1. Load the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset=pd.read_csv('Heart_Disease_Data.txt',na_values='?')
dataset.to_csv('cleve.csv',index=None)
```

```
ds=pd.read_csv('cleve.csv')
```

```
ds.info()
```

```
<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   cp              303 non-null    int64  
 3   trestbps        303 non-null    int64  
 4   chol            303 non-null    int64  
 5   fbs             303 non-null    int64  
 6   restecg         303 non-null    int64  
 7   thalach         303 non-null    int64  
 8   exang           303 non-null    int64  
 9   oldpeak         303 non-null    float64
 10  slop            303 non-null    int64  
 11  ca              299 non-null    float64
 12  thal            301 non-null    float64
 13  pred_attribute  303 non-null    int64  
dtypes: float64(3), int64(11)
memory usage: 33.3 KB
```

```
ds.head(6)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop	ca	th
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3
5	56	1	2	120	236	0	0	178	0	0.8	1	0.0	3

```
ds.shape
```

```
(303, 14)
```

2. Explore the Data

Let's describe the dataset using descriptive statistics.

```
ds.describe()
```

	age	sex	cp	trestbps	chol	fbs	reste
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.9900
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.9949
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.0000
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.0000
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.0000
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.0000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.0000

Check if there are any Null values in the dataset

```
ds.isnull()
```

```
          age    sex     cp  trestbps   chol     fbs  restecg  thalach  exang  oldpeak   sl
```

```
print("No of Nan values in the dataframe are : ", sum(ds.isnull().any()))
```

```
No of Nan values in the dataframe are :  2
```

```
# pred_attribute is the target attribute
ds.pred_attribute.value_counts()
```

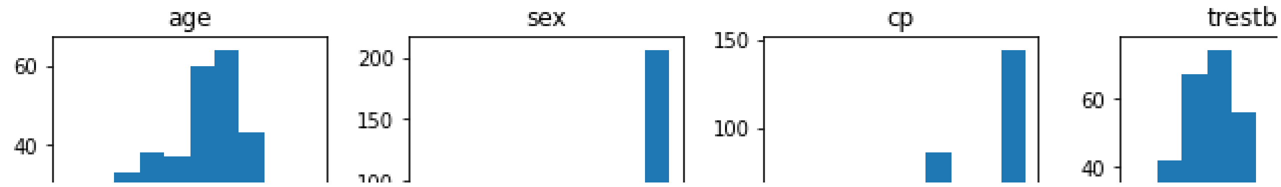
```
0    164
1     55
2     36
3     35
4     13
Name: pred_attribute, dtype: int64
```

Double-click (or enter) to edit

```
302  False  False  False     False  False  False    False    False  False    False  Fal
```

```
ds.hist(figsize=(12,12), grid=False)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f83766b9f50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f83766b11d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f837668ab50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f837663abd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f83765f8110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f83765aa610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8376561b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8376599fd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8376525050>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8376558650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f83764c5f50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8376488490>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f837643e990>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f83763f4e90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f83763b63d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f837636b8d0>]],
      dtype=object)
```

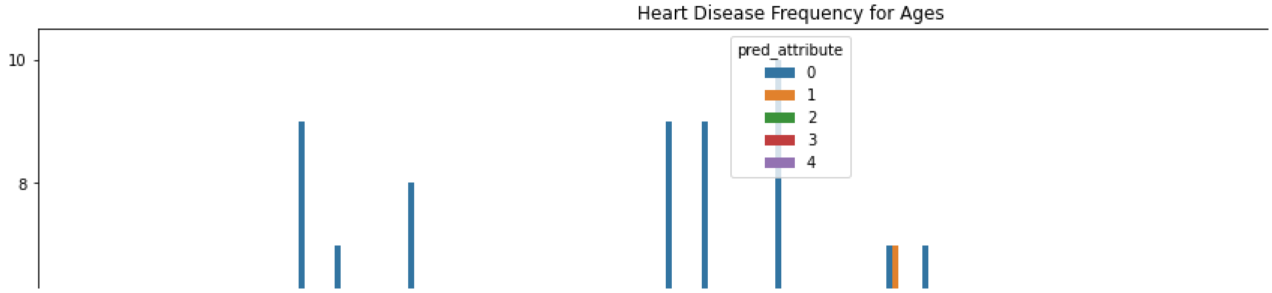


target attribute classes

- 0 - healthy
- 1 - sick level 1
- 2 - sick level 2
- 3 - sick level 3
- 4 - sick level 4

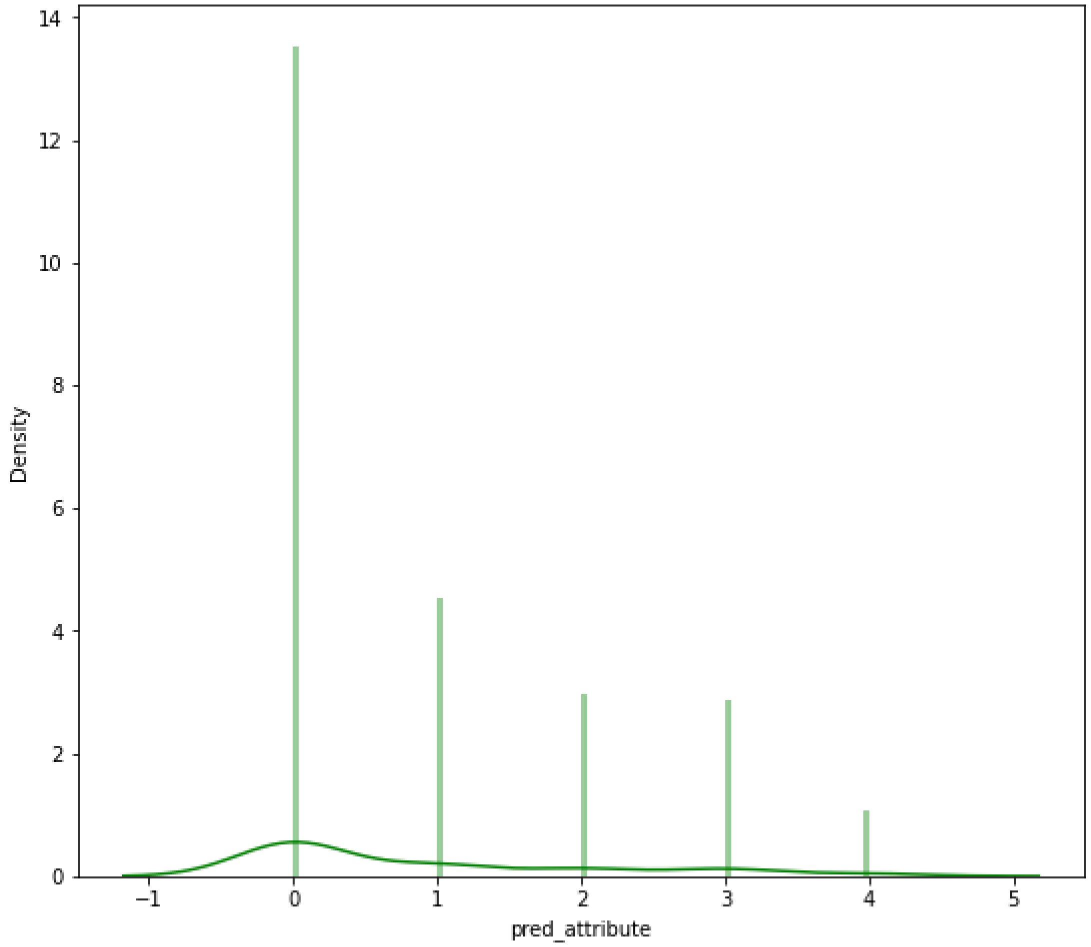
```
plt.figure(figsize=(18,8))
sns.countplot(x='age', data = ds, hue="pred_attribute")
plt.title('Heart Disease Frequency for Ages')
```

```
Text(0.5, 1.0, 'Heart Disease Frequency for Ages')
```

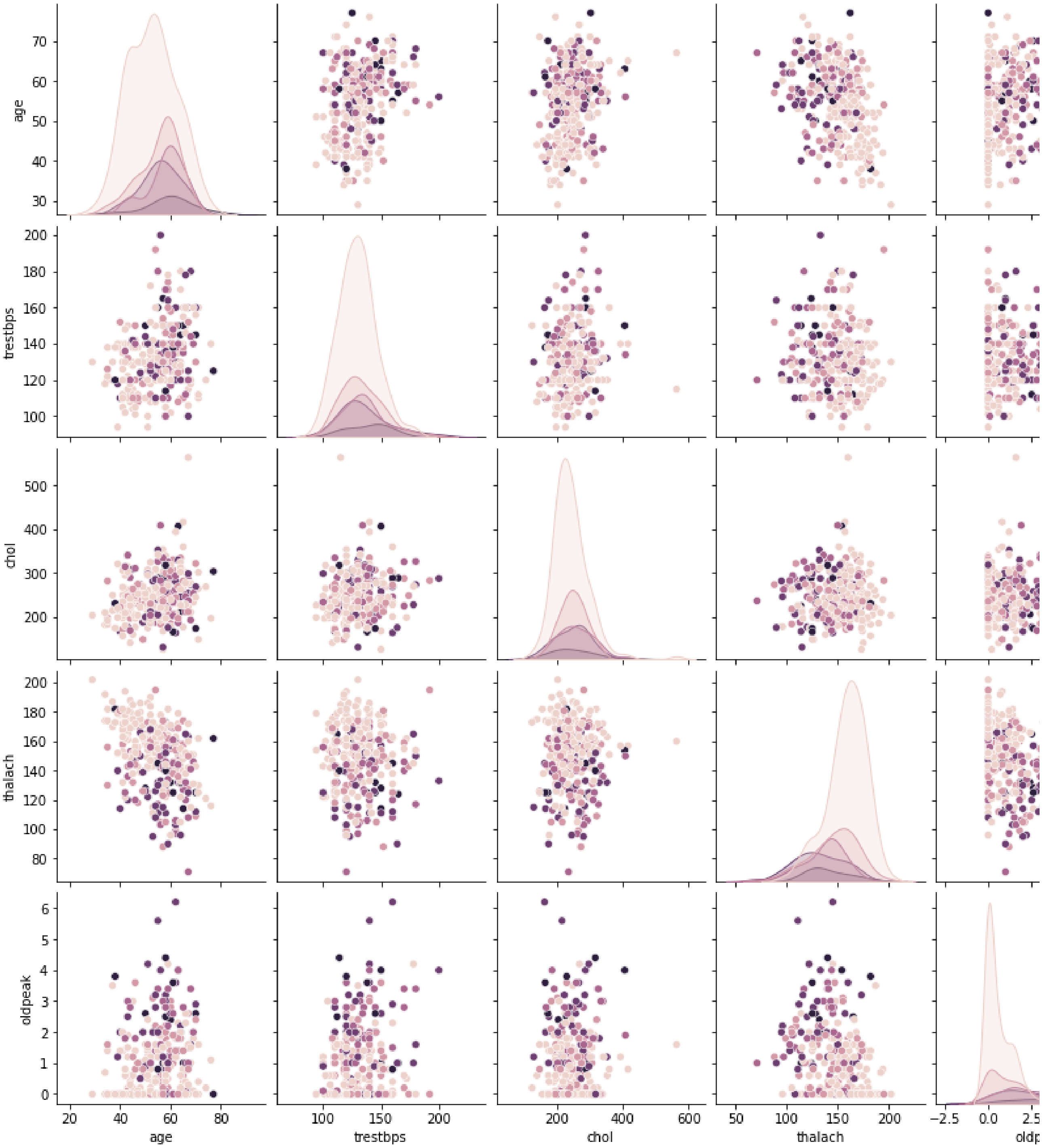


```
import matplotlib.pyplot as plt
import seaborn as sns
print(ds['pred_attribute'].describe())
plt.figure(figsize=(9, 8))
sns.distplot(ds['pred_attribute'], color='g', bins=100, hist_kws={'alpha': 0.4});
```

```
count    303.000000
mean       0.937294
std        1.228536
min        0.000000
25%        0.000000
50%        0.000000
75%        2.000000
max        4.000000
Name: pred_attribute, dtype: float64
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
```



```
sns.pairplot(ds.loc[:, ['age','trestbps','chol','thalach','oldpeak', 'pred_attribute']], h
<seaborn.axisgrid.PairGrid at 0x7f8375cebd90>
```



```
data = pd.get_dummies(ds, drop_first=False)
data.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slop', 'ca', 'thal', 'pred_attribute'],
          dtype='object')
```

```
ds['ca'] = ds['ca'].fillna(ds['ca'].mode()[0])
ds['thal'] = ds['thal'].fillna(ds['thal'].mode()[0])
```

3. Build the Predictive Model

```
from sklearn.model_selection import train_test_split
import numpy as np
```

```
y = ds['pred_attribute']
ds.drop('pred_attribute', axis=1, inplace=True)
x = ds
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y)
```

```
y_train.shape
```

```
     (227,)
```

```
x_train.shape
```

```
     (227, 13)
```

```
x_test.shape
```

```
     (76, 13)
```

```
y_test.shape
```

```
     (76,)
```

3.1- k Nearest Neighbor (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(x_train,y_train)
```

```
     KNeighborsClassifier(n_neighbors=10)
```

```
p=knn.predict_proba(x_test)
```

4. Evaluate the Model

```
from sklearn.metrics import log_loss
log_loss(y_test,p)
```

```
2.472053807472817
```

```
#calculate average
i=0
average=0
for j in range(0,10):
    x_train, x_test, y_train, y_test = train_test_split(x,y)
    knn.fit(x_train,y_train)
    pred = knn.predict_proba(x_test)
    i=log_loss(y_test,p)
    average=average+i
average=average/10
print(average)
```

```
4.82149991360461
```

✓ 0s completed at 4:53 PM ● ×