```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objects as go


%matplotlib inline


data = pd.read_csv('heart_disease_uci.csv')
data.head()


data.dropna(inplace = True)
from sklearn.utils import shuffle
data = shuffle(data)
data.info()

fig = go.Figure()
fig.add_trace(go.Box(y=data['age'].values , name='Age', marker_color =
'green',boxmean=True))
fig.add_trace(go.Box(y=data[data['sex']=='Male']['age'].values, name ='Male
only', marker_color = 'blue', boxmean = True))
fig.add_trace(go.Box(y=data[data['sex']=='Female']['age'].values, name ='Female
only', marker_color = 'red', boxmean = True))
fig.update_layout(title = 'Age Distribution(all)', yaxis_title = 'Age', title_x
= 0.5)
fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.show()


group_labels = ['Age Distribution'] # name of the dataset
fig = ff.create_distplot([data.age], group_labels)
fig.update_layout(title = 'Age Distribution(all)', yaxis_title = 'propotion',
xaxis_title = 'Age', title_x = 0.5)
fig.show()


df=data['sex'].value_counts().reset_index().rename(columns={'index':'sex','sex':
'count'})
fig = go.Figure([go.Pie(labels=['Male', 'Female'],values=df['count'], hole =
0.5)])
fig.update_traces(hoverinfo='label+percent', textinfo='value+percent',
textfont_size=15,insidetextorientation='radial')
fig.update_layout(title="Male to Female ratio in the study",title_x=0.5)
fig.show()

df=data['dataset'].value_counts().reset_index().rename(columns={'index':'dataset
','dataset':'count'})
fig = go.Figure([go.Pie(labels=df['dataset'],values=df['count'], hole = 0.5)])
fig.update_traces(hoverinfo='label+percent', textinfo='value+percent',
textfont_size=15,insidetextorientation='radial')
fig.update_layout(title="Dataset Contributors",title_x=0.5)
fig.show()

df=data['cp'].value_counts().reset_index().rename(columns={'index':'cp','cp':'co
unt'})
fig = go.Figure([go.Pie(labels=df['cp'],values=df['count'], hole = 0.5)])
fig.update_traces(hoverinfo='label+percent', textinfo='value+percent',
```

```
textfont_size=15,insidetextorientation='radial')
fig.update_layout(title="Chest Pain Conditions",title_x=0.5)
fig.show()

fig = go.Figure()
fig.add_trace(go.Box(y=data['trestbps'].values , name='BP at Rest for all',
marker_color = 'green',boxmean=True))
fig.add_trace(go.Box(y=data[data['sex']=='Male']['trestbps'].values, name ='Male
only', marker_color = 'blue', boxmean = True))
fig.add_trace(go.Box(y=data[data['sex']=='Female']['trestbps'].values, name
='Female only', marker_color = 'red', boxmean = True))
fig.update_layout(title = 'BP Distribution', yaxis_title = 'Blood Pressure
(mm/Hg)', title_x = 0.5)
fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.show()

fig = go.Figure()
fig.add_trace(go.Box(y=data['trestbps'].values , name='BP at Rest for all',
marker_color = 'green',boxmean=True))
fig.add_trace(go.Box(y=data[data['num']== 0]['trestbps'].values, name ='No
Disease', marker_color = 'blue', boxmean = True))
fig.add_trace(go.Box(y=data[data['num'] !=0]['trestbps'].values, name ='Heart
Disease', marker_color = 'red', boxmean = True))
fig.update_layout(title = 'BP Distribution (at rest)', yaxis_title = 'Blood
Pressure (mm/Hg)', title_x = 0.5)
fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.show()


fig = go.Figure()
fig.add_trace(go.Violin(y=data['chol'].values , name='All Patient', marker_color
= 'green'))
fig.add_trace(go.Violin(y=data[data['num']== 0]['chol'].values, name ='No
Disease', marker_color = 'blue'))
fig.add_trace(go.Violin(y=data[data['num'] ==4]['chol'].values, name ='Heart
Disease', marker_color = 'red'))
fig.update_layout(title = 'Cholesterol Level Distribution', yaxis_title =
'Cholesterol Level', title_x = 0.5)
fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.show()

data.head(30)

# In some of the features, there is space will will create problem later on.
# So we rename those attributes to handle problems in the future.

# data["restecg"].replace({"lv hypertrophy": "lv_hypertrophy","st-t
abnormality": "stt_abnormality" }, inplace=True)
data['thal'].replace({'fixed defect':'fixed_defect' , 'reversable defect':
'reversable_defect' }, inplace =True)
data['cp'].replace({'typical angina':'typical_angina', 'atypical angina':
'atypical_angina' }, inplace =True)


data_tmp = data[['age','sex','cp', 'trestbps', 'chol', 'fbs',  'thalch',
'exang', 'oldpeak', 'slope', 'ca', 'thal']].copy()
data_tmp['target'] = ((data['num'] > 0)*1).copy()
data_tmp['sex'] = (data['sex'] == 'Male')*1
data_tmp['fbs'] = (data['fbs'])*1
data_tmp['exang'] = (data['exang'])*1
```

```
data_tmp.columns = ['age', 'sex', 'chest_pain_type', 'resting_blood_pressure',
              'cholesterol', 'fasting_blood_sugar',
              'max_heart_rate_achieved', 'exercise_induced_angina',
              'st_depression', 'st_slope_type', 'num_major_vessels',
              'thalassemia_type', 'target']
data_tmp.head(15)


data = pd.get_dummies(data_tmp, drop_first=False)
data.columns

from sklearn.model_selection import train_test_split
y = data['target']
X = data.drop('target', axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
print(f'Shape of X_train: {X_train.shape}')
print(f'Shape of y_train: {y_train.shape}')
print(f'Shape of X_test: {X_test.shape}')
print(f'Shape of y_test: {y_test.shape}')

X_train=(X_train-np.min(X_train))/(np.max(X_train)-np.min(X_train)).values
X_test=(X_test-np.min(X_test))/(np.max(X_test)-np.min(X_test)).values

X_test

from sklearn.linear_model import LogisticRegression
logre = LogisticRegression()
logre.fit(X_train,y_train)

y_pred = logre.predict(X_test)
actual = []
predcition = []
for i,j in zip(y_test,y_pred):
    actual.append(i)
    predcition.append(j)

dic = {'Actual':actual,
       'Prediction':predcition }

result  = pd.DataFrame(dic)
import plotly.graph_objects as go

fig = go.Figure()
fig.add_trace(go.Scatter(x=np.arange(0,len(y_test)), y=y_test,
mode='markers',  name='Test'))
fig.add_trace(go.Scatter(x=np.arange(0,len(y_test)), y=y_pred,  mode='markers',
name='Pred'))

from sklearn.metrics import accuracy_score
print('The Accuracy Score is: ', accuracy_score(y_test,y_pred))


from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True)

import sklearn
print('Area Under ROC-Curve: ', sklearn.metrics.roc_auc_score(y_test,y_pred))
```

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred, drop_intermediate = False)
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)


print(logre.intercept_)
plt.figure(figsize=(10,12))
coeffecients = pd.DataFrame(logre.coef_.ravel(),X.columns)
coeffecients.columns = ['Coeffecient']
coeffecients.sort_values(by=['Coeffecient'],inplace=True,ascending=False)
coeffecients
```