Importing Pandas and Numpy

```python
import pandas as pd
import numpy as np
```

Loading Heart CSV DataSet

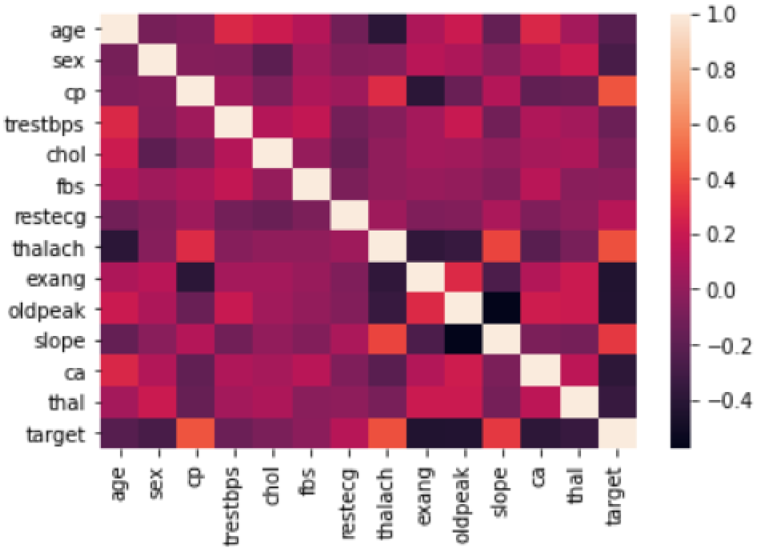
```python
data = pd.read_csv("../input/heart-disease-uci/heart.csv")
data
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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

Selectiing Features Base On Heat Map and Using Correlation Method

```
import seaborn as sns
sns.heatmap(data.corr())
```

```
<AxesSubplot:>
```



Selecting Features "'age', 'trestbps', 'chol', 'thalach', 'oldpeak'" in x variable

```
x = data[['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]
```

Doing OneHotEncoding using get_dummies to avoid Multi Collinearity

```
sex = pd.get_dummies(data['sex'], drop_first=True)
cp = pd.get_dummies(data['cp'], drop_first=True)
fbs = pd.get_dummies(data['fbs'], drop_first=True)
restecg = pd.get_dummies(data['restecg'], drop_first=True)
exang = pd.get_dummies(data['exang'], drop_first=True)
slope = pd.get_dummies(data['slope'], drop_first=True)
ca = pd.get_dummies(data['ca'], drop_first=True)
thal = pd.get_dummies(data['thal'], drop_first=True)
```

Concating x and the OneHotEncoded values into X variable

```
X = pd.concat([x, sex, cp, fbs, restecg, exang, slope, ca, thal ], axis=1)
X
```

```
X = pd.concat([x, sex, cp, fbs, restecg, exang, slope, ca, thal ], axis=1)
X
```

	age	trestbps	chol	thalach	oldpeak	1	1	2	3	1	...	1	1	2	1	2	3	4	1	2	3
0	63	145	233	150	2.3	1	0	0	1	1	...	0	0	0	0	0	0	0	1	0	0
1	37	130	250	187	3.5	1	0	1	0	0	...	0	0	0	0	0	0	0	0	1	0
2	41	130	204	172	1.4	0	1	0	0	0	...	0	0	1	0	0	0	0	0	1	0
3	56	120	236	178	0.8	1	1	0	0	0	...	0	0	1	0	0	0	0	0	1	0
4	57	120	354	163	0.6	0	0	0	0	0	...	1	0	1	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	140	241	123	0.2	0	0	0	0	0	...	1	1	0	0	0	0	0	0	0	1
299	45	110	264	132	1.2	1	0	0	1	0	...	0	1	0	0	0	0	0	0	0	1
300	68	144	193	141	3.4	1	0	0	0	1	...	0	1	0	0	1	0	0	0	0	1
301	57	130	131	115	1.2	1	0	0	0	0	...	1	1	0	1	0	0	0	0	0	1
302	57	130	236	174	0.0	0	1	0	0	0	...	0	1	0	1	0	0	0	0	1	0

303 rows × 22 columns

Adding 'target' features in y variable

```
y=data['target']
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Spliting Data into training and testing part

```python
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, random_state=40)
```

Using `RandomForestClassifier` Algorithm for training the model and predicting with trees=1000

```python
from sklearn.ensemble import RandomForestClassifier
rfc_model = RandomForestClassifier(n_estimators=100, max_depth=10)
rfc_model
rfc_model.fit(X_train, y_train)
rfc_y_pred = rfc_model.predict(X_test)
rfc_y_pred

from sklearn.metrics import accuracy_score
print("Random Forest Classifier Accuracy: ", accuracy_score(y_test, rfc_y_pred))
```

```
Random Forest Classifier Accuracy:  0.9180327868852459
```

Using `LogisticRegression` Algorithm for training the model and predicting

```
from sklearn.linear_model import LogisticRegression
lr_model=LogisticRegression()
lr_model.fit(X_train, y_train)
lr_y_model= lr_model.predict(X_test)
lr_y_model
from sklearn.metrics import accuracy_score
print("Logistic Regression Accuracy: ", accuracy_score(y_test, lr_y_model))
```

```
Logistic Regression Accuracy:  0.8524590163934426
```

Using `DecisionTreeClassifier` Algorithm for training the model and predicting

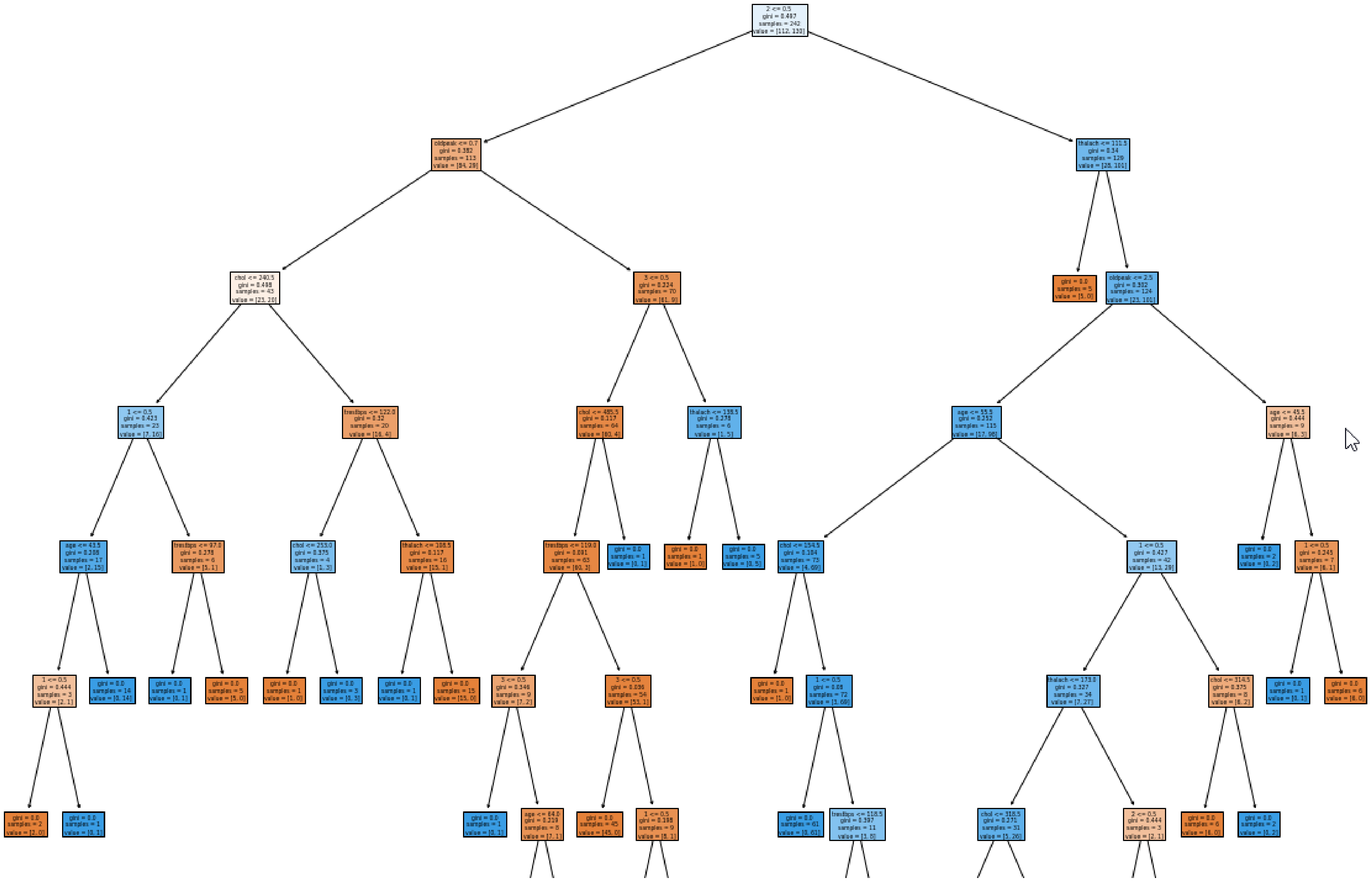
```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
dt_y_pred=dt_model.predict(X_test)
dt_y_pred
from sklearn.metrics import accuracy_score
print("Decision Tree Accuracy: ", accuracy_score(y_test, dt_y_pred))
```

```
Decision Tree Accuracy:  0.7049180327868853
```

Visualizing Decision Tree Classifier

```python
from matplotlib import pyplot as plt
from sklearn import tree
plt.figure(figsize=(20,20))
tree.plot_tree(dt_model, feature_names=X.columns, filled=True)
```

```
[Text(634.2890625, 1032.8400000000001, '2 <= 0.5\ngini = 0.497\nsamples = 242\nvalue = [112, 130]'),
 Text(372.0, 924.1200000000001, 'oldpeak <= 0.7\ngini = 0.382\nsamples = 113\nvalue = [84, 29]'),
 Text(209.25, 815.4000000000001, 'chol <= 240.5\ngini = 0.498\nsamples = 43\nvalue = [23, 20]'),
 Text(116.25, 706.6800000000001, '1 <= 0.5\ngini = 0.423\nsamples = 23\nvalue = [7, 16]'),
 Text(69.75, 597.96, 'age <= 43.5\ngini = 0.208\nsamples = 17\nvalue = [2, 15]'),
 Text(46.5, 489.24, '1 <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(23.25, 380.5200000000001, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(69.75, 380.5200000000001, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(93.0, 489.24, 'gini = 0.0\nsamples = 14\nvalue = [0, 14]'),
 Text(162.75, 597.96, 'trestbps <= 97.0\ngini = 0.278\nsamples = 6\nvalue = [5, 1]'),
 Text(139.5, 489.24, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(186.0, 489.24, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(302.25, 706.6800000000001, 'trestbps <= 122.0\ngini = 0.32\nsamples = 20\nvalue = [16, 4]'),
 Text(255.75, 597.96, 'chol <= 253.0\ngini = 0.375\nsamples = 4\nvalue = [1, 3]'),
 Text(232.5, 489.24, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(279.0, 489.24, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(348.75, 597.96, 'thalach <= 108.5\ngini = 0.117\nsamples = 16\nvalue = [15, 1]'),
 Text(325.5, 489.24, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(372.0, 489.24, 'gini = 0.0\nsamples = 15\nvalue = [15, 0]'),
 Text(534.75, 815.4000000000001, '3 <= 0.5\ngini = 0.224\nsamples = 70\nvalue = [61, 9]'),
 Text(488.25, 706.6800000000001, 'chol <= 485.5\ngini = 0.117\nsamples = 64\nvalue = [60, 4]'),
 Text(465.0, 597.96, 'trestbps <= 119.0\ngini = 0.091\nsamples = 63\nvalue = [60, 3]'),
 Text(418.5, 489.24, '3 <= 0.5\ngini = 0.346\nsamples = 9\nvalue = [7, 2]'),
 Text(395.25, 380.5200000000001, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(441.75, 380.5200000000001, 'age <= 64.0\ngini = 0.219\nsamples = 8\nvalue = [7, 1]'),
 Text(418.5, 271.80000000000007, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]'),
 Text(465.0, 271.80000000000007, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(511.5, 489.24, '3 <= 0.5\ngini = 0.036\nsamples = 54\nvalue = [53, 1]'),
 Text(488.25, 380.5200000000001, 'gini = 0.0\nsamples = 45\nvalue = [45, 0]'),
 Text(534.75, 380.5200000000001, '1 <= 0.5\ngini = 0.198\nsamples = 9\nvalue = [8, 1]'),
 Text(511.5, 271.80000000000007, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]'),
 Text(558.0, 271.80000000000007, 'oldpeak <= 1.9\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(534.75, 163.08000000000004, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(581.25, 163.08000000000004, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(511.5, 597.96, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(581.25, 706.6800000000001, 'thalach <= 138.5\ngini = 0.278\nsamples = 6\nvalue = [1, 5]'),
 Text(558.0, 597.96, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
```



```
from sklearn.ensemble import GradientBoostingClassifier
GB_model = GradientBoostingClassifier(n_estimators=1000)
GB_model.fit(X_train, y_train)
y_pred_GB = GB_model.predict(X_test)
y_pred_GB
from sklearn.metrics import accuracy_score
print("Gradient Boost Accuracy: ", accuracy_score(y_test, y_pred_GB))
```

```
Gradient Boost Accuracy:  0.7868852459016393
```

Choosing RandomForestClassifier Model and saving it using Joblib

```
import joblib
joblib_file = "RandomForest_Heart_Prediction.h5"
joblib.dump(lr_model, joblib_file)
```

```
['RandomForest_Heart_Prediction.h5']
```