/pandas/pros-and-cons-of-using-pandas (Accessed: February 1, 2023).
* Real Python (2021) *When to use a list comprehension in Python*, *Real Python*. Real Python. Available at: https://realpython.com/list-comprehension-python/ (Accessed: January 28, 2023).
* *The Complete Guide to pandas dataframe* (2022) *Databricks*. Available at: https://www.databricks.com/glossary/pandas-dataframe (Accessed: January 29, 2023).
* McBride, M. (no date) *Key advantages of NumPy*, *PythonInformer*. Available at: https://www.pythoninformer.com/python-libraries/numpy/advantages/ (Accessed: January 29, 2023).
* *Random Forest vs decision tree: Key differences* (no date) *KDnuggets*. Available at: https://www.kdnuggets.com/2022/02/random-forest-decision-tree-key-differences.html (Accessed: February 1, 2023).
* Albert, S. (2018) *Why random forest is the greatest!*, *Medium*. The Making Of... a Data Scientist. Available at: https://medium.com/diogo-menezes-borges/random-forests-8ae226855565 (Accessed: February 1, 2023).