```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots

from sklearn.metrics import classification_report


import warnings
warnings.filterwarnings("ignore")
```

```
DF=pd.read_csv('/content/heart.csv')
DF.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	52	1	0	125	212	0	1	168	0	1.0	2	2	:
1	53	1	0	140	203	1	0	155	1	3.1	0	0	:
2	70	1	0	145	174	0	1	125	1	2.6	0	0	:
3	61	1	0	148	203	0	1	161	0	0.0	2	1	:
4	62	0	0	138	294	1	1	106	0	1.9	1	3	:

Next steps: Generate code with DF View recommended plots

```
nRow, nCol = DF.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
There are 1025 rows and 14 columns
```

```
DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
DF.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: object
```

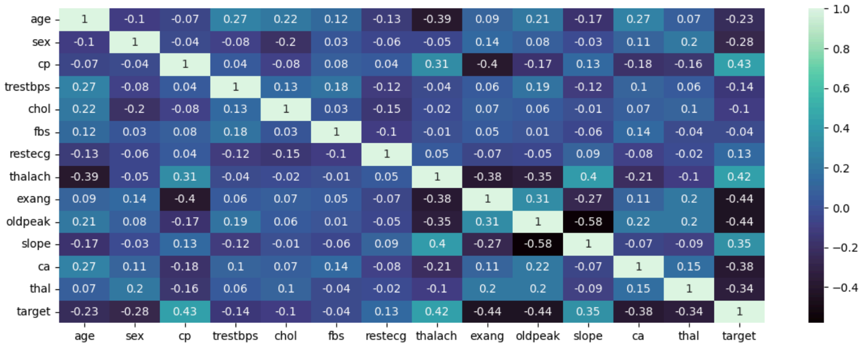
```
DF.describe().round(2).style.background_gradient()
```

	age	sex	cp	trestbps	chol	fbs
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.430000	0.700000	0.940000	131.610000	246.000000	0.150000
std	9.070000	0.460000	1.030000	17.520000	51.590000	0.360000
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	275.000000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000

```
DF_corr= DF.corr()
```

```
plt.figure(figsize = (14,5))
sns.heatmap(round(DF_corr,2),annot=True,cmap = 'mako')
```

```
<Axes: >
```

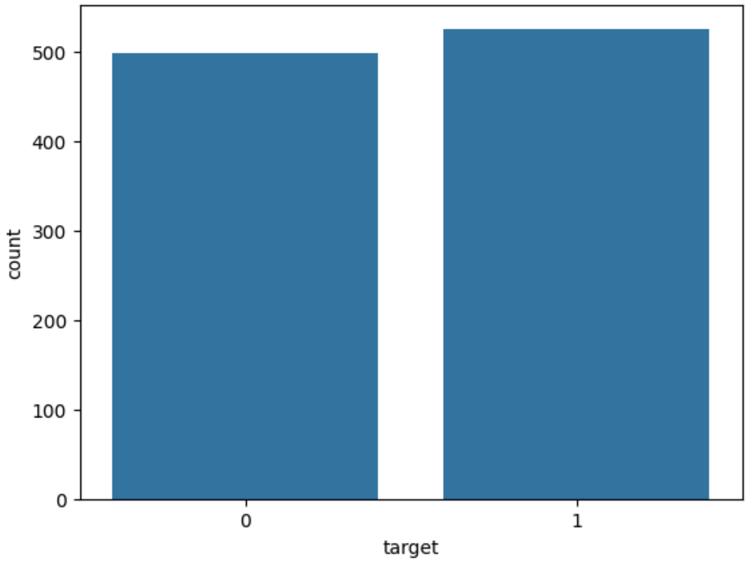


	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.1	-0.07	0.27	0.22	0.12	-0.13	-0.39	0.09	0.21	-0.17	0.27	0.07	-0.23
sex	-0.1	1	-0.04	-0.08	-0.2	0.03	-0.06	-0.05	0.14	0.08	-0.03	0.11	0.2	-0.28
cp	-0.07	-0.04	1	0.04	-0.08	0.08	0.04	0.31	-0.4	-0.17	0.13	-0.18	-0.16	0.43
trestbps	0.27	-0.08	0.04	1	0.13	0.18	-0.12	-0.04	0.06	0.19	-0.12	0.1	0.06	-0.14
chol	0.22	-0.2	-0.08	0.13	1	0.03	-0.15	-0.02	0.07	0.06	-0.01	0.07	0.1	-0.1
fbs	0.12	0.03	0.08	0.18	0.03	1	-0.1	-0.01	0.05	0.01	-0.06	0.14	-0.04	-0.04
restecg	-0.13	-0.06	0.04	-0.12	-0.15	-0.1	1	0.05	-0.07	-0.05	0.09	-0.08	-0.02	0.13
thalach	-0.39	-0.05	0.31	-0.04	-0.02	-0.01	0.05	1	-0.38	-0.35	0.4	-0.21	-0.1	0.42
exang	0.09	0.14	-0.4	0.06	0.07	0.05	-0.07	-0.38	1	0.31	-0.27	0.11	0.2	-0.44
oldpeak	0.21	0.08	-0.17	0.19	0.06	0.01	-0.05	-0.35	0.31	1	-0.58	0.22	0.2	-0.44
slope	-0.17	-0.03	0.13	-0.12	-0.01	-0.06	0.09	0.4	-0.27	-0.58	1	-0.07	-0.09	0.35
ca	0.27	0.11	-0.18	0.1	0.07	0.14	-0.08	-0.21	0.11	0.22	-0.07	1	0.15	-0.38
thal	0.07	0.2	-0.16	0.06	0.1	-0.04	-0.02	-0.1	0.2	0.2	-0.09	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.1	-0.04	0.13	0.42	-0.44	-0.44	0.35	-0.38	-0.34	1

```
Mis_features=['thal','ca','slope','exang','restecg','fbs','cp','sex']
DF[Mis_features] = DF[Mis_features].astype(object)
```

```
sns.countplot(data=DF,x='target')
```

```
<Axes: xlabel='target', ylabel='count'>
```



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

```
Dum_DF.head()
```

	age	trestbps	chol	thalach	oldpeak	target	thal_1	thal_2	thal_3	ca_1	...
0	52	125	212	168	1.0	0	False	False	True	False	...
1	53	140	203	155	3.1	0	False	False	True	False	...
2	70	145	174	125	2.6	0	False	False	True	False	...
3	61	148	203	161	0.0	0	False	False	True	True	...
4	62	138	294	106	1.9	0	False	True	False	False	...

5 rows × 23 columns

```
X = Dum_DF.drop(['target'], axis=1)
y = Dum_DF['target']
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train , y_test = train_test_split(X, y, test_size=0.2 , random_state=42)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV


hyper_parameters = {'batch_size':['auto',100], 'max_iter':[200,500],'hidden_layer_sizes':[5,(5,5,5)],'learning_rate_init': [0.05,0.01,0
gs  =GridSearchCV(MLPClassifier(),hyper_parameters,scoring='roc_auc',n_jobs=-1
                  ,return_train_score=False,verbose=0,cv=5)
clf =gs.fit(x_train, y_train)
print('The best combination is:')
print(clf.best_params_)

print('The best Accuracy is:')
print(clf.best_score_)
```

```
The best combination is:
{'batch_size': 'auto', 'hidden_layer_sizes': 5, 'learning_rate_init': 0.05, 'max_iter': 200}
The best Accuracy is:
0.9884341648051626
```

```
#Use random forest classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
forest = RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=1)
```

```
forest.fit(x_train, y_train)
```

```
▾                          RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=1)
```

```
model=forest
```

```
model.score(x_test,y_test)
```

```
0.9609756097560975
```

```
#XGB
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
model = XGBClassifier(n_jobs=5,learning_rate=0.005)
model.fit(x_train,y_train)
```

```
▾                                   XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.005, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=5,
              num_parallel_tree=None, random_state=None, ...)
```

```
predictions = model.predict(x_test)
```

```
model.score(x_test,y_test)
```

```
0.8292682926829268
```

```
import pandas as pd

# Dictionary containing model names and their accuracy
accuracy_data = {
    "Model": ["Random Forest", "MLP", "XG Boost"],
    "Accuracy (%)": [ 96.09, 98.84, 82.92]
}

# Create a DataFrame from the accuracy data
accuracy_df = pd.DataFrame(accuracy_data)


# Display the DataFrame
print(accuracy_df)
```

```
           Model  Accuracy (%)
0  Random Forest         96.09
1            MLP         98.84
2       XG Boost         82.92
```