DSC 495 Final Project

Anagh Shrivastava

Student ID: 200479396

In [1]:
```
# Importing Libraries for calculating numerical summaries, plotting graphs and using statistical analys
is
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Reading the file for the dataset
df = pd.read_csv('heart_disease_uci_og.csv')
```

In [3]:
```
# Displaying the top 5 entries of the dataset
df.head(5)
```

Out[3]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversable defect	1
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0

In [4]:
```
# Displaying the last 5 entries of the dataset
df.tail()
```

Out[4]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
915	916	54	Female	VA Long Beach	asymptomatic	127.0	333.0	True	st-t abnormality	154.0	False	0.0	NaN	NaN	NaN	1
916	917	62	Male	VA Long Beach	typical angina	NaN	139.0	False	st-t abnormality	NaN	NaN	NaN	NaN	NaN	NaN	0
917	918	55	Male	VA Long Beach	asymptomatic	122.0	223.0	True	st-t abnormality	100.0	False	0.0	NaN	NaN	fixed defect	2
918	919	58	Male	VA Long Beach	asymptomatic	NaN	385.0	True	lv hypertrophy	NaN	NaN	NaN	NaN	NaN	NaN	0
919	920	62	Male	VA Long Beach	atypical angina	120.0	254.0	False	lv hypertrophy	93.0	True	0.0	NaN	NaN	NaN	1

In [5]:
```
# Displaying the shape of the dataset
df.shape
```

Out[5]:
```
(920, 16)
```

In [6]:
```
# Dispalying the information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        920 non-null    int64
 1   age       920 non-null    int64
 2   sex       920 non-null    object
 3   dataset   920 non-null    object
 4   cp        920 non-null    object
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object
 8   restecg   918 non-null    object
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object
 13  ca        309 non-null    float64
 14  thal      434 non-null    object
 15  num       920 non-null    int64
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

In [7]:
```
# Counting the different types of cp
df['cp'].value_counts()
```

Out[7]:
```
asymptomatic       496
non-anginal        204
atypical angina    174
typical angina      46
Name: cp, dtype: int64
```

In [8]:
```
# Dropping the 'ca' column from the dataset
df = df.drop(columns='ca', axis=1)
```

In [9]:
```
# Dispalying the information of the new dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 15 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        920 non-null    int64
 1   age       920 non-null    int64
 2   sex       920 non-null    object
 3   dataset   920 non-null    object
 4   cp        920 non-null    object
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object
 8   restecg   918 non-null    object
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object
 13  thal      434 non-null    object
 14  num       920 non-null    int64
dtypes: float64(4), int64(3), object(8)
memory usage: 107.9+ KB
```

In [10]:
```
# Checking if there are any null values in dataset
df.isna().sum()
```

Out[10]:
```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
thal        486
num           0
dtype: int64
```

In [11]:
```
# Displaying the shape of the new dataset
df.shape
```

Out[11]:
```
(920, 15)
```

In [12]:
```
# Column Name change
df.columns = ['ID', 'Age', 'Sex', 'Location_of_Study', 'Type_of_Chestpain', 'BP_resting', 'cholestrol_l
evel', 'Bloodsugar_Fasting', 'ECG_resting', 'Heartrate_Maximum', 'Angina_exercise', 'depression', 'slop
e', 'thal','target']
```

In [13]:
```
# Dropping the null values
df.dropna(inplace=True)
```

In [14]:
```
# Displaying the sum of the null values of columns
df.isnull().sum()
```

Out[14]:
```
ID                    0
Age                   0
Sex                   0
Location_of_Study     0
Type_of_Chestpain     0
BP_resting            0
cholestrol_level      0
Bloodsugar_Fasting    0
ECG_resting           0
Heartrate_Maximum     0
Angina_exercise       0
depression            0
slope                 0
thal                  0
target                0
dtype: int64
```

In [15]:
```
# Shape of data
df.shape
```

Out[15]:
```
(371, 15)
```

In [16]:
```
# Discription of the data
df.describe()
```

Out[16]:

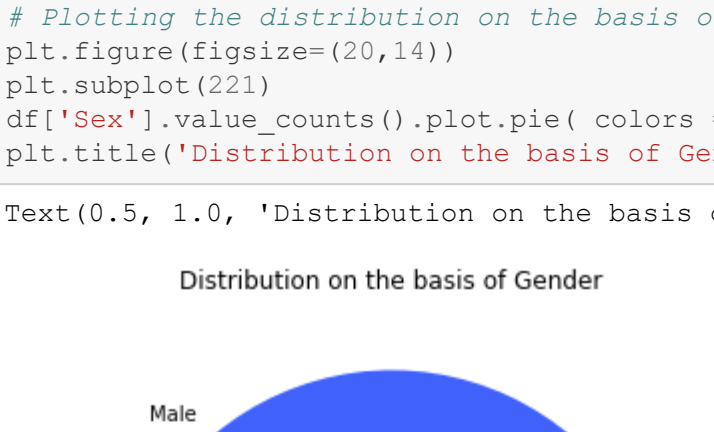
	ID	Age	BP_resting	cholestrol_level	Heartrate_Maximum	depression	target
count	371.00000	371.000000	371.000000	371.000000	371.000000	371.000000	371.000000
mean	251.09434	54.757412	132.137466	215.469003	143.711590	1.013747	1.083558
std	225.80125	9.037186	17.930797	97.011484	25.961934	1.118179	1.241631
min	1.00000	29.000000	94.000000	0.000000	60.000000	-1.000000	0.000000
25%	94.50000	48.000000	120.000000	197.000000	125.000000	0.000000	0.000000
50%	187.00000	56.000000	130.000000	233.000000	147.000000	0.800000	1.000000
75%	280.50000	61.000000	140.000000	270.500000	163.000000	1.600000	2.000000
max	904.00000	77.000000	200.000000	564.000000	202.000000	6.200000	4.000000

In [17]:
```
# Displaying the Series containing counts of unique rows in the 'target' column
df['target'].value_counts()
```

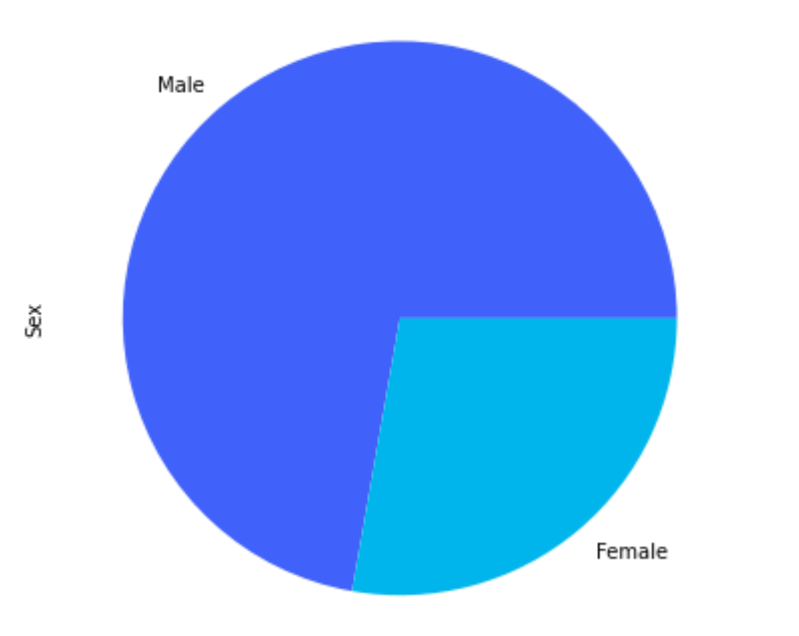
Out[17]:
```
0    171
1     82
3     52
2     50
4     16
Name: target, dtype: int64
```

In [18]:
```
# Plotting the distribution on the basis of Gender of Patients
plt.figure(figsize=(20,14))
plt.subplot(221)
df['Sex'].value_counts().plot.pie( colors = sns.color_palette('rainbow',7), labels =['Male','Female'])
plt.title('Distribution on the basis of Gender')
```

Out[18]:
```
Text(0.5, 1.0, 'Distribution on the basis of Gender')
```

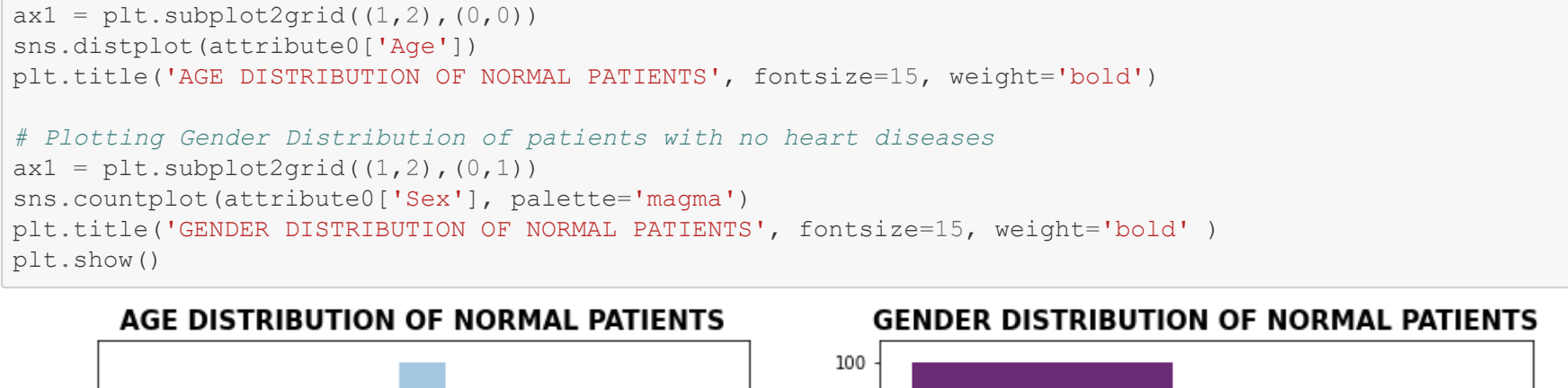


In [19]:
```
# Plotting the distribution on the basis of Age of Patients
ax = sns.distplot(df['Age'])
plt.title('Distribution on the basis of Age')
plt.show()
```

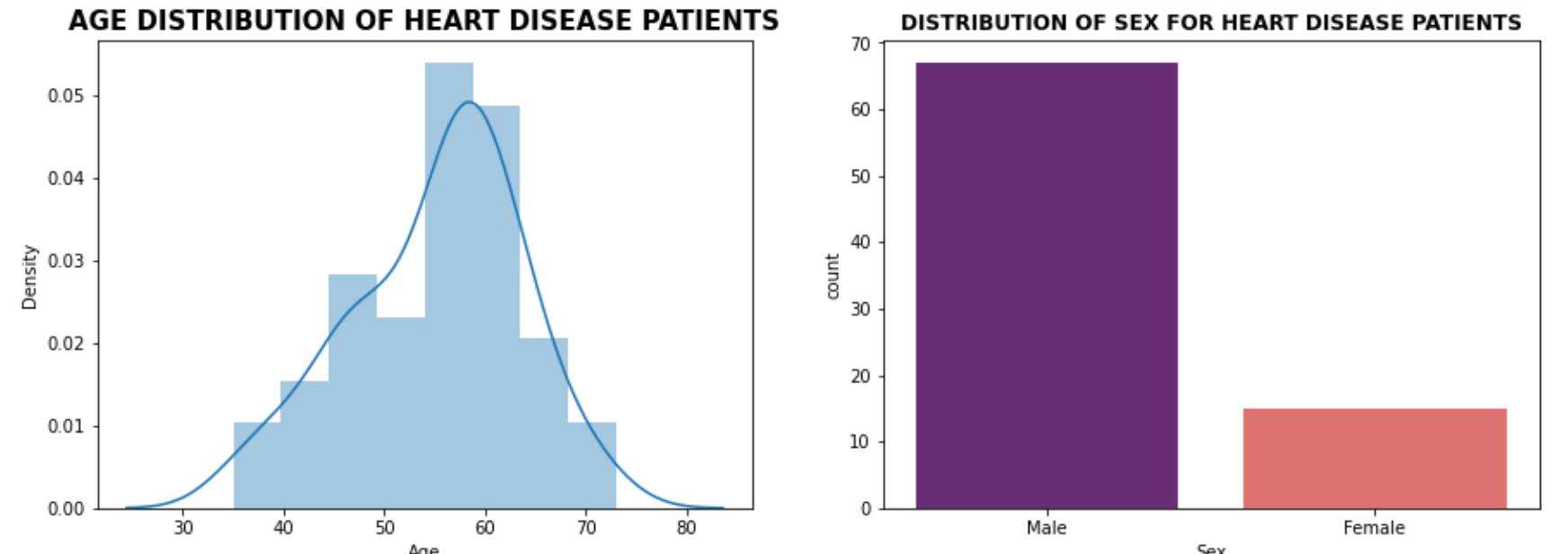


In [20]:
```
attribute0  = df[df['target'] == 0] # Assigning names to patients with no heart diseases
attribute1  = df[df['target'] == 1] # Assigning names to patients with heart disease stage 1
attribute2  = df[df['target'] == 2] # Assigning names to patients with heart disease stage 2
attribute3  = df[df['target'] == 3] # Assigning names to patients with heart disease stage 3
attribute4  = df[df['target'] == 4] # Assigning names to patients with heart disease stage 4
```

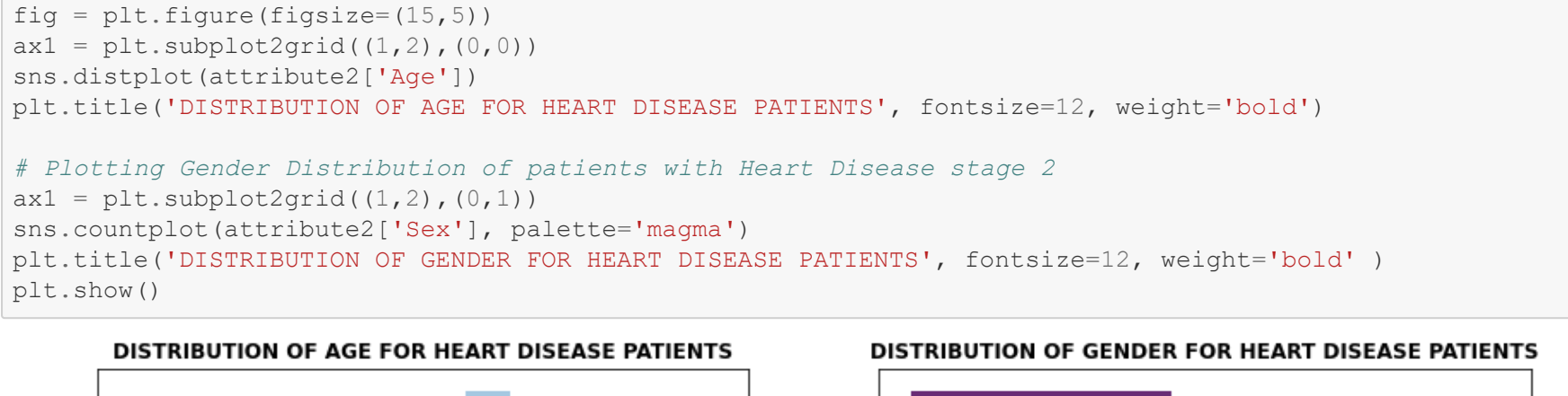
In [21]:
```
# Plotting Distribution of age of patients with no heart diseases
fig = plt.figure(figsize=(15,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.distplot(attribute0['Age'])
plt.title('AGE DISTRIBUTION OF NORMAL PATIENTS', fontsize=15, weight='bold')

# Plotting Gender Distribution of patients with no heart diseases
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attribute0['Sex'], palette='magma')
plt.title('GENDER DISTRIBUTION OF NORMAL PATIENTS', fontsize=15, weight='bold' )
plt.show()
```



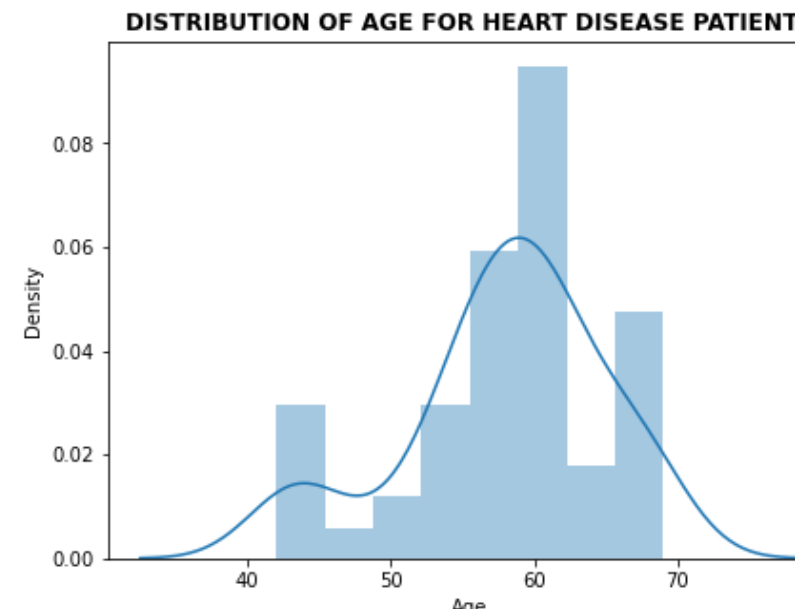
In [22]:
```
# Plotting Distribution of age of patients with Heart Disease stage 1
fig = plt.figure(figsize=(15,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.distplot(attribute1['Age'])
plt.title('AGE DISTRIBUTION OF HEART DISEASE PATIENTS', fontsize=15, weight='bold')

# Plotting Gender Distribution of patients with Heart Disease stage 1
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attribute1['Sex'], palette='magma')
plt.title('DISTRIBUTION OF SEX FOR HEART DISEASE PATIENTS', fontsize=12, weight='bold' )
plt.show()
```



In [23]:
```
# Plotting Distribution of Age of patients with Heart Disease stage 2
fig = plt.figure(figsize=(15,5))
ax1 = plt.subplot2grid((1,2),(0,0))
sns.distplot(attribute2['Age'])
plt.title('DISTRIBUTION OF AGE FOR HEART DISEASE PATIENTS', fontsize=12, weight='bold')

# Plotting Gender Distribution of patients with Heart Disease stage 2
ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attribute2['Sex'], palette='magma')
plt.title('DISTRIBUTION OF GENDER FOR HEART DISEASE PATIENTS', fontsize=12, weight='bold' )
plt.show()
```



In [24]:
```
# Plotting Distribution of different types of chest pains for heaert patients
# ax1 = plt.subplot2grid((1,2),(0,1))
sns.countplot(attribute1['Type_of_Chestpain'], palette='magma')
plt.title('CHEST PAIN OF HEART PATIENTS', fontsize=15, weight='bold' )
plt.show()
```



In [25]:
```
# Plotting different types of ECG Resting Rate with respect to Patients with no heart diseases and diff
erent stages of heart diseases:
plot_criteria= ['ECG_resting', 'target']
cm = sns.light_palette("green", as_cmap=True)
(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') * 100,2)).style.bac
kground_gradient(cmap = cm)
```

Out[25]:

target	0	1	2	3	4
ECG_resting					
lv hypertrophy	39.770000	42.680000	36.000000	44.230000	62.500000
normal	59.060000	53.660000	48.000000	44.230000	18.750000
st-t abnormality	1.170000	3.660000	16.000000	11.540000	18.750000

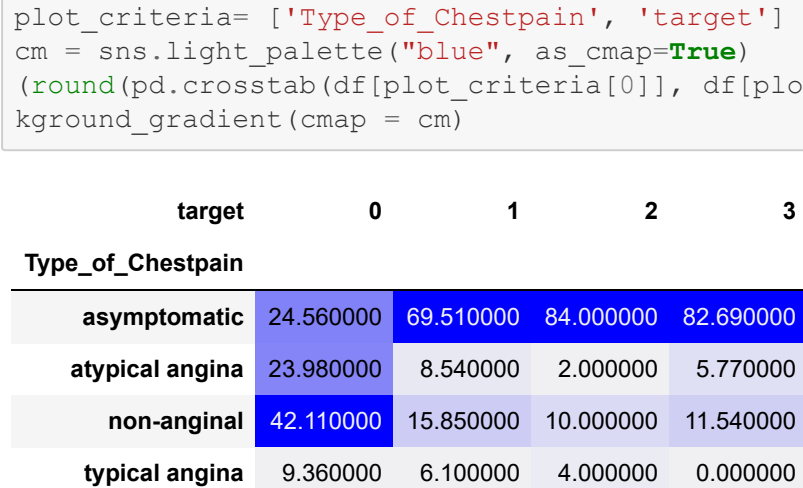
In [26]:
```
# Plotting different types of Chest Pain with respect to Patients with no heart diseases and different
 stages of heart diseases:
plot_criteria= ['Type_of_Chestpain', 'target']
cm = sns.light_palette("blue", as_cmap=True)
(round(pd.crosstab(df[plot_criteria[0]], df[plot_criteria[1]], normalize='columns') * 100,2)).style.bac
kground_gradient(cmap = cm)
```

Out[26]:

target	0	1	2	3	4
Type_of_Chestpain					
asymptomatic	24.560000	69.510000	84.000000	82.690000	87.500000
atypical angina	23.980000	8.540000	2.000000	5.770000	0.000000
non-anginal	42.110000	15.850000	10.000000	11.540000	6.250000
typical angina	9.360000	6.100000	4.000000	0.000000	6.250000

In [27]:
```
# Plotting a scatterplot of Resting Blood Pressure with respect to the Cholestrol level of Patients
sns.scatterplot(x = 'BP_resting', y = 'cholestrol_level', hue = 'target', data = df)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5281a51f0>
```



In [28]:
```
# One Hot Encoding the categorical columns
df = pd.get_dummies(df, drop_first=False)
df.columns
```

Out[28]:
```
Index(['ID', 'Age', 'BP_resting', 'cholestrol_level', 'Heartrate_Maximum',
       'depression', 'target', 'Sex_Female', 'Sex_Male',
       'Location_of_Study_Cleveland', 'Location_of_Study_Hungary',
       'Location_of_Study_Switzerland', 'Location_of_Study_VA Long Beach',
       'Type_of_Chestpain_asymptomatic', 'Type_of_Chestpain_atypical angina',
       'Type_of_Chestpain_non-anginal', 'Type_of_Chestpain_typical angina',
       'Bloodsugar_Fasting_False', 'Bloodsugar_Fasting_True',
       'ECG_resting_lv hypertrophy', 'ECG_resting_normal',
       'ECG_resting_st-t abnormality', 'Angina_exercise_False',
       'Angina_exercise_True', 'slope_downsloping', 'slope_flat',
       'slope_upsloping', 'thal_fixed defect', 'thal_normal',
       'thal_reversable defect'],
      dtype='object')
```

In [29]:
```
X = df.drop(['target'],axis=1) #input columns for the random forest model
y = df['target'] #target variable / heart disease predictor
```

In [30]:
```
#  Importing Train and Test libraries for the use of Modelling
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [31]:
```
# Splitting the dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2,shuffle=True, rando
m_state=42) # 20% of the entries will be used for testing the model
```

In [32]:
```
print('Distribution of target variable in training set')
print(y_train.value_counts()) # To view the distribution of target variable in the training set

print('Distribution of target variable in test set') # To view the distribution of target variable in t
he test set
print(y_test.value_counts())
```

```
Distribution of target variable in training set
0    136
1     65
3     42
2     40
4     13
Name: target, dtype: int64
Distribution of target variable in test set
0    35
1    17
2    10
3    10
4     3
Name: target, dtype: int64
```

In [33]:
```
#To view number of rows and columns in the Training Set
print('Training Set')
print(X_train.shape)
print(y_train.shape)

#To view number of rows and columns in the Test Set
print('Test Set')
print(X_test.shape)
print(y_test.shape)
```

```
Training Set
(296, 29)
(296,)
Test Set
(75, 29)
(75,)
```

In [34]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler() # Bringing all the column values to a similar scale to make the model more effi
cient
X_train[['Age','BP_resting','cholestrol_level','Heartrate_Maximum','depression']] = scaler.fit_transfor
m(X_train[['Age','BP_resting','cholestrol_level','Heartrate_Maximum','depression']])
X_train.head()
```

Out[34]:

	ID	Age	BP_resting	cholestrol_level	Heartrate_Maximum	depression	Sex_Female	Sex_Male	Location_of_Study_Cleveland
152	153	0.767442	0.214286	1.000000	0.740741	0.361111	1	0	1
291	292	0.488372	0.387755	0.606383	0.785185	0.305556	1	0	1
72	73	0.651163	0.265306	0.473404	0.288889	0.388889	0	1	1
231	232	0.488372	0.877551	0.579787	0.422222	0.611111	1	0	1
75	76	0.720930	0.673469	0.638298	0.674074	0.250000	1	0	1

5 rows × 29 columns

In [35]:
```
# Bringing all the column values to a similar scale to make the model more efficient
X_test[['Age','BP_resting','cholestrol_level','Heartrate_Maximum','depression']] = scaler.transform(X_t
est[['Age','BP_resting','cholestrol_level','Heartrate_Maximum','depression']])
X_test.head()
```

Out[35]:

	ID	Age	BP_resting	cholestrol_level	Heartrate_Maximum	depression	Sex_Female	Sex_Male	Location_of_Study_Cleveland
116	117	0.558140	0.469388	0.374113	0.777778	0.138889	0	1	1
126	127	0.511628	1.081633	0.510638	0.540741	0.694444	1	0	1
650	651	0.488372	0.265306	0.000000	0.237037	0.180556	0	1	0
701	702	0.697674	1.081633	0.000000	0.592593	0.277778	1	0	0
90	91	0.744186	0.265306	0.535461	0.674074	0.194444	0	1	1

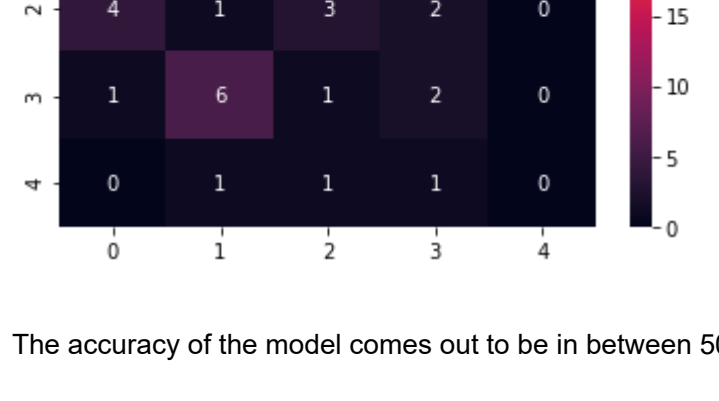
5 rows × 29 columns

In [36]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(criterion='entropy',n_estimators=100) # Using a random forest classifier
 as our model
model.fit(X_train, y_train) # Fitting the model on our training data
y_pred_rfe = model.predict(X_test) # Making predictions using the testing data
```

In [37]:
```
CM=confusion_matrix(y_test,y_pred_rfe) # Plotting the confusion matrix to see how accurate our model's
 predictions are
sns.heatmap(CM, annot=True)

acc= accuracy_score(y_test, y_pred_rfe)

model_results =pd.DataFrame([['Random Forest',acc]], columns = ['Model', 'Accuracy'])

model_results
```

Out[37]:

	Model	Accuracy
0	Random Forest	0.56



The accuracy of the model comes out to be in between 50 % to 58.6667 %. (The maximum accuracy achieved was 58.6667 %)