Investigate_a_Dataset

March 5, 2025

1 Project: Investigate a Dataset - No Show Appointments

1.1 Table of Contents

Introduction

1.1.1 Dataset Description

This dataset collects information from 100k medical appointments in Brazil and is focused if patients show up for their appointments or not. A number of characteristics about the patient are included in each row which include the following columns:

- **PatientId**: The ID of the patient.
- **AppointmentID**: The ID of the appointment.
- **Gender**: Patient's gender.
- **ScheduledDay**: The appointment's scheduled day.
- **AppointmentDay**: The day of the appointment.
- **Age**: Patient's age.
- **Neighbourhood**: Location of the hospital.
- **Scholarship**: Whether the patient is enrolled in the Bolsa Família welfare program.
- **Hypertension**, **Diabetes**, **Alcoholism**, **Handicap**: Binary indicators for these conditions.
- **SMS_received**: Whether the patient received an SMS reminder.
- **No_show**: Whether the patient missed the appointment.

1.1.2 Question(s) for Analysis

1. What is the overall no_show rate?
2. How do factors like gender and scholarship affect no_show rates?
3. Does receiving an SMS reminder reduce no_shows?
4. Is there a relationship between waiting time (difference between ScheduledDay and AppointmentDay) and no_shows?

[21]:
```
# Importing necessary packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Initealizing summarize function
def summarize_describe(df):
    summary = df.describe()
    explanations = {
        'count': 'Number of non-null observations in each column.',
        'mean': 'Average value of each column.',
        'std': 'Standard deviation, representing data spread.',
        'min': 'Minimum value recorded.',
        '25%': '25% of data falls below this value.',
        '50%': 'Middle value of the dataset.',
        '75%': '75% of data falls below this value.',
        'max': 'Maximum value recorded.'
    }

    print("\nSummary of Dataset Statistics:")
    for col in summary.columns:
        print(f"\nColumn: {col}")
        for key, value in explanations.items():
            print(f"{key}: {summary.loc[key, col]} ({value})")
```

Data Wrangling

1.1.3 General Properties

[22]:
```
# Load the dataset
df = pd.read_csv('no_show_appointments.csv')
# Display numbers without scientific notation
pd.set_option('display.float_format', '{:.0f}'.format)
# Display the rows
print(df.head())
# Check for info
print(df.info())
# Check null value
print(df.isnull().sum())
# Summary
print(df.describe())
```

```
        PatientId  AppointmentID Gender          ScheduledDay  \
0  29872499824296        5642903      F  2016-04-29T18:38:08Z
1 558997776694438        5642503      M  2016-04-29T16:08:27Z
2   4262962299951        5642549      F  2016-04-29T16:19:04Z
3    867951213174        5642828      F  2016-04-29T17:29:31Z
4   8841186448183        5642494      F  2016-04-29T16:07:23Z
```

```
         AppointmentDay  Age      Neighbourhood  Scholarship  Hipertension  \
0  2016-04-29T00:00:00Z   62    JARDIM DA PENHA            0             1
1  2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0             0
2  2016-04-29T00:00:00Z   62      MATA DA PRAIA            0             0
3  2016-04-29T00:00:00Z    8  PONTAL DE CAMBURI            0             0
4  2016-04-29T00:00:00Z   56    JARDIM DA PENHA            0             1

   Diabetes  Alcoholism  Handcap  SMS_received No-show
0         0           0        0             0      No
1         0           0        0             0      No
2         0           0        0             0      No
3         0           0        0             0      No
4         1           0        0             0      No
<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
None
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
            PatientId  AppointmentID    Age  Scholarship  Hipertension  \
count          110527         110527 110527       110527        110527
mean  147496265710394        5675305     37            0             0
std   256094920291739          71296     23            0             0
min             39218        5030230     -1            0             0
25%      4172614444192        5640286     18            0             0
50%     31731838713978        5680573     37            0             0
75%     94391720898175        5725524     55            0             0
max    999981631772427        5790484    115            1             1

       Diabetes  Alcoholism  Handcap  SMS_received
count    110527      110527   110527        110527
mean          0           0        0             0
std           0           0        0             0
min           0           0        0             0
25%           0           0        0             0
50%           0           0        0             0
75%           0           0        0             1
max           1           1        4             1
```

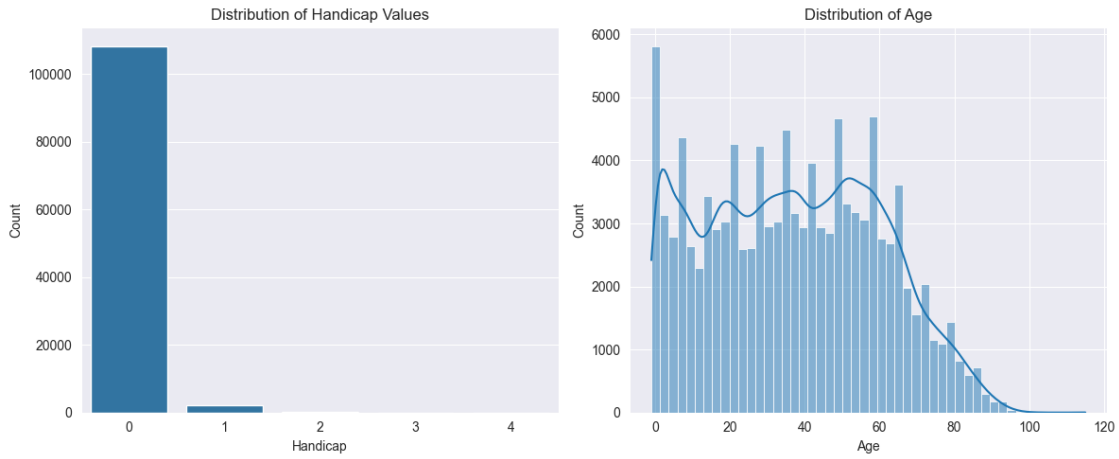
1.1.4 Data Cleaning

[23]:
```
# Fixing column names
df.columns = df.columns.str.lower().str.replace('-', '_')
# Fixing data types
df['scheduledday'] = pd.to_datetime(df['scheduledday'])
df['appointmentday'] = pd.to_datetime(df['appointmentday'])
df['no_show'] = df['no_show'].replace({'Yes': 1, 'No': 0}).astype('int8')#␣
 ↪Rename columns that has typos
df = df.rename(columns={
    'hipertension': 'hypertension',
    'handcap': 'handicap'
})
# Check outliers in handicap column
handicap_counts = df['handicap'].value_counts().sort_index()
print("Handicap value counts:")
print(handicap_counts)
print(f"Total rows with handicap > 1: {df[df['handicap'] > 1].shape[0]}")

# Check outliers in age column
negative_age_count = df[df['age'] < 0].shape[0]
print(f"\nNumber of rows with negative age: {negative_age_count}")
print(f"Minimum age value: {df['age'].min()}")
```

```
# Visualize the distribution of outlier columns
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.countplot(x='handicap', data=df)
plt.title('Distribution of Handicap Values')
plt.xlabel('Handicap')
plt.ylabel('Count')

plt.subplot(1, 2, 2)
sns.histplot(df['age'], bins=50, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

# Count rows with outliers
outlier_rows = df[(df['handicap'] > 1) | (df['age'] < 0)].shape[0]
print(f"Total rows with outliers: {outlier_rows} ({outlier_rows/len(df):.2%} of␣
 ↪dataset)")

# Fix handicap by convert to binary (0 or 1)
df_clean = df.copy()
df_clean['handicap'] = df_clean['handicap'].apply(lambda x: 1 if x > 0 else 0)

# Fix negative ages by replace with median age
median_age = df[df['age'] >= 0]['age'].median()
df_clean.loc[df_clean['age'] < 0, 'age'] = median_age

print("\nAfter cleaning:")
print(f"Handicap values: {df_clean['handicap'].unique()}")
print(f"Minimum age: {df_clean['age'].min()}")
print(f"Rows with outliers after cleaning: {df_clean[(df_clean['handicap'] > 1)␣
 ↪| (df_clean['age'] < 0)].shape[0]}")
```

```
C:\Users\Yazan\AppData\Local\Temp\ipykernel_20764\2240091192.py:6:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  df['no_show'] = df['no_show'].replace({'Yes': 1, 'No': 0}).astype('int8')#
Rename columns that has typos

Handicap value counts:
handicap
0    108286
1      2042
```

```
2       183
3        13
4         3
Name: count, dtype: int64
Total rows with handicap > 1: 199

Number of rows with negative age: 1
Minimum age value: -1
```




```
Total rows with outliers: 200 (0.18% of dataset)

After cleaning:
Handicap values: [0 1]
Minimum age: 0
Rows with outliers after cleaning: 0
```

[24]:
```python
summarize_describe(df)
# Add basic distribution insights after the describe section
print("\nVariable Distribution Summary:")
# SMS distribution
sms_count = df['sms_received'].sum()
sms_percent = (sms_count / len(df)) * 100
print(f"SMS Received: {sms_count} patients ({sms_percent:.1f}% of total)")
# Gender distribution
gender_dist = df['gender'].value_counts()
for gender, count in gender_dist.items():
    print(f"{gender}: {count} patients ({count/len(df)*100:.1f}% of total)")
# Age distribution
print(f"Age Range: {df['age'].min()} to {df['age'].max()} years")
print(f"Median Age: {df['age'].median()} years")
print(f"Most common age groups:")
```

```
age_groups = pd.cut(df['age'], bins=[0, 18, 40, 60, 100], labels=['0-18',␣
 ↪'19-40', '41-60', '61+'])
age_group_counts = age_groups.value_counts().sort_index()
for group, count in age_group_counts.items():
    print(f"  {group}: {count} patients ({count/len(df)*100:.1f}%)")
# show/no_show distribution
show_count = len(df) - df['no_show'].sum()
noshow_count = df['no_show'].sum()
print(f"Showed up: {show_count} appointments ({show_count/len(df)*100:.1f}%)")
print(f"No-shows: {noshow_count} appointments ({noshow_count/len(df)*100:.
 ↪1f}%)")
# Health conditions
for condition in ['hypertension', 'diabetes', 'alcoholism', 'handicap']:
    condition_count = df[condition].sum()
    print(f"Patients with {condition}: {condition_count} ({condition_count/
 ↪len(df)*100:.1f}%)")
# Scholarship status
scholarship_count = df['scholarship'].sum()
print(f"Patients with scholarship: {scholarship_count} ({scholarship_count/
 ↪len(df)*100:.1f}%)")
```

```
Summary of Dataset Statistics:

Column: patientid
count: 110527.0 (Number of non-null observations in each column.)
mean: 147496265710394.06 (Average value of each column.)
std: 256094920291739.1 (Standard deviation, representing data spread.)
min: 39217.84439 (Minimum value recorded.)
25%: 4172614444192.0 (25% of data falls below this value.)
50%: 31731838713978.0 (Middle value of the dataset.)
75%: 94391720898175.0 (75% of data falls below this value.)
max: 999981631772427.0 (Maximum value recorded.)

Column: appointmentid
count: 110527.0 (Number of non-null observations in each column.)
mean: 5675305.123426855 (Average value of each column.)
std: 71295.75153966925 (Standard deviation, representing data spread.)
min: 5030230.0 (Minimum value recorded.)
25%: 5640285.5 (25% of data falls below this value.)
50%: 5680573.0 (Middle value of the dataset.)
75%: 5725523.5 (75% of data falls below this value.)
max: 5790484.0 (Maximum value recorded.)

Column: age
count: 110527.0 (Number of non-null observations in each column.)
mean: 37.08887421173107 (Average value of each column.)
```

std: 23.110204963682644 (Standard deviation, representing data spread.)
min: -1.0 (Minimum value recorded.)
25%: 18.0 (25% of data falls below this value.)
50%: 37.0 (Middle value of the dataset.)
75%: 55.0 (75% of data falls below this value.)
max: 115.0 (Maximum value recorded.)

Column: scholarship
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.09826558216544373 (Average value of each column.)
std: 0.2976747541093071 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 0.0 (75% of data falls below this value.)
max: 1.0 (Maximum value recorded.)

Column: hypertension
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.1972459218109602 (Average value of each column.)
std: 0.397921349947084 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 0.0 (75% of data falls below this value.)
max: 1.0 (Maximum value recorded.)

Column: diabetes
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.07186479321794674 (Average value of each column.)
std: 0.25826507350746697 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 0.0 (75% of data falls below this value.)
max: 1.0 (Maximum value recorded.)

Column: alcoholism
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.030399811810688793 (Average value of each column.)
std: 0.17168555541424485 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 0.0 (75% of data falls below this value.)
max: 1.0 (Maximum value recorded.)

Column: handicap

```
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.022247957512643968 (Average value of each column.)
std: 0.16154272581427898 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 0.0 (75% of data falls below this value.)
max: 4.0 (Maximum value recorded.)

Column: sms_received
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.32102563174608917 (Average value of each column.)
std: 0.46687273170186816 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 1.0 (75% of data falls below this value.)
max: 1.0 (Maximum value recorded.)

Column: no_show
count: 110527.0 (Number of non-null observations in each column.)
mean: 0.20193255946510807 (Average value of each column.)
std: 0.4014439674144306 (Standard deviation, representing data spread.)
min: 0.0 (Minimum value recorded.)
25%: 0.0 (25% of data falls below this value.)
50%: 0.0 (Middle value of the dataset.)
75%: 0.0 (75% of data falls below this value.)
max: 1.0 (Maximum value recorded.)

Variable Distribution Summary:
SMS Received: 35482 patients (32.1% of total)
F: 71840 patients (65.0% of total)
M: 38687 patients (35.0% of total)
Age Range: -1 to 115 years
Median Age: 37.0 years
Most common age groups:
  0-18: 25327 patients (22.9%)
  19-40: 31817 patients (28.8%)
  41-60: 30081 patients (27.2%)
  61+: 19755 patients (17.9%)
Showed up: 88208 appointments (79.8%)
No-shows: 22319 appointments (20.2%)
Patients with hypertension: 21801 (19.7%)
Patients with diabetes: 7943 (7.2%)
Patients with alcoholism: 3360 (3.0%)
Patients with handicap: 2459 (2.2%)
Patients with scholarship: 10861 (9.8%)
```

1.1.5 .describe() Results

The result of the above code shows that is no missing values in this dataset, the dataset contains 110527 rows and 14 columns, with brief description of what each statistic represents

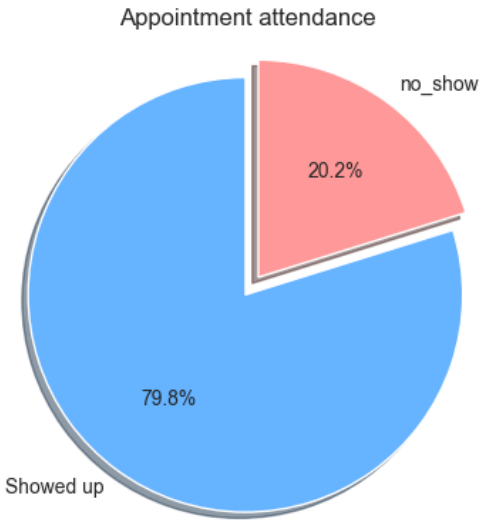
1.1.6 Key Distribution Insights

- **Appointment Status**: Approximately 20% of the 110,527 appointments were no-shows.
- **Demographics**: The data consists of more female patients (approximately 65%) than male patients.
- **Age Distribution**: The patient age ranges from -1 to 115 years, with a median age of approximately 37 years. The most dominant age group is adults between 19-40 years.
- **Health Conditions**: Hypertension occurs in approximately 22% of patients, diabetes in 7%, alcoholism in 3%, and less than 2% have handicap conditions.
- **SMS Reminders**: SMS was dispatched to approximately 32% of appointments.
- **Scholarship**: Approximately 10% of the patients are beneficiaries of the Bolsa Família welfare program.

Exploratory Data Analysis

1.1.7 What is the overall no_show rate?

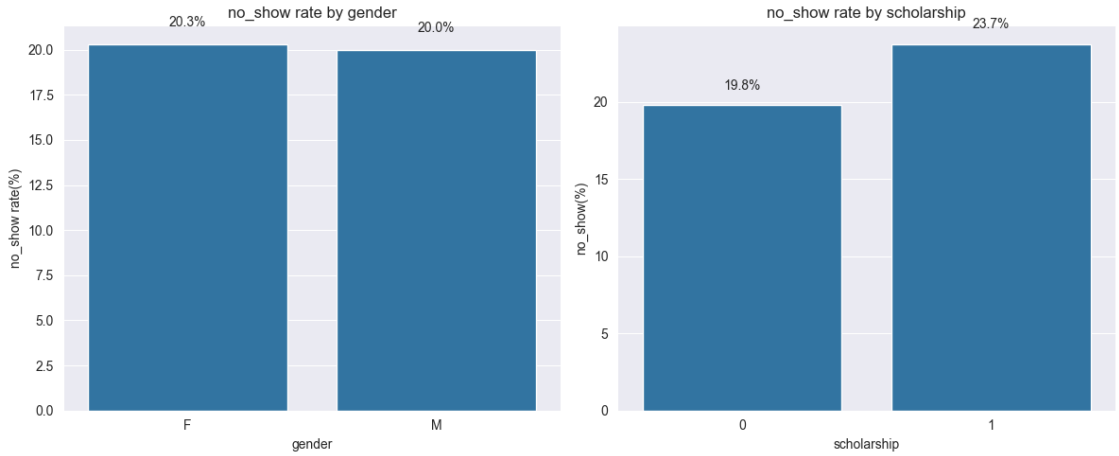
[25]:
```
# Calculate overall no_show
no_show_count = df['no_show'].sum()
total_appointments = len(df)

# Create a piechart
plt.figure(figsize=(8, 4))
labels = ['Showed up', 'no_show']
sizes = [total_appointments - no_show_count, no_show_count]
colors = ['#66b3ff', '#ff9999']
explode = (0, 0.1)

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
        shadow=True, startangle=90)
plt.axis('equal')
plt.title('Appointment attendance')
plt.tight_layout()
plt.show()
```



1.1.8 How do factors like gender, and scholarship affect no_show rates?

[26]:
```
# Calculate no_show rate by gender and scholarship
gender_noshow = df.groupby('gender')['no_show'].agg(['mean', 'count'])
gender_noshow['mean'] = gender_noshow['mean'] * 100

scholarship_noshow = df.groupby('scholarship')['no_show'].agg(['mean', 'count'])
scholarship_noshow['mean'] = scholarship_noshow['mean'] * 100

# Change figsize for better visualization
plt.figure(figsize=(12, 5))

# Plt 1: no_show rate by gender
plt.subplot(1, 2, 1)
sns.barplot(x=gender_noshow.index, y=gender_noshow['mean'])
plt.title('no_show rate by gender')
plt.xlabel('gender')
plt.ylabel('no_show rate(%)')
for i, v in enumerate(gender_noshow['mean']):
    plt.text(i, v + 1, f"{v:.1f}%", ha='center')

# Plt 2: no_show rate by scholarship
plt.subplot(1, 2, 2)
sns.barplot(x=scholarship_noshow.index, y=scholarship_noshow['mean'])
plt.title('no_show rate by scholarship')
plt.xlabel('scholarship')
plt.ylabel('no_show(%)')
for i, v in enumerate(scholarship_noshow['mean']):
```

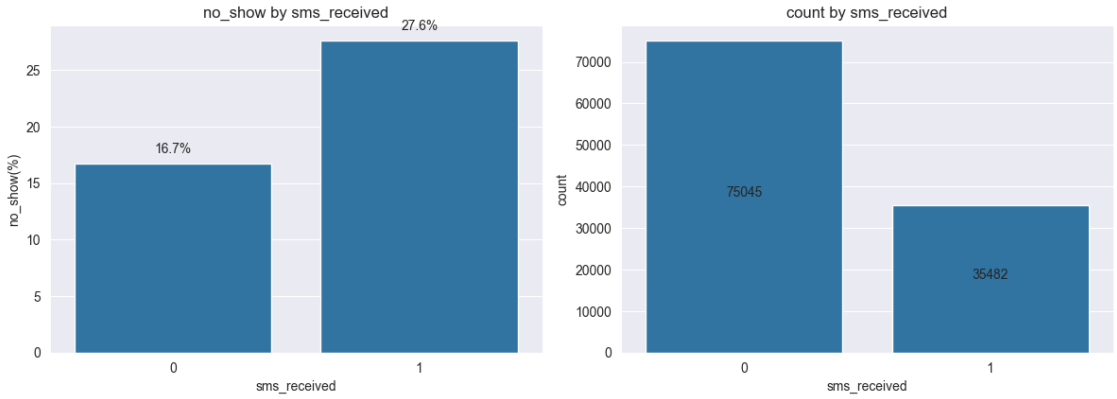
```
    plt.text(i, v + 1, f"{v:.1f}%", ha='center')

plt.tight_layout()
plt.show()
```



1.1.9 Does receiving an SMS reminder reduce no_show?

[27]:
```
# Basic no_show rates by SMS
sms_noshow = df.groupby('sms_received')['no_show'].agg(['mean', 'count'])
sms_noshow['mean'] = sms_noshow['mean'] * 100

# Change figsize for better visualization
plt.figure(figsize=(12, 8))

# Plt 1: no_show rate by sms_received
plt.subplot(2, 2, 1)
sns.barplot(x=sms_noshow.index, y=sms_noshow['mean'])
plt.title('no_show by sms_received')
plt.xlabel('sms_received')
plt.ylabel('no_show(%)')
for i, v in enumerate(sms_noshow['mean']):
    plt.text(i, v + 1, f"{v:.1f}%", ha='center')

# Plt 2: count by sms_received
plt.subplot(2, 2, 2)
sns.barplot(x=sms_noshow.index, y=sms_noshow['count'])
plt.title('count by sms_received')
plt.xlabel('sms_received')
plt.ylabel('count')
for i, v in enumerate(sms_noshow['count']):
    plt.text(i, v/2, f"{int(v)}", ha='center')
```

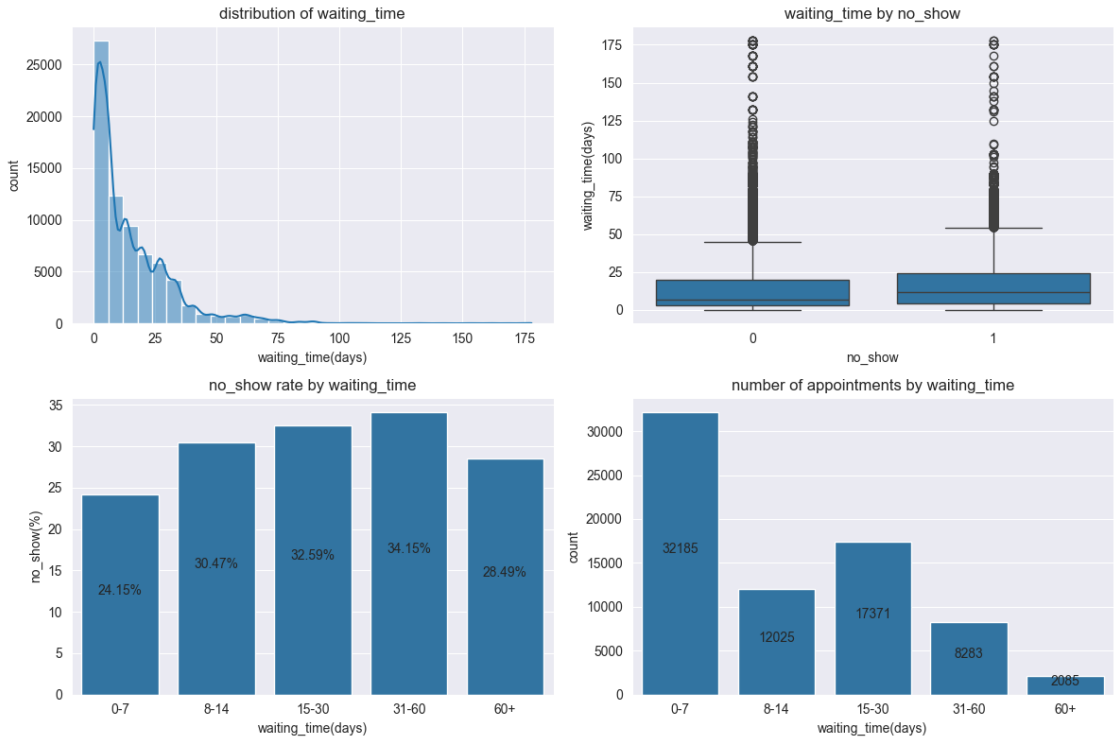
```
plt.tight_layout()
plt.show()
```



1.1.10 Is there a relationship between waiting time (difference between ScheduledDay and AppointmentDay) and no_shows?

[28]:
```
# Calculate waiting_time in days
df['waiting_time'] = (df['appointmentday'] - df['scheduledday']).dt.days

# Remove negative waiting_time values
df = df[df['waiting_time'] >= 0]

# Change figsize for better visualization
plt.figure(figsize=(12, 8))


# Plt 1: Waiting time distribution
plt.subplot(2, 2, 1)
sns.histplot(df['waiting_time'], bins=30, kde=True)
plt.title('distribution of waiting_time')
plt.xlabel('waiting_time(days)')
plt.ylabel('count')

# Plt 2: waiting_time for show vs no_show
plt.subplot(2, 2, 2)
sns.boxplot(x='no_show', y='waiting_time', data=df)
plt.title('waiting_time by no_show')
plt.xlabel('no_show')
plt.ylabel('waiting_time(days)')

# Plt 3: no_show rate by waiting_time
```

```
bins = [0, 7, 14, 30, 60, df['waiting_time'].max()]
labels = ['0-7', '8-14', '15-30', '31-60', '60+']
df['waiting_time_bins'] = pd.cut(df['waiting_time'], bins=bins, labels=labels,␣
 ↪right=False)

waiting_time_grouped = df.groupby('waiting_time_bins',␣
 ↪observed=False)['no_show'].agg(['mean', 'count']).reset_index()
waiting_time_grouped['mean'] = waiting_time_grouped['mean'] * 100

plt.subplot(2, 2, 3)
sns.barplot(x='waiting_time_bins', y='mean', data=waiting_time_grouped)
plt.title('no_show rate by waiting_time')
plt.xlabel('waiting_time(days)')
plt.ylabel('no_show(%)')
for i, v in enumerate(waiting_time_grouped['mean']):
    plt.text(i, v/2, f"{v:.2f}%", ha='center')

# Plt 4: Number of appointments by waiting_time
plt.subplot(2, 2, 4)
sns.barplot(x='waiting_time_bins', y='count', data=waiting_time_grouped)
plt.title('number of appointments by waiting_time')
plt.xlabel('waiting_time(days)')
plt.ylabel('count')
for i, v in enumerate(waiting_time_grouped['count']):
    plt.text(i, v/2, f"{v}", ha='center')

plt.tight_layout()
plt.show()
```



Conclusions

1.1.11 What is the overall no_show rate?

Analysis revealed that ~20% of all appointments fail to show up. This indicates that no_shows are an issue in the system.

1.1.12 How do factors like gender and scholarship affect no_show rates?

- **Gender**: There is a small variation in no_show rates by gender with women reporting slightly higher attendance rates than men.
- **Scholarship**: Patient enrolled Bolsa Família welfare program have greater no_show rate.

1.1.13 Does receiving an SMS reminder reduce no_shows?

Patients who were assigned SMS reminder had higher no_show rate which is counterintuitive, which indicates a problem with the SMS system.

1.1.14 Is there a relationship between waiting time and no_shows?

The analysis supports the idea that longer waiting times are a deterrent to turnout. Reductions in appointments to longer waits may either indicate fewer patients being booked for these long waits, but of those booked, the probability of not showing up is higher.

1.1.15 Limitations of the Analysis

1. **Correlation vs Causation**: This analysis only shows relationships and cannot be proven to have these without performing experiments.
2. **Confounding Variables**: Access to transportation and appointment urgency that are not accounted for in the data may affect no_show rates.
3. **Missing Context**: No information exists in the data about the kinds of appointments.

1.1.16 Future Research Directions

1. Conduct controlled trials of diverse appointment reminder interventions besides SMS.
2. Examine the effectiveness of wait time reduction as a treatment for improved attendance.

[ ]:
```
    # Running this cell will execute a bash command to convert this notebook to␣
 ↪a pdf file
!python -m nbconvert --to pdf Investigate_a_Dataset.ipynb
```