Medical-appointment-dataset-analysis

Dataset Description

A person makes a doctor appointment, receives all the instructions and no-show. Who to blame? 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.

Dataset Description

A person makes a doctor appointment, receives all the instructions and no-show. Who to blame? 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.

Columns Description

PatientId: Identification of a patient.

AppointmentID: Identification of each appointment.

Gender: Male or Female.

AppointmentDay: The day of the actuall appointment, when they have to visit the doctor.

ScheduledDay: The day someone called or registered the appointment, this is before appointment of course.

Age: How old is the patient.

Neighbourhood: Where the appointment takes place.

Scholarship: True of False, indicates whether or not the patient is enrolled in Brasilian welfare program Bolsa Família.

Hipertension: True or False.

Diabetes: True or False.

Alcoholism: True or False.

Handcap: True or False.

SMS_received: 1 or more messages sent to the patient.

No-show: True (if the patient did not show up), or False (if the patient did show up).

EDA Questions

Q1: How often do men go to hospitals compared to women? Which of them is more likely to show up?

Q2: Does recieving an SMS as a reminder affect whether or not a patient may show up? is it correlated with number of days before the appointment?

Q3: Does having a scholarship affects showing up on a hospital appointment? What are the age groups affected by this?

Q4: Does having certain deseases affect whather or not a patient may show up to their appointment? is it affected by gender?

Environment set-up

```
# importing lib.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# getting the csv file directory
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Data Wrangling

in this section, we'd load our data from a CSV file to a pandas dataframe, and then take a quick dive into exploring our dataset in details.

```
# loading dataset from csv file and showing its first 5 rows
df = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
df.head()
```

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighl
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	J/
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	J/
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	P(
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	J/

We'll move next into exploring our dataset by going through its data types, NaNs or duplicated rows, and any columns that may need to be dropped or parsed.

```
# viewing main info about df
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
```

We can notice there are no NaNs at all in our data

PatientId and AppointmentId columns wouldn't be helpful during analysis.

ScheduledDay and AppointmentDay needs to be casted to date data type.

we may append a new column for days until appointment.

Gender needs to be converted into a categoy type

Scholarship Hipertension Diabetes Alcoholism Handcap better be boolean data type.

No-show needs to be parsed and casted to boolean too.

```
# checking for duplicates
df.duplicated().sum()
```

```
0
```

our dataset has no duplicated rows either.

```
# exploring the unique values of each column
df.nunique()
```

	0
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

Handcap and Age columns has inconsistant unique values. SMS_received would be casted to boolean data type.

```
# exploring handcap values
df['Handcap'].value_counts()
```

	count
Handcap	
0	108286
1	2042
2	183
3	13
4	3

dtype: int64

we'd be only intrested in rows with 0 or 1 values.

```
# exploring age column distribution
df['Age'].describe()
```

	Age
count	110527.000000
mean	37.088874
std	23.110205
min	-1.000000
25%	18.000000
50%	37.000000
75%	55.000000
max	115.000000

dtype: float64

Age column would need to be handled.

Exploration Summery

our dataset consists of 110527 rows with 14 columns, and has no NaNs nor duplicated values.

PatientId and AppointmentId columns wouldn't be helpful during analysis.

ScheduledDay and AppointmentDay needs to be casted to date data type.

we may append a new column for days until appointment.

Gender needs to be casted into a categoy type

Scholarship, Hipertension, Diabetes, Alcoholism and SMS_recieved better be boolean data type.

No-show column needs to be parsed and asted to boolean type.

Handcap colume needs to be cleaned to have only 0 and 1 values.

Age columns has inconsistant unique values that needs to be handled.

Data Cleaning

in this section, we'd perform some operations on our dataset based on the previous findings to make our analysis more accurate and clear.

Dropping PatientId and AppointmentId columns

```
columns_to_drop = ['PatientId', 'AppointmentID']
df.drop(columns=[col for col in columns_to_drop if col in df.columns],
df.columns
```

```
Index(['Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Neighbourhood',
       'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'Handcap',
       'SMS_received', 'No-show'],
      dtype='object')
```

Handling date data type

```
df.AppointmentDay.unique
```

```
pandas.core.series.Series.unique
def unique() -> ArrayLike
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/series.py
Return unique values of Series object.

Uniques are returned in order of appearance. Hash table-based unique,
therefore does NOT sort.
```

it looks like all hours are set to 00:00:00, so we would want to extract onl the year, month and day data

```
# extracting only day, month and year values
df['ScheduledDay'] = df['ScheduledDay'].str[:10]
df['AppointmentDay'] = df['AppointmentDay'].str[:10]

# changing data type
df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])

# confirming changes
print(df[['AppointmentDay', 'ScheduledDay']].dtypes)
df.head()
```

```
AppointmentDay    datetime64[ns]
ScheduledDay      datetime64[ns]
dtype: object
```

	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hiper
0	F	2016-04-29	2016-04-29	62	JARDIM DA PENHA	0	
1	M	2016-04-29	2016-04-29	56	JARDIM DA PENHA	0	
2	F	2016-04-29	2016-04-29	62	MATA DA PRAIA	0	

Now, we'd move into appending a new column that holds number of days to the appointment.

```
# making new due days column
df['due-days'] = df['AppointmentDay'] - df['ScheduledDay']

# converting data type
df['due-days'] = df['due-days'].dt.days

# drop sch and appoint col
df.drop(['AppointmentDay', 'ScheduledDay'], axis = 1, inplace = True)
```

We'll move into exploring this new column.

```
# viewing summery statistics
df['due-days'].describe()
```

	due-days
count	110527.000000
mean	10.183702
std	15.254996
min	-6.000000
25%	0.000000
50%	4.000000
75%	15.000000
max	179.000000

dtype: float64

We seem to have some negative values here, we'll drop them.

```
# viewing negative days values
df[df['due-days'] < 0 ]
```

	Gender	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcohol
27033	M	38	RESISTÊNCIA	0	0	0	
55226	F	19	SANTO ANTÔNIO	0	0	0	
64175	F	22	CONSOLAÇÃO	0	0	0	
71533	F	81	SANTO	0	0	0	

```
# dropping these values and confirming changes
df.drop(df[df['due-days'] < 0].index, inplace = True)
df['due-days'].describe()
```

	due-days
count	110522.000000
mean	10.184253
std	15.255115
min	0.000000
25%	0.000000
50%	4.000000
75%	15.000000
max	179.000000

dtype: float64

Converting Gender and No-show to categorical variables

```
# converting column and confirming changes
df['Gender'] = df['Gender'].astype('category')

df['Gender'].dtypes
```

```
CategoricalDtype(categories=['F', 'M'], ordered=False,
categories_dtype=object)
```

Converting Scholarship, Hipertension, Diabetes, Alcoholism, Handcap and SMS_recieved to boolean data type

```
# converting columns to bool and confirming changes
cols = ['Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', 'SMS_r
df[cols] = df[cols].astype('bool')
df[cols].dtypes
```

	0
Scholarship	bool
Hipertension	bool
Diabetes	bool
Alcoholism	bool
SMS_received	bool

dtype: object

Parsing and casting No-show column

```
# mapping alues to be more familiar
df.loc[df['No-show'] == 'Yes', 'No-show'] = 0
df.loc[df['No-show'] == 'No', 'No-show'] = 1

# casting dt type and confirming changes
df['No-show'] = df['No-show'].astype(bool)
df['No-show'].dtypes
```

```
dtype('bool')
```

Cleaning Handcap column

```
# viewing rows with values of handcap > 1
df[df['Handcap'] > 1]
```

	Gender	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoho
946	M	94	BELA VISTA	False	True	True	I
1665	M	64	SANTA MARTHA	False	True	False	
1666	M	64	SANTA MARTHA	False	True	False	
2071	M	64	SANTA MARTHA	False	True	False	
2091	F	11	ANDORINHAS	False	False	False	I
...	...	...	...	...	...	...	
108376	F	44	ROMÃO	False	True	True	I
109484	M	64	DA PENHA	False	True	True	I
109733	F	34	JUCUTUQUARA	False	False	False	I
109975	M	39	PRAIA DO SUÁ	True	False	False	I
110107	F	44	RESISTÊNCIA	False	False	False	I

We have 199 rows with inconsistant values, we'd replace them with 1 to treat them as beeing handcaped

```
# filling the bigger values with 1
df.loc[df['Handcap'].isin([2, 3, 4]), 'Handcap'] = 1

# casting type and confirming changes
df['Handcap'] = df['Handcap'].astype('bool')
df['Handcap'].unique()
```

```
array([False,  True])
```

Cleaning Age column

```
#exploring values below 0
df[df['Age'] < 0]
```

	Gender	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcohol…

we have one value with negative age, so we will drop it

```
# dropping row with negative age and confirming changes
df.drop(df[df['Age'] < 0].index, inplace = True)
df[df['Age'] < 0]
```

Gender	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism

```
df.head()
```

	Gender	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoholism
0	F	62	JARDIM DA PENHA	False	True	False	False
1	M	56	JARDIM DA PENHA	False	False	False	False
2	F	62	MATA DA PRAIA	False	False	False	False

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 110521 entries, 0 to 110526
Data columns (total 11 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Gender         110521 non-null  category
 1   Age            110521 non-null  int64
 2   Neighbourhood  110521 non-null  object
 3   Scholarship    110521 non-null  bool
 4   Hipertension   110521 non-null  bool
 5   Diabetes       110521 non-null  bool
 6   Alcoholism     110521 non-null  bool
 7   Handcap        110521 non-null  bool
```

```
 8   SMS_received   110521 non-null  bool
 9   No-show        110521 non-null  bool
 10  due-days       110521 non-null  int64
dtypes: bool(7), category(1), int64(2), object(1)
memory usage: 4.2+ MB
```

We endded up with a datafram of 110521 rows and 11 columns, and everything looks tidy and clean. We'd proceed in visualizing it to extract meaningful insights from it.

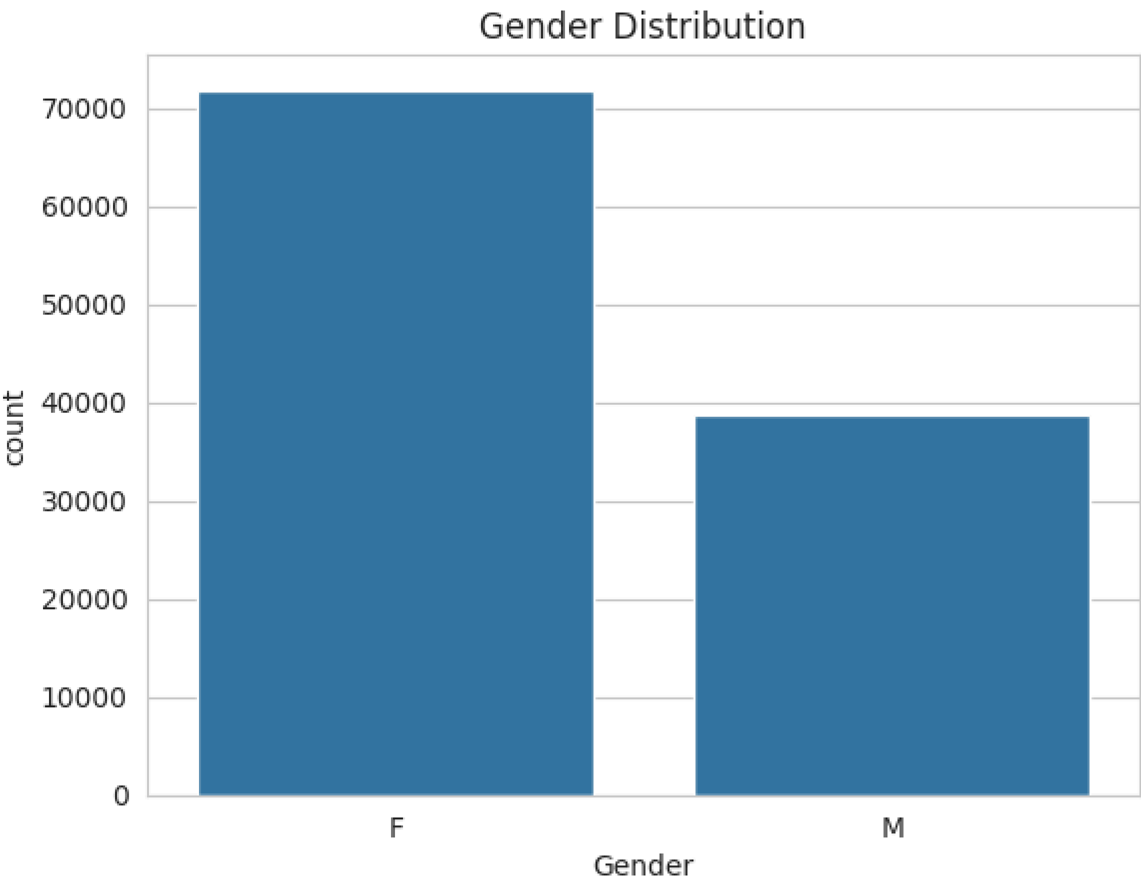
Data Visualization and EDA

Now that our data is clean, we'd perform some EDA on it in order to extract useful insights from it.

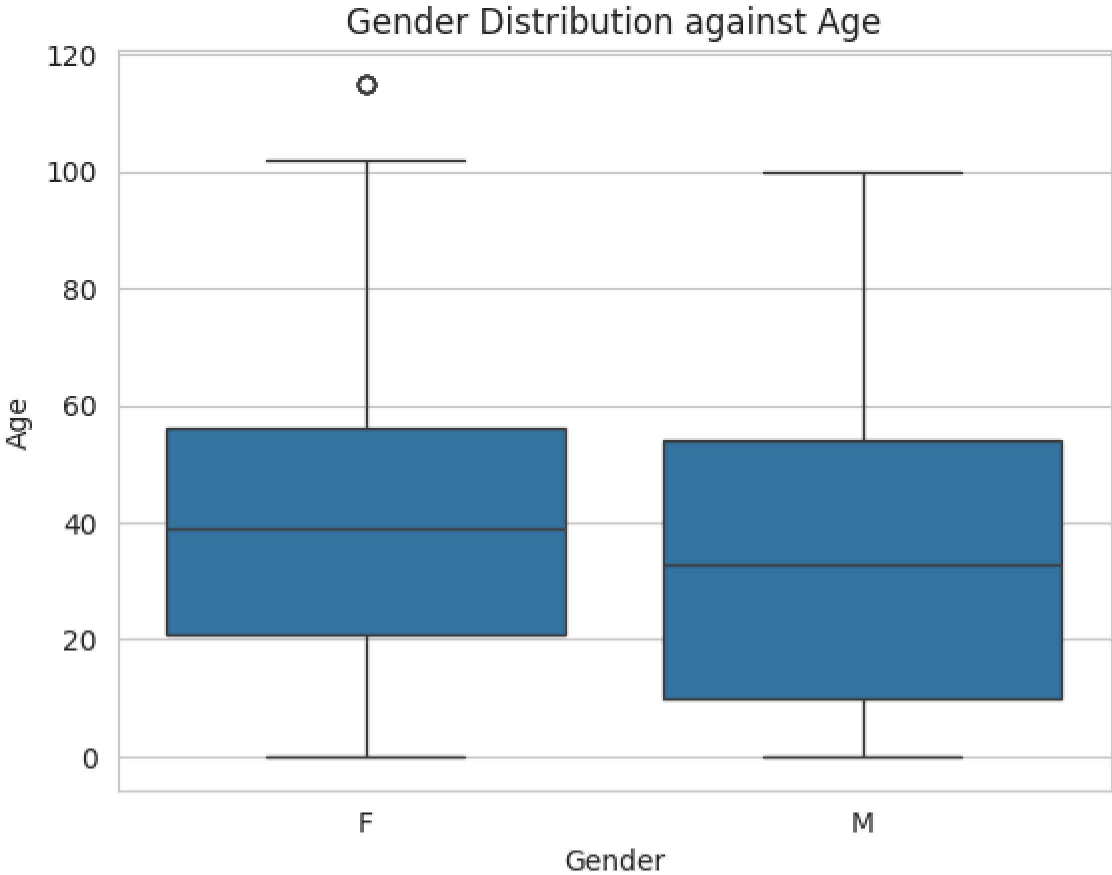
```
# setting seaborn configurations
sns.set_style("whitegrid")
```

How often do men go to hospitals compared to women? Which of them is more likely to show up?

```
# viewing count plot of gender distribution in our dataset
sns.countplot(x = 'Gender', data = df)
plt.title("Gender Distribution")
plt.show()
```




```
# viewing count plot of gender distribution against age in our dataset
sns.boxplot(x = 'Gender', y = 'Age', data = df)
plt.title("Gender Distribution against Age")
plt.show()
```



we can notice that nearly half of our dataset conists of women with wider age destribution and some outliers, all of which achiees a rate higher than men.

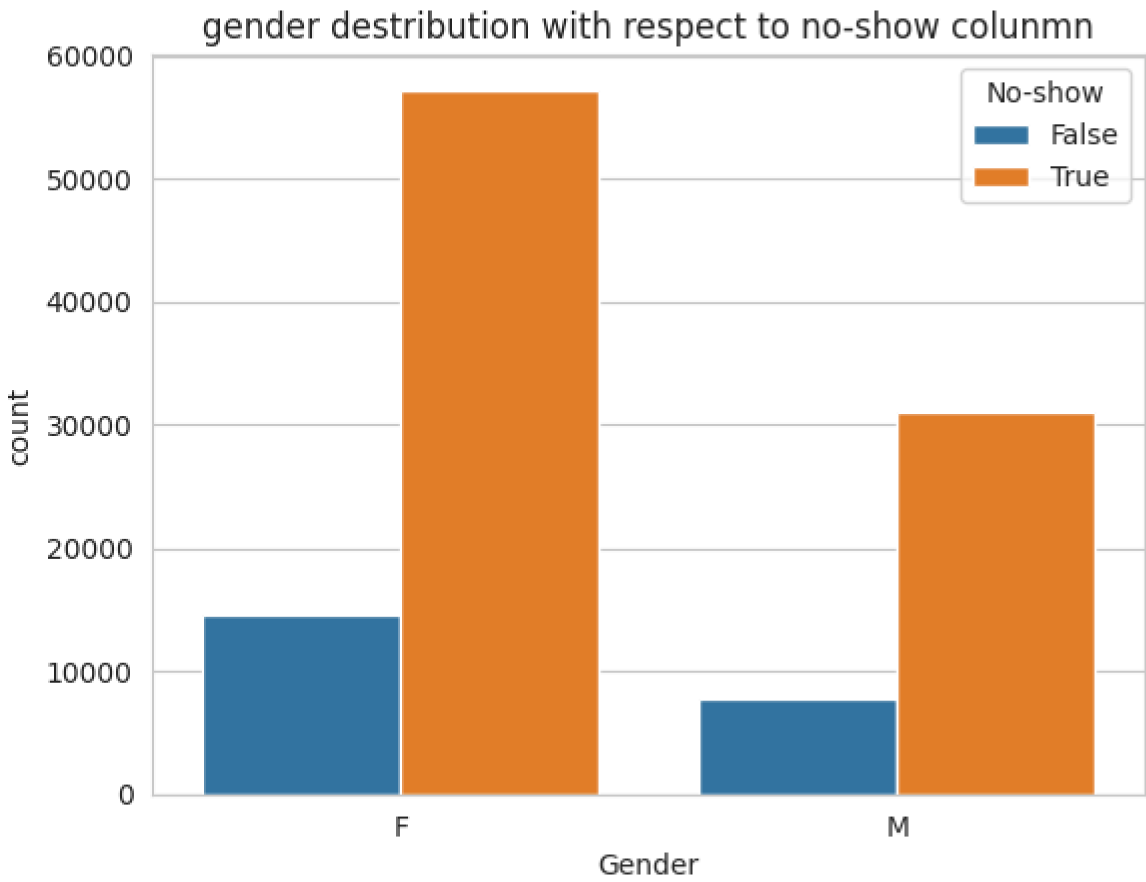
```
df['No-show'].value_counts()
```

No-show	count
True	88207
False	22314

dtype: int64

it is obvious that 79.8% of our patients did show up on their appointments and only 20.1% of them did not. lets dive deeper to see if this is related to gender.

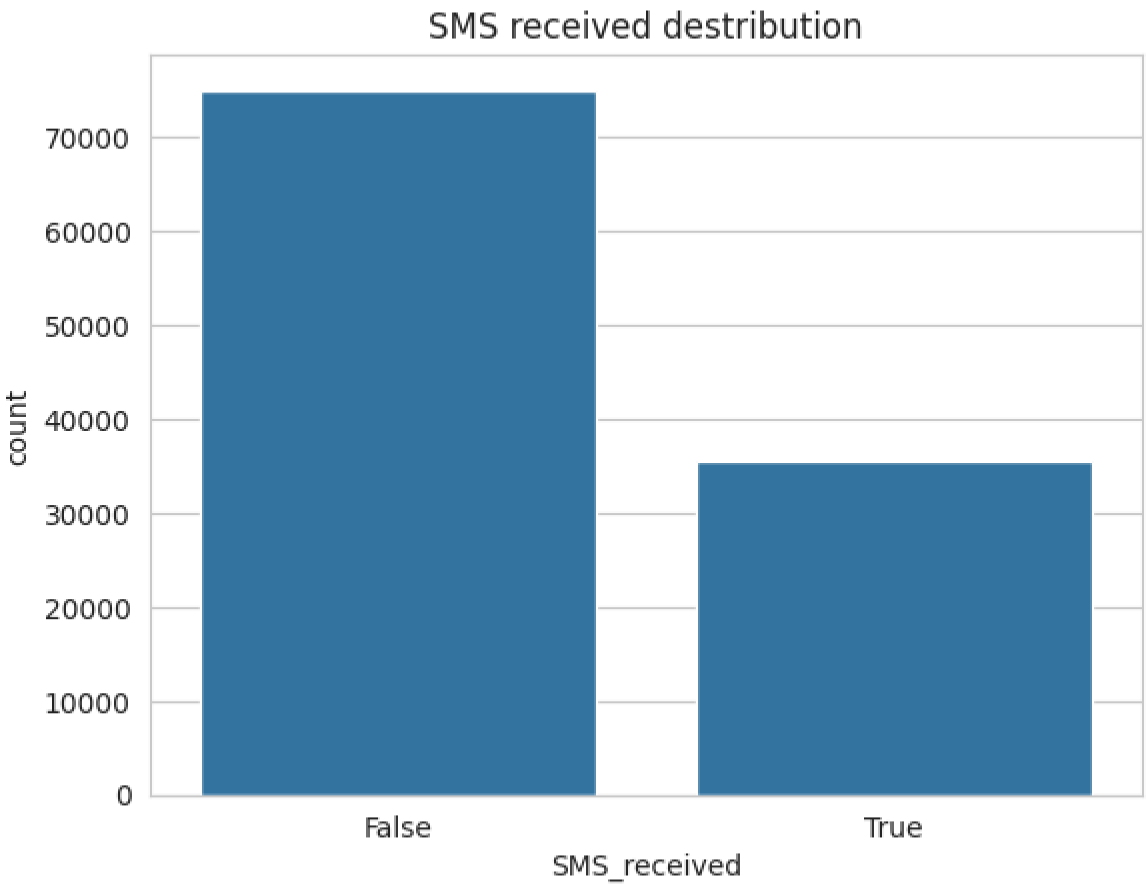
```
# showing the gender destribution with respect to the no-show colunmn
sns.countplot(x = 'Gender', data = df, hue = 'No-show')
plt.title('gender destribution with respect to no-show colunmn')
plt.show()
```



from the above chart, we can come up with a conclusion that women do show up on their appointments more often than men do, but this may be affected by the percentage of women on this dataset.

Does recieving an SMS as a reminder affect whether or not a patient may show up? is it correlated with number of days before the appointment?

```
# viewing count plot of recieving SMS distribution in our dataset
sns.countplot(x = 'SMS_received', data = df)
plt.title("SMS received destribution")
plt.show()
```



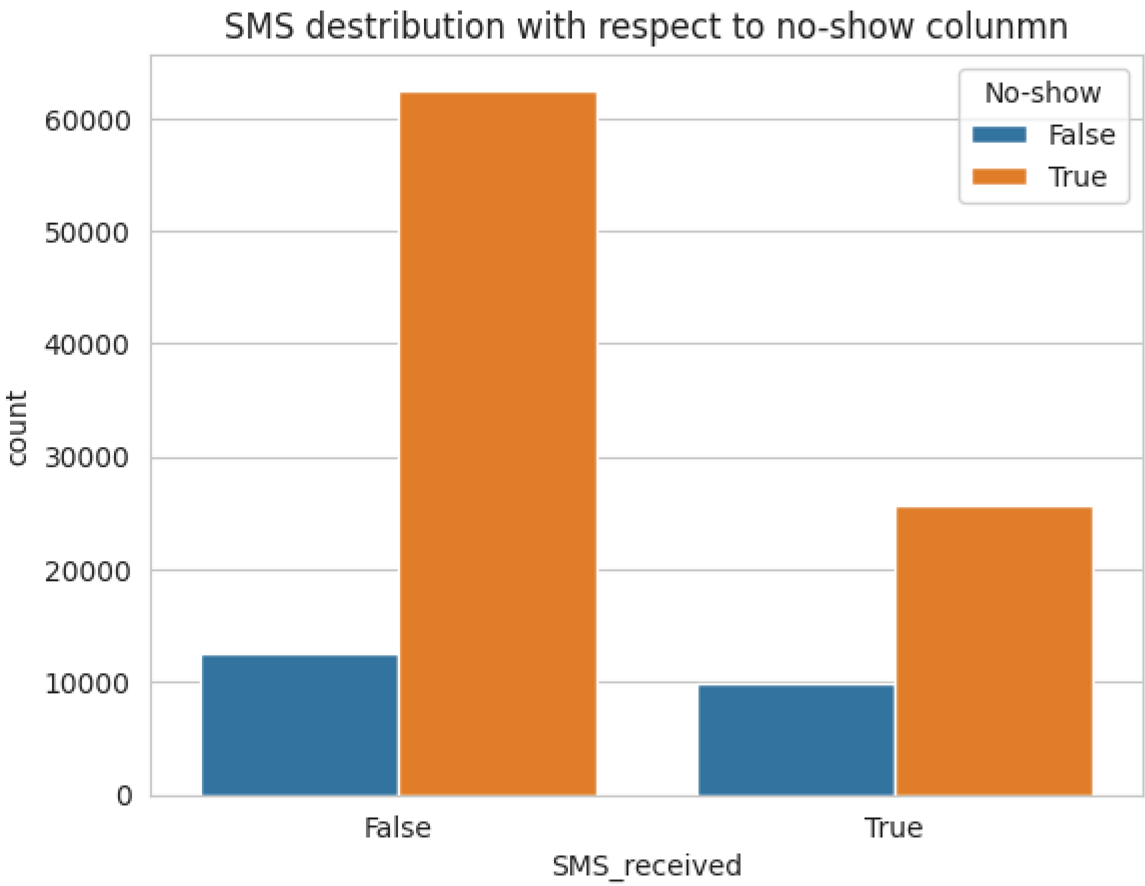

```
df['SMS_received'].value_counts()
```

SMS_received	count
False	75039
True	35482

dtype: int64

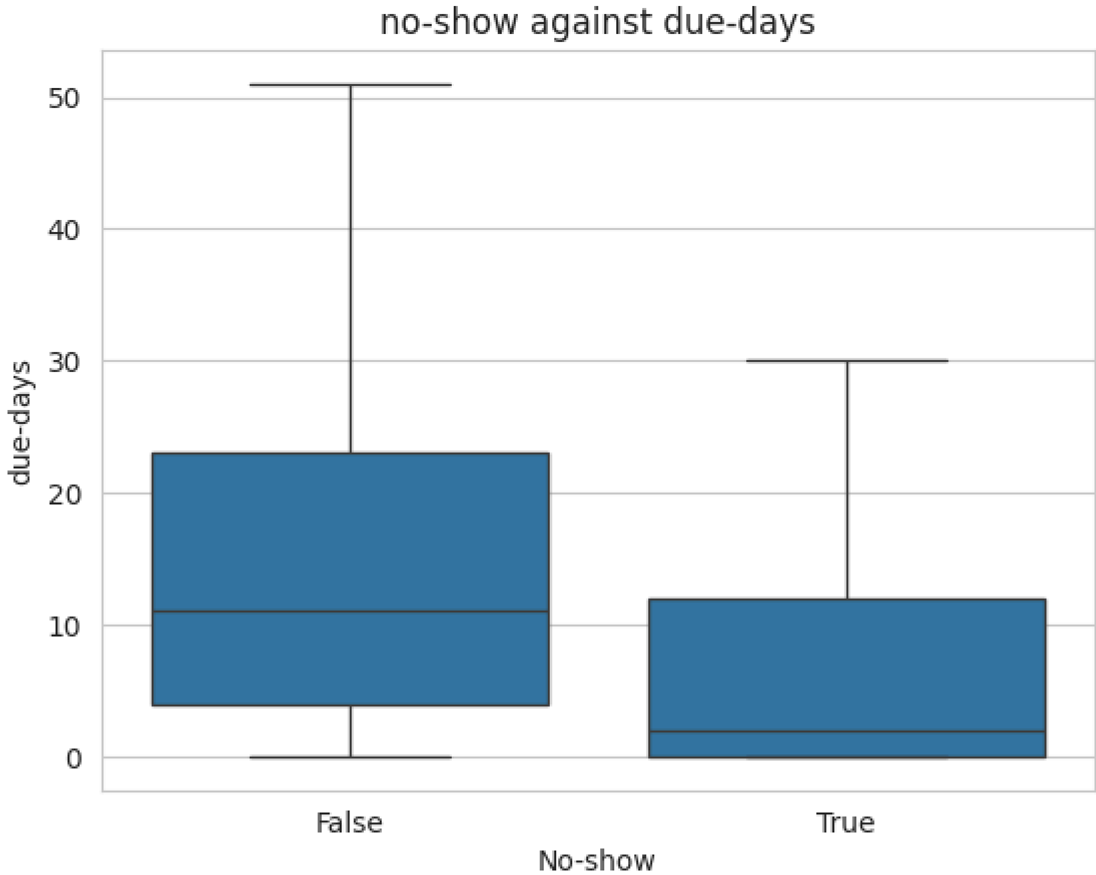
we can see that 67.8% of our patients did not reciee any SMS reminder of their appointments, cound this be affecting their showin up?

```
# showing the sms destribution with respect to the no-show colunmn
sns.countplot(x = 'SMS_received', data = df, hue = 'No-show')
plt.title('SMS destribution with respect to no-show colunmn')
plt.show()
```



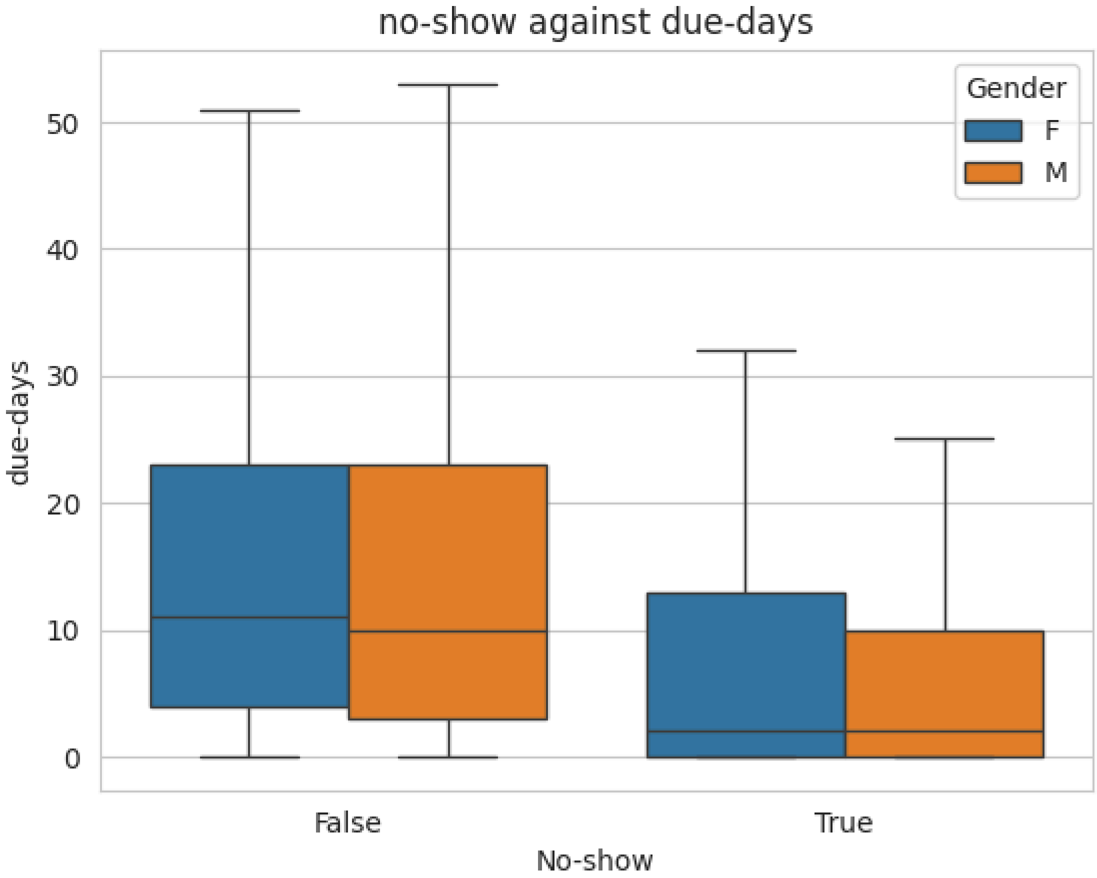
we can see that our previous deduction was not quiet correct, as the vast majority of our patients did not recieve any SMS reminder and yet they showed up on their appointments.

```
# viewing the correlation between no-show and due-days without outliers
sns.boxplot(x = 'No-show', y = 'due-days', data = df, showfliers = Fals
plt.title('no-show against due-days')
plt.show()
```



it is clear that there is a correlation between number od due days and whether a patient shows up or not. patient with appointments from 0 to 30 days tend to show up more regularly, while patients with higher number of days tend to not show up.

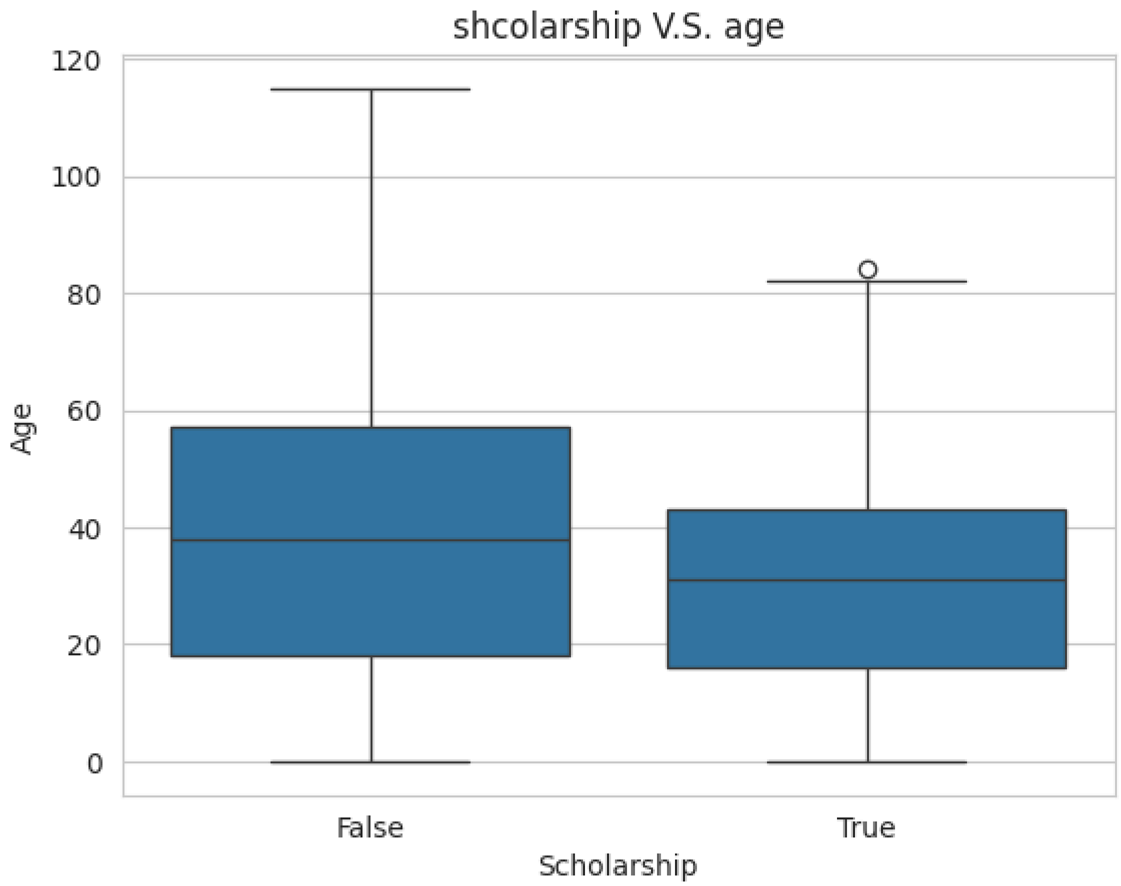
```
# viewing the correlation between no-show and due-days without outliers
sns.boxplot(x = 'No-show', y = 'due-days', data = df, hue = 'Gender', s
plt.title('no-show against due-days')
plt.show()
```



Gender does not affect number of due days and showing up at an appointment that much.

Does having a scholarship affects showing up on a hospital appointment? What are the age groups affected by this?

```
# plotting having a scholarship against age
sns.boxplot(x = 'Scholarship', y = 'Age', data = df)
plt.title('shcolarship V.S. age')
plt.show()
```

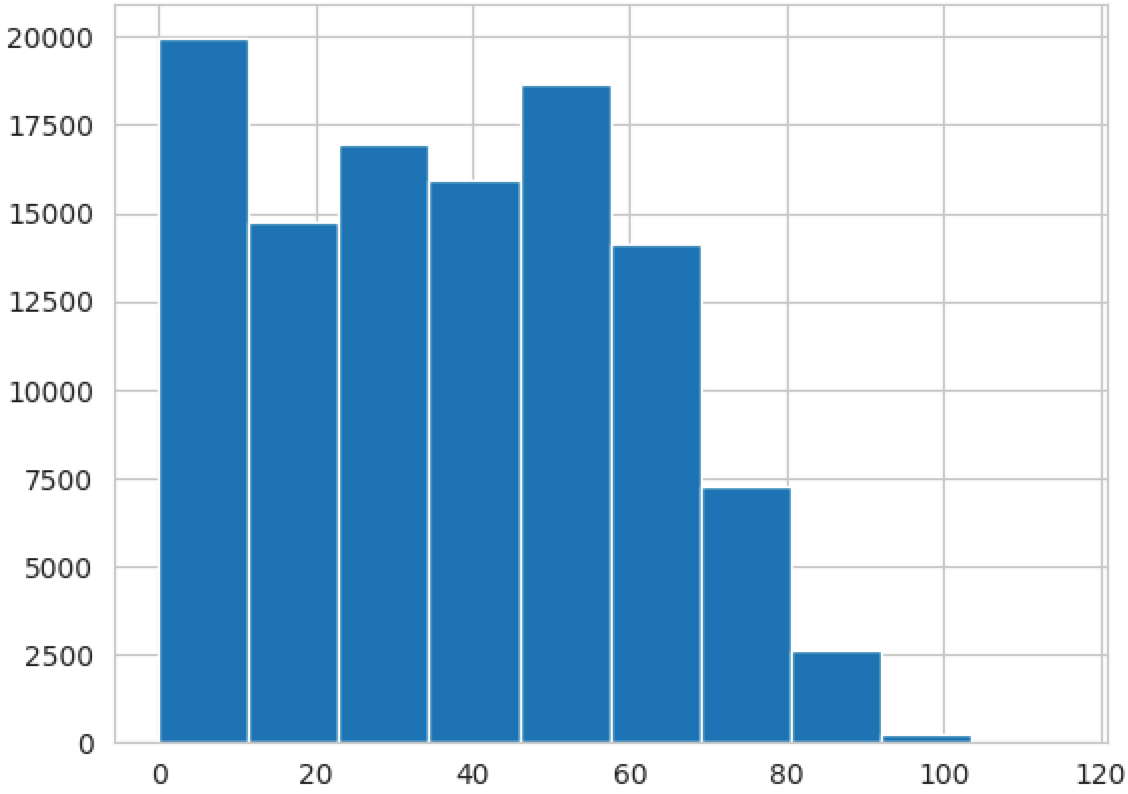



```
# plotting having a scholarship against no show with respect to gender
sns.barplot(x = 'Scholarship', y = 'No-show', hue = 'Gender', data = df
plt.title('shcolarship V.S. no show with respect to gender')
plt.show()
```

shcolarship V.S. no show with respect to gender

```
# ploting age destribution
df['Age'].hist()
```

<Axes: >



we can see that having a scolarship does not affect showing up to a doctor appointment that much and that huge age group is enrolled to that scholarship and also enrol their babies on.

Does having certain deseas affects whather or not a patient may show up to their appointment? is it affected by gender?

```
# plotting deseases against no show
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.countplot(x = 'Hipertension', data = df, hue= 'No-show')
plt.subplot(2,2,2)
sns.countplot(x = 'Diabetes', data = df, hue= 'No-show')
```