Mouhamed DIOP
March 18, 2025

Project: France Traffic

1 Table of Contents

- 1 - Introduction
  - 1.1 Importing Libraries
- 2 - Data Wrangling
  - 2.1 - importing Data
  - 2.2 - Cleaning Data
    - 2.2.1 - Remove nulls
    - 2.2.2 - Remove Outliers
  - 2.3 - Creating new columns
- 3 - Exploratory Data Analysis
  - 3.1 - Histogram
  - 3.2 - Plot

- 3.3 - HeatMap
- 4 - Deep Learning Model -Dimension Reduction
  - Spliting Data
  - Training Model
  - Evaluate Model
- 5 - Conclusions
  - 5.1 - Limitations

2 Introduction

Title_categories Column:

- `IMAGINE R` : combines the annual Imagine R School and Imagine R Student packages reserved for pupils, apprentices and students which allows to travel at will all year round and in all of Ile-de-France.
- `NAVIGO` : includes the Navigo Annuel, Navigo Mois and Navigo Semaine packages.
- `AMETHYSTE` : includes the Amethyst packages: package reserved for seniors or disabled under conditions of means or status, and residing in the Île-de-France region.
- `TST` : groups together weekly and monthly reduced fare packages granted to beneficiaries of the Transportation Solidarity Reduction program, to travel within the selected zones in all the modes of transport in the Île-de-France region.
- `FGT` : accounts for the Navigo Gratuité Transport Packages, a package that allows certain receiving social assistance to travel free of charge throughout the Paris Region.
- `OTHER TITLE` : accounts for special packages.
- `NON DEFINED` : records validations for which the type of ticket is not defined (anomalies).

NB_VALID : Number of validations. 1 validation = 1 person

ID REFA LDA : `-1` means that the data is not defined.

[1]:
```python
#importing Libraries for exploring and Visualize Data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import seaborn as sns
%matplotlib inline
```

Data Wrangling In this section we will load data and perform some cleaning for the data finding duplicates and missing value editing the data type of the columns

2.0.1 Importing Data

Load data and explore it and make some notes for analysis later

[2]:
```python
# Load data
df=pd.read_csv('../input/public-transport-traffic-data-in-france/
↪Travel_titles_validations_in_Paris_and_suburbs.csv')
df.head()
```

[2]:

	DATE	STATION_NAME	ID_REFA_LDA	TITLE_CATEGORY	NB_VALID
0	21/07/2019	LA TOUR MAUBOURG	71242.0	NAVIGO	1141
1	21/07/2019	PARMENTIER	71801.0	NOT DEFINED	Less than 5
2	21/07/2019	PARMENTIER	71801.0	TST	97
3	21/07/2019	PEREIRE-LEVALLOIS	71453.0	FGT	53
4	21/07/2019	PERNETY	412687.0	OTHER	36

[3]:
```
#rows and columns
df.shape
```

[3]: (883958, 5)

[4]:
```
#information about data
# check data type of each coulmn
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 883958 entries, 0 to 883957
Data columns (total 5 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            883958 non-null  object
 1   STATION_NAME    883958 non-null  object
 2   ID_REFA_LDA     882459 non-null  float64
 3   TITLE_CATEGORY  883958 non-null  object
 4   NB_VALID        883958 non-null  object
dtypes: float64(1), object(4)
memory usage: 33.7+ MB
```

[5]:
```
#some statistical information about data
df.describe()
```

[5]:

	ID_REFA_LDA
count	882459.000000
mean	69150.277691
std	27606.821859
min	-1.000000
25%	66338.000000
50%	71158.000000
75%	71756.000000
max	415852.000000

[6]:
```
#check for removing -1 values
df.ID_REFA_LDA.nunique
```

[6]:
```
<bound method IndexOpsMixin.nunique of 0          71242.0
1          71801.0
2          71801.0
```

```
3          71453.0
4         412687.0
            ...
883953     74040.0
883954     73653.0
883955     71673.0
883956     71043.0
883957     73650.0
Name: ID_REFA_LDA, Length: 883958, dtype: float64>
```

Data Cleaning i'm going to clean my data, to ensure that the data if ready for my analysis. ### Removing Nulls

[7]:
```
#check null values
df.isnull().sum()
```

[7]:
```
DATE                  0
STATION_NAME          0
ID_REFA_LDA        1499
TITLE_CATEGORY        0
NB_VALID              0
dtype: int64
```

[8]:
```
#Drop null values
df.dropna(inplace=True)
df.shape
```

[8]:
```
(882459, 5)
```

[9]:
```
#confirmation of data is clean from
df.isnull().sum().any()
```

[9]:
```
False
```

- Note: There is null values in form of 0 value in the integer and float datatype. it may be wrong values or it is correct depend on the column as `0 in the year or month column is null value` But in another columns like arr_delay or arr_cancelled 0 is a valued number

[10]:
```
#Number of 0 values
for column_name in df.columns:
    column = df[column_name]
    # Get the count of Zeros in column
    count = (column == 0).sum()
    print('Count of zeros in column ', column_name, ' is : ', count)
```

```
Count of zeros in column  DATE  is :  0
Count of zeros in column  STATION_NAME  is :  0
Count of zeros in column  ID_REFA_LDA  is :  0
Count of zeros in column  TITLE_CATEGORY  is :  0
```

```
Count of zeros in column  NB_VALID  is :  0
```

2.0.2 Checking for Duplicates

[11]:
```python
#check Duplicates
df.duplicated().sum()
```

[11]:
```
1241
```

[12]:
```python
#remove duplicate but leave one
df.drop_duplicates(keep='first',inplace=True)
df.shape
```

[12]:
```
(881218, 5)
```

[13]:
```python
#showing -1 values in ID_REFA_LDA
df.query('ID_REFA_LDA == -1')
```

[13]:
```
              DATE      STATION_NAME  ID_REFA_LDA TITLE_CATEGORY     NB_VALID
98      21/07/2019   VILLETANEUSE U         -1.0       IMAGINE R          130
99      21/07/2019  PIERREFITTE T11         -1.0               ?  Less than 5
100     21/07/2019  DUGNY COURNEUVE         -1.0       AMETHYSTE           11
274     21/07/2019       ROSA PARKS         -1.0             TST          633
1038    19/08/2019          Inconnu         -1.0           OTHER           43
...            ...              ...          ...             ...          ...
883338  29/08/2019  STAINS CERISAIE         -1.0             FGT          176
883339  29/08/2019      BOURGET T11         -1.0          NAVIGO         1528
883488  29/08/2019       ROSA PARKS         -1.0       AMETHYSTE          142
883489  29/08/2019       ROSA PARKS         -1.0             FGT          441
883490  29/08/2019       ROSA PARKS         -1.0       IMAGINE R          799

[24210 rows x 5 columns]
```

[14]:
```python
#drop -1 as it is a undefind value for the data so it looks like null value
df.drop(df[df.ID_REFA_LDA == -1].index, inplace=True)
```

[15]:
```python
#showing Less than 5 values in NB_VALID
df.query('NB_VALID == "Less than 5"')
```

[15]:
```
              DATE                STATION_NAME  ID_REFA_LDA TITLE_CATEGORY  \
1       21/07/2019                  PARMENTIER      71801.0    NOT DEFINED
15      21/07/2019  PONT-MARIE (CITE DES ARTS)      71217.0   DAILY NAVIGO
20      21/07/2019       PORTE DE CLIGNANCOURT      72059.0    NOT DEFINED
24      21/07/2019        PORTE DE SAINT-CLOUD      71084.0    NOT DEFINED
26      21/07/2019              RICHARD LENOIR      73648.0    NOT DEFINED
...            ...                         ...          ...            ...
883870  02/09/2019             VAL-D'ARGENTEUIL      65110.0   DAILY NAVIGO
883876  02/09/2019          VILLENNES-SUR-SEINE      64949.0              ?
```

```
883888  02/09/2019     CERGY-SAINT-CHRISTOPHE      66858.0          OTHER
883905  02/09/2019                     NATION      71673.0   DAILY NAVIGO
883908  02/09/2019        NEUVILLE UNIVERSITE      66436.0   DAILY NAVIGO

           NB_VALID
1       Less than 5
15      Less than 5
20      Less than 5
24      Less than 5
26      Less than 5
...             ...
883870  Less than 5
883876  Less than 5
883888  Less than 5
883905  Less than 5
883908  Less than 5

[122895 rows x 5 columns]
```

[16]:
```
#Change less than 5 value to be 5 so we can change the type of this column to␣
↪integer to make the numbers useful
df['NB_VALID'] = df['NB_VALID'].replace({'Less than 5' : 5})
```

[17]:
```
#Change datatype of NB_VALID to Numeric (integer) datatype
df['NB_VALID'] = pd.to_numeric(df.NB_VALID)
```

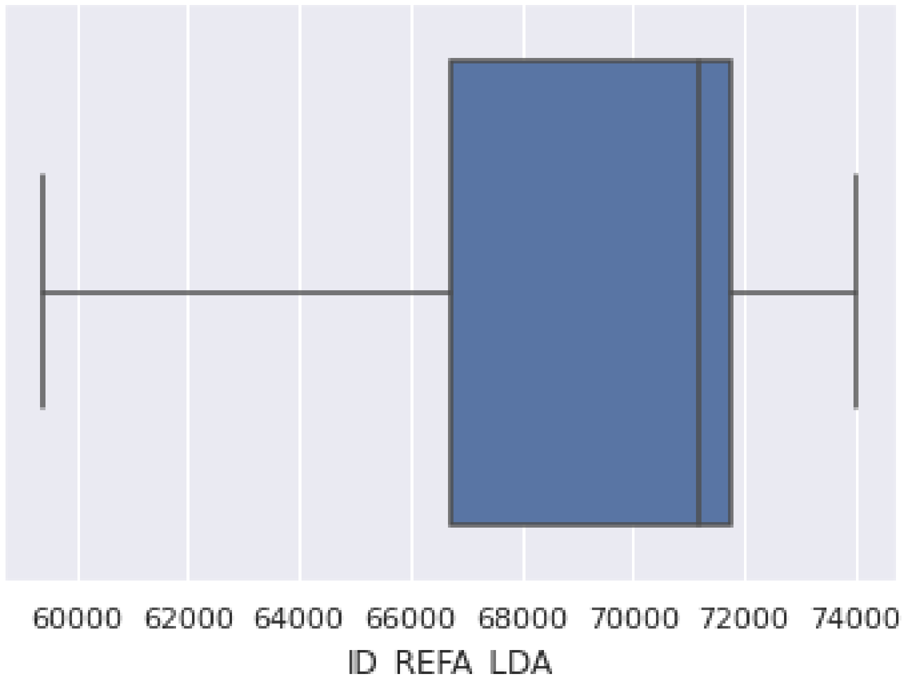
[18]:
```
#define threshold values to category with them
l = df.NB_VALID.min() - 1
me = df.NB_VALID.quantile(0.25)
mh = df.NB_VALID.quantile(0.5)
h = df.NB_VALID.quantile(0.75)
maxx = df.NB_VALID.max()
#define bins that will use in creating categories
binss = (l,me,mh,h,maxx)
#Extract a categorical feature from NB_VALID to make it the target
df['NB_Category'] = pd.cut(df.NB_VALID, bins=binss, labels=[1,2,3,4])
```

2.1 Remove Outliers

[19]:
```
#Dfine boundries for outliers
q_low = df["ID_REFA_LDA"].quantile(0.00001)
q_hi  = df["ID_REFA_LDA"].quantile(0.99)
#clear outliers
df = df[(df["ID_REFA_LDA"] < q_hi) & (df["ID_REFA_LDA"] > q_low)]
```

[20]:
```
#check outliers
sns.set()
```

```
sns.boxplot(data = df , x = df.ID_REFA_LDA);
```



Extract Columns

[21]:
```
#Change datatype of Release data to date datatype
df['DATE'] = pd.to_datetime(df.DATE)
#make a new column month to more uderstanding the effect of time to data
df['month'] = df['DATE'].dt.month
#make a new colum day from Date to drop the date later
df['Day'] = df['DATE'].dt.day
```

[22]:
```
df.shape
```

[22]:
```
(847660, 8)
```

- we can see the their is no outliers at all after clearing them

[23]:
```
# View the index number and label for each column
for x, y in enumerate(df.columns):
    print(x, y)
```

```
0 DATE
1 STATION_NAME
2 ID_REFA_LDA
3 TITLE_CATEGORY
4 NB_VALID
```

```
5 NB_Category
6 month
7 Day
```

[24]:
```python
#Statistical Information about data after preprocessing
df.describe()
```

[24]:

	ID_REFA_LDA	NB_VALID	month	Day
count	847660.000000	847660.000000	847660.000000	847660.000000
mean	69347.075890	771.126769	8.154616	16.934739
std	3874.674618	2789.785948	2.912933	7.497386
min	59420.000000	5.000000	1.000000	7.000000
25%	66731.000000	14.000000	7.000000	10.000000
50%	71184.000000	87.000000	9.000000	16.000000
75%	71756.000000	379.000000	10.000000	24.000000
max	74002.000000	125007.000000	12.000000	31.000000

[25]:
```python
#correlation of data
df.corr()
```

[25]:

	ID_REFA_LDA	NB_VALID	month	Day
ID_REFA_LDA	1.000000	0.105305	-0.013032	-0.002990
NB_VALID	0.105305	1.000000	-0.002996	-0.005830
month	-0.013032	-0.002996	1.000000	0.400086
Day	-0.002990	-0.005830	0.400086	1.000000

Exploratory Data Analysis

[26]:
```python
#histogram visualization of data
sns.set();
df.hist(figsize=(15,10));
```

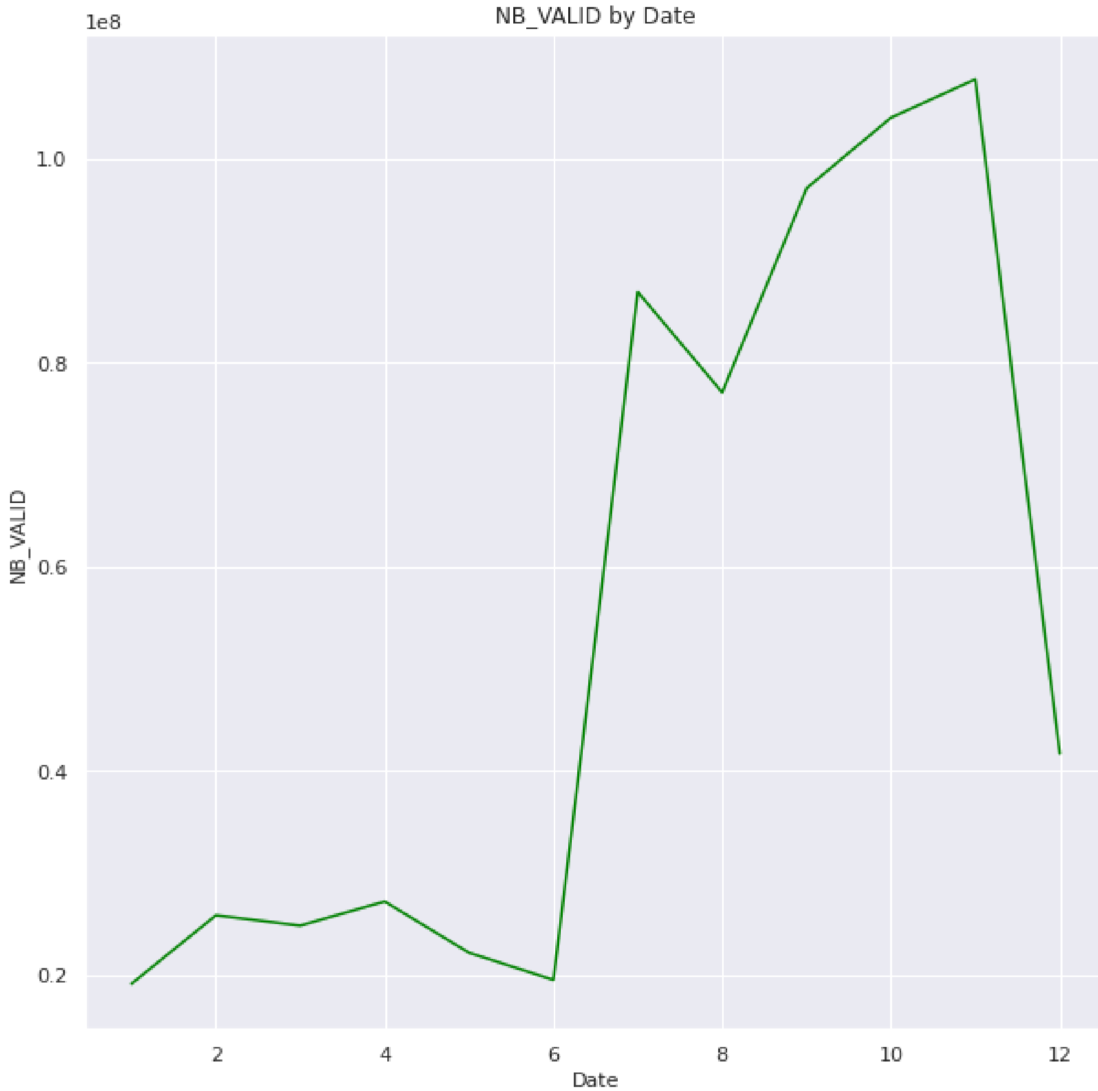
[27]:
```
#group NB_VALID and ID_REFA_LDA by month
date=df.groupby(df.month)[['NB_VALID','ID_REFA_LDA']].sum()
date.head()
```

[27]:

month	NB_VALID	ID_REFA_LDA
1	19156587	2.057595e+09
2	25865935	2.101168e+09
3	24864526	2.084706e+09
4	27225443	2.112458e+09
5	22197690	1.878157e+09

[28]:
```
#visualization of Date with NB_VALID
sns.set()
plt.figure(figsize=(10,10))

# x-axis
plt.xlabel('Date', fontsize = 11)
# y-axis
plt.ylabel('NB_VALID', fontsize = 11)
# Title
plt.title('NB_VALID by Date')
# Legend
```

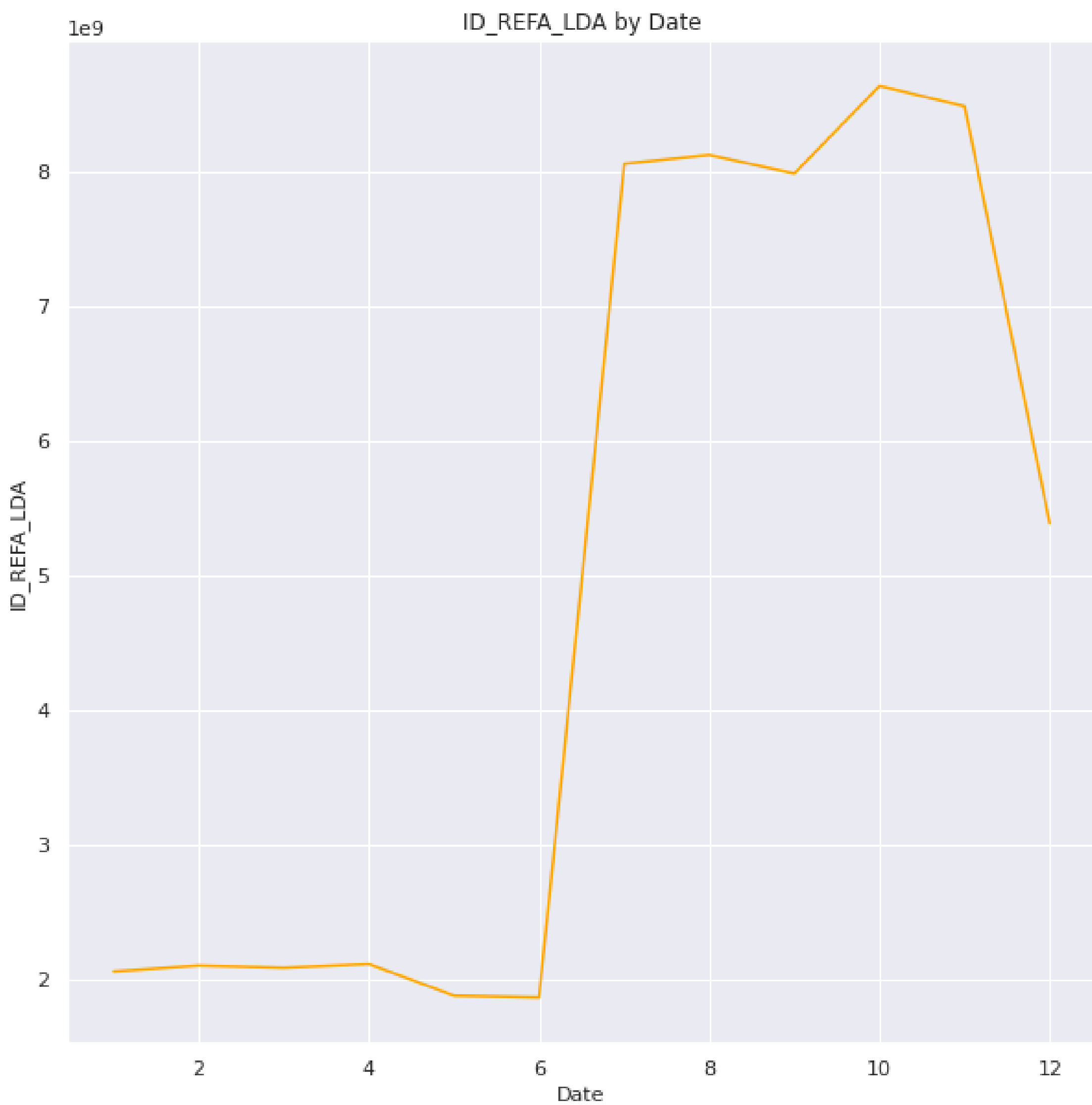
```python
# Plot Line chart
plt.plot(date.NB_VALID,color='green')

# Display plot
plt.show()
```



- We can notice that NB_valid start increase from june “6 month” and the first 5 months NB_valid doesn’t exceed 30 Million validations

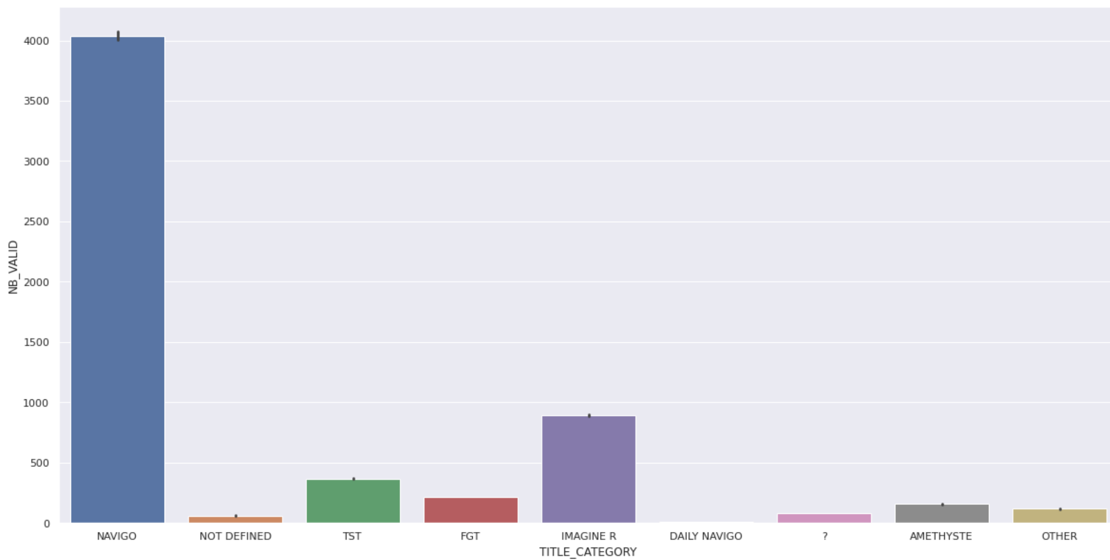
[29]:
```python
#visualization of DATE with ID_REFA_LDA
sns.set()
plt.figure(figsize=(10,10))
```

```python
# x-axis
plt.xlabel('Date', fontsize = 11)
# y-axis
plt.ylabel('ID_REFA_LDA', fontsize = 11)
# Title
plt.title('ID_REFA_LDA by Date')
# Legend

# Plot Line chart
plt.plot(date.ID_REFA_LDA,color='orange')

# Display plot
plt.show()
```



- We can notice that ID_REFA_LDA start increase from june "6 month" and the first 5 months NB_valid doesn't exceed 2.3 trillion ID_REFA_LDA

[30]:
```
#visualization of TITLE CATEGORY with NB_VALID
sns.set()
plt.figure(figsize=(20,10))
sns.barplot(data=df, x = df.TITLE_CATEGORY, y = df.NB_VALID)
plt.show()
```



- NAVIGO title category is the most selled ticket all over the stations

[31]:
```
#group station name wiht NB_VALID and ID_REFA_LDA sum
ST_df=df.groupby(df.STATION_NAME)[['NB_VALID','ID_REFA_LDA']].sum()
ST_df
```

[31]:

STATION_NAME	NB_VALID	ID_REFA_LDA
ABBESSES	465990	99076184.0
ABLON	171216	80960856.0
ACHERES-GRAND-CORMIER	11577	58475430.0
ACHERES-VILLE	765695	102456768.0
AEROPORT CHARLES DE GAULLE 1	980196	102887208.0
...	...	...
VOLTAIRE (LEON BLUM)	1715582	105185500.0
VOSVES	3839	30368350.0
VULAINES-SUR-SEINE-SAMOREAU	7160	44743380.0
WAGRAM	799601	105777463.0

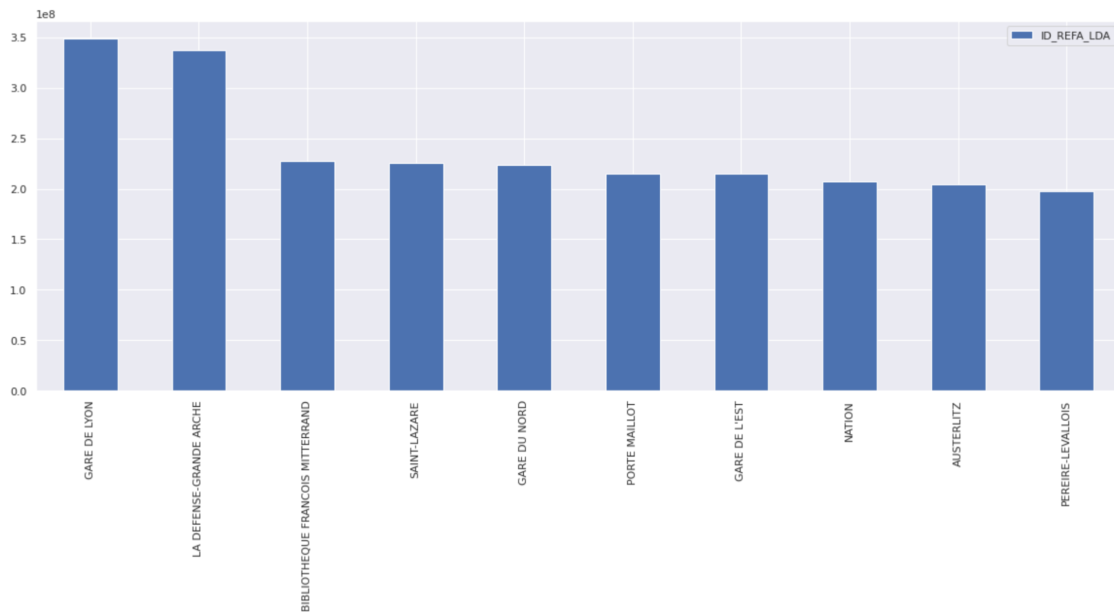
```
YERRES                              680726   83659388.0

[660 rows x 2 columns]
```

[32]:
```
#station name with the sum of ID_REFA_LDA
d10 = ST_df.nlargest(10, 'ID_REFA_LDA')
T10 = d10.loc[:,['ID_REFA_LDA']].head(10)
#station name with the sum of NB_VALID
dN10 = ST_df.nlargest(10, 'NB_VALID')
N10 = dN10.loc[:,['NB_VALID']].head(10)
```

[33]:
```
T10
```

[33]:
```
                                    ID_REFA_LDA
STATION_NAME
GARE DE LYON                        348619110.0
LA DEFENSE-GRANDE ARCHE             337059621.0
BIBLIOTHEQUE FRANCOIS MITTERRAND    227527388.0
SAINT-LAZARE                        225814680.0
GARE DU NORD                        223513300.0
PORTE MAILLOT                       214850790.0
GARE DE L'EST                       214719231.0
NATION                              206991624.0
AUSTERLITZ                          203872910.0
PEREIRE-LEVALLOIS                   197210280.0
```

[34]:
```
#Visualize bar plot
sns.set()
T10.plot.bar(figsize=(20,7));
```



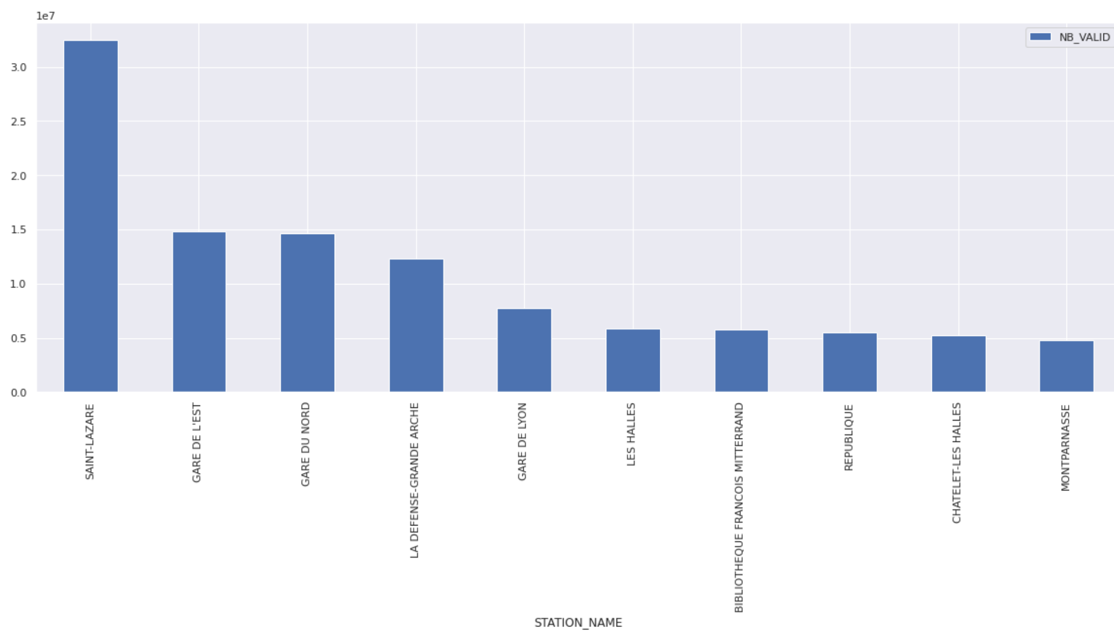
- HOTEL DE VILLE Station has the highest ID_REFA_LDA

[35]:
```
N10
```

[35]:

```
                                  NB_VALID
STATION_NAME
SAINT-LAZARE                      32477264
GARE DE L'EST                     14853378
GARE DU NORD                      14651981
LA DEFENSE-GRANDE ARCHE           12303820
GARE DE LYON                       7758677
LES HALLES                         5833991
BIBLIOTHEQUE FRANCOIS MITTERRAND   5748016
REPUBLIQUE                         5549338
CHATELET-LES HALLES                5272294
MONTPARNASSE                       4797831
```

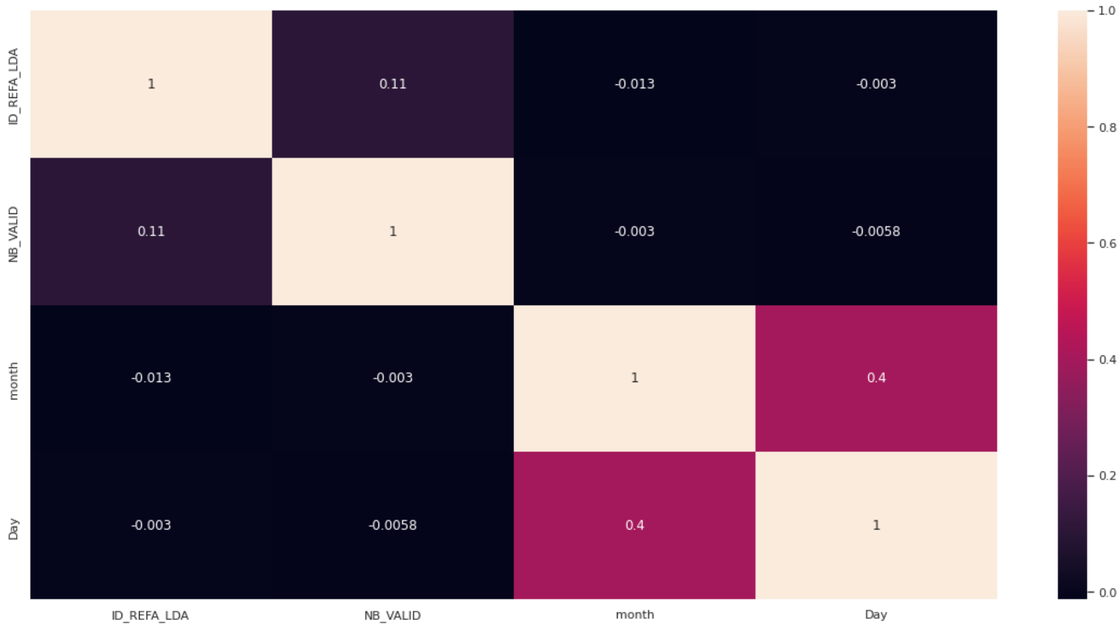
[36]:
```
#Visualize bar plot
sns.set()
N10.plot.bar(figsize=(20,7));
```



- SAINT-LAZARE Station has the highest NB_VALID

[37]:
```
#making heatmap for all data
plt.figure(figsize=(20,10))
```

```
sns.heatmap(df.corr(),annot=True)
plt.show()
```



[38]:
```
#one hot encoding for the categorical columns to can be trained in the model
df = pd.get_dummies(df,columns=['TITLE_CATEGORY', 'STATION_NAME'])
df
```

[38]:
```
              DATE  ID_REFA_LDA  NB_VALID NB_Category  month  Day  \
0       2019-07-21      71242.0      1141           4      7   21
1       2019-07-21      71801.0         5           1      7   21
2       2019-07-21      71801.0        97           3      7   21
3       2019-07-21      71453.0        53           2      7   21
5       2019-07-21      71639.0        25           2      7   21
...            ...          ...       ...         ...    ...  ...
883952  2019-04-09      73671.0        84           2      4    9
883954  2019-04-09      73653.0      2805           4      4    9
883955  2019-04-09      71673.0     14377           4      4    9
883956  2019-04-09      71043.0      4613           4      4    9
883957  2019-04-09      73650.0      1019           4      4    9

        TITLE_CATEGORY_?  TITLE_CATEGORY_AMETHYSTE  \
0                      0                         0
1                      0                         0
2                      0                         0
3                      0                         0
5                      0                         0
```

```
...                  ...                        ...
883952                 0                          0
883954                 0                          0
883955                 0                          0
883956                 0                          0
883957                 0                          0

        TITLE_CATEGORY_DAILY NAVIGO  TITLE_CATEGORY_FGT  ...  \
0                                 0                   0  ...
1                                 0                   0  ...
2                                 0                   0  ...
3                                 0                   1  ...
5                                 0                   1  ...
...                             ...                 ...  ...
883952                            0                   1  ...
883954                            0                   0  ...
883955                            0                   0  ...
883956                            0                   0  ...
883957                            0                   0  ...

        STATION_NAME_VIROFLAY RIVE DROITE  STATION_NAME_VIROFLAY RIVE GAUCHE  \
0                                       0                                  0
1                                       0                                  0
2                                       0                                  0
3                                       0                                  0
5                                       0                                  0
...                                   ...                                ...
883952                                  0                                  0
883954                                  0                                  0
883955                                  0                                  0
883956                                  0                                  0
883957                                  0                                  0

        STATION_NAME_VIRY-CHATILLON  STATION_NAME_VITRY-SUR-SEINE  \
0                                 0                             0
1                                 0                             0
2                                 0                             0
3                                 0                             0
5                                 0                             0
...                             ...                           ...
883952                            0                             0
883954                            0                             0
883955                            0                             0
883956                            0                             0
883957                            0                             0

        STATION_NAME_VOLONTAIRES  STATION_NAME_VOLTAIRE (LEON BLUM)  \
```

```
0                              0                                  0
1                              0                                  0
2                              0                                  0
3                              0                                  0
5                              0                                  0
...                          ...                                ...
883952                         0                                  0
883954                         0                                  0
883955                         0                                  0
883956                         0                                  0
883957                         0                                  0

        STATION_NAME_VOSVES  STATION_NAME_VULAINES-SUR-SEINE-SAMOREAU  \
0                         0                                         0
1                         0                                         0
2                         0                                         0
3                         0                                         0
5                         0                                         0
...                     ...                                       ...
883952                    0                                         0
883954                    0                                         0
883955                    0                                         0
883956                    0                                         0
883957                    0                                         0

        STATION_NAME_WAGRAM  STATION_NAME_YERRES
0                         0                    0
1                         0                    0
2                         0                    0
3                         0                    0
5                         0                    0
...                     ...                  ...
883952                    0                    0
883954                    0                    0
883955                    0                    0
883956                    0                    0
883957                    0                    0

[847660 rows x 675 columns]
```

[39]:
```
#split the data from the target data
x = df.drop(labels=['DATE','NB_VALID','NB_Category'], axis=1)
y = df.NB_Category
```

[40]:
```
#X Y shape
print('The X data shape : ', x.shape)
print('The Target shape : ', y.shape)
```

```
The X data shape :  (847660, 672)
The Target shape :  (847660,)
```

3 Deep Learning Model

Dimension Reduction

Spliting Data

Building Model

Training Model

Evaluate Model

Dimension Reduction

[41]:
```python
# #Import PCA to make dimension reduction
# from sklearn.decomposition import PCA
# #take an object from PCA model
# pca = PCA(n_components=0.95)
# #train PCA on the data
# x_pca = pca.fit_transform(x)
# #Shape of the data
# print('Shape of the Data before PCA : ', x.shape)
# print('Shape of the Data after PCA : ', x_pca.shape)
```

[42]:
```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

x = scaler.fit_transform(x)
```

Spliting Data

[43]:
```python
#split data to x_train, x_test, y_train, y_test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1,␣
↪random_state=40,shuffle=True)
```

[44]:
```python
#clear memory
del df
del x
del y
del l
del me
del mh
del h
del maxx
del d10
del ST_df
```

```
del T10
del dN10
del N10
```

[45]:
```
#Shape of the data
print('Shape of the x_train data : ', x_train.shape)
print('Shape of the y_train data : ', y_train.shape)
print('Shape of the x_test data : ', x_test.shape)
print('Shape of the y_test data : ', y_test.shape)
```

```
Shape of the x_train data :  (762894, 672)
Shape of the y_train data :  (762894,)
Shape of the x_test data :  (84766, 672)
Shape of the y_test data :  (84766,)
```

3.1 Building Model

[46]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization,Dropout
import tensorflow.keras as tf
```

[74]:
```
# Build the neural network
model = Sequential([

Dense(1024, activation='relu'), # Hidden 1
BatchNormalization(),# Hidden 2
Dense(512, activation='relu'), # Hidden 3
Dense(256, activation='relu'), # Hidden 6
Dense(256, activation='relu'), # Hidden 6
Dense(128, activation='relu'), # Hidden 8
Dense(64, activation='relu'),# Hidden 10
Dense(32, activation='relu'),# Hidden 11
Dense(5, activation = 'softmax')
])
```

3.2 Training Model

[48]:
```
model.compile(optimizer=tf.optimizers.Adam(),
                loss=tf.losses.SparseCategoricalCrossentropy(),
                metrics=['accuracy'])
```

[49]:
```
history = model.fit(x = x_train, y = y_train, epochs = 65 , validation_split=0.
↪15, batch_size = 64)
```

```
2022-09-14 07:42:26.238653: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR
Optimization Passes are enabled (registered 2)
```

```
Epoch 1/65
10133/10133 [==============================] - 123s 12ms/step - loss: 0.5695 -
accuracy: 0.7598 - val_loss: 0.4822 - val_accuracy: 0.7955
Epoch 2/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.4674 -
accuracy: 0.8033 - val_loss: 0.4504 - val_accuracy: 0.8121
Epoch 3/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.4255 -
accuracy: 0.8214 - val_loss: 0.4036 - val_accuracy: 0.8301
Epoch 4/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.4016 -
accuracy: 0.8300 - val_loss: 0.3865 - val_accuracy: 0.8359
Epoch 5/65
10133/10133 [==============================] - 123s 12ms/step - loss: 0.3868 -
accuracy: 0.8355 - val_loss: 0.3794 - val_accuracy: 0.8394
Epoch 6/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3762 -
accuracy: 0.8399 - val_loss: 0.3709 - val_accuracy: 0.8437
Epoch 7/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3681 -
accuracy: 0.8439 - val_loss: 0.4223 - val_accuracy: 0.8235
Epoch 8/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3611 -
accuracy: 0.8466 - val_loss: 0.3618 - val_accuracy: 0.8486
Epoch 9/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.3551 -
accuracy: 0.8490 - val_loss: 0.3563 - val_accuracy: 0.8495
Epoch 10/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3499 -
accuracy: 0.8512 - val_loss: 0.3456 - val_accuracy: 0.8546
Epoch 11/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3459 -
accuracy: 0.8531 - val_loss: 0.3488 - val_accuracy: 0.8536
Epoch 12/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.3414 -
accuracy: 0.8549 - val_loss: 0.4444 - val_accuracy: 0.8163
Epoch 13/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3378 -
accuracy: 0.8561 - val_loss: 0.3415 - val_accuracy: 0.8556
Epoch 14/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.3377 -
accuracy: 0.8574 - val_loss: 0.3393 - val_accuracy: 0.8578
Epoch 15/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3315 -
accuracy: 0.8592 - val_loss: 0.3420 - val_accuracy: 0.8571
Epoch 16/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3290 -
accuracy: 0.8592 - val_loss: 0.3416 - val_accuracy: 0.8584
```

```
Epoch 17/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3261 -
accuracy: 0.8607 - val_loss: 0.3384 - val_accuracy: 0.8588
Epoch 18/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3219 -
accuracy: 0.8620 - val_loss: 0.3359 - val_accuracy: 0.8597
Epoch 19/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3201 -
accuracy: 0.8628 - val_loss: 1.8331 - val_accuracy: 0.8609
Epoch 20/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.3206 -
accuracy: 0.8642 - val_loss: 0.3328 - val_accuracy: 0.8620
Epoch 21/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3142 -
accuracy: 0.8648 - val_loss: 0.3316 - val_accuracy: 0.8634
Epoch 22/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3129 -
accuracy: 0.8653 - val_loss: 0.3354 - val_accuracy: 0.8603
Epoch 23/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3099 -
accuracy: 0.8666 - val_loss: 0.3378 - val_accuracy: 0.8623
Epoch 24/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3085 -
accuracy: 0.8669 - val_loss: 0.3307 - val_accuracy: 0.8633
Epoch 25/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.4143 -
accuracy: 0.8675 - val_loss: 0.3565 - val_accuracy: 0.8655
Epoch 26/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3056 -
accuracy: 0.8687 - val_loss: 0.3345 - val_accuracy: 0.8655
Epoch 27/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3036 -
accuracy: 0.8683 - val_loss: 261.6591 - val_accuracy: 0.8637
Epoch 28/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.3054 -
accuracy: 0.8693 - val_loss: 0.3362 - val_accuracy: 0.8635
Epoch 29/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2986 -
accuracy: 0.8707 - val_loss: 0.3406 - val_accuracy: 0.8674
Epoch 30/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2977 -
accuracy: 0.8712 - val_loss: 0.3358 - val_accuracy: 0.8677
Epoch 31/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2954 -
accuracy: 0.8725 - val_loss: 38.9897 - val_accuracy: 0.8686
Epoch 32/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2933 -
accuracy: 0.8733 - val_loss: 10.3306 - val_accuracy: 0.8668
```

```
Epoch 33/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2929 -
accuracy: 0.8734 - val_loss: 0.3229 - val_accuracy: 0.8675
Epoch 34/65
10133/10133 [==============================] - 122s 12ms/step - loss: 0.6435 -
accuracy: 0.8745 - val_loss: 0.3267 - val_accuracy: 0.8707
Epoch 35/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2869 -
accuracy: 0.8754 - val_loss: 0.3267 - val_accuracy: 0.8653
Epoch 36/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2868 -
accuracy: 0.8756 - val_loss: 0.3432 - val_accuracy: 0.8691
Epoch 37/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2835 -
accuracy: 0.8767 - val_loss: 0.3609 - val_accuracy: 0.8689
Epoch 39/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2814 -
accuracy: 0.8779 - val_loss: 0.3311 - val_accuracy: 0.8694
Epoch 40/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2803 -
accuracy: 0.8786 - val_loss: 0.3305 - val_accuracy: 0.8702
Epoch 41/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2779 -
accuracy: 0.8793 - val_loss: 0.3600 - val_accuracy: 0.8708
Epoch 42/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2773 -
accuracy: 0.8798 - val_loss: 3.2884 - val_accuracy: 0.8693
Epoch 43/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2777 -
accuracy: 0.8798 - val_loss: 0.3247 - val_accuracy: 0.8727
Epoch 44/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2745 -
accuracy: 0.8811 - val_loss: 0.5285 - val_accuracy: 0.8686
Epoch 45/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2865 -
accuracy: 0.8811 - val_loss: 0.3488 - val_accuracy: 0.8715
Epoch 46/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2724 -
accuracy: 0.8816 - val_loss: 0.3311 - val_accuracy: 0.8719
Epoch 47/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2701 -
accuracy: 0.8830 - val_loss: 0.4054 - val_accuracy: 0.8503
Epoch 48/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2707 -
accuracy: 0.8834 - val_loss: 0.3283 - val_accuracy: 0.8695
Epoch 49/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2803 -
accuracy: 0.8837 - val_loss: 306.1778 - val_accuracy: 0.8667
```

```
Epoch 50/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2881 -
accuracy: 0.8848 - val_loss: 0.3405 - val_accuracy: 0.8735
Epoch 51/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2641 -
accuracy: 0.8855 - val_loss: 0.3391 - val_accuracy: 0.8746
Epoch 52/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2626 -
accuracy: 0.8862 - val_loss: 0.5994 - val_accuracy: 0.8054
Epoch 53/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2622 -
accuracy: 0.8869 - val_loss: 0.3344 - val_accuracy: 0.8708
Epoch 54/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2588 -
accuracy: 0.8879 - val_loss: 0.3304 - val_accuracy: 0.8718
Epoch 55/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2574 -
accuracy: 0.8884 - val_loss: 0.3155 - val_accuracy: 0.8738
Epoch 56/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2565 -
accuracy: 0.8889 - val_loss: 0.3274 - val_accuracy: 0.8770
Epoch 57/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2539 -
accuracy: 0.8898 - val_loss: 0.3174 - val_accuracy: 0.8755
Epoch 58/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2545 -
accuracy: 0.8903 - val_loss: 0.3388 - val_accuracy: 0.8724
Epoch 59/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2534 -
accuracy: 0.8905 - val_loss: 0.3318 - val_accuracy: 0.8771
Epoch 60/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2492 -
accuracy: 0.8925 - val_loss: 0.3312 - val_accuracy: 0.8790
Epoch 61/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2519 -
accuracy: 0.8921 - val_loss: 0.3231 - val_accuracy: 0.8783
Epoch 62/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2471 -
accuracy: 0.8927 - val_loss: 0.3101 - val_accuracy: 0.8782
Epoch 63/65
10133/10133 [==============================] - 120s 12ms/step - loss: 0.2461 -
accuracy: 0.8937 - val_loss: 0.3322 - val_accuracy: 0.8714
Epoch 64/65
10133/10133 [==============================] - 121s 12ms/step - loss: 0.2458 -
accuracy: 0.8946 - val_loss: 0.3385 - val_accuracy: 0.8770
Epoch 65/65
10133/10133 [==============================] - 122s 12ms/step - loss: 0.2498 -
accuracy: 0.8940 - val_loss: 0.3161 - val_accuracy: 0.8811
```

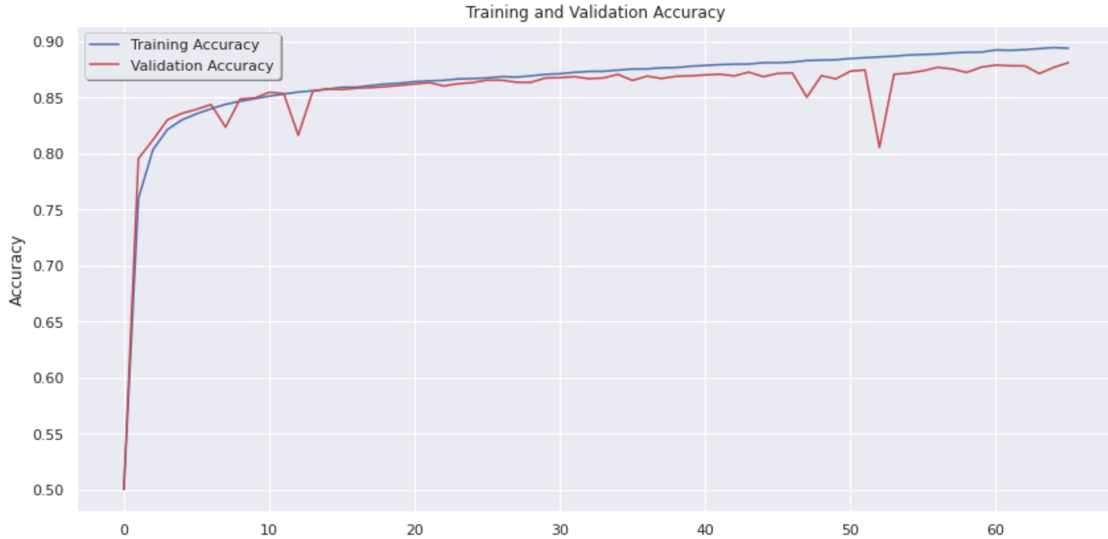
3.3 Evaluate Model

[50]:
```
model.evaluate(x_test,y_test)
```

```
2649/2649 [==============================] - 8s 3ms/step - loss: 0.3122 -
accuracy: 0.8815
```

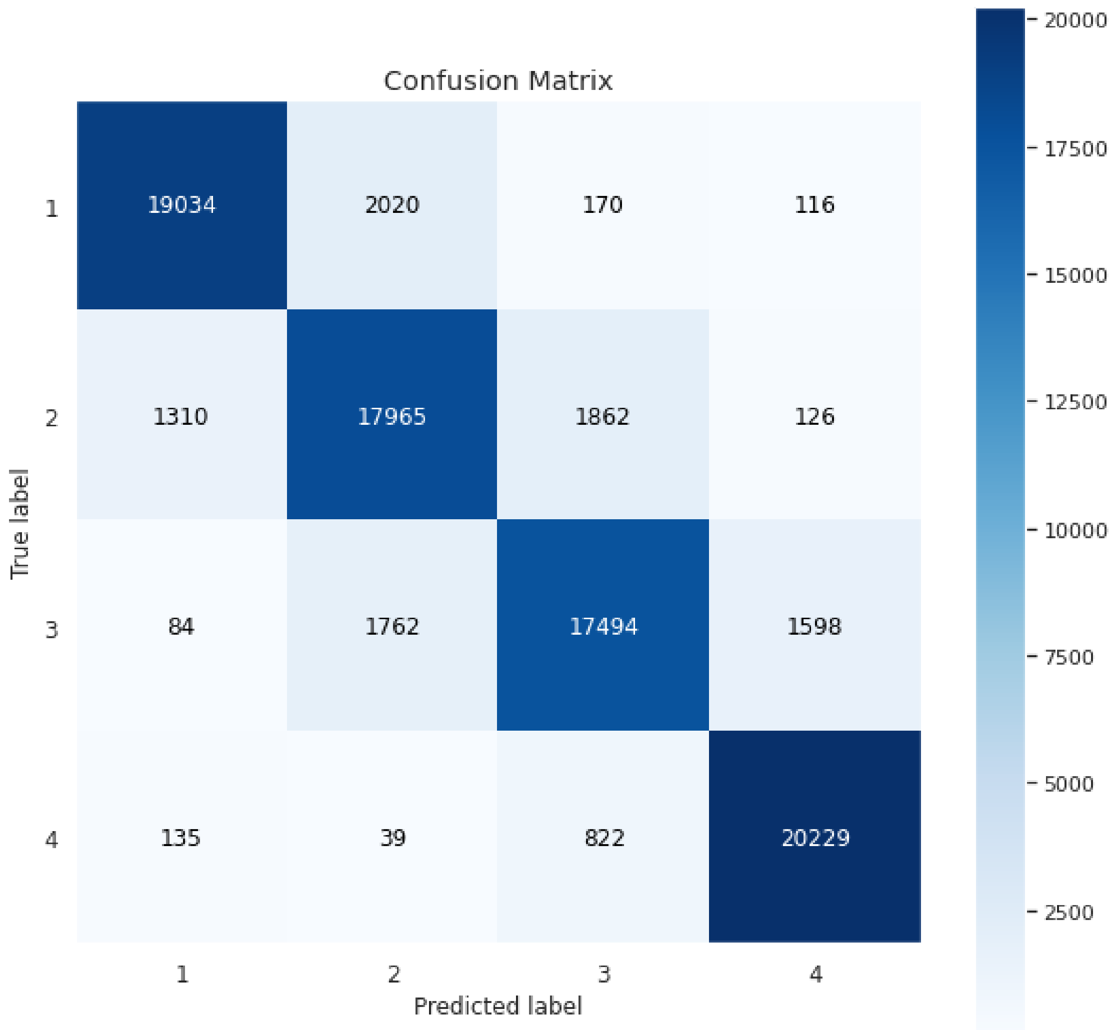
[50]:
```
[0.31220489740371704, 0.8815091252326965]
```

[70]:
```
sns.set()
acc = [0.5] + history.history['accuracy']
val_acc = [0.5] + history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12))
ax1.plot(acc,color='b', label='Training Accuracy')
ax1.plot(val_acc,color='r', label='Validation Accuracy')
ax1.legend(loc='best', shadow=True)
ax1.set_ylabel('Accuracy')
ax1.set_title('Training and Validation Accuracy')

ax2.plot(loss,color='b', label='Training Loss')
ax2.plot(val_loss,color='r', label='Validation Loss')
ax2.legend(loc='best', shadow=True)
ax2.set_ylabel('Cross Entropy')
ax2.set_title('Training and Validation Loss')
ax2.set_xlabel('epoch')
plt.tight_layout()

plt.show()
```



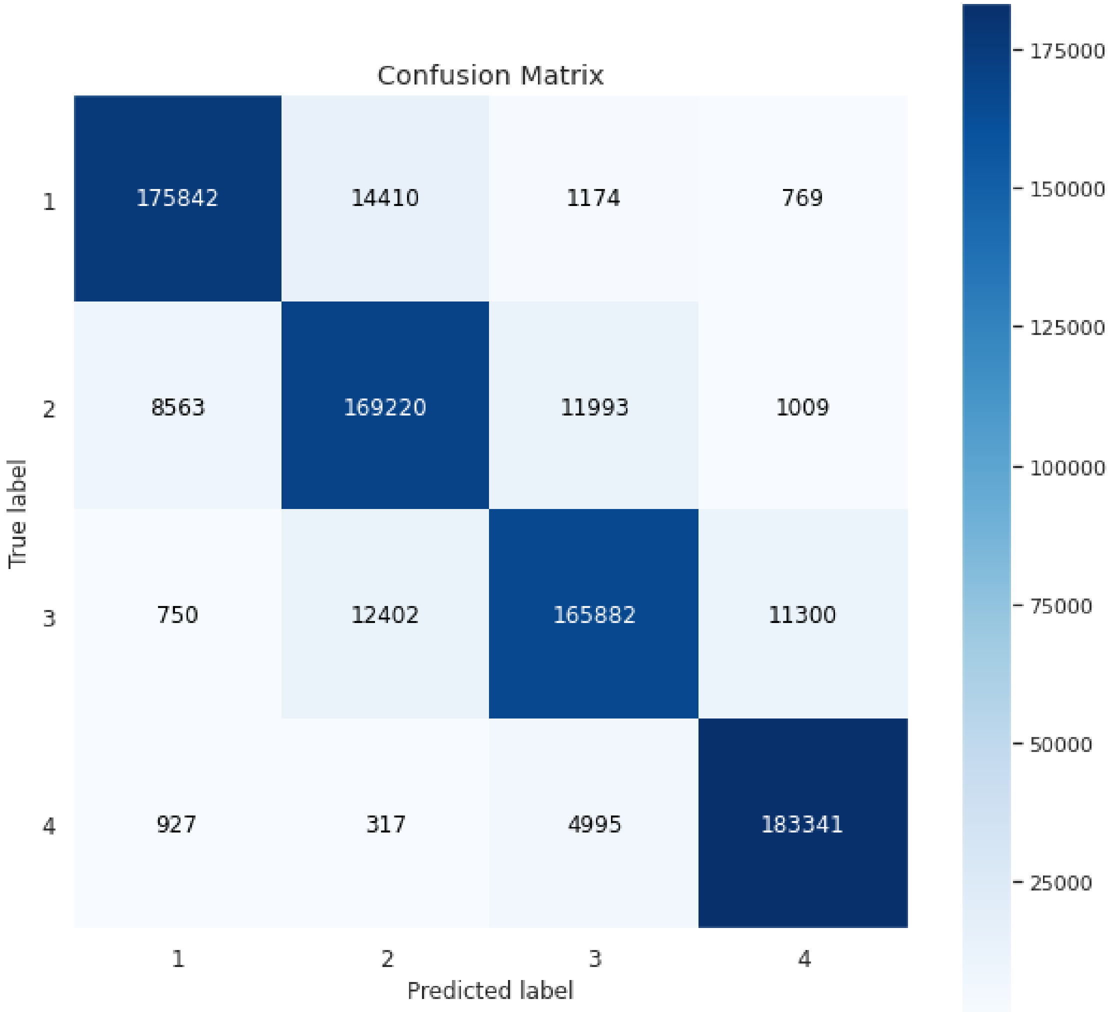
[52]:
```
#predict for Test set
y_pred = model.predict(x_test)
```

[53]:
```
#Confusion_Matrix for Test
from scikitplot.metrics import plot_confusion_matrix
cm = plot_confusion_matrix(y_test.ravel(), np.
↪argmax(y_pred,axis=1),figsize=(10,10))
```



[54]:
```
#prediction for Train set
x_train_prediction = model.predict(x_train)
```

[55]:
```
#Confusion_Matrix for Train
cmt = plot_confusion_matrix(y_train.ravel(), np.
 ↪argmax(x_train_prediction,axis=1),figsize=(10,10))
```



[71]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

# accuracy score on training data
training_data_accuracy = accuracy_score(y_train.ravel(),np.
↪argmax(x_train_prediction,axis=1))
print('Accuracy score of training data : {}%\n'.
↪format(round(training_data_accuracy,5)*100))

#Classification Report Training
print('\n Classification Report in Training: \n',classification_report(y_train.
↪ravel(), np.argmax(x_train_prediction,axis=1)))

print('============================================================================\n============

# accuracy score on Test data
```

```
test_data_accuracy = accuracy_score(y_test.ravel(), np.argmax(y_pred,axis=1))
print('Accuracy score of test data : {}% \n'.
 ↪format(round(test_data_accuracy,5)*100))

#Classification Report Test
print('\n Classification Report in Training: \n',classification_report(y_test.
 ↪ravel(), np.argmax(y_pred,axis=1)))
```

```
Accuracy score of training data : 91.007%


 Classification Report in Training:
               precision    recall  f1-score   support

           1       0.94      0.91      0.93    192195
           2       0.86      0.89      0.87    190785
           3       0.90      0.87      0.89    190334
           4       0.93      0.97      0.95    189580

    accuracy                           0.91    762894
   macro avg       0.91      0.91      0.91    762894
weighted avg       0.91      0.91      0.91    762894

==========================================================================
==========================================================================
Accuracy score of test data : 88.151%


 Classification Report in Training:
               precision    recall  f1-score   support

           1       0.93      0.89      0.91     21340
           2       0.82      0.84      0.83     21263
           3       0.86      0.84      0.85     20938
           4       0.92      0.95      0.93     21225

    accuracy                           0.88     84766
   macro avg       0.88      0.88      0.88     84766
weighted avg       0.88      0.88      0.88     84766
```

[72]:
```
model.save('v3_model.h5')
```

[73]:
```
model.save_weights('v2_weights.h5')
```

[ ]:
```
```