Lending Club Loan Project-02

March 26, 2022

1 Name: Sunil Pradhan

1.1 Project Name: Lending Club Loan Data Analysis

1.1.1 OBJECTIVE-

For companies like Lending Club correctly predicting whether or not a loan will be a default is very important. In this project, using the historical data from 2007 to 2015, you have to build a deep learning model to predict the chance of default for future loans.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
df=pd.read_csv("loan_data.csv")
df.head()
```

[2]:
```
   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
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 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   fico               9578 non-null   int64
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

[4]:
```
df.describe()
```

[4]:
```
       credit.policy     int.rate  installment  log.annual.inc          dti  \
count    9578.000000  9578.000000  9578.000000     9578.000000  9578.000000
mean        0.804970     0.122640   319.089413       10.932117    12.606679
std         0.396245     0.026847   207.071301        0.614813     6.883970
min         0.000000     0.060000    15.670000        7.547502     0.000000
25%         1.000000     0.103900   163.770000       10.558414     7.212500
50%         1.000000     0.122100   268.950000       10.928884    12.665000
75%         1.000000     0.140700   432.762500       11.291293    17.950000
max         1.000000     0.216400   940.140000       14.528354    29.960000

              fico  days.with.cr.line     revol.bal   revol.util  \
count  9578.000000        9578.000000  9.578000e+03  9578.000000
mean    710.846314        4560.767197  1.691396e+04    46.799236
std      37.970537        2496.930377  3.375619e+04    29.014417
min     612.000000         178.958333  0.000000e+00     0.000000
25%     682.000000        2820.000000  3.187000e+03    22.600000
50%     707.000000        4139.958333  8.596000e+03    46.300000
75%     737.000000        5730.000000  1.824950e+04    70.900000
```

```
max      827.000000       17639.958330  1.207359e+06   119.000000

       inq.last.6mths  delinq.2yrs      pub.rec  not.fully.paid
count     9578.000000  9578.000000  9578.000000     9578.000000
mean         1.577469     0.163708     0.062122        0.160054
std          2.200245     0.546215     0.262126        0.366676
min          0.000000     0.000000     0.000000        0.000000
25%          0.000000     0.000000     0.000000        0.000000
50%          1.000000     0.000000     0.000000        0.000000
75%          2.000000     0.000000     0.000000        0.000000
max         33.000000    13.000000     5.000000        1.000000
```

[5]:
```
df.isna().sum()
```

[5]:
```
credit.policy        0
purpose              0
int.rate             0
installment          0
log.annual.inc       0
dti                  0
fico                 0
days.with.cr.line    0
revol.bal            0
revol.util           0
inq.last.6mths       0
delinq.2yrs          0
pub.rec              0
not.fully.paid       0
dtype: int64
```

[6]:
```
df["purpose"].unique()
```

[6]:
```
array(['debt_consolidation', 'credit_card', 'all_other',
       'home_improvement', 'small_business', 'major_purchase',
       'educational'], dtype=object)
```

[7]:
```
from sklearn.preprocessing import LabelEncoder
```

[8]:
```
le=LabelEncoder()
df["purpose"]=le.fit_transform(df["purpose"])
df.head()
```

[8]:
```
   credit.policy  purpose  int.rate  installment  log.annual.inc    dti  fico  \
0              1        2    0.1189       829.10       11.350407  19.48   737
1              1        1    0.1071       228.22       11.082143  14.29   707
2              1        2    0.1357       366.86       10.373491  11.63   682
3              1        2    0.1008       162.34       11.350407   8.10   712
```

```
4              1         1    0.1426       102.92       11.299732  14.97   667

   days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
0        5639.958333      28854        52.1               0            0
1        2760.000000      33623        76.7               0            0
2        4710.000000       3511        25.6               1            0
3        2699.958333      33667        73.2               1            0
4        4066.000000       4740        39.5               0            1

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

[9]:
```
#Drop duplicated values
df=df.drop_duplicates()
df.head()
```

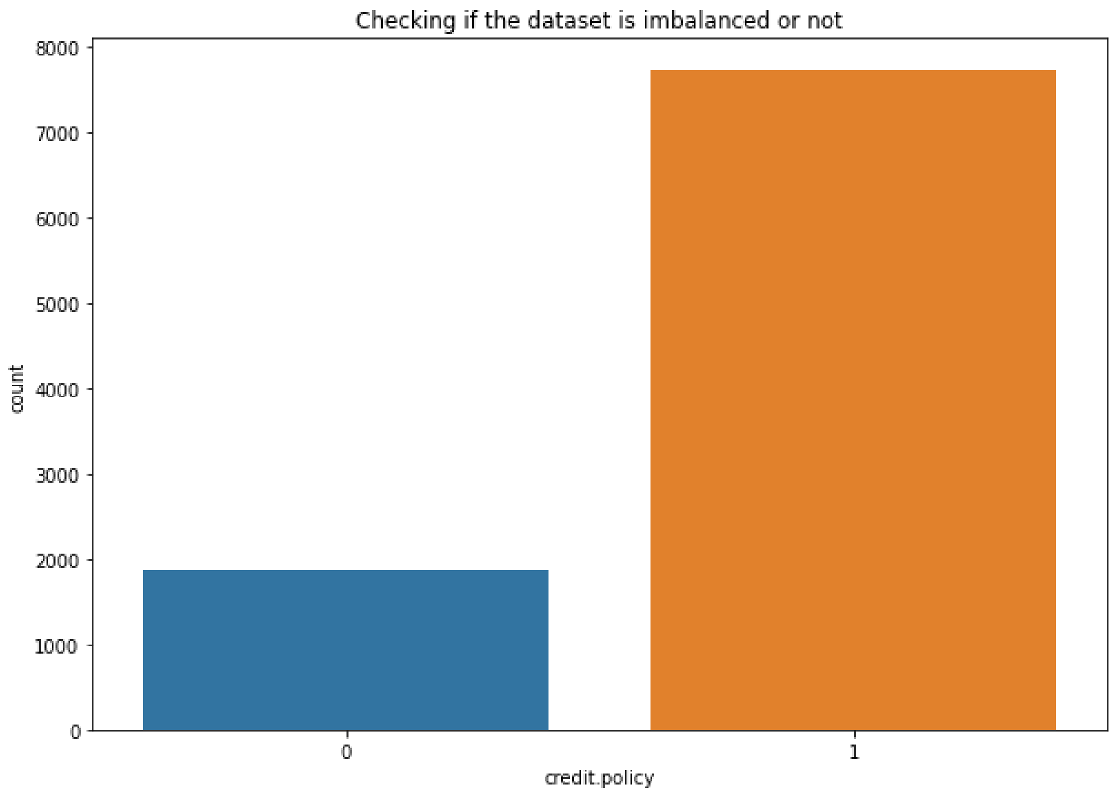
[9]:
```
   credit.policy  purpose  int.rate  installment  log.annual.inc    dti  fico  \
0              1        2    0.1189       829.10       11.350407  19.48   737
1              1        1    0.1071       228.22       11.082143  14.29   707
2              1        2    0.1357       366.86       10.373491  11.63   682
3              1        2    0.1008       162.34       11.350407   8.10   712
4              1        1    0.1426       102.92       11.299732  14.97   667

   days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
0        5639.958333      28854        52.1               0            0
1        2760.000000      33623        76.7               0            0
2        4710.000000       3511        25.6               1            0
3        2699.958333      33667        73.2               1            0
4        4066.000000       4740        39.5               0            1

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

[10]:
```
df["credit.policy"].value_counts()
```

[10]:
```
1    7710
0    1868
Name: credit.policy, dtype: int64
```

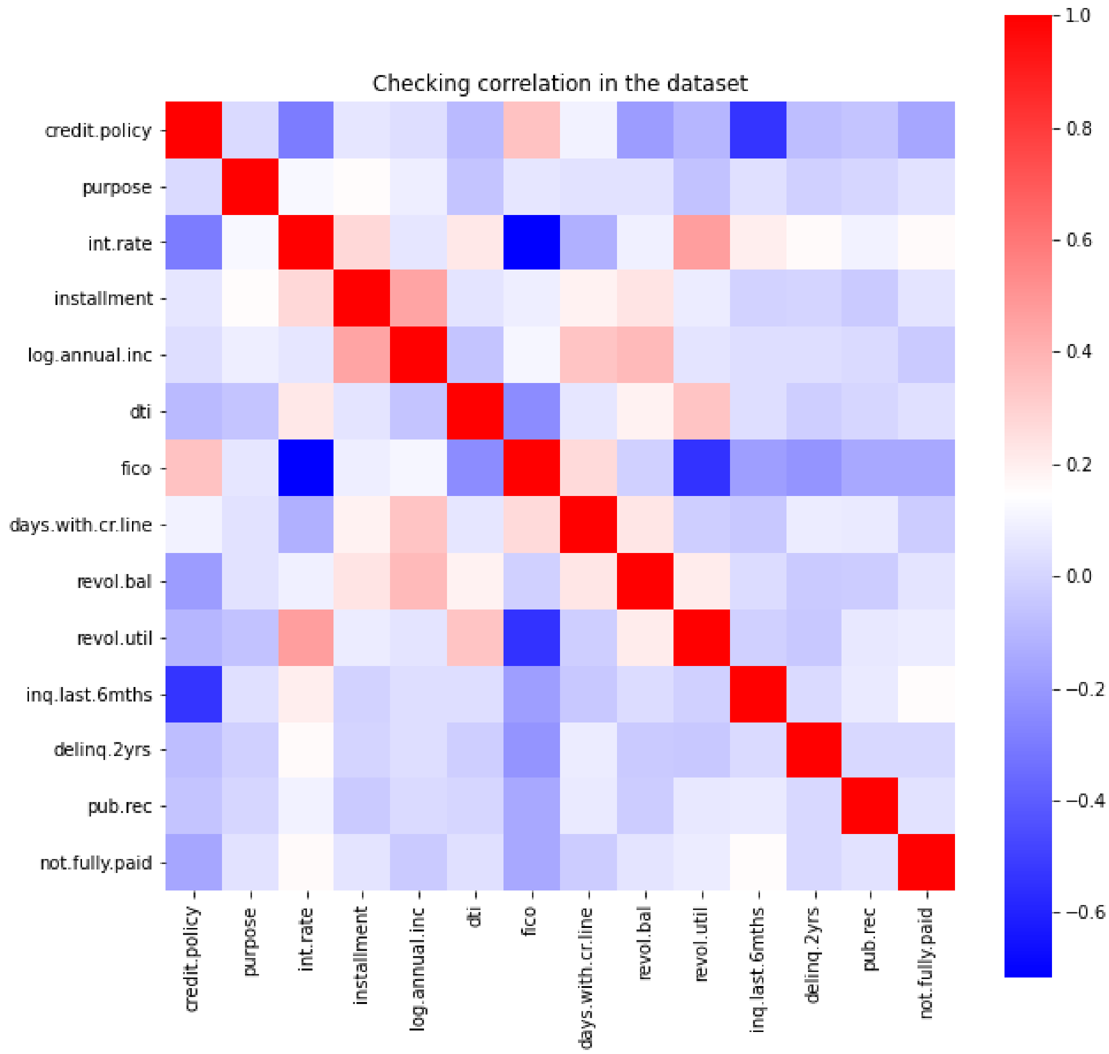
[11]:
```
plt.figure(figsize=(10,7))
sns.countplot(df["credit.policy"])
plt.title("Checking if the dataset is imbalanced or not")
```

[11]: Text(0.5, 1.0, 'Checking if the dataset is imbalanced or not')



[12]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data=df.corr(),cmap="bwr",square=True)
plt.title("Checking correlation in the dataset")
```

[12]: Text(0.5, 1.0, 'Checking correlation in the dataset')



Since,no features have strong correlation.so, all the features are highly relevent and consider for model.

[13]:
```
df.head()
```

[13]:
```
   credit.policy  purpose  int.rate  installment  log.annual.inc    dti  fico  \
0              1        2    0.1189       829.10       11.350407  19.48   737
1              1        1    0.1071       228.22       11.082143  14.29   707
2              1        2    0.1357       366.86       10.373491  11.63   682
3              1        2    0.1008       162.34       11.350407   8.10   712
4              1        1    0.1426       102.92       11.299732  14.97   667

   days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
0        5639.958333      28854        52.1               0            0
1        2760.000000      33623        76.7               0            0
```

```
2        4710.000000        3511         25.6               1             0
3        2699.958333       33667         73.2               1             0
4        4066.000000        4740         39.5               0             1

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

[14]:
```python
from sklearn.model_selection import train_test_split
```

[15]:
```python
x=df.drop("credit.policy",axis=1)
y=df[["credit.policy"]]
```

[16]:
```python
x.head()
```

[16]:
```
   purpose  int.rate  installment  log.annual.inc    dti  fico  \
0        2    0.1189       829.10       11.350407  19.48   737
1        1    0.1071       228.22       11.082143  14.29   707
2        2    0.1357       366.86       10.373491  11.63   682
3        2    0.1008       162.34       11.350407   8.10   712
4        1    0.1426       102.92       11.299732  14.97   667

   days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
0        5639.958333      28854        52.1               0            0
1        2760.000000      33623        76.7               0            0
2        4710.000000       3511        25.6               1            0
3        2699.958333      33667        73.2               1            0
4        4066.000000       4740        39.5               0            1

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

[17]:
```python
y.head()
```

[17]:
```
   credit.policy
0              1
1              1
2              1
3              1
4              1
```

[18]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪25,random_state=25)
```

[19]:
```
x_train.shape,x_test.shape
```

[19]: ((7183, 13), (2395, 13))

[20]:
```
y_train.shape,y_test.shape
```

[20]: ((7183, 1), (2395, 1))

[21]:
```
from sklearn.preprocessing import StandardScaler
```

[22]:
```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

1.1.2 Architect the model

[23]:
```
import tensorflow
```

[24]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

[25]:
```
x_train.shape[1],
```

[25]: (13,)

[26]:
```
model=Sequential()
model.add(Dense(64,activation="relu",input_shape=(x_train.shape[1],)))
model.add(Dense(32,activation="relu"))
model.add(Dense(16,activation="relu"))
model.add(Dense(1,activation="sigmoid"))
```

[27]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                896

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 16)                528

 dense_3 (Dense)             (None, 1)                 17
```

```
=================================================================
Total params: 3,521
Trainable params: 3,521
Non-trainable params: 0
_________________________________________________________________
```

[28]:
```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

[29]:
```
result=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100)
result
```

```
Epoch 1/100
225/225 [==============================] - 1s 3ms/step - loss: 0.3691 -
accuracy: 0.8412 - val_loss: 0.2719 - val_accuracy: 0.8935
Epoch 2/100
225/225 [==============================] - 0s 2ms/step - loss: 0.2366 -
accuracy: 0.9070 - val_loss: 0.2523 - val_accuracy: 0.9044
Epoch 3/100
225/225 [==============================] - 0s 2ms/step - loss: 0.2144 -
accuracy: 0.9123 - val_loss: 0.2342 - val_accuracy: 0.9094
Epoch 4/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1957 -
accuracy: 0.9238 - val_loss: 0.2264 - val_accuracy: 0.9094
Epoch 5/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1794 -
accuracy: 0.9304 - val_loss: 0.2131 - val_accuracy: 0.9194
Epoch 6/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1676 -
accuracy: 0.9362 - val_loss: 0.1975 - val_accuracy: 0.9269
Epoch 7/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1563 -
accuracy: 0.9424 - val_loss: 0.1897 - val_accuracy: 0.9299
Epoch 8/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1446 -
accuracy: 0.9460 - val_loss: 0.2002 - val_accuracy: 0.9203
Epoch 9/100
225/225 [==============================] - 1s 2ms/step - loss: 0.1361 -
accuracy: 0.9504 - val_loss: 0.1805 - val_accuracy: 0.9299
Epoch 10/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1275 -
accuracy: 0.9550 - val_loss: 0.1773 - val_accuracy: 0.9382
Epoch 11/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1216 -
accuracy: 0.9559 - val_loss: 0.1691 - val_accuracy: 0.9407
Epoch 12/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1113 -
accuracy: 0.9598 - val_loss: 0.1627 - val_accuracy: 0.9441
```

```
Epoch 13/100
225/225 [==============================] - 0s 2ms/step - loss: 0.1055 -
accuracy: 0.9627 - val_loss: 0.1814 - val_accuracy: 0.9403
Epoch 14/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0990 -
accuracy: 0.9667 - val_loss: 0.1623 - val_accuracy: 0.9491
Epoch 15/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0927 -
accuracy: 0.9669 - val_loss: 0.1551 - val_accuracy: 0.9474
Epoch 16/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0903 -
accuracy: 0.9681 - val_loss: 0.1508 - val_accuracy: 0.9503
Epoch 17/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0841 -
accuracy: 0.9733 - val_loss: 0.1547 - val_accuracy: 0.9491
Epoch 18/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0803 -
accuracy: 0.9756 - val_loss: 0.1428 - val_accuracy: 0.9532
Epoch 19/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0780 -
accuracy: 0.9745 - val_loss: 0.1464 - val_accuracy: 0.9495
Epoch 20/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0711 -
accuracy: 0.9780 - val_loss: 0.1528 - val_accuracy: 0.9457
Epoch 21/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0684 -
accuracy: 0.9798 - val_loss: 0.1429 - val_accuracy: 0.9528
Epoch 22/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0661 -
accuracy: 0.9780 - val_loss: 0.1618 - val_accuracy: 0.9503
Epoch 23/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0655 -
accuracy: 0.9777 - val_loss: 0.1430 - val_accuracy: 0.9532
Epoch 24/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0610 -
accuracy: 0.9811 - val_loss: 0.1539 - val_accuracy: 0.9466
Epoch 25/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0601 -
accuracy: 0.9805 - val_loss: 0.1769 - val_accuracy: 0.9453
Epoch 26/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0616 -
accuracy: 0.9801 - val_loss: 0.1552 - val_accuracy: 0.9441
Epoch 27/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0542 -
accuracy: 0.9830 - val_loss: 0.1449 - val_accuracy: 0.9549
Epoch 28/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0509 -
accuracy: 0.9847 - val_loss: 0.1500 - val_accuracy: 0.9557
```

```
Epoch 29/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0503 -
accuracy: 0.9861 - val_loss: 0.1550 - val_accuracy: 0.9553
Epoch 30/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0521 -
accuracy: 0.9826 - val_loss: 0.1497 - val_accuracy: 0.9557
Epoch 31/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0459 -
accuracy: 0.9869 - val_loss: 0.1553 - val_accuracy: 0.9532
Epoch 32/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0498 -
accuracy: 0.9850 - val_loss: 0.1742 - val_accuracy: 0.9461
Epoch 33/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0454 -
accuracy: 0.9862 - val_loss: 0.1591 - val_accuracy: 0.9566
Epoch 34/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0446 -
accuracy: 0.9877 - val_loss: 0.1597 - val_accuracy: 0.9562
Epoch 35/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0455 -
accuracy: 0.9858 - val_loss: 0.1555 - val_accuracy: 0.9562
Epoch 36/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0436 -
accuracy: 0.9865 - val_loss: 0.1588 - val_accuracy: 0.9549
Epoch 37/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0392 -
accuracy: 0.9883 - val_loss: 0.1526 - val_accuracy: 0.9570
Epoch 38/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0376 -
accuracy: 0.9884 - val_loss: 0.1635 - val_accuracy: 0.9553
Epoch 39/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0372 -
accuracy: 0.9901 - val_loss: 0.1558 - val_accuracy: 0.9545
Epoch 40/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0378 -
accuracy: 0.9891 - val_loss: 0.1547 - val_accuracy: 0.9587
Epoch 41/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0362 -
accuracy: 0.9890 - val_loss: 0.1867 - val_accuracy: 0.9511
Epoch 42/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0354 -
accuracy: 0.9891 - val_loss: 0.1765 - val_accuracy: 0.9578
Epoch 43/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0365 -
accuracy: 0.9882 - val_loss: 0.1700 - val_accuracy: 0.9537
Epoch 44/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0320 -
accuracy: 0.9904 - val_loss: 0.1676 - val_accuracy: 0.9591
```

```
Epoch 45/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0360 -
accuracy: 0.9883 - val_loss: 0.1687 - val_accuracy: 0.9549
Epoch 46/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0279 -
accuracy: 0.9916 - val_loss: 0.1866 - val_accuracy: 0.9516
Epoch 47/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0272 -
accuracy: 0.9929 - val_loss: 0.1891 - val_accuracy: 0.9516
Epoch 48/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0301 -
accuracy: 0.9910 - val_loss: 0.2027 - val_accuracy: 0.9453
Epoch 49/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0333 -
accuracy: 0.9905 - val_loss: 0.2032 - val_accuracy: 0.9461
Epoch 50/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0299 -
accuracy: 0.9911 - val_loss: 0.1847 - val_accuracy: 0.9507
Epoch 51/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0278 -
accuracy: 0.9929 - val_loss: 0.1878 - val_accuracy: 0.9516
Epoch 52/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0271 -
accuracy: 0.9926 - val_loss: 0.1987 - val_accuracy: 0.9503
Epoch 53/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0263 -
accuracy: 0.9916 - val_loss: 0.1898 - val_accuracy: 0.9553
Epoch 54/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0269 -
accuracy: 0.9914 - val_loss: 0.1877 - val_accuracy: 0.9582
Epoch 55/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0241 -
accuracy: 0.9929 - val_loss: 0.1908 - val_accuracy: 0.9541
Epoch 56/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0269 -
accuracy: 0.9921 - val_loss: 0.2042 - val_accuracy: 0.9574
Epoch 57/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0207 -
accuracy: 0.9951 - val_loss: 0.2082 - val_accuracy: 0.9578
Epoch 58/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0210 -
accuracy: 0.9936 - val_loss: 0.2302 - val_accuracy: 0.9474
Epoch 59/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0277 -
accuracy: 0.9914 - val_loss: 0.1946 - val_accuracy: 0.9478
Epoch 60/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0222 -
accuracy: 0.9932 - val_loss: 0.2126 - val_accuracy: 0.9557
```

```
Epoch 61/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0239 -
accuracy: 0.9925 - val_loss: 0.2076 - val_accuracy: 0.9524
Epoch 62/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0262 -
accuracy: 0.9908 - val_loss: 0.2178 - val_accuracy: 0.9562
Epoch 63/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0269 -
accuracy: 0.9916 - val_loss: 0.2072 - val_accuracy: 0.9541
Epoch 64/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0196 -
accuracy: 0.9947 - val_loss: 0.2126 - val_accuracy: 0.9570
Epoch 65/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0167 -
accuracy: 0.9965 - val_loss: 0.2114 - val_accuracy: 0.9566
Epoch 66/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0181 -
accuracy: 0.9957 - val_loss: 0.2153 - val_accuracy: 0.9537
Epoch 67/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0207 -
accuracy: 0.9926 - val_loss: 0.2775 - val_accuracy: 0.9344
Epoch 68/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0245 -
accuracy: 0.9925 - val_loss: 0.2152 - val_accuracy: 0.9524
Epoch 69/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0225 -
accuracy: 0.9933 - val_loss: 0.2114 - val_accuracy: 0.9570
Epoch 70/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0146 -
accuracy: 0.9967 - val_loss: 0.2193 - val_accuracy: 0.9562
Epoch 71/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0148 -
accuracy: 0.9958 - val_loss: 0.2131 - val_accuracy: 0.9578
Epoch 72/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0217 -
accuracy: 0.9925 - val_loss: 0.3095 - val_accuracy: 0.9399
Epoch 73/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0229 -
accuracy: 0.9918 - val_loss: 0.2115 - val_accuracy: 0.9574
Epoch 74/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0173 -
accuracy: 0.9947 - val_loss: 0.2110 - val_accuracy: 0.9537
Epoch 75/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0123 -
accuracy: 0.9969 - val_loss: 0.2153 - val_accuracy: 0.9582
Epoch 76/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0145 -
accuracy: 0.9962 - val_loss: 0.2247 - val_accuracy: 0.9553
```

```
Epoch 77/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0135 -
accuracy: 0.9962 - val_loss: 0.2317 - val_accuracy: 0.9570
Epoch 78/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0152 -
accuracy: 0.9947 - val_loss: 0.2641 - val_accuracy: 0.9428
Epoch 79/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0241 -
accuracy: 0.9926 - val_loss: 0.2312 - val_accuracy: 0.9553
Epoch 80/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0130 -
accuracy: 0.9969 - val_loss: 0.2198 - val_accuracy: 0.9541
Epoch 81/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0112 -
accuracy: 0.9982 - val_loss: 0.2250 - val_accuracy: 0.9516
Epoch 82/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0130 -
accuracy: 0.9961 - val_loss: 0.2325 - val_accuracy: 0.9553
Epoch 83/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0121 -
accuracy: 0.9969 - val_loss: 0.2560 - val_accuracy: 0.9507
Epoch 84/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0242 -
accuracy: 0.9910 - val_loss: 0.2624 - val_accuracy: 0.9478
Epoch 85/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0257 -
accuracy: 0.9916 - val_loss: 0.2355 - val_accuracy: 0.9520
Epoch 86/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0233 -
accuracy: 0.9925 - val_loss: 0.2279 - val_accuracy: 0.9578
Epoch 87/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0112 -
accuracy: 0.9974 - val_loss: 0.2536 - val_accuracy: 0.9537
Epoch 88/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0096 -
accuracy: 0.9978 - val_loss: 0.2537 - val_accuracy: 0.9491
Epoch 89/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0097 -
accuracy: 0.9981 - val_loss: 0.2580 - val_accuracy: 0.9541
Epoch 90/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0092 -
accuracy: 0.9982 - val_loss: 0.2463 - val_accuracy: 0.9574
Epoch 91/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0077 -
accuracy: 0.9990 - val_loss: 0.2509 - val_accuracy: 0.9507
Epoch 92/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0116 -
accuracy: 0.9971 - val_loss: 0.2368 - val_accuracy: 0.9537
```

```
Epoch 93/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0137 -
accuracy: 0.9964 - val_loss: 0.2471 - val_accuracy: 0.9553
Epoch 94/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0202 -
accuracy: 0.9929 - val_loss: 0.2783 - val_accuracy: 0.9507
Epoch 95/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0108 -
accuracy: 0.9971 - val_loss: 0.2532 - val_accuracy: 0.9562
Epoch 96/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0217 -
accuracy: 0.9922 - val_loss: 0.2553 - val_accuracy: 0.9520
Epoch 97/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0112 -
accuracy: 0.9967 - val_loss: 0.2934 - val_accuracy: 0.9545
Epoch 98/100
225/225 [==============================] - 1s 2ms/step - loss: 0.0102 -
accuracy: 0.9971 - val_loss: 0.2473 - val_accuracy: 0.9562
Epoch 99/100
225/225 [==============================] - 1s 3ms/step - loss: 0.0106 -
accuracy: 0.9971 - val_loss: 0.2966 - val_accuracy: 0.9478
Epoch 100/100
225/225 [==============================] - 0s 2ms/step - loss: 0.0172 -
accuracy: 0.9944 - val_loss: 0.2608 - val_accuracy: 0.9516
```

[29]:
```
<keras.callbacks.History at 0x1c204ad8910>
```

[30]:
```python
pred=(model.predict(x_test)>0.5)*1.0
pred
```

[30]:
```
array([[1.],
       [1.],
       [1.],
       …,
       [1.],
       [1.],
       [1.]])
```

[31]:
```python
from sklearn.metrics import confusion_matrix,classification_report
```

[32]:
```python
print(confusion_matrix(pred,y_test))
```

```
[[ 406   56]
 [  60 1873]]
```

[33]:
```python
print(classification_report(pred,y_test))
```

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

```
          0.0       0.87      0.88      0.88       462
          1.0       0.97      0.97      0.97      1933

     accuracy                           0.95      2395
    macro avg       0.92      0.92      0.92      2395
 weighted avg       0.95      0.95      0.95      2395
```

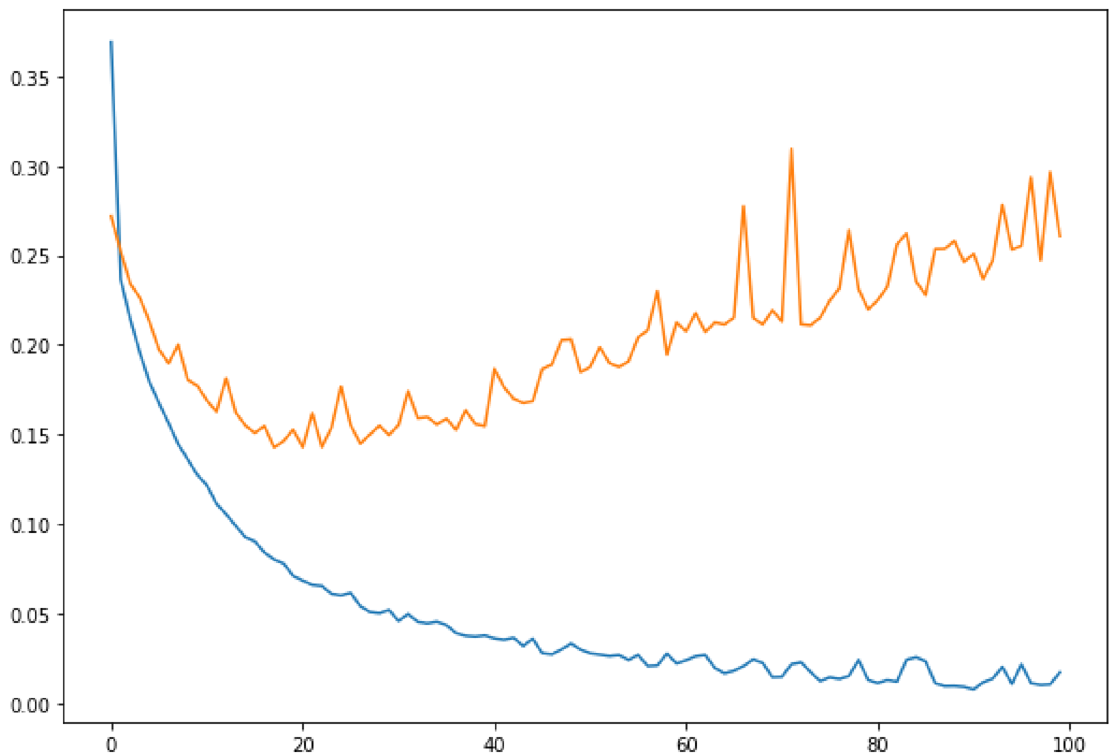
[34]:
```
df1=pd.DataFrame(result.history)
df1.head()
```

[34]:

	loss	accuracy	val_loss	val_accuracy
0	0.369062	0.841153	0.271902	0.893528
1	0.236558	0.907003	0.252330	0.904384
2	0.214423	0.912293	0.234218	0.909395
3	0.195716	0.923848	0.226374	0.909395
4	0.179404	0.930391	0.213123	0.919415

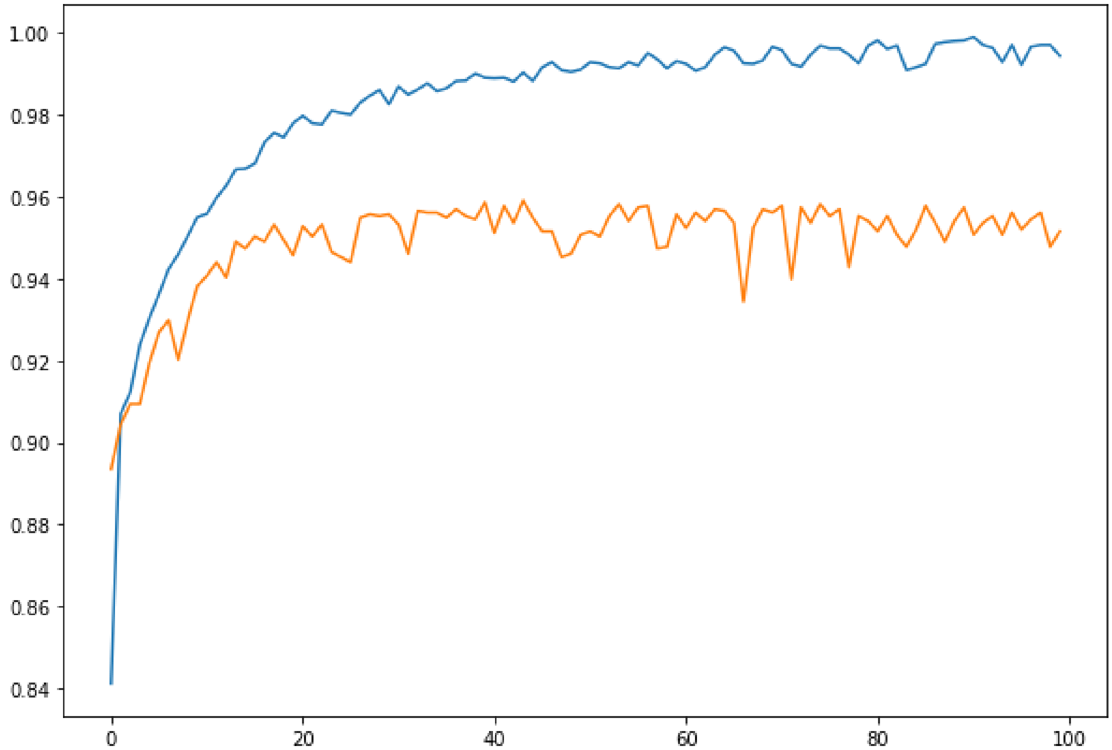
[35]:
```
plt.figure(figsize=(10,7))
plt.plot(df1["loss"])
plt.plot(df1["val_loss"])
```

[35]: [<matplotlib.lines.Line2D at 0x1c206fe6040>]



[36]:
```
plt.figure(figsize=(10,7))
plt.plot(df1["accuracy"])
plt.plot(df1["val_accuracy"])
```

[36]: [<matplotlib.lines.Line2D at 0x1c207034cd0>]



2 THANK YOU…!!!