loan_data_proj

January 22, 2022

1 Load Packages

[87]:
```
# Ignore the warnings

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Model, layers
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.layers import LSTM

import warnings
warnings.filterwarnings('ignore')

import numpy as np
from numpy import expand_dims
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,␣
↪classification_report
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.datasets import load_digits

from tensorflow.keras.layers import Dense,Activation
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
```

[88]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Model, layers
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.layers import LSTM
```

2 Load Dataset

[89]:
```
# Mount the drive
df = pd.read_csv("loan_data.csv")



df.head()
```

[89]:
```
   credit.policy             purpose  int.rate  installment  log.annual.inc  \
0              1  debt_consolidation    0.1189       829.10       11.350407
1              1         credit_card    0.1071       228.22       11.082143
2              1  debt_consolidation    0.1357       366.86       10.373491
3              1  debt_consolidation    0.1008       162.34       11.350407
4              1         credit_card    0.1426       102.92       11.299732

     dti  fico  days.with.cr.line  revol.bal  revol.util  inq.last.6mths  \
0  19.48   737       5639.958333      28854        52.1               0
1  14.29   707       2760.000000      33623        76.7               0
2  11.63   682       4710.000000       3511        25.6               1
3   8.10   712       2699.958333      33667        73.2               1
4  14.97   667       4066.000000       4740        39.5               0

   delinq.2yrs  pub.rec  not.fully.paid
0            0        0               0
1            0        0               0
2            0        0               0
3            0        0               0
4            1        0               0
```

[90]:
```
#credit_data['class'] = credit_data['class']-1
df.shape
```

[90]:
```
(9578, 14)
```

[91]:
```
# Dataset class distribution
df["not.fully.paid"].value_counts(normalize =  True)
```

[91]:
```
0    0.839946
1    0.160054
Name: not.fully.paid, dtype: float64
```

[98]:
```
df.info()
df = df.drop(["not.fully.paid"], axis =1)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
```

```
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit.policy      9578 non-null   int64
 1   purpose            9578 non-null   object
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   days.with.cr.line  9578 non-null   float64
 7   revol.bal          9578 non-null   int64
 8   revol.util         9578 non-null   float64
 9   inq.last.6mths     9578 non-null   int64
 10  delinq.2yrs        9578 non-null   int64
 11  pub.rec            9578 non-null   int64
 12  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(6), object(1)
memory usage: 972.9+ KB
```

3 1. Data Preprocessing - EDA

[101]:
```
import numpy as np

# Create correlation matrix
corr_matrix = df.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
 ↪bool))

# Find features with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.50)]

# Drop features
df.drop(to_drop, axis=1, inplace=True)
```

[102]:
```
df.shape
```

[102]:
```
(9578, 11)
```

[103]:
```
df.isnull().sum()
```

[103]:
```
credit.policy        0
purpose              0
int.rate             0
```

```
installment          0
log.annual.inc       0
dti                  0
days.with.cr.line    0
revol.bal            0
revol.util           0
delinq.2yrs          0
pub.rec              0
dtype: int64
```

[ ]:

No missing values present

4 Target Column

[104]:
```
y = loan_data["not.fully.paid"]
y.shape
```

[104]:
```
(9578,)
```

4.1 Encoding Categorical Data

[105]:
```
cat_cols = df.select_dtypes(include="object").columns.tolist()
print(cat_cols)
```

```
['purpose']
```

[106]:
```
cat_data = df[cat_cols]
cat_data.head()
```

[106]:

	purpose
0	debt_consolidation
1	credit_card
2	debt_consolidation
3	debt_consolidation
4	credit_card

[107]:
```
#convert categorical variable into dummy
cat_df = pd.get_dummies(cat_data['purpose'],prefix= 'purpose',drop_first=True)
print(cat_data.shape)
print(cat_df.shape)
```

```
(9578, 1)
(9578, 6)
```

[108]:
```
df_encoded = pd.concat([cat_df, df.drop(cat_cols,axis=1)],axis=1)
df_encoded.shape
```

[108]:
```
(9578, 16)
```

[109]:
```
df_encoded.head()
```

[109]:

	purpose_credit_card	purpose_debt_consolidation	purpose_educational	\
0	0	1	0	
1	1	0	0	
2	0	1	0	
3	0	1	0	
4	1	0	0	

	purpose_home_improvement	purpose_major_purchase	purpose_small_business	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

	credit.policy	int.rate	installment	log.annual.inc	dti	\
0	1	0.1189	829.10	11.350407	19.48	
1	1	0.1071	228.22	11.082143	14.29	
2	1	0.1357	366.86	10.373491	11.63	
3	1	0.1008	162.34	11.350407	8.10	
4	1	0.1426	102.92	11.299732	14.97	

	days.with.cr.line	revol.bal	revol.util	delinq.2yrs	pub.rec
0	5639.958333	28854	52.1	0	0
1	2760.000000	33623	76.7	0	0
2	4710.000000	3511	25.6	0	0
3	2699.958333	33667	73.2	0	0
4	4066.000000	4740	39.5	1	0

[110]:
```
cols = df_encoded.columns.tolist()
for col in cols:
    if len(set(df_encoded[col].tolist())) < 2:
        df_encoded.drop(col)
```

[111]:
```
df_encoded.shape
```

[111]:
```
(9578, 16)
```

[ ]:
```
```

4.2 Train Test Split

[112]:
```
# Need to scale the continuous variables and combine for later processing
df_x_train, df_x_test, df_y_train,df_y_test = train_test_split(df_encoded,
                                                               y,
                                                               train_size=0.7,
                                                               random_state=42)

print("Df x train shape", df_x_train.shape)
print("Df x test shape", df_x_test.shape)
print("Df y train shape", df_y_train.shape)
print("Df y test shape", df_y_test.shape)
```

```
Df x train shape (6704, 16)
Df x test shape (2874, 16)
Df y train shape (6704,)
Df y test shape (2874,)
```

[114]:
```
print(pd.Series(df_y_train).value_counts(normalize=True))
print(pd.Series(df_y_test).value_counts(normalize=True))
```

```
0    0.840841
1    0.159159
Name: not.fully.paid, dtype: float64
0    0.837857
1    0.162143
Name: not.fully.paid, dtype: float64
```

[115]:
```
df_x_train, df_x_test, df_y_train,df_y_test = train_test_split(df_encoded,
                                                               y, stratify = y,
                                                               train_size=0.7,
                                                               random_state=42)
```

[117]:
```
print(pd.Series(df_y_train).value_counts(normalize=True))
print(pd.Series(df_y_test).value_counts(normalize=True))
```

```
0    0.839946
1    0.160054
Name: not.fully.paid, dtype: float64
0    0.839944
1    0.160056
Name: not.fully.paid, dtype: float64
```

[119]:
```
df_x_train.head()
df_x_train.shape
```

[119]: (6704, 16)

[120]:
```
df_x_test.head()
```

[120]:

	purpose_credit_card	purpose_debt_consolidation	purpose_educational	\
1102	0	0	0	
5350	0	1	0	
3323	0	0	0	
2147	0	0	0	
7471	0	1	0	

	purpose_home_improvement	purpose_major_purchase	\
1102	0	0	
5350	0	0	
3323	1	0	
2147	0	0	
7471	0	0	

	purpose_small_business	credit.policy	int.rate	installment	\
1102	0	1	0.1008	51.69	
5350	0	1	0.0894	317.72	
3323	0	1	0.1221	399.78	
2147	1	1	0.0963	77.03	
7471	0	1	0.0988	483.16	

	log.annual.inc	dti	days.with.cr.line	revol.bal	revol.util	\
1102	10.714418	21.57	2309.958333	1949	48.7	
5350	11.050890	13.64	3989.958333	10326	27.8	
3323	11.018564	18.89	4109.958333	17710	60.7	
2147	10.043249	2.14	5340.041667	948	13.5	
7471	11.512925	4.12	6750.041667	8361	12.1	

	delinq.2yrs	pub.rec
1102	0	0
5350	0	0
3323	0	0
2147	0	1
7471	0	0

4.3 Scaling Numercial Data

[121]:
```
# remove the class column from credit data
num_cols = df.iloc[:,:-1].select_dtypes(include = np.number).columns.tolist()
num_cols
```

[121]:
```
['credit.policy',
 'int.rate',
 'installment',
```

```
 'log.annual.inc',
 'dti',
 'days.with.cr.line',
 'revol.bal',
 'revol.util',
 'delinq.2yrs']
```

[122]:
```
# Trianing numerical data
train_num_data = df_x_train[num_cols]
train_num_data.head()
```

[122]:

	credit.policy	int.rate	installment	log.annual.inc	dti	\
3407	1	0.1347	72.94	11.338572	15.01	
6212	1	0.1913	557.24	10.915088	2.38	
260	1	0.0743	180.23	10.628618	2.32	
3558	1	0.1253	281.12	11.225243	4.16	
6313	1	0.1322	287.31	11.225243	7.73	

	days.with.cr.line	revol.bal	revol.util	delinq.2yrs
3407	4050.000000	27385	51.4	0
6212	4740.000000	491	98.2	0
260	2610.041667	2650	6.6	0
3558	5699.958333	13573	30.8	0
6313	7950.000000	4167	19.8	1

4.3.1 Train Data fit_transformation

[123]:
```
scaler = MinMaxScaler()
train_num_data_scaled = scaler.fit_transform(train_num_data)
train_num_df_scaled = pd.DataFrame(train_num_data_scaled,
                                   index=train_num_data.index,
                                   columns=num_cols)
train_num_df_scaled.head()
```

[123]:

	credit.policy	int.rate	installment	log.annual.inc	dti	\
3407	1.0	0.477621	0.062854	0.571577	0.501001	
6212	1.0	0.839514	0.594374	0.507729	0.079439	
260	1.0	0.091432	0.180605	0.464538	0.077437	
3558	1.0	0.417519	0.291332	0.554490	0.138852	
6313	1.0	0.461637	0.298125	0.554490	0.258011	

	days.with.cr.line	revol.bal	revol.util	delinq.2yrs
3407	0.221648	0.022682	0.431933	0.000000
6212	0.261167	0.000407	0.825210	0.000000
260	0.139176	0.002195	0.055462	0.000000
3558	0.316148	0.011242	0.258824	0.000000

```
6313           0.445017    0.003451     0.166387      0.076923
```

4.3.2 Test data transformation

[124]:
```
# Trianing numerical data
test_num_data = df_x_test[num_cols]
test_num_data.head()
```

[124]:
```
      credit.policy  int.rate  installment  log.annual.inc    dti  \
1102              1    0.1008        51.69       10.714418  21.57
5350              1    0.0894       317.72       11.050890  13.64
3323              1    0.1221       399.78       11.018564  18.89
2147              1    0.0963        77.03       10.043249   2.14
7471              1    0.0988       483.16       11.512925   4.12

      days.with.cr.line  revol.bal  revol.util  delinq.2yrs
1102        2309.958333       1949        48.7            0
5350        3989.958333      10326        27.8            0
3323        4109.958333      17710        60.7            0
2147        5340.041667        948        13.5            0
7471        6750.041667       8361        12.1            0
```

[125]:
```
test_num_data_scaled = scaler.transform(test_num_data)
test_num_df_scaled = pd.DataFrame(test_num_data_scaled,
                                  index=test_num_data.index,
                                  columns=num_cols)
test_num_df_scaled.head()
```

[125]:
```
      credit.policy  int.rate  installment  log.annual.inc       dti  \
1102            1.0  0.260870     0.039532        0.477474  0.719960
5350            1.0  0.187980     0.331501        0.528203  0.455274
3323            1.0  0.397059     0.421562        0.523329  0.630507
2147            1.0  0.232097     0.067343        0.376282  0.071429
7471            1.0  0.248082     0.513071        0.597864  0.137517

      days.with.cr.line  revol.bal  revol.util  delinq.2yrs
1102           0.121989   0.001614    0.409244          0.0
5350           0.218209   0.008553    0.233613          0.0
3323           0.225082   0.014668    0.510084          0.0
2147           0.295534   0.000785    0.113445          0.0
7471           0.376290   0.006925    0.101681          0.0
```

4.4 Final Train and Test Datasets

[126]:
```
X_train_df = pd.concat([train_num_df_scaled, df_x_train.
 ↪drop(num_cols,axis=1)],axis=1)
X_test_df = pd.concat([test_num_df_scaled, df_x_test.
 ↪drop(num_cols,axis=1)],axis=1)
y_train_df = df_y_train
y_test_df = df_y_test
```

[127]:
```
print("X train shape", X_train_df.shape)
print("X test shape", X_test_df.shape)
print("y train shape", y_train_df.shape)
print("y test shape", y_test_df.shape)
```

```
X train shape (6704, 16)
X test shape (2874, 16)
y train shape (6704,)
y test shape (2874,)
```

4.5 Coverting data to Array

[128]:
```
X_train = np.array(X_train_df)
X_test = np.array(X_test_df)
y_train = expand_dims(y_train_df, axis=1)
y_test = expand_dims(y_test_df, axis=1)

print("X train shape", X_train.shape)
print("X test shape", X_test.shape)
print("y train shape", y_train.shape)
print("y test shape", y_test.shape)
```

```
X train shape (6704, 16)
X test shape (2874, 16)
y train shape (6704, 1)
y test shape (2874, 1)
```

[129]:
```
print(X_train[:1])
print(y_test[:1])
```

```
[[1.         0.47762148 0.06285394 0.57157686 0.50100134 0.22164816
  0.02268174 0.43193277 0.         0.         0.         0.
  0.         0.         0.         0.        ]]
[[0]]
```

5 2. Define & Compile Keras Model

[130]:
```
def DNN_Binary_Classification_KS(_input_dim,metrics):

    # Layer 1
    model = Sequential()
    model.add(Dense(300,activation='relu', input_shape=(_input_dim,)))

    # Layer 2
    model.add(Dense(300,activation='relu'))

    # Layer 3
    model.add(Dense(1, activation='sigmoid'))

    sgd_opt = SGD(lr=0.01)

    print(model.summary)

    # Model compilation
    model.compile(loss='binary_crossentropy',optimizer = sgd_opt,␣
↪metrics=metrics)

    return model
```

5.1 3. Fit Keras Model

[138]:
```
# Model training
input_dim = 16
training_epochs = 50
batch_size = 32
metrics = ["accuracy"]
valid_set = (X_test, y_test)
```

[139]:
```
DNN_Binary_Classification_KS(16,metrics).summary()
```

```
<bound method Network.summary of
<tensorflow.python.keras.engine.sequential.Sequential object at 0x7f6cc861a850>>
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 300)               5100
_________________________________________________________________
dense_22 (Dense)             (None, 300)               90300
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 301
```

```
=================================================================
Total params: 95,701
Trainable params: 95,701
Non-trainable params: 0
_________________________________________________________________
```

[140]:
```
%%time
bin_class_model = DNN_Binary_Classification_KS(input_dim,metrics)

# fit the model
history = bin_class_model.fit(X_train,y_train,
                              validation_data = valid_set,
                              batch_size=batch_size,
                              epochs=training_epochs)
```

```
<bound method Network.summary of
<tensorflow.python.keras.engine.sequential.Sequential object at 0x7f6cc86bb550>>
Epoch 1/50
210/210 [==============================] - 1s 2ms/step - loss: 0.4828 -
accuracy: 0.8399 - val_loss: 0.4366 - val_accuracy: 0.8399
Epoch 2/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4352 -
accuracy: 0.8399 - val_loss: 0.4330 - val_accuracy: 0.8399
Epoch 3/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4322 -
accuracy: 0.8399 - val_loss: 0.4308 - val_accuracy: 0.8399
Epoch 4/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4297 -
accuracy: 0.8399 - val_loss: 0.4290 - val_accuracy: 0.8399
Epoch 5/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4276 -
accuracy: 0.8399 - val_loss: 0.4275 - val_accuracy: 0.8399
Epoch 6/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4258 -
accuracy: 0.8399 - val_loss: 0.4263 - val_accuracy: 0.8399
Epoch 7/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4242 -
accuracy: 0.8399 - val_loss: 0.4252 - val_accuracy: 0.8399
Epoch 8/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4227 -
accuracy: 0.8399 - val_loss: 0.4244 - val_accuracy: 0.8399
Epoch 9/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4216 -
accuracy: 0.8399 - val_loss: 0.4237 - val_accuracy: 0.8399
Epoch 10/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4206 -
accuracy: 0.8399 - val_loss: 0.4231 - val_accuracy: 0.8399
Epoch 11/50
```

```
210/210 [==============================] - 0s 2ms/step - loss: 0.4197 -
accuracy: 0.8399 - val_loss: 0.4226 - val_accuracy: 0.8399
Epoch 12/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4189 -
accuracy: 0.8399 - val_loss: 0.4222 - val_accuracy: 0.8399
Epoch 13/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4180 -
accuracy: 0.8399 - val_loss: 0.4219 - val_accuracy: 0.8399
Epoch 14/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4177 -
accuracy: 0.8399 - val_loss: 0.4215 - val_accuracy: 0.8399
Epoch 15/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4171 -
accuracy: 0.8399 - val_loss: 0.4212 - val_accuracy: 0.8399
Epoch 16/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4166 -
accuracy: 0.8399 - val_loss: 0.4210 - val_accuracy: 0.8399
Epoch 17/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4161 -
accuracy: 0.8399 - val_loss: 0.4208 - val_accuracy: 0.8399
Epoch 18/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4157 -
accuracy: 0.8399 - val_loss: 0.4206 - val_accuracy: 0.8399
Epoch 19/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4154 -
accuracy: 0.8399 - val_loss: 0.4204 - val_accuracy: 0.8399
Epoch 20/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4151 -
accuracy: 0.8399 - val_loss: 0.4204 - val_accuracy: 0.8399
Epoch 21/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4149 -
accuracy: 0.8399 - val_loss: 0.4202 - val_accuracy: 0.8399
Epoch 22/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4147 -
accuracy: 0.8399 - val_loss: 0.4201 - val_accuracy: 0.8399
Epoch 23/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4142 -
accuracy: 0.8399 - val_loss: 0.4200 - val_accuracy: 0.8399
Epoch 24/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4141 -
accuracy: 0.8399 - val_loss: 0.4202 - val_accuracy: 0.8399
Epoch 25/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4139 -
accuracy: 0.8399 - val_loss: 0.4203 - val_accuracy: 0.8399
Epoch 26/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4139 -
accuracy: 0.8399 - val_loss: 0.4200 - val_accuracy: 0.8399
Epoch 27/50
```

```
210/210 [==============================] - 0s 2ms/step - loss: 0.4137 -
accuracy: 0.8399 - val_loss: 0.4198 - val_accuracy: 0.8399
Epoch 28/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4136 -
accuracy: 0.8399 - val_loss: 0.4198 - val_accuracy: 0.8399
Epoch 29/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4133 -
accuracy: 0.8399 - val_loss: 0.4197 - val_accuracy: 0.8399
Epoch 30/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4131 -
accuracy: 0.8399 - val_loss: 0.4207 - val_accuracy: 0.8399
Epoch 31/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4131 -
accuracy: 0.8399 - val_loss: 0.4197 - val_accuracy: 0.8399
Epoch 32/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4129 -
accuracy: 0.8399 - val_loss: 0.4200 - val_accuracy: 0.8399
Epoch 33/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4129 -
accuracy: 0.8399 - val_loss: 0.4196 - val_accuracy: 0.8399
Epoch 34/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4126 -
accuracy: 0.8399 - val_loss: 0.4199 - val_accuracy: 0.8399
Epoch 35/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4127 -
accuracy: 0.8399 - val_loss: 0.4201 - val_accuracy: 0.8399
Epoch 36/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4127 -
accuracy: 0.8399 - val_loss: 0.4195 - val_accuracy: 0.8399
Epoch 37/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4125 -
accuracy: 0.8399 - val_loss: 0.4197 - val_accuracy: 0.8399
Epoch 38/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4126 -
accuracy: 0.8399 - val_loss: 0.4195 - val_accuracy: 0.8403
Epoch 39/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4124 -
accuracy: 0.8399 - val_loss: 0.4197 - val_accuracy: 0.8399
Epoch 40/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4123 -
accuracy: 0.8399 - val_loss: 0.4195 - val_accuracy: 0.8399
Epoch 41/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4123 -
accuracy: 0.8401 - val_loss: 0.4194 - val_accuracy: 0.8399
Epoch 42/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4122 -
accuracy: 0.8399 - val_loss: 0.4194 - val_accuracy: 0.8399
Epoch 43/50
```

```
210/210 [==============================] - 0s 2ms/step - loss: 0.4121 -
accuracy: 0.8399 - val_loss: 0.4193 - val_accuracy: 0.8399
Epoch 44/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4121 -
accuracy: 0.8399 - val_loss: 0.4193 - val_accuracy: 0.8399
Epoch 45/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4121 -
accuracy: 0.8399 - val_loss: 0.4193 - val_accuracy: 0.8399
Epoch 46/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4119 -
accuracy: 0.8399 - val_loss: 0.4195 - val_accuracy: 0.8399
Epoch 47/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4119 -
accuracy: 0.8398 - val_loss: 0.4193 - val_accuracy: 0.8399
Epoch 48/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4118 -
accuracy: 0.8396 - val_loss: 0.4198 - val_accuracy: 0.8403
Epoch 49/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4117 -
accuracy: 0.8399 - val_loss: 0.4192 - val_accuracy: 0.8403
Epoch 50/50
210/210 [==============================] - 0s 2ms/step - loss: 0.4117 -
accuracy: 0.8401 - val_loss: 0.4191 - val_accuracy: 0.8403
CPU times: user 35.9 s, sys: 4.58 s, total: 40.5 s
Wall time: 21.7 s
```

5.2 5. Predictions

[ ]:
```
# y_train_pred = bin_class_model.predict(X_train)
# y_train_pred = np.argmax(bin_class_model.predict(X_train), axis=-1)
# y_train_class = np.argmax(y_train_pred,axis=1)
# y_test_pred = bin_class_model.predict(X_test)
# y_test_class = np.argmax(y_test_pred,axis=1)
# y_test_pred = np.argmax(bin_class_model.predict(X_test), axis=-1)
```

[141]:
```
y_train_pred =  np.where(bin_class_model.predict(X_train) > 0.5,1,0)
y_test_pred =  np.where(bin_class_model.predict(X_test) > 0.5,1,0)
print(type(y_train_pred))
print(type(y_test_pred))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

[142]:
```
# training accuracy
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support
```

```
              0       0.84      1.00      0.91      5631
              1       0.50      0.00      0.00      1073

       accuracy                           0.84      6704
      macro avg       0.67      0.50      0.46      6704
   weighted avg       0.79      0.84      0.77      6704
```

[143]:
```
# test accuracy
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      1.00      0.91      2414
           1       0.67      0.00      0.01       460

    accuracy                           0.84      2874
   macro avg       0.75      0.50      0.46      2874
weighted avg       0.81      0.84      0.77      2874
```

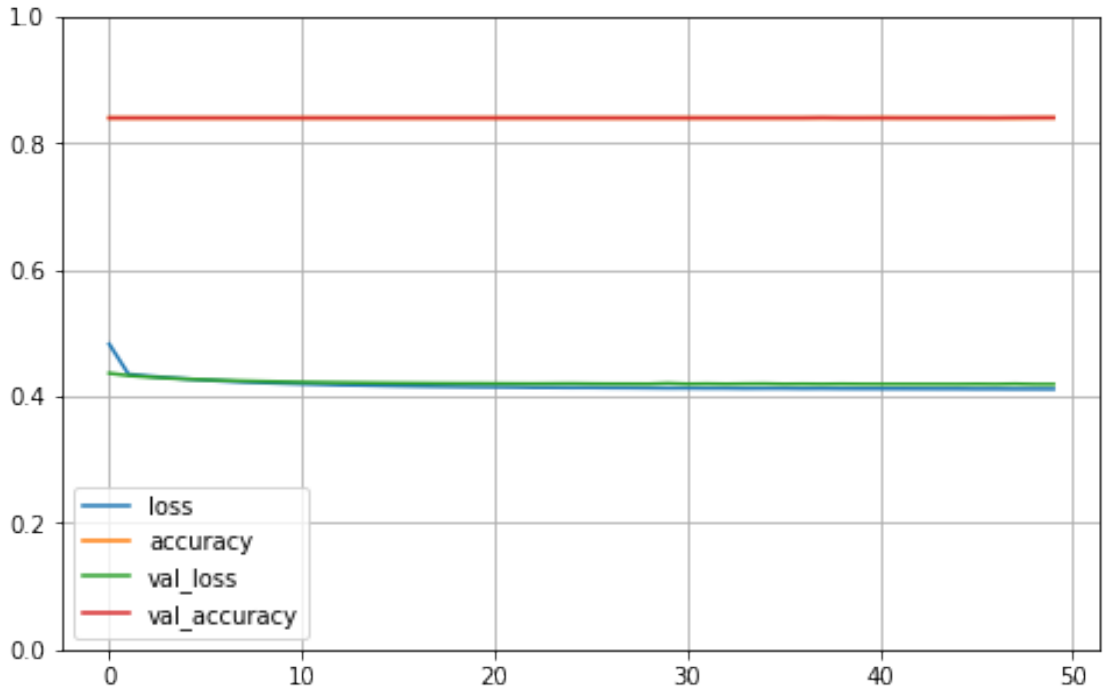
[144]:
```
pd.DataFrame(history.history)
```

[144]:

	loss	accuracy	val_loss	val_accuracy
0	0.482841	0.839946	0.436627	0.839944
1	0.435226	0.839946	0.432953	0.839944
2	0.432248	0.839946	0.430770	0.839944
3	0.429730	0.839946	0.428968	0.839944
4	0.427574	0.839946	0.427488	0.839944
5	0.425757	0.839946	0.426328	0.839944
6	0.424192	0.839946	0.425215	0.839944
7	0.422744	0.839946	0.424416	0.839944
8	0.421632	0.839946	0.423692	0.839944
9	0.420637	0.839946	0.423065	0.839944
10	0.419738	0.839946	0.422558	0.839944
11	0.418890	0.839946	0.422183	0.839944
12	0.418013	0.839946	0.421855	0.839944
13	0.417669	0.839946	0.421527	0.839944
14	0.417099	0.839946	0.421218	0.839944
15	0.416556	0.839946	0.420979	0.839944
16	0.416116	0.839946	0.420778	0.839944
17	0.415744	0.839946	0.420593	0.839944
18	0.415392	0.839946	0.420445	0.839944
19	0.415063	0.839946	0.420443	0.839944
20	0.414905	0.839946	0.420246	0.839944
21	0.414651	0.839946	0.420131	0.839944

```
22  0.414180  0.839946  0.420047      0.839944
23  0.414079  0.839946  0.420181      0.839944
24  0.413885  0.839946  0.420313      0.839944
25  0.413902  0.839946  0.420005      0.839944
26  0.413654  0.839946  0.419824      0.839944
27  0.413552  0.839946  0.419794      0.839944
28  0.413296  0.839946  0.419710      0.839944
29  0.413070  0.839946  0.420719      0.839944
30  0.413114  0.839946  0.419666      0.839944
31  0.412914  0.839946  0.419997      0.839944
32  0.412899  0.839946  0.419606      0.839944
33  0.412604  0.839946  0.419940      0.839944
34  0.412740  0.839946  0.420092      0.839944
35  0.412739  0.839946  0.419524      0.839944
36  0.412457  0.839946  0.419723      0.839944
37  0.412561  0.839946  0.419456      0.840292
38  0.412355  0.839946  0.419741      0.839944
39  0.412284  0.839946  0.419480      0.839944
40  0.412264  0.840095  0.419391      0.839944
41  0.412233  0.839946  0.419405      0.839944
42  0.412145  0.839946  0.419345      0.839944
43  0.412074  0.839946  0.419339      0.839944
44  0.412072  0.839946  0.419277      0.839944
45  0.411886  0.839946  0.419519      0.839944
46  0.411909  0.839797  0.419288      0.839944
47  0.411784  0.839648  0.419788      0.840292
48  0.411745  0.839946  0.419175      0.840292
49  0.411701  0.840095  0.419138      0.840292
```

[145]:
```
import matplotlib.pyplot as plt
import seaborn as sns
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



5.3 6. Save Model

[146]:
```
import time
import os

def saveModel_path(model_dir="Deep_Learning_Tutorial"):
    os.makedirs(model_dir, exist_ok=True)
    fileName = time.strftime("loan_Model_%Y_%m_%d_%H_%M_%S_.h5")
    model_path = os.path.join(model_dir, fileName)
    print(f"your model will be saved at the following location\n{model_path}")
    return model_path
```