محمد کاظم رجبی

this projcet contain 2 data-set the fitst one is train data set and the second is test data set

Train dataset has 590540 rows and 434 columns.

Test dataset has 506691 rows and 433 columns.

the number of columns is 434 so i should dercrease the number thus i use groupby (also you can use pca)and also there are 414 columns with missing data.

i detect the columns has many null and nan values and also i ignore the column has big number ,then i seperate the is fruad column.

with one hot encoder convert the categorical data to numbers and after all filters finally i get 158 columns.

I detect column that have nan values and count the nan values and omit the columns that has high number of nan for example transaction id has 3444 nan values,therefore i omit this colmn.

After omiting all these kind of column i get 48 columns.Next id ignore the transition column because it is not important and i divide the data set to 2 different category.

I consider the is fruad column as lable.After that i make validation data set for preventing over fiting and use standardscaler for feature scaling.

I test different models with different layers and functions to achihve higer accuracy.It means with different size of latent space and nodes in various outencoders.I append all models in github and follow the code file to see the results.

Model 1: model = tf.keras.models.Sequential()

#start encoder

model.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

model.add(tf.keras.layers.Dense(units=35, activation='relu'))

model.add(tf.keras.layers.Dense(units=20, activation='relu'))

#end of encoder start code later

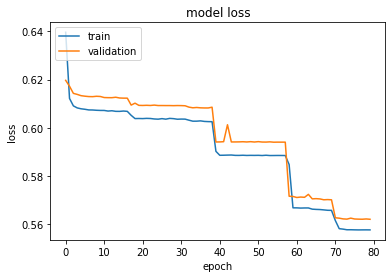
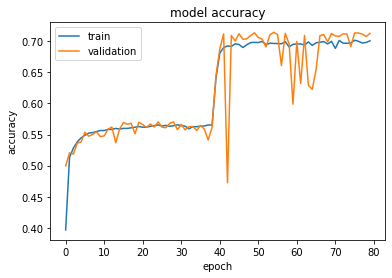
model.add(tf.keras.layers.Dense(units=15, activation='relu'))

#start decoder later

model.add(tf.keras.layers.Dense(units=20, activation='relu'))

model.add(tf.keras.layers.Dense(units=35, activation='relu'))

model.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

Model 2:

model2 = tf.keras.models.Sequential()

#start encoder later

model2.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

model2.add(tf.keras.layers.Dense(units=35, activation='relu'))

model2.add(tf.keras.layers.Dense(units=15, activation='relu'))

model2.add(tf.keras.layers.Dense(units=10, activation='relu'))

model2.add(tf.keras.layers.Dense(units=5, activation='relu'))

#end of encoder and code later

model2.add(tf.keras.layers.Dense(units=10, activation='relu'))

model2.add(tf.keras.layers.Dense(units=15, activation='relu'))

model2.add(tf.keras.layers.Dense(units=35, activation='relu'))

model2.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

model2.compile(metrics=['accuracy'],

                    loss='mean\_squared\_error',

                    optimizer='adam')

model 3:

model3 = tf.keras.models.Sequential()

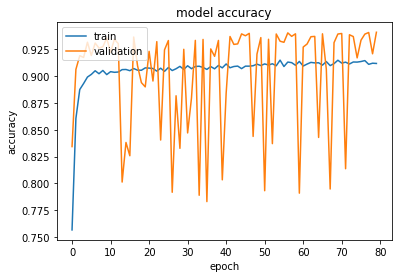
#start encoder

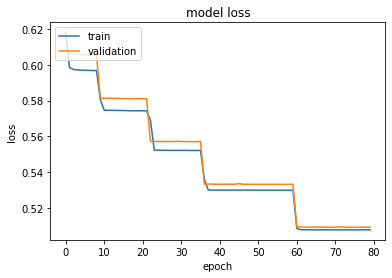
model3.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

#end of encoder-code layer

model3.add(tf.keras.layers.Dense(units=35, activation='relu'))

model3.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))





Model 4:

model4 = tf.keras.models.Sequential()

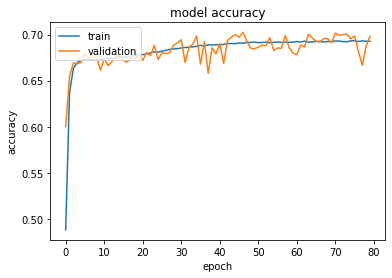
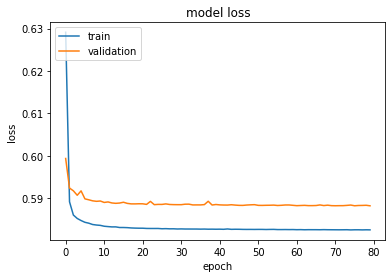
#start encoder layer

model4.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

#end of encoder -code layer

model4.add(tf.keras.layers.Dense(units=10, activation='relu'))

model4.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

Model 5:

model5= tf.keras.models.Sequential()

#start encoder layer

model5.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

#end of encoder -code layer

model5.add(tf.keras.layers.Dense(units=15, activation='relu'))

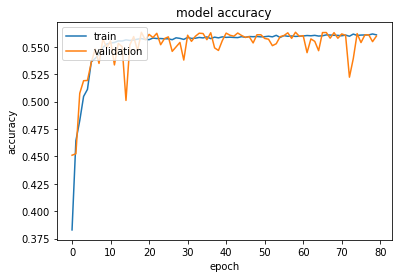
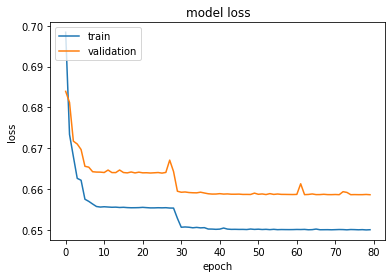
#decoder layer

model5.add(tf.keras.layers.Dense(units=25, activation='relu'))

model5.add(tf.keras.layers.Dense(units=35, activation='relu'))

model5.add(tf.keras.layers.Dense(units=40, activation='relu'))

model5.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

Model 6:

model6= tf.keras.models.Sequential()

#start encoder

model6.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

model6.add(tf.keras.layers.Dense(units=40, activation='relu'))

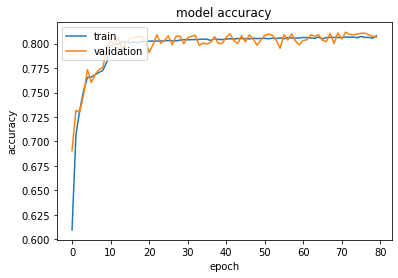
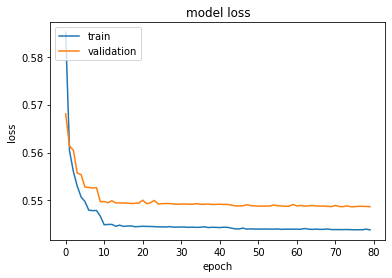
model6.add(tf.keras.layers.Dense(units=35, activation='relu'))

#end of encdoer -code layer

model6.add(tf.keras.layers.Dense(units=25, activation='relu'))

#decodr layer

model6.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

Model 7:

model7= tf.keras.models.Sequential()

#start encoder

model7.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

model7.add(tf.keras.layers.Dropout(rate=0.3))

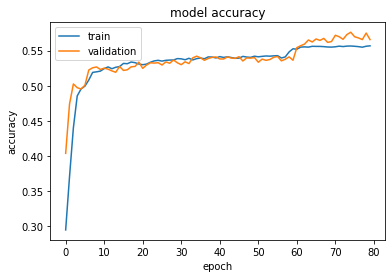
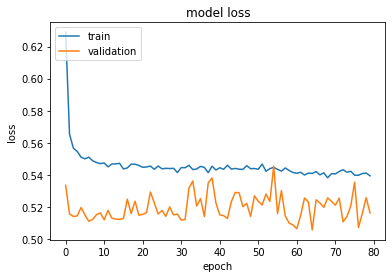
#end of encoder-code layer

model7.add(tf.keras.layers.Dense(units=25, activation='relu'))

#decoder layer

model7.add(tf.keras.layers.Dropout(rate=0.3))

model7.add(tf.keras.layers.Dense(units=inputshape, activation='relu'))

The best model is model 3 so I use this model to detect fraud.I use mean error(between actual value and predicted value) to find fraud

mean\_error = np.mean(np.power(testdata - predictions, 2), axis=1)

and the I join that with is fraud column.I use aoc and roc for classification (if the number is equal to 50% it is not good and it act like random coin).  
with changing and test different values of treshold we can classify .(the best threshold is 0.4)