Medical Appointment No Shows

Why do 30% of patients miss their scheduled appointments?

Dataset Description

The dataset collects information from more than 100k medical appointments in Brazil and is focused on the question of whether or not patients show up for their appointment. The Project Medical Appointments No Shows is investigating Medical Appointment No Shows dataset which contains historical data for more than 110K appointments made across different medical facilities in Brazil for more than 60k patients, for each record there are 14 Variables, metadata related to appointments date, patients gender, age, medical condition, social support coverage and facilities and 1 TARGET variable "Wither the patient attended the appointment or not.

Project Target

What is the explanation for a person making a doctor appointment, receives all the instructions and no-show. Who to blame?

Data Definition

PatientId: Identification of a patient that is unique for each person.	Age: How old is the patient.	Alcoholism: T or F.
AppointmentID: Identification of each appointment.	Neighborhood: Where the appointment takes place.	Handcap: T or F.
Gender: Male or Female.	Scholarship: 1 or 0 (this is a program in Brazil to support poor people with their cost of living),	SMS_received: 1 or more messages sent to the patient.
AppointmentDay: The day of the actual appointment, when they have to visit the doctor.	Hipertension: T or F.	No-show: "Yes" or "No" ("No" means they showed up on their appointments while "Yes" means they didn't!).
ScheduledDay: The day someone called or registered the appointment, this is before appointment of course.	Diabetes: T or F.	

Data Definition

```
In [4]: df.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
```

```
# Outcomes:
- This shows that we have 14 columns in total besides index, and we have 110527 record.
- None of the data fields have NULL values.
- Datatype should be changed for both "ScheduledDay" and "AppointmentDay".
- PatientId is float while it supposed to be an integer.
```

In [5]:
```
# View column data types
df.dtypes
```

Out[5]:
```
PatientId          float64
AppointmentID        int64
Gender              object
ScheduledDay        object
AppointmentDay      object
Age                  int64
Neighbourhood       object
Scholarship          int64
Hipertension         int64
Diabetes             int64
Alcoholism           int64
Handcap              int64
SMS_received         int64
No-show             object
dtype: object
```

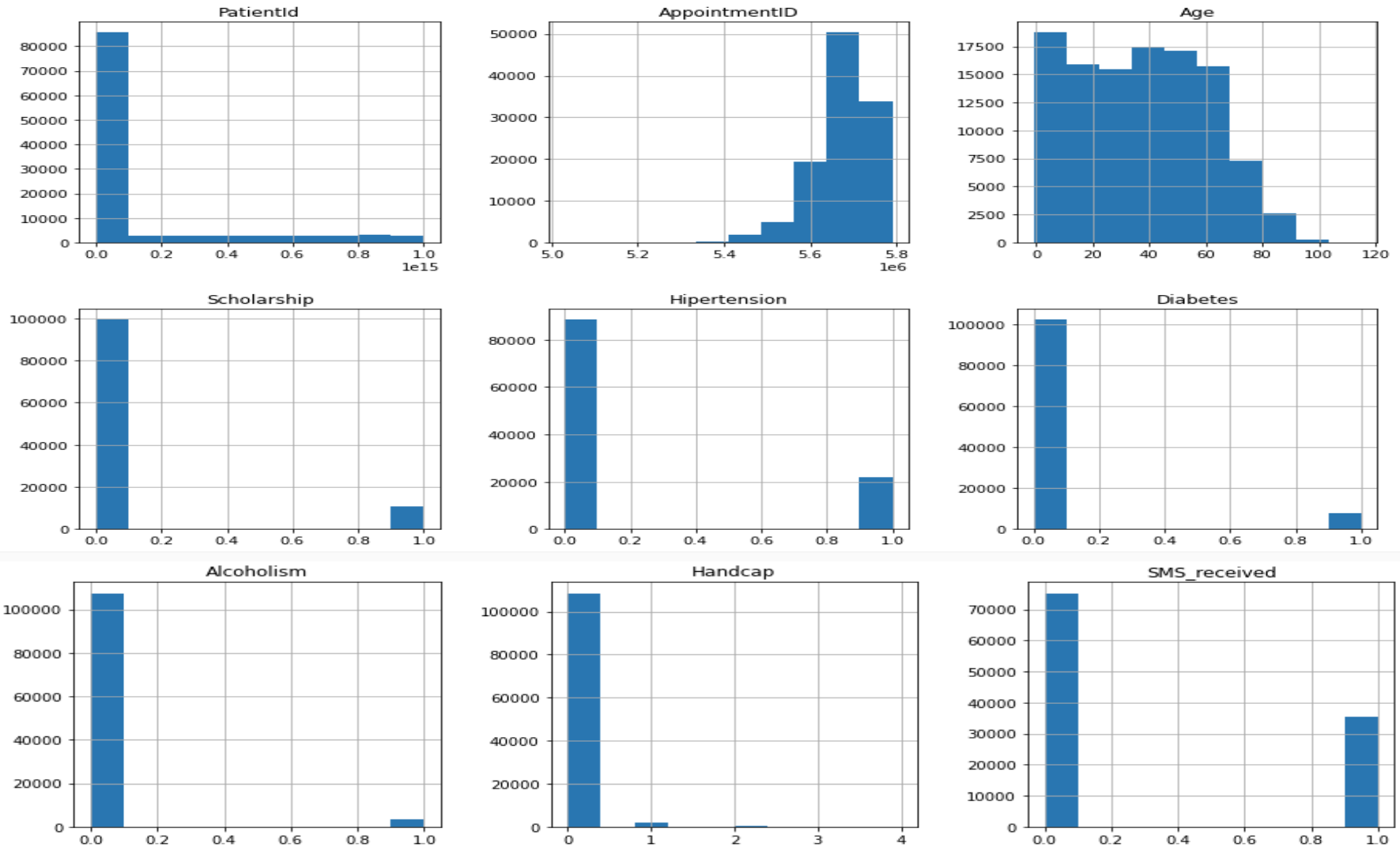
In [6]:
```
# Check Duplicates
df.duplicated().sum()
```

Out[6]:
```
0
```

Outcomes:

- There is no duplicate rows in this dataset.

Histograms shows count of each category for different features



Outcomes:

- AppointmentId and PatientId have no statistical meanings that's why we need to change their datatype into strings.
- Age date is right skewed wich means data includes more young patients than old!
- Percentage of Alcoholisim and Handcap is extremly low among patients in this dataset.
- Percentage of patients who have diabetes and patients who have a scholarship doesn't exceed 5%.
- Percentage of patients who have hipertension as around 23% which is noticable.
- SMS have been sent to more than 30% of cases.

Data Cleaning

- Edit the "No-show" column to be in positive form instead of negativity.

```
In [9]: # Edit the reversed meaning of No show to be positive,
        # 0 will mean that patient didn't come to his appointement.
        # 1 will mean that patient came to his appointment.

        df["No-show"].replace({"Yes":0,"No":1},inplace=True)

In [10]: df["No-show"] = pd.to_numeric(df["No-show"]) # Change datatype for column to be numeric integer

In [11]: df=df.rename(columns={"No-show":"Show_up"}) # Rename column to give positive meaning.

In [12]: df.info() # Make sure of that column name changed correctly and also datatype.

<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  Show_up         110527 non-null  int64
dtypes: float64(1), int64(9), object(4)
memory usage: 11.8+ MB
```

Data Cleaning

- Edit the "ScheduledDay", and "AppointmentDay" columns' datatype to be Datetime.

```
In [13]: df["ScheduledDay"] = pd.to_datetime(df["ScheduledDay"])

In [14]: df["AppointmentDay"] = pd.to_datetime(df["AppointmentDay"])

In [15]: df.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  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  Show_up         110527 non-null  int64
dtypes: datetime64[ns, UTC](2), float64(1), int64(9), object(2)
memory usage: 11.8+ MB
```

```
In [16]: df.head()
```

Out[16]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SM
0	2.987250e+13	5642903	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	JARDIM DA PENHA	0	1	0	0	0	
1	5.589978e+14	5642503	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	JARDIM DA PENHA	0	0	0	0	0	

Data Cleaning

- Edit the "PatientId", "AppointmentID" columns' datatype to be String.

```
In [17]: df['PatientId'] = df['PatientId'].apply(str)

In [18]: df['AppointmentID'] = df['AppointmentID'].apply(str)

In [19]: df.info() # Make sure of datatype changed.

<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  object
 1   AppointmentID   110527 non-null  object
 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  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  Show_up         110527 non-null  int64
dtypes: datetime64[ns, UTC](2), int64(8), object(4)
memory usage: 11.8+ MB

In [20]: df.describe(include="all") # Check for results
```

Data Cleaning

- Remove Row with negative age Value.

```
In [21]: df[df["Age"]<0] # show colmn with this problem
```

Out[21]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism	Han
99832	465943158731293.0	5775010	F	2016-06-06 08:58:13+00:00	2016-06-06 00:00:00+00:00	-1	ROMÃO	0	0	0	0	

```
In [22]: df = df.drop([99832]) # Drop the column with index=99832
```

```
In [23]: df[df["Age"]<0] # Make sure that row has been dropped
```

Out[23]:

PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap	SMS_rec

```
In [ ]:
```

Algorithms

- Classification> Logistic Regression Model.
- Regression > Linear Regression Model.

Tools

- **Pandas**: a library offers data structures and operations for manipulating numerical tables and time series.
- **Numpy**: a library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- **Matplotlib**: a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- **Seaborn**: a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python

```
In [1]: # Import all libraries needed in analysis
        import pandas as pd # data processing, CSV file - Dataframe
        import numpy as np # linear algebra - Arraies
        import matplotlib.pyplot as plt # plotting - Visualization
        import seaborn as sns # Visualization
        from datetime import datetime
        %matplotlib inline
```

Questions for Analysis

- What is the percentage of patients who show up on their appointments vs. who not?

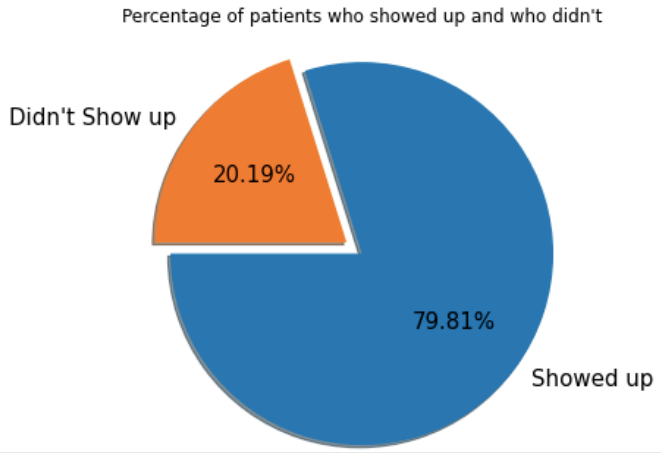
In [24]:
```
Num_show_up = df[df["Show_up"]==1].count()["PatientId"] # Count of patients who showed up on thier appoinments.
Num_no_show_up = df[df["Show_up"]==0].count()["PatientId"] # Count of patients who didn't show up on thier appoinments.
```

In [25]:
```
print(Num_show_up,Num_no_show_up)
```

```
88207 22319
```

In [26]:
```
label_Names = ["Showed up", "Didn't Show up"]
data = [Num_show_up, Num_no_show_up]


explode = (0, 0.15) # Only explode the didn't show up slice.
plt.axis('equal'); # To keep aspect ratio equal to appear as a fine circle.
plt.pie(data,radius=1.5,shadow=True ,labels = label_Names,explode=explode, startangle=180,autopct='%0.2f%%',textprops =
plt.title("Percentage of patients who showed up and who didn't",y=1.2);
# Autopct to show percentage, 0.2 for two decimal place
```



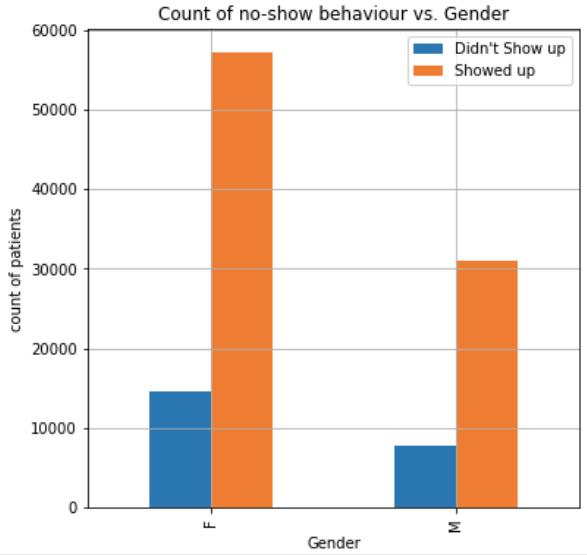
Questions for Analysis

- Is one gender more committed to medical schedules than another?

Gender	Show_up											
F	0	14594	14594	14594	14594	14594	14594	14594	14594	14594	14594	14594
	1	57245	57245	57245	57245	57245	57245	57245	57245	57245	57245	57245
M	0	7725	7725	7725	7725	7725	7725	7725	7725	7725	7725	7725
	1	30962	30962	30962	30962	30962	30962	30962	30962	30962	30962	30962

In [28]:
```
x["PatientId"].unstack().plot(kind="bar",stacked=False,figsize=(6,6),grid=True) # Plot for the grouped dataframe
plt.title("Count of no-show behaviour vs. Gender") # Change title for the plot
plt.legend(["Didn't Show up", "Showed up"]); # Change legend for each color of data
plt.ylabel("count of patients")
```

Out[28]:
```
Text(0, 0.5, 'count of patients')
```



Questions for Analysis

- Where do most appointments take place?

```
In [35]: df.describe(include="all")
```

```
<ipython-input-35-7150bae0ffd8>:1: FutureWarning: Treating datetime data as categorical rather than numeric in `.desc
ribe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence
this warning and adopt the future behavior now.
  df.describe(include="all")
<ipython-input-35-7150bae0ffd8>:1: FutureWarning: Treating datetime data as categorical rather than numeric in `.desc
ribe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence
this warning and adopt the future behavior now.
  df.describe(include="all")
```

Out[35]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabet
count	110526	110526	110526	110526	110526	110526.000000	110526	110526.000000	110526.000000	110526.0000
unique	62298	110526	2	103548	27	NaN	81	NaN	NaN	N
top	822145925426128.0	5770869	F	2016-05-06 07:09:54+00:00	2016-06-06 00:00:00+00:00	NaN	JARDIM CAMBURI	NaN	NaN	N
freq	88	1	71839	24	4691	NaN	7717	NaN	NaN	N
first	NaN	NaN	NaN	2015-11-10 07:13:56+00:00	2016-04-29 00:00:00+00:00	NaN	NaN	NaN	NaN	N
last	NaN	NaN	NaN	2016-06-08 20:07:23+00:00	2016-06-08 00:00:00+00:00	NaN	NaN	NaN	NaN	N
mean	NaN	NaN	NaN	NaN	NaN	37.089219	NaN	0.098266	0.197248	0.0718
std	NaN	NaN	NaN	NaN	NaN	23.110026	NaN	0.297676	0.397923	0.2582
min	NaN	NaN	NaN	NaN	NaN	0.000000	NaN	0.000000	0.000000	0.0000
25%	NaN	NaN	NaN	NaN	NaN	18.000000	NaN	0.000000	0.000000	0.0000
50%	NaN	NaN	NaN	NaN	NaN	37.000000	NaN	0.000000	0.000000	0.0000
75%	NaN	NaN	NaN	NaN	NaN	55.000000	NaN	0.000000	0.000000	0.0000
max	NaN	NaN	NaN	NaN	NaN	115.000000	NaN	1.000000	1.000000	1.0000

Questions for Analysis

- Are patients who received SMS messages reminding them of the appointment likely to attend?

```
print("Show_up percentage with SMS is {}%, while {}%  without SMS.".format(round(df_SMS*100,2),round(df_No_SMS*100,2)))
```

```
Show_up percentage with SMS is 79.75%, while 86.08%  without SMS.
```

In [49]:
```
# More than 65
df_SMS = df.loc[(df["SMS_received"]==1) & (df["Age"] >=65)]["Show_up"].mean() # Childs patients who recieved SMS
df_No_SMS = df.loc[(df["SMS_received"]==0) & (df["Age"]>=65)]["Show_up"].mean() # Childs patients who recieved SMS
print("Show_up percentage with SMS is {}%, while {}%  without SMS.".format(round(df_SMS*100,2),round(df_No_SMS*100,2)))
```

```
Show_up percentage with SMS is 80.31%, while 86.34%  without SMS.
```

Outcomes:

- it seems like the percentage of Show ups increase with SMS with increasing age.

In [50]:
```
# Let's Explore duration effect
median_duration=df.groupby("Show_up").mean()["Duration"][0]
```

In [51]:
```
# In case of low durations:
df_SMS = df.loc[(df["SMS_received"]==1) & (df["Duration"]<=median_duration)]["Show_up"].mean() #Childs patients who rec
df_No_SMS = df.loc[(df["SMS_received"]==0) & (df["Duration"]<=median_duration)]["Show_up"].mean() #Childs patients who
print("Show_up percentage with SMS is {}%, while {}%  without SMS.".format(round(df_SMS*100,2),round(df_No_SMS*100,2)))
```

```
Show_up percentage with SMS is 74.33%, while 86.52%  without SMS.
```

In [52]:
```
# In case of large durations:
df_SMS = df.loc[(df["SMS_received"]==1) & (df["Duration"]>=median_duration)]["Show_up"].mean() #Childs patients who rec
df_No_SMS = df.loc[(df["SMS_received"]==0) & (df["Duration"]>=median_duration)]["Show_up"].mean() #Childs patients who
print("Show_up percentage with SMS is {}%, while {}%  without SMS.".format(round(df_SMS*100,2),round(df_No_SMS*100,2)))
```

```
Show_up percentage with SMS is 70.11%, while 62.9%  without SMS.
```

Outcomes:

- Percentage of patients who shows up increases when the duration is larger.

Questions for Analysis

- What is the percentage of patients diagnosed with diabetes, hypertension, alcoholism and disability?

```
 10  Alcoholism      110526 non-null  int64
 11  Handcap         110526 non-null  int64
 12  SMS_received    110526 non-null  int64
 13  Show_up         110526 non-null  int64
 14  Duration        110526 non-null  int64
dtypes: datetime64[ns, UTC](2), int64(9), object(4)
memory usage: 13.5+ MB
```

In [54]:
```
all_count = 110526 # Number of data records
```

In [55]:
```
# Percentage of people diagnosed with Diabetes
diabets_count = df[df["Diabetes"]==1]["PatientId"].count()
diabetes_percent = round(diabets_count*100/all_count,2)
print("Percentage of patients who diagnosed with Diabetes is {}%.".format(diabetes_percent))
```

```
Percentage of patients who diagnosed with Diabetes is 7.19%.
```

In [56]:
```
# Percentage of people diagnosed with Diabetes
diabets_count = df[df["Hipertension"]==1]["PatientId"].count()
diabetes_percent = round(diabets_count*100/all_count,2)
print("Percentage of patients who diagnosed with Hipertension is {}%.".format(diabetes_percent))
```

```
Percentage of patients who diagnosed with Hipertension is 19.72%.
```

In [57]:
```
# Percentage of people diagnosed with Diabetes
diabets_count = df[df["Alcoholism"]==1]["PatientId"].count()
diabetes_percent = round(diabets_count*100/all_count,2)
print("Percentage of patients who diagnosed with Alcoholism is {}%.".format(diabetes_percent))
```

```
Percentage of patients who diagnosed with Alcoholism is 3.04%.
```

In [58]:
```
# Percentage of people diagnosed with Diabetes
diabets_count = df[df["Handcap"]==1]["PatientId"].count()
diabetes_percent = round(diabets_count*100/all_count,2)
print("Percentage of patients who diagnosed with Handcap is {}%.".format(diabetes_percent))
```

```
Percentage of patients who diagnosed with Handcap is 1.85%.
```

Questions for Analysis

- Is drinking alcohol a cause of missing appointments?

```
In [59]: # Count of patients with alchoholism who show up
         alcohol_show = df.loc[(df["Alcoholism"]==1) & (df["Show_up"]==1)]["PatientId"].count()

In [60]: # Count of patients with alchoholism who don't show up
         alcohol_No_show = df.loc[(df["Alcoholism"]==1) & (df["Show_up"]==0)]["PatientId"].count()

In [61]: # Clculate percentages
         alcohol_show_percent = round(alcohol_show*100 / all_count,2)
         alcohol_No_show_percent = round(alcohol_No_show*100 / all_count,2)

In [62]: print("Percentage of show ups when patients have alchoholism is {}%, while not show ups is {}%.".format(alcohol_show_pe

Percentage of show ups when patients have alchoholism is 2.43%, while not show ups is 0.61%.
```

Answer

- It doesn't make them missing the appointments as most of them didn't miss theirs.

Questions for Analysis

- Is the duration between registration and appointment affect the ability to show up?

4- Is the duration between registration and appointment affect the ability to show up?

In [36]:
```
# To answer this question we need to add new column contain the duration between regestiration and appointment
df["Duration"] = (df["AppointmentDay"].dt.date) - (df["ScheduledDay"].dt.date) #.dt.date to differ only date part
```

In [37]:
```
df["Duration"] = df["Duration"].dt.days # To convert column to numerical column contain number of days
```

In [38]:
```
df.head() # Show the new column
```

Out[38]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism	Handcap
0	29872499824296.0	5642903	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	JARDIM DA PENHA	0	1	0	0	0
1	558997776694438.0	5642503	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	JARDIM DA PENHA	0	0	0	0	0
2	4262962299951.0	5642549	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	MATA DA PRAIA	0	0	0	0	0
3	867951213174.0	5642828	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	PONTAL DE CAMBURI	0	0	0	0	0
4	8841186448183.0	5642494	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	JARDIM DA PENHA	0	1	1	0	0

In [39]:
```
df.groupby("Show_up").mean()["Duration"]
```

Out[39]:
```
Show_up
0    15.831489
1     8.754759
Name: Duration, dtype: float64
```

Answer:

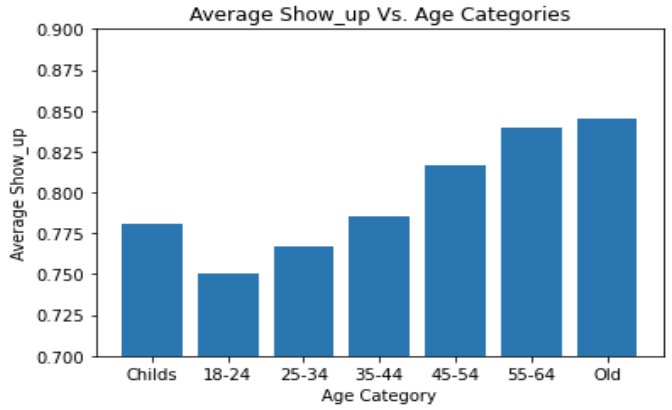
- Patients Who didn't show up have an average of 15 days between registeration day and their appointments.
- Patients Who show up have an average of 8 days between registeration day and their appointments.
- As Duration increases, the ability of patients to show up on their appointments decreases.

Questions for Analysis

- Do older patients more committed to medical schedules than others?

In [65]:
```
# Calculate mean for each age category
mean_childs = childs["Show_up"].mean()
mean_18_24 = age_18_24["Show_up"].mean()
mean_25_34 = age_25_34["Show_up"].mean()
mean_35_44 = age_35_44["Show_up"].mean()
mean_45_54 = age_45_54["Show_up"].mean()
mean_55_64 = age_55_64["Show_up"].mean()
mean_old = old["Show_up"].mean()
```

In [66]:
```
# Draw Bar plot for the data
locations = [1, 2, 3, 4, 5, 6, 7]
heights = [mean_childs, mean_18_24, mean_25_34, mean_35_44, mean_45_54, mean_55_64, mean_old]
labels = ["Childs","18-24","25-34", "35-44", "45-54", "55-64", "Old"]
plt.bar(locations, heights, tick_label=labels)
plt.title('Average Show_up Vs. Age Categories')
plt.ylim([0.7,0.9]) #to focus on this spot to feel the change
plt.xlabel('Age Category')
plt.ylabel('Average Show_up');
```



Results

- Percentage of patients who show up on their appointments represents 79.8%
- Percentage of patients who Don't show up on their appointments represents 20.2%-
- Both genders have same commitment to medical schedules.
- "JARDIM CAMBURI" is the most frequent place.
- Patients Who didn't show up have an average of 15 days between registration day and their appointments.
- Patients Who show up have an average of 8 days between registration day and their appointments.
- As Duration increases, the ability of patients to show up on their appointments decreases.
- Older patients are more committed to their appointments' schedules than younger ones.

Limitations

- Source of data should record timing of sending SMS to Patients to better investigate effect of this on Show ups and why it is not effective as much as expected.
- Most of the data are categorical which made most of plots are meaningless.
- Some patients who marked as no show up, in real they may show up but on another day, this is shown on a different record but this will be better noticed and enhance result if recorded that patient has made a reschedule instead of being recorded as No show.

The End

Thank you