1. Feature Transformation

- Transform categorical values into numerical values (discrete)

```
#reading the data set
data = pd.read_csv("C:/Users/david/Desktop/personal/AI/course-4_PC AIML - Deep Learning/project/1596018188_datasets (1)/loan_data
data.head()
```

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not.f
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	

2. Exploratory data analysis of different factors of the dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   credit.policy               9578 non-null   int64
 1   int.rate                    9578 non-null   float64
 2   installment                 9578 non-null   float64
 3   log.annual.inc              9578 non-null   float64
 4   dti                         9578 non-null   float64
 5   fico                        9578 non-null   int64
 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
 13  purpose_all_other           9578 non-null   uint8
 14  purpose_credit_card         9578 non-null   uint8
 15  purpose_debt_consolidation  9578 non-null   uint8
 16  purpose_educational         9578 non-null   uint8
 17  purpose_home_improvement    9578 non-null   uint8
 18  purpose_major_purchase      9578 non-null   uint8
 19  purpose_small_business      9578 non-null   uint8
dtypes: float64(6), int64(7), uint8(7)
memory usage: 1.0 MB
```

```
df.shape
```

```
(9578, 20)
```

]:
```
df.describe().T
```

]:

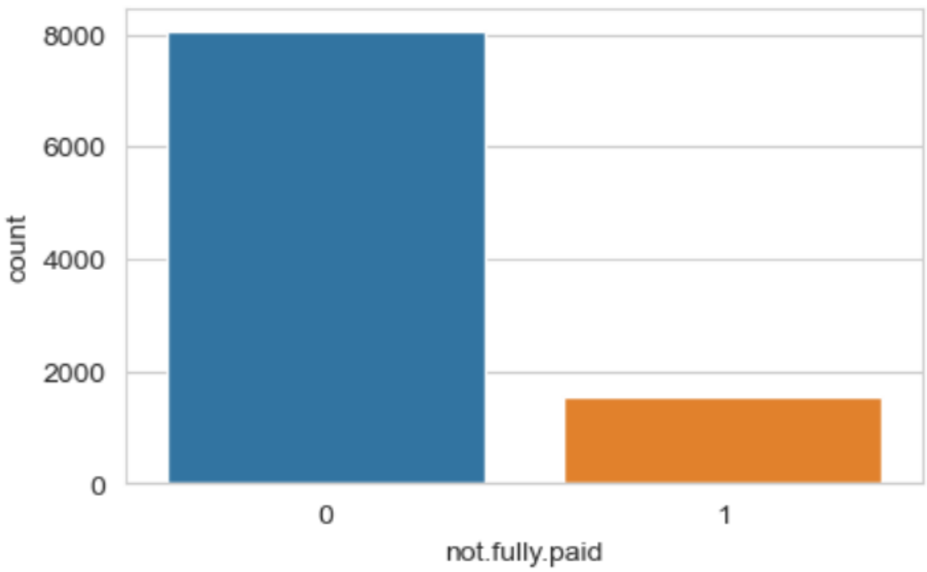
	count	mean	std	min	25%	50%	75%	max
credit.policy	9578.0	0.804970	0.396245	0.000000	1.000000	1.000000	1.000000	1.000000e+00
int.rate	9578.0	0.122640	0.026847	0.060000	0.103900	0.122100	0.140700	2.164000e-01
installment	9578.0	319.089413	207.071301	15.670000	163.770000	268.950000	432.762500	9.401400e+02
log.annual.inc	9578.0	10.932117	0.614813	7.547502	10.558414	10.928884	11.291293	1.452835e+01
dti	9578.0	12.606679	6.883970	0.000000	7.212500	12.665000	17.950000	2.996000e+01
fico	9578.0	710.846314	37.970537	612.000000	682.000000	707.000000	737.000000	8.270000e+02
days.with.cr.line	9578.0	4560.767197	2496.930377	178.958333	2820.000000	4139.958333	5730.000000	1.763996e+04
revol.bal	9578.0	16913.963876	33756.189557	0.000000	3187.000000	8596.000000	18249.500000	1.207359e+06
revol.util	9578.0	46.799236	29.014417	0.000000	22.600000	46.300000	70.900000	1.190000e+02
inq.last.6mths	9578.0	1.577469	2.200245	0.000000	0.000000	1.000000	2.000000	3.300000e+01
delinq.2yrs	9578.0	0.163708	0.546215	0.000000	0.000000	0.000000	0.000000	1.300000e+01
pub.rec	9578.0	0.062122	0.262126	0.000000	0.000000	0.000000	0.000000	5.000000e+00
not.fully.paid	9578.0	0.160054	0.366676	0.000000	0.000000	0.000000	0.000000	1.000000e+00
purpose_all_other	9578.0	0.243370	0.429139	0.000000	0.000000	0.000000	0.000000	1.000000e+00
purpose_credit_card	9578.0	0.131760	0.338248	0.000000	0.000000	0.000000	0.000000	1.000000e+00
purpose_debt_consolidation	9578.0	0.413134	0.492422	0.000000	0.000000	0.000000	1.000000	1.000000e+00

```
df['not.fully.paid'].value_counts()
```

```
0     8045
1     1533
Name: not.fully.paid, dtype: int64
```

:
```
plt.figure(figsize=(5,3))
sns.countplot(x='not.fully.paid',data=df)
```

: <Axes: xlabel='not.fully.paid', ylabel='count'>

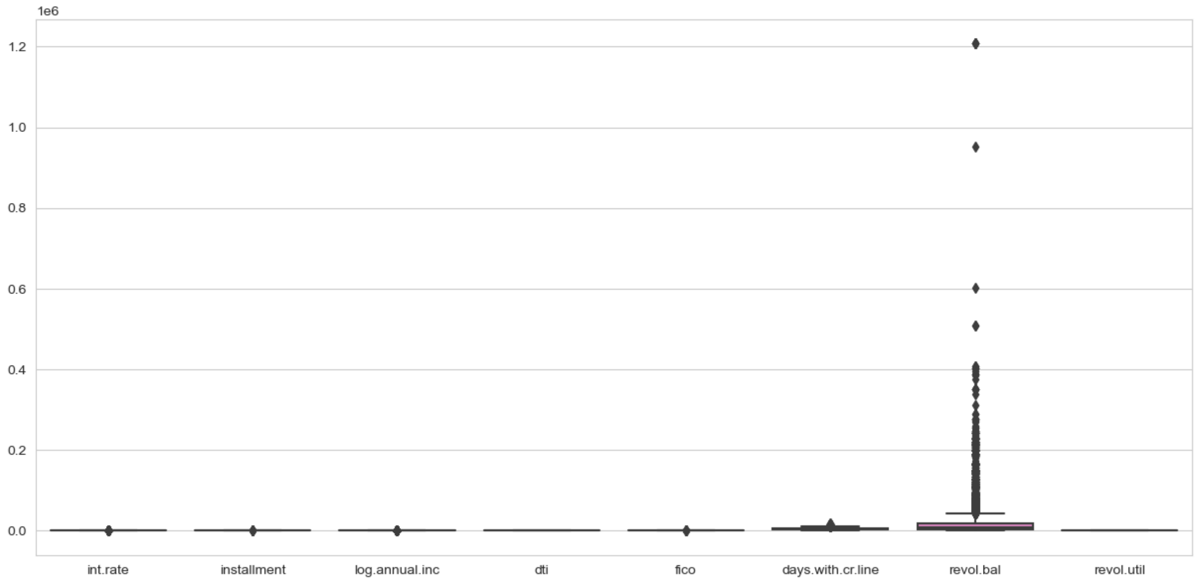


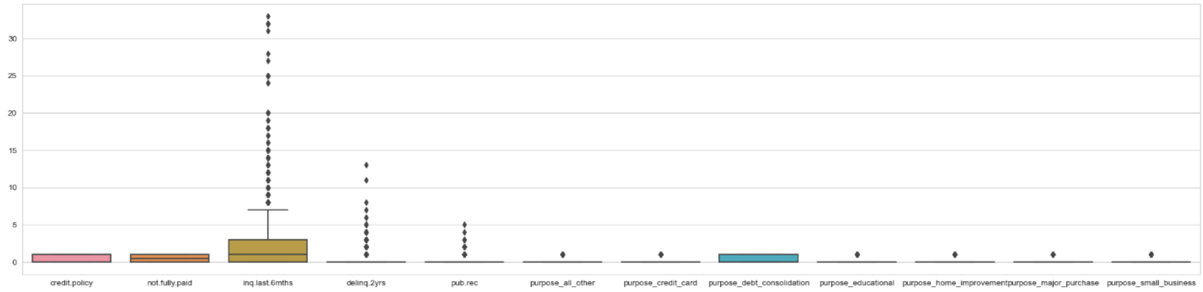
```python
#handling imbalanced dataset - using resample

not_fully_paid_0 = df[df['not.fully.paid'] == 0]
not_fully_paid_1 = df[df['not.fully.paid'] == 1]

print('not_fully_paid_0', not_fully_paid_0.shape)
print('not_fully_paid_1', not_fully_paid_1.shape)
```

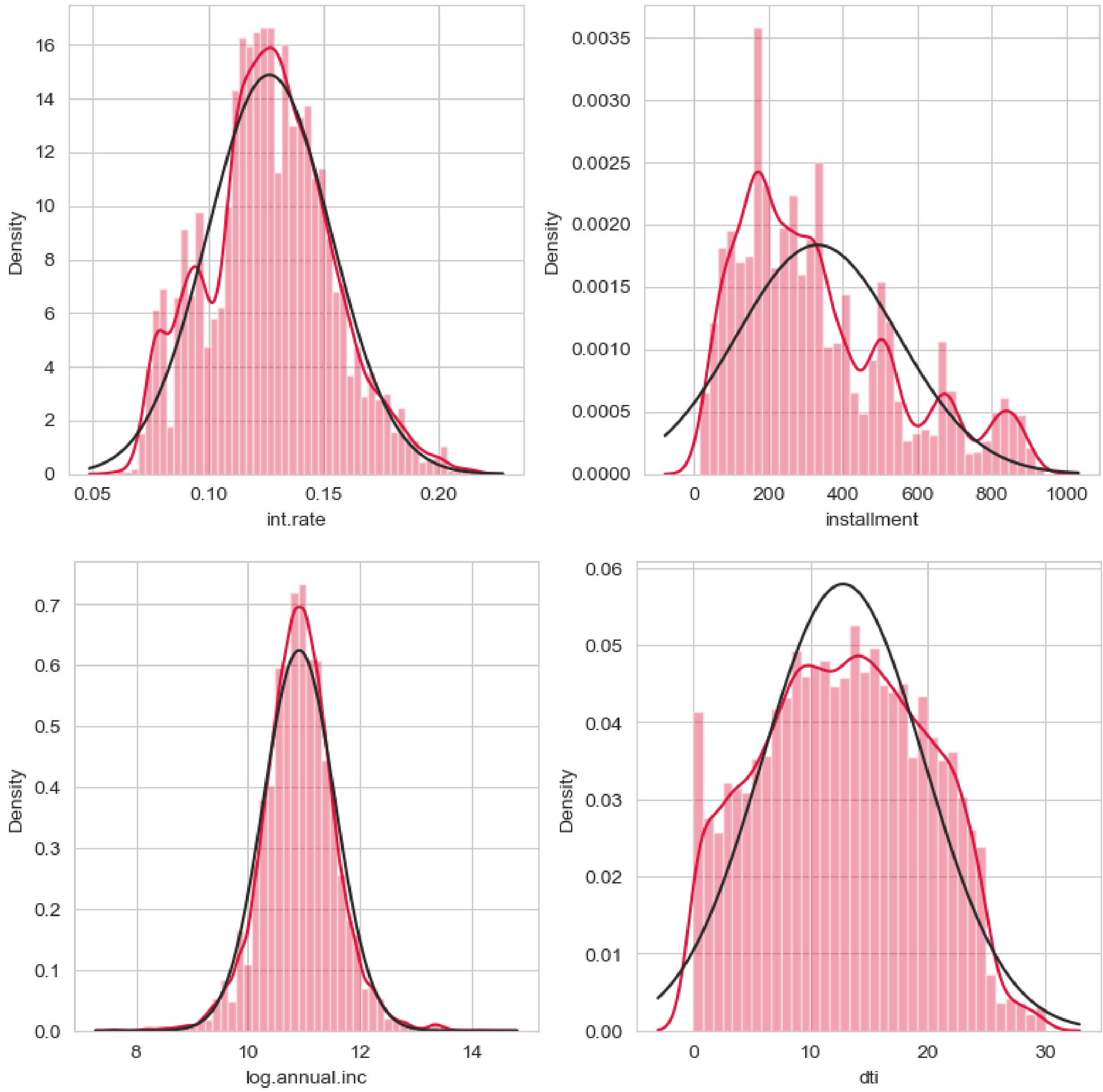
```
not_fully_paid_0 (8045, 20)
not_fully_paid_1 (1533, 20)
```

```python
#imbalanced data handled
df['not.fully.paid'].value_counts()
```

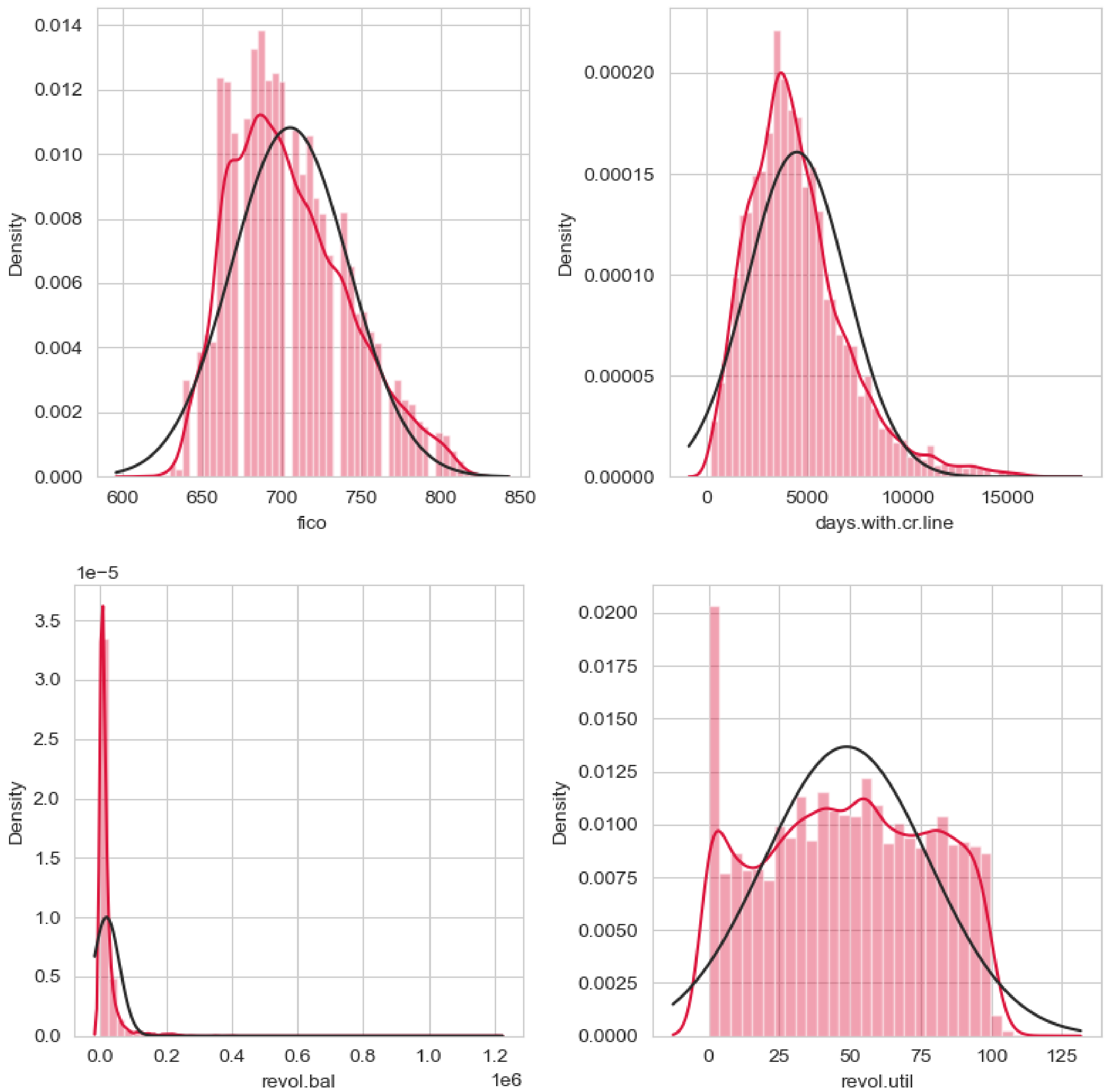
```
0    8045
1    8045
Name: not.fully.paid, dtype: int64
```



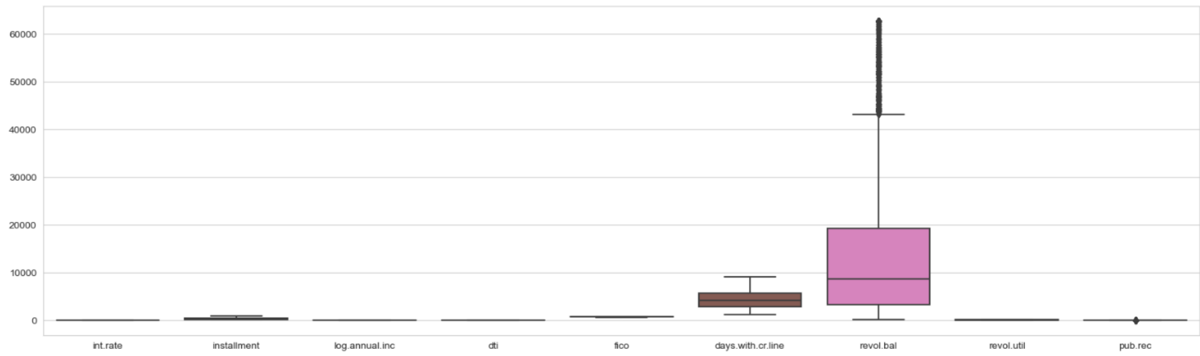


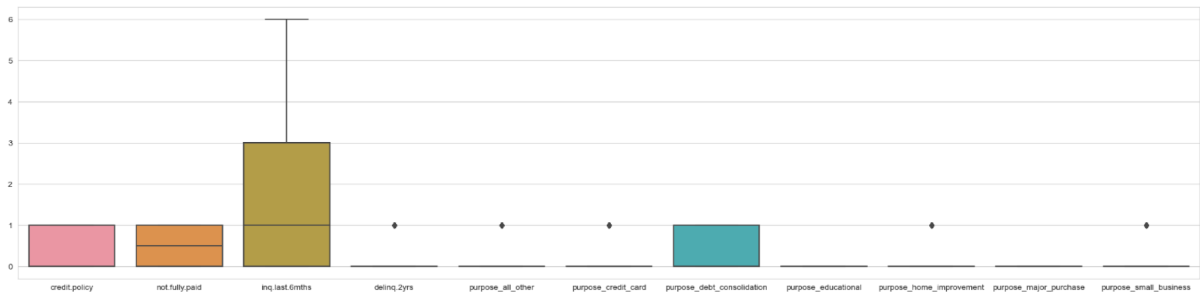


Density
int.rate
0.05
0.10
0.15
0.20
0
2
4
6
8
10
12
14
16
installment
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
0.0035
0
200
400
600
800
1000
log.annual.inc
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
8
10
12
14
dti
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30


Density
fico
days.with.cr.line
1e−5
revol.bal
1e6
revol.util

```
Outlier caps for credit.policy
  --95p: 1.0 / 0 values exceed that
  --3sd: 2.1 / 0 values exceed that
  --99p: 1.0 / 0 values exceed that

Outlier caps for int.rate
  --95p: 0.2 / 805 values exceed that
  --3sd: 0.2 / 36 values exceed that
  --99p: 0.2 / 158 values exceed that

Outlier caps for installment
  --95p: 807.6 / 801 values exceed that
  --3sd: 983.4 / 0 values exceed that
  --99p: 878.9 / 157 values exceed that

Outlier caps for log.annual.inc
  --95p: 11.9 / 800 values exceed that
  --3sd: 12.8 / 168 values exceed that
  --99p: 12.6 / 156 values exceed that

Outlier caps for dti
  --95p: 23.8 / 800 values exceed that
  --3sd: 33.5 / 0 values exceed that
  --99p: 26.8 / 160 values exceed that

Outlier caps for fico
  --95p: 777.0 / 672 values exceed that
```



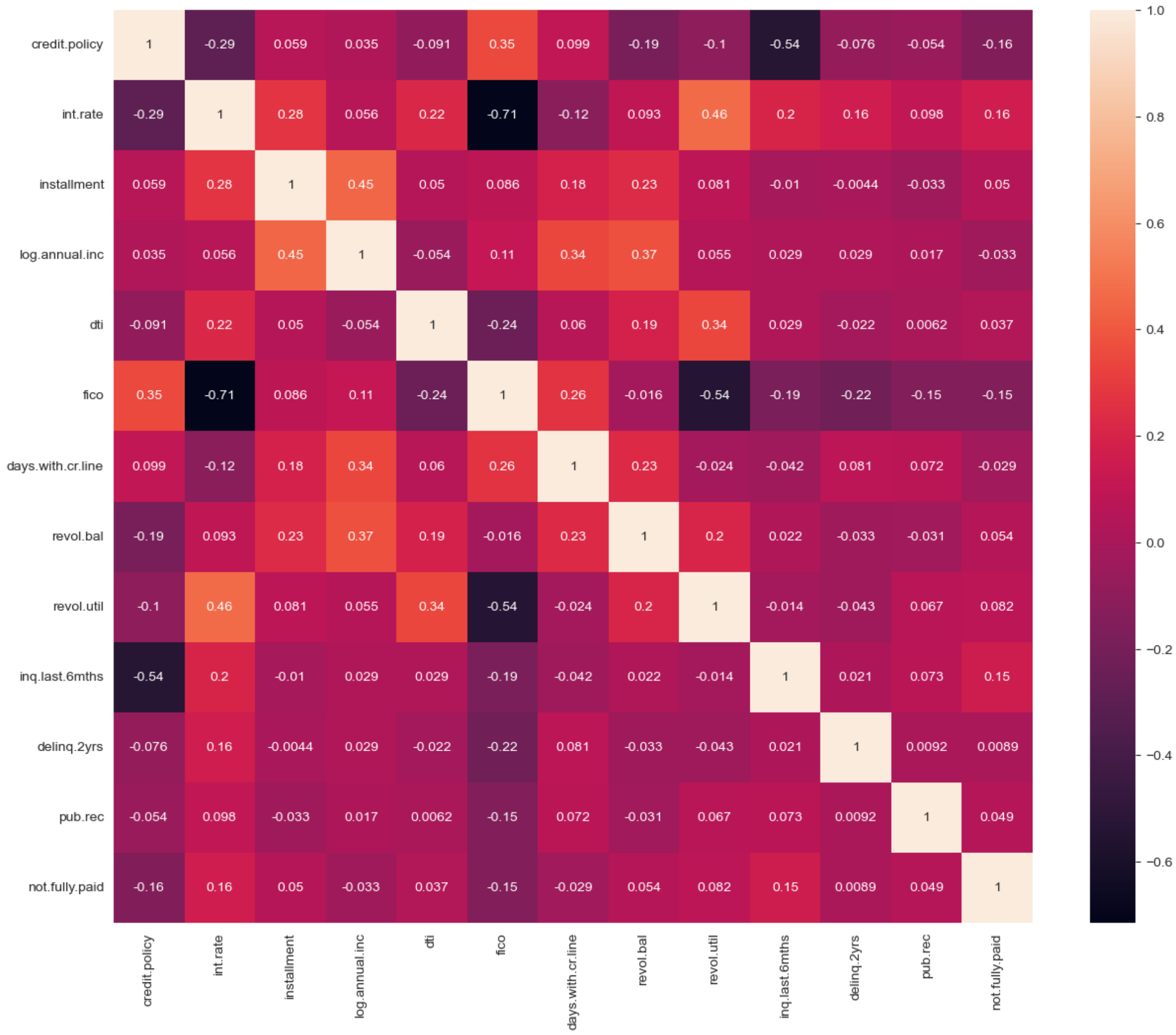


3. Additional Feature Engineering

- You will check the correlation between features and will drop those features which have a strong correlation

```
df.corr()
```

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delin
credit.policy	1.000000	-0.282765	0.053693	0.019055	-0.072089	0.367873	0.086316	-0.096409	-0.084485	-0.584724	-0.0
int.rate	-0.282765	1.000000	0.259678	0.077954	0.197713	-0.702432	-0.111021	0.105973	0.427965	0.197232	0.1
installment	0.053693	0.259678	1.000000	0.477643	0.027296	0.112259	0.196378	0.327932	0.057627	-0.003825	0.0
log.annual.inc	0.019055	0.077954	0.477643	1.000000	-0.030829	0.105569	0.373146	0.496275	0.083805	0.031435	0.0
dti	-0.072089	0.197713	0.027296	-0.030829	1.000000	-0.208458	0.101486	0.294946	0.318520	0.017553	-0.0
fico	0.367873	-0.702432	0.112259	0.105569	-0.208458	1.000000	0.249862	-0.016108	-0.492924	-0.187073	-0.2
days.with.cr.line	0.086316	-0.111021	0.196378	0.373146	0.101486	0.249862	1.000000	0.342745	0.023656	-0.013847	0.0
revol.bal	-0.096409	0.105973	0.327932	0.496275	0.294946	-0.016108	0.342745	1.000000	0.346329	-0.005507	-0.0
revol.util	-0.084485	0.427965	0.057627	0.083805	0.318520	-0.492924	0.023656	0.346329	1.000000	-0.040142	-0.0
inq.last.6mths	-0.584724	0.197232	-0.003825	0.031435	0.017553	-0.187073	-0.013847	-0.005507	-0.040142	1.000000	-0.0
delinq.2yrs	-0.047963	0.164196	0.003193	0.015278	-0.021818	-0.229699	0.081987	-0.058498	-0.030781	-0.000095	1.0
pub.rec	-0.064586	0.102742	-0.028703	0.010706	0.030215	-0.162760	0.083195	-0.049377	0.080320	0.102095	-0.0
not.fully.paid	-0.193486	0.217160	0.078110	-0.044045	0.041874	-0.209217	-0.034872	0.054858	0.103689	0.190637	0.0
purpose_all_other	-0.025119	-0.120703	-0.205774	-0.084775	-0.124782	0.054284	-0.077735	-0.119198	-0.120135	0.009150	0.0
purpose_credit_card	0.012249	-0.043799	-0.000311	0.076602	0.078221	-0.007528	0.047579	0.115706	0.089308	-0.044536	-0.0
purpose_debt_consolidation	0.024277	0.089997	0.116552	-0.047857	0.181831	-0.140818	0.001324	0.047707	0.204263	-0.058960	-0.0



4. Modeling

- After applying EDA and feature engineering, you are now ready to build the predictive models
- In this part, you will create a deep learning model using Keras with Tensorflow backend

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 200)               4000

 dropout (Dropout)           (None, 200)               0

 dense_1 (Dense)             (None, 200)               40200

 dropout_1 (Dropout)         (None, 200)               0

 dense_2 (Dense)             (None, 200)               40200

 dropout_2 (Dropout)         (None, 200)               0

 dense_3 (Dense)             (None, 1)                 201

=================================================================
Total params: 84601 (330.47 KB)
Trainable params: 84601 (330.47 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
history = model.fit(X_train_scale,y_train,
                    validation_data=(X_val_scale, y_val),epochs=100,verbose=1)
```

```
0.8923
Epoch 95/100
385/385 [==============================] - 2s 4ms/step - loss: 0.2043 - accuracy: 0.9170 - val_loss: 0.3175 - val_accuracy:
0.8790
Epoch 96/100
385/385 [==============================] - 2s 4ms/step - loss: 0.2034 - accuracy: 0.9180 - val_loss: 0.3220 - val_accuracy:
0.8840
Epoch 97/100
385/385 [==============================] - 2s 4ms/step - loss: 0.1912 - accuracy: 0.9231 - val_loss: 0.3263 - val_accuracy:
0.8873
Epoch 98/100
385/385 [==============================] - 2s 4ms/step - loss: 0.1941 - accuracy: 0.9211 - val_loss: 0.3277 - val_accuracy:
0.8808
Epoch 99/100
385/385 [==============================] - 2s 4ms/step - loss: 0.2003 - accuracy: 0.9207 - val_loss: 0.3371 - val_accuracy:
0.8813
Epoch 100/100
385/385 [==============================] - 2s 4ms/step - loss: 0.1973 - accuracy: 0.9183 - val_loss: 0.3008 - val_accuracy:
0.8942
```

```
X_train_pred= model.predict(X_train_scale)
X_test_pred=model.predict(X_test_scale)
```

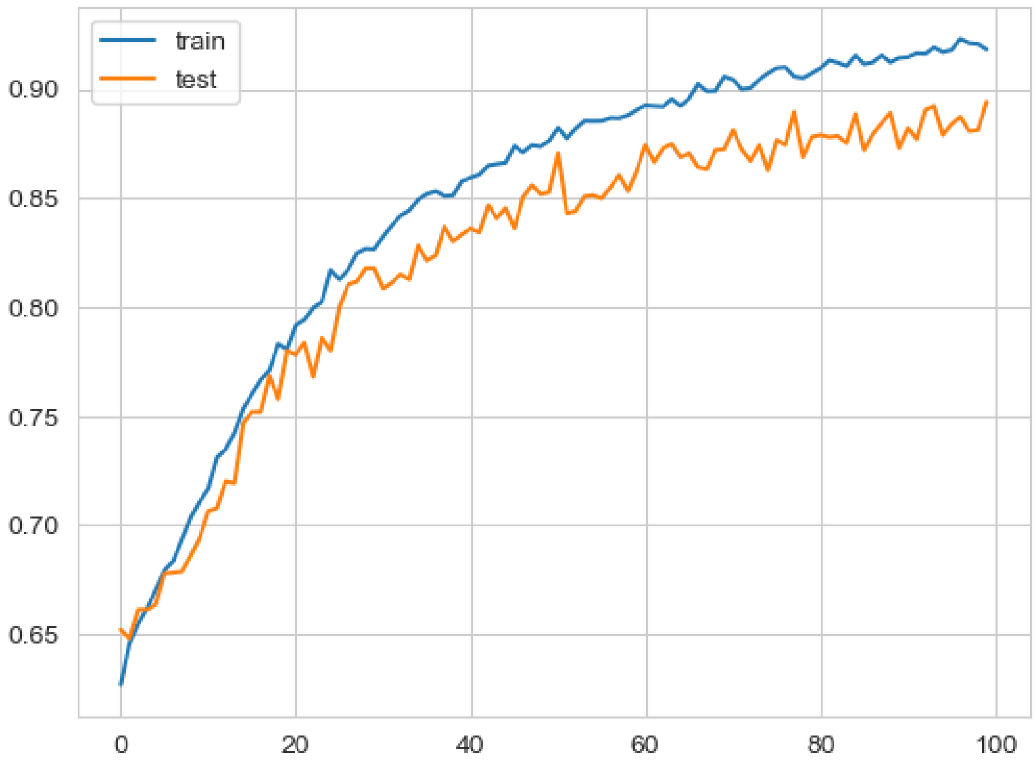
```
385/385 [==============================] - 1s 2ms/step
51/51 [==============================] - 0s 2ms/step
```

```
cm = confusion_matrix(y_pred=X_train_pred > 0.5,y_true=y_train)
cm
```

```
array([[6049,  105],
       [  41, 6113]], dtype=int64)
```

```
cm = confusion_matrix(y_pred=X_test_pred > 0.5,y_true=y_test)
cm
```

```
array([[658, 146],
       [ 26, 779]], dtype=int64)
```



```
#training score
model.evaluate(X_train_scale,y_train)
```

```
385/385 [==============================] - 1s 2ms/step - loss: 0.0628 - accuracy: 0.9881

[0.0628020241856575, 0.9881377816200256]
```

```
#validatn score
model.evaluate(X_val_scale,y_val)
```

```
68/68 [==============================] - 0s 2ms/step - loss: 0.3008 - accuracy: 0.8942

[0.3008177578449249, 0.8941555619239807]
```

```
#testing score
model.evaluate(X_test_scale,y_test)
```

```
51/51 [==============================] - 0s 2ms/step - loss: 0.3353 - accuracy: 0.8931

[0.33529841899871826, 0.8931013345718384]
```

```
predictions =(model.predict(X_test_scale)>0.5)
predictions2 =(model.predict(X_val_scale)>0.5)
```

```
51/51 [==============================] - 0s 2ms/step
68/68 [==============================] - 0s 2ms/step
```

```
accuracy_score(y_test, predictions)
```

```
0.8931013051584835
```

```
accuracy_score(y_val,predictions2)
```

```
0.8941555453290382
```

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.96      0.82      0.88       804
           1       0.84      0.97      0.90       805

    accuracy                           0.89      1609
   macro avg       0.90      0.89      0.89      1609
weighted avg       0.90      0.89      0.89      1609
```

```
print(classification_report(y_val, predictions2))
```

```
              precision    recall  f1-score   support

           0       0.95      0.83      0.89      1087
           1       0.85      0.96      0.90      1086

    accuracy                           0.89      2173
   macro avg       0.90      0.89      0.89      2173
weighted avg       0.90      0.89      0.89      2173
```