drugactivity

March 19, 2022

0.0.1 Importing libraries and data

[1]:
```
import pandas as pd
from sklearn.metrics import f1_score
import numpy as np
import time
```

[2]:
```
train_df = pd.read_csv('train_data.txt',delimiter = "\t", header=None)
test_df = pd.read_csv('train_data.txt',delimiter = "\t", header=None)
```

[3]:
```
#extracting data from compressed format train_df
train_df[1] = train_df[1].str.replace(" ","")
train_df[1] = train_df[1].str.replace("","")[1: -1]

#extracting data from compressed format test_df
test_df[1] = test_df[1].str.replace(" ","")
test_df[1] = test_df[1].str.replace("","")[1: -1]
```

0.0.2 Features and Labels

[4]:
```
# Setting up df1 to hold x variables
df1 = train_df[1].str.split(' ', expand=True)
df1 = df1.apply(pd.to_numeric)
df1 = df1.replace(np.nan, 0)

#setting up df2 to y variable
df2 = train_df[0]
df2 = df2.apply(pd.to_numeric)
df2 = df2.replace(np.nan, 0)
```

[5]:
```
#for x
X1 = df1.iloc[:,:]

#for y
y1 = df2.values.reshape(-1,1)
```

0.0.3 Imbalanced Data

Undersampling

[6]:
```
from imblearn.under_sampling import NearMiss

undersample = NearMiss(version=1, n_neighbors=3)
X, y = undersample.fit_resample(X1, y1)
```

0.0.4 Selecting best attributes

[7]:
```
#feature selection variance method
from sklearn.feature_selection import VarianceThreshold
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
features = sel.fit_transform(X)

#feature selection Kbest
from sklearn.feature_selection import SelectKBest, chi2
x = SelectKBest(chi2, k=6500).fit_transform(features, y)
```

0.0.5 Splitting data

[8]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=20)
```

1 Models

1.0.1 Decision trees

[9]:
```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
```

[34]:
```
clf = clf.fit(x_train,y_train)

#timing our model time performance in predicting
dt_t0 = time.time()
dt_pred = clf.predict(x_test)
dt_t1 = time.time()
```

[36]:
```
print("-------------------------------------------------------------")
print("Decision trees Model F1 Score: ",f1_score(y_test, dt_pred))
print("-------------------------------------------------------------")
print("Time taken fo prediction: ",(dt_t1 - dt_t0))
print("-------------------------------------------------------------")
```

```
------------------------------------------------------
Decision trees Model F1 Score:  0.8837209302325582
------------------------------------------------------
Time taken fo prediction:  0.0011909008026123047
------------------------------------------------------
```

1.0.2 Naive Bayes

[12]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

[13]:
```
model.fit(x_train,y_train)

#timing our model time performance in predicting
nb_t0 = time.time()
nb_pred = model.predict(x_test)
nb_t1 = time.time()
```

[14]:
```
print("------------------------------------------------------")
print("Naive Bayes Model F1 Score: ",f1_score(y_test,nb_pred))
print("------------------------------------------------------")
print("Time taken fo prediction: ",(nb_t1 - nb_t0))
print("------------------------------------------------------")
```

```
------------------------------------------------------
Naive Bayes Model F1 Score:  0.8947368421052632
------------------------------------------------------
Time taken fo prediction:  0.005876064300537109
------------------------------------------------------
```

2 Prediction

Setting our input array

[15]:
```
df3 = test_df[1].str.split(' ', expand=True)
df3 = df3.apply(pd.to_numeric)
df3 = df3.replace(np.nan, 0)

#setting x_ to be used as input array for prediction
x_ = df3.iloc[:,:]
```

[16]:
```
#feature selection variance method
from sklearn.feature_selection import VarianceThreshold
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
features_ = sel.fit_transform(x_)

#feature selection Kbest method
```

```
from sklearn.feature_selection import SelectKBest, chi2
x_1 = SelectKBest(chi2, k=6500).fit_transform(features_, y1)
```

2.0.1 Decision tree predictions

[17]:
```
output_dt = clf.predict(x_1).reshape(-1,1)
output_dt = pd.DataFrame(output_dt)
#saving to a text file
output_dt.to_csv(r'dt_model.txt', header=None, index=None, sep=' ', mode='a')
```

2.0.2 Naive Bayes predictions

[18]:
```
output_nb = model.predict(x_1).reshape(-1,1)
output_nb = pd.DataFrame(output_nb)
#saving to a text file
output_nb.to_csv(r'nb_model.txt', header=None, index=None, sep=' ', mode='a')
```

3 Results

[32]:
```
output_dt.value_counts()
```

[32]:
```
0    779
1     21
dtype: int64
```

[33]:
```
output_nb.value_counts()
```

[33]:
```
0    786
1     14
dtype: int64
```

[ ]:
```

```