Machine learning project

Team members:

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Group** |
| Rajab Hassan Abd Al-Monem  **(Team leader)** |  | 1 |
| Ziad Ahmed Saeed |  | 2 |
| Yusuf Sameh Hussen Mohammed |  | 1 |
| Mohammed Mahmoud Mohammed |  | 2 |
| Zienab Abdelsamea Mohamed Ahmed |  | 2 |

***Introduction:***

* The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.
* The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

***What is the technique we want to do?***

***DECISION TREE***

***First of all we will start by Decision tree:***

* Import the libraries

Text

Description automatically generated

Here, we need pandas to deal with data as data frame , using numpy to apply mathematics in data , using matplotlib and seaborn to visualize the graphs and display the plots between the features , and we need sklearn to use the built in function to build the model of decision tree.

* Load the data

Text

Description automatically generated with medium confidence

We upload our data and separate it using semicolon to make it easier to deal with it and display it .

* Knowing the info of the data

The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

**Table

Description automatically generatedwe can notice that the data contain 11 features of type object and 5 features of type integer and 5 features of type float**

* Knowing the unique values of each column in the data

Table

Description automatically generated

The nunique() method returns the number of unique values for each column. By specifying the column axis ( axis='columns' ), the nunique() method searches column-wise and returns the number of unique values for each row.

* Knowing the statistics of each column in the data

Text

Description automatically generated with medium confidence

The describe() function computes a summary of statistics pertaining to the DataFrame columns. This function gives the mean, std and IQR values. And function excludes the character columns and given summary about numeric columns.

* Plot the histogram of each feature

Graphical user interface, chart

Description automatically generated

Chart, bar chart

Description automatically generated

Now we can see what need to be modify it by discretization.

* Knowing the label feature counts Graphical user interface, text

  Description automatically generated

When we see that we can understand that the data is imbalance, but decision tree can deal with it

***Clean the data:***

* Duplicates

***Graphical user interface, text, application, email

Description automatically generated***

* Collect the categoric features and transforming it into numerical

Graphical user interface, text, application

Description automatically generated

This function takes all the features of the data and check if that is category store it into list.

* Transforming it into numerical

***Graphical user interface, text, application, email

Description automatically generated***

This function takes each category type of feature and transform it into numeric one.

* Dealing with unknown

We need to Unknow the index of unknown values in each column

***Graphical user interface, text, application

Description automatically generated***

* Transforming the categoric into numeric

***Graphical user interface

Description automatically generated with medium confidence***

***The correlation:***

Graphical user interface, text

Description automatically generated

Chart, timeline

Description automatically generated

**So we can see that there columns have a strong corelated with others so we will deal it to make our data independent.**

Graphical user interface, application

Description automatically generated

**Now we dropped our highly strong correlated and make one of them represent the same meaning.**

* ***Splitting the data into training and testing datasets:***
* The data was split into 80/20 ratio where the training dataset was 80% of the original data and the testing dataset had the remaining 20%.

Graphical user interface, text

Description automatically generated

* The splitting of the data was done before dealing with the unknown values and discretization to minimize the amount of bias in the testing dataset.

***The attribute duration was measured in seconds which made its values significantly larger than other numerical attributes in the dataset so we decided to convert it from seconds to minutes***

Text

Description automatically generated

* ***Dealing with unknown values:***

Here we attempted to deal with unknown values in our dataset using various means.

* Randomizing the unknown where we will change its value to another value that exists within the same attribute.

For example, the attribute **[Job]** has an unknown value when it is randomized it will become one of the other possible values of this attribute

*e.g. (admin, services, housemaid, etc)*

* Removing the records containing unknown values.

***This is used in one of two scenarios:***

* When the unknown value count is low in comparison to the total records count but is still large enough to cause a variation in the values if randomized.

Text

Description automatically generated

* When the unknown value exists in two attributes at the same time in a single record.

This happens between the two attributes **[Housing]** and **[Loan]** where the unknown values of the two attributes are always present together (This was confirmed using crosstab)

Graphical user interface, text, application, website

Description automatically generated

Text

Description automatically generated with medium confidence

**Dropping an attribute that has an overwhelming amount of unknown values.**

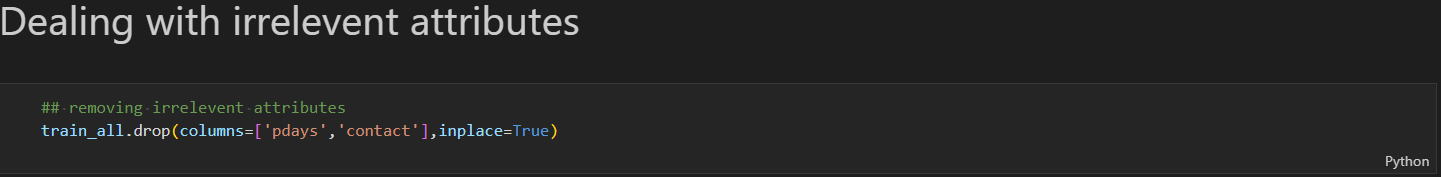
* This is present in the attribute **[Default]** since the unknown value count is above 30% of the entire attribute.

Graphical user interface, text

Description automatically generated

* ***Dropping irrelevant attributes:***

The attributes [pdays] and [contact] were irrelevant to our analysis so they were dropped.



* ***Discretization***

Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy. In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss. There are two forms of data discretization first is supervised discretization, and the second is unsupervised.

Supervised: refers to a method in which the class data is used. One of its famous methods is decision tree

Unsupervised: refers to a method depending upon the way which operation proceeds like K-means.

There are also common methods that is widely used like binning and binarization.

**In our data we used binning that because it’s more suitable.**

***Binning has 2 techniques***

* Binning by width: The simplest binning approach is to partition the range of the variable into k equal-width intervals
* Binning by frequency: In equal-frequency binning we divide the range [A, B] of the variable into intervals that contain (approximately) equal number of points

The columns which we have discretized are

[ age, duration, campaign, month, cons.price.index, cons.conf.idx , nr.employed ]

***NOTE***

we used panda.Cut for width and panda.qcut for frequency

* **We will talk about every one of them**

**AGE**

We used binning by width that because we want to segment people to reasonable three categorize **young, mature, and retired.**

Chart, bar chart

Description automatically generated

**Duration**

We used binning by frequency that we want 4 different categorize each one has the same or approximate number of calls that take different intervals of minutes

Table

Description automatically generated

**Campaign**

We used binning by width. We used 8 bins because we want to make various categorizes that include different types of people who made deposit in the last campaign

A picture containing chart

Description automatically generated

**Nr.employed**

We used binning by width. Since this column is quarterly rated and we have 10 months, so we want to make three categorizes to indicate three different intervals of number of employees

Chart, bar chart

Description automatically generated

**Month**

As I mentioned before we have 10 months, so let’s make it in 3 bins

Chart, bar chart

Description automatically generated

**Cons.price.index**

We wanted 2 categorize that represent good or bad

Graphical user interface, table

Description automatically generated

**Cons.conf.index**

The same as in Cons.price.index we wanted 2 categorize that represent good or bad

Chart

Description automatically generated with medium confidence

**First we build the model**

Graphical user interface, application

Description automatically generated with medium confidence

Plotting a confusion matrix on the test dataset to see how it is divided

Graphical user interface, text, application, email

Description automatically generated

Checking the initial accuracy of the model

Graphical user interface, application

Description automatically generated with medium confidence

We begin the tunning of the model to achieve higher accuracy through tunning the hyperparameters to obtain the optimal hyperparameter for our model

**In the preparation phase for cross validation:**

We use (cost complexity pruning path) from sklearn library to determine the different values of our hyperparameter which is Alpha in our case.

Graphical user interface, text, application

Description automatically generated

By plotting the accuracy of every alpha value we can determine which alpha value to use for cross validationGraphical user interface, application

Description automatically generated

Now we use 100-fold cross validation create 5 different training and testing datasets that are then used to train and test the tree.

***NOTE***

***We use 100-fold because we don't have tons of data...***

Graphical user interface, text, application

Description automatically generated

Now we repeat the previous step across all alpha values, but with only 10-fold cross validation to decrease the run time

Graphical user interface, text, application, email

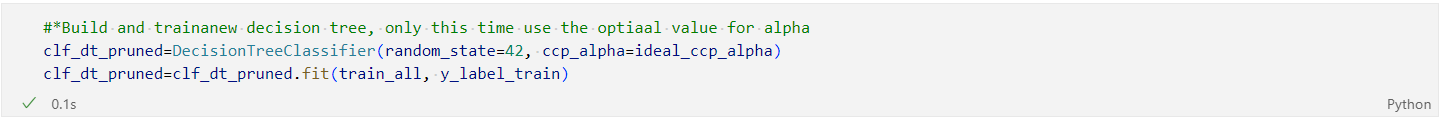
Description automatically generated

From the previous graph we can see that the best alpha value is between 0.0001 and 0.002, but we don’t know exactly the best value so we will put them into a list to get the alpha value with the highest accuracy.

Graphical user interface, text, application

Description automatically generated

Since now we have the best alpha value for our model, we will rebuild the model using that value.



Now we check the accuracy of the model after the tunning.

Background pattern

Description automatically generated with medium confidence

***We can see that the tunning of the hyperparameter increased the accuracy of our model significantly from 66.2% to 88.2%***

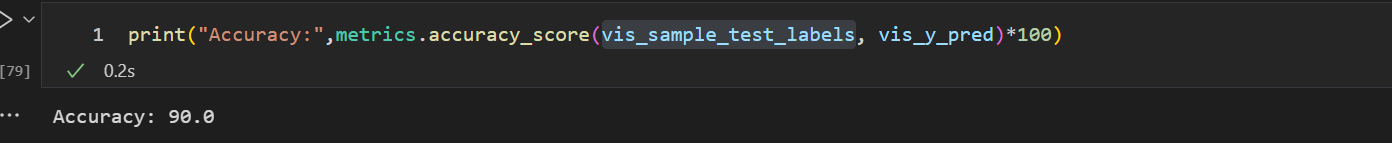
* Visualization

We took sample from data to make visualization and see accuracy

Text

Description automatically generated with medium confidence

***Accuracy was very good it is 90%***



Also, confusion matrix is good enough

Graphical user interface, text

Description automatically generated

***Now let’s make visualization to our tree***

A picture containing qr code

Description automatically generated

**Let’s see what happened to our sample after tunning**

Text

Description automatically generated

***The accuracy has increased from 90% to 91%***

**Let’s see final visualization**

Qr code

Description automatically generated

Note: since our data is imbalanced, we tried to use sampling techniques (oversampling and under sampling)

We used 3 different techniques in under sampling

1. Near miss with its three versions
2. NCL
3. ENN

In oversampling we used smote technique

All these techniques got low accuracy and very bad confusion matrix

So we prefer to use data as it does

# *NAÏVE BAYES CLASSIFIER*

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

**Naïve base make some assumption:**

The fundamental Naive Bayes assumption is that each feature makes an:

* independent
* equal
* **Bayes’ Theorem**

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred

**There are 5 algorithms in naïve bayes**

1. ***Gaussian Naive Bayes classifier***
2. ***Multinomaial Naive Bayes classifier***
3. ***Complement Naive Bayes classifier***
4. ***Bernoulli Naive Bayes classifier***
5. ***Categorical Naive Bayes classifier***

We have chosen **Categorical** naïve bayes because the algorithm made for categorical data

* **Categorical** implements the categorical naive Bayes algorithm for categorically distributed data. It assumes that each feature, which is described by the index i, has its own categorical distribution.

For each feature i in the training set X, categoricalNB estimates a categorical distribution for each feature i of X conditioned on the class y. The index set of the samples is defined as J={1,..,m} with m as the number of samples.

***Since our data has categorical attributes and we made discretization on numerical attributes to convert them into discrete categorical***

**Note**

**preprocessing here is just same in Decision Tree, but we made all preprocessing on the whole data**

* **We made a visualization to see the shape of distribution.**

Graphical user interface, text

Description automatically generated

* **Splitting data into train and test data**

Graphical user interface, text, website

Description automatically generated

**Now we are ready for applying the algorithm**

Graphical user interface, application

Description automatically generated

**Now let’s see the accuracy and confusion matrix**

Graphical user interface, text

Description automatically generated

***Accuracy is 89.3%***

*K-nearest neighbors (KNN)*

***Introduction:***

* The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.
* The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).
* Import the libraries

Text

Description automatically generated

Here, we need pandas to deal with data as data frame, using numpy to apply mathematics in data , using matplotlib and seaborn to visualize the graphs and display the plots between the features , and we need sklearn to use the built in function to build the model of KNN.

* Load the data

Graphical user interface, application

Description automatically generated

We upload our data and separate it using semicolon to make it easier to deal with it and display it .

* Knowing the info of the data

The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

Graphical user interface

Description automatically generated with low confidence

**we can notice that the data contain 11 features of type object and 5 features of type integer and 5 features of type float**

***Correlation:***

We use correlation matrix to know the relation between the attributes, so we want to know that the highly strong correlated with dependent features to remove them

Graphical user interface

Description automatically generated

Then we make it in a heat map to visualize them

Chart, waterfall chart

Description automatically generated

So we remove the above 0.8 correlated

Graphical user interface, application

Description automatically generated

* Dealing with data

We have been transforming the label attribute to numeric 0 for no and yes refers to yes

* Remove irrelevant attributes

Graphical user interface, text, application

Description automatically generated

* Converting data type

We need to convert categorical to numerical to use KNN we just use get\_dummies() which is used for data manipulation. It converts categorical data into dummy or indicator variables

Text

Description automatically generated with medium confidence

* We made Function to calculate the accuracy and print it for the model

Text

Description automatically generated

*Splitting the data into training and testing datasets:*

The data was split into a 80/20 ratio where the training dataset was 80% of the original data and the testing dataset had the remaining 20%.

Graphical user interface, text, application

Description automatically generated

***Randomizing unknown values:***

Since we used the **‘get\_dummies’** the categorical attributes became in the shape of a **sparse matrix**, and it doesn't make sense to keep track of unknown columns,

so we want to randomize the unknown values.

This function does exactly that, it removes the unknown column of the sparse matrix and gives its value to a random column of the same attribute.

For example, we will remove **'job\_unknown'** and give its value to a

random column from the sparse matrix of unique values of the attribute **'job'**

Text

Description automatically generated

***Scaling the numerical attributes:***

We are going to scale the numerical attributes so they all have the same weight when calculating nearest neighbor.

This will be done on both the training and testing dataset so it matters when calculating the distance.

**First:**

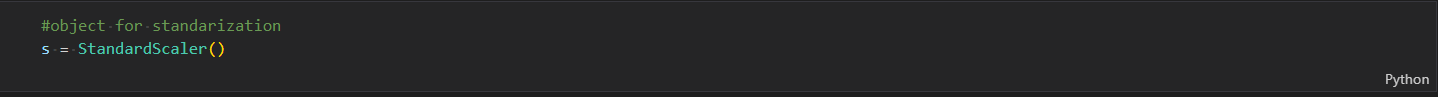
Splitting the categorical attributes from the numerical ones.

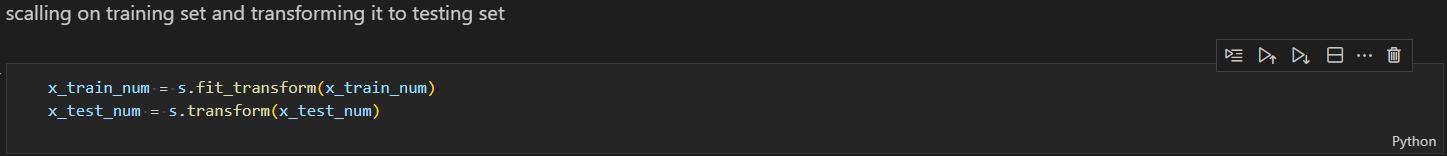
Text

Description automatically generated

**Second:**

The scaling is done using the Standard Scaler object.





**Third:**

The scaled numerical and categorical attributes are merged together in both the training and testing datasets.

Graphical user interface, text, application

Description automatically generated

***Tuning the hyperparameter (K):***

We try different K-Values to see which one has the most accuracy and least error rate for our dataset.

Text

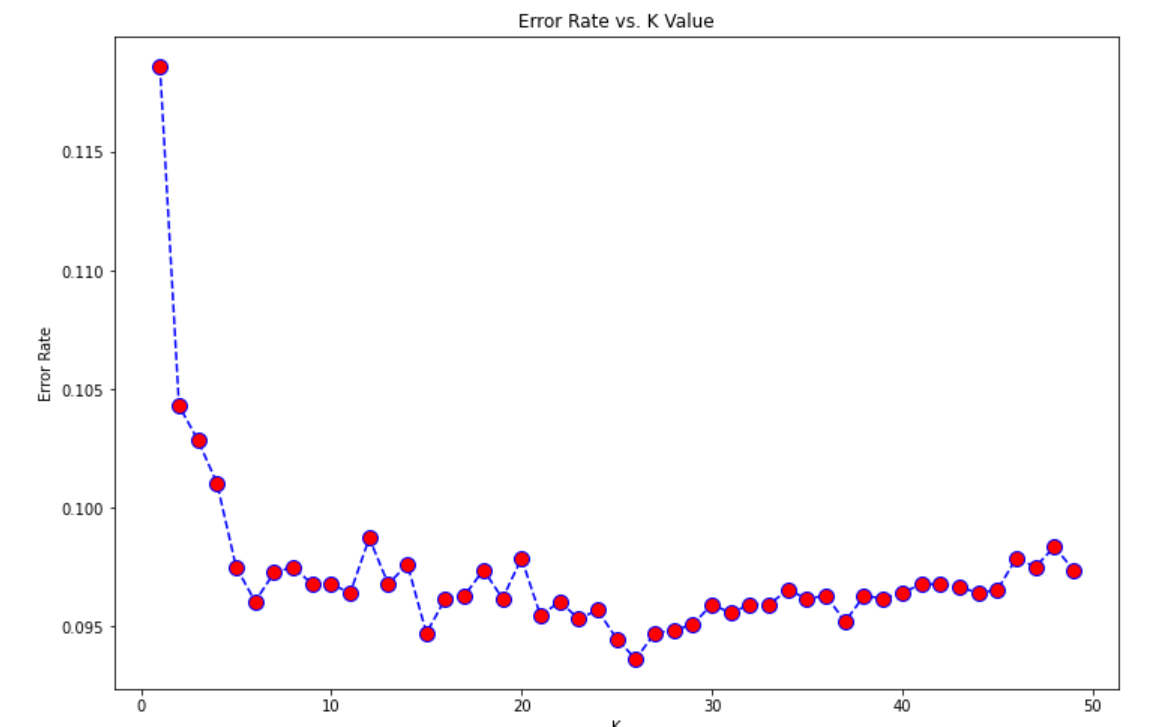
Description automatically generated

Text

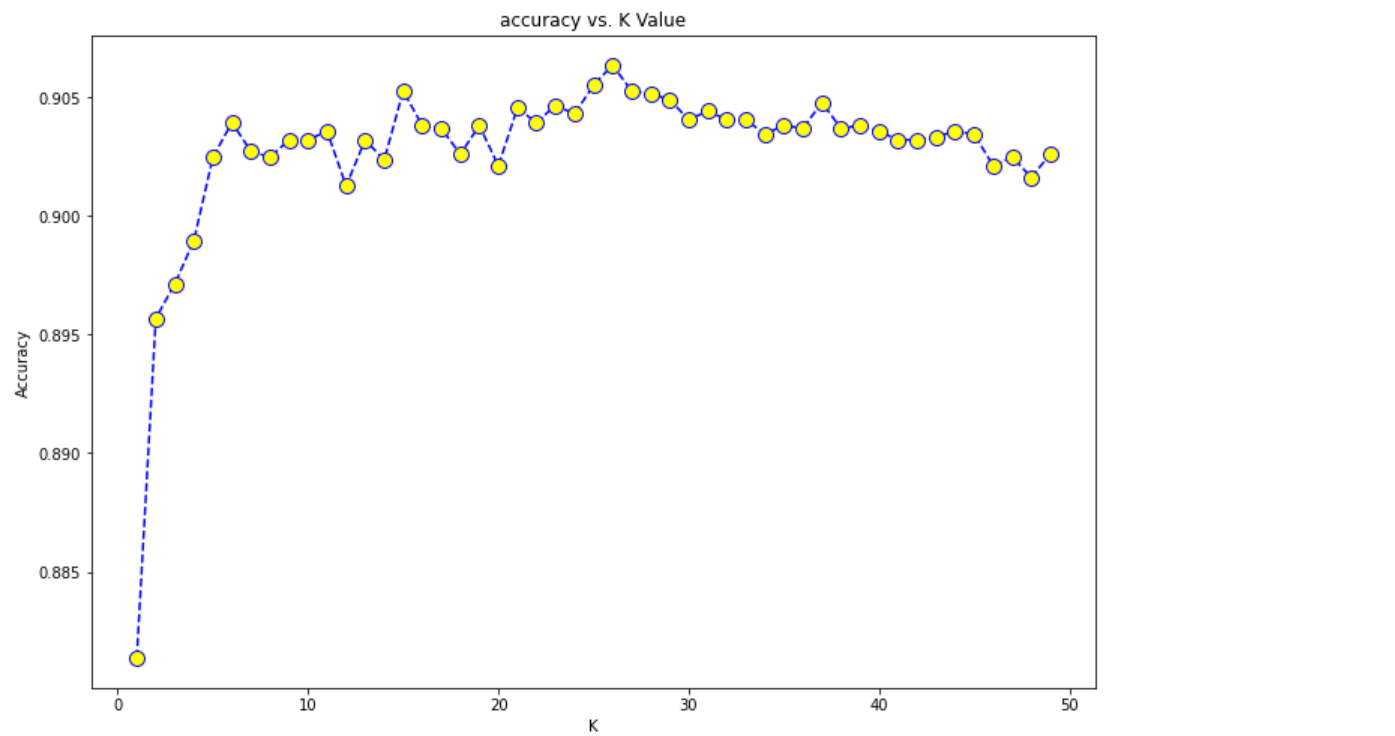
Description automatically generated

Graphical user interface, application, Teams

Description automatically generated



Visualizing accuracy rate



***After getting the best K-Value:***

The modeling of the KNN is done using the K-Value that we obtained through our previous visualization.

Graphical user interface, text

Description automatically generated with medium confidence

***Accuracy becomes 90.6%***

Plotting a confusion matrix on the test dataset to see how it is divided

Text

Description automatically generated