```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
get_ipython().run_line_magic('matplotlib', '')
import warnings
warnings.filterwarnings('ignore')


# In[3]:


hcare=pd.read_excel("1645792390_cepl_dataset.xlsx")


# hcare.head()

# In[4]:


hcare.tail()


# In[5]:


hcare.info()


# In[6]:


hcare.dtypes


# In[7]:


#Checking for missing Values
hcare.isnull().sum(axis=0)


# In[8]:


hcare.describe()


# In[9]:


# For visualizations
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns


# In[10]:


fig = plt.figure(figsize = (40,30))
hcare.hist(ax = fig.gca());


# From the above histogram plots, we can see that the features are skewed and not normally distributed. Also, the scales are different between one and another.
#
# <b><font size="3">Understanding the Data</font></b>

# In[11]:


# Creating a correlation heatmap
sns.heatmap(hcare.corr(),annot=True, cmap='terrain', linewidths=0.1)
fig=plt.gcf()
fig.set_size_inches(20,20)
plt.show()


# *From the above HeatMap, we can see that cp and thalach are the features with highest positive correlation whereas exang, oldpeak and ca are negatively correlated.While other feature

# <b>Outlier Detection</b>
#
# *Since the dataset is not large, we cannot discard the outliers. We will treat the outliers as potential observations.*

# In[13]:


# Boxplots
fig_dims = (15,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.boxplot(data=hcare, ax=ax);


# <b>Handling Imbalance</b>
#
# Imbalance in a dataset leads to inaccuracy and high precision, recall scores. There are certain resampling techniques such as undersampling and oversampling to handle these issues.
#
# Considering our dataset, the response variable target has two outcomes "Patients with Heart Disease" and "Patients without Heart Disease". Let us now observe their distribution in th

# In[14]:


hcare["target"].value_counts()


# From the above chart, we can conclude even when the distribution is not exactly 50:50, but still the data is good enough to use on machine learning algorithms and to predict standard

# <b>Train-Test Split</b>

# Let us distribute the data into **training** and **test** datasets using the **train_test_split()** function.
```

```
# In[15]:


X = hcare.drop("target",axis=1)
y = hcare["target"]


# <b>Logistic Regression</b>

# In[16]:


from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,stratify=y,random_state=7)


# In[17]:


from sklearn.linear_model import LogisticRegression


# In[18]:


lr = LogisticRegression()
lr.fit(X_train, y_train)


# In[19]:


pred = lr.predict(X_test)


# In[20]:


from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# In[21]:


# Accuracy on Test data
accuracy_score(y_test, pred)


# In[22]:


# Accuracy on Train data
accuracy_score(y_train, lr.predict(X_train))


# <b>Building a predictive system</b>

# In[23]:


import warnings
in_data = (57,0,0,140,241,0,1,123,1,0.2,1,0,3)

# Changing the input data into a numpy array
in_data_as_numpy_array = np.array(in_data)

# Reshaping the numpy array as we predict it
in_data_reshape = in_data_as_numpy_array.reshape(1,-1)
pred = lr.predict(in_data_reshape)
print(pred)

if(pred[0] == 0):
    print('The person does not have heart disease.')
else:
    print('The person has heart disease.')


# In[ ]:
```