heart-failure-prediction

Use the "Run" button to execute the code.

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="heart-failure-prediction")
```

```
[jovian] Updating notebook "btech60309-19/heart-failure-prediction" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/btech60309-19/heart-failure-
prediction
```

```
'https://jovian.ai/btech60309-19/heart-failure-prediction'
```

```
!pip install opendatasets
```

```
Collecting opendatasets
  Using cached opendatasets-0.1.20-py3-none-any.whl (14 kB)
Collecting kaggle
  Downloading kaggle-1.5.12.tar.gz (58 kB)
     |████████████████████████████████| 58 kB 4.6 MB/s eta 0:00:011
Requirement already satisfied: click in /opt/conda/lib/python3.9/site-packages (from
opendatasets) (8.0.1)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.9/site-packages (from
opendatasets) (4.62.3)
Requirement already satisfied: six>=1.10 in /opt/conda/lib/python3.9/site-packages (fron
kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /opt/conda/lib/python3.9/site-packages (from
kaggle->opendatasets) (2021.5.30)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.9/site-package
(from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /opt/conda/lib/python3.9/site-packages (from
kaggle->opendatasets) (2.26.0)
Collecting python-slugify
  Downloading python_slugify-5.0.2-py2.py3-none-any.whl (6.7 kB)
Requirement already satisfied: urllib3 in /opt/conda/lib/python3.9/site-packages (from
kaggle->opendatasets) (1.26.7)
Collecting text-unidecode>=1.3
  Downloading text_unidecode-1.3-py2.py3-none-any.whl (78 kB)
     |████████████████████████████████| 78 kB 10.2 MB/s eta 0:00:01
```

```
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/lib/python3.9/si
packages (from requests->kaggle->opendatasets) (2.0.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-packages
(from requests->kaggle->opendatasets) (3.1)
Building wheels for collected packages: kaggle
  Building wheel for kaggle (setup.py) ... done
  Created wheel for kaggle: filename=kaggle-1.5.12-py3-none-any.whl size=73051
sha256=b19c542fbd70481d0251e09501d6954175725b422dffb9ebc3a632aa55d57864
  Stored in directory:
/home/jovyan/.cache/pip/wheels/ac/b2/c3/fa4706d469b5879105991d1c8be9a3c2ef329ba9fe2ce50
Successfully built kaggle
Installing collected packages: text-unidecode, python-slugify, kaggle, opendatasets
Successfully installed kaggle-1.5.12 opendatasets-0.1.20 python-slugify-5.0.2 text-
unidecode-1.3
```

```
import opendatasets as od
```

```
od.download('https://www.kaggle.com/fedesoriano/heart-failure-prediction')
```

```
Please provide your Kaggle credentials to download this dataset. Learn more:
http://bit.ly/kaggle-creds
Your Kaggle username: swetsheersh
Your Kaggle Key: ········
Downloading heart-failure-prediction.zip to ./heart-failure-prediction

100%|██████████| 8.56k/8.56k [00:00<00:00, 5.85MB/s]
```

```
import os
import pandas as pd
os.listdir('./heart-failure-prediction')
```

```
['heart.csv']
```

```
heart=pd.read_csv('./heart-failure-prediction/heart.csv')
```

```
heart
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
0	40	M	ATA	140	289	0	Normal	172	N	0.0	
1	49	F	NAP	160	180	0	Normal	156	N	1.0	
2	37	M	ATA	130	283	0	ST	98	N	0.0	
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
4	54	M	NAP	150	195	0	Normal	122	N	0.0	
...	...	...	...	...	...	...	...	...	...	...	
913	45	M	TA	110	264	0	Normal	132	N	1.2	
914	68	M	ASY	144	193	1	Normal	141	N	3.4	
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	
916	57	F	ATA	130	236	0	LVH	174	N	0.0	
917	38	M	NAP	138	175	0	Normal	173	N	0.0	

918 rows × 12 columns

```
heart.isna().sum()
```

```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

```
heart.describe()
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

```
heart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

```
jovian.commit()
```

```
[jovian] Updating notebook "btech60309-19/heart-failure-prediction" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/btech60309-19/heart-failure-
prediction
```

```
'https://jovian.ai/btech60309-19/heart-failure-prediction'
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib
import os
%matplotlib inline

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 150)
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (10, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
!pip install plotly
```

```
Collecting plotly
  Using cached plotly-5.5.0-py2.py3-none-any.whl (26.5 MB)
Collecting tenacity>=6.2.0
  Downloading tenacity-8.0.1-py3-none-any.whl (24 kB)
Requirement already satisfied: six in /opt/conda/lib/python3.9/site-packages (from
plotly) (1.16.0)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.5.0 tenacity-8.0.1
```

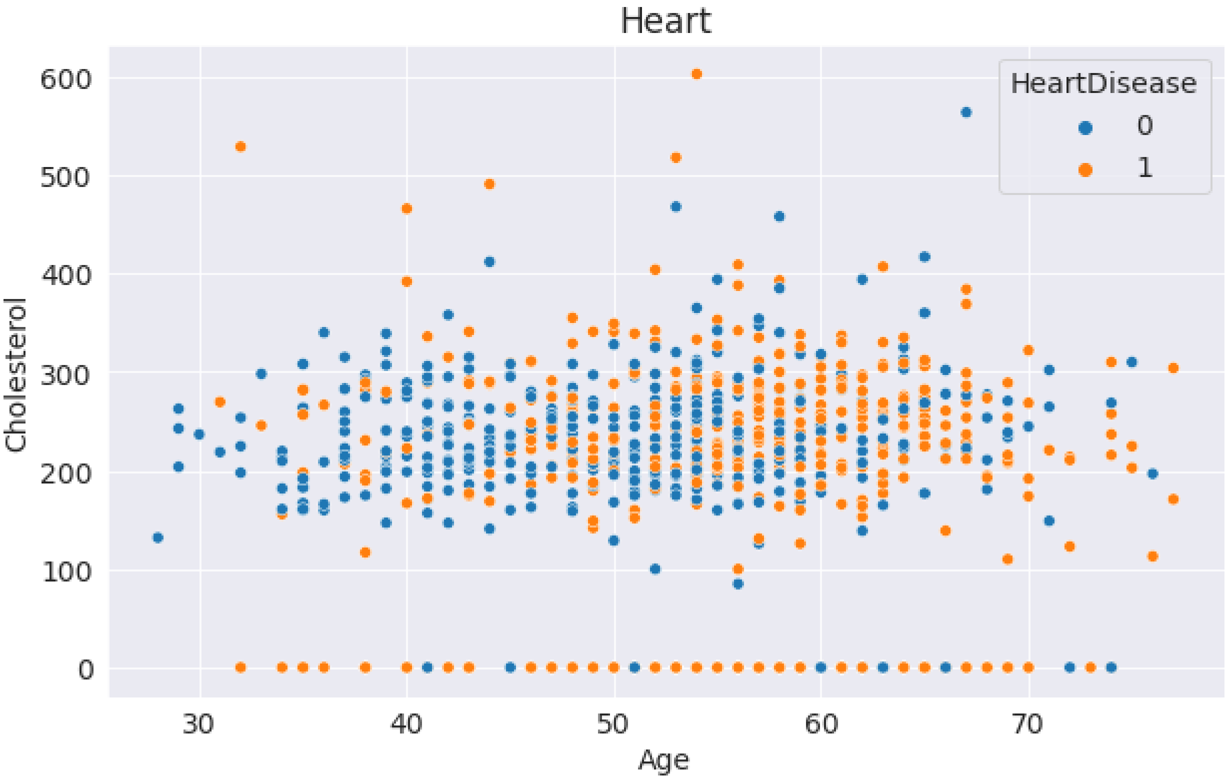
```
import plotly.express as px
```

```
sns.scatterplot(heart.Age,
                heart.Cholesterol,
                hue=heart.HeartDisease
                 );

# Chart title
plt.title("Heart");
```

```
/opt/conda/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
```



```
px.scatter(heart,x='Age',y='Cholesterol',color='HeartDisease')
```

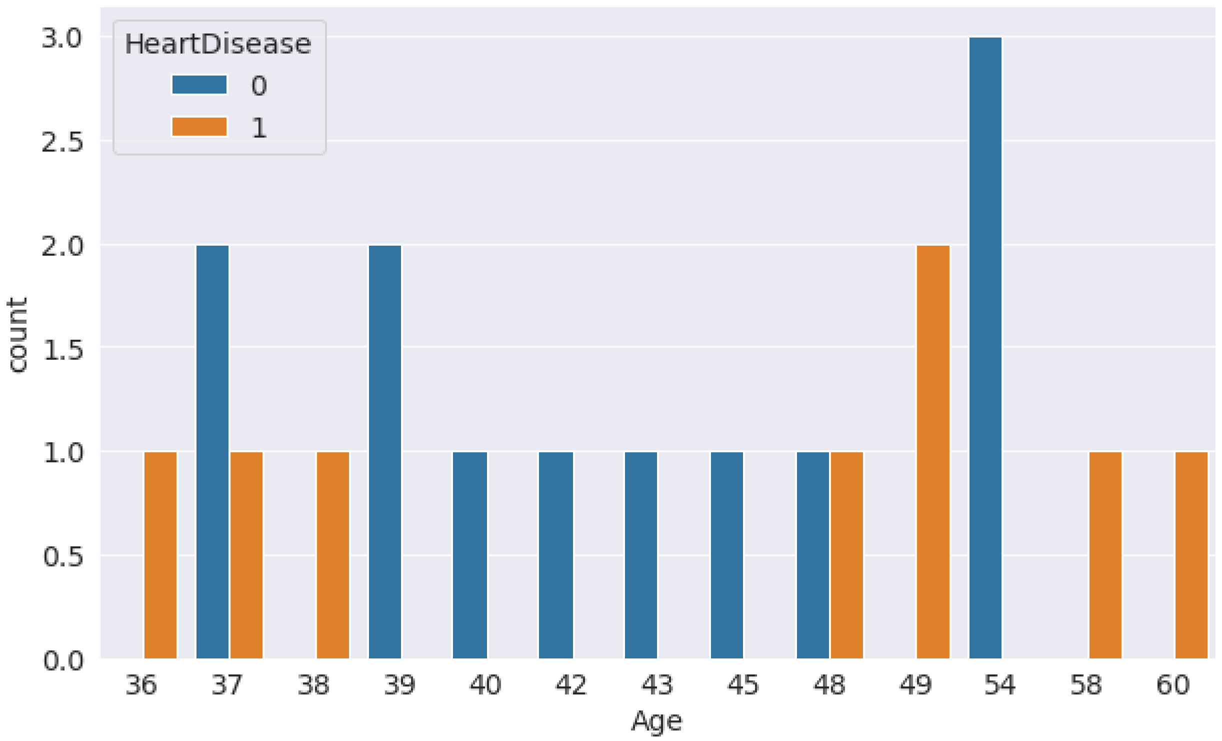
```
heart
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
0	40	M	ATA	140	289	0	Normal	172	N	0.0	
1	49	F	NAP	160	180	0	Normal	156	N	1.0	
2	37	M	ATA	130	283	0	ST	98	N	0.0	
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	
4	54	M	NAP	150	195	0	Normal	122	N	0.0	
...	...	...	...	...	...	...	...	...	...	...	
913	45	M	TA	110	264	0	Normal	132	N	1.2	
914	68	M	ASY	144	193	1	Normal	141	N	3.4	
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	
916	57	F	ATA	130	236	0	LVH	174	N	0.0	
917	38	M	NAP	138	175	0	Normal	173	N	0.0	

918 rows × 12 columns

```
sns.countplot(x=heart['Age'].head(20),hue=heart['HeartDisease'])
```

```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



```
heart.columns
```

```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

```
input_cols=['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope']
```

```
target='HeartDisease'
```

```
x_train=heart[input_cols]
target=heart[target]
```

```
x_train
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
0	40	M	ATA	140	289	0	Normal	172	N	0.0	
1	49	F	NAP	160	180	0	Normal	156	N	1.0	
2	37	M	ATA	130	283	0	ST	98	N	0.0	
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	
4	54	M	NAP	150	195	0	Normal	122	N	0.0	
...	...	...	...	...	...	...	...	...	...	...	
913	45	M	TA	110	264	0	Normal	132	N	1.2	
914	68	M	ASY	144	193	1	Normal	141	N	3.4	
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
916	57	F	ATA	130	236	0	LVH	174	N	0.0	
917	38	M	NAP	138	175	0	Normal	173	N	0.0	

918 rows × 11 columns

```
heart.nunique()
```

```
Age                50
Sex                 2
ChestPainType       4
RestingBP          67
Cholesterol       222
FastingBS           2
RestingECG          3
MaxHR             119
ExerciseAngina      2
Oldpeak            53
ST_Slope            3
HeartDisease        2
dtype: int64
```

```
numeric_cols=['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
cat_cols=['Sex', 'ChestPainType', 'FastingBS','RestingECG', 'ExerciseAngina', 'ST_Slope
```

```
!pip install sklearn
from sklearn.impute import SimpleImputer
```

```
Requirement already satisfied: sklearn in /opt/conda/lib/python3.9/site-packages (0.0)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.9/site-packages
(from sklearn) (1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.9/site-
packages (from scikit-learn->sklearn) (2.2.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.9/site-packages
(from scikit-learn->sklearn) (1.0.1)
Requirement already satisfied: numpy>=1.14.6 in /opt/conda/lib/python3.9/site-packages
(from scikit-learn->sklearn) (1.20.3)
Requirement already satisfied: scipy>=1.1.0 in /opt/conda/lib/python3.9/site-packages
(from scikit-learn->sklearn) (1.7.1)
```

```
imputer=SimpleImputer(strategy='mean')
```

```
imputer.fit(heart[numeric_cols])
```

```
SimpleImputer()
```

```
heart[numeric_cols]=imputer.transform(x_train[numeric_cols])
```

```
x_train[numeric_cols]
```

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak
0	40	140	289	172	0.0
1	49	160	180	156	1.0
2	37	130	283	98	0.0
3	48	138	214	108	1.5
4	54	150	195	122	0.0
...	...	...	...	...	...
913	45	110	264	132	1.2
914	68	144	193	141	3.4
915	57	130	131	115	1.2
916	57	130	236	174	0.0
917	38	138	175	173	0.0

918 rows × 5 columns

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder=OneHotEncoder(sparse=False, handle_unknown='ignore').fit(x_train[cat_cols])
```

```
encoded_cols=list(encoder.get_feature_names(cat_cols))
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and
will be removed in 1.2. Please use get_feature_names_out instead.
```

```
len(encoded_cols)
```

16

```
x_train[encoded_cols]=encoder.transform(x_train[cat_cols])
```

```
x_train
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
0	40	M	ATA	140	289	0	Normal	172	N	0.0	
1	49	F	NAP	160	180	0	Normal	156	N	1.0	

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
2	37	M	ATA	130	283	0	ST	98	N	0.0	
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	
4	54	M	NAP	150	195	0	Normal	122	N	0.0	
...	...	...	...	...	...	...	...	...	...	...	
913	45	M	TA	110	264	0	Normal	132	N	1.2	
914	68	M	ASY	144	193	1	Normal	141	N	3.4	
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	
916	57	F	ATA	130	236	0	LVH	174	N	0.0	
917	38	M	NAP	138	175	0	Normal	173	N	0.0	

918 rows × 27 columns

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler=MinMaxScaler().fit(x_train[numeric_cols])
```

```
x_train[numeric_cols].describe().loc[['min', 'max']]
```

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak
min	28.0	0.0	0.0	60.0	-2.6
max	77.0	200.0	603.0	202.0	6.2

```
x_train[numeric_cols]=scaler.transform(x_train[numeric_cols])
```

```
x_train[numeric_cols].describe().loc[['min', 'max']]
```

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak
min	0.0	0.0	0.0	0.0	0.0
max	1.0	1.0	1.0	1.0	1.0

```
x_train=x_train[numeric_cols + encoded_cols]
```

```
x_train
```

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
0	0.244898	0.70	0.479270	0.788732	0.295455	0.0	1.0	0.0	1.0
1	0.428571	0.80	0.298507	0.676056	0.409091	1.0	0.0	0.0	0.0
2	0.183673	0.65	0.469320	0.267606	0.295455	0.0	1.0	0.0	1.0
3	0.408163	0.69	0.354892	0.338028	0.465909	1.0	0.0	1.0	0.0
4	0.530612	0.75	0.323383	0.436620	0.295455	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
913	0.346939	0.55	0.437811	0.507042	0.431818	0.0	1.0	0.0	0.0
914	0.816327	0.72	0.320066	0.570423	0.681818	0.0	1.0	1.0	0.0
915	0.591837	0.65	0.217247	0.387324	0.431818	0.0	1.0	1.0	0.0
916	0.591837	0.65	0.391376	0.802817	0.295455	1.0	0.0	0.0	1.0
917	0.204082	0.69	0.290216	0.795775	0.295455	0.0	1.0	0.0	0.0

918 rows × 21 columns

```
jovian.commit()
```

```
[jovian] Updating notebook "btech60309-19/heart-failure-prediction" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/btech60309-19/heart-failure-
prediction
```

```
'https://jovian.ai/btech60309-19/heart-failure-prediction'
```

```
from sklearn.model_selection import train_test_split
```

```
train_df,val_df,train_target,val_target = train_test_split(x_train,target, test_size=0.
```

```
train_df
```

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
557	0.571429	0.685	0.344942	0.436620	0.500000	0.0	1.0	0.0	0.0
260	0.367347	0.700	0.456053	0.739437	0.295455	0.0	1.0	0.0	1.0
235	0.224490	0.600	0.331675	0.704225	0.409091	0.0	1.0	0.0	1.0
218	0.551020	0.700	0.325041	0.633803	0.295455	0.0	1.0	0.0	1.0
382	0.306122	0.575	0.000000	0.598592	0.522727	0.0	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...
106	0.408163	0.600	0.421227	0.352113	0.295455	1.0	0.0	1.0	0.0
270	0.346939	0.600	0.373134	0.563380	0.295455	0.0	1.0	1.0	0.0
860	0.653061	0.650	0.419569	0.591549	0.454545	0.0	1.0	1.0	0.0
435	0.653061	0.760	0.000000	0.408451	0.295455	0.0	1.0	1.0	0.0
102	0.244898	0.750	0.650083	0.492958	0.522727	1.0	0.0	1.0	0.0

826 rows × 21 columns

```
val_df
```

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
668	0.714286	0.700	0.323383	0.838028	0.295455	1.0	0.0	0.0	1.0
30	0.510204	0.725	0.859038	0.492958	0.295455	0.0	1.0	0.0	0.0

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
377	0.755102	0.800	0.000000	0.436620	0.431818	0.0	1.0	1.0	0.0
535	0.571429	0.650	0.000000	0.436620	0.409091	0.0	1.0	1.0	0.0
807	0.530612	0.540	0.512438	0.676056	0.295455	0.0	1.0	0.0	1.0
793	0.795918	0.625	0.421227	0.725352	0.318182	0.0	1.0	1.0	0.0
363	0.571429	0.600	0.000000	0.619718	0.295455	0.0	1.0	1.0	0.0
583	0.836735	0.710	0.449420	0.464789	0.329545	0.0	1.0	0.0	0.0
165	0.367347	0.700	0.451078	0.809859	0.522727	0.0	1.0	0.0	0.0
483	0.612245	0.600	0.000000	0.323944	0.465909	0.0	1.0	1.0	0.0
773	0.571429	0.600	0.320066	0.718310	0.511364	0.0	1.0	0.0	0.0
551	0.693878	0.600	0.364842	0.183099	0.295455	0.0	1.0	0.0	0.0
768	0.734694	0.650	0.502488	0.436620	0.522727	1.0	0.0	1.0	0.0
694	0.571429	0.600	0.391376	0.830986	0.386364	0.0	1.0	0.0	1.0
718	0.591837	0.825	0.479270	0.450704	0.409091	0.0	1.0	1.0	0.0
312	0.265306	0.625	0.000000	0.816901	0.477273	0.0	1.0	1.0	0.0
713	0.734694	0.700	0.519071	0.514085	0.318182	1.0	0.0	0.0	0.0
309	0.591837	0.475	0.000000	0.859155	0.375000	0.0	1.0	1.0	0.0
846	0.224490	0.590	0.363184	0.563380	0.431818	0.0	1.0	1.0	0.0
616	0.795918	0.575	0.935323	0.704225	0.477273	1.0	0.0	0.0	0.0
355	0.795918	0.725	0.000000	0.457746	0.295455	0.0	1.0	0.0	0.0
39	0.408163	0.750	0.376451	0.492958	0.409091	1.0	0.0	1.0	0.0
231	0.244898	0.650	0.466003	0.753521	0.295455	0.0	1.0	0.0	0.0
822	0.612245	0.525	0.398010	0.661972	0.363636	0.0	1.0	0.0	0.0
603	0.938776	0.775	0.514096	0.366197	0.465909	0.0	1.0	1.0	0.0
63	0.367347	0.600	0.459370	0.457746	0.409091	0.0	1.0	1.0	0.0
192	0.408163	0.650	0.406302	0.704225	0.295455	0.0	1.0	0.0	1.0
481	0.836735	0.700	0.000000	0.408451	0.579545	0.0	1.0	0.0	0.0
866	0.326531	0.650	0.363184	0.901408	0.295455	0.0	1.0	0.0	1.0
67	0.081633	0.550	0.373134	0.873239	0.295455	0.0	1.0	0.0	1.0
72	0.489796	0.600	0.301824	0.633803	0.295455	0.0	1.0	1.0	0.0
655	0.244898	0.760	0.369818	0.852113	0.295455	0.0	1.0	1.0	0.0
679	0.714286	0.725	0.386401	0.633803	0.556818	0.0	1.0	0.0	0.0
139	0.306122	0.750	0.409619	0.492958	0.522727	0.0	1.0	1.0	0.0
732	0.571429	1.000	0.477612	0.514085	0.750000	1.0	0.0	1.0	0.0
824	0.183673	0.650	0.414594	0.894366	0.693182	0.0	1.0	0.0	0.0
174	0.489796	0.700	0.441128	0.521127	0.522727	0.0	1.0	1.0	0.0
896	0.387755	0.650	0.419569	0.838028	0.295455	0.0	1.0	0.0	0.0
499	0.693878	0.675	0.492537	0.492958	0.409091	0.0	1.0	1.0	0.0
70	0.591837	0.700	0.439469	0.598592	0.409091	0.0	1.0	0.0	1.0
716	0.795918	0.600	0.393035	0.077465	0.409091	0.0	1.0	1.0	0.0

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
23	0.326531	0.750	0.477612	0.633803	0.636364	0.0	1.0	0.0	1.0
541	0.979592	0.520	0.187396	0.422535	0.693182	0.0	1.0	0.0	0.0
799	0.510204	0.650	0.407960	0.795775	0.295455	0.0	1.0	0.0	0.0
672	0.653061	0.600	0.295191	0.253521	0.295455	1.0	0.0	0.0	0.0
826	0.469388	0.625	0.406302	0.746479	0.568182	0.0	1.0	0.0	0.0
250	0.326531	0.675	0.814262	0.528169	0.295455	0.0	1.0	1.0	0.0
752	0.571429	0.625	0.412935	0.591549	0.431818	0.0	1.0	1.0	0.0
350	0.510204	0.600	0.000000	0.422535	0.295455	0.0	1.0	1.0	0.0
758	0.469388	0.625	0.353234	0.457746	0.454545	0.0	1.0	0.0	0.0
759	0.530612	0.960	0.469320	0.950704	0.295455	0.0	1.0	0.0	1.0
107	0.122449	0.750	0.354892	0.760563	0.295455	0.0	1.0	0.0	1.0
445	0.551020	0.680	0.378109	0.450704	0.477273	0.0	1.0	0.0	0.0
141	0.448980	0.700	0.565506	0.457746	0.579545	0.0	1.0	1.0	0.0
650	0.408163	0.650	0.424544	0.633803	0.295455	0.0	1.0	1.0	0.0
544	0.673469	0.700	0.494196	0.422535	0.295455	1.0	0.0	0.0	1.0
110	0.632653	0.650	0.311774	0.450704	0.409091	1.0	0.0	0.0	1.0
593	0.734694	0.650	0.427861	0.492958	0.295455	0.0	1.0	1.0	0.0
519	0.714286	0.480	0.505804	0.429577	0.409091	0.0	1.0	1.0	0.0
907	0.326531	0.600	0.280265	0.591549	0.613636	0.0	1.0	1.0	0.0
675	0.591837	0.750	0.208955	0.795775	0.318182	0.0	1.0	0.0	0.0
280	0.653061	0.600	0.407960	0.528169	0.295455	0.0	1.0	0.0	0.0
136	0.306122	0.600	0.356551	0.809859	0.295455	1.0	0.0	0.0	1.0
422	0.755102	0.750	0.391376	0.316901	0.295455	0.0	1.0	1.0	0.0
208	0.000000	0.650	0.218905	0.880282	0.295455	0.0	1.0	0.0	1.0
442	0.469388	0.640	0.000000	0.457746	0.431818	0.0	1.0	1.0	0.0
86	0.755102	0.850	0.436153	0.366197	0.522727	0.0	1.0	1.0	0.0
44	0.306122	0.600	0.290216	0.422535	0.409091	0.0	1.0	1.0	0.0
531	0.734694	0.715	0.507463	0.387324	0.500000	0.0	1.0	1.0	0.0
913	0.346939	0.550	0.437811	0.507042	0.431818	0.0	1.0	0.0	0.0
634	0.244898	0.700	0.330017	0.830986	0.454545	0.0	1.0	0.0	0.0
290	0.408163	0.550	0.349917	0.549296	0.295455	0.0	1.0	0.0	0.0
338	0.714286	0.700	0.000000	0.626761	0.522727	0.0	1.0	1.0	0.0
357	0.510204	0.600	0.000000	0.246479	0.295455	0.0	1.0	0.0	1.0
292	0.510204	0.650	0.301824	0.619718	0.295455	0.0	1.0	1.0	0.0
227	0.204082	0.460	0.194030	0.521127	0.579545	0.0	1.0	1.0	0.0
591	0.612245	0.500	0.353234	0.352113	0.295455	0.0	1.0	1.0	0.0
425	0.653061	0.800	0.442786	0.683099	0.352273	0.0	1.0	0.0	1.0
789	0.122449	0.590	0.301824	0.802817	0.295455	0.0	1.0	0.0	0.0
522	0.448980	0.720	0.578773	0.422535	0.409091	0.0	1.0	1.0	0.0

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA
861	0.755102	0.550	0.411277	0.690141	0.363636	0.0	1.0	1.0	0.0
352	0.571429	0.600	0.000000	0.281690	0.181818	0.0	1.0	1.0	0.0
493	0.469388	0.685	0.562189	0.471831	0.488636	0.0	1.0	0.0	0.0
60	0.428571	0.500	0.419569	0.802817	0.295455	0.0	1.0	0.0	1.0
598	0.551020	0.600	0.374793	0.471831	0.488636	0.0	1.0	1.0	0.0
722	0.653061	0.750	0.427861	0.683099	0.590909	1.0	0.0	1.0	0.0
426	0.571429	0.630	0.275290	0.563380	0.295455	0.0	1.0	0.0	1.0
468	0.693878	0.760	0.253731	0.260563	0.477273	0.0	1.0	1.0	0.0
66	0.346939	0.660	0.492537	0.591549	0.295455	1.0	0.0	1.0	0.0
332	0.204082	0.500	0.000000	0.838028	0.170455	0.0	1.0	0.0	0.0
375	0.918367	0.800	0.000000	0.429577	0.295455	1.0	0.0	0.0	0.0
381	0.448980	0.575	0.000000	0.422535	0.352273	0.0	1.0	1.0	0.0

```
train_target
```

```
557    1
260    0
235    0
218    0
382    1
      ..
106    0
270    0
860    1
435    0
102    1
Name: HeartDisease, Length: 826, dtype: int64
```

```
len(val_target)
```

```
92
```

```
jovian.commit()
```

```
[jovian] Updating notebook "btech60309-19/heart-failure-prediction" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/btech60309-19/heart-failure-
prediction
```

```
'https://jovian.ai/btech60309-19/heart-failure-prediction'
```

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

```
model=LogisticRegression(solver='liblinear')
```

```
model.fit(train_df,train_target)
```

```
LogisticRegression(solver='liblinear')
```

```
pred=model.predict(train_df)
```

```
print(model.coef_.tolist())
```

```
[[0.6636763929634307, -0.03624671646388043, -1.7420098735456697, -0.5594161008880127,
1.8043899424890677, -0.6827890695652646, 0.6041332315396141, 1.0777271131154214,
-0.7442328782759892, -0.4235185174417452, 0.011368444576650361, -0.6520137780667163,
0.5733579400410825, 0.16209948518749306, -0.04740677158267605, -0.19334855163046916,
-0.5489776117174245, 0.4703217736917812, -0.021606613743019556, 1.1714206628929604,
-1.2284698871756172]]
```

```
print(model.intercept_)
```

```
[-0.07865584]
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
accuracy_score(pred,train_target)
```

```
0.8619854721549637
```

```
pred1=model.predict(val_df)
```

```
accuracy_score(pred1,val_target)
```

```
0.8804347826086957
```

```
val_probs = model.predict_proba(val_df)
val_probs
```

```
array([[0.98258563, 0.01741437],
       [0.71131789, 0.28868211],
       [0.02626967, 0.97373033],
       [0.02647756, 0.97352244],
       [0.95692554, 0.04307446],
       [0.06346221, 0.93653779],
       [0.1273922 , 0.8726078 ],
       [0.88934738, 0.11065262],
       [0.16686088, 0.83313912],
       [0.06874992, 0.93125008],
       [0.27087716, 0.72912284],
       [0.8734168 , 0.1265832 ],
       [0.37122695, 0.62877305],
       [0.94200347, 0.05799653],
       [0.04856241, 0.95143759],
       [0.28713702, 0.71286298],
       [0.97848585, 0.02151415],
       [0.10619948, 0.89380052],
       [0.18464867, 0.81535133],
       [0.84646701, 0.15353299],
       [0.19025218, 0.80974782],
       [0.21266814, 0.78733186],
       [0.94960411, 0.05039589],
       [0.22555601, 0.77444399],
       [0.15607651, 0.84392349],
       [0.07947952, 0.92052048],
       [0.95333256, 0.04666744],
       [0.22122111, 0.77877889],
       [0.94768725, 0.05231275],
       [0.96327122, 0.03672878],
       [0.18507726, 0.81492274],
       [0.79025684, 0.20974316],
       [0.24480067, 0.75519933],
       [0.06444165, 0.93555835],
       [0.11133741, 0.88866259],
       [0.73913651, 0.26086349],
       [0.06131543, 0.93868457],
       [0.94309103, 0.05690897],
       [0.07004774, 0.92995226],
       [0.35829366, 0.64170634],
       [0.11477531, 0.88522469],
       [0.2942615 , 0.7057385 ],
       [0.41079132, 0.58920868],
       [0.77690247, 0.22309753],
       [0.89545555, 0.10454445],
       [0.16130174, 0.83869826],
       [0.36873032, 0.63126968],
       [0.01685843, 0.98314157],
       [0.03340082, 0.96659918],
       [0.61623063, 0.38376937],
       [0.95189385, 0.04810615],
       [0.9643624 , 0.0356376 ],
       [0.23329919, 0.76670081],
       [0.07750585, 0.92249415],
       [0.21979906, 0.78020094],
       [0.86766088, 0.13233912],
       [0.77615956, 0.22384044],
       [0.05034723, 0.94965277],
       [0.48173786, 0.51826214],
       [0.13757798, 0.86242202],
       [0.73477417, 0.26522583],
       [0.9026262 , 0.0973738 ],
       [0.98888318, 0.01111682],
       [0.02246735, 0.97753265],
       [0.94533634, 0.05466366],
       [0.01170023, 0.98829977],
       [0.01448862, 0.98551138],
       [0.06161976, 0.93838024],
       [0.02007199, 0.97992801],
       [0.40058276, 0.59941724],
       [0.73167634, 0.26832366],
       [0.92467883, 0.07532117],
       [0.16763526, 0.83236474],
       [0.39738801, 0.60261199],
       [0.71052999, 0.28947001],
       [0.04391905, 0.95608095],
       [0.71312197, 0.28687803],
       [0.3384332 , 0.6615668 ],
       [0.89210346, 0.10789654],
       [0.46989142, 0.53010858],
       [0.65233802, 0.34766198],
       [0.15000755, 0.84999245],
       [0.27500507, 0.72499493],
       [0.9559003 , 0.0440997 ],
       [0.13344864, 0.86655136],
       [0.31135251, 0.68864749],
       [0.93973034, 0.06026966],
       [0.33046627, 0.66953373],
       [0.93150657, 0.06849343],
       [0.91825184, 0.08174816],
       [0.94952431, 0.05047569],
       [0.03834889, 0.96165111]])
```

```
jovian.commit()
```

```
[jovian] Updating notebook "btech60309-19/heart-failure-prediction" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/btech60309-19/heart-failure-
prediction
```

```
'https://jovian.ai/btech60309-19/heart-failure-prediction'
```

```
confusion_matrix(pred1, val_target, normalize='true')
```

```
array([[0.86486486, 0.13513514],
       [0.10909091, 0.89090909]])
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree=DecisionTreeClassifier(random_state=42)
```

```
tree.fit(train_df, train_target)
```

```
DecisionTreeClassifier(random_state=42)
```

```
train_preds1 = tree.predict(train_df)
```

```
tree.score(train_df,train_target)
```

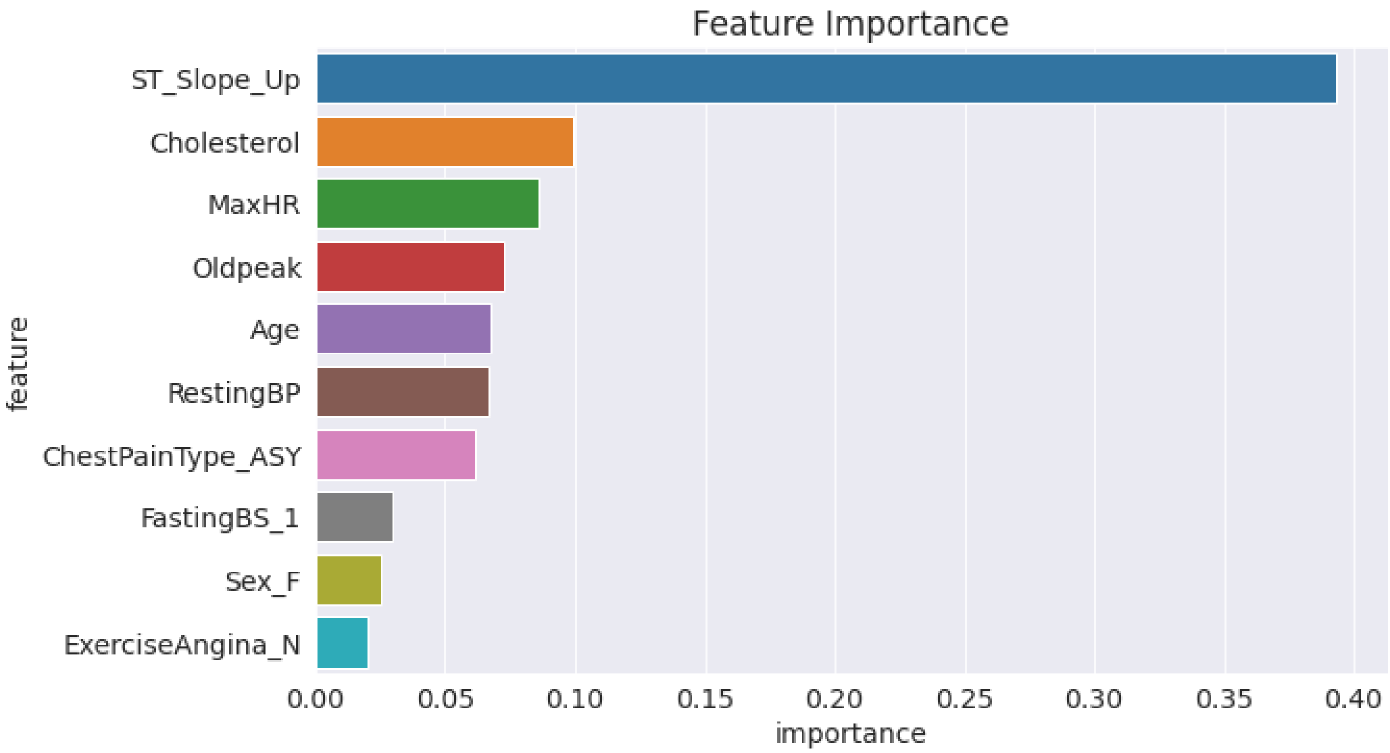
```
1.0
```

```
tree.score(val_df,val_target)
```

```
0.782608695652174
```

```
importance_df = pd.DataFrame({
    'feature':train_df.columns,
    'importance': tree.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



```
model = DecisionTreeClassifier(max_depth=8, random_state=42)
```

```
model.fit(train_df, train_target)
```

```
DecisionTreeClassifier(max_depth=8, random_state=42)
```

```
model.score(train_df, train_target)
```

```
0.9539951573849879
```

```python
model.score(val_df,val_target)
```

```
0.8152173913043478
```

```python
from sklearn.ensemble import RandomForestClassifier
```

```python
model = RandomForestClassifier(n_jobs=-1, random_state=42)
```

```python
model.fit(train_df, train_target)
```

```
RandomForestClassifier(n_jobs=-1, random_state=42)
```

```python
model.score(train_df, train_target)
```

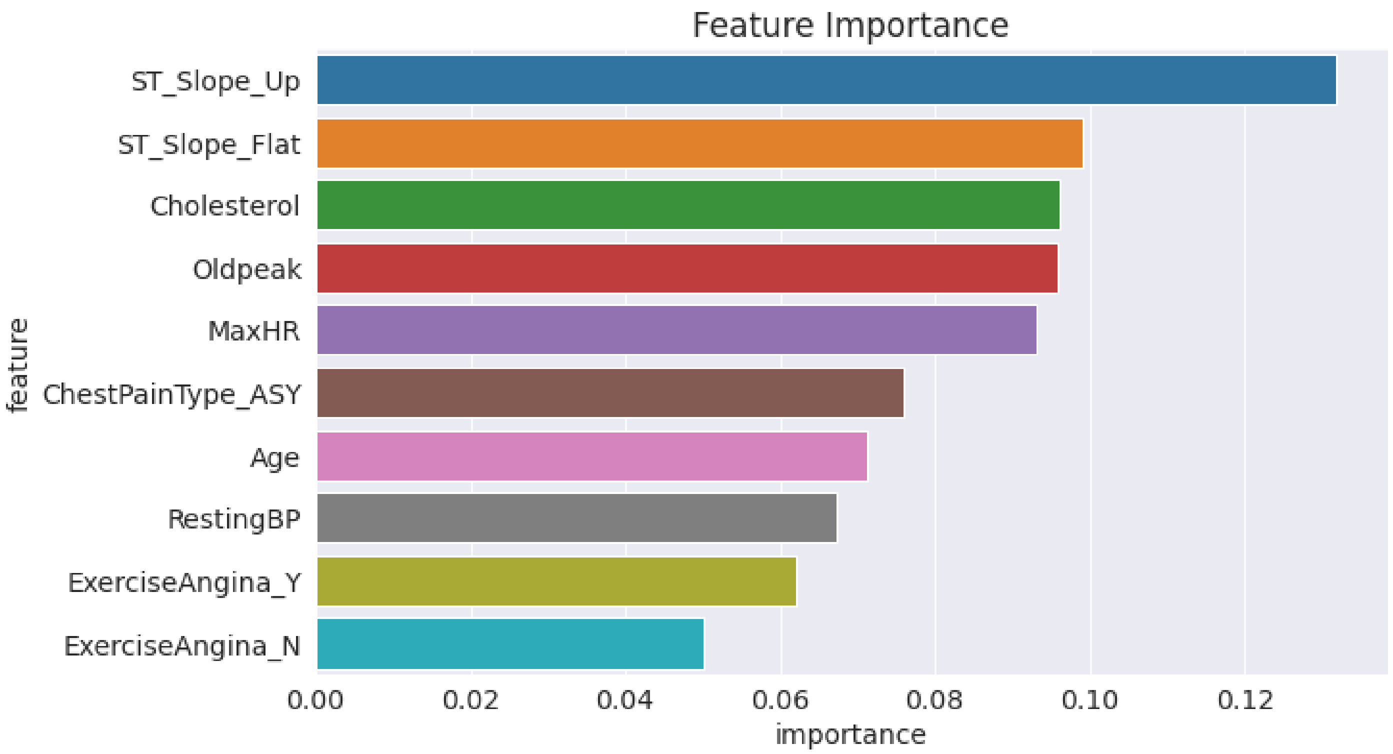
```
1.0
```

```python
model.score(val_df,val_target)
```

```
0.9021739130434783
```

```python
importance_df = pd.DataFrame({
    'feature': train_df.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```python
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



```
model = RandomForestClassifier(random_state=42, n_jobs=-1, n_estimators=90)
```

```
model.fit(train_df, train_target)
```

```
RandomForestClassifier(n_estimators=90, n_jobs=-1, random_state=42)
```

```
model.score(train_df, train_target)
```

```
1.0
```

```
model.score(val_df,val_target)
```

```
0.9130434782608695
```

```
def test_params(**params):
    model = RandomForestClassifier(random_state=42, n_jobs=-1,n_estimators=90, **params
    return model.score(train_df, train_target), model.score(val_df, val_target)
```

```
test_params(max_depth=5)
```

```
(0.9043583535108959, 0.8913043478260869)
```

```
test_params(max_depth=26)
```

```
(1.0, 0.9130434782608695)
```

```
test_params(max_features='log2')
```

```
(1.0, 0.9130434782608695)
```

```
test_params(bootstrap=False)
```

```
(1.0, 0.9130434782608695)
```

```
!pip install xgboost
from xgboost import XGBClassifier
```

```
Collecting xgboost
  Downloading xgboost-1.5.1-py3-none-manylinux2014_x86_64.whl (173.5 MB)
     |████████████████████████████████| 173.5 MB 8.9 kB/s  eta 0:00:01     |
██████████████                  | 77.3 MB 88.0 MB/s eta 0:00:02
Requirement already satisfied: scipy in /opt/conda/lib/python3.9/site-packages (from
xgboost) (1.7.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-packages (from
xgboost) (1.20.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.5.1
```

```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=1000,max_depth=25,learning
```

```
model.fit(train_df, train_target)
```

```
/opt/conda/lib/python3.9/site-packages/xgboost/sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[16:29:27] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.01, max_delta_step=0,
              max_depth=25, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=1000, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
model.score(train_df, train_target)
```

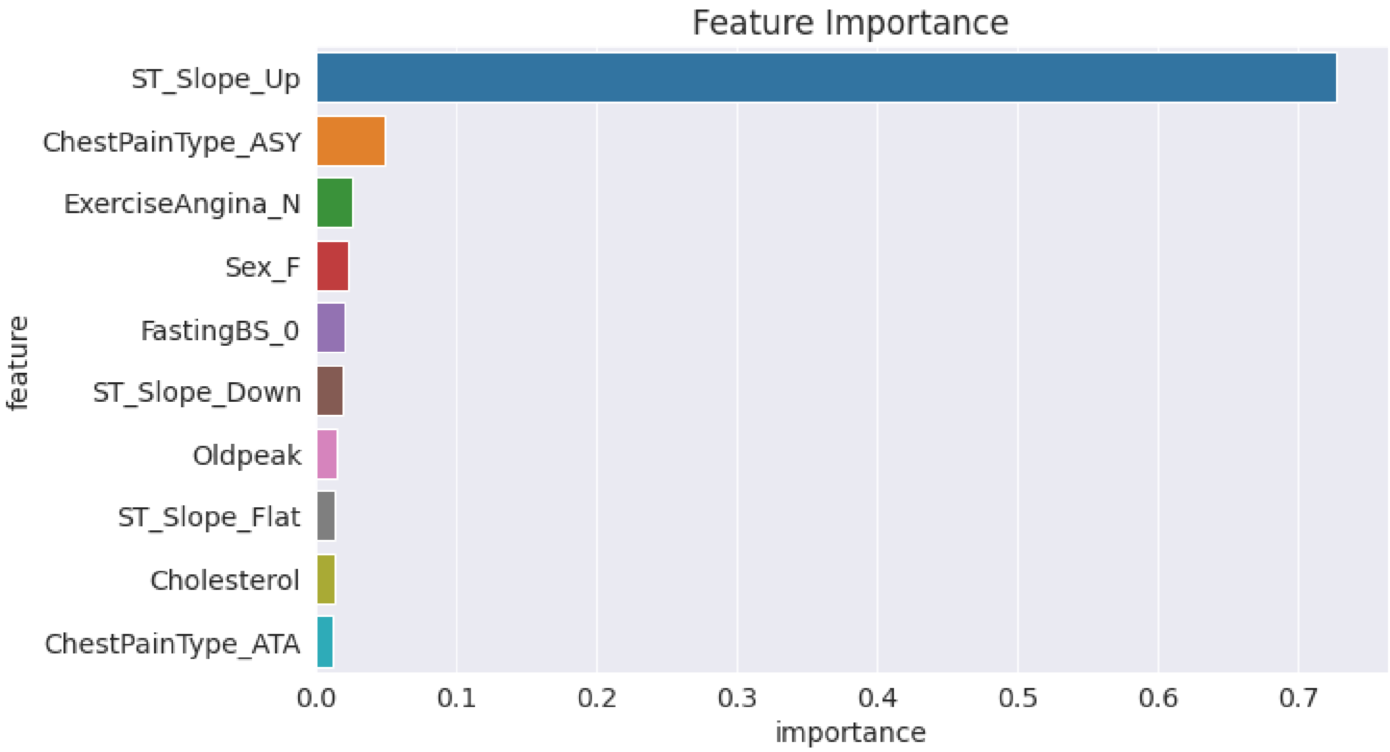
```
1.0
```

```
model.score(val_df,val_target)
```

```
0.9130434782608695
```

```
importance_df = pd.DataFrame({
    'feature': train_df.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



```
import joblib
```

```
heart_model = {
    'model': model,
    'imputer': imputer,
    'scaler': scaler,
    'encoder': encoder,
    'input_cols': input_cols,
    'target_col': target,
    'numeric_cols': numeric_cols,
    'categorical_cols': cat_cols,
    'encoded_cols': encoded_cols
}
```

```
joblib.dump(heart_model, 'heart_model.joblib')
```

```
['heart_model.joblib']
```

```
titanic2 = joblib.load('heart_model.joblib')
```

```
jovian.commit()
```

```
[jovian] Updating notebook "btech60309-19/heart-failure-prediction" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/btech60309-19/heart-failure-prediction
```

```
'https://jovian.ai/btech60309-19/heart-failure-prediction'
```

```
confusion_matrix(model.predict(val_df), val_target, normalize='true')
```

```
array([[0.89473684, 0.10526316],
       [0.07407407, 0.92592593]])
```

```
jovian.commit()
```