Week 9 Deliverables

Group Name: Data Go

Team member's details :

Name:Yuheng Chen

Email: yuh363@gmail.com

Country: USA

College/Company:N/A

Specialization: Data Science

Name: Terry Chou

Email: techch26@gmail.com

Country: USA

College/Company: N/A

Specialization: Data Science

Name: Rishi Aluri

Email: rishialuri@gmail.com

Country: UK

College/Company: N/A

Specialization: Data Science

Name: Justine Pile

Email: justypile@gmail.com

Country: US

College/Company: N/A

Specialization: Data Science

Problem description

The objective of this project is to develop a predictive model for ABC Bank to determine the likelihood of customers subscribing to their term deposit product. By analyzing customer interactions with the bank and other financial institutions, the machine learning model will identify potential clients who are more likely to purchase the product. This will enable the bank to optimize their marketing efforts by focusing resources on customers with a higher probability of conversion, thus enhancing campaign efficiency and reducing costs. The project will assess model performance with and without using the "duration" feature while also addressing any data imbalance through suitable techniques.

Github Repo link

https://github.com/yuh39/Bank_Marketing_Group_Project/tree/main/docs/Week9

Data cleansing and transformation done on the data

Yuheng Chen – bank-full.csv:

1. Data type:

```
<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
```

shape: 45211 rows × 17 columns

2. Check missing values, I replace 'unknown' values with NaN values

```
## Check missing values
d1.isna().sum()
```

```
age              0
job            288
marital          0
education     1857
default          0
balance          0
housing          0
loan             0
contact      13020
day              0
month            0
duration         0
campaign         0
pdays            0
previous         0
y                0
dtype: int64
```

a. Separate the original data set to training and testing and using random forest model to predict the missing values
b. Replace only null values in the testing data from predictions and get final dataframe

```
array([['management', 'tertiary', nan],
       ['technician', 'secondary', nan],
       ['entrepreneur', 'secondary', nan],
       ...,
       ['management', nan, 'cellular'],
       ['student', nan, 'cellular'],
       [nan, nan, 'cellular']], dtype=object)
```

⇒

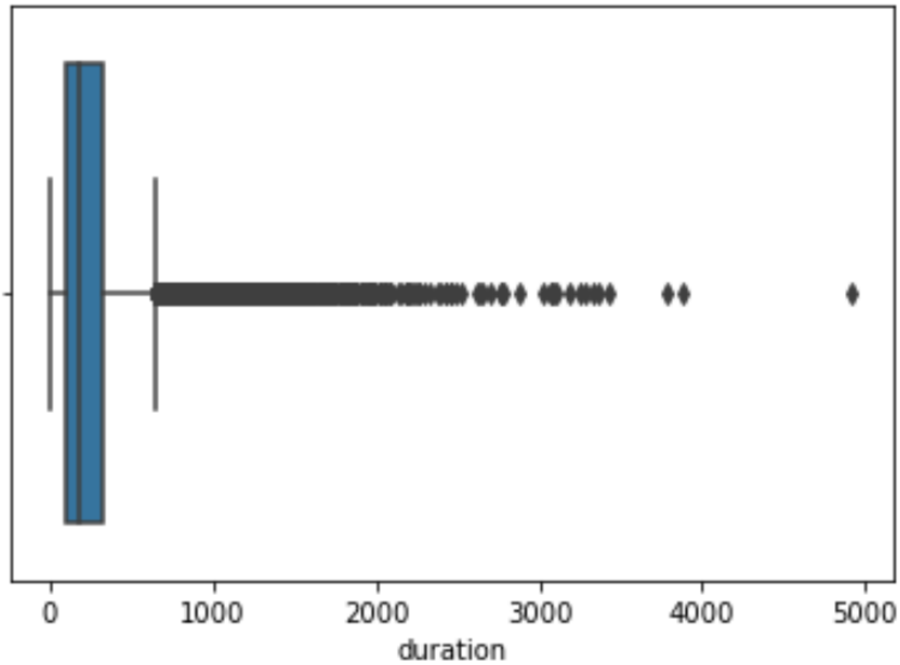
```
array([['management', 'tertiary', 'cellular'],
       ['technician', 'secondary', 'cellular'],
       ['entrepreneur', 'secondary', 'cellular'],
       ...,
       ['management', 'secondary', 'cellular'],
       ['student', 'tertiary', 'cellular'],
       ['management', 'secondary', 'cellular']], dtype=object)
```

c. Output data with no missing values

```
df = pd.concat([d1_testing, d1_training])
df.isna().sum()
```

```
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
y            0
dtype: int64
```

3. We have outliers in the duration column

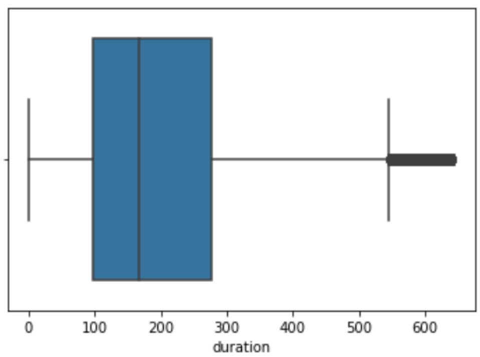


a. Remove outliers in duration column, the dataframe shape changed

```
Old Shape:  (45211, 16)
New Shape:  (41976, 18)
```

```
: sns.boxplot(x = d1['duration'])
```

```
: <AxesSubplot:xlabel='duration'>
```



bank-addtional.csv

Terry Chou – bank-addtional-full.csv

Data Info

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
```

Data Problems

- Roughly **26%** of rows contains one or more “unknown” value in this dataset:

```
# rows with "unknown" proportion
len(df_bank_add_full[(df_bank_add_full.job =='unknown')|(df_bank_add_full.marital =='unknown') |(df_bank_add_full.education =
```

```
0.25978440322424007
```

- Column “poutcome” contains **86%** of “nonexistent” value:

```
# poutcome "nonexistent" proportion
nonexistent_count = len(df_bank_add_full[(df_bank_add_full.poutcome =='nonexistent')])
nonexistent_count/len(df_bank_add_full)
```

```
0.8634310964358551
```

- The dataset has **12 duplicate rows**:

```
duplicate = df_bank_add_full[df_bank_add_full.duplicated()]
print("Duplicate Rows :")
len(duplicate)
```

```
Duplicate Rows :

12
```

Data Cleaning Approach

- Using **Random Forest Classifier** ML model to predict the 26% “unknown” values
- Dropped the duplicate rows
- Dropped the “poutcome” column

Function Steps:

1. Replace “Unknown” with NA
2. identify columns with NAs (‘Job’, ‘marital’, ‘education’, ‘default’, ‘housing’, ‘loan’)
    a. **df_not_missing**: rows without any NA
    b. **df_missing**: rows with NA

```
# define independent columns and target columns
df_not_missing = df_new[~df_new[missing_columns].isna().any(axis = 1)]
df_missing = df_new[df_new[missing_columns].isnull().any(axis=1)]
```

3. Using **the rows without any NA (df_not_missing)** to train the model
    a. **y** : target columns (‘Job’, ‘marital’, ‘education’, ‘default’, ‘housing’, ‘loan’) of **df_not_missing**
    b. **x** : independent columns: the rest of the columns, excluding ‘y’ and ‘poutcome’, of **df_not_missing**

```
# define x variable for training data
x = df_not_missing.drop(columns = missing_columns)
x = x.drop(columns = ['y','poutcome'])
x = pd.get_dummies(x, columns=x.select_dtypes(include=['object']).columns)
```

```
# define y variable for training data, and the column to be filled
y = df_not_missing[column]
```

4. Apply the model on **the rows with NA (df_missing)** to predict and fill the NA values in the target columns – One column at a time.
    a. **x_missing**: independent columns of **df_missing**
    b. **y_missing**: target columns of **df_missing**

```python
# define x variable for testing data
x_missing = df_missing.drop(columns = missing_columns)
x_missing = x_missing.drop(columns = ['y','poutcome'])
x_missing = pd.get_dummies(x_missing, columns=x_missing.select_dtypes(include=['
```

```python
#define the column to be filled
y_missing = df_missing[column]
```

5. concatenate the **df_not_missing** rows back with the updated **df_missing** rows
6. dropped the duplicate columns
7. dropped the ‘poutcome’ column

Result:

Before Cleaning: Many unknown values

```python
df_bank_add_full.loc[filtered_index]
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.ra
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1
5	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1
7	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1
10	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1
15	54	retired	married	basic.9y	unknown	yes	yes	telephone	may	mon	...	1	999	0	nonexistent	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41118	34	technician	married	unknown	no	yes	no	cellular	nov	tue	...	2	999	2	failure	-1
41120	60	admin.	married	unknown	no	no	no	cellular	nov	tue	...	2	999	0	nonexistent	-1
41122	34	technician	married	unknown	no	no	no	cellular	nov	tue	...	3	999	0	nonexistent	-1
41135	54	technician	married	unknown	no	yes	no	cellular	nov	thu	...	1	999	1	failure	-1
41175	34	student	single	unknown	no	yes	no	cellular	nov	thu	...	1	999	2	failure	-1

After Cleaning:

- The “unknown” value were replaced by the predicted values,
- The “poutcome” column is removed

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	emp.var.rate
1	57	services	married	high.school	no	no	no	telephone	may	mon	149	1	999	0	1.1
5	45	services	married	basic.9y	no	no	no	telephone	may	mon	198	1	999	0	1.1
7	41	blue-collar	married	basic.4y	no	no	no	telephone	may	mon	217	1	999	0	1.1
10	41	blue-collar	married	high.school	no	no	no	telephone	may	mon	55	1	999	0	1.1
15	54	retired	married	basic.9y	no	yes	yes	telephone	may	mon	174	1	999	0	1.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41118	34	technician	married	university.degree	no	yes	no	cellular	nov	tue	162	2	999	2	-1.1
41120	60	admin.	married	high.school	no	no	no	cellular	nov	tue	333	2	999	0	-1.1
41122	34	technician	married	university.degree	no	no	no	cellular	nov	tue	985	3	999	0	-1.1
41135	54	technician	married	university.degree	no	yes	no	cellular	nov	thu	222	1	999	1	-1.1
41175	34	student	single	university.degree	no	yes	no	cellular	nov	thu	180	1	999	2	-1.1

No Duplicate rows

```
duplicate = df_test[df_test.duplicated()]
print("Duplicate Rows :")
len(duplicate)
```

```
Duplicate Rows :

0
```

Justine Pile - bank.csv

1. Import the csv into a dataframe

```
[2]: # Import the bank.csv file into a pandas dataframe
df = pd.read_csv("Bank_Marketing_Group_Project/dataset/bank+marketing/bank/bank.csv")
df.head()
```

[2]:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcome";"y"
0	30;"unemployed";"married";"primary";"no";1787;...
1	33;"services";"married";"secondary";"no";4789;...
2	35;"management";"single";"tertiary";"no";1350;...
3	30;"management";"married";"tertiary";"no";1476;...
4	59;"blue-collar";"married";"secondary";"no";0;...

2. Separate the data into columns using the ; as a delimiter with str.split

```
[5]: # Seperate the df into seperate columns using the ; as the delimeter and keep the column names as-is
df = df['age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"poutcome";"y
df.head()
```

[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	30	"unemployed"	"married"	"primary"	"no"	1787	"no"	"no"	"cellular"	19	"oct"	79	1	-1	0	"unknown"	"no"
1	33	"services"	"married"	"secondary"	"no"	4789	"yes"	"yes"	"cellular"	11	"may"	220	1	339	4	"failure"	"no"
2	35	"management"	"single"	"tertiary"	"no"	1350	"yes"	"no"	"cellular"	16	"apr"	185	1	330	1	"failure"	"no"
3	30	"management"	"married"	"tertiary"	"no"	1476	"yes"	"yes"	"unknown"	3	"jun"	199	4	-1	0	"unknown"	"no"
4	59	"blue-collar"	"married"	"secondary"	"no"	0	"yes"	"no"	"unknown"	5	"may"	226	1	-1	0	"unknown"	"no"

3. Rename the columns

```
[6]: #Rename the columns in df
new_column_names = [
    "age", "job", "marital", "education", "default", "balance",
    "housing", "loan", "contact", "day", "month", "duration",
    "campaign", "pdays", "previous", "poutcome", "y"
]

df.columns = new_column_names
df.head()
```

[6]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	"unemployed"	"married"	"primary"	"no"	1787	"no"	"no"	"cellular"	19	"oct"	79	1	-1	0	"unknown"	"no"
1	33	"services"	"married"	"secondary"	"no"	4789	"yes"	"yes"	"cellular"	11	"may"	220	1	339	4	"failure"	"no"
2	35	"management"	"single"	"tertiary"	"no"	1350	"yes"	"no"	"cellular"	16	"apr"	185	1	330	1	"failure"	"no"
3	30	"management"	"married"	"tertiary"	"no"	1476	"yes"	"yes"	"unknown"	3	"jun"	199	4	-1	0	"unknown"	"no"
4	59	"blue-collar"	"married"	"secondary"	"no"	0	"yes"	"no"	"unknown"	5	"may"	226	1	-1	0	"unknown"	"no"

4. Verify there are no duplicates

```
[4]: # Verify there are no duplicate rows in dataframe
duplicate_rows = df.duplicated().sum()

if duplicate_rows == 0:
    print("No duplicate rows found in the DataFrame.")
else:
    print(f"{duplicate_rows} duplicate rows found in the DataFrame.")
```

```
No duplicate rows found in the DataFrame.
```

5. Check for null values

```
[7]: # Check for null values in df
null_values = df.isnull().sum()
print(null_values)
```

```
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
```

Rishi Aluri - bank-additional.csv

1. Remove ‘unkown’ from dataset. Introduces NA values.

```
# Replace 'unkown' with NaN

df.replace('unknown', np.nan, inplace=True)
```

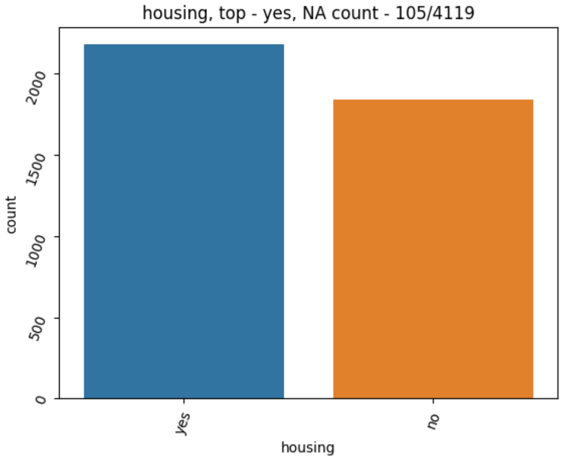
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             4119 non-null   int64
 1   job             4119 non-null   object
 2   marital         4119 non-null   object
 3   education       4119 non-null   object
 4   default         4119 non-null   object
 5   housing         4119 non-null   object
 6   loan            4119 non-null   object
 7   contact         4119 non-null   object
 8   month           4119 non-null   object
 9   day_of_week     4119 non-null   object
 10  duration        4119 non-null   int64
 11  campaign        4119 non-null   int64
 12  pdays           4119 non-null   int64
 13  previous        4119 non-null   int64
 14  poutcome        4119 non-null   object
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   object
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             4119 non-null   int64
 1   job             4080 non-null   object
 2   marital         4108 non-null   object
 3   education       3952 non-null   object
 4   default         3316 non-null   object
 5   housing         4014 non-null   object
 6   loan            4014 non-null   object
 7   contact         4119 non-null   object
 8   month           4119 non-null   object
 9   day_of_week     4119 non-null   object
 10  duration        4119 non-null   int64
 11  campaign        4119 non-null   int64
 12  pdays           4119 non-null   int64
 13  previous        4119 non-null   int64
 14  poutcome        4119 non-null   object
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   object
dtypes: float64(5), int64(5), object(11)
```

2. Cleaning categorical columns by imputing highest frequency value.

```
# Clean categorical columns - Impute Missing Values

 for col in df.select_dtypes(include=['object']).columns:

   df[col].fillna(df[col].mode().values[0], inplace=True)
```

Eg-




```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             4119 non-null   float64
 1   job             4119 non-null   category
 2   marital         4119 non-null   category
 3   education       4119 non-null   category
 4   default         4119 non-null   float64
 5   housing         4119 non-null   float64
 6   loan            4119 non-null   float64
 7   contact         4119 non-null   category
 8   month           4119 non-null   category
 9   day_of_week     4119 non-null   category
 10  duration        4119 non-null   float64
 11  campaign        4119 non-null   float64
 12  pdays           4119 non-null   float64
 13  previous        4119 non-null   float64
 14  poutcome        4119 non-null   category
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   float64
dtypes: category(7), float64(14)
memory usage: 480.5 KB
```

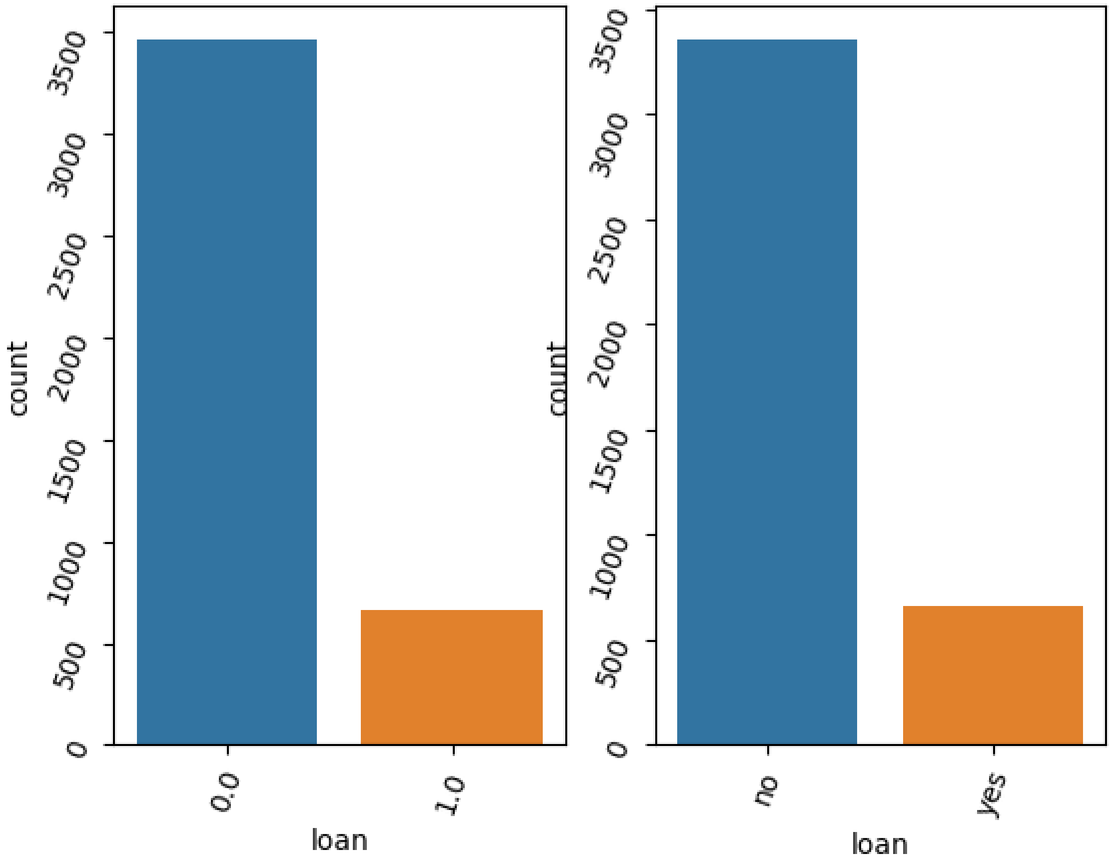
3. Column Transformations

```
# Column Transformations

for col in df.select_dtypes(include=['int64']).columns:

  df[col]=df[col].astype(np.float64)



for col in ['default', 'housing', 'loan', 'y']:

  df[col]=np.where(df[col]=='yes', 1.0, 0.0)



for col in df.select_dtypes(include=['object']).columns:

  df[col]=df[col].astype('category')
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
age	4119.0	NaN	NaN	NaN	40.11362	10.313362	18.0	32.0	38.0	47.0	88.0
job	4119	12	admin.	1012	NaN	NaN	NaN	NaN	NaN	NaN	NaN
marital	4119	4	married	2509	NaN	NaN	NaN	NaN	NaN	NaN	NaN
education	4119	8	university.degree	1264	NaN	NaN	NaN	NaN	NaN	NaN	NaN
default	4119	3	no	3315	NaN	NaN	NaN	NaN	NaN	NaN	NaN
housing	4119	3	yes	2175	NaN	NaN	NaN	NaN	NaN	NaN	NaN
loan	4119	3	no	3349	NaN	NaN	NaN	NaN	NaN	NaN	NaN
contact	4119	2	cellular	2652	NaN	NaN	NaN	NaN	NaN	NaN	NaN
month	4119	10	may	1378	NaN	NaN	NaN	NaN	NaN	NaN	NaN
day_of_week	4119	5	thu	860	NaN	NaN	NaN	NaN	NaN	NaN	NaN
duration	4119.0	NaN	NaN	NaN	256.788055	254.703736	0.0	103.0	181.0	317.0	3643.0
campaign	4119.0	NaN	NaN	NaN	2.537266	2.568159	1.0	1.0	2.0	3.0	35.0
pdays	4119.0	NaN	NaN	NaN	960.42219	191.922786	0.0	999.0	999.0	999.0	999.0
previous	4119.0	NaN	NaN	NaN	0.190337	0.541788	0.0	0.0	0.0	0.0	6.0
poutcome	4119	3	nonexistent	3523	NaN	NaN	NaN	NaN	NaN	NaN	NaN
emp.var.rate	4119.0	NaN	NaN	NaN	0.084972	1.563114	-3.4	-1.8	1.1	1.4	1.4
cons.price.idx	4119.0	NaN	NaN	NaN	93.579704	0.579349	92.201	93.075	93.749	93.994	94.767
cons.conf.idx	4119.0	NaN	NaN	NaN	-40.499102	4.594578	-50.8	-42.7	-41.8	-36.4	-26.9
euribor3m	4119.0	NaN	NaN	NaN	3.621356	1.733591	0.635	1.334	4.857	4.961	5.045
nr.employed	4119.0	NaN	NaN	NaN	5166.481695	73.667904	4963.6	5099.1	5191.0	5228.1	5228.1
y	4119	2	no	3668	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             4119 non-null   float64
 1   job             4119 non-null   category
 2   marital         4119 non-null   category
 3   education       4119 non-null   category
 4   default         4119 non-null   float64
 5   housing         4119 non-null   float64
 6   loan            4119 non-null   float64
 7   contact         4119 non-null   category
 8   month           4119 non-null   category
 9   day_of_week     4119 non-null   category
 10  duration        4119 non-null   float64
 11  campaign        4119 non-null   float64
 12  pdays           4119 non-null   float64
 13  previous        4119 non-null   float64
 14  poutcome        4119 non-null   category
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   float64
dtypes: category(7), float64(14)
memory usage: 480.5 KB
```

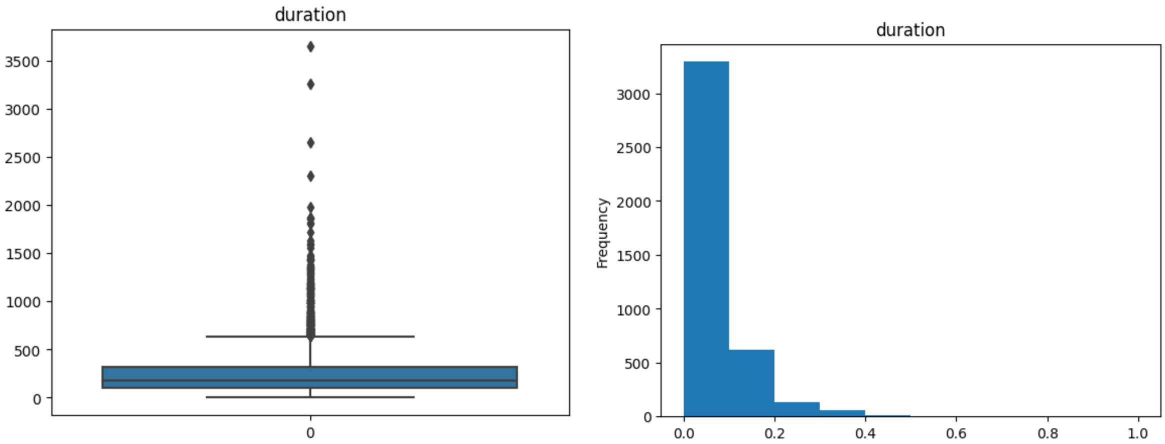
eg-



4. Cleaning numerical values by normalizing:

```
# Clean numerical columns - Normalize using MinMax

for col in df.select_dtypes(include=['float64']).columns:

  min_value = df[col].min()

  max_value = df[col].max()

  df[col]=((df[col] - min_value) / (max_value - min_value))
```

eg-


duration
3500
3000
2500
2000
1500
1000
500
0
0
duration
Frequency
3000
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0