In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
df = pd.read_csv("heart.csv")
pd.concat([df.head(10), df.tail(10)])
```

Out [ ]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
5	39	M	NAP	120	339	0	Normal	170	N	0.0	Up	0
6	45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
7	54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
8	37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
9	48	F	ATA	120	284	0	Normal	120	N	0.0	Up	0
908	63	M	ASY	140	187	0	LVH	144	Y	4.0	Up	1
909	63	F	ASY	124	197	0	Normal	136	Y	0.0	Flat	1
910	41	M	ATA	120	157	0	Normal	182	N	0.0	Up	0
911	59	M	ASY	164	176	1	LVH	90	N	1.0	Flat	1
912	57	F	ASY	140	241	0	Normal	123	Y	0.2	Flat	1
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

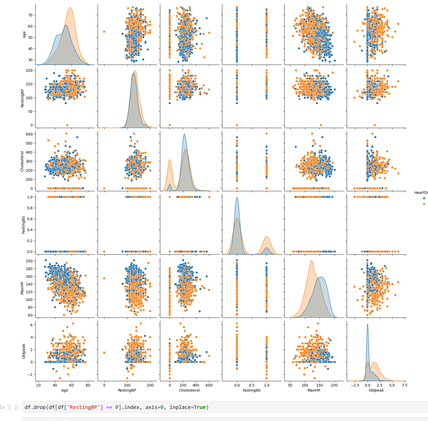
In [ ]:
```
df.describe(include="all")
```

Out [ ]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
count	918.000000	918	918	918.000000	918.000000	918.000000	918	918.000000	918	918.000000	918	918.000000
unique	NaN	2	4	NaN	NaN	NaN	3	NaN	2	NaN	3	NaN
top	NaN	M	ASY	NaN	NaN	NaN	Normal	NaN	N	NaN	Flat	NaN
freq	NaN	725	496	NaN	NaN	NaN	552	NaN	547	NaN	460	NaN
mean	53.510893	NaN	NaN	132.396514	198.799564	0.233115	NaN	136.809368	NaN	0.887364	NaN	0.553
std	9.432617	NaN	NaN	18.514154	109.384145	0.423046	NaN	25.460334	NaN	1.066570	NaN	0.497
min	28.000000	NaN	NaN	0.000000	0.000000	0.000000	NaN	60.000000	NaN	-2.600000	NaN	0.000
25%	47.000000	NaN	NaN	120.000000	173.250000	0.000000	NaN	120.000000	NaN	0.000000	NaN	0.000
50%	54.000000	NaN	NaN	130.000000	223.000000	0.000000	NaN	138.000000	NaN	0.600000	NaN	1.000
75%	60.000000	NaN	NaN	140.000000	267.000000	0.000000	NaN	156.000000	NaN	1.500000	NaN	1.000
max	77.000000	NaN	NaN	200.000000	603.000000	1.000000	NaN	202.000000	NaN	6.200000	NaN	1.000

In [ ]:
```
import seaborn as sns

sns.pairplot(df, hue='HeartDisease')
```

Out [ ]:
```
<seaborn.axisgrid.PairGrid at 0x14b23e7e510>
```



In [ ]:
```
df.drop(df[df['RestingBP'] == 0].index, axis=0, inplace=True)
```

In [ ]:
```
y = df['HeartDisease']
X = df.drop('HeartDisease', axis=1)
```

In [ ]:
```
from sklearn.preprocessing import OneHotEncoder

categorials = X.select_dtypes('object').columns
integer = X.select_dtypes(['int', 'float']).columns

encoder = OneHotEncoder(handle_unknown='ignore')

X_encoded = encoder.fit_transform(X[categorials])
X_encoded = pd.DataFrame.sparse.from_spmatrix(X_encoded, columns=encoder.get_feature_names_out(), index=X.index)

X = pd.concat([X[integer],X_encoded], axis=1)
pd.concat([X.head(10), X.tail(10)])
```

Out [ ]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA	ChestPainType_NAP	ChestPainTyp
0	40	140	289	0	172	0.0	0	1.0	0	1.0	0	
1	49	160	180	0	156	1.0	1.0	0	0	0	1.0	
2	37	130	283	0	98	0.0	0	1.0	0	1.0	0	
3	48	138	214	0	108	1.5	1.0	0	1.0	0	0	
4	54	150	195	0	122	0.0	0	1.0	0	0	1.0	
5	39	120	339	0	170	0.0	0	1.0	0	0	1.0	
6	45	130	237	0	170	0.0	1.0	0	0	1.0	0	
7	54	110	208	0	142	0.0	0	1.0	0	1.0	0	
8	37	140	207	0	130	1.5	0	1.0	1.0	0	0	
9	48	120	284	0	120	0.0	1.0	0	0	1.0	0	
908	63	140	187	0	144	4.0	0	1.0	1.0	0	0	
909	63	124	197	0	136	0.0	1.0	0	1.0	0	0	
910	41	120	157	0	182	0.0	0	1.0	0	1.0	0	
911	59	164	176	1	90	1.0	0	1.0	1.0	0	0	
912	57	140	241	0	123	0.2	1.0	0	1.0	0	0	
913	45	110	264	0	132	1.2	0	1.0	1.0	0	0	
914	68	144	193	1	141	3.4	0	1.0	1.0	0	0	
915	57	130	131	0	115	1.2	0	1.0	1.0	0	0	
916	57	130	236	0	174	0.0	1.0	0	0	1.0	0	
917	38	138	175	0	173	0.0	0	1.0	0	0	1.0	

In [ ]:
```
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)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out [ ]:
```
((733, 20), (184, 20), (733,), (184,))
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

clft = DecisionTreeClassifier(random_state=42)
clft.fit(X_train, y_train)
report = classification_report(y_test, clft.predict(X_test))
for line in str(report).split('\n'):
    print(line)
```

```
              precision    recall  f1-score   support

           0       0.70      0.86      0.77        72
           1       0.89      0.76      0.82       112

    accuracy                           0.80       184
   macro avg       0.80      0.81      0.80       184
weighted avg       0.82      0.80      0.80       184

```

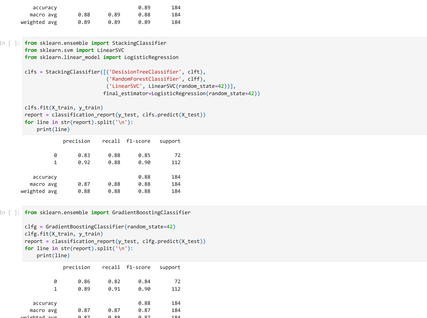
```
x:\_Netology\DS_ModelsParametersModules\env\Lib\site-packages\sklearn\utils\validation.py:877: UserWarning: pandas.DataFrame with sparse co
lumns found.It will be converted to a dense numpy array.
  warnings.warn(
x:\_Netology\DS_ModelsParametersModules\env\Lib\site-packages\sklearn\utils\validation.py:877: UserWarning: pandas.DataFrame with sparse co
lumns found.It will be converted to a dense numpy array.
  warnings.warn(
```

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

clff = RandomForestClassifier(random_state=42)
clff.fit(X_train, y_train)
report = classification_report(y_test, clff.predict(X_test))
for line in str(report).split('\n'):
    print(line)
```

```
x:\_Netology\DS_ModelsParametersModules\env\Lib\site-packages\sklearn\utils\validation.py:877: UserWarning: pandas.DataFrame with sparse co
lumns found.It will be converted to a dense numpy array.
  warnings.warn(
              precision    recall  f1-score   support

           0       0.85      0.86      0.86        72
           1       0.91      0.90      0.91       112

    accuracy                           0.89       184
   macro avg       0.88      0.88      0.88       184
weighted avg       0.89      0.89      0.89       184

```

```
x:\_Netology\DS_ModelsParametersModules\env\Lib\site-packages\sklearn\utils\validation.py:877: UserWarning: pandas.DataFrame with sparse co
lumns found.It will be converted to a dense numpy array.
  warnings.warn(
```

In [ ]:
```
plt.figure(figsize=(14, 6))
plt.barh(*zip(*sorted(zip(X.columns, clff.feature_importances_), key=lambda x: x[1], reverse=True)))
plt.show()
```



In [ ]:
```
import warnings

warnings.filterwarnings('ignore', message='pandas.DataFrame with sparse columns found.*')
```

In [ ]:
```
from sklearn.ensemble import BaggingClassifier

clfb = BaggingClassifier(clft, random_state=42)
clfb.fit(X_train, y_train)
report = classification_report(y_test, clfb.predict(X_test))
for line in str(report).split('\n'):
    print(line)
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        72
           1       0.93      0.88      0.90       112

    accuracy                           0.89       184
   macro avg       0.88      0.89      0.88       184
weighted avg       0.89      0.89      0.89       184

```

In [ ]:
```
from sklearn.ensemble import StackingClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression

clfs = StackingClassifier([('DecisionTreeClassifier', clft),
                           ('RandomForestClassifier', clff),
                           ('LinearSVC', LinearSVC(random_state=42))],
                          final_estimator=LogisticRegression(random_state=42))

clfs.fit(X_train, y_train)
report = classification_report(y_test, clfs.predict(X_test))
for line in str(report).split('\n'):
    print(line)
```

```
              precision    recall  f1-score   support

           0       0.83      0.88      0.85        72
           1       0.92      0.88      0.90       112

    accuracy                           0.88       184
   macro avg       0.87      0.88      0.88       184
weighted avg       0.88      0.88      0.88       184

```

In [ ]:
```
from sklearn.ensemble import GradientBoostingClassifier

clfg = GradientBoostingClassifier(random_state=42)
clfg.fit(X_train, y_train)
report = classification_report(y_test, clfg.predict(X_test))
for line in str(report).split('\n'):
    print(line)
```

```
              precision    recall  f1-score   support

           0       0.86      0.82      0.84        72
           1       0.89      0.91      0.90       112

    accuracy                           0.88       184
   macro avg       0.87      0.87      0.87       184
weighted avg       0.87      0.88      0.87       184

```

In [ ]:
```
from catboost import CatBoostClassifier

clfc = CatBoostClassifier(random_state=42, verbose=0)
clfc.fit(X_train, y_train)
report = classification_report(y_test, clfc.predict(X_test))
for line in str(report).split('\n'):
    print(line)
```

```
              precision    recall  f1-score   support

           0       0.84      0.86      0.85        72
           1       0.91      0.89      0.90       112

    accuracy                           0.88       184
   macro avg       0.87      0.88      0.88       184
weighted avg       0.88      0.88      0.88       184

```