```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
#import libraries
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# Load data
data = pd.read_csv('/content/gdrive/MyDrive/Loan/loan_data (1).csv')
data.head()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CI
0	100002	1	Cash loans	M	N	Y	
1	100003	0	Cash loans	F	N	N	
2	100004	0	Revolving loans	M	Y	Y	
3	100006	0	Cash loans	F	N	Y	
4	100007	0	Cash loans	M	N	Y	

5 rows × 122 columns

```
data.tail()
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	C
307506	456251	0	Cash loans	M	N	N	
307507	456252	0	Cash loans	F	N	Y	
307508	456253	0	Cash loans	F	N	Y	
307509	456254	1	Cash loans	F	N	Y	
307510	456255	0	Cash loans	F	N	N	

5 rows × 122 columns

```
data.shape
```

```
(307511, 122)
```

```
data.TARGET.value_counts()
```

```
0    282686
1     24825
Name: TARGET, dtype: int64
```

```
data.TARGET.value_counts()*100/data.shape[0]
```

```
0    91.927118
1     8.072882
Name: TARGET, dtype: float64
```

```
num_defaults = data['TARGET'].sum()

# calculate the percentage of defaulters
percentage_defaults = num_defaults / len(data) * 100

print("Percentage of defaults:", percentage_defaults)
```

```
Percentage of defaults: 8.072881945686495
```

```
numerical_feature_columns = list(data._get_numeric_data().columns)
numerical_feature_columns
```

```
 'YEARS_BEGINEXPLUATATION_MODE',
 'YEARS_BUILD_MODE',
 'COMMONAREA_MODE',
 'ELEVATORS_MODE',
 'ENTRANCES_MODE',
 'FLOORSMAX_MODE',
 'FLOORSMIN_MODE',
 'LANDAREA_MODE',
 'LIVINGAPARTMENTS_MODE',
 'LIVINGAREA_MODE',
 'NONLIVINGAPARTMENTS_MODE',
 'NONLIVINGAREA_MODE',
 'APARTMENTS_MEDI',
 'BASEMENTAREA_MEDI',
 'YEARS_BEGINEXPLUATATION_MEDI',
 'YEARS_BUILD_MEDI',
 'COMMONAREA_MEDI',
 'ELEVATORS_MEDI',
 'ENTRANCES_MEDI',
 'FLOORSMAX_MEDI',
 'FLOORSMIN_MEDI',
 'LANDAREA_MEDI',
 'LIVINGAPARTMENTS_MEDI',
 'LIVINGAREA_MEDI',
 'NONLIVINGAPARTMENTS_MEDI',
 'NONLIVINGAREA_MEDI',
 'TOTALAREA_MODE',
 'OBS_30_CNT_SOCIAL_CIRCLE',
 'DEF_30_CNT_SOCIAL_CIRCLE',
 'OBS_60_CNT_SOCIAL_CIRCLE',
 'DEF_60_CNT_SOCIAL_CIRCLE',
 'DAYS_LAST_PHONE_CHANGE',
 'FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_3',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
 'FLAG_DOCUMENT_14',
 'FLAG_DOCUMENT_15',
 'FLAG_DOCUMENT_16',
 'FLAG_DOCUMENT_17',
 'FLAG_DOCUMENT_18',
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21',
 'AMT_REQ_CREDIT_BUREAU_HOUR',
 'AMT_REQ_CREDIT_BUREAU_DAY',
 'AMT_REQ_CREDIT_BUREAU_WEEK',
 'AMT_REQ_CREDIT_BUREAU_MON',
 'AMT_REQ_CREDIT_BUREAU_QRT',
 'AMT_REQ_CREDIT_BUREAU_YEAR']
```

```
categorical_feature_columns = list(set(data.columns)-set(data._get_numeric_data().columns))
categorical_feature_columns
```

```
['OCCUPATION_TYPE',
 'WEEKDAY_APPR_PROCESS_START',
 'FLAG_OWN_REALTY',
 'FONDKAPREMONT_MODE',
 'NAME_CONTRACT_TYPE',
 'EMERGENCYSTATE_MODE',
 'WALLSMATERIAL_MODE',
 'NAME_TYPE_SUITE',
 'HOUSETYPE_MODE',
 'FLAG_OWN_CAR',
 'NAME_HOUSING_TYPE',
 'NAME_FAMILY_STATUS',
 'CODE_GENDER',
 'ORGANIZATION_TYPE',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE']
```

Check For Null Value

```
for name in numerical_feature_columns:
  if data[name].isna().sum()>0:
    print(f"{name}: {data[name].isna().sum()}")
```

```
CNT_FAM_MEMBERS: 2
EXT_SOURCE_1: 173378
EXT_SOURCE_2: 660
EXT_SOURCE_3: 60965
APARTMENTS_AVG: 156061
BASEMENTAREA_AVG: 179943
YEARS_BEGINEXPLUATATION_AVG: 150007
YEARS_BUILD_AVG: 204488
COMMONAREA_AVG: 214865
ELEVATORS_AVG: 163891
ENTRANCES_AVG: 154828
FLOORSMAX_AVG: 153020
FLOORSMIN_AVG: 208642
LANDAREA_AVG: 182590
LIVINGAPARTMENTS_AVG: 210199
LIVINGAREA_AVG: 154350
NONLIVINGAPARTMENTS_AVG: 213514
NONLIVINGAREA_AVG: 169682
APARTMENTS_MODE: 156061
BASEMENTAREA_MODE: 179943
YEARS_BEGINEXPLUATATION_MODE: 150007
YEARS_BUILD_MODE: 204488
COMMONAREA_MODE: 214865
ELEVATORS_MODE: 163891
ENTRANCES_MODE: 154828
FLOORSMAX_MODE: 153020
FLOORSMIN_MODE: 208642
LANDAREA_MODE: 182590
LIVINGAPARTMENTS_MODE: 210199
LIVINGAREA_MODE: 154350
NONLIVINGAPARTMENTS_MODE: 213514
NONLIVINGAREA_MODE: 169682
APARTMENTS_MEDI: 156061
BASEMENTAREA_MEDI: 179943
YEARS_BEGINEXPLUATATION_MEDI: 150007
YEARS_BUILD_MEDI: 204488
COMMONAREA_MEDI: 214865
ELEVATORS_MEDI: 163891
ENTRANCES_MEDI: 154828
FLOORSMAX_MEDI: 153020
FLOORSMIN_MEDI: 208642
LANDAREA_MEDI: 182590
LIVINGAPARTMENTS_MEDI: 210199
LIVINGAREA_MEDI: 154350
NONLIVINGAPARTMENTS_MEDI: 213514
NONLIVINGAREA_MEDI: 169682
TOTALAREA_MODE: 148431
OBS_30_CNT_SOCIAL_CIRCLE: 1021
DEF_30_CNT_SOCIAL_CIRCLE: 1021
OBS_60_CNT_SOCIAL_CIRCLE: 1021
DEF_60_CNT_SOCIAL_CIRCLE: 1021
DAYS_LAST_PHONE_CHANGE: 1
AMT_REQ_CREDIT_BUREAU_HOUR: 41519
AMT_REQ_CREDIT_BUREAU_DAY: 41519
AMT_REQ_CREDIT_BUREAU_WEEK: 41519
AMT_REQ_CREDIT_BUREAU_MON: 41519
AMT_REQ_CREDIT_BUREAU_QRT: 41519
AMT_REQ_CREDIT_BUREAU_YEAR: 41519
```

```
# Identify columns with more than 50% missing values
cols_to_drop = data.columns[data.isnull().mean() > 0.5]

# Drop the columns
df = data.drop(cols_to_drop, axis=1)
df
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	100002	1	Cash loans	M	N	Y
1	100003	0	Cash loans	F	N	N
2	100004	0	Revolving loans	M	Y	Y
3	100006	0	Cash loans	F	N	Y
4	100007	0	Cash loans	M	N	Y

```
categorical_feature_columns_df = list(set(df.columns)-set(df._get_numeric_data().columns))
categorical_feature_columns_df
```

```
['OCCUPATION_TYPE',
 'WEEKDAY_APPR_PROCESS_START',
 'FLAG_OWN_REALTY',
 'NAME_CONTRACT_TYPE',
 'EMERGENCYSTATE_MODE',
 'NAME_TYPE_SUITE',
 'FLAG_OWN_CAR',
 'NAME_HOUSING_TYPE',
 'NAME_FAMILY_STATUS',
 'CODE_GENDER',
 'ORGANIZATION_TYPE',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE']
```

```
numerical_feature_columns_df = list(df._get_numeric_data().columns)
numerical_feature_columns_df
```

```
 'DAYS_REGISTRATION',
 'DAYS_ID_PUBLISH',
 'FLAG_MOBIL',
 'FLAG_EMP_PHONE',
 'FLAG_WORK_PHONE',
 'FLAG_CONT_MOBILE',
 'FLAG_PHONE',
 'FLAG_EMAIL',
 'CNT_FAM_MEMBERS',
 'REGION_RATING_CLIENT',
 'REGION_RATING_CLIENT_W_CITY',
 'HOUR_APPR_PROCESS_START',
 'REG_REGION_NOT_LIVE_REGION',
 'REG_REGION_NOT_WORK_REGION',
 'LIVE_REGION_NOT_WORK_REGION',
 'REG_CITY_NOT_LIVE_CITY',
 'REG_CITY_NOT_WORK_CITY',
 'LIVE_CITY_NOT_WORK_CITY',
 'EXT_SOURCE_2',
 'EXT_SOURCE_3',
 'YEARS_BEGINEXPLUATATION_AVG',
 'FLOORSMAX_AVG',
 'YEARS_BEGINEXPLUATATION_MODE',
 'FLOORSMAX_MODE',
 'YEARS_BEGINEXPLUATATION_MEDI',
 'FLOORSMAX_MEDI',
 'TOTALAREA_MODE',
 'OBS_30_CNT_SOCIAL_CIRCLE',
 'DEF_30_CNT_SOCIAL_CIRCLE',
 'OBS_60_CNT_SOCIAL_CIRCLE',
 'DEF_60_CNT_SOCIAL_CIRCLE',
 'DAYS_LAST_PHONE_CHANGE',
 'FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_3',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
```

```
        'AMT_REQ_CREDIT_BUREAU_WEEK',
        'AMT_REQ_CREDIT_BUREAU_MON',
        'AMT_REQ_CREDIT_BUREAU_QRT',
        'AMT_REQ_CREDIT_BUREAU_YEAR']

for name in categorical_feature_columns_df:
  if df[name].isna().sum()>0:
    print(f"{name}: {df[name].isna().sum()}")
```

```
OCCUPATION_TYPE: 96391
EMERGENCYSTATE_MODE: 145755
NAME_TYPE_SUITE: 1292
```

```
for name in numerical_feature_columns_df:
  if df[name].isna().sum()>0:
    print(f"{name}: {df[name].isna().sum()}")
```

```
AMT_ANNUITY: 12
AMT_GOODS_PRICE: 278
CNT_FAM_MEMBERS: 2
EXT_SOURCE_2: 660
EXT_SOURCE_3: 60965
YEARS_BEGINEXPLUATATION_AVG: 150007
FLOORSMAX_AVG: 153020
YEARS_BEGINEXPLUATATION_MODE: 150007
FLOORSMAX_MODE: 153020
YEARS_BEGINEXPLUATATION_MEDI: 150007
FLOORSMAX_MEDI: 153020
TOTALAREA_MODE: 148431
OBS_30_CNT_SOCIAL_CIRCLE: 1021
DEF_30_CNT_SOCIAL_CIRCLE: 1021
OBS_60_CNT_SOCIAL_CIRCLE: 1021
DEF_60_CNT_SOCIAL_CIRCLE: 1021
DAYS_LAST_PHONE_CHANGE: 1
AMT_REQ_CREDIT_BUREAU_HOUR: 41519
AMT_REQ_CREDIT_BUREAU_DAY: 41519
AMT_REQ_CREDIT_BUREAU_WEEK: 41519
AMT_REQ_CREDIT_BUREAU_MON: 41519
AMT_REQ_CREDIT_BUREAU_QRT: 41519
AMT_REQ_CREDIT_BUREAU_YEAR: 41519
```

```
n_features=['AMT_ANNUITY','AMT_GOODS_PRICE','CNT_FAM_MEMBERS','EXT_SOURCE_2','EXT_SOURCE_3','YEARS_BEGINEXPLUATATION_AVG','FLOORSMAX_AVG'
'YEARS_BEGINEXPLUATATION_MEDI','FLOORSMAX_MEDI','TOTALAREA_MODE','OBS_30_CNT_SOCIAL_CIRCLE','DEF_30_CNT_SOCIAL_CIRCLE','OBS_60_CNT_SOCIAL
'AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK','AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUREAU_

# compute mean of the feature column(s)
mean = df[n_features].mean()
```

```
# fill missing values with mean
df[n_features] = df[n_features].fillna(mean)
```

```
df[n_features].isnull().sum()
```

```
AMT_ANNUITY                   0
AMT_GOODS_PRICE               0
CNT_FAM_MEMBERS               0
EXT_SOURCE_2                  0
EXT_SOURCE_3                  0
YEARS_BEGINEXPLUATATION_AVG   0
FLOORSMAX_AVG                 0
YEARS_BEGINEXPLUATATION_MODE  0
FLOORSMAX_MODE                0
YEARS_BEGINEXPLUATATION_MEDI  0
FLOORSMAX_MEDI                0
TOTALAREA_MODE                0
OBS_30_CNT_SOCIAL_CIRCLE      0
DEF_30_CNT_SOCIAL_CIRCLE      0
OBS_60_CNT_SOCIAL_CIRCLE      0
DEF_60_CNT_SOCIAL_CIRCLE      0
DAYS_LAST_PHONE_CHANGE        0
AMT_REQ_CREDIT_BUREAU_HOUR    0
AMT_REQ_CREDIT_BUREAU_DAY     0
AMT_REQ_CREDIT_BUREAU_WEEK    0
AMT_REQ_CREDIT_BUREAU_MON     0
AMT_REQ_CREDIT_BUREAU_QRT     0
AMT_REQ_CREDIT_BUREAU_YEAR    0
dtype: int64
```

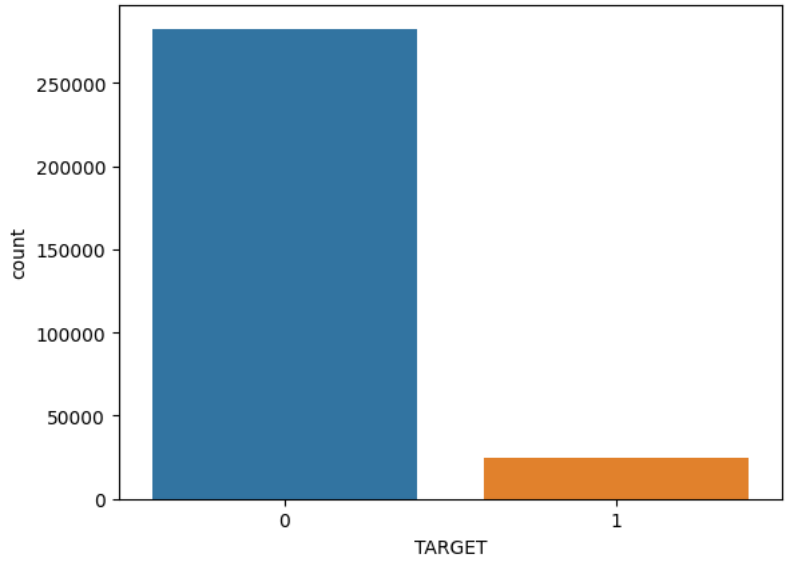
```
df['NAME_TYPE_SUITE'].fillna(df['NAME_TYPE_SUITE'].mode()[0],inplace=True)

df['EMERGENCYSTATE_MODE'].fillna(df['EMERGENCYSTATE_MODE'].mode()[0],inplace=True)

df['OCCUPATION_TYPE'].fillna(df['OCCUPATION_TYPE'].mode()[0],inplace=True)


sns.countplot(x=df["TARGET"])
```

```
<Axes: xlabel='TARGET', ylabel='count'>
```




```
#LabelEncoder

from sklearn.preprocessing import LabelEncoder

cols_to_encode = ['NAME_HOUSING_TYPE','OCCUPATION_TYPE','NAME_EDUCATION_TYPE','ORGANIZATION_TYPE','FLAG_OWN_REALTY','NAME_FAMILY_STATUS',
                  'NAME_INCOME_TYPE','CODE_GENDER','NAME_CONTRACT_TYPE','NAME_TYPE_SUITE','EMERGENCYSTATE_MODE']
# # Create an instance of the LabelEncoder
le = LabelEncoder()

# # Apply label encoding on the selected columns
df[cols_to_encode] = df[cols_to_encode].apply(le.fit_transform)
```

SMOTE to handle imbalance data

```
x=df.drop('TARGET', axis=1)
y=df['TARGET']


y.value_counts()
```

```
0    282686
1     24825
Name: TARGET, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
smk = SMOTE()
x_train_smote,y_train_smote=smk.fit_resample(x,y)


y_train_smote.value_counts()
```

```
1    282686
0    282686
Name: TARGET, dtype: int64
```

```
print(x_train_smote.shape)
print(y_train_smote.shape)
```

```
(565372, 80)
(565372,)
```

```
# Split the data set into training and testing
X_train, X_test, y_train, y_test = train_test_split(x_train_smote,y_train_smote, test_size=0.2, random_state=2,stratify=y_train_smote)
```

```
y_train.value_counts()
```

```
1    226149
0    226148
Name: TARGET, dtype: int64
```

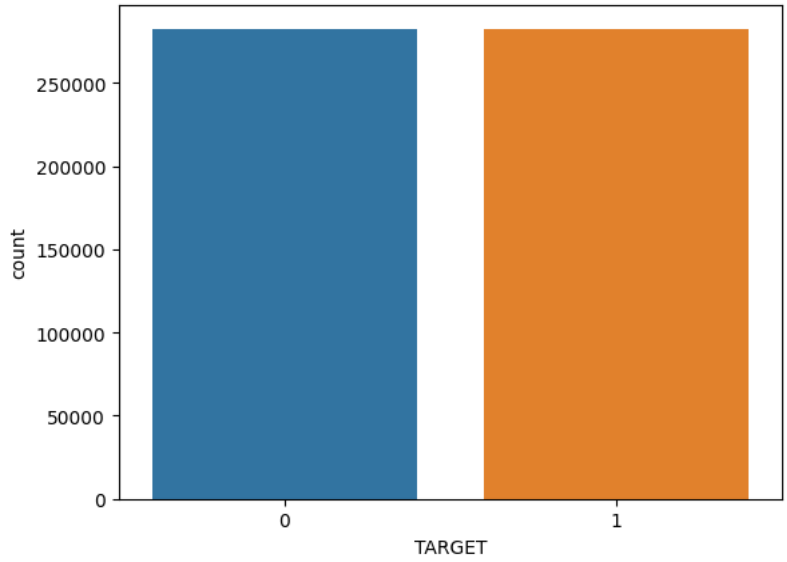
```
y_test.value_counts()
```

```
0    56538
1    56537
Name: TARGET, dtype: int64
```

```
# create a new dataframe
df_resampled = pd.DataFrame(x_train_smote, columns=df.drop('TARGET', axis=1).columns)
df_resampled['TARGET'] = y_train_smote

# plot a countplot of the resampled data
sns.countplot(x='TARGET', data=df_resampled)
```

```
<Axes: xlabel='TARGET', ylabel='count'>
```




```
scaler = StandardScaler()
x_train_std = scaler.fit_transform(X_train)
x_test_std=scaler.transform(X_test)
```

```
x_train_std.shape
```

```
(452297, 80)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization,Input
from tensorflow.keras.metrics import Precision, Recall
```

```
model=Sequential()
model.add(Input(shape=(80,)))
model.add(Dense(20,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(20,activation='relu'))
model.add(BatchNormalization())
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy',Precision(),Recall()])
```

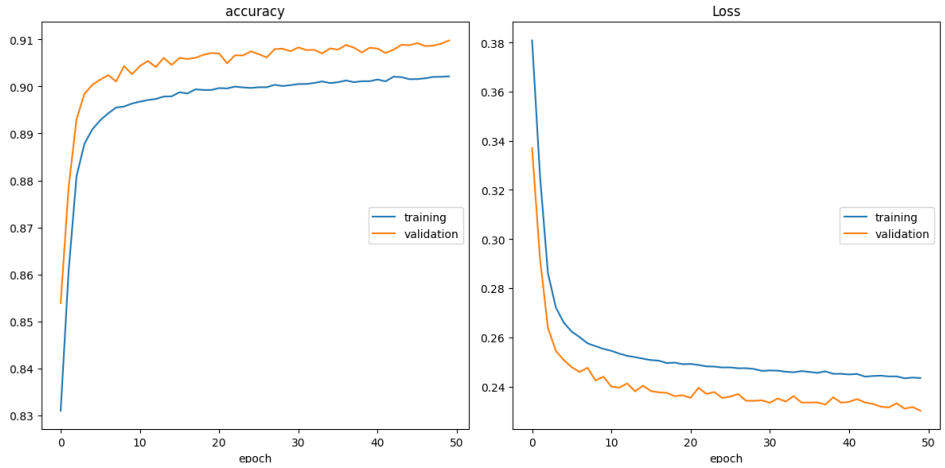
```
!pip install livelossplot
```

```
Collecting livelossplot
  Downloading livelossplot-0.5.5-py3-none-any.whl (22 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from livelossplot) (3.7.1)
Requirement already satisfied: bokeh in /usr/local/lib/python3.10/dist-packages (from livelossplot) (2.4.3)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (3.1.2)
```

```
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (1.22.4)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (23.1)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (8.4.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (6.0.1)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (6.3.1)
Requirement already satisfied: typing-extensions>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (4.7
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (4.41.0
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (2.8
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=2.9->bokeh->livelossplot) (
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->liveloss
Installing collected packages: livelossplot
Successfully installed livelossplot-0.5.5
```

```
from livelossplot import PlotLossesKerasTF
```

```
model.fit(x_train_std,y_train,epochs=50, batch_size=100,validation_data=(x_test_std,y_test),callbacks=[PlotLossesKerasTF()])
```




```
from sklearn.metrics import classification_report
```

```
t= model.predict(x_test_std)
```

```
3534/3534 [==============================] - 6s 2ms/step
```

```
print(type(y_test))
```

```
<class 'pandas.core.series.Series'>
```

```
print(len(t))
```

```
113075
```

```
t1 = []
for i in range(len(t)):
    if t[i] >= 0.5:
        t1.append(1)
    else:
        t1.append(0)
```

```
new_pred = pd.Series(t1)
print(new_pred)
```

```
0         1
1         0
2         0
3         0
4         0
         ..
113070    0
113071    0
113072    0
113073    1
113074    1
Length: 113075, dtype: int64
```

```
y_test
```

```
490491    1
201509    0
75201     0
77468     0
47528     0
         ..
129770    0
58364     0
58798     0
453274    1
422931    1
Name: TARGET, Length: 113075, dtype: int64
```

```
type(y_test)
```

```
pandas.core.series.Series
```

```
print(classification_report(y_test,new_pred))
```

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

```
                0       0.88      0.95      0.91     56538
                1       0.94      0.87      0.91     56537

         accuracy                           0.91    113075
        macro avg       0.91      0.91      0.91    113075
     weighted avg       0.91      0.91      0.91    113075
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,new_pred)
```

```
array([[53619,  2919],
       [ 7283, 49254]])
```

```
conf = confusion_matrix(y_test,new_pred)

tp,fp,fn,tn = confusion_matrix(y_test,new_pred).ravel()
specificity = tn / (tn+fp)
sensitivity= tp / (tp+fn)
print('TP,FP,FN,TN',tp,fp,fn,tn)
print('sensitivity =', sensitivity)
print('specificity =', specificity)
```

```
TP,FP,FN,TN 53619 2919 7283 49254
sensitivity = 0.8804144363075104
specificity = 0.9440515209016158
```

```
from sklearn.metrics import roc_auc_score
auc_score1 = roc_auc_score(y_test, new_pred)
print('roc_auc_score is',auc_score1)
```

```
roc_auc_score is 0.9097763555635503
```

✓ 0s completed at 5:56 PM