**SAMPLE CODE:**

import pandas as pd

import matplotlib.pyplot as plt

from functools import reduce

dfs = pd.read\_csv('heart.csv')

dfs

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target

0 52 1 0 125 212 0 1 168 0 1.0 2 2 3 0

1 53 1 0 140 203 1 0 155 1 3.1 0 0 3 0

2 70 1 0 145 174 0 1 125 1 2.6 0 0 3 0

3 61 1 0 148 203 0 1 161 0 0.0 2 1 3 0

4 62 0 0 138 294 1 1 106 0 1.9 1 3 2 0

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

1020 59 1 1 140 221 0 1 164 1 0.0 2 0 2 1

1021 60 1 0 125 258 0 0 141 1 2.8 1 1 3 0

1022 47 1 0 110 275 0 0 118 1 1.0 1 1 2 0

1023 50 0 0 110 254 0 0 159 0 0.0 2 0 2 1

1024 54 1 0 120 188 0 1 113 0 1.4 1 1 3 0

1025 rows × 14 columns

1.age = age 2.sex = sex 3.chest pain type (4 values) = cp 6.resting blood pressure = trestbps 10.serum cholestoral in mg/dl =chol 4.fasting blood sugar > 120 mg/dl =fbs 11.resting electrocardiographic results (values 0,1,2) =restecg 9.maximum heart rate achieved =thalach 5.exercise induced angina =exang 8.ST depression induced by exercise relative to rest =oldpeak 7.the slope of the peak exercise ST segment =slope

dfs.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

dfs.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

# age

dfs['age'].unique()

array([52, 53, 70, 61, 62, 58, 55, 46, 54, 71, 43, 34, 51, 50, 60, 67, 45,

63, 42, 44, 56, 57, 59, 64, 65, 41, 66, 38, 49, 48, 29, 37, 47, 68,

76, 40, 39, 77, 69, 35, 74], dtype=int64)

import seaborn as sns

plt.style.use('seaborn')

dfs['sex'].hist(bins=20)

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\1338920274.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0\_8-<style>'. Alternatively, directly use the seaborn API instead.

plt.style.use('seaborn')

<Axes: >

0= Female 1= male

cp = chest pain type

dfs['cp'].unique()

array([0, 1, 2, 3], dtype=int64)

plt.style.use('seaborn')

dfs['cp'].hist(bins=20)

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\712749592.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0\_8-<style>'. Alternatively, directly use the seaborn API instead.

plt.style.use('seaborn')

<Axes: >

0 = Typical Agina 1 = Atypical Agina 2 = Non-aginal pain 3= Asymptomatic

dfs

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target

0 52 1 0 125 212 0 1 168 0 1.0 2 2 3 0

1 53 1 0 140 203 1 0 155 1 3.1 0 0 3 0

2 70 1 0 145 174 0 1 125 1 2.6 0 0 3 0

3 61 1 0 148 203 0 1 161 0 0.0 2 1 3 0

4 62 0 0 138 294 1 1 106 0 1.9 1 3 2 0

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

1020 59 1 1 140 221 0 1 164 1 0.0 2 0 2 1

1021 60 1 0 125 258 0 0 141 1 2.8 1 1 3 0

1022 47 1 0 110 275 0 0 118 1 1.0 1 1 2 0

1023 50 0 0 110 254 0 0 159 0 0.0 2 0 2 1

1024 54 1 0 120 188 0 1 113 0 1.4 1 1 3 0

1025 rows × 14 columns

dfs['restecg'].unique()

array([1, 0, 2], dtype=int64)

plt.style.use('seaborn')

dfs['restecg'].hist(bins=20)

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\3929014074.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0\_8-<style>'. Alternatively, directly use the seaborn API instead.

plt.style.use('seaborn')

<Axes: >

0=Normal 1=ST 2=LVH

dfs['exang'].unique()

array([0, 1], dtype=int64)

plt.style.use('seaborn')

dfs['exang'].hist(bins=20)

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\3683925537.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0\_8-<style>'. Alternatively, directly use the seaborn API instead.

plt.style.use('seaborn')

<Axes: >

0= no 1 = yes

dfs['slope'].unique()

array([2, 0, 1], dtype=int64)

plt.style.use('seaborn')

dfs['slope'].hist(bins=20)

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\1635372270.py:1: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0\_8-<style>'. Alternatively, directly use the seaborn API instead.

plt.style.use('seaborn')

<Axes: >

0=Up 1= Flat 2=Down

dfs

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target

0 52 1 0 125 212 0 1 168 0 1.0 2 2 3 0

1 53 1 0 140 203 1 0 155 1 3.1 0 0 3 0

2 70 1 0 145 174 0 1 125 1 2.6 0 0 3 0

3 61 1 0 148 203 0 1 161 0 0.0 2 1 3 0

4 62 0 0 138 294 1 1 106 0 1.9 1 3 2 0

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

1020 59 1 1 140 221 0 1 164 1 0.0 2 0 2 1

1021 60 1 0 125 258 0 0 141 1 2.8 1 1 3 0

1022 47 1 0 110 275 0 0 118 1 1.0 1 1 2 0

1023 50 0 0 110 254 0 0 159 0 0.0 2 0 2 1

1024 54 1 0 120 188 0 1 113 0 1.4 1 1 3 0

1025 rows × 14 columns

sns.distplot(dfs["trestbps"])

dfs['trestbps'].skew()

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\3593152484.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(dfs["trestbps"])

0.739768226050074

sns.distplot(dfs["chol"])

dfs['chol'].skew()

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\3484324216.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(dfs["chol"])

1.0740727783354815

sns.distplot(dfs["thalach"])

dfs['thalach'].skew()

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\3966685782.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(dfs["thalach"])

-0.5137771771417209

sns.distplot(dfs["fbs"])

dfs['fbs'].skew()

C:\Users\JP\AppData\Local\Temp\ipykernel\_19132\1544357994.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(dfs["fbs"])

1.9713385276901942

dfs

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target

0 52 1 0 125 212 0 1 168 0 1.0 2 2 3 0

1 53 1 0 140 203 1 0 155 1 3.1 0 0 3 0

2 70 1 0 145 174 0 1 125 1 2.6 0 0 3 0

3 61 1 0 148 203 0 1 161 0 0.0 2 1 3 0

4 62 0 0 138 294 1 1 106 0 1.9 1 3 2 0

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

1020 59 1 1 140 221 0 1 164 1 0.0 2 0 2 1

1021 60 1 0 125 258 0 0 141 1 2.8 1 1 3 0

1022 47 1 0 110 275 0 0 118 1 1.0 1 1 2 0

1023 50 0 0 110 254 0 0 159 0 0.0 2 0 2 1

1024 54 1 0 120 188 0 1 113 0 1.4 1 1 3 0

1025 rows × 14 columns

X = dfs.drop(['target','ca','thal'], axis = 1)

y = dfs['target']

from imblearn.over\_sampling import RandomOverSampler

ro=RandomOverSampler()

x\_data,y\_data=ro.fit\_resample(X,y)

from collections import Counter

print("Actual Data:",Counter(y))

Actual Data: Counter({1: 526, 0: 499})

print("Artificial Data:",Counter(y\_data))

Artificial Data: Counter({0: 526, 1: 526})

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_data, y\_data, test\_size = 0.2, random\_state = 0)

x\_train.shape

(841, 11)

x\_test.shape

(211, 11)

y\_train.shape

(841,)

y\_test.shape

(211,)

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import RobustScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

rfc = RandomForestClassifier(n\_estimators=60, random\_state=23)

rfc.fit(x\_train,y\_train)

RandomForestClassifier

RandomForestClassifier(n\_estimators=60, random\_state=23)

rfc.score(x\_train,y\_train)

1.0

from sklearn.metrics import accuracy\_score

y\_pred = rfc.predict(x\_test)

accuracy\_score(y\_pred,y\_test)

1.0

import sklearn.metrics

print(sklearn.metrics.classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 1.00 1.00 1.00 114

1 1.00 1.00 1.00 97

accuracy 1.00 211

macro avg 1.00 1.00 1.00 211

weighted avg 1.00 1.00 1.00 211

y\_pred = rfc.predict(x\_test )

y\_true=y\_test

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_true,y\_pred)

cm

array([[114, 0],

[ 0, 97]], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()

import pickle

pickle.dump(rfc,open('heart\_random.pkl','wb'))

random = pickle.load(open('heart\_random.pkl','rb'))

from sklearn.ensemble import BaggingClassifier

models=BaggingClassifier()

from sklearn.metrics import accuracy\_score,precision\_score,recall\_score,f1\_score

# import xgboost as xgb

# import lightgbm as lgb

models.fit(x\_train, y\_train)

BaggingClassifier

BaggingClassifier()

models.score(x\_train, y\_train)

1.0

from sklearn.metrics import accuracy\_score

y\_pred = models.predict(x\_test )

accuracy\_score(y\_pred,y\_test)

1.0

import sklearn.metrics

print(sklearn.metrics.classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 1.00 1.00 1.00 114

1 1.00 1.00 1.00 97

accuracy 1.00 211

macro avg 1.00 1.00 1.00 211

weighted avg 1.00 1.00 1.00 211

y\_pred = rfc.predict(x\_test )

y\_true=y\_test

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_true,y\_pred)

cm

array([[114, 0],

[ 0, 97]], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()

import pickle

pickle.dump(models,open('heart\_bagging.pkl','wb'))

randoms = pickle.load(open('heart\_bagging.pkl','rb'))

pip install xgboost

Collecting xgboost

Using cached xgboost-2.0.0-py3-none-win\_amd64.whl (99.7 MB)

Requirement already satisfied: scipy in c:\users\jp\anaconda3\lib\site-packages (from xgboost) (1.10.0)

Requirement already satisfied: numpy in c:\users\jp\anaconda3\lib\site-packages (from xgboost) (1.23.5)

Installing collected packages: xgboost

Successfully installed xgboost-2.0.0

Note: you may need to restart the kernel to use updated packages.

import xgboost as xgb

xg = xgb.XGBClassifier()

xg.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=None, 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=None,

num\_parallel\_tree=None, random\_state=None, ...)

xg.score(x\_train, y\_train)

1.0

from sklearn.metrics import accuracy\_score

y\_pred = xg.predict(x\_test )

accuracy\_score(y\_pred,y\_test)

1.0

import sklearn.metrics

print(sklearn.metrics.classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 1.00 1.00 1.00 114

1 1.00 1.00 1.00 97

accuracy 1.00 211

macro avg 1.00 1.00 1.00 211

weighted avg 1.00 1.00 1.00 211

y\_pred = rfc.predict(x\_test )

y\_true=y\_test

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_true,y\_pred)

cm

array([[114, 0],

[ 0, 97]], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()

import pickle

pickle.dump(xg,open('heart\_xgb.pkl','wb'))

xg = pickle.load(open('heart\_xgb.pkl','rb'))

pip install lightgbm

Collecting lightgbm

Using cached lightgbm-4.1.0-py3-none-win\_amd64.whl (1.3 MB)

Requirement already satisfied: scipy in c:\users\jp\anaconda3\lib\site-packages (from lightgbm) (1.10.0)

Requirement already satisfied: numpy in c:\users\jp\anaconda3\lib\site-packages (from lightgbm) (1.23.5)

Installing collected packages: lightgbm

Successfully installed lightgbm-4.1.0

Note: you may need to restart the kernel to use updated packages.

import lightgbm as lgb

lg = lgb.LGBMClassifier()

lg.fit(x\_train,y\_train)

[LightGBM] [Info] Number of positive: 429, number of negative: 412

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000169 seconds.

You can set `force\_row\_wise=true` to remove the overhead.

And if memory is not enough, you can set `force\_col\_wise=true`.

[LightGBM] [Info] Total Bins 362

[LightGBM] [Info] Number of data points in the train set: 841, number of used features: 11

[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.510107 -> initscore=0.040434

[LightGBM] [Info] Start training from score 0.040434

[LightGBM] [Warning] No further splits with positive gain, best gain: -inf

[LightGBM] [Warning] No further splits with positive gain, best gain: -inf

[LightGBM] [Warning] No further splits with positive gain, best gain: -inf

LGBMClassifier

LGBMClassifier()

lg.score(x\_train, y\_train)

1.0

from sklearn.metrics import accuracy\_score

y\_pred = lg.predict(x\_test )

accuracy\_score(y\_pred,y\_test)

1.0

import sklearn.metrics

print(sklearn.metrics.classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 1.00 1.00 1.00 114

1 1.00 1.00 1.00 97

accuracy 1.00 211

macro avg 1.00 1.00 1.00 211

weighted avg 1.00 1.00 1.00 211

y\_pred = rfc.predict(x\_test )

y\_true=y\_test

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_true,y\_pred)

cm

array([[114, 0],

[ 0, 97]], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()

import pickle

pickle.dump(rfc,open('heart\_lgb.pkl','wb'))

lgb = pickle.load(open('heart\_lgb.pkl','rb'))