In [1]:
```
#start with the name of allah
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from pandas.api.types import is_string_dtype
from pandas.api.types import is_numeric_dtype
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv('Heart Disease.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	Ge
0	No	16.60	Yes	No	No	3	30	No	Female	55-59	White	Yes	Yes	V
1	No	20.34	No	No	Yes	0	0	No	Female	80 or older	White	No	Yes	V
2	No	26.58	Yes	No	No	20	30	No	Male	65-69	White	Yes	Yes	
3	No	24.21	No	No	No	0	0	No	Female	75-79	White	No	No	
4	No	23.71	No	No	No	28	0	Yes	Female	40-44	White	No	Yes	V

In [4]:
```
df.isnull().sum()
```

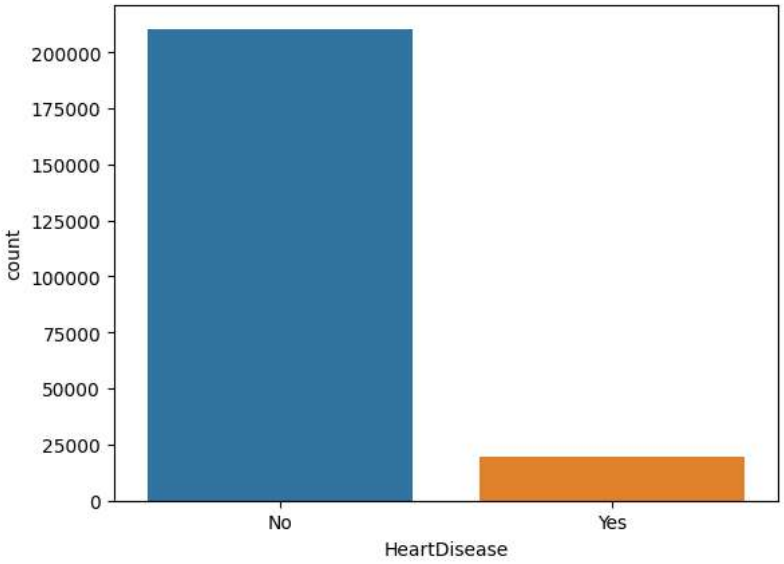
Out[4]:
```
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

In [5]:
```
df.HeartDisease.value_counts()
```

Out[5]:
```
No     210435
Yes     19742
Name: HeartDisease, dtype: int64
```

In [6]:
```
sns.countplot(df,x='HeartDisease')
```

Out[6]: <AxesSubplot: xlabel='HeartDisease', ylabel='count'>



In [7]:
```
from pandas_profiling import ProfileReport
```

In [8]:
```
ProfileReport(df)
```

Summarize dataset: 100% 43/43 [00:19<00:00, 2.49it/s, Completed]

Generate report structure: 100% 1/1 [00:03<00:00, 3.77s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.09s/it]

Overview

Dataset statistics

Number of variables	18
Number of observations	230177
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	7428
Duplicate rows (%)	3.2%
Total size in memory	31.6 MiB
Average record size in memory	144.0 B

Variable types

Boolean	9
Numeric	4
Categorical	5

Alerts

Dataset has 7428 (3.2%) duplicate rows	Duplicates
`HeartDisease` is highly imbalanced (57.8%)	Imbalance
`AlcoholDrinking` is highly imbalanced (63.7%)	Imbalance
`Stroke` is highly imbalanced (76.6%)	Imbalance
`Race` is highly imbalanced (51.5%)	Imbalance
`Diabetic` is highly imbalanced (62.1%)	Imbalance
`KidneyDisease` is highly imbalanced (77.3%)	Imbalance

Out[8]:

In [9]:
```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
```

In [10]:
```
le=LabelEncoder()
```

In [11]:
```
ordinal = OrdinalEncoder(categories=[['Poor','Fair','Good','Very good','Excellent']])
```

In [12]:
```
df[['GenHealth']] = ordinal.fit_transform(df[['GenHealth']])
```

In [ ]:
```

```

In [13]:
```
ordi =  OrdinalEncoder(categories=[['Yes','Yes (during pregnancy)','No, borderline diabetes','No']])
```

In [14]:
```
df[['Diabetic']]= ordi.fit_transform(df[['Diabetic']])
```

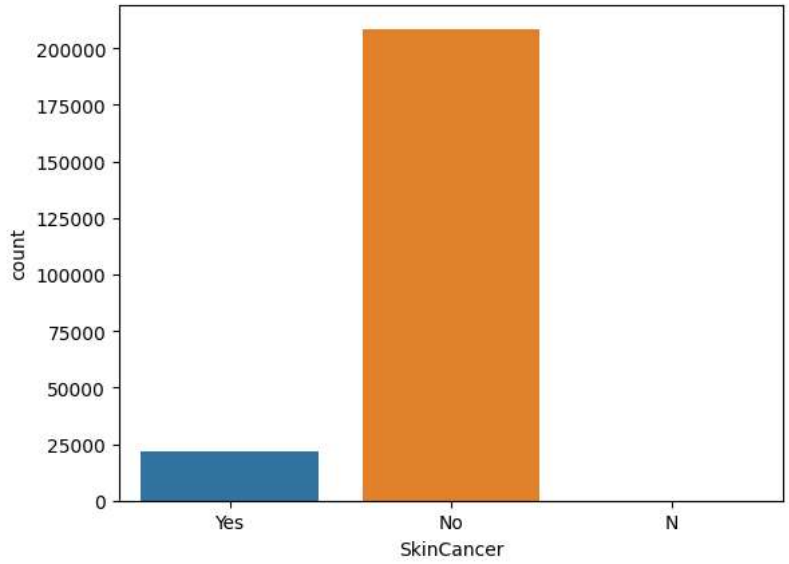
In [15]:
```
df.head()
```

Out[15]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	Ge
0	No	16.60	Yes	No	No	3	30	No	Female	55-59	White	0.0	Yes	
1	No	20.34	No	No	Yes	0	0	No	Female	80 or older	White	3.0	Yes	
2	No	26.58	Yes	No	No	20	30	No	Male	65-69	White	0.0	Yes	
3	No	24.21	No	No	No	0	0	No	Female	75-79	White	3.0	No	
4	No	23.71	No	No	No	28	0	Yes	Female	40-44	White	3.0	Yes	

In [16]:
```
sns.countplot(df,x='SkinCancer')
```

Out[16]:
```
<AxesSubplot: xlabel='SkinCancer', ylabel='count'>
```



In [17]:
```
df.SkinCancer.value_counts()
```

Out[17]:
```
No     208602
Yes     21574
N           1
Name: SkinCancer, dtype: int64
```

In [18]:
```
ordi = OrdinalEncoder(categories=[['Yes','N','No']])
```

In [19]:
```
df[['SkinCancer']]=ordi.fit_transform(df[['SkinCancer']])
```

In [20]:
```
df.head()
```

Out[20]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	Ge
0	No	16.60	Yes	No	No	3	30	No	Female	55-59	White	0.0	Yes	
1	No	20.34	No	No	Yes	0	0	No	Female	80 or older	White	3.0	Yes	
2	No	26.58	Yes	No	No	20	30	No	Male	65-69	White	0.0	Yes	
3	No	24.21	No	No	No	0	0	No	Female	75-79	White	3.0	No	
4	No	23.71	No	No	No	28	0	Yes	Female	40-44	White	3.0	Yes	

In [21]:
```
df.Sex.value_counts()
```

Out[21]:
```
Female    120951
Male      109226
Name: Sex, dtype: int64
```

In [22]:
```
df.DiffWalking.value_counts()
```

Out[22]:
```
No     198224
Yes     31953
Name: DiffWalking, dtype: int64
```

In [23]:
```
df.Stroke.value_counts()
```

Out[23]:
```
No     221403
Yes      8774
Name: Stroke, dtype: int64
```

In [24]:
```
df.KidneyDisease.value_counts()
```

Out[24]:
```
No     221708
Yes      8469
Name: KidneyDisease, dtype: int64
```

In [25]:
```
df.Race.value_counts()
```

Out[25]:
```
White                            177373
Black                             18707
Hispanic                          16910
Other                              7632
Asian                              6053
American Indian/Alaskan Native     3502
Name: Race, dtype: int64
```

In [26]:
```
df.AgeCategory.value_counts()
```

Out[26]:
```
65-69          24662
60-64          24301
70-74          22518
55-59          21566
50-54          18194
80 or older    17921
75-79          15674
45-49          15459
18-24          14998
40-44          14765
35-39          14665
30-34          13397
25-29          12057
Name: AgeCategory, dtype: int64
```

In [27]:
```
inalEncoder(categories=[['65-69','60-64','70-74','55-59','50-54','80 or older','75-79','45-49','18-24','40-44','35-39','30-34','2
```

In [28]:
```
df[['AgeCategory']] = ordinal3.fit_transform(df[['AgeCategory']])
```

In [29]:
```
for i in df.columns:
    if is_numeric_dtype(df[i]):

        continue
    else:
        df[i]=le.fit_transform(df[i])
```

In [ ]:
```

```

In [30]:
```
df.head()
```

Out[30]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHe
0	0	16.60	1	0	0	3	30	0	0	3.0	5	0.0	1	
1	0	20.34	0	0	1	0	0	0	0	5.0	5	3.0	1	
2	0	26.58	1	0	0	20	30	0	1	0.0	5	0.0	1	
3	0	24.21	0	0	0	0	0	0	0	6.0	5	3.0	0	
4	0	23.71	0	0	0	28	0	1	0	9.0	5	3.0	1	

In [31]:
```
from pycaret.classification import *
```

In [32]:
```
setup(df,target='HeartDisease',
     normalize=True,
     numeric_imputation='median',

     fix_imbalance=True,
     fix_imbalance_method='randomoversampler',
     log_experiment=True,
     )
```

	Description	Value
0	Session id	1983
1	Target	HeartDisease
2	Target type	Binary
3	Original data shape	(230177, 18)
4	Transformed data shape	(363662, 18)
5	Transformed train set shape	(294608, 18)
6	Transformed test set shape	(69054, 18)
7	Numeric features	17
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	median
11	Categorical imputation	mode
12	Fix imbalance	True
13	Fix imbalance method	randomoversampler
14	Normalize	True
15	Normalize method	zscore
16	Fold Generator	StratifiedKFold
17	Fold Number	10
18	CPU Jobs	-1
19	Use GPU	False
20	Log Experiment	MlflowLogger
21	Experiment Name	clf-default-name
22	USI	88c0

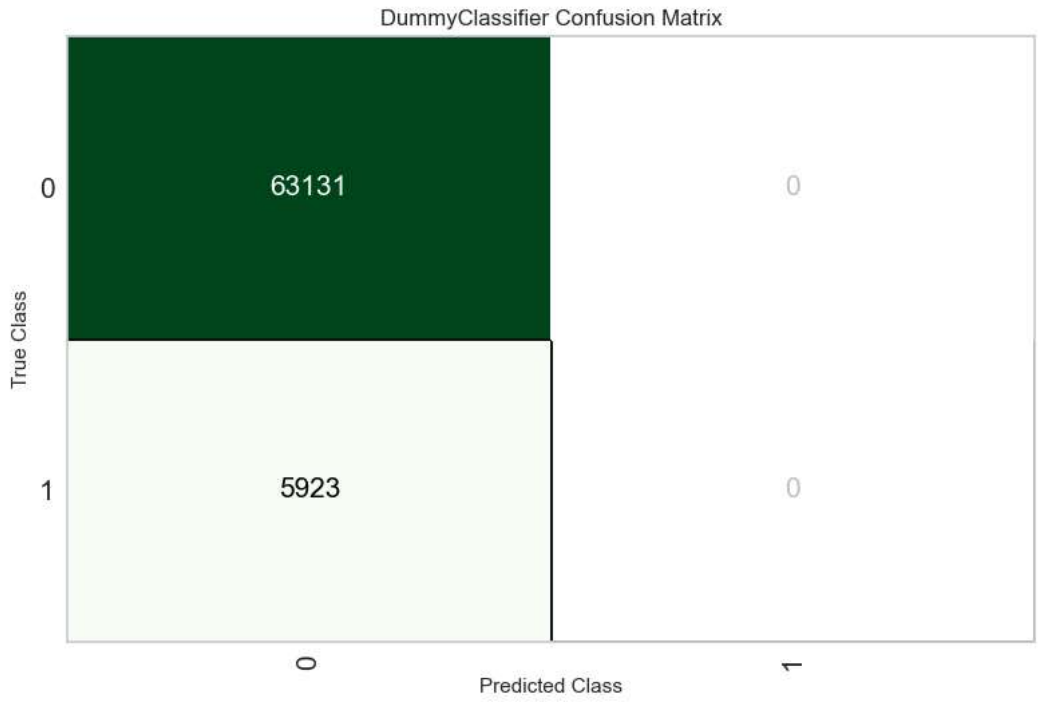
```
2024/01/04 22:09:19 WARNING mlflow.utils.git_utils: Failed to import Git (the Git executable is probably not on your PATH), so
Git SHA is not available. Error: Failed to initialize: Bad git executable.
The git executable must be specified in one of the following ways:
    - be included in your $PATH
    - be set via $GIT_PYTHON_GIT_EXECUTABLE
    - explicitly set via git.refresh()

All git commands will error until this is rectified.

This initial warning can be silenced or aggravated in the future by setting the
$GIT_PYTHON_REFRESH environment variable. Use one of the following values:
    - quiet|q|silence|s|none|n|0: for no warning or exception
    - warn|w|warning|1: for a printed warning
    - error|e|raise|r|2: for a raised exception

Example:
    export GIT_PYTHON_REFRESH=quiet
```

Out[32]:
```
<pycaret.classification.oop.ClassificationExperiment at 0x1fa15b89990>
```

In [36]:
```
dum=create_model('dummy')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9143	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

In [46]:
```
plot_model(dum,plot='confusion_matrix')
```



In [34]:
```
create_model('knn')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7963	0.6999	0.4826	0.2062	0.2890	0.1919	0.2156
1	0.7965	0.6959	0.4768	0.2050	0.2867	0.1895	0.2125
2	0.7909	0.6897	0.4863	0.2017	0.2852	0.1865	0.2114
3	0.7902	0.6889	0.4823	0.1999	0.2826	0.1837	0.2082
4	0.7861	0.7011	0.5051	0.2017	0.2883	0.1888	0.2165
5	0.7954	0.6987	0.4855	0.2061	0.2893	0.1920	0.2162
6	0.7951	0.6981	0.4906	0.2070	0.2911	0.1938	0.2186
7	0.7900	0.7013	0.4841	0.2004	0.2834	0.1845	0.2092
8	0.7893	0.6903	0.4732	0.1969	0.2781	0.1786	0.2023
9	0.7912	0.6907	0.4841	0.2015	0.2846	0.1860	0.2105
Mean	0.7921	0.6954	0.4851	0.2026	0.2858	0.1875	0.2121
Std	0.0033	0.0048	0.0081	0.0031	0.0037	0.0044	0.0046

Out[34]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
```

In [37]:
```
tune_model(dum,n_iter=250)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.9143	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.9142	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

```
Fitting 10 folds for each of 4 candidates, totalling 40 fits
Original model was better than the tuned model, hence it will be returned. NOTE: The display metrics are for the tuned model (n
ot the original one).
```

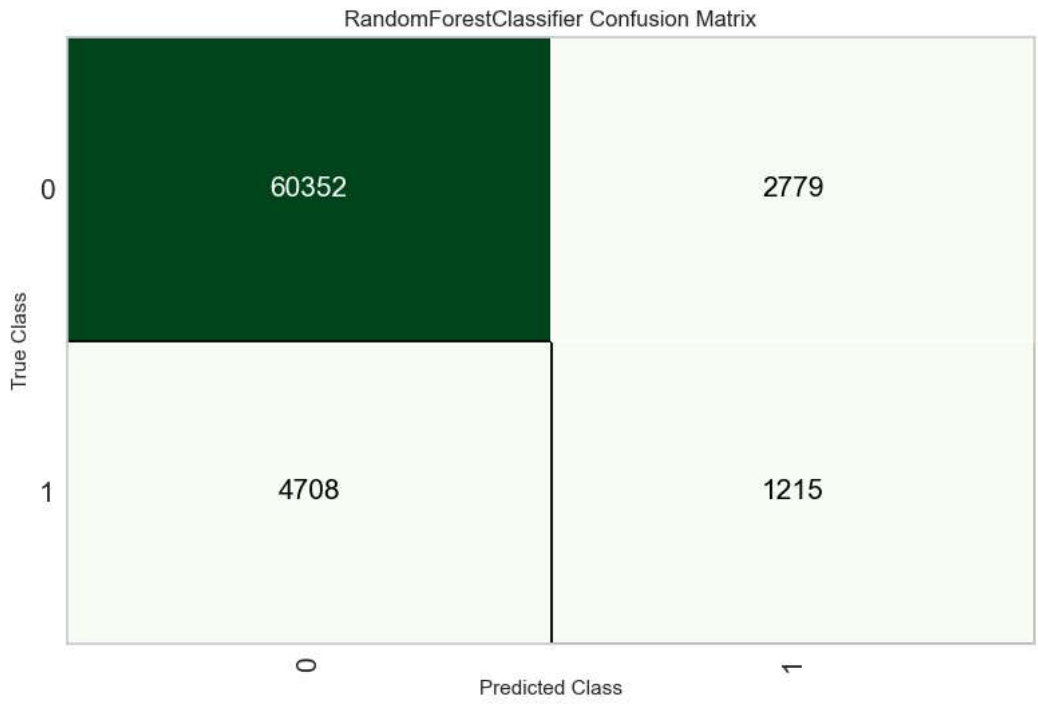
Out[37]:
```
DummyClassifier(constant=None, random_state=1983, strategy='prior')
```

In [40]:
```
rf=create_model('rf')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8916	0.7900	0.1918	0.2964	0.2329	0.1774	0.1823
1	0.8902	0.7880	0.1954	0.2909	0.2338	0.1771	0.1811
2	0.8927	0.7896	0.1997	0.3070	0.2420	0.1870	0.1921
3	0.8920	0.7853	0.1933	0.2990	0.2348	0.1796	0.1846
4	0.8929	0.7988	0.1968	0.3063	0.2396	0.1850	0.1902
5	0.8964	0.7973	0.2077	0.3329	0.2558	0.2033	0.2099
6	0.8941	0.7905	0.2127	0.3224	0.2563	0.2019	0.2070
7	0.8931	0.7900	0.2019	0.3103	0.2446	0.1899	0.1949
8	0.8946	0.7865	0.1925	0.3133	0.2385	0.1853	0.1916
9	0.8937	0.7845	0.2135	0.3207	0.2563	0.2016	0.2064
Mean	0.8931	0.7901	0.2005	0.3099	0.2435	0.1888	0.1940
Std	0.0016	0.0045	0.0078	0.0122	0.0090	0.0096	0.0099

In [ ]:
```
```

In [44]:
```
plot_model(rf,plot='confusion_matrix')
```



In [45]:
```
predict_model(rf)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Random Forest Classifier	0.8916	0.7955	0.2051	0.3042	0.2450	0.1890	0.1933

Out[45]:

	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth
220421	25.510000	1	0	0	0	14	0	0	3.0	5	3.0	1	3.0
18882	21.260000	0	1	0	0	7	0	0	11.0	5	3.0	1	3.0
71344	31.750000	0	0	0	0	12	0	0	4.0	5	3.0	1	4.0
176487	25.100000	1	0	0	0	0	0	1	10.0	5	3.0	1	4.0
717	24.959999	1	0	0	2	0	0	0	6.0	5	3.0	1	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
229551	32.279999	1	0	0	30	30	0	1	12.0	4	3.0	1	2.0
79857	52.299999	0	0	0	0	30	0	1	11.0	5	3.0	0	2.0
65843	29.860001	0	1	0	30	0	0	0	4.0	5	0.0	1	2.0
192483	26.750000	0	0	0	0	0	0	0	10.0	3	3.0	0	4.0
118984	41.810001	0	0	0	0	30	0	1	8.0	4	0.0	1	2.0

69054 rows × 20 columns

In [47]:
```
save_model(rf,'Random_Fores')
```

```
Transformation Pipeline and Model Successfully Saved
```

Out[47]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['BMI', 'Smoking',
                                              'AlcoholDrinking', 'Stroke',
                                              'PhysicalHealth', 'MentalHealth',
                                              'DiffWalking', 'Sex',
                                              'AgeCategory', 'Race', 'Diabetic',
                                              'PhysicalActivity', 'GenHealth',
                                              'SleepTime', 'Asthma',
                                              'KidneyDisease', 'SkinCancer'],
                                     transformer=SimpleImputer(add_indi...
                  RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                         class_weight=None, criterion='gini',
                                         max_depth=None, max_features='sqrt',
                                         max_leaf_nodes=None, max_samples=None,
                                         min_impurity_decrease=0.0,
                                         min_samples_leaf=1, min_samples_split=2,
                                         min_weight_fraction_leaf=0.0,
                                         n_estimators=100, n_jobs=-1,
                                         oob_score=False, random_state=1983,
                                         verbose=0, warm_start=False))],
          verbose=False),
 'Random_Fores.pkl')
```

In [ ]: