#Red Wine Quality Dataset

```
from google.colab import files


uploaded = files.upload()

import pandas as pd
wn = pd.read_csv('winequality2.csv')
wn.head()
wn.isnull().values.any()
wn.describe()

x = wn.drop('quality', axis = 1)
y = wn['quality']
from sklearn.model_selection import train_test_split
!pip install fastai wwf bayesian-optimization -q --upgrade
from bayes_opt import BayesianOptimization
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import VarianceThreshold
from imblearn.over_sampling import SMOTE
import numpy as np
from sklearn.datasets import make_classification
rng = np.random.RandomState(42)
x, y = make_classification(random_state=rng)
oversample=SMOTE()
wn=oversample.fit_resample(x,y)
import sklearn.model_selection
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y,
random_state=42, stratify=y)
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(random_state=42)
clf = clf.fit(x_train, y_train)
y_hat = clf.predict(x_test)
print("RF Accuracy", sklearn.metrics.accuracy_score(y_test, y_hat))
from sklearn.preprocessing import StandardScaler
pipe = Pipeline([
('scaler', StandardScaler()),
('selector', VarianceThreshold()),
('classifier', RandomForestClassifier()),
```

```
])
pipe.fit(x_train, y_train)
print('Test set score: ' + str(pipe.score(x_test,y_test)))
from sklearn.model_selection import cross_val_score
def
objective(max_depth,max_leaf_nodes,max_samples,min_samples_leaf,min_samples_s
plit,n_estimators,n_jobs):
   model = RandomForestClassifier(
max_depth=int(max_depth),
max_leaf_nodes=int(max_leaf_nodes),
max_samples=min(max_samples,0.999),
min_samples_leaf=int(min_samples_leaf),
n_estimators=int(n_estimators),
n_jobs=-1,
random_state=42)

   return -1.0 * cross_val_score(model, x_train, y_train, cv=3,
scoring="neg_mean_squared_error").mean()
param_bounds = {
   'max_depth': (1,500),
   'max_leaf_nodes': (10,100),
   'max_samples': (0.1,0.8),
   'min_samples_leaf': (1,6),
    'min_samples_split': (1,6),
   'n_estimators': (50,700),
   'n_jobs': (-1,400)
}
opt = BayesianOptimization(f=objective, pbounds=param_bounds, random_state=42)
opt.maximize(init_points=5, n_iter=50)
best_params = opt.max['params']
best_params
final_model = RandomForestClassifier(
                              max_depth=int(best_params['max_depth']),
                              max_leaf_nodes=int(best_params['max_leaf_nodes']),
                              max_samples=(best_params['max_samples']),
                              min_samples_leaf=int(best_params['min_samples_leaf']),
                              min_samples_split=int((best_params['min_samples_split'])),
                              n_estimators=int(best_params['n_estimators']),
                              n_jobs=int(best_params['n_jobs']),
```

```
                                random_state=42)

final_model.fit(x_train, y_train)
print('Test set score: ' + str(final_model.score(x_test,y_test)))
print('Test set score: ' + str(pipe.score(x_test,y_test)))
```

#Brain Stroke Dataset

```
from google.colab import files


uploaded = files.upload()

import pandas as pd
st = pd.read_csv('brain_stroke.csv')
st.head()
print(st['gender'].unique())
st['gender'].value_counts()
print(st['ever_married'].unique())
st['ever_married'].value_counts()
print(st['work_type'].unique())
st['work_type'].value_counts()
print(st['Residence_type'].unique())
st['Residence_type'].value_counts()
print(st['smoking_status'].unique())
st['smoking_status'].value_counts()
ohed = pd.get_dummies(st, columns = ['gender', 'ever_married', 'work_type',
'Residence_type', 'smoking_status'])
ohed.head()
ohed.describe()
from sklearn.model_selection import train_test_split
x = ohed.drop('stroke', axis = 1)
y = ohed['stroke']
!pip install fastai wwf bayesian-optimization -q --upgrade
from bayes_opt import BayesianOptimization
import numpy as np
from sklearn.datasets import make_classification
```

```
from imblearn.over_sampling import SMOTE
oversample=SMOTE()
ohed=oversample.fit_resample(x,y)
import sklearn.model_selection
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y,
random_state=42, stratify=y)
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(random_state=42)
clf = clf.fit(x_train, y_train)
y_hat = clf.predict(x_test)
print("RF Accuracy", sklearn.metrics.accuracy_score(y_test, y_hat))
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import VarianceThreshold
pipe = Pipeline([
('scaler', StandardScaler()),
('selector', VarianceThreshold()),
('classifier', RandomForestClassifier()),

])
pipe.fit(x_train, y_train)
print('Test set score: ' + str(pipe.score(x_test,y_test)))
from sklearn.model_selection import cross_val_score
def
objective(max_depth,max_leaf_nodes,max_samples,min_samples_leaf,min_samples_s
plit,n_estimators,n_jobs):
    model = RandomForestClassifier(
max_depth=int(max_depth),
max_leaf_nodes=int(max_leaf_nodes),
max_samples=min(max_samples,0.999),
min_samples_leaf=int(min_samples_leaf),
n_estimators=int(n_estimators),
n_jobs=-1,
random_state=42)

    return -1.0 * cross_val_score(model, x_train, y_train, cv=3,
scoring="neg_mean_squared_error").mean()
param_bounds = {
    'max_depth': (1,500),
    'max_leaf_nodes': (10,100),
```

```
    'max_samples': (0.1,0.8),
    'min_samples_leaf': (1,6),
     'min_samples_split': (1,6),
    'n_estimators': (50,700),
    'n_jobs': (-1,400)
}
opt = BayesianOptimization(f=objective, pbounds=param_bounds, random_state=42)
opt.maximize(init_points=5, n_iter=50)
best_params = opt.max['params']
best_params
final_model = RandomForestClassifier(
                                max_depth=int(best_params['max_depth']),
                                max_leaf_nodes=int(best_params['max_leaf_nodes']),
                                max_samples=(best_params['max_samples']),
                                min_samples_leaf=int(best_params['min_samples_leaf']),
                                min_samples_split=int(round(best_params['min_samples_split'])),
                                n_estimators=int(best_params['n_estimators']),
                                n_jobs=int(best_params['n_jobs']),
                                random_state=42)

final_model.fit(x_train, y_train)
print('Test set score: ' + str(final_model.score(x_test,y_test)))
print('Test set score: ' + str(pipe.score(x_test,y_test)))
```