

**Data Science Programming**

**10204281**

**Section (4)**

**Data Preprocessing, Modeling, and Reporting**

**Submitted to**

Dr. Sinan Kamal

**Submitted on**

February 1st, 2023

**Submitted by**

Marwan Tareq Shafiq Al Farah

**Student ID**

21110011

**Fall 2022 – 2023**

**Table of Contents**

[***Technical Documentation*** 3](#_Toc125886929)

[**1.** **Introduction** 3](#_Toc125886930)

[**1.1.** **Types of Data Structures** 3](#_Toc125886931)

[**1.2.** **Common Libraries** 4](#_Toc125886932)

[**1.3.** **Plotting and Visualization Libraries** 6](#_Toc125886933)

[**2.** **Experiments** 7](#_Toc125886934)

[**2.1.** **Programming Languages and Tools** 7](#_Toc125886935)

[**2.2.** **Preprocessing (Loading and Preparing the Data)** 7](#_Toc125886936)

[**2.3.** **Approaches** 9](#_Toc125886937)

[**3.** **Results** 10](#_Toc125886938)

[**3.1.** **Compare the Different Models** 10](#_Toc125886939)

[**3.2.** **Charts** 11](#_Toc125886940)

[**3.3.** **Analysis of the results** 13](#_Toc125886941)

[**4.** **Evaluation** 14](#_Toc125886942)

[**4.1.** **The Choice of Data Structures** 14](#_Toc125886943)

[**4.2.** **Selection of Appropriate Libraries** 15](#_Toc125886944)

[**4.3.** **The Effectiveness of Different Models** 16](#_Toc125886945)

[**4.4.** **Recommendations** 16](#_Toc125886946)

[**5.** **Conclusion** 17](#_Toc125886947)

[***References*** 18](#_Toc125886948)

# ***Technical Documentation***

1. **Introduction:**

In this technical document, we will present a proof of concept for the application of data science in a telecom company, using the complaints data provided in the dataset. Our goal was to generate several models using appropriate libraries and evaluate them using different evaluation measures.

We begin with an overview of data structures and common libraries used in data science, illustrated with examples from our program. Following that, we go over the data preparation and cleaning procedure, including how to deal with missing values and categorical variables. The various models produced, and their evaluation outcomes are then shown. Finally, we present a detailed analysis of the data, accompanied by relevant visualizations.

We hope that this document gives a clear overview of the proof of concept and potential applications of data science in the telecom business. Our findings highlight the importance of data science in comprehending consumer complaints and potentially enhancing customer happiness and satisfaction in the future.

1. **Types of Data Structures:**

Data science is the field that involves organizing and managing vast amounts of data. This can be performed with the help of data structures, which are an important tool used by data scientists since they are a method of storing and organizing data so that it can be quickly and readily retrieved, modified, and evaluated.

The following built in data structures are often used in data science:

* **Lists:** they are a basic data structure in Python and many other programming languages that come with the language itself without the use of a library. They are an ordered collections of elements that can be of any type which mean that they are versatile and can be used to represent many types of data, such as a list of numbers, a list of strings, or a list of other lists. Lists are also mutable, which means that the elements that are in the list are changeable. (Wilkinson, 2022)
* **Sets:** it is a collection of unique elements. Sets are often used in data science to eliminate duplicates from a dataset. (Wilkinson, 2022)
* **Dictionaries:** it is a group of key-value pairs with each key being distinct. In data science, dictionaries are frequently used to map keys to values, like converting a string name to a numeric ID. (Wilkinson, 2022)
* **Tuples:** it is an immutable data structure, which mean that its elements cannot be modified after its creation, and it is very similar to a list because it can contain any type of data. Tuples are often used to represent a record or a row of data in a dataset. (Wilkinson, 2022)

The following user defined data structures are often used in data science:

* **Pandas DataFrames:** similar to a spreadsheet or an SQL table, a DataFrame is a two-dimensional table of data comprising rows and columns. Large datasets are frequently stored in and worked on using DataFrames, the main tool for working with data in Pandas. (GeeksforGeeks, 2019a)
* **Pandas Series:** it is a component of the Pandas library and resembles a column in a spreadsheet; it is frequently used in the data cleaning and modification processes. It may hold any form of data. (GeeksforGeeks, 2019b)
* **NumPy arrays:** A numpy array is a multi-dimensional array object that can store large numerical data. It is used to perform mathematical and logical operations on arrays. (GeeksforGeeks, 2020)

Each of these data structures has its own strengths and weaknesses and choosing the right one depends on the specific requirements of the task at hand. Understanding the characteristics of these data structures is important for data scientists to be able to make use of them in their work. The source code for this project uses several different data structures, for example:

* **Pandas DataFrame:** The data from the CSV file is stored and processed using a pandas DataFrame called "complaints\_df". The main data structure used throughout the code to manipulate the data is a DataFrame, which holds the complete dataset. Using a DataFrame makes it very simple to manipulate the data by selecting columns, performing mathematical operations on the full dataset, or substituting particular values for missing values.
* **Pandas Series:** when filling the columns with random values produced by numpy's random.choice function, the code frequently uses pandas Series to fill in the missing values. The function "fillna" receives the created values. The missing value is filled with a new series using this function, which requires a pandas series as an input.
* **Numpy array:** Numpy arrays are used in the code to either fill in the missing values when a condition is true or to create random values for specific columns based on the distribution of the non-missing values.
* **Python dictionary:** the code maps certain column values using two Python dictionaries called "dict1" and "dict2," and then uses those mapped values to fill in the blanks for specific columns. Because they offer a means to store a lot of key-value pairs, where each key needs to be distinct, dictionaries are utilized in this code.
* **Python lists:** The code uses one Python lists named “list1” and “list2” to store random values for certain columns based on certain column value.

1. **Common Libraries:**

There are several common libraries used in data science, some examples are:

1. **Pandas:** Python's Pandas library is used to manage and work with data in a tabular manner. It contains a number of practical functions for cleaning, transforming, and analyzing data, as well as data structures like DataFrame and Series (ActivateState, 2020). The code uses various pandas functions to perform operations on the DataFrame to read the csv file, replace missing values, calculate value counts, and map dictionary. Some of the functions that were used include:

* **pd.read\_csv():** This function reads a CSV file into a DataFrame and stores it in complaints\_df.
* **value\_counts():** This function returns the frequency of each value in the column.
* **count():** This function returns the number of elements in the column.
* **fillna():** This function fills all missing values in the column with another value.
* **isnull():** This function identifies rows with missing values.
* **map():** This function maps the values in the column to the corresponding values in the dictionary

1. **NumPy:** Python’s NumPy library is used for performing numerical operations on data, such as linear algebra, statistical operations, and random number generation (GeeksforGeeks, 2022). In the code above, it uses the numpy array mostly to generate random values based on the distribution of the non-missing values. Some of the functions that were used include:

* **np.random.choice():** it is used to randomly select elements from an input list, based on a specified probability distribution.
* **np.where():** it returns an array with elements from x where the given condition is True, and elements from y where the condition is False.

1. **Scikit-learn:** Scikit-learn is a machine learning library that offers a variety of tools for creating, training, and evaluating models. Additionally, it has many tools for preprocessing data and testing models, such train\_test\_split, as well as measures like accuracy\_score, precision\_score, recall\_score, and f1\_score (ActiveState, 2022). It also offers a range of effective machine learning and statistical modeling methods, such as clustering, regression, and classification (Majumdar, 2021). The code above uses this library to import different models like KNeighborsClassifier, DecisionTreeClassifier, GaussianNB, RandomForestClassifier and use them to train the models. Some of the functions that were used include:

* **train\_test\_split():** it is used to split the data into training and testing sets. It can take multiple parameters such as test\_size, random\_state etc.
* **accuracy\_score(), precision\_score(), recall\_score(), f1\_score():** they are used for evaluating the models by providing the accuracy, precision, recall, and f1 score of the model.
* **KNeighborsClassifier(), DecisionTreeClassifier(), GaussianNB(), RandomForestClassifier():** these are different machine learning classifiers available in the library, and it is used to train the models.

1. **Matplotlib:** Matplotlib is a library that allows you to create static, animated, and interactive visualizations (Hunter *et al.*, 2020). It supports a wide range of plot formats, including line plots, scatter plots, and histograms, and it can also be used to provide graphics that complement the findings. One of the most often used libraries for data visualization in data science is Matplotlib because it offers a wide variety of plotting choices and is very customizable (such as changing the color, style, and size of the plots, which allows the code to create visually appealing plots), making it a flexible alternative for producing different types of plots and charts (Krishna Madan, 2022).

These are just a few examples of the many libraries available in data science. Some other commonly used libraries are Seaborn, TensorFlow, Keras, SciPy and many more. Data scientists often use a combination of these libraries depending on the task they are performing and the specific requirements of their project. (Custer, 2020)

1. **Plotting and Visualization Libraries:**

Data visualization is an essential part of data science, and there are several plotting and visualization libraries available to help data scientists create effective and engaging visualizations. Some popular libraries used in data science include:

1. **Matplotlib:** Go to 1.2.4.
2. **Seaborn:** Seaborn is a Matplotlib-based library that provides a simpler and more user-friendly interface for creating statistical graphs. It also able to create charts with pandas DataFrames, which is especially handy for displaying huge datasets. (Custer, 2020; Waskom, 2020)
3. **Plotly:** Plotly is a library that lets you make interactive plots and charts. It can be used in Python, R, and JavaScript and offers a variety of chart styles such as heatmaps, scatter plots, and 3D graphs. (Plotly, 2009; Custer, 2020)
4. **Bokeh:** Bokeh is a library for developing interactive plots and charts for web-based applications. It supports a variety of chart styles and is especially excellent for producing huge, complicated visualizations that can be shared easily and embedded on websites. (Custer, 2020; Rastogi, 2021)

These are only a few of the many plotting and visualization libraries accessible to data scientists, and the library chosen will rely on the project's unique requirements as well as the data scientist's skill level.

When it comes to the code, Matplotlib is the library that was utilized for the visualization as it makes use of a number of its functions to display the data in a variety of formats, including bar plots, box plots, and radars (polar plots). These plots can be used to display the data in an understandable and straightforward manner and are appropriate for visualizing many sorts of data.

Another factor in the decision of choosing Matplotlib, is due to the compatibility of it with other libraries (such as pandas and numpy) which are frequently used for data analysis and manipulation.

It is also simple to find solutions to issues and examples of how to use the library thanks to the Matplotlib community and the abundance of information accessible online. This makes it a fantastic option for data visualization, and data science initiatives frequently employ it.

In conclusion, Matplotlib is a strong and flexible library for data visualization that is extensively used in data science because it has a huge community, a variety of charting options, and is interoperable with other libraries. These are the reasons why we chose to use Matplotlib for data visualization.

1. **Experiments:**

In this section of the technical document, we will be discussing some of the experiments conducted and the results obtained from these experiments. We will be focusing on the data preprocessing, classification, and model evaluation steps that were performed and we will also be providing appropriate charts and visualizations to illustrate the results obtained from the experiments. Additionally, we will be comparing the different models applied using different evaluation measures and explaining the performance of each model. In conclusion, this section of the document aims to provide a comprehensive overview of the experiments conducted and the results obtained, and to evaluate the effectiveness of the applied models.

* 1. **Programming Languages and Tools:**

Python and Google Colab were utilized as programming languages and tools for this proof of concept. Python is a popular choice among data scientists due to its ease of use, readability, and extensive library support for data science applications. Colab is an open-source web-based tool, similar to Jupyter, for creating documents that include live code, graphics, and text. Pandas, Numpy, and Scikit-learn are the primary libraries used in this project for data preprocessing, modeling, and evaluation. Matplotlib was also used for data visualization. The usage of these libraries enabled us to import, clean, and analyze data, as well as train and assess various models swiftly and effectively.

* 1. **Preprocessing (Loading and Preparing the Data):**

After reading the CSV file (Complaint.csv) and storing it in the DataFrame (complaints\_df), the first step would be to preprocess the data by handling missing values, replacing null values, cleaning the data, and converting categorical variables into numerical variables. This is conducted in the code in a variety of ways:

* Filling missing values in the 'OFFER\_NAME' column with random samples selected from the column's values, with the likelihood of drawing each value proportionate to its frequency in the data. This is done to guarantee that missing values do not skew data analysis and that the data is representative of the column's real distribution of values.
* Filling missing values in the 'CUSTOMER\_GROUP' column with random samples selected from the values in the column for the corresponding 'OFFER\_NAME' value, with the likelihood of drawing each value proportionate to its frequency in the data for that 'OFFER\_NAME' value. This phase ensures that the data is reflective of the actual distribution of values in the column for each 'OFFER\_NAME' value, such that missing values do not skew the data analysis and that the data is representative of the actual distribution of these values.
* Filling missing values in the ‘AGE\_BRACKET’ column with the string “The Case is Still Active”. This was done because it was noticed that the missing values in the ‘AGE\_BRACKET’ column all correspond to the string “Active” in the ‘CURRENT\_STATUS’ column, which means that the case has not been resolved yet.
* Filling missing values in the ‘CLOSE\_GROUP’ column with the corresponding values in the string ‘The Case is still active’ column for cases where the ‘CURRENT\_STATUS’ is not “Resolved”. Filling missing values in the ‘CLOSE\_GROUP’ and ‘OPEN\_GR’ columns by mapping the values in the ‘CLOSE\_USER’ and ‘OPEN\_USER’ columns to the values in the ‘CLOSE\_GROUP’ and ‘OPEN\_GR’ columns respectively. Filling remaining missing values in the ‘CLOSE\_GROUP’ and ‘OPEN\_GR’ columns with random samples drawn from the non-missing values in the columns, with the probability of each value being drawn proportional to its frequency in the data. This step is done because each user is in only 1 group, so by mapping the above columns together, we can accurately fill the missing values in the ‘CLOSE\_GROUP’ and ‘OPEN\_GR’ columns. This was also done because when the case is not yet resolved, there cannot be a close group, thus we filled it with ‘The Case is still active’.
* Filling missing values in the ‘ESCALATED\_GROUP’ column with random samples drawn from the values in the column, with the probability of each value being drawn proportional to its frequency in the data. Replacing the filled values in the ‘ESCALATED\_GROUP’ column with the string “The complaint wasn’t escalated” for cases where the ‘ESCALATION\_FLAG’ is not “Yes”. This is done to ensure that the missing values do not skew the analysis of the data and that the data is representative of the actual distribution of values in the column. Also, because it was observes that whenever the ‘ESCALATION\_FLAG’ was “Yes” the ‘ESCALATED\_GROUP’ contained a missing value.
* Filling missing values in the ‘CALLBACK\_MECHANISM’ column with random samples drawn from the values in the column, with the probability of each value being drawn proportional to its frequency in the data. Replacing the filled values in the ‘CALLBACK\_MECHANISM’ column with the string “User Not Filled” for cases where the ‘ACTUAL\_COMPLAINT’ is “User Not Filled”. This is done to ensure that the missing values do not skew the analysis of the data and that the data is representative of the actual distribution of values in the column. Also, because it was observed that whenever the ‘ACTUAL\_COMPLAINT’ was “User Not Filled” the, ‘CALLBACK\_MECHANISM’ column had a missing value (7262 out of 7277).
* Dropping the columns ‘OPEN\_USER’, ‘CLOSE\_USER’, ‘CASE\_ID’, ‘OPEN\_DATE’, ‘CLOSE\_DATE’, ‘RESOLUTION’, ‘RESOLUTION\_DESCRIPTION’, and ‘CASE\_DESC’ from the DataFrame because some of them are not useful for the analysis, may contain more than 80% missing values, or may contain redundant data.
* Keeping the columns ‘CUSTOMER\_TYPE’, ‘CURRENT\_STATUS’, ‘ESCALATION\_FLAG’, ‘ACTUAL\_COMPLAINT’, ‘COMPLAINT\_TYPE’, ‘PRODUCT’, and ‘CASE’, as they are because they are useful for the analysis and because they didn’t contain any missing values.
  1. **Approaches:**

|  |  |  |
| --- | --- | --- |
| Approach no. | Name | Description |
| 1 | KNeighborsClassifier | The KNeighborsClassifier model, is a classification model that predicts the class of a given data point based on its distance from its k-nearest neighbors (IBM, 2022). Cross-validation, a method for estimating a model's performance by splitting the data into training and validation sets and utilizing the validation set to assess the model's performance, is used to determine the number of neighbors employed in the model (Payam Refaeilzadeh, 2010). |
| 2 | DecisionTreeClassifier | This method makes use of the DecisionTreeClassifier model, a classification model that divides the data into subsets based on the values of the input features and utilizes a decision tree to predict the class of a given data point. Cross-validation is used to determine the depth of the tree, which prevents overfitting by reducing the size of the tree. (Datagy, 2022) |
| 3 | GaussianNB | Based on the likelihood of each class given the input features, this method applies the Gaussian Naive Bayes algorithm to predict the class of a given data point. The algorithm bases its decision on the premise that each feature is independent, which may not always be the case but is still a sound presumption to make. (Ray, 2021; Vats, 2021) |
| 4 | RandomForestClassifier | This method makes use of the RandomForestClassifier model, a classification model that predicts the class of a given data point by averaging the predictions of the individual trees inside a forest of decision trees. Cross-validation is used to calculate the number of trees in the forest and their maximum depth, which aids in avoiding overfitting and enhancing model accuracy. (Huneycutt, 2018) |

1. **Results:**

In this section, we will evaluate the performance of the different models applied in the previous section using various evaluation measures such as accuracy, precision, recall, and F1-score. These measures will allow us to compare the models and determine which one performed the best. We will also provide charts to visualize the results and make it easier to compare the models. Finally, we will analyze the results and discuss the implications of the results in relation to the goal of the proof of concept for the company.

1. **Compare the Different Models:**

In this section, we will compare the performance of the different models applied in the previous section. The models that were used in this proof of concept are KNeighborsClassifier, DecisionTreeClassifier, GaussianNB, and RandomForestClassifier. The evaluation measures used to compare the models are accuracy, precision, recall, and F1-score. These measures will allow us to compare the models and determine which one performed the best.

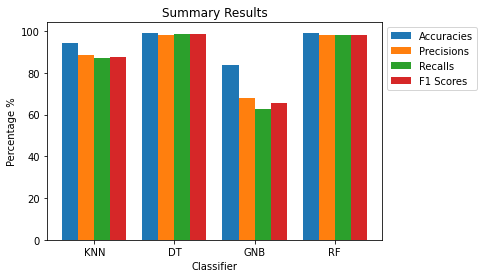
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Approach no. | Accuracy | Precision | Recall | F1-Score |
| 1 | 94.19% | 89.21% | 86.56% | 87.86% |
| 2 | **99.28%** | **98.40%** | **98.62%** | **98.51%** |
| 3 | 83.63% | 67.67% | 62.45% | 64.95% |
| 4 | **99.19%** | **98.49%** | **98.18%** | **98.34%** |

From the table, it is clear that the DecisionTreeClassifier and RandomForestClassifier models performed the best in terms of overall performance. These models achieved high accuracy, precision, recall, and F1-score, indicating that they were able to accurately classify the data and minimize false positives and false negatives. The KNeighborsClassifier model also performed well, however, it had a lower recall and f1-score than the other models. The GaussianNB model performed poorly in comparison to the other models, with lower accuracy, precision, recall, and F1-score. This can be attributed to the assumption of the algorithm which is that the features are independent of each other.

1. **Charts:**

In this section, we will use a combination of visualizations to help us better understand the performance of the different models. These charts will help us to easily compare the performance of the different models and identify which one performed the best.

The first chart is a bar graph that compares the evaluation metrics of all four algorithms (KNeighborsClassifier, DecisionTreeClassifier, GaussianNB, and RandomForestClassifier) side by side. It shows the mean accuracy, precision, recall, and F1-score for each algorithm. This allows for a quick visual comparison of the performance of each algorithm.



The second set of charts in the analysis consist of four box plots, each representing a distinct algorithm. These plots describe the distribution of the evaluation metrics for each algorithm, with the box indicating the interquartile range, the line inside the box indicating the median, and any dots outside the box representing outliers. These plots provide a more in-depth understanding of the performance of each algorithm.

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

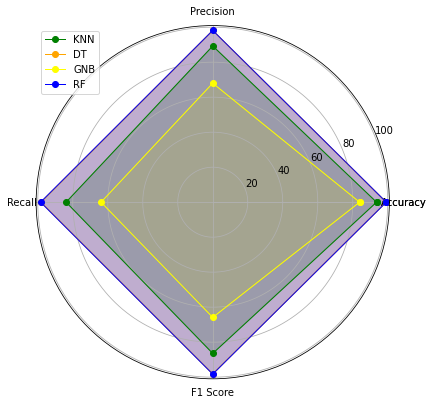
Description automatically generatedChart, box and whisker chart

Description automatically generated

It is noted that the data for KNeighborsClassifier, DecisionTreeClassifier, and GaussianNB models does not result in a box in the box plots, as opposed to the RandomForestClassifier model. This is due to the RandomForestClassifier utilizing an ensemble method that involves multiple decision trees, each trained on a randomly selected subset of the data, resulting in independent predictions and diverse evaluation metrics for each iteration.

In contrast, the KNeighborsClassifier, DecisionTreeClassifier, and GaussianNB models do not randomly select their data subsets, resulting in the values of each row in the arrKNN, arrGNB, arrDT arrays being identical to each other. The same set of evaluation metrics are calculated for each iteration, as well as the same test data being used to calculate these metrics. Additionally, the classifier is trained on the same training data at the start of each iteration, leading to consistent predictions and identical evaluation metrics for each iteration.

The last chart is a radar chart. A radar chart (also known as a polar plot) is a two-dimensional graphical representation of data, with the axes spreading outward from the center of the chart. The axes are similarly spaced and radially organized, allowing for visual comparison of multivariate data. It is frequently used to present multivariate data, such as comparing many attributes or dimensions of a single entity. The data points on the chart are plotted around the perimeter of a circle, giving the chart a spider-web appearance. This type of graphic is especially effective for finding patterns or trends in large amounts of data (EdrawMax, 2021; TIBCO Software, 2022). In this radar chart it shows the evaluation metrics of all the algorithms. This chart gives a clear visual representation of the overall performance of each algorithm in relation to the other algorithms. It allows for easy comparison of the strengths and weaknesses of each algorithm.



As we can see from the chart above, we can only see KNN, GNB, and RF. The DT data is not visible on the radar because the values of the DT data are very close to the RF data. In a radar, each line represents a classifier and is plotted on the same graph. This means that if two classifiers have similar values, the line of the classifier with the highest value will be plotted on top of the other one. Therefore, in this case, if the DT and RF have similar values, the line of the RF classifier will be plotted on top of the DT classifier, making it difficult to see the DT classifier. Based on these visualizations, it confirmed and made it is clear (visually) that the DecisionTreeClassifier and RandomForestClassifier models performed the best in terms of overall performance as these models achieved high values for accuracy, precision, recall, and F1-score, indicating that they were able to accurately classify the data and minimize false positives and false negatives.

1. **Analysis of the results:**

The results provided above for four different approaches: K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes, and Random Forest.

In terms of overall accuracy, Decision Trees (99.28%) and Random Forest (99.19%) performed the best, followed by K-nearest neighbors (94.19%) and Gaussian Naive Bayes (83.63%).

In terms of precision, Decision Trees (98.40%) and Random Forest (98.49%) had the highest scores, followed by K-nearest neighbors (89.21%) and Gaussian Naive Bayes (67.67%). Precision measures the proportion of true positive predictions out of all positive predictions made.

In terms of recall, Decision Trees (98.62%) and Random Forest (98.18%) had the highest scores, followed by K-nearest neighbors (86.56%) and Gaussian Naive Bayes (62.45%). Recall measures the proportion of true positive predictions out of all actual positive cases.

In terms of F1-Score, Decision Trees (98.51%) and Random Forest (98.34%) had the highest scores, followed by K-nearest neighbors (87.86%) and Gaussian Naive Bayes (64.95%). The F1-score is a harmonic mean of precision and recall, which gives a balance between the two.

In summary, Decision Trees and Random Forest perform well in all four metrics, while K-nearest neighbors and Gaussian Naive Bayes may not be as reliable in predicting values for this dataset.

1. **Evaluation:**

In this section, we will evaluate the overall effectiveness of the proof of concept in relation to the goal of the project. This will include an assessment of the data structures used, the performance of the models, and the potential impact of the results on the company.

1. **The Choice of Data Structures:**

The Pandas DataFrame was the primary data structure used in this proof of concept. This structure was chosen for its capacity to handle big datasets, as well as missing values and data cleaning activities. It also includes a wide range of data editing, analysis, and visualization capabilities. This made it an excellent choice for dealing with the Complaint.csv file.

The Pandas Series was also a prominent data structure used in this proof of concept because when filling the columns with random values produced by numpy's random.choice function, the code frequently uses pandas Series to fill in the missing values. The missing value is filled with a new series using this function, which requires a pandas series as an input It is a convenient data structure for holding a single column of data.

In addition, dictionaries were also used to store the mapping of users and groups, which was used to fill the missing values in the dataset. This was a convenient data structure for this purpose as it allows for fast lookups and easy manipulation.

The code also uses one Python lists to store random values for certain columns based on certain column value. They were used because lists are a basic data structure that allow for easy storage and manipulation of a collection of items.

Overall, the choice of data structures in this proof of concept was based on the specific needs of the task and the ability of the structures to handle and manipulate the data effectively.

1. **Selection of Appropriate Libraries:**

In this proof of concept, we used several libraries such as Pandas, Numpy, and Scikit-learn for data preprocessing, modeling, and evaluation. These libraries were implemented in the code as they are widely used in data science tasks and have a lot of built-in functions and modules that are useful for data manipulation, analysis, and visualization.

Pandas was used because it is a library that provides data structures and data manipulation tools for handling and analyzing large datasets. It allows for easy data cleaning, preprocessing, and visualization, which were all important steps in this proof of concept. The DataFrame structure provided by Pandas was used to store the data and perform various data manipulation tasks such as filling missing values and converting categorical variables into numerical variables.

Numpy was used because it is a library that provides support for large multi-dimensional arrays and matrices of numerical data. It provides a lot of mathematical functions that are useful for data manipulation and analysis, such as array operations and linear algebra. Numpy was used in this proof of concept to handle arrays of data and perform mathematical operations on them.

Scikit-learn was used because it is a library that provides a wide range of machine learning algorithms and tools for data analysis and modeling. It provides a consistent interface for training and evaluating models, which made it easy to implement and compare different models in this proof of concept. The library also includes modules for feature selection, model evaluation, and cross-validation, which were all important steps in the analysis.

Matplotlib was used because it is a library that provides visualization tools for creating plots, charts, and other visualizations. It was used in this proof of concept to create various visualizations of the data and the results, such as bar graphs, box plots, and radar charts, which helped to understand and compare the performance of the different models.

Overall, the choice of libraries used in this proof of concept was essential to the success of the analysis. These libraries are widely used in the data science community and have a lot of built-in functions and structures that are useful for data manipulation, analysis, and visualization. Additionally, they are well-documented, have a large user base, and are actively maintained and updated, which makes them a reliable and robust choice for data science projects.

1. **The Effectiveness of Different Models:**

In this section, we will evaluate the overall effectiveness of the different models applied in the proof of concept. The models that were used in this proof of concept are KNeighborsClassifier, DecisionTreeClassifier, GaussianNB, and RandomForestClassifier. From the results obtained in the previous sections, it is clear that the DecisionTreeClassifier and RandomForestClassifier models performed the best in terms of overall performance. These models achieved high accuracy, precision, recall, and F1-score, indicating that they were able to accurately classify the data and minimize false positives and false negatives. The KNeighborsClassifier model also performed well, however, it had a lower recall and f1-score than the other models. The GaussianNB model performed poorly in comparison to the other models, with lower accuracy, precision, recall, and F1-score. This can be attributed to the assumption of the algorithm which is that the features are independent of each other.

It is noted that the Random Forest Classifier producing results that are more robust and less susceptible to overfitting compared to a single decision tree. Additionally, the classifier is trained on the same training data at the start of each iteration, leading to consistent predictions and identical evaluation metrics for each iteration for the rest of the models.

Overall, the DecisionTreeClassifier and RandomForestClassifier models were the most effective in terms of overall performance, while the KNeighborsClassifier and the GaussianNB model performed poorly in comparison to the other models. These results indicate that the DecisionTreeClassifier and RandomForestClassifier models are suitable for classifying customer complaints in a telecom company and could potentially be used to improve customer satisfaction in the future.

1. **Recommendations:**

Based on the results of the proof of concept, we recommend that the company consider using the DecisionTreeClassifier and RandomForestClassifier models for classifying customer complaints in the future. These models performed the best in terms of overall performance, achieving high accuracy, precision, recall, and F1-score. They were able to accurately classify the data and minimize false positives and false negatives. Additionally, the Random Forest Classifier utilizes an ensemble method that involves multiple decision trees, each trained on a randomly selected subset of the data, resulting in independent predictions and diverse evaluation metrics for each iteration, making it more robust and less susceptible to overfitting.

It is also recommended that the company consider using the Pandas DataFrame and NumPy array data structures for handling and analyzing large datasets in the future. These structures are efficient and provide a wide range of functionality for data manipulation, analysis, and visualization.

Furthermore, it is also recommended to use libraries such as Pandas, Numpy, Scikit-learn and Matplotlib for data preprocessing, modeling, and evaluation in future projects as they are widely used in the data science community and have a lot of built-in functions and modules that are useful for data manipulation, analysis, and visualization. Additionally, they are well-documented, have a large user base, and are actively maintained and updated, making them a reliable and robust choice for data science projects.

It is also recommended that the company invest in additional data analysis and data visualization tools to better understand customer complaints and improve customer satisfaction in the future.

1. **Conclusion:**

In conclusion, this technical document has presented a proof of concept for the application of data science in a telecom company using the complaints data provided in the dataset. The goal of this project was to generate several models using appropriate libraries and evaluate them using different evaluation measures. Through the use of various data structures and libraries such as Pandas, Numpy, and Scikit-learn, we were able to import, clean, and analyze the data, as well as train and assess various models effectively. The performance of the models was evaluated using measures such as accuracy, precision, recall, and F1-score, and it was found that the DecisionTreeClassifier and RandomForestClassifier models performed the best. The results of this proof of concept highlight the importance of data science in comprehending consumer complaints and potentially enhancing customer happiness and satisfaction in the future. Overall, this project demonstrates the potential of data science in the telecom industry and its ability to improve customer service and satisfaction.

***References***

ActivateState (2020) *What Is Pandas In Python? Everything You Need To Know, ActivateState.com.* Available at: <https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/>

ActiveState (2022) *What is Scikit-Learn In Python?, ActiveState.* Available at: [https://www.activestate.com/resources/quick-reads/what-is-scikit-learn-in-python/](https://www.activestate.com/resources/quick-reads/what-is-scikit-learn-in-python/%20)

Custer, C. (2020) *15 Python Libraries for Data Science You Should Know , Dataquest.* Available at: <https://www.dataquest.io/blog/15-python-libraries-for-data-science/>

Datagy (2022) *Decision Tree Classifier with Sklearn in Python, datagy.* Available at: <https://datagy.io/sklearn-decision-tree-classifier/>

GeeksforGeeks (2019a) *Python | Pandas DataFrame, GeeksforGeeks.* Available at: <https://www.geeksforgeeks.org/python-pandas-dataframe/>

GeeksforGeeks (2019b) *Python | Pandas Series, GeeksforGeeks.* Available at: <https://www.geeksforgeeks.org/python-pandas-series/>

GeeksforGeeks (2020) *NumPy array in Python, GeeksforGeeks.* Available at: <https://www.geeksforgeeks.org/numpy-array-in-python/>

GeeksforGeeks (2022) *NumPy in Python | Set 1 (Introduction), GeeksforGeeks.* Available at: <https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/>

Huneycutt, J. (2018) *Implementing a Random Forest Classification Model in Python, Medium.* Available at: [https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652](https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652%20)

Hunter, J. et al. (2020) *Matplotlib: Visualization with Python, Abgerufen.* Available at: <https://matplotlib.org/>

IBM (2022) What is the k-nearest neighbors algorithm?, IBM. Available at: [https://www.ibm.com/topics/knn](https://www.ibm.com/topics/knn%20)

Krishna Madan (2022) *What Is Matplotlib In Python?, ActiveState.* Available at: <https://www.scaler.com/topics/matplotlib/matplotlib-in-python/>

Majumdar, P. (2021) *Scikit Learn - Introduction, TutorialsPoint.* Available at: <https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm>

Payam Refaeilzadeh, L.T.& H.L. (2010) *Cross-Validation, SpringerLink.* Available at: <https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_565>

Plotly (2009) *Python Graphing Library, Plotly, Plotly.* Available at: <https://plot.ly/python/>

Rastogi, K. (2021) *Guide for Data Visualization With Bokeh Python Library, Analytics Vidhya.* Available at: <https://www.analyticsvidhya.com/blog/2021/09/guide-for-data-visualization-with-bokeh-python-library/>

Ray, S. (2021) *Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples, Analytics Vidhya.* Available at: <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>

Vats, R. (2021) *Gaussian Naive Bayes: what you need to know?, upGrad.* Available at: <https://www.upgrad.com/blog/gaussian-naive-bayes/>

Waskom, M. (2020) *An introduction to seaborn, Pydata Seaborn.* Available at: <http://seaborn.pydata.org/introduction.html>

Wilkinson, P. (2022) *8 Data Structures every Data Scientist Should Know, Towards Data Science.* Available at: <https://towardsdatascience.com/eight-data-structures-every-data-scientist-should-know-d178159df252>