In [40]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import confusion_matrix,classification_report

def ANN(X_train,y_train,X_test,y_test,loss,weights):
    model=keras.Sequential([
        keras.layers.Dense(26,input_shape=(26,),activation='relu'),
        keras.layers.Dense(15,input_shape=(26,),activation='relu'),
        keras.layers.Dense(1,activation='sigmoid')

])

    model.compile(optimizer='adam',
                  loss=loss,
                  metrics=['accuracy'])
    if weights==-1:
        model.fit(X_train,y_train,epochs=100)
    else:
        model.fit(X_train,y_train,epochs=100,class_weight=weights)

    print(model.evaluate(X_test,y_test))

    y_preds=model.predict(X_test)
    y_preds=np.round(y_preds)

    print('Classification Report: \n',classification_report(y_test,y_preds))

    return y_preds

```

In [ ]:
```
y_preds=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

Undersampling Majority Class

In [42]:
```
# class count

count_class_0,count_class_1=df1.Churn.value_counts()

#Divide by class
df_class_0=df2[df2['Churn']==0]
df_class_1=df2[df2['Churn']==1]
```

In [43]:
```
count_class_0,count_class_1
```

Out[43]: (5163, 1869)

In [44]:
```
df_class_0.shape
```

Out[44]: (5163, 27)

In [45]:
```
df_class_1.shape
```

Out[45]: (1869, 27)

In [46]:
```
df_class_0_under=df_class_0.sample(count_class_1)
df_test_under=pd.concat([df_class_0_under,df_class_1],axis=0)
print('Random under- sampling:')
print(df_test_under.Churn.value_counts())
```

```
Random under- sampling:
0    1869
1    1869
Name: Churn, dtype: int64
```

In [47]:
```
X=df_test_under.drop('Churn',axis='columns')
y=df_test_under['Churn']

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=15,stratify=y)
```

In [48]:
```
y_train.value_counts()
```

Out[48]:
```
0    1495
1    1495
Name: Churn, dtype: int64
```

In [ ]:
```
y_preds=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

Method 2: Oversampling

Minority class by duplication

In [50]:
```
count_class_0,count_class_1
```

Out[50]:
```
(5163, 1869)
```

In [51]:
```
df_class_1_over=df_class_1.sample(count_class_0,replace=True)
df_test_over=pd.concat([df_class_0,df_class_1_over],axis=0)

print('Random over-sampling')
print(df_test_over.Churn.value_counts())
```

```
Random over-sampling
0    5163
1    5163
Name: Churn, dtype: int64
```

In [52]:
```
X=df_test_over.drop('Churn',axis='columns')
y=df_test_over['Churn']

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=15,stratify=y)
```

In [53]:
```
y_train.value_counts()
```

Out[53]:
```
1    4130
0    4130
Name: Churn, dtype: int64
```

In [ ]:
```
y_preds=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

Method 3: Over sampling

SMOTE (Syntheric Miniority Over Sampling Technique)

In [55]:
```
X=df2.drop('Churn',axis='columns')
y=df2['Churn']

```

In [56]:
```
y.value_counts()
```

Out[56]:
```
0    5163
1    1869
Name: Churn, dtype: int64
```

In [57]:
```
from imblearn.over_sampling import SMOTE

smote=SMOTE(sampling_strategy='minority')
X_sm,y_sm=smote.fit_resample(X,y)

```

In [58]:
```
y_sm.value_counts()
```

Out[58]:
```
0    5163
1    5163
Name: Churn, dtype: int64
```

In [59]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_sm,y_sm,test_size=0.2,random_state=15,stratify=y_
```

In [60]:
```
1 y_train.value_counts()
```

Out[60]:
```
1    4130
0    4130
Name: Churn, dtype: int64
```

In [61]:
```
1 y_test.value_counts()
```

Out[61]:
```
1    1033
0    1033
Name: Churn, dtype: int64
```

In [ ]:
```
1 y_preds=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

Method4: Use of Ensemble method with undersampling

In [86]:
```
1 df2.Churn.value_counts()
```

Out[86]:
```
0    5163
1    1869
Name: Churn, dtype: int64
```

In [88]:
```
1 X=df2.drop('Churn',axis='columns')
2 y=df2['Churn']
3
```

In [89]:
```
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=15,stratify=y)
```

In [90]:
```
y_train.value_counts()
```

Out[90]:
```
0    4130
1    1495
Name: Churn, dtype: int64
```

In [91]:
```
4130/3
```

Out[91]:
```
1376.6666666666667
```

In [92]:
```
df3=X_train.copy()
df3['Churn']=y_train
```

In [93]:
```
df3_class_0=df3[df3.Churn==0]
df3_class_1=df3[df3.Churn==1]
```

In [94]:
```
df3_class_0.shape,df3_class1.shape
```

Out[94]:
```
((4130, 27), (1495, 27))
```

In [99]:
```
def get_train_batch(df_majority,df_minority,start,end):


    df_train=pd.concat([df_majority[start:end],df_minority],axis=0)

    X_train=df_train.drop('Churn',axis='columns')
    y_train=df_train.Churn

    return X_train,y_train
```

In [ ]:
```
X_train,y_train=get_train_batch(df3_class_0,df3_class_1,0,1495)
y_pred1=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

In [ ]:
```
X_train,y_train=get_train_batch(df3_class_0,df3_class_1,1495,2990)
y_pred2=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

In [ ]:
```
X_train,y_train=get_train_batch(df3_class_0,df3_class_1,2990,4130)
y_pred3=ANN(X_train,y_train,X_test,y_test,'binary_crossentropy',-1)
```

In [103]:
```
vote1=0
vote2=0
vote3=1
vote1+vote2+vote3
```

Out[103]: 1

In [104]:
```
vote1=0
vote2=1
vote3=1
vote1+vote2+vote3
```

Out[104]: 2

In [113]:
```
y_pred_final=y_pred1.copy()

for i in range(len(y_pred1)):
        n_ones=y_pred1[i]+y_pred2[i]+y_pred3[i]
        if n_ones>1:
            y_pred_final[i]=1
        else:
            y_pred_final[i]=0
```

In [114]:
```
print(classification_report(y_test,y_pred_final))
```

```
              precision    recall  f1-score   support

           0       0.90      0.66      0.76      1033
           1       0.46      0.81      0.59       374

    accuracy                           0.70      1407
   macro avg       0.68      0.73      0.67      1407
weighted avg       0.79      0.70      0.72      1407
```

In [ ]:
```
Thank you ~~
```