**Group\_9\_Task 1**:

Data Correction, Fill the missing values and out of range values for columns.

**Approach:**

The first task here is to find the missing values and out of range values which are declared by ourselves and these missing values and out of range values are replaced with most similar rows. We could replace the missing data and out of range values based on the nearest neighbor algorithm which would check the suitable row from the column range given and replace the value.

**Dataset:** We have used the input dataset as ‘‘/user/kaggle/kaggle\_data/marketing\_campaign.csv’’ which is from the input datapath provided.

**Brief description of the dataset and predicting variable:**

The data set is about a marketing survey conducted on people who are visited a store in course of period. It has collected the data of the person which includes the birth year, annual income, marital status. Data set is also talks about how many products like meat, fruits, wine, fish and sweet that a person bought. Also, number of purchases whether it is a gold, store, or web purchase. Each person in the data set is allocated with the unique Id to avoid the redundancy.

Data also keep track of the number of times a person visited the store website, the complaints that are received and revenue of the person who visited the store. Here, in this data set the income of the person is the decision factor because store management will check the number of items that a person has purchased during course of time. Data can be used to do analysis on store’s frequent products and purchases made by the customers which will help the store management to keep the demand of those products high or low in feature.

The input data set marketing\_campaign.csv has total of 31 columns with 'Marital\_Status','Education' and 'Dt\_Customer’ as a string data type and ‘Income’ as double data type. Each column has more than 50 rows in the data set. For data correction, we have found that only Income column has the null values in the data set. So have chosen the income column to fill the values and out of range values.

**Input Columns:** Income, ID

**Output Column:** columns without null and out range values and casted to integer type

* First we have imported all the required libraries. We have read the input csv file and created a data frame for it.
* Then we have checked for the entire dataset to see which of the columns had null values from which we got income as output. Then imported the KNNImputer and defined the n\_neighbhor as 10, that is it would replace the necessary columns by checking 10 columns aroung for best suitable value.
* Then using StringIndexer we have converted all the columns with string values to double and later type casted it to integer for all the columns to be uniform.
* Then we have defined the out of range values for the column Income as (0,10000) and for ID as (1,1000) and replaced the out of range values with nearest neighbor and suitable value available. Then we have fir and transformed the data frame and wrote the output to a file which would be used as an input for the next tasks.

**Instructions to compile and run the program:**

**Imports Used:**

import pandas as pd

import numpy as np

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn import metrics

from pyspark import SparkContext, SparkConf

from pyspark.sql import functions as f

from pyspark.sql import SQLContext

from pyspark.sql.types import \*

from pyspark.ml.feature import StringIndexer

**Input:** As mentioned, we have used the input dataset as ‘‘/user/kaggle/kaggle\_data/marketing\_campaign.csv’’ which is from the input datapath provided csv’

**Output**: Output is written to the file (task\_1\_output.csv) and it will be saved to local (/home/username/ task\_1\_output.csv)

1. Open the cluster and login into it.

2. Upload the Group\_9\_Task\_1.py file into session storage(local or csh)

3. Command: spark-submit Group\_9\_Task\_1.py

**Discussion of Results:**

The main constraint of the given dataset is it should not contain any null values and the out of range values for the two columns Income and ID. We resolved these issues using KNNInmputer using the nearest neighbor algorithm and converted all the string values using StrinIndexer and Type cast it to integer. Dropped the unnecessary columns after appending the values obtained from KNNImputer.

**References:** [KNNImputer for Missing Value Imputation in Python using scikit-learn | DataScience+ (datascienceplus.com)](https://datascienceplus.com/knnimputer-for-missing-value-imputation-in-python-using-scikit-learn/#:~:text=KNNImputer%20for%20Missing%20Value%20Imputation%20in%20Python%20using,%3D%20KNNImputer%20%28n_neighbors%3D2%29%20Copy%203%20Impute%2FFill%20Missing%20Values)