In [30]:

```
import os
import pandas as pd
import numpy as np
```

In [31]:

```
os.chdir("T:\\My Dir\\Downloads\\916e0d48-7-dataset\\dataset")
print(os.listdir())
test = pd.read_csv('test.csv')
train = pd.read_csv('train.csv')
```

```
['test.csv', 'train.csv']
```

In [32]:

```
test.head(2)
```

Out[32]:

	patient_id	name_of_drug	review_by_patient	drug_approved_by_UIC	number_of_times_prescr
0	163740	Mirtazapine	"I've tried a few antidepressants over th...	28-Feb-12	
1	39293	Contrave	"Contrave combines drugs that were used for al...	5-Mar-17	

In [33]:

```
train.head(2)
```

Out[33]:

	patient_id	name_of_drug	use_case_for_drug	review_by_patient	effectiveness_rating	drug_a
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	

In [34]:

```
1 test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10760 entries, 0 to 10759
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   patient_id                  10760 non-null  int64 
 1   name_of_drug                10760 non-null  object
 2   review_by_patient           10760 non-null  object
 3   drug_approved_by_UIC        10760 non-null  object
 4   number_of_times_prescribed  10760 non-null  int64 
 5   use_case_for_drug           10760 non-null  object
 6   effectiveness_rating        10760 non-null  int64 
dtypes: int64(3), object(4)
memory usage: 588.6+ KB
```

In [35]:

```
1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32165 entries, 0 to 32164
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   patient_id                  32165 non-null  int64  
 1   name_of_drug                32165 non-null  object 
 2   use_case_for_drug           32165 non-null  object 
 3   review_by_patient           32165 non-null  object 
 4   effectiveness_rating        32165 non-null  int64  
 5   drug_approved_by_UIC        32165 non-null  object 
 6   number_of_times_prescribed  32165 non-null  int64  
 7   base_score                  32165 non-null  float64
dtypes: float64(1), int64(3), object(4)
memory usage: 2.0+ MB
```

Merging text columns

In [36]:

```
1 train['review_by_patient'] = train['review_by_patient'] + train['use_case_for_drug'] +
2 test['review_by_patient'] = test['review_by_patient'] + test['use_case_for_drug'] + tes
3 
4 from gensim.parsing.preprocessing import preprocess_documents
5 
6 t1 = preprocess_documents(train['review_by_patient'])
7 t2 = preprocess_documents(test['review_by_patient'])
```

In [37]:

```
l1 = []
for i in t1:
  s1 = ''
  for j in i:
    s1 = s1 + j + ' '
  l1.append(s1)
l2 = []

for i in t2:
  s2 = ''
  for j in i:
    s2 = s2 + j + ' '
  l2.append(s2)
train['review_by_patient'] = l1
test['review_by_patient'] = l2
```

Tfidf for vectorizing the text column

In [38]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

reg = TfidfVectorizer()
text = train['review_by_patient']
test_text = test['review_by_patient']
text_to_vector = reg.fit_transform(text)
test_text_to_vector = reg.transform(test_text)
Y = train['base_score']
```

Normalizing the train column

In [42]:

```
from numpy import linalg as LA

t = text_to_vector.toarray()
l = [LA.norm(i) for i in t]

for i in range(len(l)):
  for j in k:
    if l[i] == j:
      l[i] = k.index(j) + 1
train['new_column'] = l
X_train = train[['effectiveness_rating', 'number_of_times_prescribed', 'new_column']]
X_train
```

Out[42]:

	effectiveness_rating	number_of_times_prescribed	new_column
0	9	27	1.0
1	8	192	1.0
2	5	17	1.0
3	9	37	1.0
4	2	43	1.0
...	...	...	...
32160	9	89	1.0
32161	6	0	1.0
32162	9	25	1.0
32163	8	22	1.0
32164	2	35	1.0

32165 rows × 3 columns

In [43]:

```
k = []
for i in train['new_column']:
  if i not in k:
    k.append(i)
k
```

Out[43]:

```
[1.0,
 0.9999999999999999,
 1.0000000000000002,
 0.9999999999999998,
 0.9999999999999997,
 0.9999999999999996,
 1.0000000000000004,
 0.9999999999999993,
 0.9999999999999994]
```

Normalizing the test column

In [44]:

```
t = test_text_to_vector.toarray()
l = [LA.norm(i) for i in t]
for i in range(len(l)):
  for j in k:
    if l[i] == j:
      l[i] = k.index(j) + 1
test['new_column'] = l
X_test = test[['effectiveness_rating', 'number_of_times_prescribed', 'new_column']]
X_test
```

Out[44]:

	effectiveness_rating	number_of_times_prescribed	new_column
0	10	22	4
1	9	35	1
2	4	13	1
3	7	21	1
4	2	44	1
...	...	...	...
10755	1	2	2
10756	1	18	1
10757	10	43	2
10758	8	7	1
10759	9	46	1

10760 rows × 3 columns

In [51]:

```
k = []
for i in test['new_column']:
  if i not in k:
    k.append(i)
k
```

Out[51]:

```
[4, 1, 2, 3, 5, 6, 7]
```

Model - GradientBoostingRegressor

In [52]:

```
from sklearn.ensemble import GradientBoostingRegressor

reg1 = GradientBoostingRegressor(max_depth = 8, n_estimators = 1400)
reg1.fit(X_train,Y)
reg1.feature_importances_
```

Out[52]:

```
array([0.49013426, 0.50986574, 0.        ])
```

In [53]:

```
pred = reg1.predict(X_test)
```

In [74]:

```
import csv

with open('prediction_with_basescore.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['patient_id','base_score'])
    for i in range(len(list(pred))):
        writer.writerow([test["patient_id"][i],pred[i]])
```

In [55]:

```
pred1 = reg1.predict(X_train)

from sklearn.metrics import r2_score

score = r2_score(Y,pred1)
score
```

Out[55]:

```
0.9999999997791362
```

Q. use the input to predict the base score of a certain drug in a provided case.

In [75]:

```
result = pd.read_csv('prediction_with_basescore.csv')
result.head(11)
```

Out[75]:

	patient_id	base_score
0	163740	8.757196
1	39293	8.745033
2	208087	5.952282
3	23295	5.994917
4	97013	5.361564
5	213376	7.475084
6	79865	8.358950
7	27607	6.127480
8	60341	2.234841
9	141462	6.485981
10	50229	4.980475

In [ ]:

```

```