Project: Investigate a Dataset No-show Appointment

Table of Contents

1.0 Introduction

This report is on exploratory analysis of data on no-show appointment from clinic in brazil(1) The aim of the project is to investigate the the no-show dataset and communicate the findings(2) as part of the learning process to demostrate the skills learn in the Data analysis Process under the ALX-T Data analyst nano degree and communicate the findings

1.1 Problem Statement

We have been presented with data on non-show appointment in a clinic to explore the characteristics of patients who fail to show up on their appointment date with the aim of helping the management of the clinic to better stratigize on its patient appointment system in order to reduce the number of failed appointments in the clinic.

1.2 Research questions

The folowing questions will quide our analysis;

> 1. What is the percentage of patients that failed to show on their appointment day?
> 2. Which gender has the highest proportion of patients that failed to show up for their appointment?
> 3. What age group has the most number of patients that failed to show up?
> 4. What proportion of patients recieved sms but failed to show up on thier appointment day?
> 5. Which day of the week has the most number of no-show?

1.3 Overview of the data

This dataset collects information from 100k medical appointments in Brazil and is focused on the question of whether or not patients show up for their appointment. A number of characteristics about the patient are included in each row.(3) see the table below for column discription;

Sn	Fields	Description
0	PatientID	Patient identification number
1	AppointmentID	Identification of each appointment
2	Gender	Gender of the patient (Male or Female)
3	ScheduledDay	The day someone called or registered the appointment, this is before
4	AppointmentDay	The day of the actual appointment, when they have to visit the doctor
5	Age	How old is the patient
6	Neighbourhood	Where the appointment takes place
7	Scholarship	Indicates whether or not the patient is enrolled in Brasilian welfare program
8	Hypertension	True or False whether a patient is hypertensive or not
9	Diabetesr	True or False whether a patient is diabetic or not
10	Alcoholism	True or False whether a patient is alcoholics or not
11	Handcap	Describes the level of handicap on on scale of 0-4
12	SMS_received	1 or more messages sent to the patient
13	No-show	No' if the patient showed up to their appointment, and 'Yes' if they did not show up. (Dependent variable)

- **Importing required python libraries**

In this project we will be importing the following python libraries:

> - OS : for changing of working directory
> - Pandas : for maipulation our dataframe
> - Numpy : for number manipulation
> - Matplotlib.pyplot : for visualization of data
> - Seaborn: also for visualization of data

In [2]:
```
# Use this cell to set up import statements for all of the packages that you
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbn
# magic word %matplotlib inline' is used so that our visualizations are plotted i
#he notebook. See this page for more: http://ipython.readthedocs.io/en/stable/int
%matplotlib inline
```

- **Set working directory and load data**

In [3]:
```
#set working directory for the project
os.chdir('C:/Users/USER/3D Objects/Udacity_Project_01')
#load data into dataframe 'df_appointment'
df_appointment=pd.read_csv('C:/Users/USER/3D Objects/Udacity_Project_01/data/nosh
```

2.0 Data Wrangling

In this section we will look at the structure of the data,its rows and column size, data type of the columns. Also we will check for possible null values, duplicate, inconsistent columnsnames, and distinct values for each column. We will then go further to clean the data by correcting the abnormalities in the data for a smooth analysis on the data.

2.1 General Properties

- **Reading in data**

In [4]:
```
# Load your data and print out a few lines. Perform operations to inspect data
#load data into dataframe 'df_appointment'
df_appointment=pd.read_csv('C:/Users/USER/3D Objects/Udacity_Project_01/data/nosh

#print out 2 lines of data
df_appointment.head(2)
```

Out[4]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Sc
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	

The table above shows a sample of our data set with two rows of data

In [5]:
```
#   This shows the number of rows and colunm in the data set, types and look for
df_appointment.shape
```

Out[5]: (110527, 14)

The data has 110527 rows (records) and 14 columns (fields)

In [6]:
```
# this code gives information about the data at a glance
df_appointment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64  
 2   Gender          110527 non-null  object 
 3   ScheduledDay    110527 non-null  object 
 4   AppointmentDay  110527 non-null  object 
 5   Age             110527 non-null  int64  
 6   Neighbourhood   110527 non-null  object 
 7   Scholarship     110527 non-null  int64  
 8   Hipertension    110527 non-null  int64  
 9   Diabetes        110527 non-null  int64  
 10  Alcoholism      110527 non-null  int64  
 11  Handcap         110527 non-null  int64  
 12  SMS_received    110527 non-null  int64  
 13  No-show         110527 non-null  object 
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

From the result above we have 110527 entries in the dataframe indexed fom 0-110526, with total number of column as 14. **Column**: shows the field or column names. **Non-Null Count**: shows the total values in the column that are not null. **Dtype**: Shows the data types of all the field/columns. We need to change Data type of ScheduledDay and AppointmentDay to Datetime. No-show to Boolean

In [7]:
```
#check the number of unique values in each column to see if there extra labels ap
df_appointment.nunique()
```

Out[7]:
```
PatientId          62299
AppointmentID     110527
Gender                 2
ScheduledDay      103549
AppointmentDay        27
Age                  104
Neighbourhood         81
Scholarship            2
Hipertension           2
Diabetes               2
Alcoholism             2
Handcap                5
SMS_received           2
No-show                2
dtype: int64
```

The table above shows the column names and the respective number of unique values in each column.

In [9]:
```
#check for null values in the data set
df_appointment.isnull().sum()
```

Out[9]:
```
PatientId         0
AppointmentID     0
Gender            0
ScheduledDay      0
AppointmentDay    0
Age               0
Neighbourhood     0
Scholarship       0
Hipertension      0
Diabetes          0
Alcoholism        0
Handcap           0
SMS_received      0
No-show           0
dtype: int64
```

The result above shows that there are no null values in any of the columns of the data set

In [9]:
```
#check for duplicates values in the data set
df_appointment.duplicated().sum()
```

Out[9]:
```
0
```

From the the result above there are no duplicates in the data

In [10]:
```
#check for white spaces in the column names
df_appointment.columns.str.contains('^\s+$')
```

Out[10]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False])
```

From the fresult above there are no white spaces in the column names

In [11]:
```
# This code gives the summary statistics of the data
df_appointment.describe()
```

Out[11]:

	PatientId	AppointmentID	Age	Scholarship	Hipertension	Diabetes
count	1.105270e+05	1.105270e+05	110527.000000	110527.000000	110527.000000	110527.000000
mean	1.474963e+14	5.675305e+06	37.088874	0.098266	0.197246	0.071865
std	2.560949e+14	7.129575e+04	23.110205	0.297675	0.397921	0.258265
min	3.921784e+04	5.030230e+06	-1.000000	0.000000	0.000000	0.000000
25%	4.172614e+12	5.640286e+06	18.000000	0.000000	0.000000	0.000000
50%	3.173184e+13	5.680573e+06	37.000000	0.000000	0.000000	0.000000
75%	9.439172e+13	5.725524e+06	55.000000	0.000000	0.000000	0.000000
max	9.999816e+14	5.790484e+06	115.000000	1.000000	1.000000	1.000000

A glance at the staticial summary of the variables shows that age has minimum value of -1 which is inappropriate as age value and maximum of 115 which is an outlayer consisering the mean and median values of age in the data set.

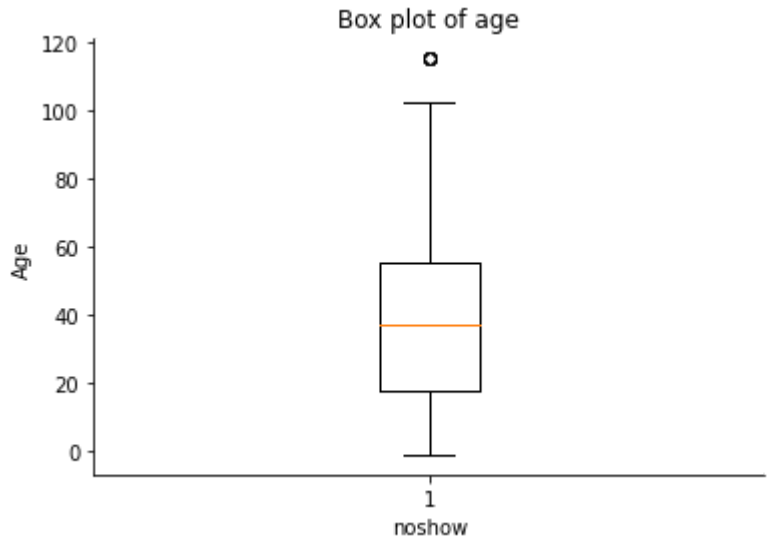
In [13]:
```
# check the distribution of age with boxplot
boxplt=plt.boxplot(df_appointment['Age'])

# Add major axis labels
plt.xlabel('noshow')
plt.ylabel('Age')

# Set boxplot title
plt.title('Box plot of age')
#show only left and bottom  borders
sbn.despine(top=True, right=True)

# Print the values of the outliers
print('Outlier in ages are :',[item.get_ydata() for item in boxplt['fliers']])
```

```
Outlier in ages are : [array([115, 115, 115, 115, 115], dtype=int64)]
```



From the boxplot above we see there are outlayers in the upper cap of the boxplot.

2.2 Data Cleaning of No-show Data Set

We will perform data cleaning activities like changing columns names to lowercase, triming white spaces in columns,changing data types to appropiate ones,creating new columns, and dropping columns and rows with outlayers and inapproriate entries.

- **cleaning columns**

In [14]:
```
#change column names to lowercase and remove "-" from column name for clearity
df_appointment.rename(columns=lambda x: x.strip().lower().replace("-", ""), inpla

#comfirm changes have been maded
df_appointment.columns
```

Out[14]:
```
Index(['patientid', 'appointmentid', 'gender', 'scheduledday',
       'appointmentday', 'age', 'neighbourhood', 'scholarship', 'hipertension',
       'diabetes', 'alcoholism', 'handcap', 'sms_received', 'noshow'],
      dtype='object')
```

Change data types

In [15]:
```
# Let us change the data types
df_appointment['scholarship'].replace({ 1: True, 0: False, }, inplace=True) #char
df_appointment['hipertension'].replace({ 1: True, 0: False, }, inplace=True)#char
df_appointment['diabetes'].replace({ 1: True, 0: False, }, inplace=True)    #char
df_appointment['alcoholism'].replace({ 1: True, 0: False, }, inplace=True)   #cha
df_appointment['sms_received'].replace({ 1: True, 0: False, }, inplace=True)
df_appointment['noshow'].replace({ 'No': True, 'Yes': False, }, inplace=True)  #c
df_appointment['age'].astype(int)      #change data type to integer
df_appointment['appointmentday'] = pd.to_datetime(df_appointment['appointmentday'
df_appointment['scheduledday'] = pd.to_datetime(df_appointment['scheduledday']) #
df_appointment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   patientid       110527 non-null  float64
 1   appointmentid   110527 non-null  int64
 2   gender          110527 non-null  object
 3   scheduledday    110527 non-null  datetime64[ns, UTC]
 4   appointmentday  110527 non-null  datetime64[ns, UTC]
 5   age             110527 non-null  int64
 6   neighbourhood   110527 non-null  object
 7   scholarship     110527 non-null  bool
 8   hipertension    110527 non-null  bool
 9   diabetes        110527 non-null  bool
 10  alcoholism      110527 non-null  bool
 11  handcap         110527 non-null  int64
 12  sms_received    110527 non-null  bool
 13  noshow          110527 non-null  bool
dtypes: bool(6), datetime64[ns, UTC](2), float64(1), int64(3), object(2)
memory usage: 7.4+ MB
```

we can see that the data types have been changed successfully

We can comfirm from the result above that all column names have been changed to lowercase letters

Drop rows with inappropriate entries

In [16]:
```
# let us view the rows that has -1 and 115 as age
df_appointment.query('(age== 115)| (age== -1)')

#let us see the percentage of the rows we want to drop
age_wrong= df_appointment.query('(age== 115)| (age== -1)').value_counts().sum() #
percent=age_wrong/len(df_appointment)*100
print("percentage to drop=",percent,"%") #print the value on the screen
```

```
percentage to drop= 0.005428537823337284 %
```

we can see that the percentage to be dropped is insignificant so, we will go ahead and drop the rows

In [17]:
```
# drop this row as it will not significantlly affect our analysis
df_appointment.drop(df_appointment.query('(age== 115)| (age== -1)') .index, inpla

#Comfirm that the row has been drop
df_appointment.query('(age== 115)| (age== -1)')
```

Out[17]:

patientid	appointmentid	gender	scheduledday	appointmentday	age	neighbourhood	scholarshi

We can comfirm from the table above that the rows with inappropriate values have been dropped.

Create new columns We need to create three new Columns in order to enable us answer our research questions efficently. The Columns are;

> **agegroup :** This variable contains the grouping age into age-groups by using Minimum, 25%, 50%, 75% and maximum summary values from our age column.
> **weekday :** This variable contains day of week, like monday, tuesday, wednessday, etc. from the appointmentday column.(4)

In [18]:
```
#create new column with name "weekday" from the appointmentday column and set the
df_appointment['weekday'] = df_appointment['appointmentday'].dt.day_name().astype
wkday=df_appointment['weekday'].value_counts()

#Create column "agegroup" with label names as label and bin cut as bins_d
label=["0-17","18-36","37-54","55-above"]
bins_d =[0,18,37,55,102]
df_appointment["agegroup"]=pd.cut(x= df_appointment["age"], right=True, bins=bins

#comfirm that the two column are created with the right data types and the columr
df_appointment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110521 entries, 0 to 110526
Data columns (total 16 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   patientid       110521 non-null  float64
 1   appointmentid   110521 non-null  int64
 2   gender          110521 non-null  object
 3   scheduledday    110521 non-null  datetime64[ns, UTC]
 4   appointmentday  110521 non-null  datetime64[ns, UTC]
 5   age             110521 non-null  int64
 6   neighbourhood   110521 non-null  object
 7   scholarship     110521 non-null  bool
 8   hipertension    110521 non-null  bool
 9   diabetes        110521 non-null  bool
 10  alcoholism      110521 non-null  bool
 11  handcap         110521 non-null  int64
 12  sms_received    110521 non-null  bool
 13  noshow          110521 non-null  bool
 14  weekday         110521 non-null  category
 15  agegroup        106982 non-null  category
dtypes: bool(6), category(2), datetime64[ns, UTC](2), float64(1), int64(3), obj
ect(2)
memory usage: 8.4+ MB
```

We can see above that the two new columns are created and the columns are reorderd neatly in the order specified.

- **Filter the data**
  We have to sellect a subset of our data to include only values of nonshow ==True and columns weekday, gender, agegroup, and sms_recieved will be sellected as these are the variables that relate to our research questions. This dataframe will be denoted as df_noshow and will be used to answer our research questions as we are more concerned about people that fail to show up on thier apppointment date.

In [19]:
```
#Select only the data for noshow column equal to False i.e those that failed to s
df_noshow=df_appointment[df_appointment.noshow== False]

#drop columns that are not usefull in our analysis
df_noshow=df_noshow.drop(columns=['patientid','appointmentid','scheduledday','app
       'scholarship'])
#reorder column names to be more cohesive
df_noshow=df_noshow[['weekday','gender','agegroup','sms_received','noshow']]
#comfirm that only the desired columns and values are selected
df_noshow.nunique()
```

Out[19]:
```
weekday         6
gender          2
agegroup        4
sms_received    2
noshow          1
dtype: int64
```

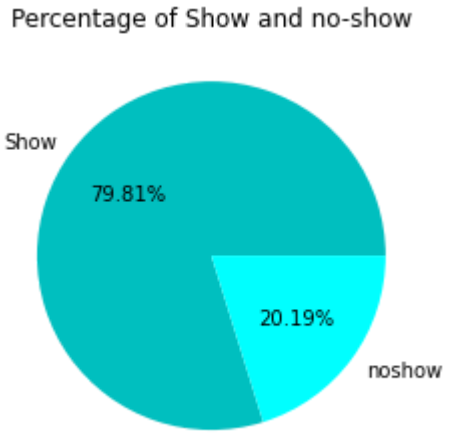
From the result above, we can see that only data on people who failed to show up for appointment with weekday, gender, agegroup, and sms_recieved are selected.

3.0 Exploratory Data Analysis

In this section we will be providing answers to our research questions by exploring the data and providing relevant visualizations(5) to support it.

Research Question 1: What is the percentage of those who fail to show up?

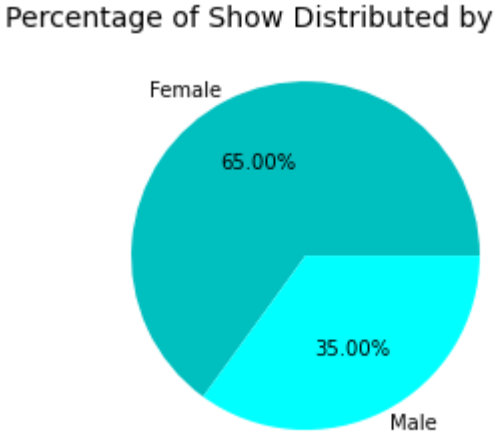
In [20]:
```
# The code gives us the percentage of those who fail to turn up
fig1, tmy = plt.subplots()
label= ["Show","noshow"]
#plot and show percentage values on the plot
tmy.pie(df_appointment['noshow'].value_counts(), labels=label,colors = ['c', 'cya
tmy.set_title('Percentage of Show and no-show');
```



From the Pie chart above, the percentage of those that fail to show up on thier appointment day is 20.19% while those that showed up is 79.81%.

Research Question 2 :Which gender has the highest proportion of patients that failed to show up for their appointment?

In [21]:
```
fig2, tmy = plt.subplots()
label= ["Female","Male"]
#plot and show percentage values on the plot [wert](1)
tmy.pie(df_appointment['gender'].value_counts(), labels=label,labeldistance=1.07,
tmy.set_title('Percentage of Show Distributed by Gender', size=14);
```

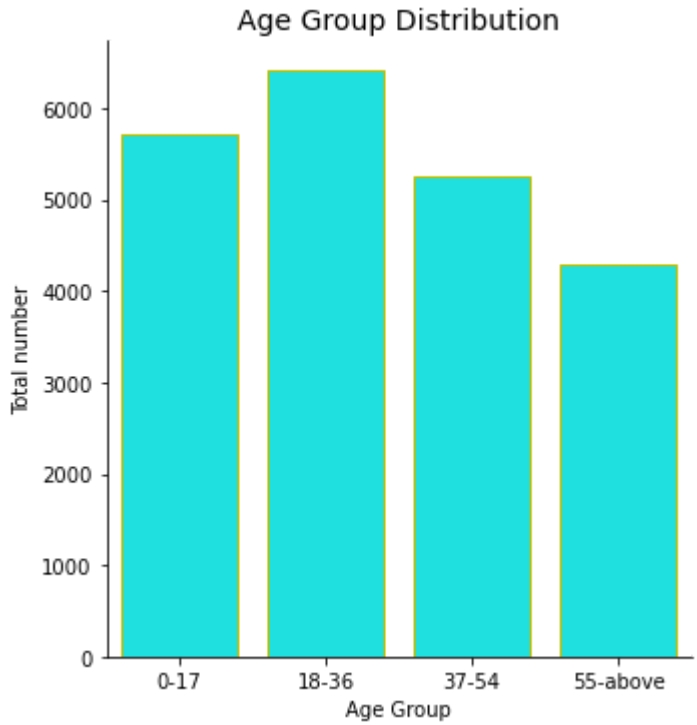


From the pie chart above 35% of those that failed to show up are male patients while 65% are female.

Research Question 3 : What age group has the most number of patients that failed to show up?

In [22]:
```
# Make the bar plot for count of age group with bar color = cyan,linewidth =1 and
sbn.catplot(x="agegroup", kind="count", color='cyan',linewidth=1,
                    edgecolor="y", data=df_noshow)

# set graph title with font size 14
plt.title("Age Group Distribution",size=14)
# set x and y axis labels
plt.xlabel("Age Group")
plt.ylabel("Total number");
```

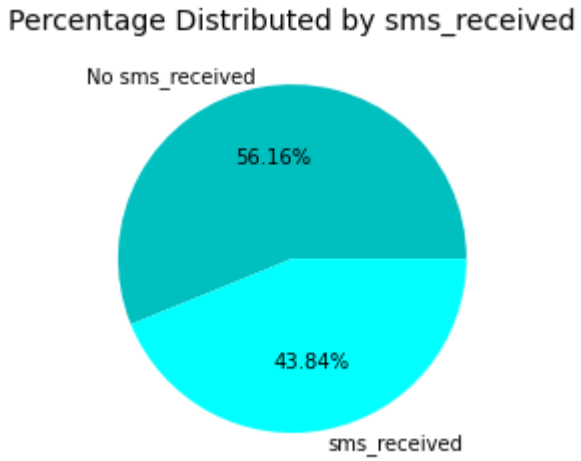


From the Bar chart above we can see that majority of the patients that fail to turn up are between 18-36 years, followed by 0-17 years, then 37-54 years and finally 55 years and above.

Research Question 4: What proportion of patients recieved sms but failed to show up on their appointment day?

In [23]:

```
# Make the bar plot for age distribution
fig1, tmy = plt.subplots()
label= ["No sms_received","sms_received"]
#plot and show percentage values on the plot
tmy.pie(df_noshow['sms_received'].value_counts(),labels=label, labeldistance=1.07
        colors = ['c', 'cyan'],autopct='%1.2f%%')
tmy.set_title('Percentage Distributed by sms_received', size=14);
```

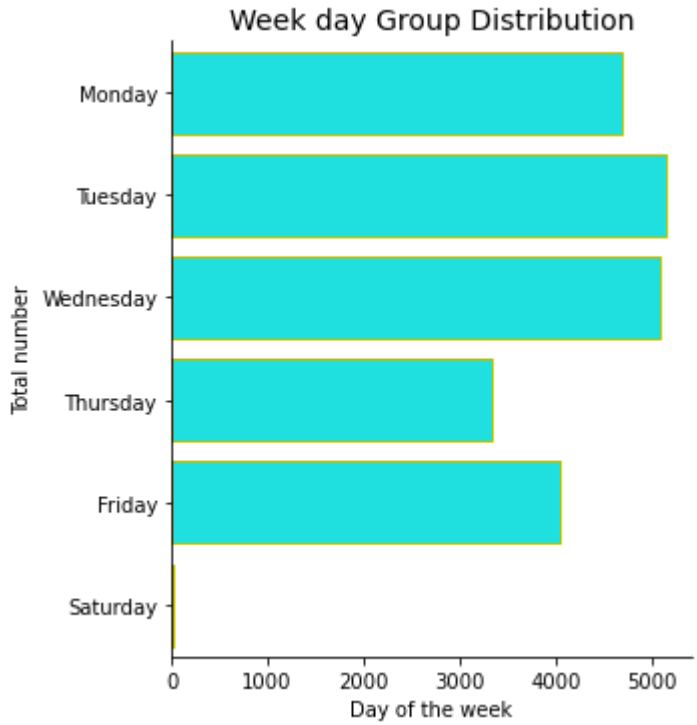


From the bar Chart above we can see that only 43.84% of patients received sms but failed to turn up on thier appointment day while 56.16% did not receive sms

Research Question 5: Which day of the week has the most number of no-show?

In [24]:
```
# Make the bar plot for count of day of the week with bar color = cyan,linewidth
# and bar edge color=yellow, then order by monday to saturday
sbn.catplot(y="weekday", kind="count", color='cyan',linewidth=1, edgecolor="y",
            order=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'])

# set graph title with font size 14
plt.title("Week day Group Distribution",size=14)
# # set x and y axis labels
plt.xlabel("Day of the week")
plt.ylabel("Total number");
```



From the Bar char above tuesdays has the highest number of no-show appointment,closely followed by wednessdays,then mondays, fridays and saturdays respectively.

4.0 Conclusions

4.1 Summary of findings

from the analysis of the data we discovered that the proportion of patients that fails to show up on thier appoint date is 20.19%. Majority of the Proportion are female who are in the age of 18-36 years of age, majority of which did not receiving sms reminders and mostly fail to show up on mondays to wednesdays.

4.2 Limitations

This analysis is limited to only exploring sellected attributes of patients that failed to show up on their appoint date and did not perfoms predictive analysis to focast patients behaviour over time because of limited number of years (2 years) which the data was collected.

4.3 Recommendation

From our finding the following recommendation could be made;

> i. The clinic should create awareness on the importance of not missing appointment dates, focusing more females patients who between 18-36 years of age.
> ii. The Clinic should intensify its sms reminders to patients to possibly reduce the number of no-show.

5.0 Reference Materials

1. Bolsa Família (https://en.wikipedia.org/wiki/Bolsa_Fam%C3%ADlia)
2. Alx-Udacity Project1 description (https://classroom.udacity.com/nanodegrees/nd002-alg-t2/parts/cd0000/modules/306f0239-bb80-45c6-bf45-37ee745a63d6/lessons/ls0526/concepts/ac0dd93f-dca2-420f-9395-8c6314443515)
3. Medical Appointment No Shows (https://www.kaggle.com/datasets/joniarroba/noshowappointments)
4.how do i get the day of week given a date (https://stackoverflow.com/questions/9847213/how-do-i-get-the-day-of-week-given-a-date)
5.Ploting categorical data with seaborn catplot() (https://seaborn.pydata.org/tutorial/categorical.html)

Back to top