CIND820

November 9, 2021

[1]:

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns
import numpy as np
df=pd.read_csv("Heart_Disease.csv")#Load the data into the system
df.info() #determine the attribute and data type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      303 non-null    int64
 1   Sex                      303 non-null    int64
 2   Chest_Pain               303 non-null    int64
 3   Resting_Blood_Pressure   303 non-null    int64
 4   Colestrol                303 non-null    int64
 5   Fasting_Blood_Sugar      303 non-null    int64
 6   Rest_ECG                 303 non-null    int64
 7   MAX_Heart_Rate           303 non-null    int64
 8   Exercised_Induced_Angina 303 non-null    int64
 9   ST_Depression            303 non-null    float64
 10  Slope                    303 non-null    int64
 11  Major_Vessels            303 non-null    object
 12  Thalessemia              303 non-null    object
 13  Target                   303 non-null    int64
dtypes: float64(1), int64(11), object(2)
memory usage: 33.3+ KB
```

[2]:

```
df.head(10) #showing the first 10 rows
```

[2]:

```
    Age  Sex  Chest_Pain  Resting_Blood_Pressure  Colestrol  \
0    63    1           1                     145        233
1    67    1           4                     160        286
2    67    1           4                     120        229
3    37    1           3                     130        250
4    41    0           2                     130        204
```

```
5   56    1           2                     120        236
6   62    0           4                     140        268
7   57    0           4                     120        354
8   63    1           4                     130        254
9   53    1           4                     140        203

   Fasting_Blood_Sugar  Rest_ECG  MAX_Heart_Rate  Exercised_Induced_Angina  \
0                    1         2             150                         0
1                    0         2             108                         1
2                    0         2             129                         1
3                    0         0             187                         0
4                    0         2             172                         0
5                    0         0             178                         0
6                    0         2             160                         0
7                    0         0             163                         1
8                    0         2             147                         0
9                    1         2             155                         1

   ST_Depression  Slope Major_Vessels Thalessemia  Target
0            2.3      3             0           6       0
1            1.5      2             3           3       2
2            2.6      2             2           7       1
3            3.5      3             0           3       0
4            1.4      1             0           3       0
5            0.8      1             0           3       0
6            3.6      3             2           3       3
7            0.6      1             0           3       0
8            1.4      2             1           7       2
9            3.1      3             0           7       1
```

[3]:

```
#checking for missing value, replacing '?'to NAN
df.replace("?", np.nan, inplace = True)
df.isnull().sum()
```

[3]:

```
Age                         0
Sex                         0
Chest_Pain                  0
Resting_Blood_Pressure      0
Colestrol                   0
Fasting_Blood_Sugar         0
Rest_ECG                    0
MAX_Heart_Rate              0
Exercised_Induced_Angina    0
ST_Depression               0
Slope                       0
Major_Vessels               4
Thalessemia                 2
```

```
Target                     0
dtype: int64
```

[4]:

```
#drop the rows that contain missing values
df.dropna(inplace= True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 297 entries, 0 to 301
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       297 non-null    int64
 1   Sex                       297 non-null    int64
 2   Chest_Pain                297 non-null    int64
 3   Resting_Blood_Pressure    297 non-null    int64
 4   Colestrol                 297 non-null    int64
 5   Fasting_Blood_Sugar       297 non-null    int64
 6   Rest_ECG                  297 non-null    int64
 7   MAX_Heart_Rate            297 non-null    int64
 8   Exercised_Induced_Angina  297 non-null    int64
 9   ST_Depression             297 non-null    float64
 10  Slope                     297 non-null    int64
 11  Major_Vessels             297 non-null    object
 12  Thalessemia               297 non-null    object
 13  Target                    297 non-null    int64
dtypes: float64(1), int64(11), object(2)
memory usage: 34.8+ KB
```

[5]:

```
#change the target value to 0 and 1
df['Target'] = df.Target.apply(lambda q: '0' if q == 0 else '1')
print(df)
df['Target'].value_counts()
```

```
     Age  Sex  Chest_Pain  Resting_Blood_Pressure  Colestrol  \
0     63    1           1                     145        233
1     67    1           4                     160        286
2     67    1           4                     120        229
3     37    1           3                     130        250
4     41    0           2                     130        204
..   …   …         …                     …        …
297   57    0           4                     140        241
298   45    1           1                     110        264
299   68    1           4                     144        193
300   57    1           4                     130        131
301   57    0           2                     130        236

     Fasting_Blood_Sugar  Rest_ECG  MAX_Heart_Rate  Exercised_Induced_Angina  \
0                      1         2             150                         0
1                      0         2             108                         1
2                      0         2             129                         1
3                      0         0             187                         0
4                      0         2             172                         0
..                   ...     ...           ...                     ...
297                    0         0             123                         1
298                    0         0             132                         0
299                    1         0             141                         0
300                    0         0             115                         1
301                    0         2             174                         0

     ST_Depression  Slope Major_Vessels Thalessemia Target
0              2.3      3             0           6      0
1              1.5      2             3           3      1
2              2.6      2             2           7      1
3              3.5      3             0           3      0
4              1.4      1             0           3      0
..             ...  ...         ...       ...  ...
297            0.2      2             0           7      1
298            1.2      2             0           7      1
299            3.4      2             2           7      1
300            1.2      2             1           7      1
301            0.0      2             1           3      1

[297 rows x 14 columns]
```

[5]:

```
0    160
1    137
Name: Target, dtype: int64
```

[6]:

```
print(df.dtypes)
```

```
Age                           int64
Sex                           int64
Chest_Pain                    int64
Resting_Blood_Pressure        int64
Colestrol                     int64
Fasting_Blood_Sugar           int64
Rest_ECG                      int64
MAX_Heart_Rate                int64
Exercised_Induced_Angina      int64
ST_Depression               float64
Slope                         int64
Major_Vessels                object
Thalessemia                  object
Target                       object
dtype: object
```

[7]:

```
#changing the data types to interger
df = df.apply(pd.to_numeric)
```

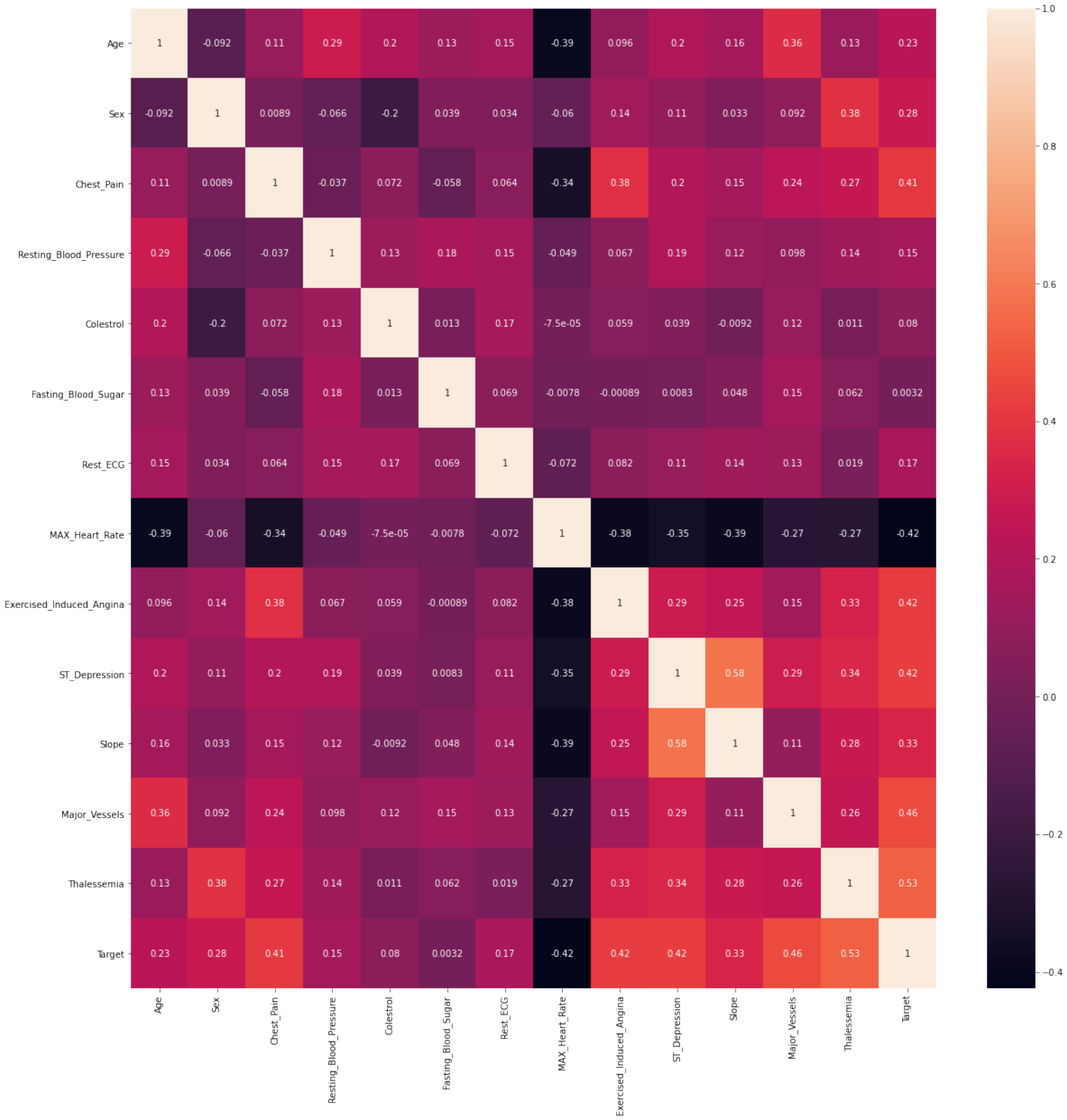
[8]:

```
print(df.dtypes)
```

```
Age                              int64
Sex                              int64
Chest_Pain                       int64
Resting_Blood_Pressure           int64
Colestrol                        int64
Fasting_Blood_Sugar              int64
Rest_ECG                         int64
MAX_Heart_Rate                   int64
Exercised_Induced_Angina         int64
ST_Depression                  float64
Slope                            int64
Major_Vessels                    int64
Thalessemia                      int64
Target                           int64
dtype: object
```

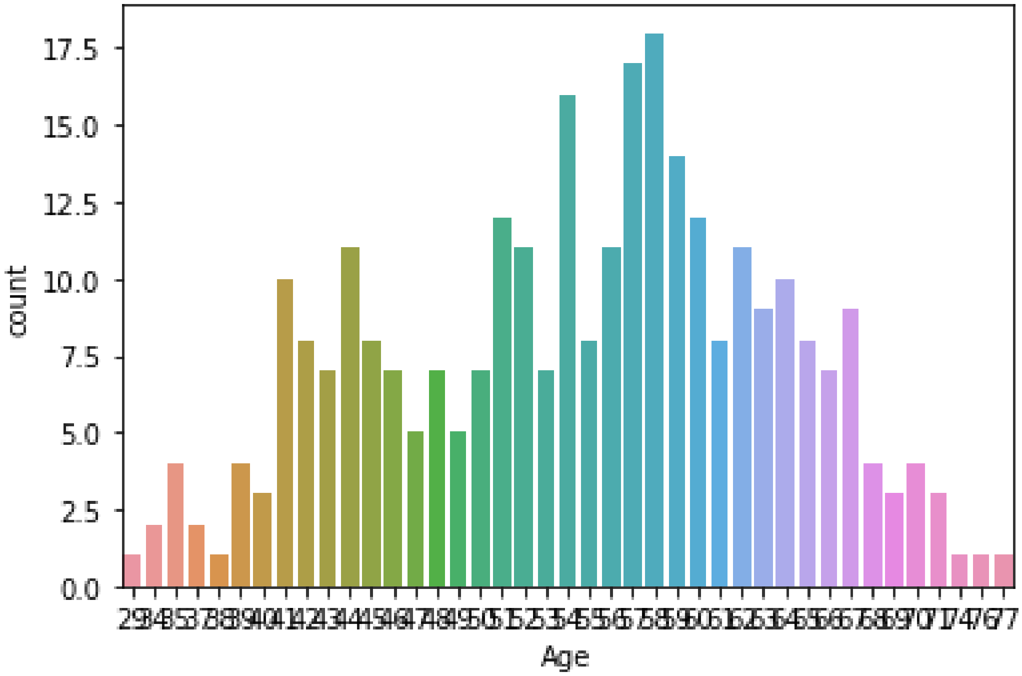
[9]:

```
#using heatmap to see the correlation between all the variable and pick the␣
 ↪variable that has the closer relationship with the targer value.
#using this method to pick the significant features to test our classificaiton␣
 ↪model
#base on the result 4 variable have been kicked out which are␣
 ↪"Resting_Blood_Pressure", "Colestrol", "MAX_Heart_Rate", and␣
 ↪"Fasting_Blood_Sugar".
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
g=sns.heatmap(df[top_corr_features].corr(),annot=True)
```



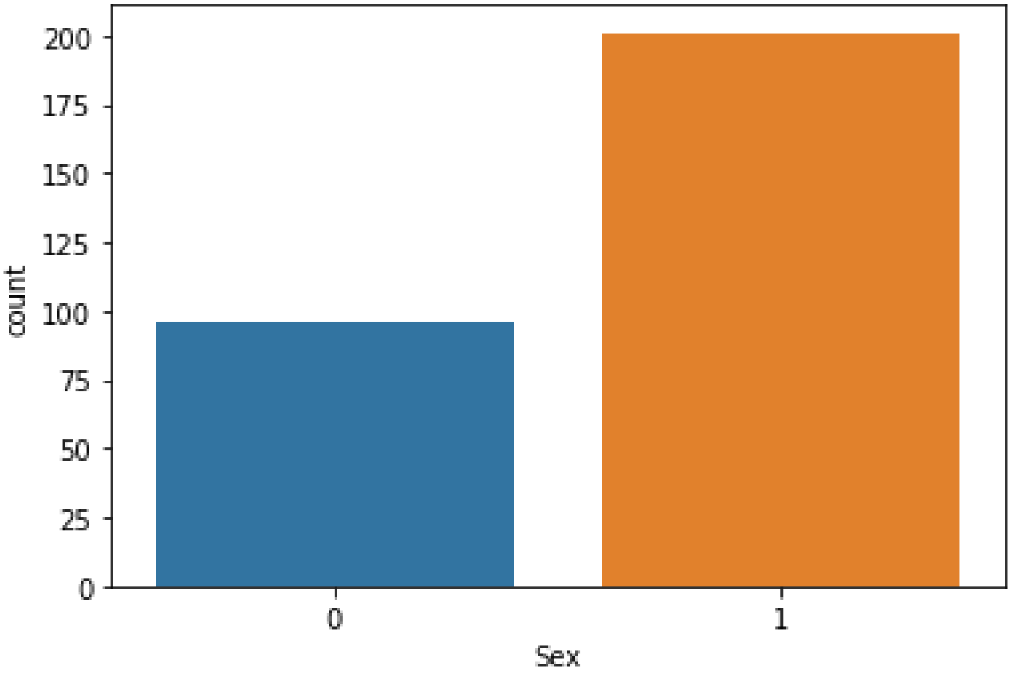
[10]:
```
#plot all the distribution of 9 variables againt target variable.
sns.countplot(x='Age', data=df)
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e434b4690>
```



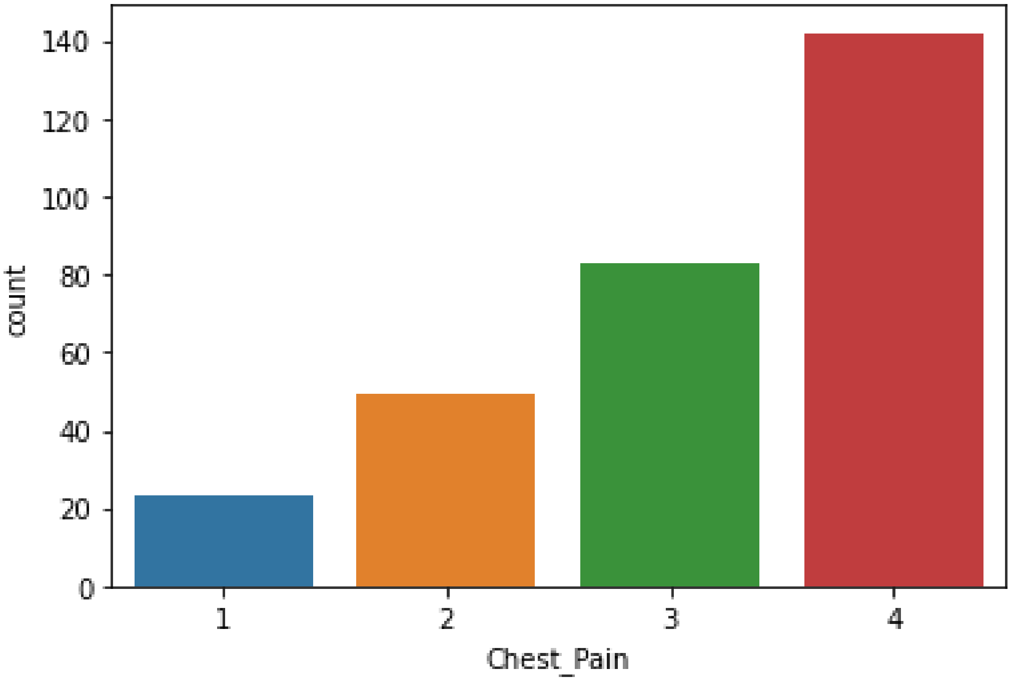
```
[11]: sns.countplot(x='Sex', data=df)
```

```
[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e43e464d0>
```



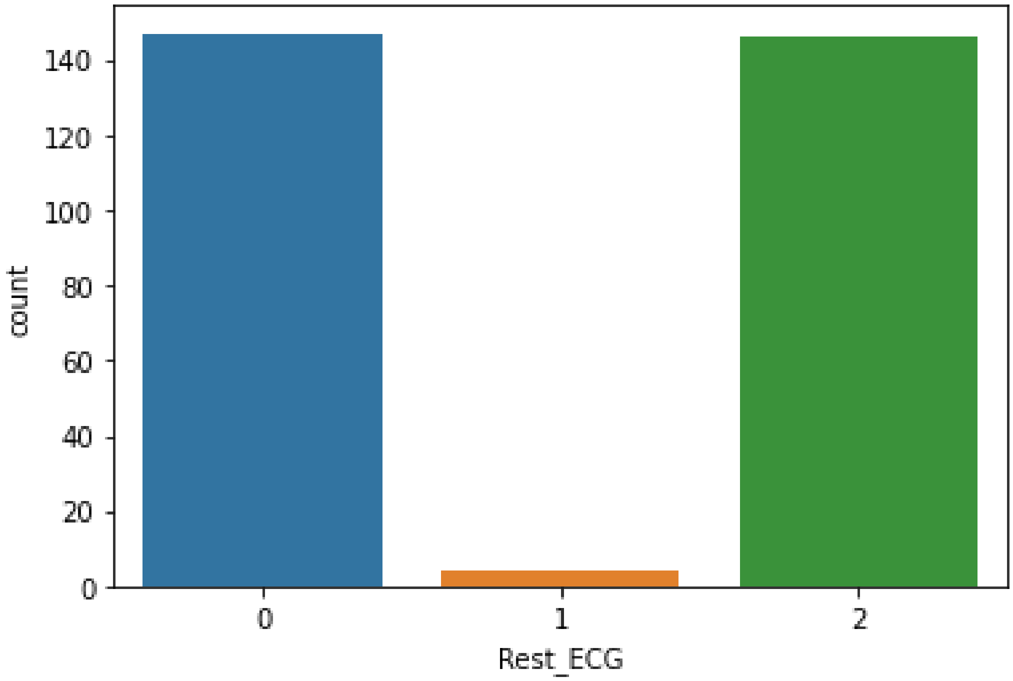
[12]:
```
sns.countplot(x='Chest_Pain', data=df)
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e43e15710>
```



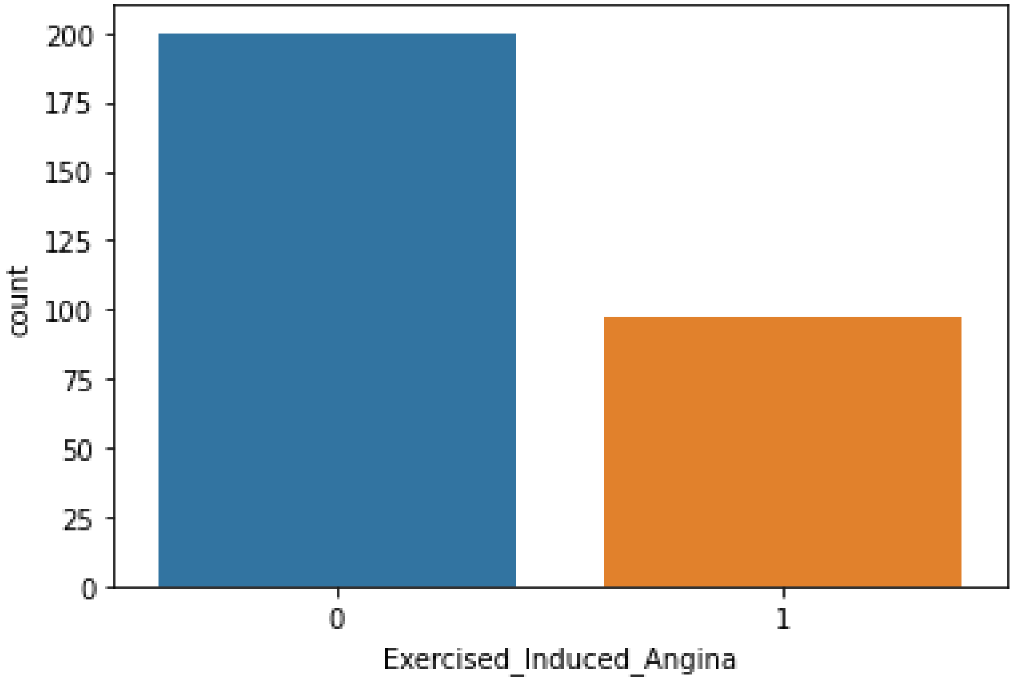
[13]:
```
sns.countplot(x='Rest_ECG', data=df)
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e43d8cc10>
```



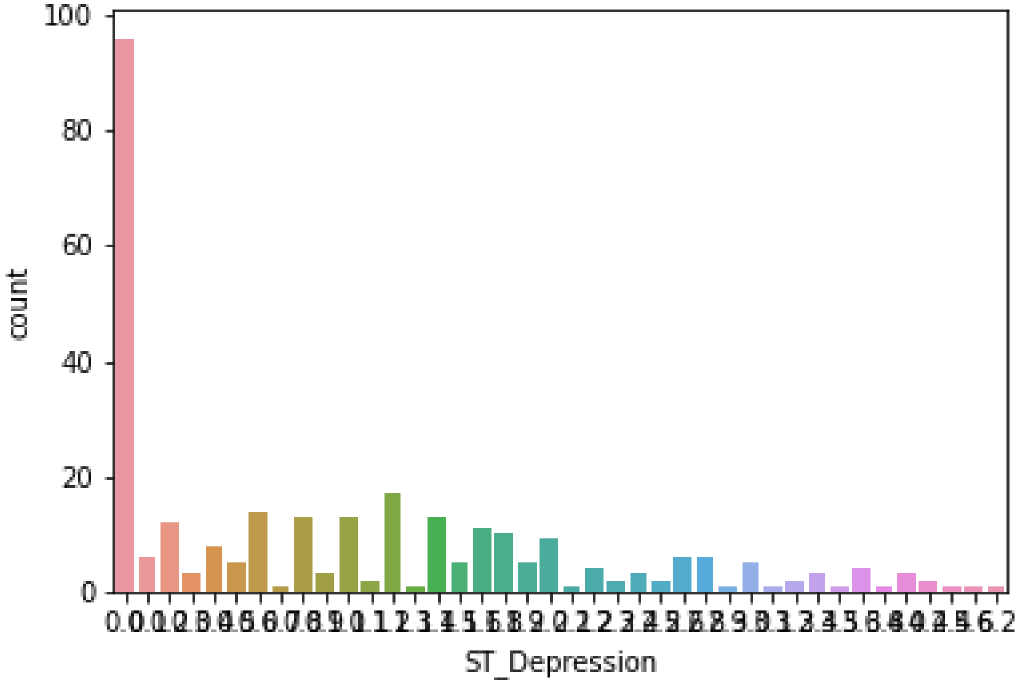
```
[14]: sns.countplot(x='Exercised_Induced_Angina', data=df)
```

```
[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e43ea1e10>
```



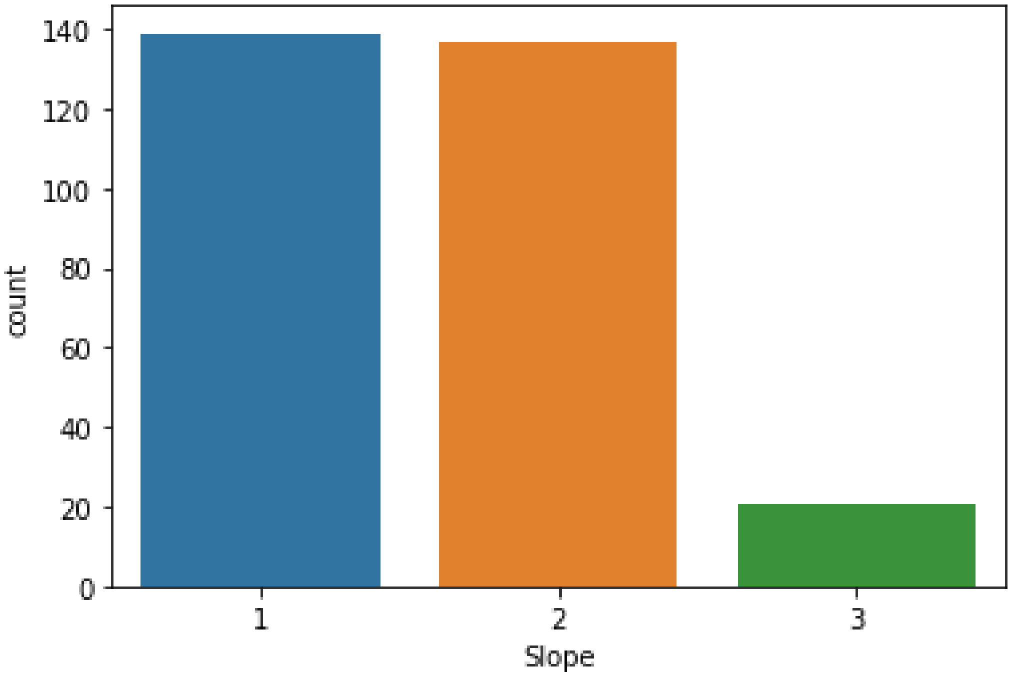
```
[15]: sns.countplot(x='ST_Depression', data=df)
```

```
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e43e1cf50>
```



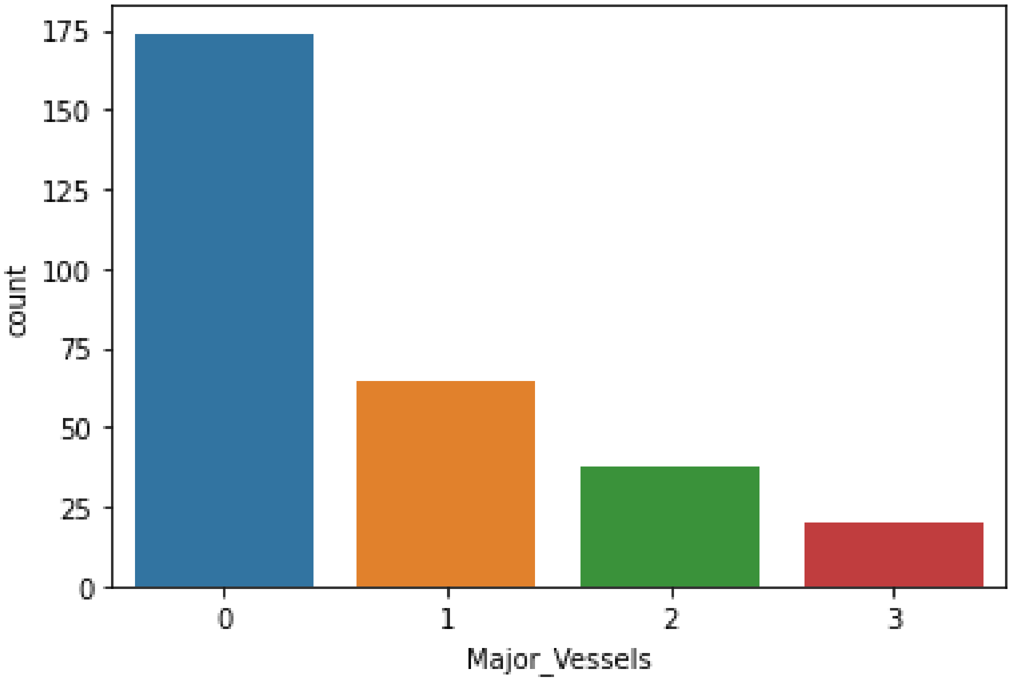
```
[16]: sns.countplot(x='Slope', data=df)
```

```
[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e43b65950>
```



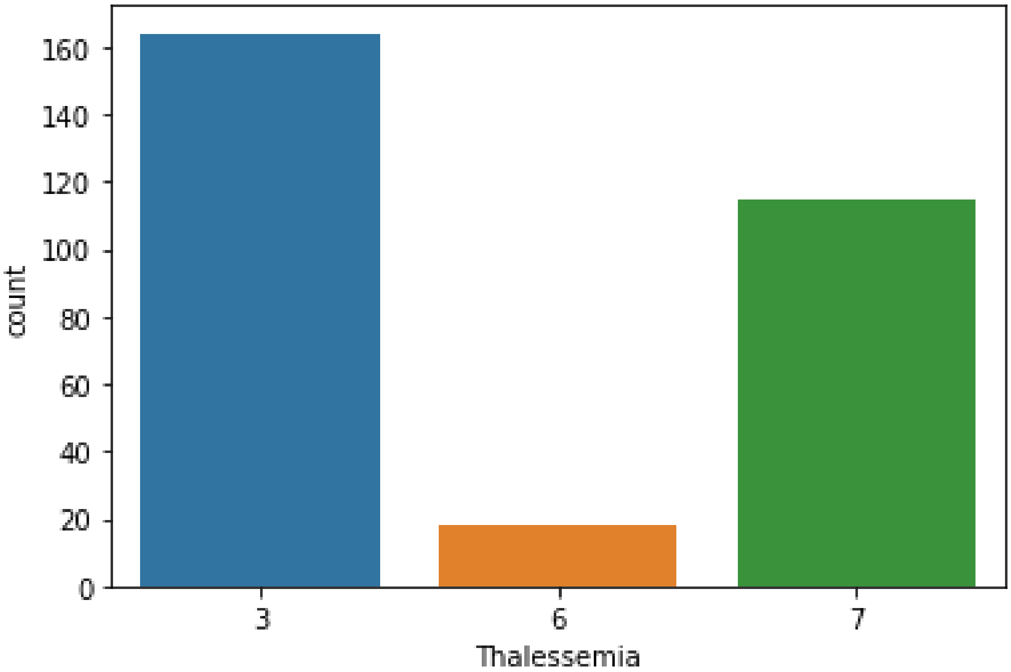
[17]:
```
sns.countplot(x='Major_Vessels', data=df)
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e43b41050>
```



```
[18]: sns.countplot(x='Thalessemia', data=df)
```

```
[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e43aada90>
```



```
[21]: import matplotlib.pyplot as plt
      import numpy as np
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import classification_report, confusion_matrix


      feature_cols = ['Age', 'Sex', 'Rest_ECG',␣
       ↪'Chest_Pain','Exercised_Induced_Angina','ST_Depression','Slope','Major_Vessels','Thalessemi
      X = df[feature_cols] # Features
      y = df.Target
```

```
[22]: #setting up training set and testing set with 0.25 size
      #this result in 74 vs 223
      from sklearn.model_selection import train_test_split
      X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
       ↪25,random_state=0)
```

```
[23]: #using logic regression algorithm to find the accuracy
      logreg = LogisticRegression()
```

```
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

[24]:

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

[24]:

```
array([[37,  2],
       [10, 26]])
```

[25]:

```
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: 0.84
Precision: 0.9285714285714286
Recall: 0.7222222222222222
```

[26]:

```
#using KNN algorithm model to find the accuracy

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=25)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
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: 0.8266666666666667
Precision: 0.896551724137931
Recall: 0.7222222222222222
```

[27]:

```
#using decision tree algorithm model to find the accuracy

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree␣
 ↪Classifier
```

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
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: 0.76
Precision: 0.75
Recall: 0.75
```

[28]:

```
#using naive bayes algorithm model to find the accuracy

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
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: 0.8533333333333334
Precision: 0.9310344827586207
Recall: 0.75
```