Eval 1 Set 1 Test Zero

Zeel Mehta 21BAI1533

Data Set Description

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type 0: Typical angina: chest pain related decrease blood supply to the heart 1: Atypical angina: chest pain not related to heart 2: Non-anginal pain: typically esophageal spasms (non heart related) 3: Asymptomatic: chest pain not showing signs of disease
4. trestbps - resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. chol - serum cholestoral in mg/dl serum = LDL + HDL + .2 * triglycerides above 200 is cause for concern
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false) '>126' mg/dL signals diabetes
7. restecg - resting electrocardiographic results 0: Nothing to note 1: ST-T Wave abnormality can range from mild symptoms to severe problems signals non-normal heart beat 2: Possible or definite left ventricular hypertrophy Enlarged heart's main pumping chamber
8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during excercise unhealthy heart will stress more
11. slope - the slope of the peak exercise ST segment 0: Upsloping: better heart rate with excercise (uncommon) 1: Flatsloping: minimal change (typical healthy heart) 2: Downslopins: signs of unhealthy heart
12. ca - number of major vessels (0-3) colored by flourosopy colored vessel means the doctor can see the blood passing through the more blood movement the better (no clots)
13. thal - thalium stress result 1,3: normal 6: fixed defect: used to be defect but ok now 7: reversable defect: no proper blood movement when excercising
14. AHD/target - have disease or not (1=yes, 0=no) (= the predicted attribute)

Add necessary libararies/ modules

In [69]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Read the dataset and display the first 40 records an Identify the columns (1 Mark)

In [70]:
```
df=pd.read_csv('/Users/zeelmehta/Desktop/FALL INTER 23/ML/testzero/
```

In [71]:
```
df.head(40)
```

Out[71]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Old
0	1	63	1	typical	145	233	1	2	150	0	
1	2	67	1	asymptomatic	160	286	0	2	108	1	
2	3	67	1	asymptomatic	120	229	0	2	129	1	
3	4	37	1	nonanginal	130	250	0	0	187	0	
4	5	41	0	nontypical	130	204	0	2	172	0	
5	6	56	1	nontypical	120	236	0	0	178	0	
6	7	62	0	asymptomatic	140	268	0	2	160	0	
7	8	57	0	asymptomatic	120	354	0	0	163	1	
8	9	63	1	asymptomatic	130	254	0	2	147	0	
9	10	53	1	asymptomatic	140	203	1	2	155	1	

In [72]:
```
df.columns
```

Out[72]:
```
Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol',
'Fbs',
       'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Tha
l', 'AHD'],
      dtype='object')
```

#Check any duplicates are there using duplicated() method. Some duplicates are there. (Hint : Unnamed: 0 - Column is creating the problem) (2 Marks)

- Display number of duplcaites in the dataset

In [73]:
```
df.drop('Unnamed: 0',axis=1)
```

Out[73]:

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slo
0	63	1	typical	145	233	1	2	150	0	2.3	
1	67	1	asymptomatic	160	286	0	2	108	1	1.5	
2	67	1	asymptomatic	120	229	0	2	129	1	2.6	
3	37	1	nonanginal	130	250	0	0	187	0	3.5	
4	41	0	nontypical	130	204	0	2	172	0	1.4	
...	...	...	...	...	...	...	...	...	...	...	
397	39	0	nonanginal	138	220	0	0	152	0	0.0	
398	57	1	nontypical	154	232	0	2	164	0	0.0	
399	58	0	asymptomatic	130	197	0	0	131	0	0.6	
400	57	1	asymptomatic	110	335	0	0	143	1	3.0	
401	47	1	nonanginal	130	253	0	0	179	0	0.0	

In [74]:
```
df.duplicated()
```

Out[74]:
```
0      False
1      False
2      False
3      False
4      False
       ...
397    False
398    False
399    False
400    False
401    False
Length: 402, dtype: bool
```

Check any Null values - Identify the suitable Imputing technique for the attribute Ca and apply the same. (2 Marks)

In [75]:
```
df['Ca'].isnull().values.any()
df['Thal'].isnull().values.any()
```

Out[75]:
```
True
```

In [76]:
```
df.dtypes
```

Out[76]:
```
Unnamed: 0      int64
Age             int64
Sex             int64
ChestPain      object
RestBP          int64
Chol            int64
Fbs             int64
RestECG         int64
MaxHR           int64
ExAng           int64
Oldpeak       float64
Slope           int64
Ca            float64
Thal           object
AHD            object
dtype: object
```

In [77]:
```
from sklearn.impute import SimpleImputer
imp=SimpleImputer(missing_values=np.NaN,strategy='mean')
df[['Ca']]=imp.fit_transform(df[['Ca']])
df
```

Out[77]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	O
0	1	63	1	typical	145	233	1	2	150	0	
1	2	67	1	asymptomatic	160	286	0	2	108	1	
2	3	67	1	asymptomatic	120	229	0	2	129	1	
3	4	37	1	nonanginal	130	250	0	0	187	0	
4	5	41	0	nontypical	130	204	0	2	172	0	
...	...	...	...	...	...	...	...	...	...	...	...
397	398	39	0	nonanginal	138	220	0	0	152	0	
398	399	57	1	nontypical	154	232	0	2	164	0	
399	400	58	0	asymptomatic	130	197	0	0	131	0	
400	401	57	1	asymptomatic	110	335	0	0	143	1	

In [78]:
```
df['Ca'].isnull().values.any()
```

Out[78]:
```
False
```

In [79]:
```
imp=SimpleImputer(missing_values=np.NaN,strategy='most_frequent')
df[['Thal']]=imp.fit_transform(df[['Thal']])
df
```

Out[79]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	O
0	1	63	1	typical	145	233	1	2	150	0	
1	2	67	1	asymptomatic	160	286	0	2	108	1	
2	3	67	1	asymptomatic	120	229	0	2	129	1	
3	4	37	1	nonanginal	130	250	0	0	187	0	
4	5	41	0	nontypical	130	204	0	2	172	0	
...	...	...	...	...	...	...	...	...	...	...	
397	398	39	0	nonanginal	138	220	0	0	152	0	
398	399	57	1	nontypical	154	232	0	2	164	0	
399	400	58	0	asymptomatic	130	197	0	0	131	0	
400	401	57	1	asymptomatic	110	335	0	0	143	1	

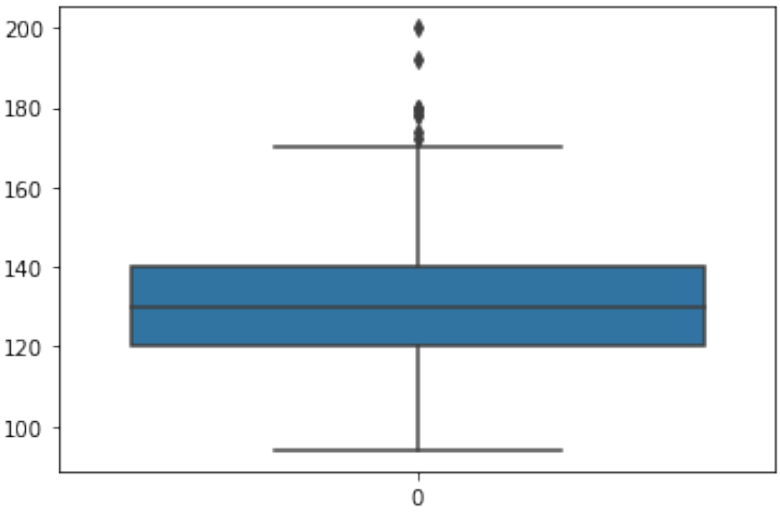
In [80]:
```
df['Thal'].isnull().values.any()
```

Out[80]:
```
False
```

Visualize the outlier data using Box Plot for the column : RestBP 1 (1 Mark)

In [81]:
```
sns.boxplot(data=df['RestBP'])
```

Out[81]:
```
<AxesSubplot:>
```



Remove the outliers from the data set based on the column : RestBP (2 Marks) USING Z Score method

In [82]:
```
#standard deviation
m=df['RestBP'].mean()
sd=df['RestBP'].std()
```

In [83]:
```
#-3sigma to +3sigma as stanard deviation Hint: df['col'].mean to co

upper_limit =
lower_limit =
print(upper_limit)
print(lower_limit)
```

```
  Input In [83]
    upper_limit =
                  ^
SyntaxError: invalid syntax
```

Visualize the Correlation between the MaxHR vs AHD (2 Marks)

In [84]:
```
df['AHD'].unique()
```

Out[84]:
```
array(['No', 'Yes'], dtype=object)
```

In [85]:
```
df[['AHD']]=df[['AHD']].apply(pd.to_numeric)
```

```
---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/_libs/lib.
pyx:2315, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "No"

During handling of the above exception, another exception occurred
:

ValueError                                Traceback (most recent c
all last)
Input In [85], in <cell line: 1>()
----> 1 df[['AHD']]=df[['AHD']].apply(pd.to_numeric)
```

```
File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/frame
.py:8839, in DataFrame.apply(self, func, axis, raw, result_type, a
rgs, **kwargs)
   8828 from pandas.core.apply import frame_apply
   8830 op = frame_apply(
   8831     self,
   8832     func=func,
   (...)
   8837     kwargs=kwargs,
   8838 )
-> 8839 return op.apply().__finalize__(self, method="apply")

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/apply
.py:727, in FrameApply.apply(self)
    724 elif self.raw:
    725     return self.apply_raw()
--> 727 return self.apply_standard()

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/apply
.py:851, in FrameApply.apply_standard(self)
    850 def apply_standard(self):
--> 851     results, res_index = self.apply_series_generator()
    853     # wrap results
    854     return self.wrap_results(results, res_index)

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/apply
.py:867, in FrameApply.apply_series_generator(self)
    864 with option_context("mode.chained_assignment", None):
    865     for i, v in enumerate(series_gen):
    866         # ignore SettingWithCopy here in case the user mut
ates
--> 867         results[i] = self.f(v)
    868         if isinstance(results[i], ABCSeries):
    869             # If we have a view on v, we need to make a co
py because
    870             #  series_generator will swap out the underlyi
ng data
    871             results[i] = results[i].copy(deep=False)

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/core/tools
/numeric.py:184, in to_numeric(arg, errors, downcast)
    182 coerce_numeric = errors not in ("ignore", "raise")
    183 try:
--> 184     values, _ = lib.maybe_convert_numeric(
    185         values, set(), coerce_numeric=coerce_numeric
    186     )
    187 except (ValueError, TypeError):
    188     if errors == "raise":

File ~/opt/anaconda3/lib/python3.9/site-packages/pandas/_libs/lib.
pyx:2357, in pandas._libs.lib.maybe_convert_numeric()
```

```
ValueError: Unable to parse string "No" at position 0
```

In [86]:
```
x=df[['MaxHR','AHD']]
sns.heatmap(x.corr())
```

Out[86]: <AxesSubplot:>

In [ ]:

In [ ]: