hyper-parameter optimization project

Authors: Mohammadjavad Abbas pour & Ahmad Yazdani

Supervisor: Dr.Taheri

Optimization in data science

Shahid Beheshti University

July , 2022

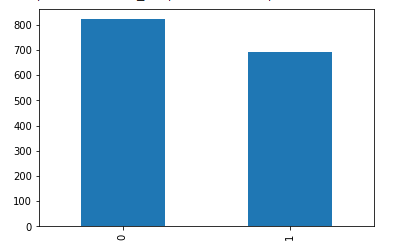
**Abstract**

This project is a hyper-parameter-optimization of the neural network, which has been implemented on Bace molecular data set . I use ray , pytorch ,schedulers and search-algorithms and more objects in this project.

## Introduction of dataset

## Type of data set is molecular. The number of records is 1513 , target variables are 0 or 1 and the number of features is 200.

## Distribution of labels plot:



In this case the distribution of labels are balance. There is an balance between the class of one and zero labels in Bace dataset.

## Preprocessing

Data partitioning, Feature selection, and Missing are the important things for prepare the data for final processing.

**Import libraries**

%%capture

try:

    import ray

except:

    !pip install -U ray

    import ray

try:

    import optuna

except:

    %pip install optuna

    import optuna

    %%capture

try:

    from featurewiz import featurewiz

except:

    !pip install featurewiz==0.1.70

import numpy as np

import pandas as pd

import torch

import torch.nn as nn

from torch.utils.data import TensorDataset

from pandas import read\_csv

from sklearn.impute import SimpleImputer

import io

import matplotlib as mpl

import matplotlib.pyplot as plt

import random

from functools import partial

import os

import torch.nn.functional as F

import torch.optim as optim

from torch.utils.data import random\_split

from torchsummary import summary

from ray import tune

from ray.tune import CLIReporter

from ray.tune.schedulers import ASHAScheduler

**Data reading**

from google.colab import files

uploaded = files.upload()

dataX = pd.read\_csv(io.BytesIO(uploaded['bace\_global\_cdf\_rdkit.csv']))

dataY = pd.read\_csv(io.BytesIO(uploaded1['bace(lables).csv']))

**Data splitting**

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

Y = dataY.iloc[:, 1]

X = dataX.iloc[:,1:201]

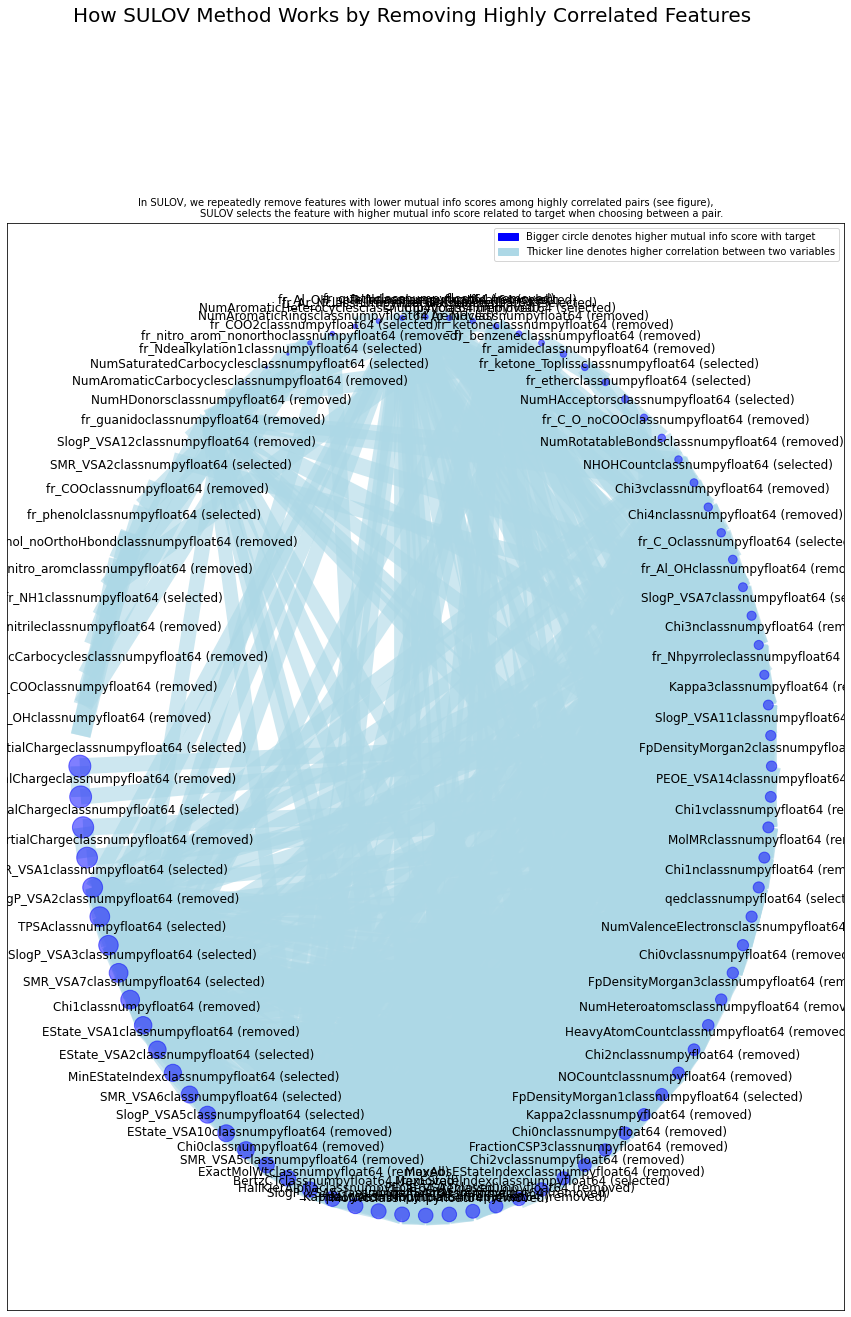
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size= 0.15,random\_state=1234)

**Feature selection**

I used Corrolation matrix and featurewiz library to select important featuresof Bace data set, then I've observed that featurewiz is better.

from featurewiz import FeatureWiz  
features = FeatureWiz(corr\_limit=0.70, feature\_engg='', category\_encoders='', dask\_xgboost\_flag=False, nrows=None, verbose=2)  
X\_train = features.fit\_transform(X\_train, Y\_train)  
X\_test = features.transform(X\_test)  
features.features ### provides the list of selected features ###

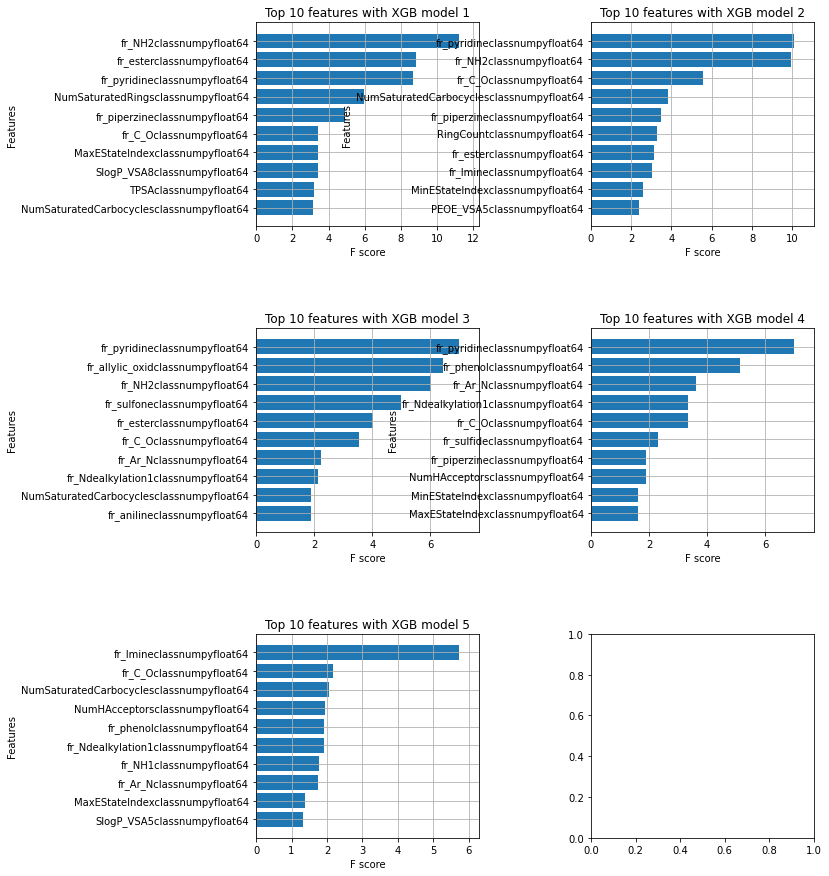
############################################################################################  
############ F A S T F E A T U R E E N G G A N D S E L E C T I O N ! ########  
# Be judicious with featurewiz. Don't use it to create too many un-interpretable features! #  
############################################################################################  
Skipping feature engineering since no feature\_engg input...  
Skipping category encoding since no category encoders specified in input...  
\*\*INFO: featurewiz can now read feather formatted files. Loading train data...  
 Shape of your Data Set loaded: (1286, 201)  
 Loaded train data. Shape = (1286, 201)  
 Some column names had special characters which were removed...  
No test data filename given...  
#######################################################################################  
######################## C L A S S I F Y I N G V A R I A B L E S ####################  
#######################################################################################  
Classifying variables in data set...  
 200 Predictors classified...  
 33 variable(s) to be removed since ID or low-information variables  
 more than 33 variables to be removed; too many to print...  
train data shape before dropping 33 columns = (1286, 201)  
 train data shape after dropping columns = (1286, 168)  
 Converted pandas dataframe into a Dask dataframe ...  
GPU active on this device  
 Tuning XGBoost using GPU hyper-parameters. This will take time...  
 After removing redundant variables from further processing, features left = 167  
No interactions created for categorical vars since feature engg does not specify it  
#### Single\_Label Binary\_Classification problem ####  
#######################################################################################  
##### Searching for Uncorrelated List Of Variables (SULOV) in 167 features ############  
#######################################################################################  
 there are no null values in dataset...  
 Removing (62) highly correlated variables:



Time taken for SULOV method = 4 seconds  
 Adding 0 categorical variables to reduced numeric variables of 105  
Final list of selected vars after SULOV = 105  
Readying dataset for Recursive XGBoost by converting all features to numeric...  
#######################################################################################  
##### R E C U R S I V E X G B O O S T : F E A T U R E S E L E C T I O N #######  
#######################################################################################  
 using regular XGBoost  
Train and Test loaded into Dask dataframes successfully after feature\_engg completed  
Current number of predictors = 105   
 XGBoost version: 1.6.1  
Number of booster rounds = 100  
 using 105 variables...  
 Time taken for regular XGBoost feature selection = 2 seconds  
 using 84 variables...  
 Time taken for regular XGBoost feature selection = 5 seconds  
 using 63 variables...  
 Time taken for regular XGBoost feature selection = 7 seconds  
 using 42 variables...  
 Time taken for regular XGBoost feature selection = 8 seconds  
 using 21 variables...  
 Time taken for regular XGBoost feature selection = 9 seconds

Total time taken for XGBoost feature selection = 12 seconds  
#######################################################################################  
##### F E A T U R E S E L E C T I O N C O M P L E T E D #######  
#######################################################################################  
Selected 56 important features. Too many to print...  
  
 Time taken for feature selection = 15 seconds  
Returning 2 dataframes: dataname and test\_data with 56 important features.  
 Time taken to create entire pipeline = 17 second(s)

["('fr\_NH2', <class 'numpy.float64'>)",  
 "('fr\_ester', <class 'numpy.float64'>)",  
 "('fr\_pyridine', <class 'numpy.float64'>)",  
 "('NumSaturatedRings', <class 'numpy.float64'>)",  
 "('fr\_piperzine', <class 'numpy.float64'>)",  
 "('fr\_C\_O', <class 'numpy.float64'>)",  
 "('MaxEStateIndex', <class 'numpy.float64'>)",  
 "('SlogP\_VSA8', <class 'numpy.float64'>)",  
 "('TPSA', <class 'numpy.float64'>)",  
 "('NumSaturatedCarbocycles', <class 'numpy.float64'>)",  
 "('PEOE\_VSA4', <class 'numpy.float64'>)",  
 "('SMR\_VSA10', <class 'numpy.float64'>)",  
 "('fr\_aryl\_methyl', <class 'numpy.float64'>)",  
 "('NumHAcceptors', <class 'numpy.float64'>)",  
 "('MinEStateIndex', <class 'numpy.float64'>)",  
 "('SlogP\_VSA5', <class 'numpy.float64'>)",  
 "('MaxPartialCharge', <class 'numpy.float64'>)",  
 "('SlogP\_VSA4', <class 'numpy.float64'>)",  
 "('VSA\_EState10', <class 'numpy.float64'>)",  
 "('PEOE\_VSA12', <class 'numpy.float64'>)",  
 "('SlogP\_VSA1', <class 'numpy.float64'>)",  
 "('PEOE\_VSA11', <class 'numpy.float64'>)",  
 "('PEOE\_VSA2', <class 'numpy.float64'>)",  
 "('SMR\_VSA6', <class 'numpy.float64'>)",  
 "('fr\_bicyclic', <class 'numpy.float64'>)",  
 "('RingCount', <class 'numpy.float64'>)",  
 "('fr\_Imine', <class 'numpy.float64'>)",  
 "('PEOE\_VSA5', <class 'numpy.float64'>)",  
 "('fr\_phenol', <class 'numpy.float64'>)",  
 "('fr\_NH0', <class 'numpy.float64'>)",  
 "('fr\_para\_hydroxylation', <class 'numpy.float64'>)",  
 "('fr\_allylic\_oxid', <class 'numpy.float64'>)",  
 "('fr\_sulfone', <class 'numpy.float64'>)",  
 "('fr\_Ar\_N', <class 'numpy.float64'>)",  
 "('fr\_Ndealkylation1', <class 'numpy.float64'>)",  
 "('fr\_aniline', <class 'numpy.float64'>)",  
 "('fr\_Ndealkylation2', <class 'numpy.float64'>)",  
 "('fr\_oxazole', <class 'numpy.float64'>)",  
 "('fr\_imidazole', <class 'numpy.float64'>)",  
 "('qed', <class 'numpy.float64'>)",  
 "('fr\_unbrch\_alkane', <class 'numpy.float64'>)",  
 "('fr\_sulfide', <class 'numpy.float64'>)",  
 "('SlogP\_VSA3', <class 'numpy.float64'>)",  
 "('SMR\_VSA7', <class 'numpy.float64'>)",  
 "('MaxAbsPartialCharge', <class 'numpy.float64'>)",  
 "('NHOHCount', <class 'numpy.float64'>)",  
 "('Chi4v', <class 'numpy.float64'>)",  
 "('SMR\_VSA1', <class 'numpy.float64'>)",  
 "('fr\_ketone\_Topliss', <class 'numpy.float64'>)",  
 "('fr\_NH1', <class 'numpy.float64'>)",  
 "('fr\_ether', <class 'numpy.float64'>)",  
 "('FpDensityMorgan1', <class 'numpy.float64'>)",  
 "('SlogP\_VSA7', <class 'numpy.float64'>)",  
 "('fr\_COO2', <class 'numpy.float64'>)",  
 "('SMR\_VSA2', <class 'numpy.float64'>)",  
 "('fr\_Nhpyrrole', <class 'numpy.float64'>)"]



It should be noted that the feature selection section is processed**only based on the information in the training set.**

**56** features were selected as influencing features on the target variable in the Bace dataset**.**

**PCA**

I have used PCA but the result of our model is not good in this way, cause of that ignored it.

**from** sklearn.decomposition **import** PCA **as** sklearnPCA

pca **=** sklearnPCA(n\_components**=**40)

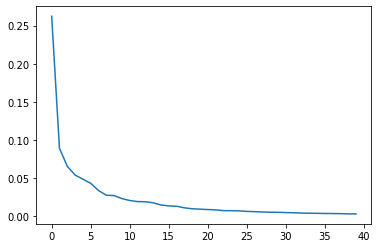
pca**.**fit(X\_train)

X\_train**=**pca**.**transform(X\_train)

X\_test**=**pca**.**transform(X\_test)

plt**.**plot(np**.**arange(40),pca**.**explained\_variance\_ratio\_)

plt**.**show()

****

**Scailing data**

from sklearn.preprocessing import StandardScaler

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

from sklearn import preprocessing

import random

np.random.seed(1234)

from sklearn import preprocessing

scaler\_data = preprocessing.MinMaxScaler()

X\_train = scaler\_data.fit\_transform(X\_train)

X\_test = scaler\_data.transform(X\_test)

scaler\_labels = preprocessing.MinMaxScaler()

**I have not used MinMaxScaler method on lables, because they are 0 or 1.**

**Transform** **to** **torch** **tensor**

device = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')

tensor\_x = torch.tensor(X\_train, dtype=torch.float).to(device)

tensor\_x2 = torch.tensor(X\_test, dtype=torch.float).to(device)

tensor\_y = torch.tensor(Y\_train, dtype=torch.float).to(device)

tensor\_y2 = torch.tensor(Y\_test, dtype=torch.float).to(device)

# create your dataset

trainset = TensorDataset(tensor\_x, tensor\_y)

testset = TensorDataset(tensor\_x2,tensor\_y2)

**Loading the Data**

def load\_data(data\_dir=None):

    return trainset, testset

### **Missing values**

Bace data set has not any missing values.

### **Accuracy and error measurement**

I used ROC\_AUC score for Bace data set ,cause that is a classification project.

from sklearn.metrics import roc\_auc\_score

def compute\_score(model, data\_loader, device="cpu"):

    model.eval()

    metric = roc\_auc\_score

    with torch.no\_grad():

        prediction\_all= torch.empty(0, device=device)

        labels\_all= torch.empty(0, device=device)

        for i, (feats, labels) in enumerate(data\_loader):

            feats=feats.to(device)

            labels=labels.to(device)

            prediction = model(feats).to(device)

            prediction = torch.sigmoid(prediction).to(device)

            prediction\_all = torch.cat((prediction\_all, prediction), 0)

            labels\_all = torch.cat((labels\_all, labels), 0)

        try:

            t = metric(labels\_all.int().cpu(), prediction\_all.cpu()).item()

        except ValueError:

            t = 0

    return t