HEART DISEASE PREDICTION

1.EDA

In [1]:
```
import pandas as pd
r=pd.read_csv('C:\\Users\\YAKSHITHA DONTHI\\Downloads\\Heart_Disease_Data.txt')
r.to_csv('C:\\Users\\YAKSHITHA DONTHI\\Downloads\\finalcleve.csv',index=None)
```

In [12]:
```
%matplotlib inline
```

In [11]:
```
df = pd.read_csv('C:\\Users\\YAKSHITHA DONTHI\\Downloads\\finalcleve.csv', na_va
df.head()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop	ca	thal	pred_attrib
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	

In [8]:
```
df.shape
```

Out[8]:
```
(303, 14)
```

In [127]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 13 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slop        303 non-null int64
ca          303 non-null float64
thal        303 non-null float64
dtypes: float64(3), int64(10)
memory usage: 30.9 KB
```

In [4]:
```
df.describe()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	tha
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.990099	149.60
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.994971	22.87
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.00
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.50
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.000000	153.00
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.000000	166.00
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.00

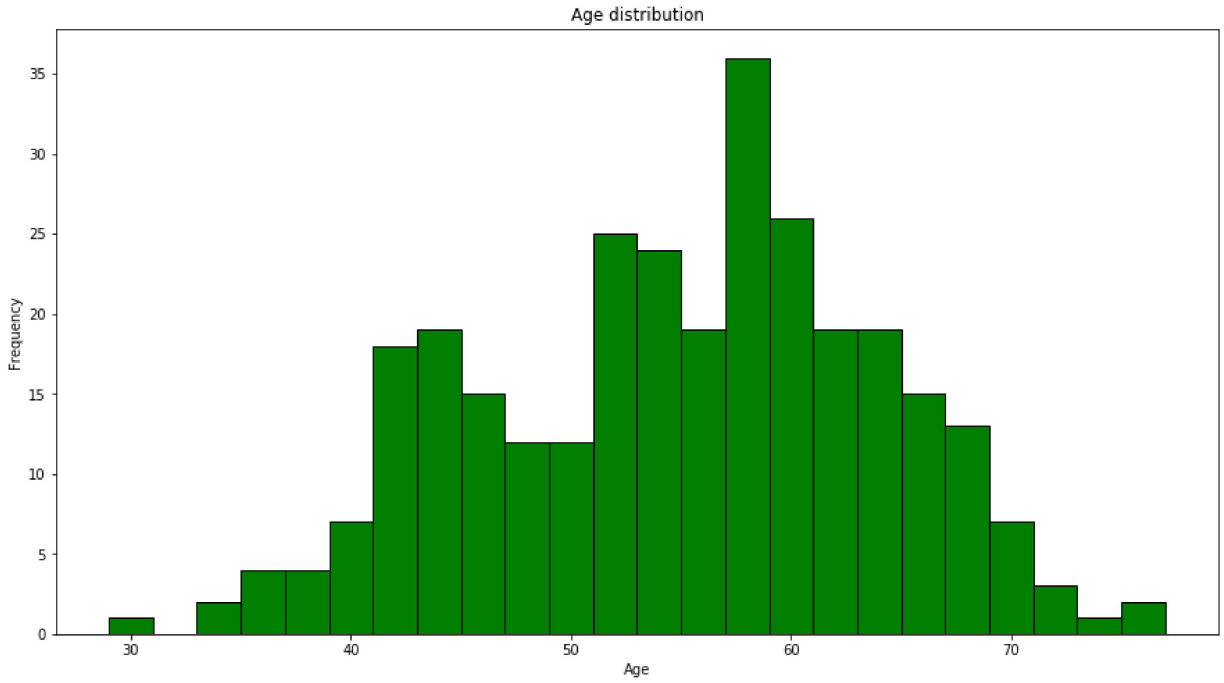
In [6]:
```
print("No of Nan values in our dataframe : ", sum(df.isnull().any()))
```

```
No of Nan values in our dataframe :  2
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
age              0
sex              0
cp               0
trestbps         0
chol             0
fbs              0
restecg          0
thalach          0
exang            0
oldpeak          0
slop             0
ca               4
thal             2
pred_attribute   0
dtype: int64
```

In [22]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,8))
plt.hist(df['age'], color='green', edgecolor='black', bins=24)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age distribution')
plt.show()
```

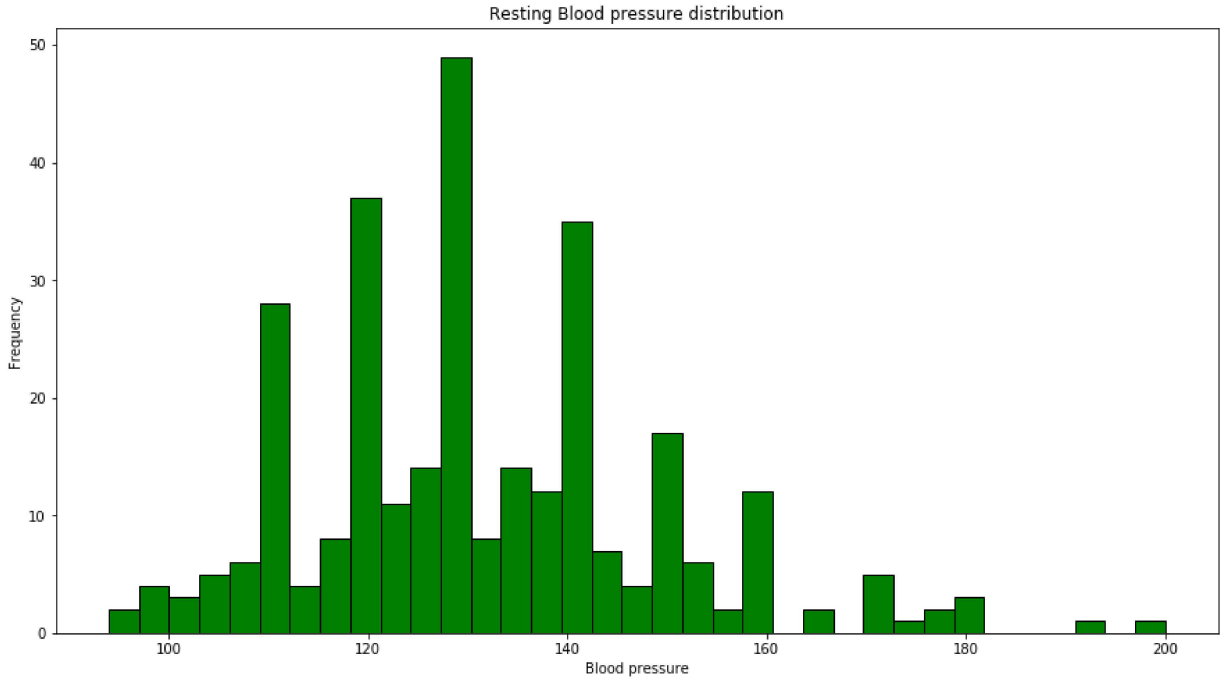


People of age 60 are more in the dataset

In [19]:
```
male = df['sex'].sum() # '1' indicates male and '0' indicates female
print('There are '+ str(male) + ' males and '+ str(df.shape[0] - male) + ' femal
```

```
There are 206 males and 97 females in the dataset
```

In [21]:
```
plt.figure(figsize=(15,8))
plt.hist(df['trestbps'], color='green', edgecolor='black', bins=int(106/3))
plt.xlabel('Blood pressure')
plt.ylabel('Frequency')
plt.title('Resting Blood pressure distribution')
plt.show()
```



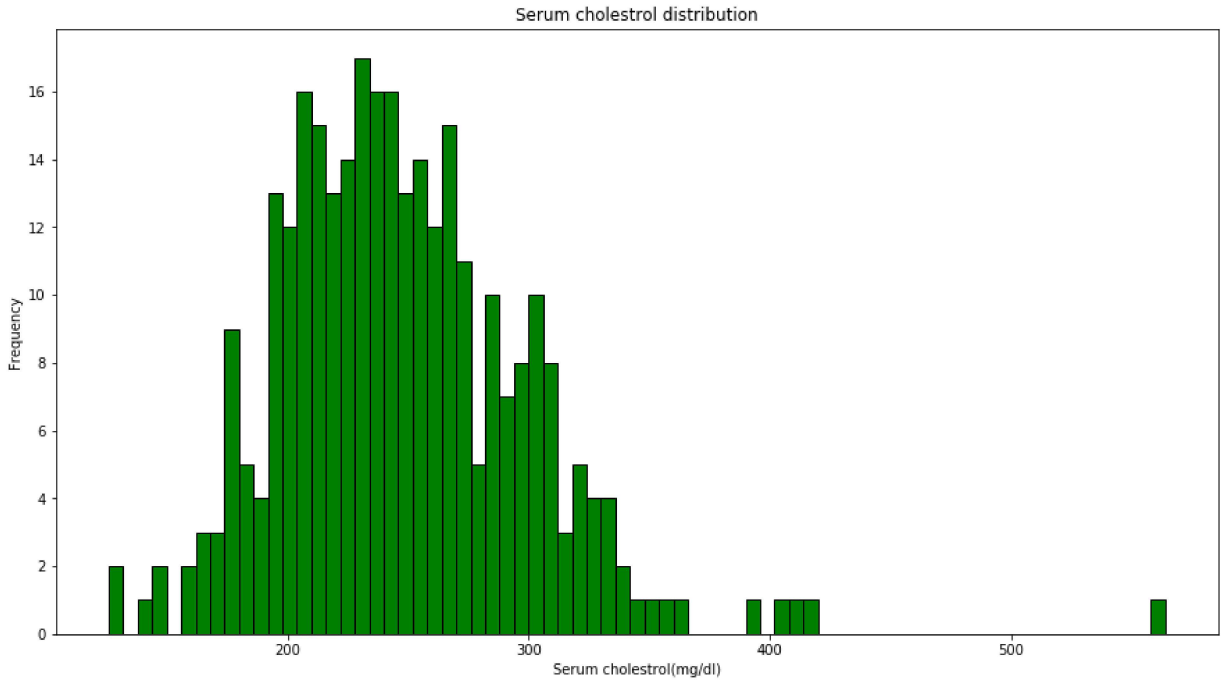
Blood pressure of 130 value is more in the dataset, next is 120 and so on

In [23]:
```
df.groupby('cp')['cp'].count()
```

Out[23]:
```
cp
1     23
2     50
3     86
4    144
Name: cp, dtype: int64
```

It says chest pain of type angina, abnang, notang, asympt are 23,50,86,144 rows respectively.

In [24]:
```
plt.figure(figsize=(15,8))
plt.hist(df['chol'], color='green', edgecolor='black', bins=73)
plt.xlabel('Serum cholestrol(mg/dl)')
plt.ylabel('Frequency')
plt.title('Serum cholestrol distribution')
plt.show()
```



Most of the cholestrol rates are between 200 and 300

In [27]:
```
df.groupby('restecg')['restecg'].count()
```

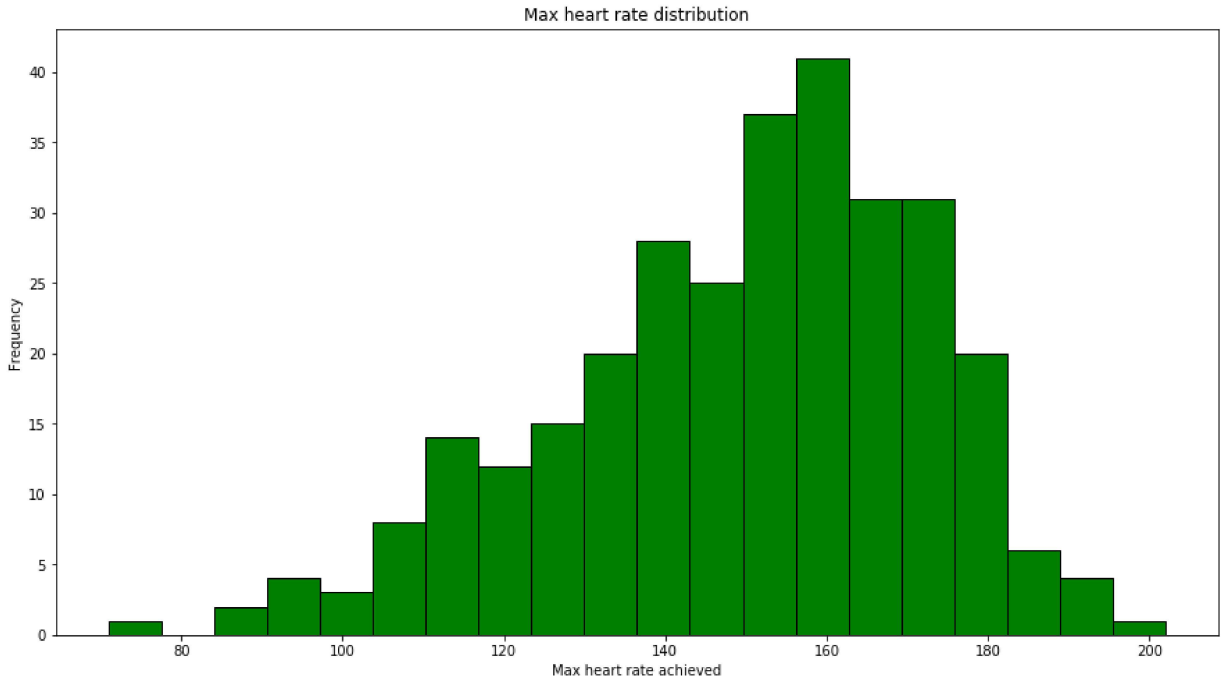
Out[27]:
```
restecg
0    151
1      4
2    148
Name: restecg, dtype: int64
```

It says most of rows resting ecg is either normal or hyper

In [26]:
```
diabetic = df['fbs'].sum() # '1' indicates yes and '0' indicates no about resting
print('There are '+ str(diabetic) + ' patients who had resting blood sugar > 120
```

```
There are 45 patients who had resting blood sugar > 120 mg/dl(diabetic) out of
the 303 patients
```

In [28]:
```
plt.figure(figsize=(15,8))
plt.hist(df['thalach'], color='green', edgecolor='black', bins=20)
plt.xlabel('Max heart rate achieved')
plt.ylabel('Frequency')
plt.title('Max heart rate distribution')
plt.show()
```

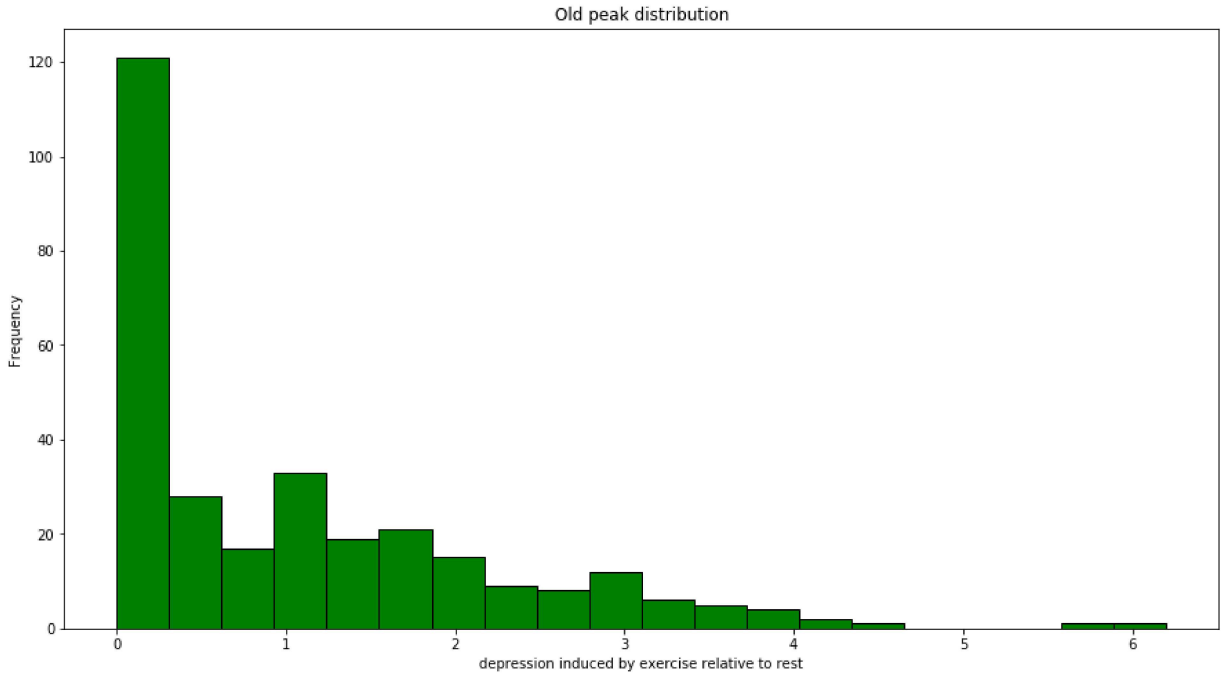


Max heart rate is between 140 to 180 which is permissible level

In [29]:
```
eig = df['exang'].sum() # '1' indicates yes and '0' indicates no about exercise
print('There are '+ str(eig) + ' patients who had exercise induced angina out of
```

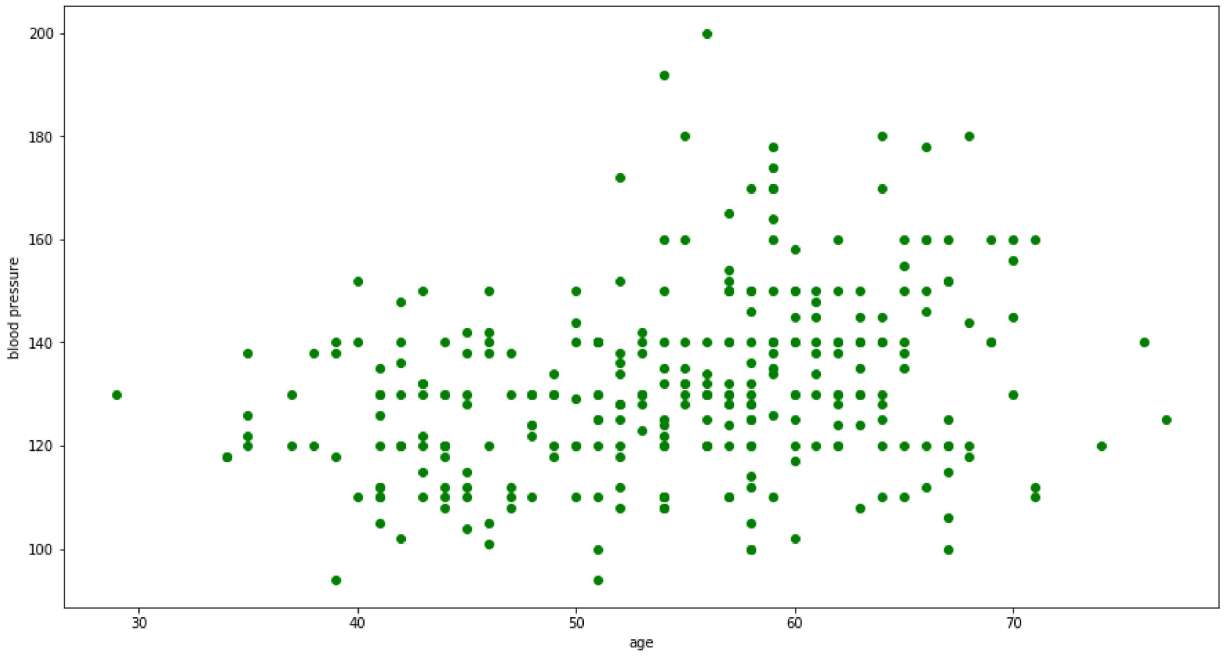
```
There are 99 patients who had exercise induced angina out of the 303 patients
```

In [30]:
```
plt.figure(figsize=(15,8))
plt.hist(df['oldpeak'], color='green', edgecolor='black', bins=20)
plt.xlabel('depression induced by exercise relative to rest')
plt.ylabel('Frequency')
plt.title('Old peak distribution')
plt.show()
```



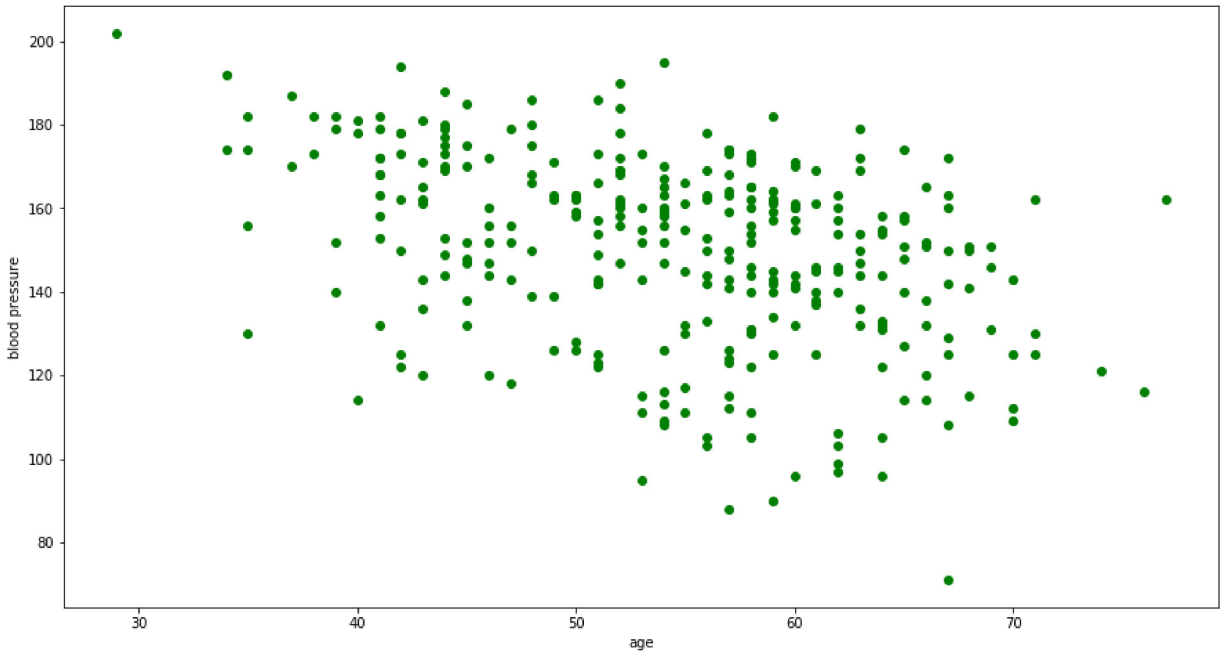
In [51]:
```
plt.figure(figsize=(15,8))
plt.scatter(df['age'],df['trestbps'],c='green')
plt.xlabel('age')
plt.ylabel('blood pressure')
```

Out[51]:
```
Text(0, 0.5, 'blood pressure')
```



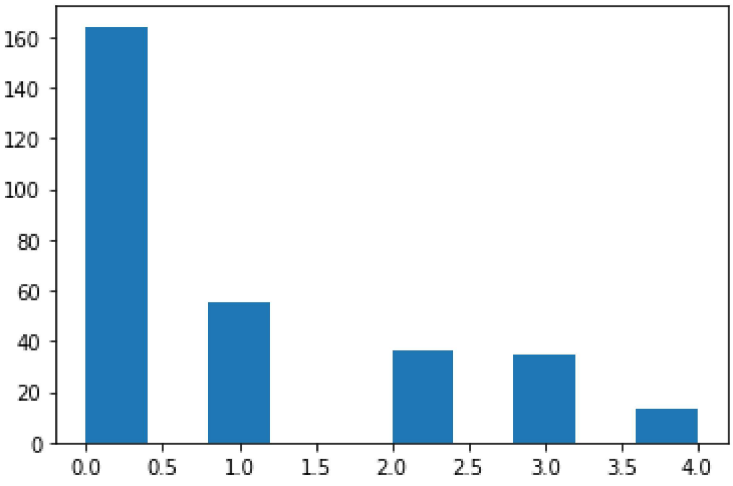
In [52]:
```
plt.figure(figsize=(15,8))
plt.scatter(df['age'],df['thalach'],c='green')
plt.xlabel('age')
plt.ylabel('blood pressure')
```

Out[52]: Text(0, 0.5, 'blood pressure')

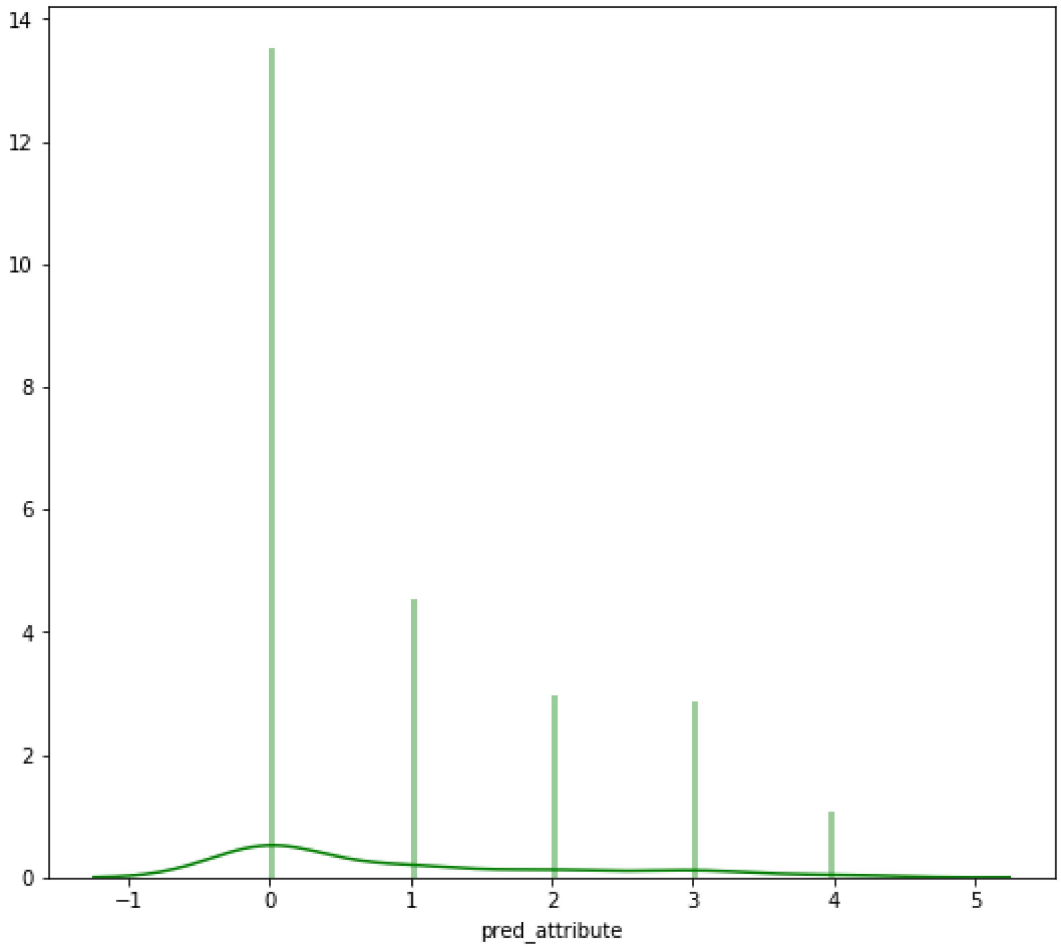


In [62]:
```
plt.hist(df['pred_attribute'])
```

Out[62]:
```
(array([164.,   0.,  55.,   0.,   0.,  36.,   0.,  35.,   0.,  13.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <a list of 10 Patch objects>)
```



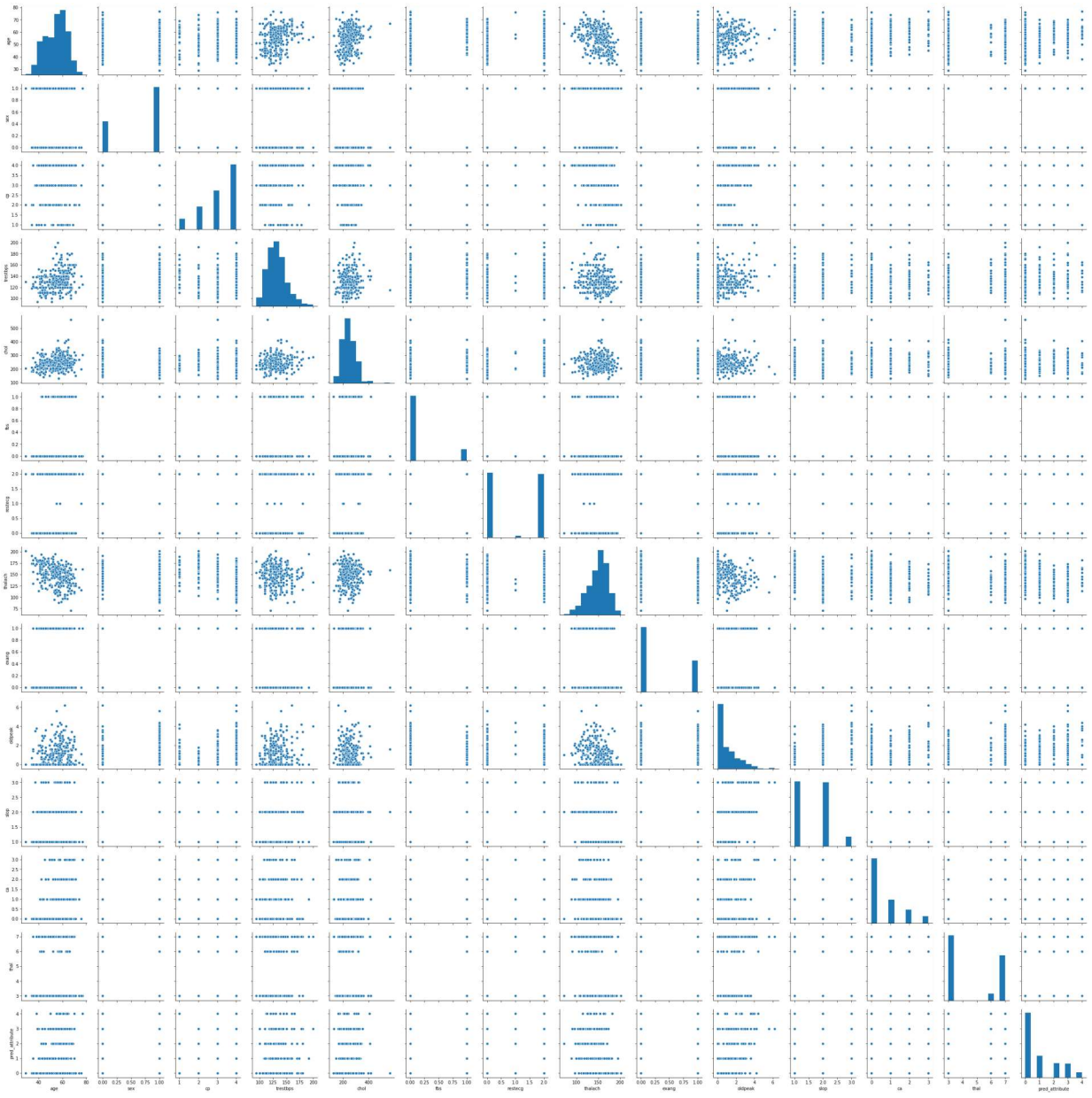
In [58]:
```
plt.figure(figsize=(9, 8))
sns.distplot(df['pred_attribute'], color='g', bins=100, hist_kws={'alpha': 0.4})
```



The target variable distribution, many people are healthy, next sick1, sick3, sick2 and sick4

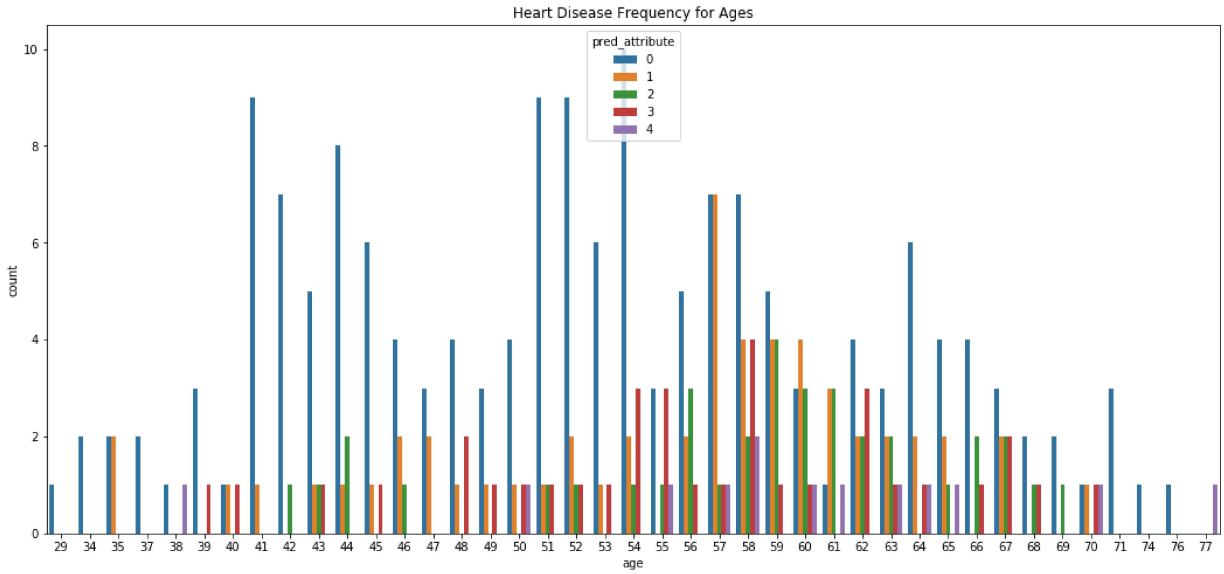
In [64]:
```
import seaborn as sns
sns.pairplot(df)
```

Out[64]:
```
<seaborn.axisgrid.PairGrid at 0x1e7b992a488>
```



In [54]:
```
import seaborn as sns
plt.figure(figsize=(18,8))
sns.countplot(x='age', data = df, hue="pred_attribute")
plt.title('Heart Disease Frequency for Ages')
```

Out[54]:
```
Text(0.5, 1.0, 'Heart Disease Frequency for Ages')
```



Preprocessing

In [31]:
```
df.groupby('slop')['slop'].count()
```

Out[31]:
```
slop
1    142
2    140
3     21
Name: slop, dtype: int64
```

In [32]:
```
df.groupby('ca')['ca'].count()
```

Out[32]:
```
ca
0.0    176
1.0     65
2.0     38
3.0     20
Name: ca, dtype: int64
```

There are 4 missing(null) values in ca and we fill it by mode

In [33]:
```
df['ca'] = df['ca'].fillna(df['ca'].mode()[0])
```

In [34]:
```
df.groupby('ca')['ca'].count()
```

Out[34]:
```
ca
0.0    180
1.0     65
2.0     38
3.0     20
Name: ca, dtype: int64
```

In [35]:
```
df.groupby('thal')['thal'].count()
```

Out[35]:
```
thal
3.0    166
6.0     18
7.0    117
Name: thal, dtype: int64
```

There are 2 missing(null) values in thal and we fill it by mode

In [36]:
```
df['thal'] = df['thal'].fillna(df['thal'].mode()[0])
```

In [37]:
```
df.groupby('thal')['thal'].count()
```

Out[37]:
```
thal
3.0    168
6.0     18
7.0    117
Name: thal, dtype: int64
```

In [38]:
```
df.isnull().sum()
```

Out[38]:
```
age              0
sex              0
cp               0
trestbps         0
chol             0
fbs              0
restecg          0
thalach          0
exang            0
oldpeak          0
slop             0
ca               0
thal             0
pred_attribute   0
dtype: int64
```

Now there are no null values in dataset

ML Model

In [73]:
```
from sklearn.model_selection import train_test_split
y = df['pred_attribute']
df.drop('pred_attribute', axis=1, inplace=True)
x = df
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get
_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

KeyError: 'pred_attribute'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-73-1b2d1697f633> in <module>
      1 from sklearn.model_selection import train_test_split
      2 from sklearn.naive_bayes import GaussianNB
----> 3 y = df['pred_attribute']
      4 df.drop('pred_attribute', axis=1, inplace=True)
      5 x = df

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem_
_(self, key)
   2978             if self.columns.nlevels > 1:
   2979                 return self._getitem_multilevel(key)
-> 2980             indexer = self.columns.get_loc(key)
   2981             if is_integer(indexer):
   2982                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get
_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(
key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=to
lerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

```
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

KeyError: 'pred_attribute'
```

first, target attribute is assigned to y and then that column is dropped from dataset and rest dataset is assigned to x

Train Test Split

In [81]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y)
```

Logisitic Regression model

In [82]:
```
cmodel = LogisticRegression(solver='lbfgs', multi_class='multinomial',C=1e-2,max_
cmodel.fit(x_train, y_train)
```

Out[82]:
```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=500,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=10, verbose=0,
                   warm_start=False)
```

In [88]:
```
y_predict = cmodel.predict_proba(x_test)
```

log loss of logistic regression

In [90]:
```
from sklearn.metrics import log_loss
log_loss(y_test,y_predict)
```

Out[90]:
```
1.1070402967610744
```

average of 10 log loss of logistic regression

In [109]:
```
t=0
avg=0
for i in range(0,10):
    x_train, x_test, y_train, y_test = train_test_split(x,y)
    cmodel.fit(x_train,y_train)
    pred = cmodel.predict_proba(x_test)
    t=log_loss(y_test,pred)
    #print(t)
    avg=avg+t
avg=avg/10
print(avg)
```

```
1.0677016746509707
```

Guassian NB Model

In [77]:
```
from sklearn.naive_bayes import  GaussianNB
```

In [112]:
```
model = GaussianNB()
model.fit(x_train,y_train)
```

Out[112]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [113]:
```
predictions = model.predict_proba(x_test)
```

log loss of Guassian NB

In [117]:
```
log_loss(y_test,predictions)
```

Out[117]:
```
1.981172611584002
```

average of 10 log loss of guassian nb

In [111]:
```
t=0
avg=0
for i in range(0,10):
    x_train, x_test, y_train, y_test = train_test_split(x,y)
    model.fit(x_train,y_train)
    pred = model.predict_proba(x_test)
    t=log_loss(y_test,pred)
    #print(t)
    avg=avg+t
avg=avg/10
print(avg)
```

```
2.299924903627974
```

we conclude logistic regression gives better accuracy

now we can use cmodel.predict("new row") to get the prediction

In [125]:
```
l=[63,1,1,145,233,1,2,150,0,2.3,3,0,6]
cmodel.predict([l])
```

Out[125]:
```
array([0], dtype=int64)
```

It correctly predicts as row belong to class 0 (Healthy) And it's done!!!