/Users/apple/Library/Containers/com.microsoft.Outlook/Data/Library/Caches/Signatures/signature_2105988401

**Data Science Programming**

10204281

**Assignment Title**

Data preprocessing, modeling, and reporting

Assignment 1

**Submitted to**

Dr. Sinan Kamal

**Submitted by**

Yousef AbuAli

**2022/2023(Fall Semester)**

**Technical Documentation**

**Part2: (report)**

1. Introduction:
   1. Data Structures:

Data structures are a way of organizing as well as storing data in a computer program. It helps in modifying and accessing these data more effectively. Each data structure has its strengths and weaknesses. using any of them depends on the requirements of the program. Some of the most common data structures in python are lists, tuples, sets, and dictionaries.

1. Lists:

* lists are built-in data structures in python. It is used to store data in an ordered way. different types of data can be stored in a list. In addition, lists are mutable, which means that the element can be changed at any time after creating the list.
* Lists are created using a square bracket and separating the items with commas.
* Lists have built-in methods that are used to modify them, such as insert (), remove (), pop (), and sort ().
* Lists are used for different purposes, such as storing ordered items, accessing items by their index, using slicing, modifying items, and changing the items contained in it using a loop.
* Lists can consume a lot of memory; therefore, it’s better not to use them with large data set.

1. Tuples:

* Tuples are built-in data structures in python. They are used to store an ordered collection of items, and they also contain different types of data. but unlike lists, they are immutable, therefore the elements cannot be changed after the tuple is created.
* Tuples are created using parentheses and separating the items with commas.
* Tuples has a variety of applications such as storing items that can be used as keys for dictionaries (because keys aren’t supposed to be modified), storing items that can be used as records in a database, as well as storing multiple values in a single variable.
* Tuples are good when storing items that will not be changed, and they are more memory efficient than lists because they are immutable. However, because they are immutable, it can be less convenient in case a modification on items is needed.

1. Sets:

* Sets are built-in data structures that can be used to store unordered collections of items. Sets are implemented as hash tables, so it can be very fast for checking whether items are a member of the set or not, adding items to a set, or removing items from a set.
* Sets are created using curly braces or using the set () constructor to change a list for example into a set. Ex. Set1=set([List]).
* Sets are used for removing duplicates from a collection of items and performing mathematical operations such as union, intersection, and difference.
* Sets are good to store a collection of unique items and to perform math operations. As well as they are more memory efficient than lists when there is a large number of duplicates. On the other hand, sets are unordered, and a list or tuple should be used if the order is needed.

1. Dictionaries:

* Dictionaries are built-in data structures that store a collection of key-value pairs. Dictionaries are also implemented as hash tables; therefore, they are very fast to look up the values by their key.
* Dictionaries are created using curly braces with key-value pairs separated by colons.
* Dictionaries are used for storing items that are associated with unique keys, looking up the values by their key, as well as modifying or adding new key-value pairs.
* Dictionaries are flexible that you can store a collection of items that are associated with a unique key, look up the values by their key, check if an item is a member of a dictionary, adding or remove items from a dictionary. However, they are unordered, and a list or tuple should be used if the order is needed. In addition, they are not indexable, so they can’t be accessed by index, they are only accessible by keys.
* Summary:
* sets and dictionaries are faster for checking, adding, and removing items because they store the data in a special way called hash tables.
* Tuples and lists are slower for checking, adding, and removing items. However, they maintain the order of the items.
* When there are a lot of items and checking, adding, or removing items needs to be done, a set or dictionary should be used. If the order of the items is important, a list or tuple should be used.
  1. Common libraries:

Python includes a huge number of libraries that provide capabilities even beyond the language's built-in capabilities. These libraries are pre-written code sets that can be easily imported and utilized in Python. They can be used to do a wide range of tasks, including data manipulation, scientific computing, machine learning, web building, and others.

Some of the most popular libraries in python include (NumPy, Pandas, and Scikit learn):

1. NumPy:

* NumPy is a Python library that helps with huge and multi-dimensional arrays and matrices.
* It includes a variety of tools for performing mathematical operations on arrays and matrices, making it helpful for scientific and numerical computation.
* It includes an array object called an array, which can store big arrays of data and execute quick operations on them.
* It is frequently used with other libraries such as Pandas and Scikit-learn for data manipulation, analysis, and machine learning.

1. Pandas:

* Pandas is a Python package that aids with data handling and manipulation by providing simple data structures and data analysis capabilities.
* It is developed on NumPy and allows you to work with structured data.
* It is widely used in data science and machine learning projects.
* It has two major data structures: Series (1-dimensional) and DataFrame (2-dimensional), which allow for operations like merging, grouping, restructuring, and much more.
* It is important in data cleaning, and data visualization.
* It can be used for:
* Read data from a CSV file and store it in a DataFrame
* Select specific columns or rows from a DataFrame
* Merge multiple data frames based on common columns.
* Handle missing data and remove duplicates.

1. Scikit learn:

* Scikit-learn, is a machine learning library in Python. It provides a wide range of supervised and unsupervised learning algorithms in Python.
* Scikit-learn provides machine learning models and makes it easy to switch between them.
* It includes algorithms for classification, regression, clustering, and pre-processing.
* It provides a lot of functionalities, such as:
* Data pre-processing
* Model selection
* Evaluation metrics
  1. Plotting and visualization libraries

There are several libraries in Python to create visualizations and plots. Such as matplotlib.

Matplotlib is one of the most common used libraries for data visualization.

It gives a lot of visualizations for data, such as line plots, bar chart, radar chart, and others.

1. Experiments:
   1. Programming languages and tool

I am using python language in this project using colab.

* python is a high-level programming language that is known for its simplicity and it’s easy to read syntax. It’s also known for it’s large set of libraries that provides wide range of built-in functions. It is used for artificial intelligence, machine learning and data science.
* Colab is a free online notebook that runs in the cloud and allows users to write python codes and execute them. It is easy to use and allows users to import as well as export data. it allows access to various libraries, such as scikit-learn.
  1. Loading and preparing data

for loading the data ‘file reading’ I used the function read\_csv() from pandas library and stored the result in a pandas data frame. The reason I used read\_csv is due to the ease of use (as it reads a csv file and convert in to a DataFrame), handling large datasets, and flexibility due to the use of DataFrames that allows performing data manipulation tasks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Column/step Name** | **Description** | **Justification** |
|  | RESOLUTION | I used drop() method from pandas library to drop the 'RESOLUTION' feature. | It contained a lot of missing values |
|  | RESOLUTION\_DESCRIPTION | I used drop() method from pandas library to drop the 'RESOLUTION\_DESCRIPTION ' feature. | It contained a lot of missing values |
|  | CASE\_DESC | I used drop() method from pandas library to drop the  ' CASE\_DESC ' feature. | It contained a lot of missing values |
|  | CALLBACK\_MECHANISM | I used drop() method from pandas library to drop the  'CALLBACK\_MECHANISM' feature. | It contained a lot of missing values |
|  | OPEN\_USER | I used drop() method from pandas library to drop the  ' OPEN\_USER ' feature. | It is unnecessary feature, and it gave better results when dropping it. |
|  | CLOSE\_USER | I used drop() method from pandas library to drop the  ' CLOSE\_USER ' feature. | It is unnecessary feature, and it gave better results when dropping it. |
|  | CASE\_ID | I used drop() method from pandas library to drop the  ' CASE\_ID ' feature. | It is unnecessary feature, and it gave better results when dropping it. |
|  | OFFER\_NAME | I used fillna() method from pandas library to fill the missing values in 'OFFER\_NAME' feature with ‘unknown’. | It is an important feature but there is no way to fill them manually. |
|  | CUSTOMER\_GROUP | I used fillna() method from pandas library to fill the missing values in 'CUSTOMER\_GROUP' feature with ‘unknown’. | It is an important feature but there is no way to fill them manually. |
|  | CLOSE\_GROUP | I used fillna() method from pandas library to fill the missing values in 'CLOSE\_GROUP' feature with ‘unknown’. | This feature might be important and it can’t be filled manually |
|  | OPEN\_GR | I used fillna() method from pandas library to fill the missing values in 'OPEN\_GR' feature with ‘unknown’. | This feature might be important and it can’t be filled manually |
|  | CLOSE\_DATE | I used fillna() method from pandas library to fill the missing values in 'OPEN\_GR' feature with ‘still in progress’. | This feature is important but the missing values in it are due to that the case in still in progress. |
|  | AGE\_BRACKET | I used fillna() method and mean() function to fill the missing values in AGE\_BRACKET. | I will fill AGE\_BRACKET with the mean to have better result than fill it with 'Still in progress' |
|  | ESCALATED\_GROUP | I used fillna() method from pandas library to fill the missing values in  ' ESCALATED\_GROUP ' feature with ‘unknown’. | This feature might be important, and it can’t be filled manually |

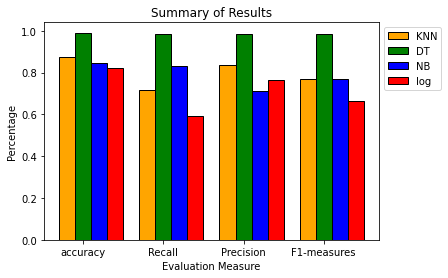
* 1. Approaches:

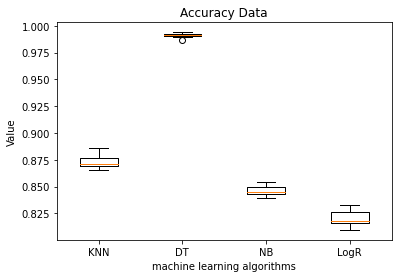
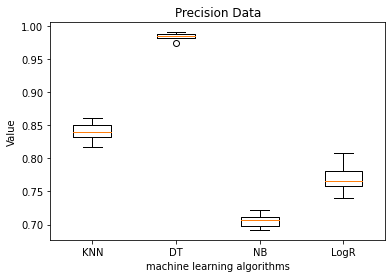
|  |  |  |
| --- | --- | --- |
| **Approach no.** | **Name** | **Description** |
|  | K-Nearest Neighbors (KNN) with n set to 3 | A supervised learning algorithm. The main idea of KNN is to find a certain number (K) of training examples that are closes to a new data point, by using the most common class in the K training examples to predict the new data point. And it is used for classification and regression  I used the data set in this algorithm where **y** was the dependent variable ‘product’ and **x** was independent variable, which means the rest of the data set without **y.**  I tried KNN algorithm twice. The first time (n) was set to 3 and the I took a sample size of 0.3 of the data. the results were really good, and they were better than the second attempt. |
|  | K-Nearest Neighbors (KNN) with n set to 5 | In this algorithm I also used the data set in this algorithm where **y** was the dependent variable ‘product’ and **x** was independent variable, which means the rest of the data set without **y.**  but this time I tried to set the number of n to 5 and I kept the sample size 0.3. the results were actually less than the first attempt. |
|  | decision tree | The decision tree algorithm is a machine learning algorithm that is used to predict new data points based on the obtained features of a certain data. this is done by creating a model that looks like a tree. It starts as a single node and splits the data into subset based on features.  In my code I chose **y as** the dependent variable ‘product’ and **x as** the rest of the data set without **y.** I set the random state in the decision tree classifier as 100.  The result was found by applying the code 10 times and taking the average of the total results, in order to get the best possible outcomes.  The result was actually really good, and it gave the best results compared to other algorithms. |
|  | Naive Bayes | Naive Bayes is a machine learning algorithm based on the baye’s theorem. It assumes that all features are dependent from each other. it is used for classification problems such as spam filtering.  I applied this algorithm on the dataset by using **y as** the dependent variable ‘product’ and **x as** the rest of the dataset.  The results were also good, and they came in between the rest of the algorithms. |
|  | logistic regression | Logistic Regression is a method that is used to fit a regression model when we want to predict a binary variable such as (1/0). It is an efficient and simple algorithm to handle large amounts of data.it is also used to determine the relative importance of each feature.  I used this algorithm with my dataset because the dependent variable is 0/1 after using the label encoder.  I applied this algorithm on my dataset, and it gave the wors result compared to the other algorithms. |

1. Results:
   1. Comparing different models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Approach no.** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| 1. K-Nearest Neighbors (KNN) with n set to 3 | 0.893 | 0.855 | 0.77 | 0.81 |
| 1. Decision Tree | **0.991** | **0.985** | **0.984** | **0.984** |
| 1. Naive Bayes | 0.847 | **0.712** | 0.832 | 0.767 |
| 1. Logistic Regression | **0.822** | 0.763 | **0.590** | **0.665** |

* 1. Charts:

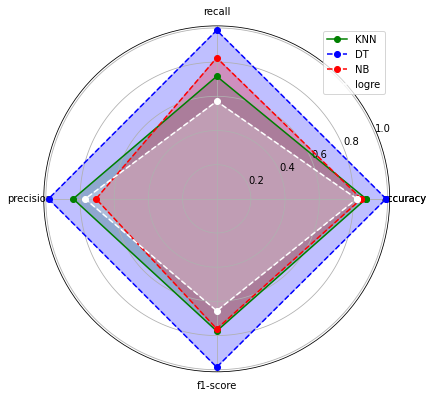


Graphical user interface

Description automatically generated with low confidence A picture containing diagram

Description automatically generated



* 1. Analysis of the results:

Now I will be discussing the results of comparing multiple machine-learing algorithm such as, K-Nearest Neighbors (KNN), Decision Tree, Naïve bayes, logistic regression. Relating to the Accuracy, Precision, Recall, and F1-score of each one of them.

1. Starting with the bar chart.

* The bar char is easy to read and understand, and it’s easy to compare the different categories in it by just looking on the height of the bars.
* It is clear that the decision tree algorithm showed the best results of the different evaluation methods. The decision tree had around **99%** for the Accuracy, and approximately **98%** for Recall, Precision, and f1-score. Which showed the best results compared to other algorithms.
* Moving on to the K-Nearest Neighbors (KNN) that showed the second-highest percentages in the evaluation methods. It showed an Accuracy of 89%, and a Recall of around 77%, Precision of approximately 85%, and around 81% for the F1-score.
* The third highest percentage was the Naïve bayes it had an Accuracy of almost 85%, a recall of 83%, a precision of **71%** which was the worst percentage in all of the precision column, and F1-score of 76%.
* In last was the Logistic Regression, which had the least percentages in all of the measures except for the precision. As it got around **82%** for the KNN, 76% for the precision, **59%** for the Recall, and around **66%** for the F1-score.

1. Another plot is the box plot.
   * + In all of the four plots the decision tree had the highest maximum and the highest minimum, as it remained over 95%.
     + In the accuracy and precision, The KNN showed a higher range of (69% to 86%) than the naïve bayes and Logistic Regression.
     + But in the precision box plot the Logistic Regression had a higher range of almost (**75%** to **80%**) than the naïve bayes, although it remains the least range in all of the other box plots.
     + In the Recall box plot, the naïve bayes had a better range than the KNN and the Logistic Regression. It got around (**81%** to **85%).**
     + Finally, in the F1 box plot, the KNN and Naïve bayes had almost the same range of (**75%** to **80%**). Which was higher than the Logistic Regression that had a range of (**63%** to **70%**).
2. **Evaluation:**

**4.1- The choice of data structures:**

* loading the CSV file into a panda’s data frame:

this method is very useful because with the help of the panda’s library it provides a lot of powerful and easy tools for dealing with the data stored in the data frame. A few of the advantages of loading a CSV file into a panda’s data frame are:

1. handling the missing value, as panda’s library finds and handles the missing values, which helps in easily filling or dropping the missing values.
2. It helps in data cleaning by providing a range of data cleaning tools for removing duplicates, fixing data types, or even for handling outliers.
3. It helps in data manipulation, as it allows for filtering, sorting, and grouping data easily.
4. Helps in data visualizations as pandas works with other important libraries, such as matplotlib, that helps in creating easy visualization of the data.

Overall, using a pandas data frame to load data into it is a very convenient way of working with structured data, as it allows for manipulating, analyzing, and visualizing data easily using a lot of strong tools.

* Using lists to store the results of the evaluation results (Accuracy, recall, precision, F1) in each loop to use it later on in the plotting and visualization.

1. Using lists allows to easily track the performance of an algorithm over multiple iterations.
2. If the outcomes of each iteration of a loop gets stored in a list makes the analyzing of the results much easier.
3. Storing the outcomes of the loop in a list can help in visualizing the results and deliver them to other because of the clear and organized format.
4. You can use multiple functions with it such as append or remove to add items on it easier.

**4.2-** Selection of libraries:

* Using libraries with machine learning helps in implementing complex algorithms as well as models without the need to write the code from scratch. These libraries help in providing functions and classes to use it for training, testing, and evaluation models. In addition, these libraries have tools for data pre-processing, evaluating performance, and visualizations, which helps in saving time and improving the accuracy of the models. I used pandas and scikit-learn libraries.

1. Pandas

I used panda’s library to load, manipulate, and prepare data for model training and evaluation. For example, I used pandas’ data frame using read csv function to load the data in it, in order to have the ability to deal with the data more easily.

I also used it to create a data frame to collect the obtain results from the algorithms to have the ability of comparing different algorithms with each other, therefore it helped with the ease of visualization of the results to know which was better.

1. scikit-learn.

I chose this library because of its simplicity and because it is easy to use. Additionally, this library contains wide range of tools that helps in machine learning models, for example:

* evaluation metrics such as accuracy, recall, precision, and F1 score for classification.
* Learning algorithms such as k-Nearest Neighbors, Decision Tree, Naïve bayes, and Logistic Regression.

Also, scikit-learn works in a compatible way with other data types such as

NumPy arrays, pandas’ data frames.

1. Matplotlib:

I used matplotlib library for visualizing the (Accuracy, recall, precision, and F1 score results of each one of the machines learning algorithms, by using the lists that I created for each one of them to create a bar chart, box blot for each evaluation metrics for the four algorithms. Matplotlib gave me the ability to control every aspect of each plot such as axis labels, plot legends, colors, and titles.

**4.3- The effectiveness of different models:**

In the beginning, the Decision tree had given me the best results compared to the other machine learning models.

The great thing about the decision tree method is that it splits the data into subset based on it values and creates a structure to represent the relationships between the features and the dependent variable. And it is affective for datasets that contains a large number of features and samples.

The KNN on the other hand stores training data and wait for a new data point to use for predictions by finding the K-nearest neighbor of this data point and uses the most common class between the selected neighbors to predict, which is why it’s considered a lazy learner.

The KNN had the second-best results, although each time I tried to increase the number of (n), the results were getting worst. I think it is because when the value of n increases, the algorithm uses more data points, therefore it becomes more sensitive to noise and outliers in the data. therefore, the number of n should be chosen appropriately.

The Naïve Bayes and Logistic Regression algorithms came in third and fourth place respectively, in terms of results. I used in my program the gaussian naïve bayes.

The decision tree is better in handling missing data than KNN and Logistic Regression, but they can be able of overfitting if the size of the tree is too large, as it might give high accuracy but it can give wrong outcome with new data.

Moreover, the tree structure can be viewed and evaluated, therefore they are easier to explain than the KNN, Logistic Regression, and naïve Bayes algorithms.

In addition, the Decision tree is better than the naïve bayes in handling the continuous data, however, the naïve bayes algorithm can work better with features that are irrelevant as well as it can work with large number of features better.

In conclusion, the decision tree algorithm is very flexible and powerful as it can handle large number of data types. But it is able to overfiring. On the other hand, the other algorithms are easier and can work noise and outliers effectively.

**4.4- Recommendations:**

According to the visualizations and the results that were given by the different machine learning algorithms, I recommend that the company should use the Decision tree in their data science applications. The decision tree has given the best results compared to the other machine learning algorithms. The decision tree is very simple to understand and evaluate, which make it an amazing choice for a lot of applications. In addition, they are able to handle numerical as well as categorical data. Moreover, it is able to handle big amount of data and many features. Also, they can be used to identify what features of the dataset are the most important.

References:

<https://w3schools.in/python-data-science/introduction-to-numpy-library/>

<https://datagy.io/python-scikit-learn-introduction/>

<https://www.geeksforgeeks.org/understanding-logistic-regression/>

<https://www.datacourses.com/what-is-scikit-learn-2021/#:~:text=Scikit-learn%20is%20both%2C%20well-documented%20and%20straightforward%20to%20learn%2Fuse,model%20to%20your%20data%20as%20a%20high-level%20library>.

<https://www.studytonight.com/pandas/pros-and-cons-of-using-pandas#:~:text=Pandas%20provide%20its%20users%20with%20a%20huge%20list,the%20information%20they%20have%20available%20in%20their%20hands>.

<https://in.coursera.org/articles/decision-tree-machine-learning>