Лабораторная работа №1

#Цель лабораторной работы:

изучение различных методов визуализация данных и создание истории на основе данных.

#Задание:

Выбрать набор данных (датасет). Вы можете найти список свободно распространяемых датасетов здесь. Для лабораторных работ не рекомендуется выбирать датасеты очень большого размера.

Создать "историю о данных" в виде юпитер-ноутбука, с учетом следующих требований:

1.История должна содержать не менее 5 шагов (где 5 - рекомендуемое количество шагов). Каждый шаг содержит график и его текстовую интерпретацию.

2.На каждом шаге наряду с удачным итоговым графиком рекомендуется в юпитер-ноутбуке оставлять результаты предварительных "неудачных" графиков. 3.Не рекомендуется повторять виды графиков, желательно создать 5 графиков различных видов.

4.Выбор графиков должен быть обоснован использованием методологии data-to-viz. Рекомендуется учитывать типичные ошибки построения выбранного вида графика по методологии data-to-viz. Если методология Вами отвергается, то просьба обосновать Ваше решение по выбору графика.

5.История должна содержать итоговые выводы. В реальных "историях о данных" именно эти выводы представляют собой основную ценность для предприятия.

Сформировать отчет и разместить его в своем репозитории на github.

1.Основные характеристики набора данных

Подключим все необходимые библиотеки:

In [1]:

```
#libraries we are going to use
import numpy as np # linear algebra
import pandas as pd # data processing
from plotly.offline import iplot, init_notebook_mode #visualization
import plotly.express as px #visualization
from plotly.subplots import make_subplots #visualization
import plotly.graph_objects as go #visualization
import matplotlib.pyplot as plt #visualization
import seaborn as sns #visualization
from sklearn import preprocessing #label encoder
from sklearn.model_selection import train_test_split #split data
from sklearn.linear_model import LogisticRegression #model
from sklearn.model_selection import cross_val_score #cross validation
from sklearn.metrics import classification_report #reports
from sklearn import metrics #confusion metrics
from sklearn.model_selection import GridSearchCV #get best parameters
```

In [2]:

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

In [11]:

```
#convert file into dataset
data = pd.read_csv("heart.csv")
```

Понимание данных

In [12]:

```
#explore first five rows in the dataset
data.head()
```

Out[12]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exercis
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

In [13]:

```
#information about dataset, type, columns names, null
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

Набор данных содержит 12 столбцов с 918 строками. 7 столбцов числовые, а остальные категориальные, пропущенных значений нет.

In [14]:

```
#statistical description of data numeric columns
data.describe()
```

Out[14]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

In [15]:

```
#description of data object columns
data.select_dtypes(include=['object']).describe()
```

Out[15]:

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
count	918	918	918	918	918
unique	2	4	3	2	3
top	M	ASY	Normal	N	Flat
freq	725	496	552	547	460

In [16]:

```
#description of percent of data object columns
col=data.select_dtypes(include=['object']).columns.tolist()
#create iteration of object columns
for i in col:
    count=data[i].value_counts()
    percent=data.groupby(['HeartDisease'])[i].value_counts(normalize=True)[1]
    display(pd.DataFrame({"Patients":count, "Percent":percent*100})\
            .sort_values("Percent", ascending=False)
            .style.set_caption('Variable: {}'.format(i))\
            .format({"Percent": "{:,.1f}%"}))
```

Variable: Sex

	Patients	Percent
M	725	90.2%
F	193	9.8%

Variable: ChestPainType

	Patients	Percent
ASY	496	77.2%
NAP	203	14.2%
ATA	173	4.7%
TA	46	3.9%

Variable: RestingECG

	Patients	Percent
Normal	552	56.1%
ST	178	23.0%
LVH	188	20.9%

Variable: ExerciseAngina

	Patients	Percent
Y	371	62.2%
N	547	37.8%

Variable: ST_Slope

	Patients	Percent
Flat	460	75.0%
Up	395	15.4%
Down	63	9.6%

2.Выбросы

In [18]:

```
# Set up the subplots grid
fig = make_subplots(rows=2, cols=3,
                    # Set the subplot titles
                    subplot_titles=['Age', 'RestingBP', 'Cholesterol','MaxHR','Oldpeak'])
#create boxplot visualization of numeric columns
fig.add_trace(go.Box(x=data.Age, name='', showlegend=False), row=1, col=1)
fig.add_trace(go.Box(x=data.RestingBP, name='', showlegend=False), row=1, col=2)
fig.add_trace(go.Box(x=data.Cholesterol, name='', showlegend=False), row=1, col=3)
fig.add_trace(go.Box(x=data.MaxHR, name='', showlegend=False), row=2, col=1)
fig.add_trace(go.Box(x=data.Oldpeak, name='', showlegend=False), row=2, col=2)

#config size
fig.update_layout(height=500, width=900)
#show visualizations
fig.show()
```

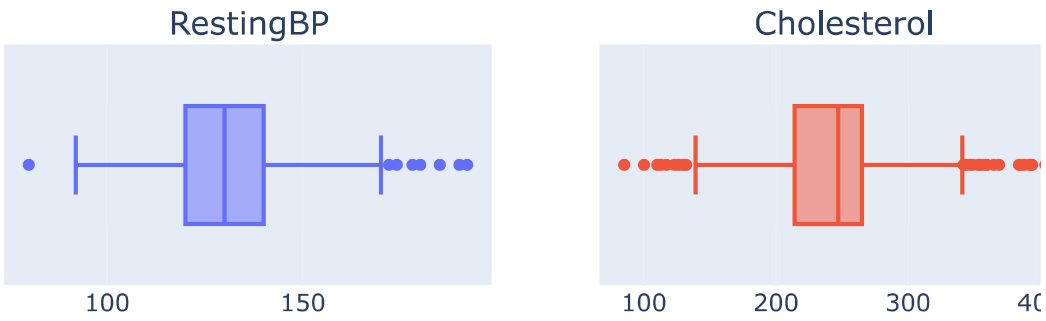
удаление выбросов немного сложно и противоречиво, потому что есть данные, которые являются реальными, но считаются выбросами, и их удаление приведет к переобучению нашей модели, но также есть ошибки при вводе значений, например, в холестерине у нас есть значения 0, и это что-то это не возможно, но иметь высокий уровень холестерина да, мы можем автоматически обнаруживать выбросы с помощью различных алгоритмов, но в этом случае я собираюсь изменить данные, которые равны 0 или очень низкие, и те, которые очень высоки по среднему значению

In [19]:

```
#creating conditions to change values to nan
conC = (data["Cholesterol"] < 78) | (data["Cholesterol"] >457)
conR = (data["RestingBP"] < 80) | (data["RestingBP"] >192)
#change values to nan
data.loc[conC,'Cholesterol'] = np.nan
data.loc[conR,'RestingBP'] = np.nan
#fill nan values to mean by group of heart disease
data['Cholesterol'] = data['Cholesterol'].fillna(data.groupby('HeartDisease')['Cholesterol'].transfo
data['RestingBP'] = data['RestingBP'].fillna(data.groupby('HeartDisease')['RestingBP'].transform('me
```

In [20]:

```
# Set up the subplots grid
fig = make_subplots(rows=1, cols=2,
                    # Set the subplot titles
                    subplot_titles=['RestingBP', 'Cholesterol'])
fig.add_trace(go.Box(x=data.RestingBP, name='', showlegend=False), row=1, col=1)
fig.add_trace(go.Box(x=data.Cholesterol, name='', showlegend=False), row=1, col=2)
#config size
fig.update_layout(height=300, width=700)
#show visualizations
fig.show()
```



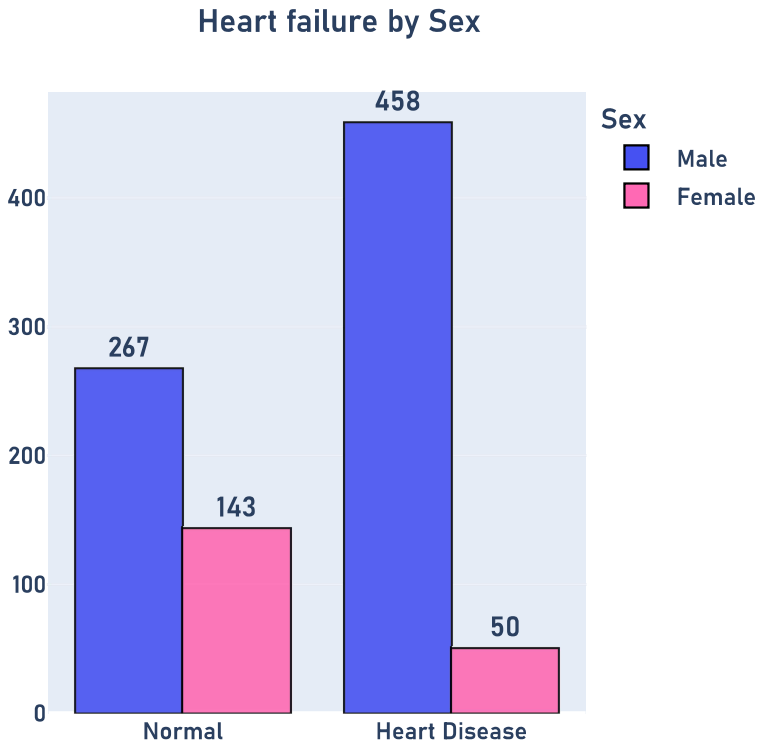
3.Исследование данных

In [21]:

```
#creating a copy of dataset for the visualization
eda= data.copy()
#change values to make better visualizations
eda['Sex'] = np.where(eda['Sex'] == 'F', 'Female', 'Male')
eda['HeartDisease'] = np.where(eda['HeartDisease'] == 0, 'Normal', 'Heart Disease')
eda['ExerciseAngina'] = np.where(eda['ExerciseAngina'] == 'N', 'No', 'Yes')
eda["ChestPainType"].replace({'TA': 'Typical Angina', 'ATA': 'Atypical Angina',
                              'NAP': 'Non-Anginal Pain', 'ASY': 'Asymptomatic'}, inplace= True)
eda["ST_Slope"].replace({'Up': 'Upsloping', 'Down': 'Downsloping',}, inplace= True)
```

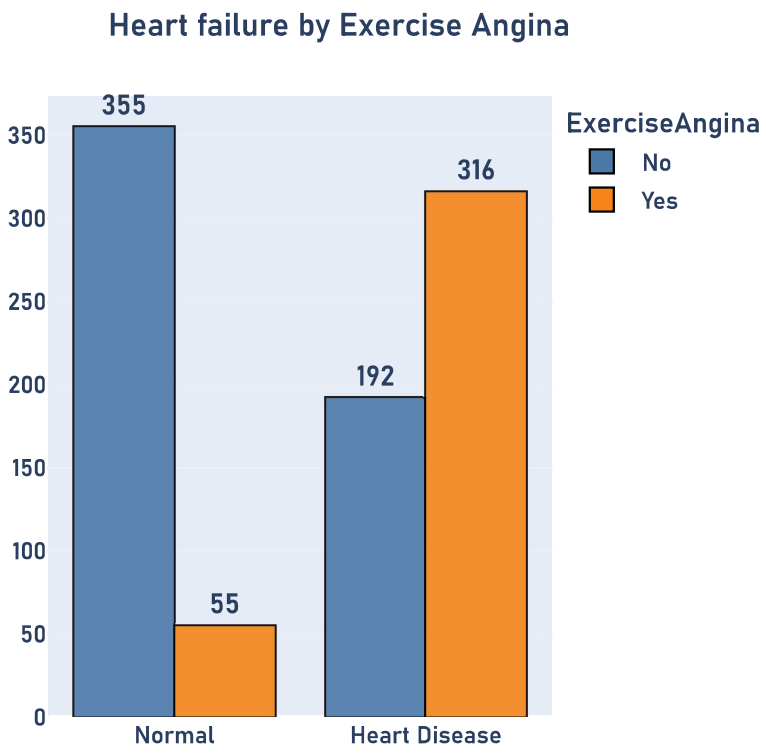
In [22]:

```
#barplot of heart failure by gender
my_scale = ['rgb(70,81,242)','rgb(255,105,180)']
#histogram with plotly
df = px.data.tips()
fig = px.histogram(eda, x="HeartDisease",
             color='Sex', barmode='group',
             color_discrete_sequence =my_scale, opacity=0.9,text_auto=True,
             height=450, width = 450)
fig.update_layout(title_text='Heart failure by Sex',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                 marker_line_width=1,marker_line_color="black")


fig.show()
```



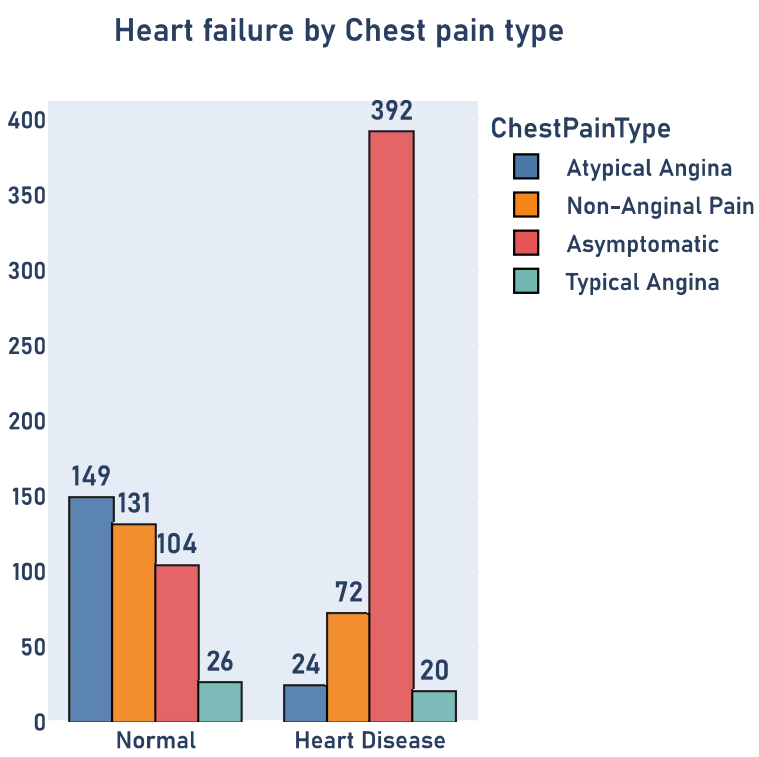
In [23]:

```
df = px.data.tips()
fig = px.histogram(eda, x="HeartDisease",
             color='ExerciseAngina', barmode='group',
                      color_discrete_sequence= px.colors.qualitative.T10,opacity=0.9,text_auto=Tru
             height=450, width = 450)
fig.update_layout(title_text='Heart failure by Exercise Angina',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                 marker_line_width=1,marker_line_color="black")


fig.show()
```



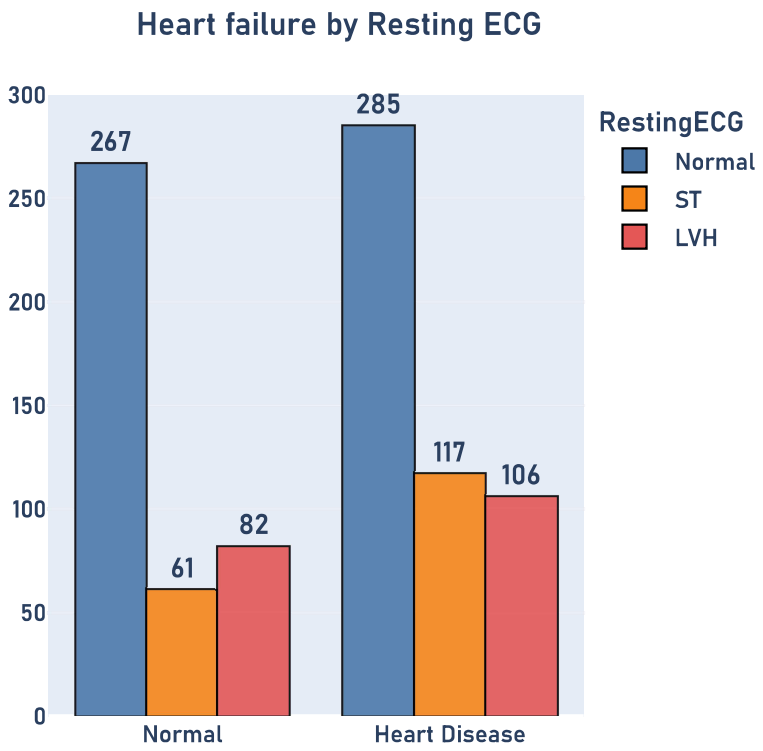
In [24]:

```
#barplot of heart failure by Chest pain type
df = px.data.tips()
fig = px.histogram(eda, x="HeartDisease",
             color='ChestPainType', barmode='group',
                      color_discrete_sequence= px.colors.qualitative.T10,opacity=0.9,text_auto=Tru
             height=450, width = 450)
fig.update_layout(title_text='Heart failure by Chest pain type',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                 marker_line_width=1,marker_line_color="black")


fig.show()
```



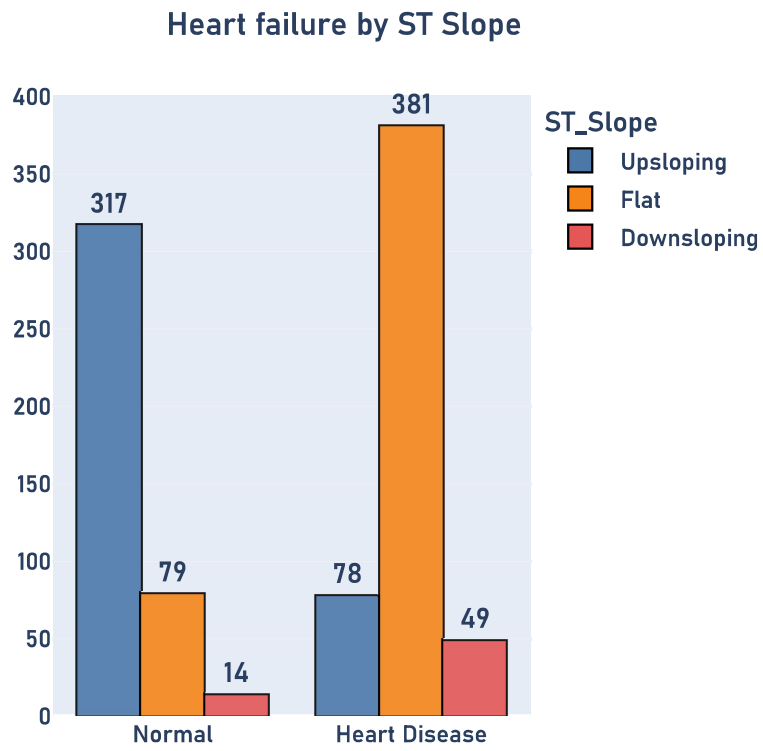
In [25]:

```
#barplot of heart failure by Resting ECG
df = px.data.tips()
fig = px.histogram(eda, x="HeartDisease",
             color='RestingECG', barmode='group',
              color_discrete_sequence= px.colors.qualitative.T10, opacity=0.9,text_auto=True,
             height=450, width = 450)
fig.update_layout(title_text='Heart failure by Resting ECG',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                 marker_line_width=1,marker_line_color="black")


fig.show()
```



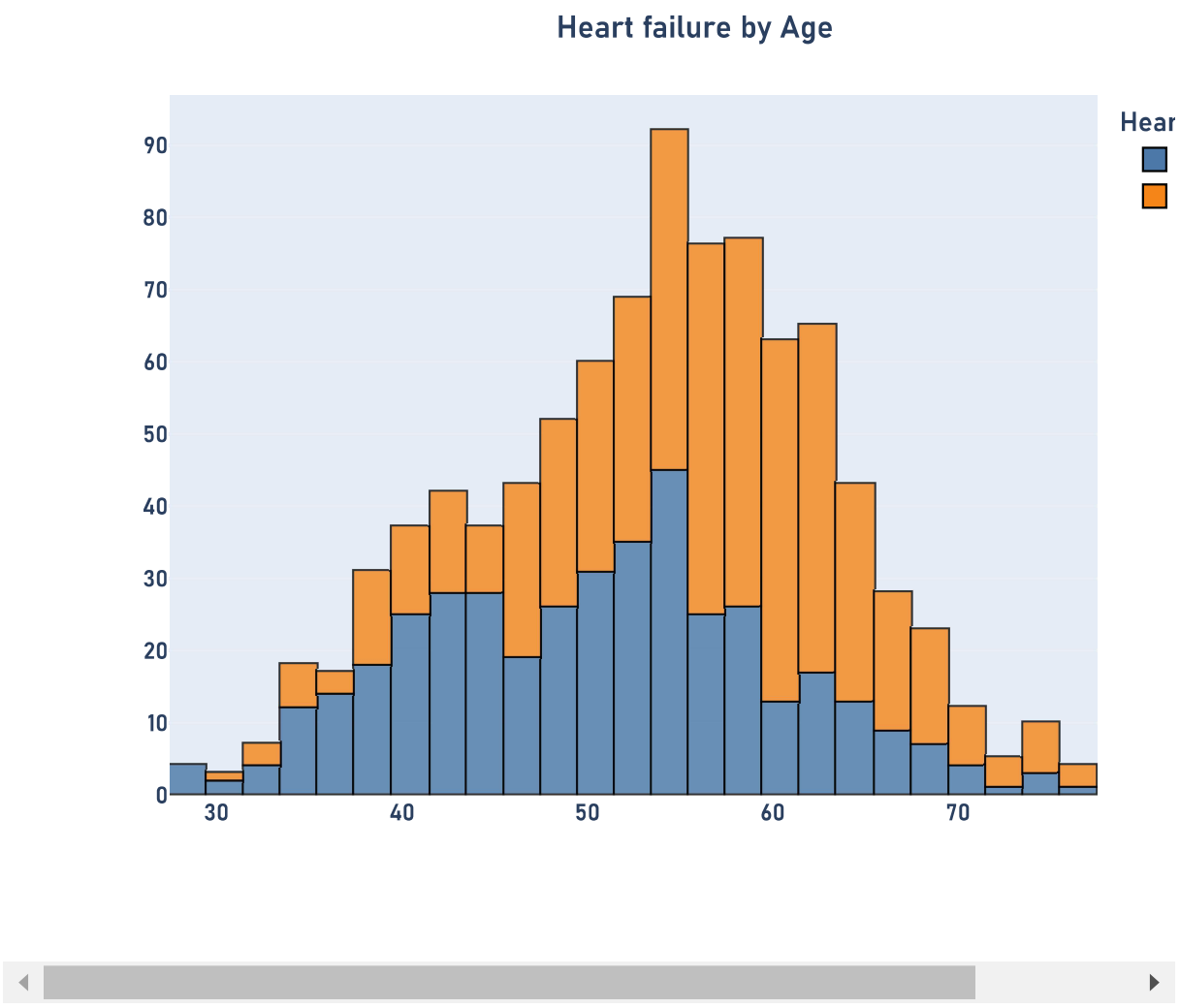
In [26]:

```
#barplot of heart failure by ST Slope
df = px.data.tips()
fig = px.histogram(eda, x="HeartDisease",
             color='ST_Slope', barmode='group',
                    color_discrete_sequence= px.colors.qualitative.T10, opacity=0.9,text_auto=True,
             height=450, width = 450)
fig.update_layout(title_text='Heart failure by ST Slope',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                 marker_line_width=1,marker_line_color="black")


fig.show()
```



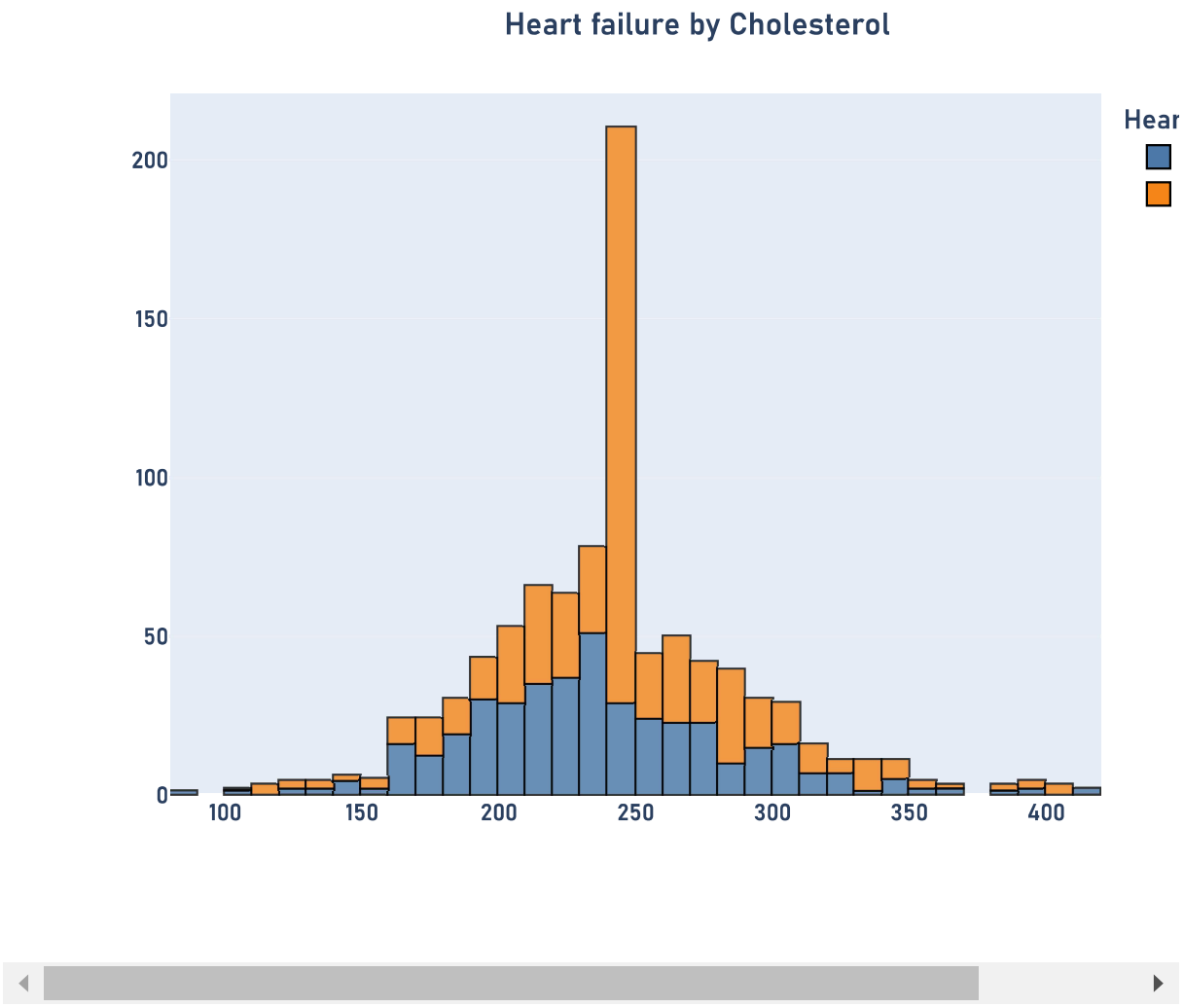
In [27]:

```
#barplot of heart failure by Age
df = px.data.tips()
fig = px.histogram(eda, x="Age",
             color='HeartDisease',color_discrete_sequence= px.colors.qualitative.T10,
                   nbins = 40,opacity=0.8,height=500, width = 700)
fig.update_layout(title_text='Heart failure by Age',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                marker_line_width=1,marker_line_color="black")


fig.show()
```



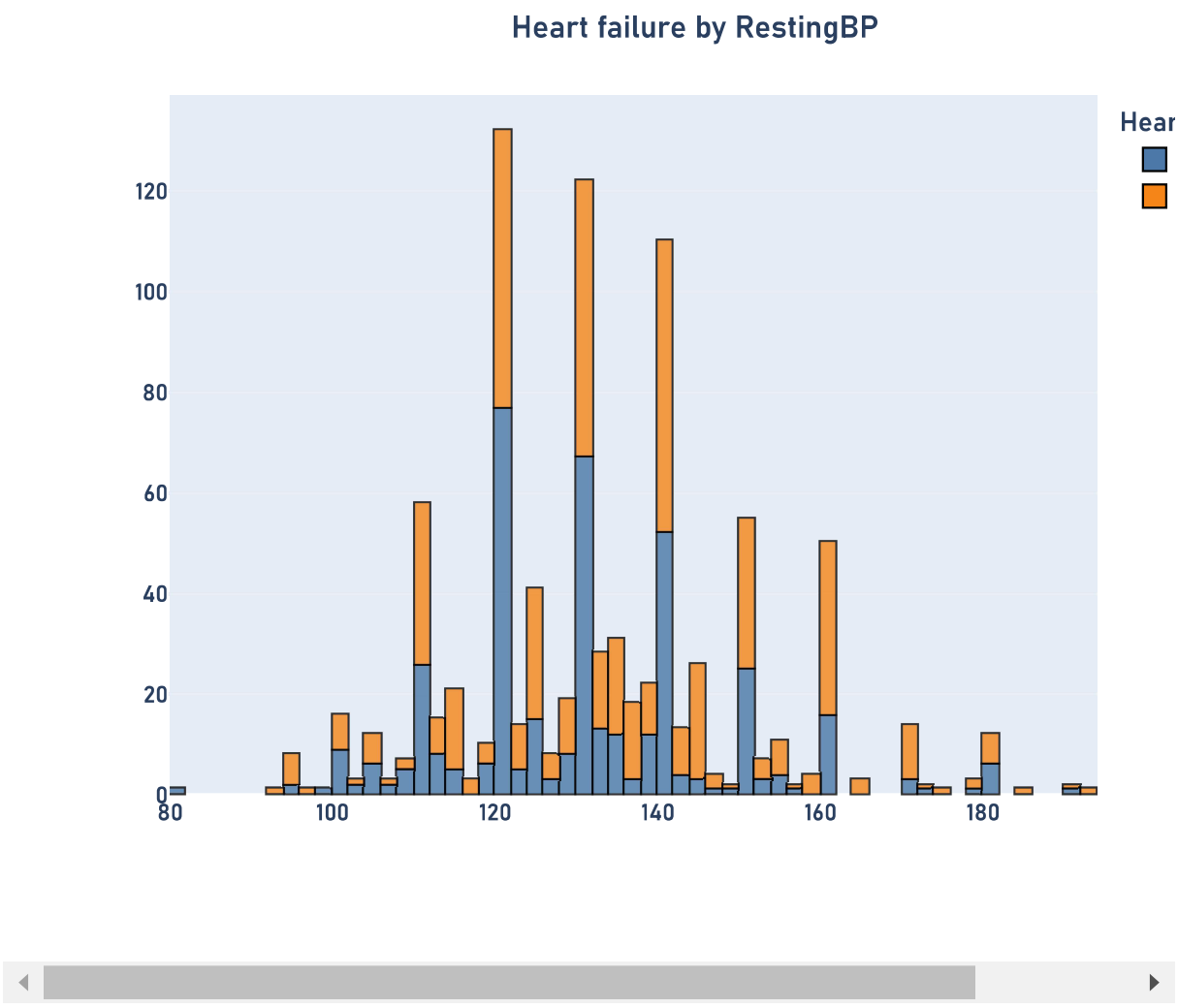
In [38]:

```
#barplot of heart failure by Age and Sex
df = px.data.tips()
fig = px.histogram(eda, x="Cholesterol",
             color='HeartDisease',color_discrete_sequence= px.colors.qualitative.T10,
                   nbins = 40,opacity=0.8,height=500, width = 700)
fig.update_layout(title_text='Heart failure by Cholesterol',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                  marker_line_width=1,marker_line_color="black")
fig.show()
```



In [29]:

```
#barplot of heart failure by Age and Sex
df = px.data.tips()
fig = px.histogram(eda, x="RestingBP",
             color='HeartDisease',color_discrete_sequence= px.colors.qualitative.T10,
                   nbins = 60,opacity=0.8,height=500, width = 700)
fig.update_layout(title_text='Heart failure by RestingBP',title_font_size=16, title_x=0.5,
                  font_family='Bahnschrift SemiBold',
                  yaxis_title=None, xaxis_title=None)
fig.update_traces(textfont_size=14, textangle=0, textposition="outside", cliponaxis=False,
                 marker_line_width=1,marker_line_color="black")


fig.show()
```



4.Построение модели

4.1Подготовить данные

In [30]:

```
#convert columns to binary, just these 2 columns
data['Sex'] = np.where(data['Sex'] == 'M', 1, 0)
data['ExerciseAngina'] = np.where(data['ExerciseAngina'] == 'Y', 1, 0)

#create target value and label
y=data.HeartDisease
X=data.drop('HeartDisease', axis=1)
#convert to binary
X=pd.get_dummies(X)
```

4.2Создать модель

In [31]:

```
#creating the parameters
c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space, 'penalty': ['l1', 'l2']}

#ML model
logreg = LogisticRegression(solver='liblinear',max_iter=10000)

#separete data 75% train 25% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, stratify=


#find the best parameter
logreg_cv = GridSearchCV(logreg, param_grid, cv=5)

# fit the model with data
logreg_cv.fit(X_train, y_train)

print("Tuned Logistic Regression Parameter: {}".format(logreg_cv.best_params_))
print("Tuned Logistic Regression Accuracy: {}".format(logreg_cv.best_score_))
```

```
Tuned Logistic Regression Parameter: {'C': 0.05179474679231213, 'penalty': 'l2'}
Tuned Logistic Regression Accuracy: 0.8459219295461757
```

In [32]:

```
#create the ML model with the parameters
logreg2=LogisticRegression(C=0.051, penalty="l2")
logreg2.fit(X_train,y_train)
print("score",logreg2.score(X_test,y_test))
```

```
score 0.8782608695652174
```

In [33]:

```
#predict data
y_pred=logreg2.predict(X_test)
```

5.Результаты

5.1Матрица путаницы

In [34]:

```
#create confusion matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[34]:

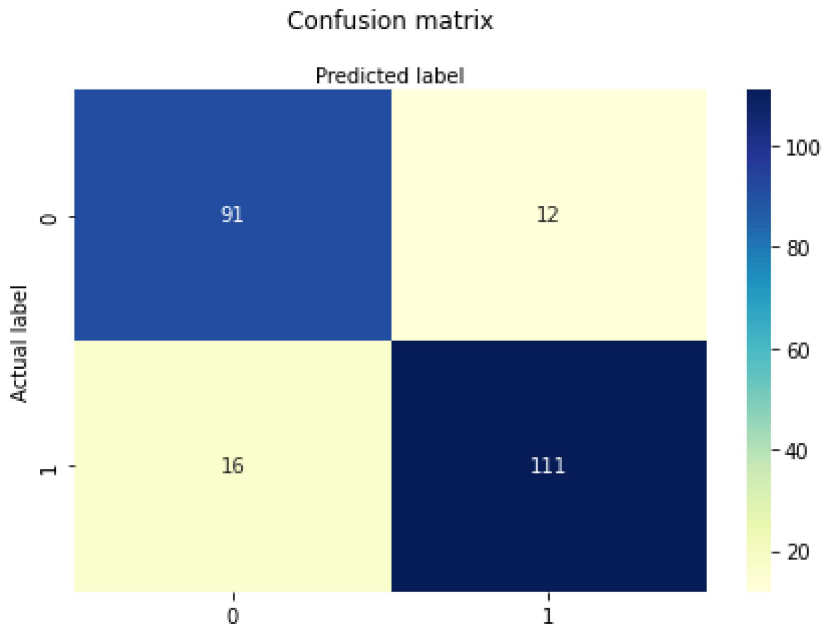
```
array([[ 91,  12],
       [ 16, 111]], dtype=int64)
```

In [35]:

```
#create hetmap of confusion matrix
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[35]:

```
Text(0.5, 257.44, 'Predicted label')
```



In [36]:

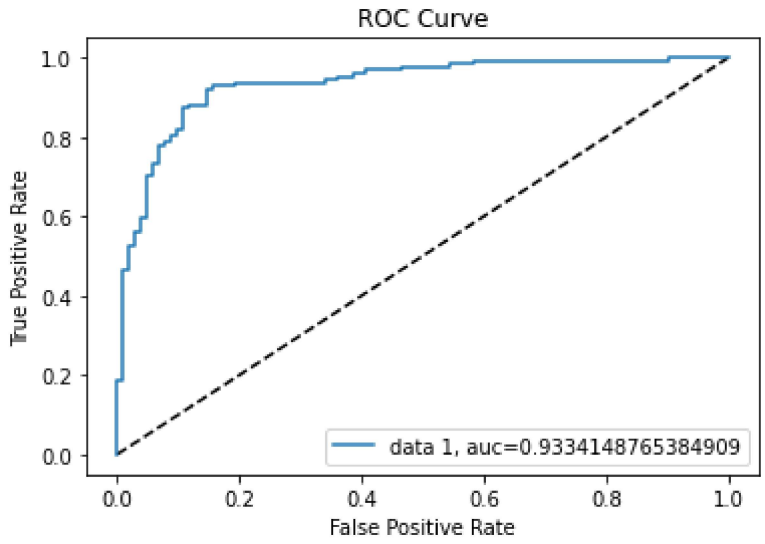
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.8782608695652174
Precision: 0.9024390243902439
Recall: 0.8740157480314961
```

5.2Кривая Рока

In [37]:

```
y_pred_proba = logreg2.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc=4)
plt.show()
```



In [ ]:

In [ ]:

In [ ]: