

Code to import csv file :

```
In [1]: from google.colab import files
uploaded = files.upload()
```

No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving heart.csv to heart.csv

Importing Libraries :

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import RobustScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.svm import LinearSVC
import warnings
warnings.filterwarnings('ignore')
```

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.

```
import pandas.util.testing as tm
```

```
In [4]: df = pd.read_csv("heart.csv")
```

In [5]: df.head()

Out[5]:

	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

In [6]: df.tail()

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

Data Type :

In [7]: df.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
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Checking for missing Value :

```
In [8]: df.isnull().sum()
```

```
Out[8]: age      0
sex        0
cp         0
trestbps   0
chol       0
fbs        0
restecg    0
thalach    0
exang      0
oldpeak    0
slope      0
ca         0
thal       0
target     0
dtype: int64
```

```
In [9]: df.describe()
```

```
Out[9]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
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
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.720000
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.020000
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
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
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
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
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

Correlation :

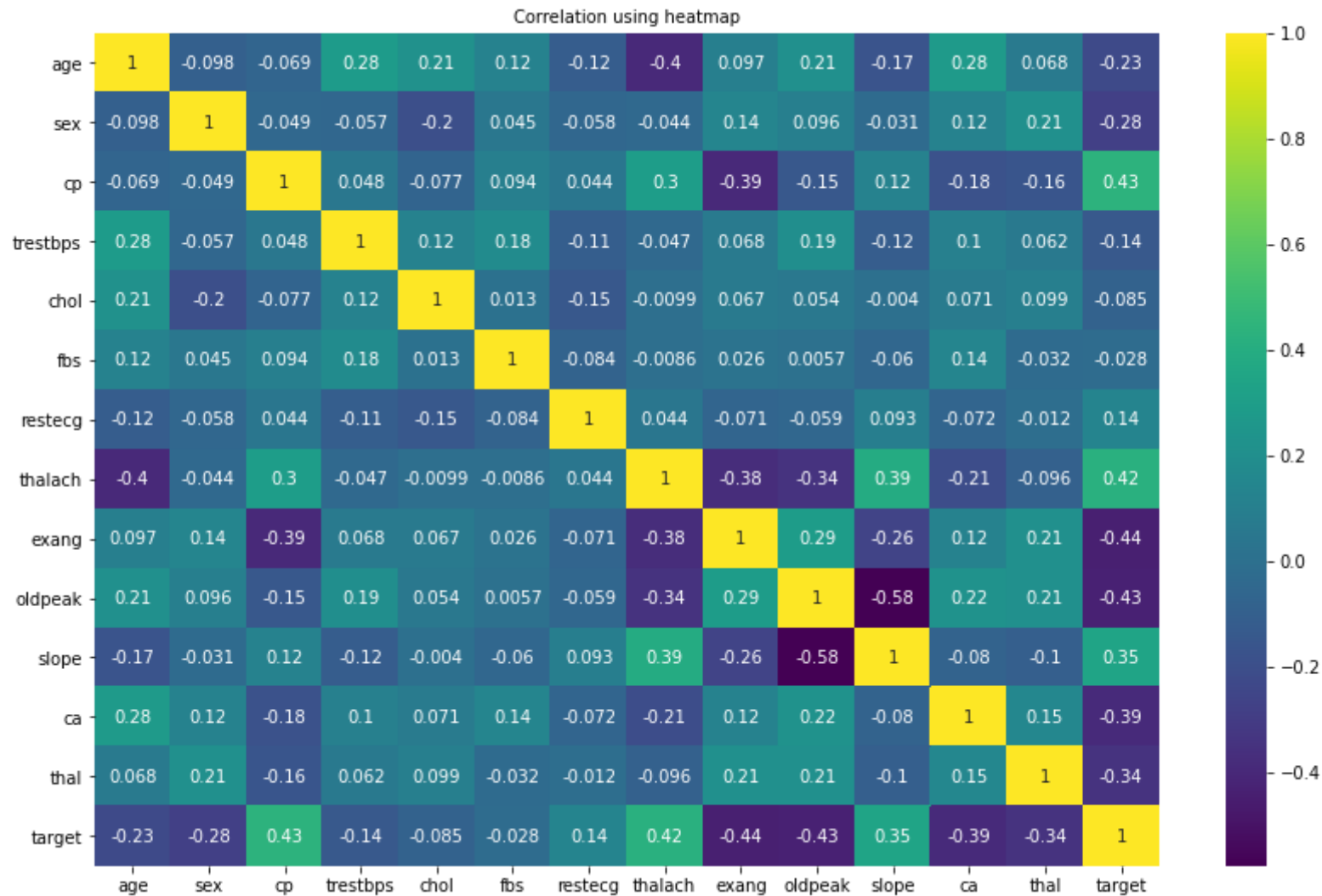
In [12]: df.corr()

Out[12]:

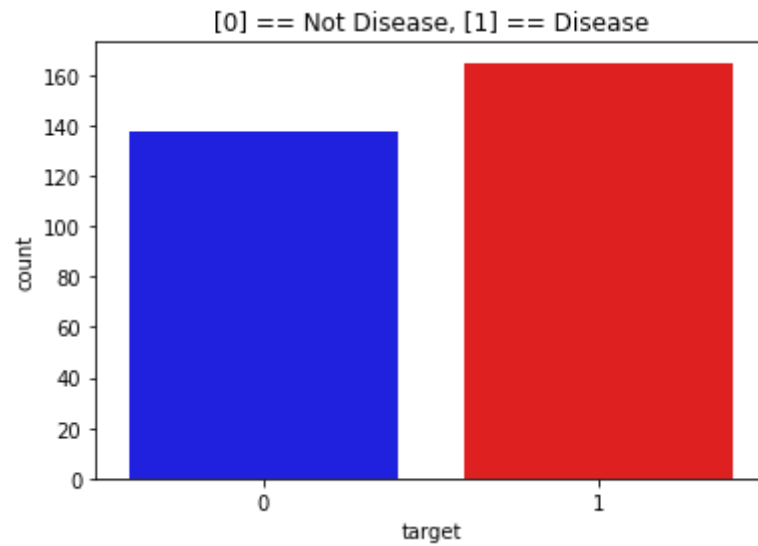
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000
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

Heatmap :

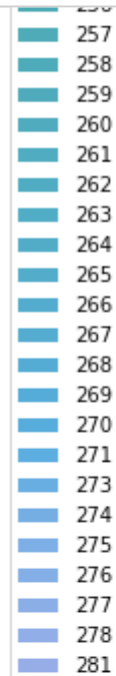
```
In [16]: corr=df.corr()
fig,ax=plt.subplots(figsize=(14,9))
sns.heatmap(corr,annot=True,cmap = 'viridis',annot_kws={"size": 10})
plt.title("Correlation using heatmap",size=10);
```



```
In [22]: sns.countplot(df.target, palette=['blue', 'red'])  
plt.title("[0] == Not Disease, [1] == Disease");
```



```
In [26]: for col in df:
plt.figure()
sns.countplot(data=df,x="target", hue=col)
plt.show()
print("-----")
```



```
In [27]: X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

```
In [28]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

```
In [29]: lsv = LinearSVC(random_state=1)
```

```
In [30]: lsv.fit(X_train, y_train)
```

```
Out[30]: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
intercept_scaling=1, loss='squared_hinge', max_iter=1000,
multi_class='ovr', penalty='l2', random_state=1, tol=0.0001,
verbose=0)
```



```
In [31]: y_pred = lsv.predict(X_test)
```

```
In [32]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.87	0.66	0.75	41
1	0.77	0.92	0.84	50
accuracy			0.80	91
macro avg	0.82	0.79	0.79	91
weighted avg	0.81	0.80	0.80	91

Soft Margin :

```
In [33]: lsv2 = LinearSVC(C=0.5,random_state=1)
```

```
In [34]: lsv2.fit(X_train,y_train)
```

```
Out[34]: LinearSVC(C=0.5, class_weight=None, dual=True, fit_intercept=True,
    intercept_scaling=1, loss='squared_hinge', max_iter=1000,
    multi_class='ovr', penalty='l2', random_state=1, tol=0.0001,
    verbose=0)
```

```
In [35]: y_pred = lsv.predict(X_test)
```

```
In [36]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.87	0.66	0.75	41
1	0.77	0.92	0.84	50
accuracy			0.80	91
macro avg	0.82	0.79	0.79	91
weighted avg	0.81	0.80	0.80	91

Polynomial :

```
In [37]: from sklearn.svm import SVC
```

```
In [38]: svc = SVC(random_state=1, kernel="poly")
```

```
In [39]: svc.fit(X_train,y_train)
```

```
Out[39]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
max_iter=-1, probability=False, random_state=1, shrinking=True, tol=0.001,
verbose=False)
```

```
In [40]: y_pred = svc.predict(X_test)
```

```
In [41]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.59	0.54	0.56	41
1	0.65	0.70	0.67	50
accuracy			0.63	91
macro avg	0.62	0.62	0.62	91
weighted avg	0.62	0.63	0.62	91

Radial Bias :

```
In [42]: svc = SVC(random_state=1,kernel="rbf")
```

```
In [43]: svc.fit(X_train,y_train)
```

```
Out[43]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
max_iter=-1, probability=False, random_state=1, shrinking=True, tol=0.001,
verbose=False)
```

```
In [44]: y_pred = svc.predict(X_test)
```

```
In [45]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.59	0.46	0.52	41
1	0.63	0.74	0.68	50
accuracy			0.62	91
macro avg	0.61	0.60	0.60	91
weighted avg	0.61	0.62	0.61	91

In []: