```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
rcParams['figure.figure.max_open_warning'] = 0
```

```
df2.get_dummies('Thalassemia',drop_first=True)
```

Preliminary analysis

```
df2.head(20)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
14	58	0	3	150	283	1	1	162	0	1	2	0	2	1
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
16	65	1	0	138	282	1	2	174	0	1.4	1	1	3	1
17	62	0	0	124	209	0	1	163	0	0	2	0	2	1
18	54	1	2	134	304	0	1	162	0	0	2	2	2	1
19	51	1	3	125	213	0	0	125	1	1.4	1	1	2	1

```
df2.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  slope    303 non-null    int64
 11  ca       303 non-null    int64
 12  thal     303 non-null    int64
 13  target   303 non-null    int64
dtype: int64
memory usage: 33.1 KB
```

```
df2.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

```
df2.shape
```

```
(303, 14)
```

```
df2.drop_duplicates(keep=False)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'df2' is not defined
```

Checking duplicate values in the data

```
df2.duplicated()
```

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9    False
10   False
11   False
12   False
13   False
14   False
15   False
16   False
17   False
18   False
19   False
dtype: bool
```

```
df2.drop_duplicates(keep=False)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'df2' is not defined
```

Preliminary statistical summary of the data

```
df2.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

```
df2.shape
```

```
(303, 14)
```

```
df2.dtypes
```

```
age       int64
sex       int64
cp        int64
trestbps  int64
chol      int64
fbs       int64
restecg   int64
thalach   int64
exang     int64
oldpeak   float64
slope     int64
ca        int64
thal      int64
target    int64
dtype: int64
```

Splitting the attributes into Numerical and Categorical

```
df2[df2['sex'] == 'Female']['sex'].value_counts()
df2[df2['sex'] == 'Male']['sex'].value_counts()
```

```
Series([], Name: sex, dtype: int64)
Series([], Name: sex, dtype: int64)
```

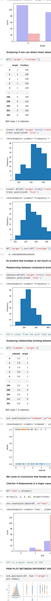
```
df2[df2['target'] == 'Positive']['target'].value_counts()
df2[df2['target'] == 'Negative']['target'].value_counts()
```

```
Series([], Name: target, dtype: int64)
Series([], Name: target, dtype: int64)
```

DISTRIBUTION OF CATEGORICAL VARIABLES

```
for i, j in enumerate(cateforicals):
    plt.subplot(4,2,i+1)
    df2.groupby(j).count().plot(kind="pie", y="_id", legend=False)
    plt.title("Distribution by %s"%j)
plt.tight_layout()
```

```
for i, j in enumerate(cateforicals):
    plt.subplot(4,2,i+1)
    df2.groupby(j).count().plot(kind="bar", y="_id", legend=False)
    plt.title("Distribution by %s"%j)
plt.tight_layout()
```



DISTRIBUTION OF NUMERIC VARIABLES

```
for i, j in enumerate(numerics):
    plt.subplot(4,2,i+1)
    df2[df2['target'] == 0][j].hist(bins=35, color='blue', label='No', alpha=0.6, density=True)
    df2[df2['target'] == 1][j].hist(bins=35, color='red', label='Yes', alpha=0.6, density=True)
    plt.legend(loc='upper right')
    plt.title("Distribution of %s"%j)
plt.tight_layout()
```

```
for i, j in enumerate(numerics):
    plt.subplot(4,2,i+1)
    df2[df2['target'] == 0].boxplot(column=j, by='target')
    df2[df2['target'] == 1].boxplot(column=j, by='target')
    plt.title("Boxplot of %s"%j)
plt.tight_layout()
```



```
df2.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Count plot of categorical variables

```
for i, j in enumerate(cateforicals):
    plt.subplot(4,2,i+1)
    df2[df2['target'] == 0][j].hist(bins=35, color='blue', label='No', alpha=0.6, density=True)
    df2[df2['target'] == 1][j].hist(bins=35, color='red', label='Yes', alpha=0.6, density=True)
    plt.legend(loc='upper right')
    plt.title("Distribution of %s"%j)
plt.tight_layout()
```



```
sns.countplot(df2['target'],hue='sex')
```



```
sns.countplot(df2['target'],hue='cp')
```



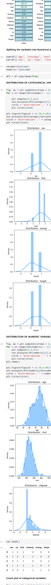
```
sns.countplot(df2['target'],hue='fbs')
```



```
sns.countplot(df2['target'],hue='restecg')
```



```
sns.countplot(df2['target'],hue='exang')
```




```
sns.countplot(df2['target'],hue='slope')
```



```
sns.countplot(df2['target'],hue='ca')
```



```
sns.countplot(df2['target'],hue='thal')
```


Inference of FOR across the hue category

```
sns.countplot(df2[df2['age'].isin([ 29, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 74, 76, 77])],hue='target',palette = 'cubehelix')
```



```
sns.countplot(df2[df2['age'].isin([ 29, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 74, 76, 77])],hue='target',palette = 'cubehelix')
```


Comparision of all patients with regards to the hue category

```
df2[df2['target'] == 'Positive']['age'].hist(bins=35,color='blue',alpha=0.6)
df2[df2['target'] == 'Negative']['age'].hist(bins=35,color='red',alpha=0.6)
```

```
Series([], Name: age, dtype: int64)
Series([], Name: age, dtype: int64)
```

Comparision of patients with regards to the hue category

```
df2[df2['target'] == 'Positive']['sex'].hist(bins=35,color='blue',alpha=0.6)
df2[df2['target'] == 'Negative']['sex'].hist(bins=35,color='red',alpha=0.6)
```

```
Series([], Name: sex, dtype: int64)
Series([], Name: sex, dtype: int64)
```

```
sns.countplot(df2['target'],hue='cp')
```


Analyzing if one can draw heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient

```
df2[df2['target'] == 1].head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Bar plot of trestbps

```
df2[df2['target'] == 1]['trestbps'].hist(bins=35)
```

```
<matplotlib.axes._subplot.AxesSubplot object at 0x7f8d9d1ed7d0>
```



```
df2[df2['target'] == 0]['trestbps'].hist(bins=35)
```

```
<matplotlib.axes._subplot.AxesSubplot object at 0x7f8d9d1ed7d0>
```


We infer a conclusion that people are more prone to CVD at low chol

Chestnut Frequencies in a major cause of CVD

```
df2.hist(figsize=(
```

```
array([[<AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```

```
sns.countplot(df2[df2['age'].isin([ 29, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 74, 76, 77])],hue='target',palette='Pastel1')
```



```
sns.countplot(df2[df2['age'].isin([ 29, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 74, 76, 77])],hue='sex',palette='Pastel1')
```



```
sns.countplot(df2[df2['age'].isin([ 29, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 74, 76, 77])],hue='cp',palette='Pastel1')
```


DISTRIBUTION OF CATEGORICAL VARIABLES

```
for i, j in enumerate(cateforicals):
    plt.subplot(4,2,i+1)
    df2.groupby(j).count().plot(kind="pie", y="_id", legend=False)
    plt.title("Distribution by %s"%j)
plt.tight_layout()
```



```
for i, j in enumerate(cateforicals):
    plt.subplot(4,2,i+1)
    df2.groupby(j).count().plot(kind="bar", y="_id", legend=False)
    plt.title("Distribution by %s"%j)
plt.tight_layout()
```



```
df2.head(20)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
14	58	0	3	150	283	1	1	162	0	1	2	0	2	1
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
16	65	1	0	138	282	1	2	174	0	1.4	1	1	3	1
17	62	0	0	124	209	0	1	163	0	0	2	0	2	1
18	54	1	2	134	304	0	1	162	0	0	2	2	2	1
19	51	1	3	125	213	0	0	125	1	1.4	1	1	2	1

Feature selection

```
df2.corr()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.203910	0.118530	0.148868	-0.398522	0.096810	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	0.141467	-0.044018	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076848	0.094444	0.070733	0.309731	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	0.146562	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.203910	-0.197912	-0.076848	0.123174	1.000000	0.013294	0.152643	-0.003032	0.067780	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.118530	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	0.148868	0.141467	0.070733	0.146562	0.152643	-0.084189	1.000000	-0.044020	0.096093	0.070111	-0.058757	0.093044	0.030847	-0.137230
thalach	-0.398522	-0.044018	0.309731	-0.046698	-0.003032	-0.008567	-0.044020	1.000000	-0.378812	-0.344187	0.093045	-0.073732	-0.096351	0.421741
exang	0.096810	0.141664	-0.394280	0.067616	0.067780	0.025665	0.096093	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	0.070111	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	-0.058757	0.093045	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	0.093044	-0.073732	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	0.030847	-0.096351	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	-0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

```
df2.shape
```

```
(303, 14)
```

```
df2.dtypes
```

```
age       int64
sex       int64
cp        int64
trestbps  int64
chol      int64
fbs       int64
restecg   int64
thalach   int64
exang     int64
oldpeak   float64
slope     int64
ca        int64
thal      int64
target    int64
dtype: int64
```

```
df2.drop(['target'],axis=1)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2
7	44	1	1	120	263	0	1	173	0	0	2	0	3
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2
14	58	0	3	150	283	1	1	162	0	1	2	0	2
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
16	65	1	0	138	282	1	2	174	0	1.4	1	1	3
17	62	0	0	124	209	0	1	163	0	0	2	0	2
18	54	1	2	134	304	0	1	162	0	0	2	2	2
19	51	1	3	125	213	0	0	125	1	1.4	1	1	2

```
df2.target
```

```
0    1
1    1
2    1
3    1
4    1
5    1
6    1
7    1
8    1
9    1
10   1
11   1
12   1
13   1
14   1
15   1
16   1
17   1
18   1
19   1
Name: target, dtype: int64
```

Evaluation of Hot Encoding(dummy) on dataset

```
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split,cross_validation,metrics,tree,ensemble,naive_bayes
from sklearn.metrics import
```

```
from sklearn.linear_model import LogisticRegression
```

```
LogisticRegression()
```

```
LogisticRegression()
```

```
plt.plot(np.arange(0,1,0.1),np.
plt.grid
```

```
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,
       1. ])
```

```
<bound_method_Axes2D.grid of <matrix_plot_1>>
```

Evaluation of Hot Encoding(dummy) on dataset

```
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier,AdaBoostClassifier,GradientBoostingClassifier,BaggingClassifier,VotingClassifier
from sklearn.naive_bayes import
```

```
from sklearn.naive_bayes import GaussianNB,BernoulliNB
```

```
GaussianNB()
```

```
GaussianNB()
```

```
from sklearn.naive_bayes import BernoulliNB
BernoulliNB().get_params()
```

```
{'alpha': 1.0, 'binarize': 0.0, 'fit_prior': True}
```

```
BernoulliNB()
```

```
BernoulliNB()
```

Evaluation of Hot Encoding(dummy) on dataset

```
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier,AdaBoostClassifier,GradientBoostingClassifier,BaggingClassifier,VotingClassifier
from sklearn.naive_bayes import
```

```
from sklearn.naive_bayes import GaussianNB,BernoulliNB
```

```
GaussianNB()
```

```
GaussianNB()
```

```
from sklearn.naive_bayes import BernoulliNB
BernoulliNB().get_params()
```

```
{'alpha': 1.0, 'binarize': 0.0, 'fit_prior': True}
```

```
BernoulliNB()
```

```
BernoulliNB()
```

Evaluation of Hot Encoding(dummy) on dataset

```
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier,AdaBoostClassifier,GradientBoost
```