DATA PIPELINE TO PREDICT OUTPUT FOR A POINT OR SET OF POINTS:

In [1]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings('ignore')
import random
```

In [2]:
```
def final_fun_1(X):

    # DATA PREPROCESSING

    # CHECKING FOR NULL VALUES
    if X.isnull().sum().sum() != 0:
        X.fillna(0)

    # REMOVING BINARY FEATURES WITH UNIQUE VALUES
    with open('Binary_features_with_uniquevalues_throughout.pkl', 'rb') as f:
        list1 = pickle.load(f)
        f.close()
    X.drop(list1,axis=1,inplace=True)


    # CONVERTING CATEGORICAL FEATURES TO NUMERICAL FEATURES USING MEAN ENCODING
    with open('targets.pkl', 'rb') as f:
        y_train = pickle.load(f)
    y_mean = y_train.mean()

    # MEAN ENCODING
    with open('mean_list.pkl', 'rb') as f:
        means_list = pickle.load(f)
        f.close()
    j = 0
    for i in X.iloc[:,1:9].columns:
        a = []
        for k in X[i].values:
            if k in list(means_list[j].keys()):
                a.append(means_list[j][k])
            else:
                a.append(y_mean)

        X[i] = a
        j+=1


    # FEATURE ENGINEERING
    # REMOVING DUPLICATE COLUMNS
    with open('Duplicated_columns.pkl', 'rb') as f:
        Duplicated_columns = pickle.load(f)
        f.close()
```

```
X.drop(Duplicated_columns,axis=1,inplace=True)

# ADDING NEW FEATURES
X['X0_X5'] = X.apply(lambda x: x['X0']+x['X5'],axis=1)
X['X5_X6'] = X.apply(lambda x: x['X5']+x['X6'],axis=1)
X['X6_X8'] = X.apply(lambda x: x['X6']+x['X8'],axis=1)

# REMOVING FEATURES WITH LESS VARIANCE
with open('less_variance_features.pkl', 'rb') as f:
    less_variance_features = pickle.load(f)
    f.close()
X.drop(less_variance_features,axis=1,inplace=True)

IDS = X['ID'].values
X.drop(['ID'],axis=1,inplace=True)

# PREDICTING USING BEST MODEL

with open('top_100_features_from_DT_rfecv.pkl', 'rb') as f:
    top_100_features_from_DT_rfecv = pickle.load(f)
    f.close()
with open('top_100_features_from_Rforest_rfecv.pkl', 'rb') as f:
    top_100_features_from_Rforest_rfecv = pickle.load(f)
    f.close()
with open('top_100_features_from_ET_rfecv.pkl', 'rb') as f:
    top_100_features_from_ET_rfecv = pickle.load(f)
    f.close()
with open('top_100_features_from_XGB_rfecv.pkl', 'rb') as f:
    top_100_features_from_XGB_rfecv = pickle.load(f)
    f.close()

with open("DT_Model.pkl","rb") as f:
    clf_1_Decision_tree = pickle.load(f)
    f.close()
with open("RF_Model.pkl","rb") as f:
    clf_2_random_forest = pickle.load(f)
    f.close()
with open("ETR_Model.pkl","rb") as f:
    clf_3_Extratrees_regr = pickle.load(f)
    f.close()
with open("XGB_Model.pkl","rb") as f:
    clf_4_Xgboost = pickle.load(f)
    f.close()
```

```
    with open("Metamodel_top_100.pkl","rb") as f:
        final_estimator = pickle.load(f)
        f.close()

    predictions_DT_test = clf_1_Decision_tree.predict(X[top_100_features_from_DT_rfecv])
    predictions_RF_test = clf_2_random_forest.predict(X[top_100_features_from_Rforest_rfecv])
    predictions_ET_test = clf_3_Extratrees_regr.predict(X[top_100_features_from_ET_rfecv])
    predictions_XGB_test = clf_4_Xgboost.predict(X[top_100_features_from_XGB_rfecv])

    dt = pd.DataFrame()
    dt['DT'] = predictions_DT_test
    dt['RF'] = predictions_RF_test
    dt['ET'] = predictions_ET_test
    dt['XGB'] = predictions_XGB_test
    dt


    predictions = pd.DataFrame()
    predictions['ID'] = IDS
    predictions['y'] = final_estimator.predict(dt)

    return predictions
```

PREDICTIONS ON A SAMPLE TEST DATA (SINGLE OR MULTI INPUT):

In [25]:
```
a = pd.read_csv('test.csv')
a.head(10)
```

Out[25]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0	0
5	8	y	aa	ai	e	d	x	g	s	0	...	1	0	0	0	0	0	0	0	0	0
6	10	x	b	ae	d	d	x	d	y	0	...	0	0	0	0	0	1	0	0	0	0
7	11	f	s	ae	c	d	h	d	a	0	...	0	0	1	0	0	0	0	0	0	0
8	12	ap	l	s	c	d	h	j	n	0	...	0	0	0	0	0	0	0	0	0	0
9	14	o	v	as	f	d	g	f	v	0	...	0	0	0	0	0	0	0	0	0	0

10 rows × 377 columns

In [26]:
```
# reference : https://www.geeksforgeeks.org/python-get-a-list-as-input-from-user/

lst = []
n = int(input("NUMBER OF THE DATAPOINTS TO BE PREDICTED: "))
if n >1:
    print("Enter the ids of the" + str(n)+' data points')
else:
    print("Enter the id of the point")
for i in range(0, n):
    ele = int(input())
    lst.append(ele)

print(lst)
```

```
NUMBER OF THE DATAPOINTS TO BE PREDICTED: 4
Enter the ids of the4 data points
23
22
17
16
[23, 22, 17, 16]
```

In [27]:
```
X_test =a.loc[a['ID'].isin(lst)]
X_test
```

Out[27]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
11	16	ay	b	b	a	d	g	l	r	0	...	0	0	1	0	0	0	0	0	0	0
12	17	al	r	e	f	d	g	h	o	0	...	0	0	0	0	0	0	0	0	0	0
16	22	ap	l	s	c	d	g	d	h	0	...	0	0	0	0	0	0	0	0	0	0
17	23	al	r	e	f	d	f	h	o	0	...	0	0	0	0	0	0	0	0	0	0

4 rows × 377 columns

In [28]:
```
final_fun_1(X_test)
```

Out[28]:

	ID	y
0	16	95.711386
1	17	88.802784
2	22	110.510655
3	23	100.947414

In [ ]:
```

```