Workshop_2

November 29, 2023

Mayank Baryal - 2358873

Task Set-I: DataFrame Reading and Writing.

Dataset: "bank.csv".

1. Load the provided dataset and import in pandas DataFrame.
2. Check info of the DataFrame and identify following:
3. columns with dtypes=object|
4. unique values of those columns.
5. check for the total number of null values in each column.
6. Drop all the columns with dtypes int and store in new DataFrame, also write the DataFrame in ".csv" with name "banknumericdata.csv"
7. Read "banknumericdata.csv" and Find the summary statistics.

[ ]:
```
# loading the provided datasheet
from google.colab import files
upload = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving bank.csv to bank.csv
```

[ ]:
```
# Task 1.1
# loading the datasheet in read mode
import pandas as pd
df = pd.read_csv('bank.csv')
```

[ ]:
```
# checking the loaded dataasheet
df.head()
```

[ ]:

	age	job	marital	education	default	balance	housing	loan	\
0	58	management	married	tertiary	no	2143	yes	no	
1	44	technician	single	secondary	no	29	yes	no	
2	33	entrepreneur	married	secondary	no	2	yes	yes	
3	47	blue-collar	married	unknown	no	1506	yes	no	
4	33	unknown	single	unknown	no	1	no	no	

```
   contact  day month  duration  campaign  pdays  previous poutcome   y
0  unknown    5   may       261         1     -1         0  unknown  no
1  unknown    5   may       151         1     -1         0  unknown  no
2  unknown    5   may        76         1     -1         0  unknown  no
3  unknown    5   may        92         1     -1         0  unknown  no
4  unknown    5   may       198         1     -1         0  unknown  no
```

[ ]:
```
# checking the info of the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

[ ]:
```
# Task 1.2.1
# Identifying columns with object datatype
dtype_object = df.select_dtypes(include=['object']).columns
dtype_object    # printing the result
```

[ ]:
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'y'],
      dtype='object')
```

[ ]:
```
# Task 1.2.2
# checking the unique values of the columns
```

```
unique_value = {col: df[col].unique() for col in dtype_object}
unique_value
```

[ ]:
```
{'job': array(['management', 'technician', 'entrepreneur', 'blue-collar',
        'unknown', 'retired', 'admin.', 'services', 'self-employed',
        'unemployed', 'housemaid', 'student'], dtype=object),
 'marital': array(['married', 'single', 'divorced'], dtype=object),
 'education': array(['tertiary', 'secondary', 'unknown', 'primary'],
dtype=object),
 'default': array(['no', 'yes'], dtype=object),
 'housing': array(['yes', 'no'], dtype=object),
 'loan': array(['no', 'yes'], dtype=object),
 'contact': array(['unknown', 'cellular', 'telephone'], dtype=object),
 'month': array(['may', 'jun', 'jul', 'aug', 'oct', 'nov', 'dec', 'jan', 'feb',
        'mar', 'apr', 'sep'], dtype=object),
 'poutcome': array(['unknown', 'failure', 'other', 'success'], dtype=object),
 'y': array(['no', 'yes'], dtype=object)}
```

[ ]:
```
# checking for null values
df.isnull()
```

[ ]:

	age	job	marital	education	default	balance	housing	loan	\
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
…	…	…	…	…	…	…	…	…	
45206	False	False	False	False	False	False	False	False	
45207	False	False	False	False	False	False	False	False	
45208	False	False	False	False	False	False	False	False	
45209	False	False	False	False	False	False	False	False	
45210	False	False	False	False	False	False	False	False	

	contact	day	month	duration	campaign	pdays	previous	poutcome	\
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
…	…	…	…	…	…	…	…	…	
45206	False	False	False	False	False	False	False	False	
45207	False	False	False	False	False	False	False	False	
45208	False	False	False	False	False	False	False	False	
45209	False	False	False	False	False	False	False	False	
45210	False	False	False	False	False	False	False	False	

```
           y
0      False
1      False
2      False
3      False
4      False
…      …
45206  False
45207  False
45208  False
45209  False
45210  False

[45211 rows x 17 columns]
```

[ ]:
```
# Task 1.2.3
# Finding the total number of null values
null_value = df.isnull().sum()
null_value
```

[ ]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

[ ]:
```
# checking data types other than object (in this datasheet case int data typse)
numeric_data = df.select_dtypes(exclude=['object'])
numeric_data
```

[ ]:
```
       age  balance  day  duration  campaign  pdays  previous
0       58     2143    5       261         1     -1         0
1       44       29    5       151         1     -1         0
2       33        2    5        76         1     -1         0
```

```
3       47     1506    5        92         1     -1         0
4       33        1    5       198         1     -1         0
…     …       …   …       …       …     …        …
45206   51      825   17       977         3     -1         0
45207   71     1729   17       456         2     -1         0
45208   72     5715   17      1127         5    184         3
45209   57      668   17       508         4     -1         0
45210   37     2971   17       361         2    188        11

[45211 rows x 7 columns]
```

[ ]:
```
# Task 1.3
# create new data sheet for the int types data
numeric_data.to_csv('banknumericdata.csv', index = False)
```

[ ]:
```
# reading the new created datasheet
bank_numeric_data = pd.read_csv('banknumericdata.csv')
bank_numeric_data
```

[ ]:
```
       age  balance  day  duration  campaign  pdays  previous
0       58     2143    5       261         1     -1         0
1       44       29    5       151         1     -1         0
2       33        2    5        76         1     -1         0
3       47     1506    5        92         1     -1         0
4       33        1    5       198         1     -1         0
…     …       …   …       …       …     …        …
45206   51      825   17       977         3     -1         0
45207   71     1729   17       456         2     -1         0
45208   72     5715   17      1127         5    184         3
45209   57      668   17       508         4     -1         0
45210   37     2971   17       361         2    188        11

[45211 rows x 7 columns]
```

[ ]:
```
# Task 1.4
# summarizing the information in the new datasheet
summary_stat = numeric_data.describe()
summary_stat
```

[ ]:
```
                age        balance           day      duration      campaign  \
count  45211.000000   45211.000000  45211.000000  45211.000000  45211.000000
mean      40.936210    1362.272058     15.806419    258.163080      2.763841
std       10.618762    3044.765829      8.322476    257.527812      3.098021
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      72.000000      8.000000    103.000000      1.000000
50%       39.000000     448.000000     16.000000    180.000000      2.000000
75%       48.000000    1428.000000     21.000000    319.000000      3.000000
```

```
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
std      100.128746      2.303441
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
```

Task Set-II: Data Imputations

Dataset: "medical_Student.csv".

1. Load the provided dataset and import in pandas DataFrame.
2. Check info of the DataFrame and identify column with missing (null) values.
3. For the column with missing values fill the values using various techniques we discussed above. Try to explain why did you select the particular methods for particular column.
4. Check for any duplicate values present in Dataset and do necessary to manage the duplicate items.

{Hint: dataset.duplicated.sum()}

[ ]:
```
# uploading data file
from google.colab import files
upload = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving medical_students_dataset.csv to medical_students_dataset.csv
```

[ ]:
```
# Task 2.1
# import pandas DataFram and reading the dataset file
import pandas as pd
df1 = pd.read_csv('medical_students_dataset.csv')
```

[ ]:
```
# checkiing the info of the data set
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Student ID      180000 non-null  float64
 1   Age             180000 non-null  float64
```

```
 2   Gender           180000 non-null  object
 3   Height           180000 non-null  float64
 4   Weight           180000 non-null  float64
 5   Blood Type       180000 non-null  object
 6   BMI              180000 non-null  float64
 7   Temperature      180000 non-null  float64
 8   Heart Rate       180000 non-null  float64
 9   Blood Pressure   180000 non-null  float64
 10  Cholesterol      180000 non-null  float64
 11  Diabetes         180000 non-null  object
 12  Smoking          180000 non-null  object
dtypes: float64(9), object(4)
memory usage: 19.8+ MB
```

[ ]:
```
# Solution 2.2
# checking for columns with missing values
missing_col = df1.columns[df1.isnull().any()]
missing_col
```

[ ]:
```
Index(['Student ID', 'Age', 'Gender', 'Height', 'Weight', 'Blood Type', 'BMI',
       'Temperature', 'Heart Rate', 'Blood Pressure', 'Cholesterol',
       'Diabetes', 'Smoking'],
      dtype='object')
```

[ ]:
```
# calculating the total number of null values
df1.isnull().sum()
```

[ ]:
```
Student ID        20000
Age               20000
Gender            20000
Height            20000
Weight            20000
Blood Type        20000
BMI               20000
Temperature       20000
Heart Rate        20000
Blood Pressure    20000
Cholesterol       20000
Diabetes          20000
Smoking           20000
dtype: int64
```

[ ]:
```
# Solution 2.3
# using mean as a way to fill null values in the age column
df1['Age'].fillna(df1['Age'].mean(), inplace = True)
```

[ ]:
```
# Solution 2.3
# using mode as a way to fill null values in the Gender column
df1['Gender'].fillna(df1['Gender'].mode()[0], inplace = True)
```

[ ]:
```
# Solution 2.3
# using median as a way to fill null values in the blood pressure column
df1['Blood Pressure'].fillna(df1['Blood Pressure'].median(), inplace = True)
```

[ ]:
```
# Solution 2.4
# checking for the duplicate data in the dataset
duplicate = df1[df1.duplicated()]
print("Number of duplicare rows:", len(duplicate))
```

```
Number of duplicare rows: 8547
```

[ ]:
```
# counting the total number of duplicate data
df1.duplicated().sum()
```

[ ]:
```
8547
```

[ ]:
```
# Solution 2.4
# deleting the duplicate datas
df1.drop_duplicates(inplace= True)
```

[ ]:
```
# checking the dataset
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 191453 entries, 0 to 199999
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Student ID      171573 non-null  float64
 1   Age             191453 non-null  float64
 2   Gender          191453 non-null  object 
 3   Height          171554 non-null  float64
 4   Weight          171559 non-null  float64
 5   Blood Type      171554 non-null  object 
 6   BMI             171562 non-null  float64
 7   Temperature     171554 non-null  float64
 8   Heart Rate      171563 non-null  float64
 9   Blood Pressure  191453 non-null  float64
 10  Cholesterol     171571 non-null  float64
 11  Diabetes        171555 non-null  object 
 12  Smoking         171569 non-null  object 
dtypes: float64(9), object(4)
memory usage: 20.4+ MB
```

[4]:
```
# importing the dataset file
from google.colab import files
upload = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving performance.csv to performance.csv
```

Task Set-III: Data Transformations

Transform variables according to the following instructions: Dataset: “performance.csv”.

1. ”School”, ”internet”, ”activities”, into binary: 0 or 1 (create new columns without overwriting the existing ones).
2. ”Medu”,”reason”, ”guardian”,”studytime”„ and ”health” into ordinal numbers based on the number cases in the data set (create news columns without overwriting the existing ones).
3. Convert column ”age” to interval datatype. i.e. Create a new column name category age whose values should be based on the frequency in the column ”age”, You can create categorical data with following interval.

   `interval1: [15-17];interval2: [18-20]; interval3: [21-all]`

4. Create a new column name passed (yes or no) whose values should be based on the values present in the G3 column (>= 8–yes, < –no).

[5]:
```
import pandas as pd
# reading the data set file
df2 = pd.read_csv('performance.csv')
```

[ ]:
```
# checking the column headers of the file
df2.columns
```

[ ]:
```
Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
       'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

[ ]:
```
# Task 3.1
# making a list of column name to change
col_transform = ["school", "activities", "internet"]
# using for loop to check for every data
for i in col_transform:
  # creating new column for the corresponding column name and adding the␣
 ↪changed value
  df2[i+"transformed"] = df2[i].map({sorted(df2[i].unique())[0]:1,␣
 ↪sorted(df2[i].unique())[1]:0})
```

```
# checking the new dataset with added columns
df2.head()
```

[ ]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  … \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home     other  …
2     GP   F   15       U     LE3       T     1     1  at_home     other  …
3     GP   F   15       U     GT3       T     4     2   health  services  …
4     GP   F   16       U     GT3       T     3     3    other     other  …

  Dalc Walc  health  absences  G1  G2  G3 schooltransformed  \
0    1    1       3         6   5   6   6                 1
1    1    1       3         4   5   5   6                 1
2    2    3       3        10   7   8  10                 1
3    1    1       5         2  15  14  15                 1
4    1    2       5         4   6  10  10                 1

  activitiestransformed internettransformed
0                     1                   1
1                     1                   0
2                     1                   0
3                     0                   0
4                     1                   1

[5 rows x 36 columns]
```

[ ]:
```
# Task 3.2
# changing the values of the defined columns into ordinal
df2['Medu_ordinal'] = pd.Categorical(df2['Medu'], categories = df2['Medu'].
 ↪unique(), ordered = True).codes
df2['Reason_ordinal'] = pd.Categorical(df2['reason'], categories =␣
 ↪df2['reason'].unique(), ordered = True).codes
df2['guardian_ordinal'] = pd.Categorical(df2['guardian'], categories =␣
 ↪df2['guardian'].unique(), ordered = True).codes
df2['Studytime_ordinal'] = pd.Categorical(df2['studytime'], categories =␣
 ↪df2['studytime'].unique(), ordered = True).codes
df2['Health_ordinal'] = pd.Categorical(df2['health'], categories =␣
 ↪df2['health'].unique(), ordered = True).codes
df2.head(2)
```

[ ]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob     Fjob  … \
0     GP   F   18       U     GT3       A     4     4  at_home  teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home    other  …

  G2 G3  schooltransformed  activitiestransformed  internettransformed  \
0  6  6                   1                      1                    1
1  5  6                   1                      1                    0
```

```
  Medu_ordinal Reason_ordinal guardian_ordinal Studytime_ordinal  \
0            0              0                0                  0
1            1              0                1                  0

  Health_ordinal
0              0
1              0

[2 rows x 41 columns]
```

[ ]:
```
# Task 3.3
# adding new column with values based on grouping of the previously defined␣
 ↪values
df2['category age'] =  pd.cut(df2['age'], [14, 17, 20, 23],␣
 ↪labels=("interval1", "interval2", "intervals"))
# checking data
df2.head(100)
```

[ ]:
```
   school sex  age address famsize Pstatus  Medu  Fedu      Mjob      Fjob  \
0      GP   F   18       U     GT3       A     4     4   at_home   teacher
1      GP   F   17       U     GT3       T     1     1   at_home     other
2      GP   F   15       U     LE3       T     1     1   at_home     other
3      GP   F   15       U     GT3       T     4     2    health  services
4      GP   F   16       U     GT3       T     3     3     other     other
..    ...  ..  ...     ...     ...     ...   ...   ...       ...       ...
95     GP   F   15       R     GT3       T     1     1   at_home     other
96     GP   M   16       R     GT3       T     4     3  services     other
97     GP   F   16       U     GT3       T     2     1     other     other
98     GP   F   16       U     GT3       T     4     4     other     other
99     GP   F   16       U     GT3       T     4     3     other   at_home

    ...  G3 schooltransformed  activitiestransformed  internettransformed  \
0   ...   6                  1                      1                    1
1   ...   6                  1                      1                    0
2   ...  10                  1                      1                    0
3   ...  15                  1                      0                    0
4   ...  10                  1                      1                    1
..  ...  ..                ...                    ...                  ...
95  ...  10                  1                      0                    0
96  ...  15                  1                      0                    0
97  ...  10                  1                      1                    1
98  ...  14                  1                      0                    0
99  ...   8                  1                      1                    0

    Medu_ordinal Reason_ordinal guardian_ordinal Studytime_ordinal  \
0              0              0                0                  0
```

```
1              1              0              1              0
2              1              1              0              0
3              0              2              0              1
4              2              2              1              0
..           …              …              …              …
95             1              2              0              3
96             0              3              0              2
97             3              0              0              0
98             0              3              0              2
99             0              0              0              1

   Health_ordinal category age
0               0    interval2
1               0    interval1
2               0    interval1
3               1    interval1
4               1    interval1
..            …          …
95              2    interval1
96              4    interval1
97              1    interval1
98              2    interval1
99              0    interval1

[100 rows x 42 columns]
```

[6]:
```
# Task 3.4
# adding new column with values based on certain condition of values of one of␣
 ↪the columns
df2['passed'] = pd.cut(df2['G3'], [0, 8, 21], right = False, labels = ['no',␣
 ↪'yes'])
# checking data
df2.head(2)
```

[6]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob     Fjob  …  \
0     GP   F   18       U     GT3       A     4     4  at_home  teacher  …
1     GP   F   17       U     GT3       T     1     1  at_home    other  …

  freetime goout  Dalc  Walc  health absences G1 G2 G3 passed
0        3     4     1     1       3        6  5  6  6     no
1        3     3     1     1       3        4  5  5  6     no

[2 rows x 34 columns]
```

[8]:
```
# creating new file for the data changes made
df2.to_csv("transformed_performance.csv", index = False)
```