

⚠️ ❤️ Risk Factors for Heart Disease on AI-Generated Data ❤️ ⚠️

Contents

1. About Notebook and Dataset
2. AI's Data Creation Process
3. Data Import and Review
 - Load and Preview Raw Data
 - Display Data Types
 - Check Data for Missing Values
 - Display Unique Values of Qualitative Variables
4. Exploratory Data Analysis
 - Descriptive Statistics for Age, Resting Blood Pressure and Serum Cholesterol in Respect to Sex
 - Distributions of Age, Resting Blood Pressure and Serum Cholesterol in Respect to Sex
 - Observation Frequencies for Levels of Qualitative Variables by Sex
 - Descriptive Statistics for Age, Serum Cholesterol and Resting Blood Pressure in Respect to Heart Disease Diagnosis
 - Scatterings of Heart Disease Diagnoses According to Age, Serum Cholesterol and Resting Blood Pressure
5. Statistical Inference
 - Examining the Potential Associations Between Sex and Other Qualitative Variables
6. Modeling
 - Predictions About the Risk of Heart Disease Based on Age + Serum Cholesterol and Age + Resting Blood Pressure
 - Predictions About the Risk of Heart Disease Based on Age + Serum Cholesterol + Resting Blood Pressure
7. Summary

About Notebook and Dataset

This notebook aims to demonstrate the basics of data preprocessing and exploratory data analysis using the Python programming language on a dataset generated by ChatGPT artificial intelligence.

The generated data contains 7 columns and 540 records. These columns are:

- 0 **id**: record numbers
- 1 **age**: age of imaginary respondents
- 2 **sex**: biological sex of imaginary respondents
- 3 **chest_pain_type**: reported type of chest pain

4 **resting_blood_pressure**: heart rate measured at rest (mmHg)

5 **serum_cholesterol**: serum cholesterol level (mm/dL)

6 **fasting_blood_sugar**: fasting blood sugar/glucose; zero indicates a level in the normal range (up to 100 mg/dL); one indicates a level above normal

7 **heart_disease_diagnosis**: diagnosis of any heart disease (Yes) or none (No)

AI's Data Creation Process

To generate the data, the ChatGPT query presented below was repeated until more than 500 records were generated. The generated data was dumped into a spreadsheet, then saved as a CSV file (*data_heartdisease.csv*), and uploaded to Kaggle as  [Risk Factors for Heart Disease Data by AI](#).



Can you generate sample data for 7 variables (Age, Sex, Chest Pain Type, Resting Blood Pressure, Serum Cholesterol, Fasting Blood Sugar, Heart Disease Diagnosis) on risk factors for heart disease diagnosis?



Sure! Here's an example dataset with 7 variables on risk factors for heart disease diagnosis:



Age	Sex	Chest Pain Type	Resting Blood Pressure	Serum Cholesterol	Fasting Blood Sugar	Heart Disease Diagnosis
45	Male	Typical Angina	130/90 mmHg	240 mg/dL	95 mg/dL	Yes
52	Female	Atypical Angina	150/95 mmHg	200 mg/dL	105 mg/dL	Yes
61	Male	Non-Anginal Pain	140/80 mmHg	260 mg/dL	80 mg/dL	Yes
34	Female	Asymptomatic	120/70 mmHg	180 mg/dL	85 mg/dL	No
47	Male	Atypical Angina	135/85 mmHg	230 mg/dL	95 mg/dL	Yes
56	Female	Non-Anginal Pain	145/92 mmHg	300 mg/dL	115 mg/dL	Yes
39	Male	Typical Angina	128/78 mmHg	190 mg/dL	90 mg/dL	No
48	Female	Asymptomatic	132/82 mmHg	220 mg/dL	100 mg/dL	No

 Regenerate response



 ChatGPT, while not the best choice when it comes to reliably assisting in what are the more

essential tasks in data science (yet), it has to be said that it does a pretty good job of generating made-up data. It does it so well, in fact, that we should start thinking seriously about how to control whether the data we are working on is real or perhaps made-up by AI. The problem is that at first glance it can be difficult (if at all) to tell whether we are dealing with real data or AI-generated data.

Data Import and Review

Load and Preview Raw Data

```
In [2]: # Import of necessary packages
import pandas as pd
import missingno as miss

# Data Loading
df = pd.read_csv('data/data_heartdisease.csv', index_col='id')

# Data preview
df.head(n=10)
```

```
Out[2]:    age   sex  chest_pain_type  resting_blood_presure  serum_cholesterol  fasting_blood_sugar
          id
1  47 Female      Non-Anginal Pain           128                 245                  0
2  62 Male        Typical Angina            130                 267                  1
3  58 Female      Atypical Angina           110                 217                  0
4  44 Male        Asymptomatic             140                 235                  0
5  50 Female      Non-Anginal Pain           140                 263                  0
6  66 Male        Asymptomatic              120                 338                  0
7  52 Female      Atypical Angina           120                 243                  1
8  58 Male        Non-Anginal Pain           150                 220                  0
9  42 Male        Atypical Angina           140                 226                  0
10 51 Female      Asymptomatic              120                 295                  0
```

Display Data Types

```
In [3]: # Datatypes preview
df.dtypes
```

```
Out[3]: age           int64
          sex            object
          chest_pain_type    object
          resting_blood_pressure   int64
          serum_cholesterol     int64
          fasting_blood_sugar    int64
          heart_disease_diagnosis object
          dtype: object
```

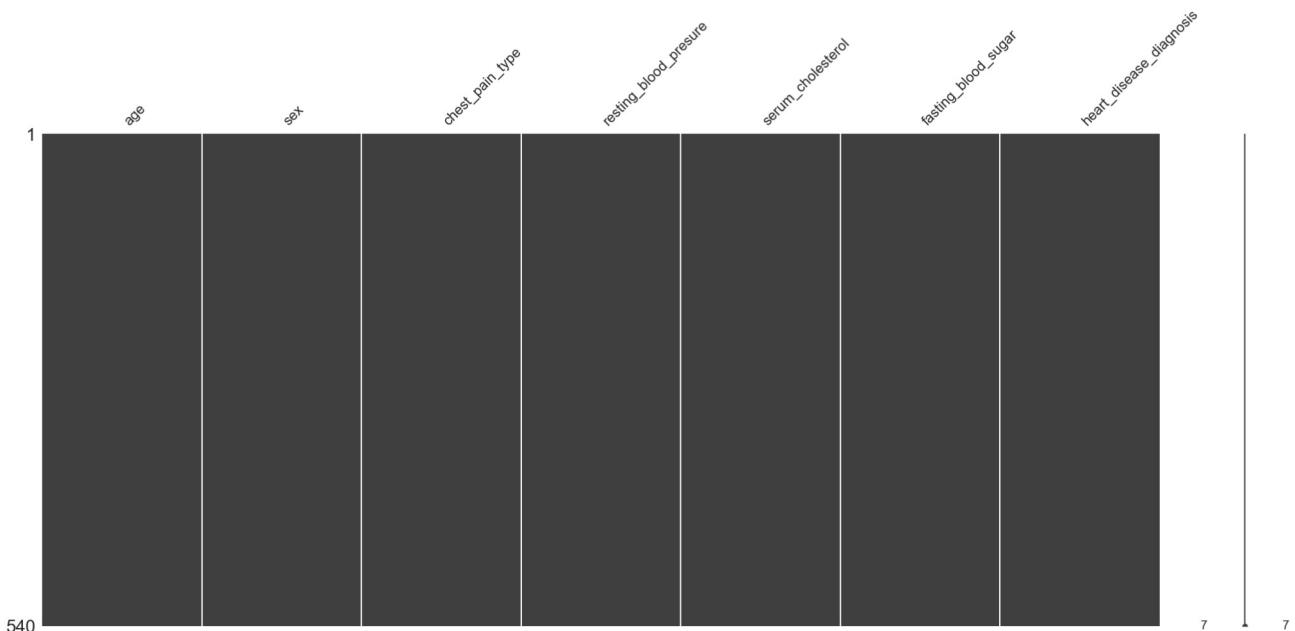
Findings:

- Variables `age`, `resting_blood_pressure`, `serum_cholesterol`, and `fasting_blood_sugar` are numerical.
- Variables `sex`, `chest_pain_type`, and `heart_disease_diagnosis` are textual.

Check Data for Missing Values

```
In [4]: # Check data for missing values
miss.matrix(df)
```

```
Out[4]: <Axes: >
```



Findings:

- There are no missing data in the data set.

Display Unique Values of Qualitative Variables

```
In [5]: for col in df:
    if (df[col].dtype == 'object') or (len(df[col].unique()) < 10):
        print(f'{col}', df[col].unique())
```

```
sex ['Female' 'Male']
chest_pain_type ['Non-Anginal Pain' 'Typical Angina' 'Atypical Angina' 'Asymptomatic']
fasting_blood_sugar [0 1]
heart_disease_diagnosis ['No' 'Yes']
```

Exploratory Data Analysis

Descriptive Statistics for Age, Resting Blood Pressure and Serum Cholesterol in Respect to Sex

```
In [6]: # Import of necessary packages
import seaborn as sns
import matplotlib.pyplot as plt

# Split data into males and females (based on 'sex' column).
df_m = df[df['sex'] == 'Male']
df_f = df[df['sex'] == 'Female']

# Set the format for displaying results by describe function.
pd.set_option('display.float_format', lambda x: '%.2f' % x)

# View results.
print("==> Males")
print(df_m.loc[:, df_m.columns != 'fasting_blood_sugar'].describe())
print("\n==> Females")
print(df_f.loc[:, df_f.columns != 'fasting_blood_sugar'].describe(), '\n')

==== Males
      age  resting_blood_pressure  serum_cholesterol
count  271.00                  271.00                271.00
mean   53.14                  130.21                246.23
std    6.34                   10.13                34.86
min    40.00                  105.00               141.00
25%   49.00                  120.00               223.00
50%   54.00                  130.00               239.00
75%   58.00                  138.00               263.00
max    67.00                  170.00               354.00

==== Females
      age  resting_blood_pressure  serum_cholesterol
count  269.00                  269.00                269.00
mean   53.10                  130.68                250.24
std    6.36                   11.93                43.12
min    39.00                  105.00               149.00
25%   48.00                  120.00               221.00
50%   53.00                  130.00               244.00
75%   58.00                  140.00               270.00
max    68.00                  170.00               564.00
```

Findings:

- The number of observations for both sexes are almost equal ($N_{Males} = 271$; $N_{Females} = 269$).
- The average `age` for both sexes is almost equal ($M_{Males} = 53.14$; $M_{Females} = 53.10$).
- The average `resting_blood_pressure` for both sexes is almost equal ($M_{Males} = 130.21$; $M_{Females} = 130.68$).
- The average `serum_cholesterol` for females is slightly higher ($M_{Males} = 246.23$; $M_{Females} = 250.24$).

Distributions of Age, Resting Blood Pressure and Serum

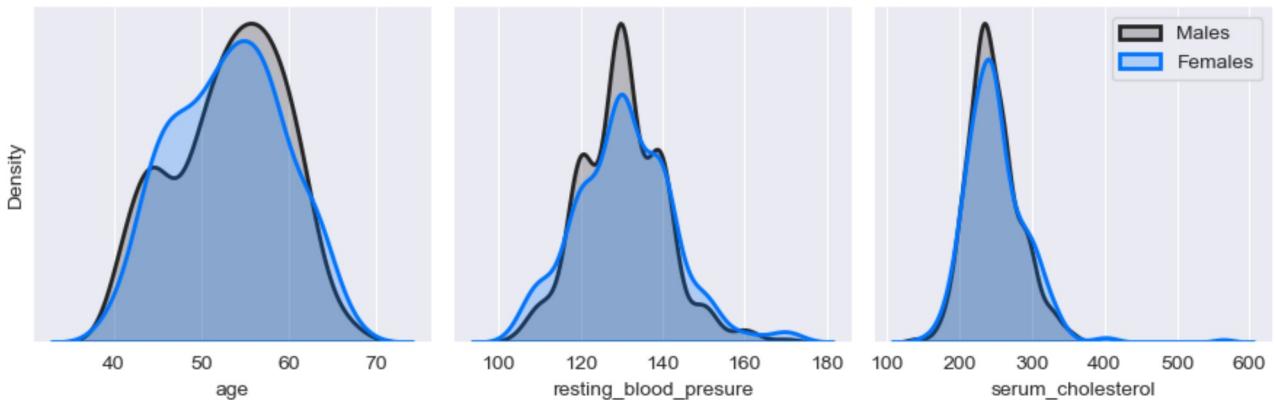
Cholesterol in Respect to Sex

```
In [7]: # Define a grid row for charts
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(9, 3))

# Define params for charts
cls = {'Male': '#242424', 'Female': '#0075FF'}
pms = {'fill': True, 'lw': 2}
vs = ['age', 'resting_blood_pressure', 'serum_cholesterol']

for a, v in zip(axes, vs):
    sns.kdeplot(data=df_m, x=v, fill=pms['fill'], linewidth=pms['lw'], color=cls['Male'])
    sns.kdeplot(data=df_f, x=v, fill=pms['fill'], linewidth=pms['lw'], color=cls['Female'])
    a.set_yticks([])
    a.set_yticklabels([])
    if v != 'age':
        a.set_ylabel('')
    a.spines['top'].set_visible(False)
    a.spines['right'].set_visible(False)

# Add legend and adjust layout
plt.legend(('Males', 'Females'), loc=0)
plt.tight_layout()
```



Findings:

- ✓ The distributions of observation counts by sex do not differ at first glance for the variables `age`, `blood_rest` and `serum_cholesterol`.
- ✓ The distribution of the `age` variable appears to be most similar to a normal distribution.
- ✓ The distributions of the `resting_blood_pressre` and `serum_cholesterol` variables appear to have positive skewness values indicating that they are right-skewed. This means that the median is lower than the average.

Observation Frequencies for Levels of Qualitative Variables by Sex

```
In [8]: # Import of necessary packages
import matplotlib

# Create contingency (cross) tables
cs_cpt = pd.crosstab(index=df['sex'], columns=df['chest_pain_type'])
cs_fbs = pd.crosstab(index=df['sex'], columns=df['fasting_blood_sugar'])
cs_hdd = pd.crosstab(index=df['sex'], columns=df['heart_disease_diagnosis'])
cs = [cs_cpt, cs_fbs, cs_hdd]

# Define a grid row for charts
```

```

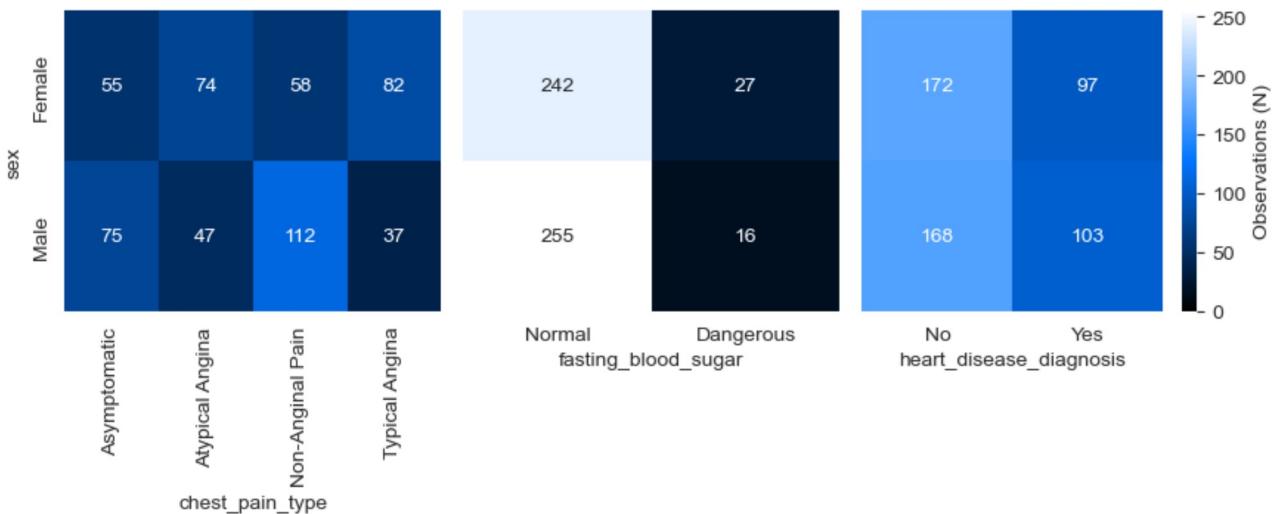
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(9, 4))
fig.suptitle('Observation frequencies expressed as COUNTS for levels of qualitative variables')
cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", ["black", "#0075FF", "white"])
vmin, vmax = 0, cs_fbs.values.max()

# # Plot cross tables
for i, (ax, c) in enumerate(zip(axes, cs)):
    kwargs = dict(vmin=vmin, vmax=vmax, cmap=cmap, annot=True, fmt='%.0f', robust=True, ax=ax)
    if i == 0:
        sns.heatmap(c, cbar=False, **kwargs)
    elif i >= 1:
        if i == 1:
            sns.heatmap(c, cbar=False, **kwargs)
            ax.set_xticklabels(['Normal', 'Dangerous'])
        else:
            sns.heatmap(c, cbar_kws={'label': 'Observations (N)'}, **kwargs)
            ax.set_yticks([])
            ax.set_yticklabels([])
            ax.set_ylabel('')

plt.tight_layout()

```

Observation frequencies expressed as COUNTS for levels of qualitative variables by sex.



```

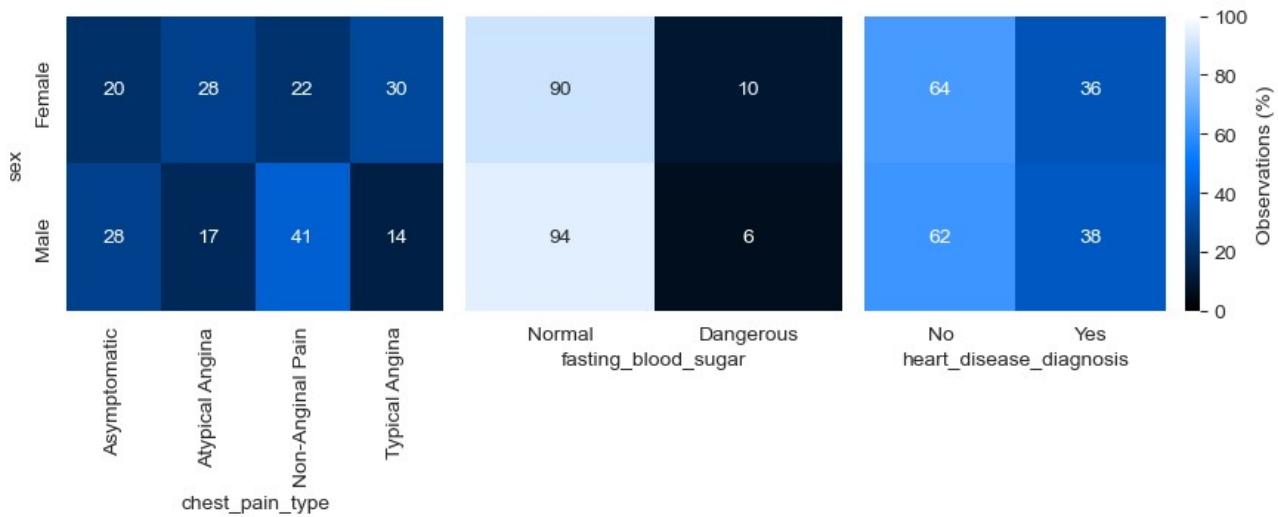
In [9]: # Define a grid row for charts
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(9, 4))
fig.suptitle('Observation frequencies expressed as PERCENTAGES for levels of qualitative variables')
cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", ["black", "#0075FF", "white"])
vmin, vmax = 0, 100

# # Plot cross tables
for i, (ax, c) in enumerate(zip(axes, cs)):
    c = c.apply(lambda r: (r / r.sum()) * 100.0, axis=1) # Convert counts to percentages
    kwargs = dict(vmin=vmin, vmax=vmax, cmap=cmap, annot=True, fmt='%.0f', robust=True, ax=ax)
    if i == 0:
        sns.heatmap(c, cbar=False, **kwargs)
    elif i >= 1:
        if i == 1:
            sns.heatmap(c, cbar=False, **kwargs)
            ax.set_xticklabels(['Normal', 'Dangerous'])
        else:
            sns.heatmap(c, cbar_kws={'label': 'Observations (%)'}, **kwargs)
            ax.set_yticks([])
            ax.set_yticklabels([])
            ax.set_ylabel('')

plt.tight_layout()

```

Observation frequencies expressed as PERCENTAGES for levels of qualitative variables by sex.



Findings:

- Only for the variable `chest_pain_type` do the observation frequencies appear to differ between sex at first glance.
- A [chi-square test of independence](#) can be used to verify the existence of a potential association between the variable `sex` and `chest_pain_type`.

Descriptive Statistics for Age, Serum Cholesterol and Resting Blood Pressure in Respect to Heart Disease Diagnosis

```
In [10]: # Generate descriptive statistics.
df_y = df[df['heart_disease_diagnosis'] == 'Yes']
df_n = df[df['heart_disease_diagnosis'] == 'No']
vars = ['age', 'serum_cholesterol', 'resting_blood_pressure']

print("==> Heart Disease: YES")
print(df_y.loc[:, vars].describe())
print("\n==> Heart Disease: NO")
print(df_n.loc[:, vars].describe(), '\n')
```

==> Heart Disease: YES

	age	serum_cholesterol	resting_blood_pressure
count	200.00	200.00	200.00
mean	56.60	273.36	134.49
std	5.70	43.49	11.47
min	41.00	166.00	108.00
25%	54.00	245.00	130.00
50%	57.00	272.00	130.00
75%	61.00	294.25	140.00
max	68.00	564.00	170.00

==> Heart Disease: NO

	age	serum_cholesterol	resting_blood_pressure
count	340.00	340.00	340.00
mean	51.07	233.44	128.06
std	5.79	27.24	10.09
min	39.00	141.00	105.00
25%	46.75	216.00	120.00
50%	51.50	234.00	130.00
75%	55.00	247.00	138.00
max	66.00	309.00	160.00

Scatterings of Heart Disease Diagnoses According to Age, Serum Cholesterol and Resting Blood Pressure

💡 In the next step, using the `seaborn` package, I want to display 2x `jointplot` in one row of one figure. The package, unfortunately, does not allow this, so we need a simple class to make this possible.

```
In [11]: import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns

# Based on the answer provided on StackOverflow: https://stackoverflow.com/a/47664533/67
class SeabornJointPlotOnGrid:

    def __init__(self, seaborngrid, fig, subplot_spec):
        self.fig = fig
        self.sg = seaborngrid
        self.subplot = subplot_spec

        if isinstance(self.sg, sns.axisgrid.JointGrid):
            self._movejointgrid()

    def _finalize(self):
        self._movejointgrid()
        h = self.sg.ax_joint.get_position().height
        h2 = self.sg.ax_marg_x.get_position().height
```

```

        r = int(np.round(h / h2))

        self._resize()

        self.subgrid = gridspec.GridSpecFromSubplotSpec(r + 1, r + 1, subplot_spec=self.spec)

        self._moveaxes(self.sg.ax_joint, self.subgrid[1:, :-1])
        self._moveaxes(self.sg.ax_marg_x, self.subgrid[0, :-1])
        self._moveaxes(self.sg.ax_marg_y, self.subgrid[1:, -1])

    def _moveaxes(self, ax, gs):
        ax.remove()
        ax.figure = self.fig
        self.fig.axes.append(ax)
        self.fig.add_axes(ax)
        ax._subplotspec = gs
        ax.set_position(gs.get_position(self.fig))
        ax.set_subplotspec(gs)

    def _finalize(self):
        plt.close(self.sg.fig)
        self.fig.canvas.mpl_connect("resize_event", self._resize)
        self.fig.canvas.draw()

    def _resize(self):
        self.sg.fig.set_size_inches(self.fig.get_size_inches())

```

👉 Now we can finally move on to display what we want to display :)

In [14]:

```

import numpy as np

# Generate jointplots and place them on a common figure using a custom class.
fig = plt.figure(figsize=(12, 6))

gs = gridspec.GridSpec(1, 2)

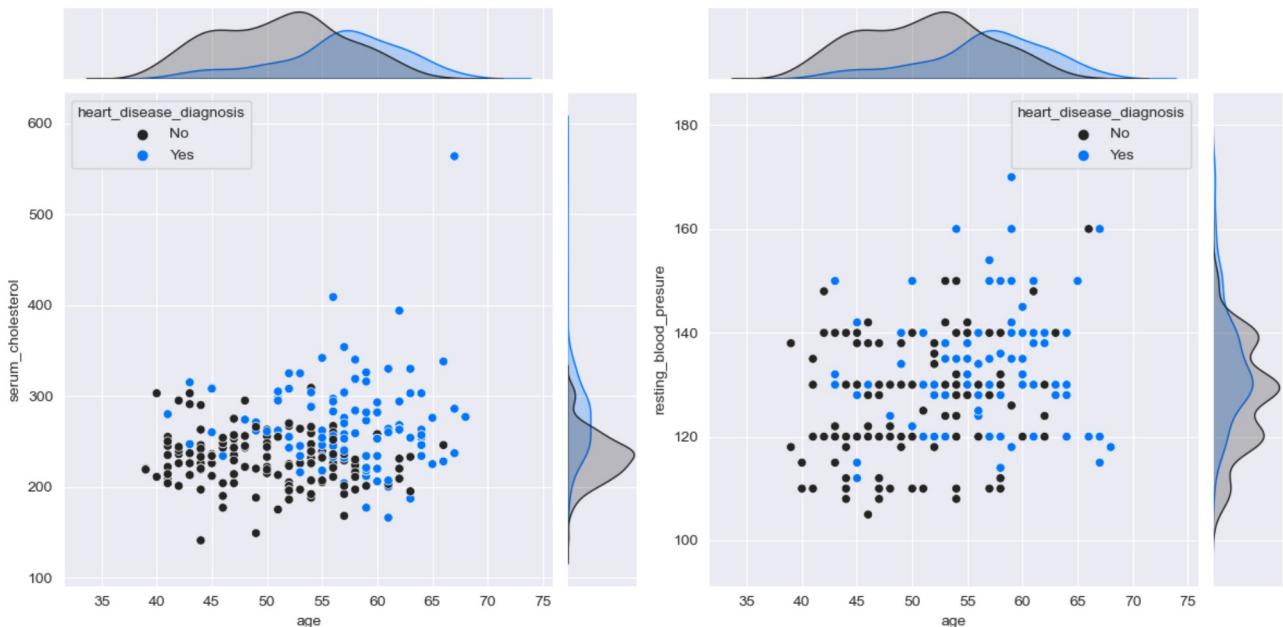
g1 = sns.jointplot(
    data=df,
    x='age',
    y='serum_cholesterol',
    hue='heart_disease_diagnosis',
    kind='scatter',
    palette=['#242424', '#0075FF']
)

g2 = sns.jointplot(
    data=df,
    x='age',
    y='resting_blood_pressre',
    hue='heart_disease_diagnosis',
    kind='scatter',
    palette=['#242424', '#0075FF']
)

mg1 = SeabornJointPlotOnGrid(g1, fig, gs[0])
mg2 = SeabornJointPlotOnGrid(g2, fig, gs[1])

gs.tight_layout(fig)

```



Findings:

- The number of people diagnosed with heart disease is lower than the number of people who have not been diagnosed. ($N_{Yes} = 200$; $N_{No} = 340$).
- People diagnosed with heart disease are older ($M_{Yes} = 56.60$; $M_{No} = 51.07$).
- People diagnosed with heart disease have higher cholesterol levels ($M_{Yes} = 273.36$; $M_{No} = 233.44$).
- People diagnosed with heart disease have higher resting blood pressure ($M_{Yes} = 134.49$; $M_{Females} = 128.06$).
- The [logistic regression model](#) can be used to make predictions about the risk of heart disease based on age, cholesterol levels and resting blood pressure.

Statistical Inference

Examining the Potential Associations Between Sex and Other Qualitative Variables

```
In [15]: # Calculate the chi-square test of independence between sex and the other qualitative variables
import scipy.stats as ss
qv = ['Chest Pain Type', 'Fasting Blood Sugar', 'Heart Disease Diagnosis']
for c, v in zip(cs, qv):
    chi2, p_value, dof, expected = ss.chi2_contingency(c, correction=False)
    if p_value < 0.05:
        p_value = str(p_value) + ' (*)'
    print(f'Sex vs {v}:')
    print(f"\u03c7\u00b2({dof}) = {chi2}, p = {p_value}\n")
print('Mark significant association for alpha level 0.05 with asterisk (*).')
```

```
Sex vs Chest Pain Type:  
χ²(3) = 43.26465043653495, p = 2.1623119758316382e-09 (*)
```

```
Sex vs Fasting Blood Sugar:  
χ²(1) = 3.1466294860551614, p = 0.07608397010787225
```

```
Sex vs Heart Disease Diagnosis:  
χ²(1) = 0.21965442921431155, p = 0.6393033521878453
```

Mark significant association for alpha level 0.05 with asterisk (*).

Findings:

- There is a statistically significant association between sex and chest_pain_type ($\chi^2(3) = 43.26, p < 0.05$).
- There are no significant associations between sex and fasting_blood_sugar ($\chi^2(1) = 3.15, p = 0.08$), as well as between sex and heart_disease_diagnosis ($\chi^2(1) = 0.22, p = 0.64$).

Modeling

Predictions About the Risk of Heart Disease Based on Age + Serum Cholesterol and Age + Resting Blood Pressure

```
In [16]: import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.linear_model import LogisticRegression  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import accuracy_score, confusion_matrix  
  
def model_and_plot(ind_var_1, ind_var_2, dep_var, axis):  
  
    # Define the input features and target variable  
    X = df[[ind_var_1, ind_var_2]]  
    y = df[dep_var].replace({'Yes': 1, 'No': 0})  
  
    # Split the data into training and testing sets  
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
  
    # Create the Logistic regression model  
    model = LogisticRegression()  
    model.fit(X_train.values, y_train.values)  
  
    # Make predictions on the testing set  
    y_pred = model.predict(X_test.values)  
  
    # Calculate the accuracy and confusion matrix  
    accuracy = accuracy_score(y_test, y_pred)  
    conf_mat = confusion_matrix(y_test, y_pred)  
  
    print(f"====> Accuracy of {dep_var} prediction based on {ind_var_1} and {ind_var_2}")  
    print("Accuracy =", round(accuracy, 4), '\n')  
    print("Confusion matrix:\n", conf_mat, '\n')  
  
    print("True positives (TP):", conf_mat[1,1])  
    print("False positives (FP):", conf_mat[0,1])
```

```

print("False negatives (FN):", conf_mat[1,0])
print("True negatives (TN):", conf_mat[0,0], '\n')

print("Precision = TP / (TP + FP) =", round(conf_mat[1,1] / (conf_mat[1,1] + conf_mat[1,0]), 2))
print("Recall = TP / (TP + FN) =", round(conf_mat[1,1] / (conf_mat[1,1] + conf_mat[0,1]), 2))

# Create a meshgrid of points to make the decision boundary plot
xx, yy = np.meshgrid(np.arange(X[ind_var_1].min()-1, X[ind_var_1].max()+1, 0.01), np.arange(X[ind_var_2].min()-1, X[ind_var_2].max()+1, 0.01))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary and the scatter plot
cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", ["#242424", "#0075FF"])

axis.contourf(xx, yy, Z, alpha=0.2, cmap=cmap)
axis.scatter(X[ind_var_1], X[ind_var_2], c=y.map({0: 0, 1: 1}), cmap=cmap, edgecolor='black', s=50)

axis.set_xlabel(ind_var_1)
axis.set_ylabel(ind_var_2)

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

model_and_plot(ind_var_1='age', ind_var_2='serum_cholesterol', dep_var='heart_disease_diagnosis')
model_and_plot(ind_var_1='age', ind_var_2='resting_blood_pressure', dep_var='heart_disease_diagnosis')

plt.tight_layout()

```

====> Accuracy of heart_disease_diagnosis prediction based on age and serum_cholesterol
Accuracy = 0.7037

Confusion matrix:

```

[[47 12]
 [20 29]]

```

True positives (TP): 29
False positives (FP): 12
False negatives (FN): 20
True negatives (TN): 47

Precision = TP / (TP + FP) = 0.7073
Recall = TP / (TP + FN) = 0.5918

====> Accuracy of heart_disease_diagnosis prediction based on age and resting_blood_pressure
Accuracy = 0.6944

Confusion matrix:

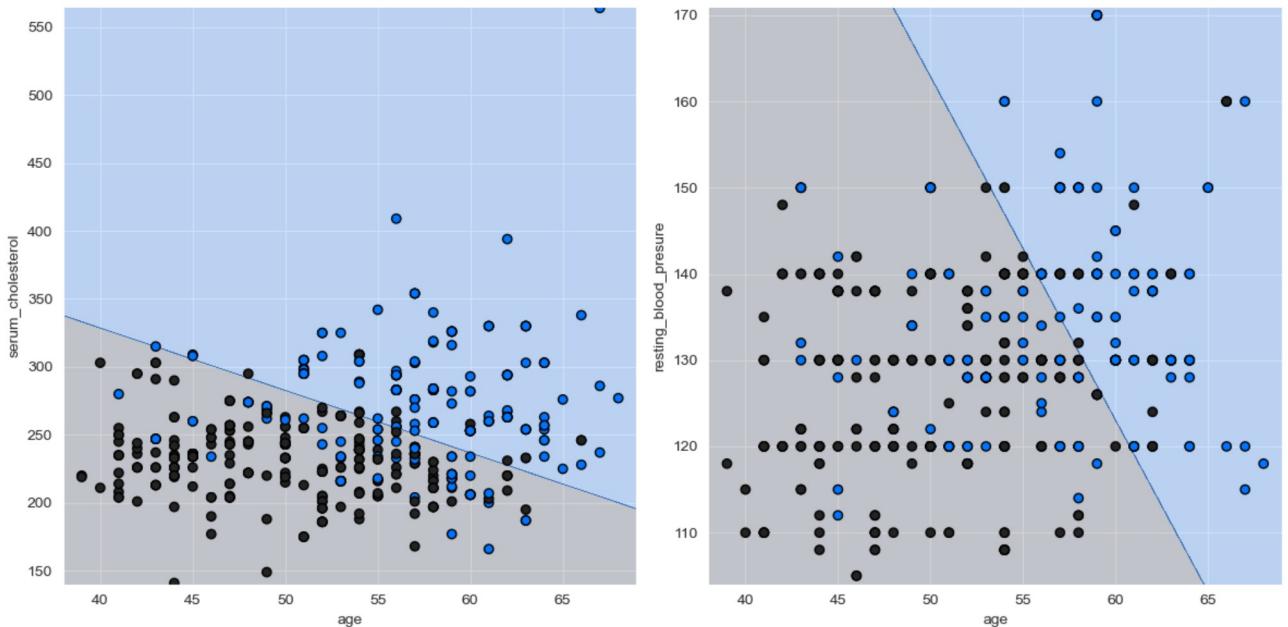
```

[[51  8]
 [25 24]]

```

True positives (TP): 24
False positives (FP): 8
False negatives (FN): 25
True negatives (TN): 51

Precision = TP / (TP + FP) = 0.75
Recall = TP / (TP + FN) = 0.4898



Findings:

- The logistic regression model based on `age` and `serum_cholesterol` has an accuracy of 0.7037, meaning that it correctly predicts the `heart_disease_diagnosis` 70.37% of the time.
- The confusion matrix for this model shows that it correctly identifies 29 true positives and 47 true negatives, but has 20 false negatives and 12 false positives.
- The precision of this model is 0.7073, indicating that out of all the positive predictions it made, 70.73% were correct.
- The recall of this model is 0.5918, indicating that out of all the actual positives, 59.18% were correctly identified by the model.
- The logistic regression model based on `age` and `resting_blood_pressure` has an accuracy of 0.6944, meaning that it correctly predicts the `heart_disease_diagnosis` 69.44% of the time.
- The confusion matrix for this model shows that it correctly identifies 24 true positives and 51 true negatives, but has 25 false negatives and 8 false positives.
- The precision of this model is 0.75, indicating that out of all the positive predictions it made, 75% were correct.
- The recall of this model is 0.4898, indicating that out of all the actual positives, 48.98% were correctly identified by the model.
- ❗ Based on the analysis, logistic regression models using `age` and `serum_cholesterol` or `age` and `resting_blood_pressure` as predictors were able to predict `heart_disease_diagnosis` with moderate accuracy. The precision and recall scores were also moderate. However, there is room for improvement in the models' performance, as evidenced by the misclassification of a significant number of cases. Further analysis and feature engineering may be necessary to improve the models' accuracy and effectiveness in predicting heart disease diagnosis based on the independent variables used in the current analysis.
- ❗ We can try combining the three independent variables `age`, `serum_cholesterol` and `resting_blood_pressure` into a single model to see if it significantly changes its accuracy in predicting `heart_disease_diagnosis`.

Predictions About the Risk of Heart Disease Based on Age + Serum Cholesterol + Resting Blood Pressure

```
In [17]: # Define the input features and target variable
X = df[['age', 'serum_cholesterol', 'resting_blood_pressure']]
```

```

y = df['heart_disease_diagnosis'].replace({'Yes': 1, 'No': 0})

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the logistic regression model
model = LogisticRegression()
model.fit(X_train.values, y_train.values)

# Make predictions on the testing set
y_pred = model.predict(X_test.values)

# Calculate the accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
conf_mat = confusion_matrix(y_test, y_pred)

print(f"====> Accuracy of heart_disease_diagnosis prediction based on age, serum_choleste
print("Accuracy =", round(accuracy, 4), '\n')
print("Confusion matrix:\n", conf_mat, '\n')

print("True positives (TP):", conf_mat[1, 1])
print("False positives (FP):", conf_mat[0, 1])
print("False negatives (FN):", conf_mat[1, 0])
print("True negatives (TN):", conf_mat[0, 0], '\n')

print("Precision = TP / (TP + FP) =", round(conf_mat[1, 1] / (conf_mat[1, 1] + conf_mat[0, 1]), 4))
print("Recall = TP / (TP + FN) =", round(conf_mat[1, 1] / (conf_mat[1, 1] + conf_mat[1, 0]), 4))

```

====> Accuracy of heart_disease_diagnosis prediction based on age, serum_cholesterol, and resting_blood_pressure

Accuracy = 0.7315

Confusion matrix:

```

[[47 12]
 [17 32]]

```

True positives (TP): 32

False positives (FP): 12

False negatives (FN): 17

True negatives (TN): 47

Precision = TP / (TP + FP) = 0.7273

Recall = TP / (TP + FN) = 0.6531

Findings:

- The logistic regression model based on age , serum_cholesterol and resting_blood_pressure has an accuracy of 0.7315, meaning that it correctly predicts the heart_disease_diagnosis 73.15% of the time.
- The confusion matrix for this model shows that it correctly identifies 32 true positives and 47 true negatives, but has 17 false negatives and 12 false positives.
- The precision of this model is 0.7273, indicating that out of all the positive predictions it made, 72.73% were correct.
- The recall of this model is 0.6531, indicating that out of all the actual positives, 65.31% were correctly identified by the model.
- Using the three features together explains heart_disease_diagnosis better than each of their two combinations, by some 3%.

Summary

About the data:

- There are no differences between men and women on the `age` , `resting_blood_pressre` , `fasting_blood_sugar` and `heart_disease_diagnosis` dimensions.
- Women, on average, have slightly higher cholesterol levels.
- There is a relationship between sex and the types of chest pain reported. Women were more likely to report typical and atypical anginas, while men were more likely to report non-anginal and asymptomatic pain.
- Age, cholesterol levels and resting blood pressure are relatively good predictors for estimating the risk of heart disease.

In general:

- ChatGPT can be used to generate interesting exercise and test data.