In [10]:
```
#Import the Operating System and Set the Working Directory
# Import the operating system. The OS module in Python provides functions for cre
# fetching its contents, changing and identifying the current directory, etc.
import os
os.getcwd()
```

Out[10]: 'C:\\Users\\Byju'

```
12/10/2021: Project 1 - Lending Club Loan Data Analysis
DESCRIPTION

Create a model that predicts whether or not a loan will be default using the
historical data.

Problem Statement:

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. As you will see later this dataset is highly
imbalanced and includes a lot of features that makes this problem more
challenging.

Domain: Finance

Analysis to be done: Perform data preprocessing and build a deep learning
prediction model.

Content:

Dataset columns and definition:

credit.policy: 1 if the customer meets the credit underwriting criteria of
LendingClub.com, and 0 otherwise.

purpose: The purpose of the loan (takes values "credit_card",
"debt_consolidation", "educational", "major_purchase", "small_business", and
"all_other").

int.rate: The interest rate of the loan, as a proportion (a rate of 11% would
be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are
assigned higher interest rates.

installment: The monthly installments owed by the borrower if the loan is
funded.

log.annual.inc: The natural log of the self-reported annual income of the
borrower.

dti: The debt-to-income ratio of the borrower (amount of debt divided by annual
income).

fico: The FICO credit score of the borrower.

days.with.cr.line: The number of days the borrower has had a credit line.
```

```
revol.bal: The borrower's revolving balance (amount unpaid at the end of the
credit card billing cycle).

revol.util: The borrower's revolving line utilization rate (the amount of the
credit line used relative to total credit available).

inq.last.6mths: The borrower's number of inquiries by creditors in the last 6
months.

delinq.2yrs: The number of times the borrower had been 30+ days past due on a
payment in the past 2 years.

pub.rec: The borrower's number of derogatory public records (bankruptcy
filings, tax liens, or judgments).

Steps to perform:

Perform exploratory data analysis and feature engineering and then apply
feature engineering. Follow up with a deep learning model to predict whether or
not the loan will be default using the historical data.

Tasks:

Feature Transformation
Transform categorical values into numerical values (discrete)

Exploratory data analysis of different factors of the dataset.

Additional Feature Engineering

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

This will help reduce the number of features and will leave you with the most
relevant features

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
```

In [14]:
```
#display the current working directory for confirmation
os.getcwd()
```

Out[14]:
```
'C:\\Users\\Byju'
```

In [15]:
```
#Lets create a course specific subdirectory under the student directory "Byju" to
if not os.path.exists('C:/Users/Byju/Dropbox/1. BNGPersonal/Purdue-SimpliLearn/3.
    #os.makedirs('/home/labsuser/Byju/Course4-PG-AI-DLTK')
    os.mkdir("C:/Users/Byju/Dropbox/1. BNGPersonal/Purdue-SimpliLearn/3. AI & ML/

#Lets set directory "Byju" as working diectory
#os.chdir("..")

#Let's explicitly reset the directory "Course3-PG-AI-ML" as the desired working d
os.chdir('C:/Users/Byju/Dropbox/1. BNGPersonal/Purdue-SimpliLearn/3. AI & ML/Asse

#display the current working directory for confirmation
os.getcwd()
```

Out[15]:
```
'C:\\Users\\Byju\\Dropbox\\1. BNGPersonal\\Purdue-SimpliLearn\\3. AI & ML\\Asse
smentProjects\\Module3\\1596018188_datasets'
```

In [17]:
```
!pip install tensorflow
!pip install keras
#!pip install tensorflow --ignore-installed
```

```
^C
Requirement already satisfied: keras in c:\users\byju\anaconda3\lib\site-packag
es (2.7.0)

WARNING: Error parsing requirements for -mportlib-metadata: [Errno 2] No such f
ile or directory: 'c:\\users\\byju\\anaconda3\\lib\\site-packages\\~mportlib_me
tadata-3.10.0.dist-info\\METADATA'
```

In [18]:
```
# import libraries

# Import NumPy (Numerical Python) open source Python library to access NumPy and
import numpy as np
# Import Pandas (Python-based data analysis toolkit ) open source Python library

import pandas as pd


#Import pyplot from matplotlib

#Matplotlib is the whole package; pylab is a module in matplotlib that gets insta
#pyplot is a module in matplotlib. Pyplot provides the state-machine interface to

import matplotlib.pyplot as plt


import random
#from itertools import combinations


import seaborn as sns

#import xgboost as xgb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import r2_score, confusion_matrix, classification_report

#from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import load_model
from pickle import dump, load

%matplotlib inline

#import pickle

import warnings
warnings.filterwarnings('ignore')

random.seed(77084017021)
```

```
Collecting tensorflow
  Downloading tensorflow-2.7.0-cp38-cp38-win_amd64.whl (430.8 MB)
Collecting tensorboard~=2.6
  Downloading tensorboard-2.7.0-py3-none-any.whl (5.8 MB)
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Collecting libclang>=9.0.1
  Downloading libclang-12.0.0-py2.py3-none-win_amd64.whl (13.1 MB)
Requirement already satisfied: keras<2.8,>=2.7.0rc0 in c:\users\byju\anaconda
3\lib\site-packages (from tensorflow) (2.7.0)
```

```
Collecting termcolor>=1.1.0
  Downloading termcolor-1.1.0.tar.gz (3.9 kB)
Collecting flatbuffers<3.0,>=1.12
  Downloading flatbuffers-2.0-py2.py3-none-any.whl (26 kB)
Requirement already satisfied: wheel<1.0,>=0.32.0 in c:\users\byju\anaconda3
\lib\site-packages (from tensorflow) (0.36.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\byju\anaconda3\lib\s
ite-packages (from tensorflow) (1.12.1)
Collecting grpcio<2.0,>=1.24.3
  Downloading grpcio-1.42.0-cp38-cp38-win_amd64.whl (3.3 MB)
Requirement already satisfied: six>=1.12.0 in c:\users\byju\anaconda3\lib\sit
e-packages (from tensorflow) (1.15.0)
Collecting absl-py>=0.4.0
  Downloading absl_py-1.0.0-py3-none-any.whl (126 kB)
Collecting gast<0.5.0,>=0.2.1
  Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Collecting astunparse>=1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting keras-preprocessing>=1.1.1
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
Collecting tensorflow-io-gcs-filesystem>=0.21.0
  Downloading tensorflow_io_gcs_filesystem-0.22.0-cp38-cp38-win_amd64.whl (1.
5 MB)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\byju\anac
onda3\lib\site-packages (from tensorflow) (3.7.4.3)
Collecting google-pasta>=0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: h5py>=2.9.0 in c:\users\byju\anaconda3\lib\sit
e-packages (from tensorflow) (2.10.0)
Requirement already satisfied: numpy>=1.14.5 in c:\users\byju\anaconda3\lib\s
ite-packages (from tensorflow) (1.20.1)
Collecting tensorflow-estimator<2.8,~=2.7.0rc0
  Downloading tensorflow_estimator-2.7.0-py2.py3-none-any.whl (463 kB)
Collecting protobuf>=3.9.2
  Downloading protobuf-3.19.1-cp38-cp38-win_amd64.whl (895 kB)
Collecting google-auth<3,>=1.6.3
  Downloading google_auth-2.3.3-py2.py3-none-any.whl (155 kB)
Collecting google-auth-oauthlib<0.5,>=0.4.1
  Downloading google_auth_oauthlib-0.4.6-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\byju\anaconda3
\lib\site-packages (from tensorboard~=2.6->tensorflow) (2.25.1)
Collecting tensorboard-data-server<0.7.0,>=0.6.0
  Downloading tensorboard_data_server-0.6.1-py3-none-any.whl (2.4 kB)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\byju\anaconda3\l
ib\site-packages (from tensorboard~=2.6->tensorflow) (1.0.1)
Requirement already satisfied: setuptools>=41.0.0 in c:\users\byju\anaconda3
\lib\site-packages (from tensorboard~=2.6->tensorflow) (52.0.0.post20210125)
Collecting tensorboard-plugin-wit>=1.6.0
  Downloading tensorboard_plugin_wit-1.8.0-py3-none-any.whl (781 kB)
Collecting markdown>=2.6.8
  Downloading Markdown-3.3.6-py3-none-any.whl (97 kB)
Collecting rsa<5,>=3.1.4
  Using cached rsa-4.8-py3-none-any.whl (39 kB)
Collecting pyasn1-modules>=0.2.1
  Using cached pyasn1_modules-0.2.8-py2.py3-none-any.whl (155 kB)
Collecting cachetools<5.0,>=2.0.0
  Using cached cachetools-4.2.4-py3-none-any.whl (10 kB)
```

```
Collecting requests-oauthlib>=0.7.0
  Downloading requests_oauthlib-1.3.0-py2.py3-none-any.whl (23 kB)
Collecting importlib-metadata>=4.4
  Downloading importlib_metadata-4.8.2-py3-none-any.whl (17 kB)
Requirement already satisfied: zipp>=0.5 in c:\users\byju\anaconda3\lib\site-
packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~=2.6->te
nsorflow) (3.4.1)
Collecting pyasn1<0.5.0,>=0.4.6
  Using cached pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\byju\anaconda3
\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow)
(2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\byju\anacond
a3\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow)
(1.26.4)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\byju\anaconda3\l
ib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (4.
0.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\byju\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2.10)
Collecting oauthlib>=3.0.0
  Downloading oauthlib-3.1.1-py2.py3-none-any.whl (146 kB)
Building wheels for collected packages: termcolor
  Building wheel for termcolor (setup.py): started
  Building wheel for termcolor (setup.py): finished with status 'done'
  Created wheel for termcolor: filename=termcolor-1.1.0-py3-none-any.whl size
=4829 sha256=ab21be95ae056eb53bf5b94619d56611d486cf3667f4b1b35d7cd564ac693578
  Stored in directory: c:\users\byju\appdata\local\pip\cache\wheels\a0\16\9c
\5473df82468f958445479c59e784896fa24f4a5fc024b0f501
Successfully built termcolor
Installing collected packages: pyasn1, rsa, pyasn1-modules, oauthlib, cacheto
ols, requests-oauthlib, importlib-metadata, google-auth, tensorboard-plugin-w
it, tensorboard-data-server, protobuf, markdown, grpcio, google-auth-oauthli
b, absl-py, termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, te
nsorboard, opt-einsum, libclang, keras-preprocessing, google-pasta, gast, fla
tbuffers, astunparse, tensorflow
  Attempting uninstall: importlib-metadata
    Found existing installation: importlib-metadata 3.10.0
    Uninstalling importlib-metadata-3.10.0:
      Successfully uninstalled importlib-metadata-3.10.0
Successfully installed absl-py-1.0.0 astunparse-1.6.3 cachetools-4.2.4 flatbu
ffers-2.0 gast-0.4.0 google-auth-2.3.3 google-auth-oauthlib-0.4.6 google-past
a-0.2.0 grpcio-1.42.0 importlib-metadata-4.8.2 keras-preprocessing-1.1.2 libc
lang-12.0.0 markdown-3.3.6 oauthlib-3.1.1 opt-einsum-3.3.0 protobuf-3.19.1 py
asn1-0.4.8 pyasn1-modules-0.2.8 requests-oauthlib-1.3.0 rsa-4.8 tensorboard-
2.7.0 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.0 tensorflow-
2.7.0 tensorflow-estimator-2.7.0 tensorflow-io-gcs-filesystem-0.22.0 termcolo
r-1.1.0
```

In [60]:
```
!pip install livelossplot
```

```
Collecting livelossplot
  Downloading livelossplot-0.5.4-py3-none-any.whl (22 kB)
Requirement already satisfied: ipython in c:\users\byju\anaconda3\lib\site-pa
ckages (from livelossplot) (7.22.0)
Requirement already satisfied: matplotlib in c:\users\byju\anaconda3\lib\site
-packages (from livelossplot) (3.3.4)
Requirement already satisfied: bokeh in c:\users\byju\anaconda3\lib\site-pack
ages (from livelossplot) (2.3.2)
Requirement already satisfied: packaging>=16.8 in c:\users\byju\anaconda3\lib
\site-packages (from bokeh->livelossplot) (20.9)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\byju\anaconda
3\lib\site-packages (from bokeh->livelossplot) (2.8.1)
Requirement already satisfied: PyYAML>=3.10 in c:\users\byju\anaconda3\lib\si
te-packages (from bokeh->livelossplot) (5.4.1)
Requirement already satisfied: typing-extensions>=3.7.4 in c:\users\byju\anac
onda3\lib\site-packages (from bokeh->livelossplot) (3.7.4.3)
Requirement already satisfied: Jinja2>=2.9 in c:\users\byju\anaconda3\lib\sit
e-packages (from bokeh->livelossplot) (2.11.3)
Requirement already satisfied: tornado>=5.1 in c:\users\byju\anaconda3\lib\si
te-packages (from bokeh->livelossplot) (6.1)
Requirement already satisfied: numpy>=1.11.3 in c:\users\byju\anaconda3\lib\s
ite-packages (from bokeh->livelossplot) (1.20.1)
Requirement already satisfied: pillow>=7.1.0 in c:\users\byju\anaconda3\lib\s
ite-packages (from bokeh->livelossplot) (8.2.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\byju\anaconda3\li
b\site-packages (from Jinja2>=2.9->bokeh->livelossplot) (1.1.1)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\byju\anaconda3\li
b\site-packages (from packaging>=16.8->bokeh->livelossplot) (2.4.7)
Requirement already satisfied: six>=1.5 in c:\users\byju\anaconda3\lib\site-p
ackages (from python-dateutil>=2.1->bokeh->livelossplot) (1.15.0)
Requirement already satisfied: colorama in c:\users\byju\anaconda3\lib\site-p
ackages (from ipython->livelossplot) (0.4.4)
Requirement already satisfied: decorator in c:\users\byju\anaconda3\lib\site-
packages (from ipython->livelossplot) (5.0.6)
Requirement already satisfied: pygments in c:\users\byju\anaconda3\lib\site-p
ackages (from ipython->livelossplot) (2.8.1)
Requirement already satisfied: pickleshare in c:\users\byju\anaconda3\lib\sit
e-packages (from ipython->livelossplot) (0.7.5)
Requirement already satisfied: backcall in c:\users\byju\anaconda3\lib\site-p
ackages (from ipython->livelossplot) (0.2.0)
Requirement already satisfied: traitlets>=4.2 in c:\users\byju\anaconda3\lib
\site-packages (from ipython->livelossplot) (5.0.5)
Requirement already satisfied: setuptools>=18.5 in c:\users\byju\anaconda3\li
b\site-packages (from ipython->livelossplot) (52.0.0.post20210125)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 i
n c:\users\byju\anaconda3\lib\site-packages (from ipython->livelossplot) (3.
0.17)
Requirement already satisfied: jedi>=0.16 in c:\users\byju\anaconda3\lib\site
-packages (from ipython->livelossplot) (0.17.2)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in c:\users\byju\anaconda3
\lib\site-packages (from jedi>=0.16->ipython->livelossplot) (0.7.0)
Requirement already satisfied: wcwidth in c:\users\byju\anaconda3\lib\site-pa
ckages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython->livelossp
lot) (0.2.5)
```

```
Requirement already satisfied: ipython-genutils in c:\users\byju\anaconda3\li
b\site-packages (from traitlets>=4.2->ipython->livelossplot) (0.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\byju\anaconda3\lib\si
te-packages (from matplotlib->livelossplot) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\byju\anaconda3\l
ib\site-packages (from matplotlib->livelossplot) (1.3.1)
Installing collected packages: livelossplot
Successfully installed livelossplot-0.5.4
```

In [62]:
```
from livelossplot import PlotLossesKeras
```

In [61]:
```
#load the dataframe
loan_df = pd.read_csv('loan_data.csv')
loan_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
```

In [20]:
```
#type() method returns class type of the argument(object) passed as parameter
type(loan_df)
```

Out[20]:
```
pandas.core.frame.DataFrame
```

In [21]:
```
# Print first few rows of the train data.
# Pandas head() method is used to return top n (5 by default) rows of a data fram
loan_df.head()
```

Out[21]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000

In [22]:
```
#Output the dimension of dataset
loan_df.shape # the dataset has 9578 rows and 14 columns
```

Out[22]:
```
(9578, 14)
```

In [ ]:
```
# structure of the train dataset
#loan_df.dtypes
```

In [24]:
```
# count of missing values
loan_df.isnull().sum() # There are no missing values
```

Out[24]:
```
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
```

In [28]:
```
#describe() method gives us summary statistics for numerical columns in the data
#(see below,purpose is not numerical, but categorical and hence missing)
#loan_df.describe(include='all').transpose()
loan_df.describe().transpose()  # Annual income is log value and hnece will be cc
```

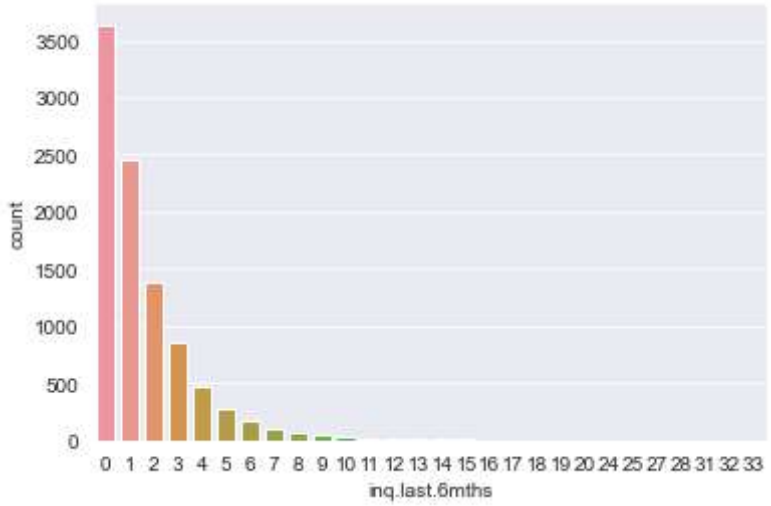
Out[28]:

	count	mean	std	min	25%	50%	
credit.policy	9578.0	0.804970	0.396245	0.000000	1.000000	1.000000	1.0
int.rate	9578.0	0.122640	0.026847	0.060000	0.103900	0.122100	0.1
installment	9578.0	319.089413	207.071301	15.670000	163.770000	268.950000	432.7
log.annual.inc	9578.0	10.932117	0.614813	7.547502	10.558414	10.928884	11.2
dti	9578.0	12.606679	6.883970	0.000000	7.212500	12.665000	17.9
fico	9578.0	710.846314	37.970537	612.000000	682.000000	707.000000	737.0
days.with.cr.line	9578.0	4560.767197	2496.930377	178.958333	2820.000000	4139.958333	5730.0
revol.bal	9578.0	16913.963876	33756.189557	0.000000	3187.000000	8596.000000	18249.5
revol.util	9578.0	46.799236	29.014417	0.000000	22.600000	46.300000	70.9
inq.last.6mths	9578.0	1.577469	2.200245	0.000000	0.000000	1.000000	2.0
delinq.2yrs	9578.0	0.163708	0.546215	0.000000	0.000000	0.000000	0.0
pub.rec	9578.0	0.062122	0.262126	0.000000	0.000000	0.000000	0.0
not.fully.paid	9578.0	0.160054	0.366676	0.000000	0.000000	0.000000	0.0

In [29]:
```
#Lets check the distribution of few features in the dataframe

loan_df['inq.last.6mths'].isnull().mean()
loan_df.groupby('inq.last.6mths')['inq.last.6mths'].count()/len(loan_df)
```

Out[29]:
```
inq.last.6mths
0     0.379724
1     0.257047
2     0.144498
3     0.090207
4     0.049593
5     0.029025
6     0.017227
7     0.010441
8     0.007517
9     0.004907
10    0.002401
11    0.001566
12    0.001566
13    0.000626
14    0.000626
15    0.000940
16    0.000313
17    0.000209
18    0.000418
19    0.000209
20    0.000104
24    0.000209
25    0.000104
27    0.000104
28    0.000104
31    0.000104
32    0.000104
33    0.000104
Name: inq.last.6mths, dtype: float64
```

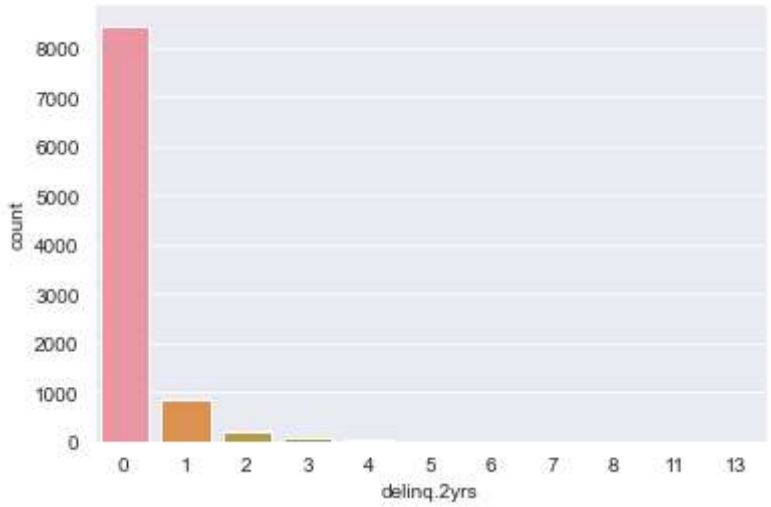
In [31]:
```
sns.set_style('darkgrid')
sns.countplot(x='inq.last.6mths', data=loan_df);
```



In [32]:
```
loan_df['delinq.2yrs'].isnull().mean()
loan_df.groupby('delinq.2yrs')['delinq.2yrs'].count()/len(loan_df)
```

Out[32]:
```
delinq.2yrs
0     0.883065
1     0.086866
2     0.020046
3     0.006786
4     0.001984
5     0.000626
6     0.000209
7     0.000104
8     0.000104
11    0.000104
13    0.000104
Name: delinq.2yrs, dtype: float64
```

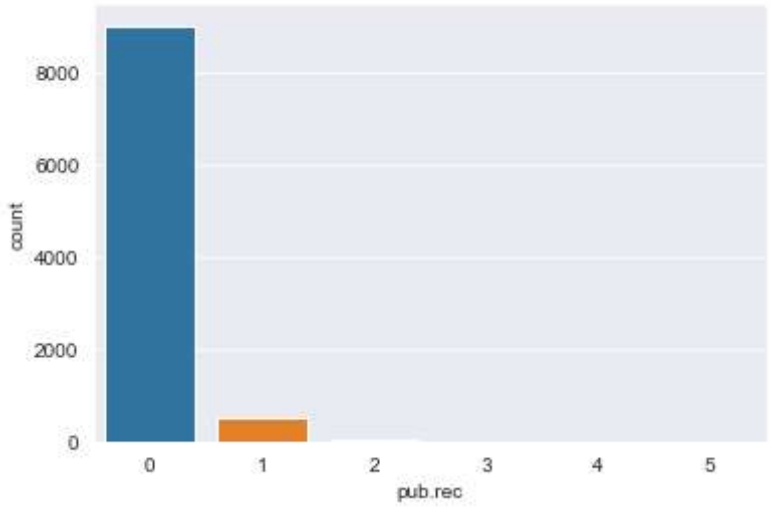
In [34]:
```
sns.set_style('darkgrid')
sns.countplot(x='delinq.2yrs', data=loan_df);
```



In [35]:
```
loan_df['pub.rec'].isnull().mean()
loan_df.groupby('pub.rec')['pub.rec'].count()/len(loan_df)
```

Out[35]:
```
pub.rec
0    0.941637
1    0.055648
2    0.001984
3    0.000522
4    0.000104
5    0.000104
Name: pub.rec, dtype: float64
```

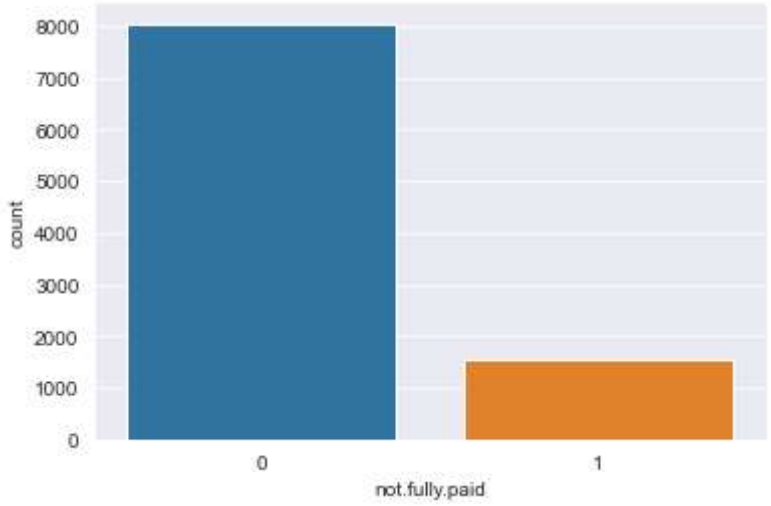
In [36]:
```
sns.set_style('darkgrid')
sns.countplot(x='pub.rec', data=loan_df);
```



In [37]:
```
loan_df['not.fully.paid'].isnull().mean()
loan_df.groupby('not.fully.paid')['not.fully.paid'].count()/len(loan_df)
```

Out[37]:
```
not.fully.paid
0    0.839946
1    0.160054
Name: not.fully.paid, dtype: float64
```

In [38]:
```
sns.set_style('darkgrid')
sns.countplot(x='not.fully.paid', data=loan_df);
```



As is evident from above exploratory analysis, the loan dataset is highly imbalanced

```
(eg: 83.9% of the records in dataset have “not.fully.paid =0 “, and only 16%
have “not.fully.paid=1”;
 similarly, 94.16% of the records in dataset have “pub.rec =0 “, and remaining
five levels represent
 <=6% of the records for this feature, and so on) and includes a lot of
features that makes this
problem more challenging. If we split the dataset into training and test sets,
ad perform a model training
with the train data, the prediction will be hence biased.
```

In [39]:
```
#Lets Transform categorical values into numerical values
cat_loan = loan_df.select_dtypes(include=['object']).copy()
cat_loan.head(8)
```

Out[39]:

	purpose
0	debt_consolidation
1	credit_card
2	debt_consolidation
3	debt_consolidation
4	credit_card
5	credit_card
6	debt_consolidation
7	all_other

In [40]:
```
s = cat_loan['purpose'].value_counts()
#s = pd.value_counts(loan_df.purpose)
s1 = pd.Series({'nunique': len(s), 'unique values': s.index.tolist()})
s.append(s1)
```

Out[40]:
```
debt_consolidation                                                  3957
all_other                                                           2331
credit_card                                                         1262
home_improvement                                                     629
small_business                                                       619
major_purchase                                                       437
educational                                                          343
nunique                                                                7
unique values         [debt_consolidation, all_other, credit_card, h...
dtype: object
```

In [41]:
```
#return the unique values in the categorical variable 'purpose' of loan_df as a N
loan_df['purpose'].unique()
```

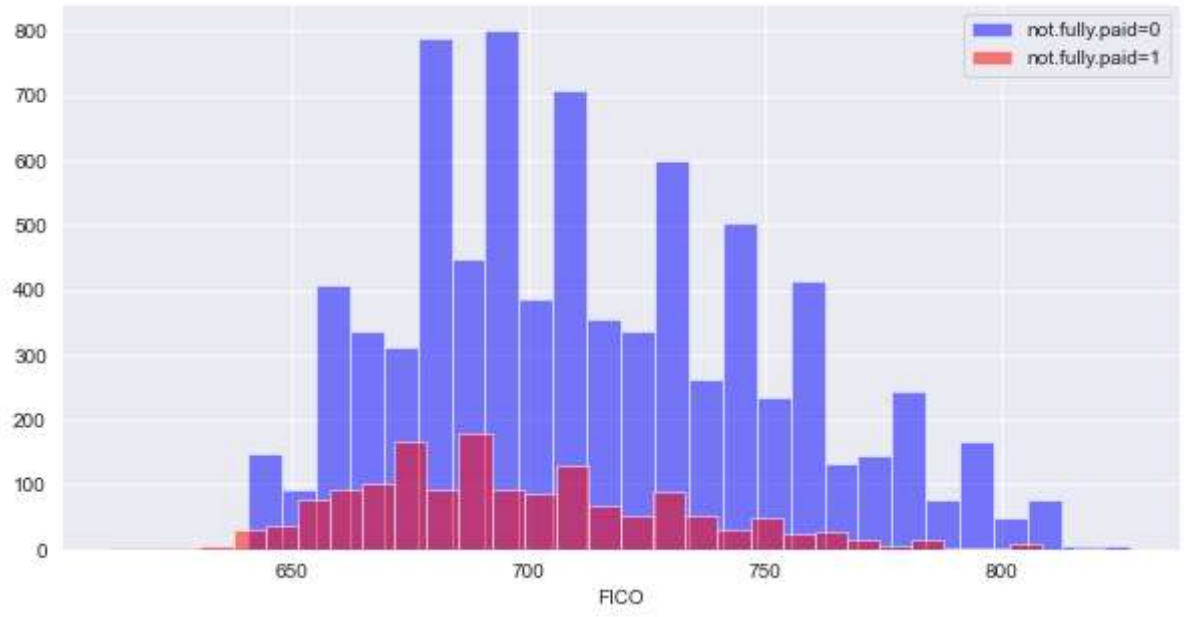
Out[41]:
```
array(['debt_consolidation', 'credit_card', 'all_other',
       'home_improvement', 'small_business', 'major_purchase',
       'educational'], dtype=object)
```

In [42]:
```
cat_loan = cat_loan.fillna({"purpose" : "credit_card"})
cleanup_nums = {"purpose": {"credit_card": 1,"debt_consolidation": 2 }}
cat_loan=cat_loan.replace(cleanup_nums)
cat_loan.head()
```

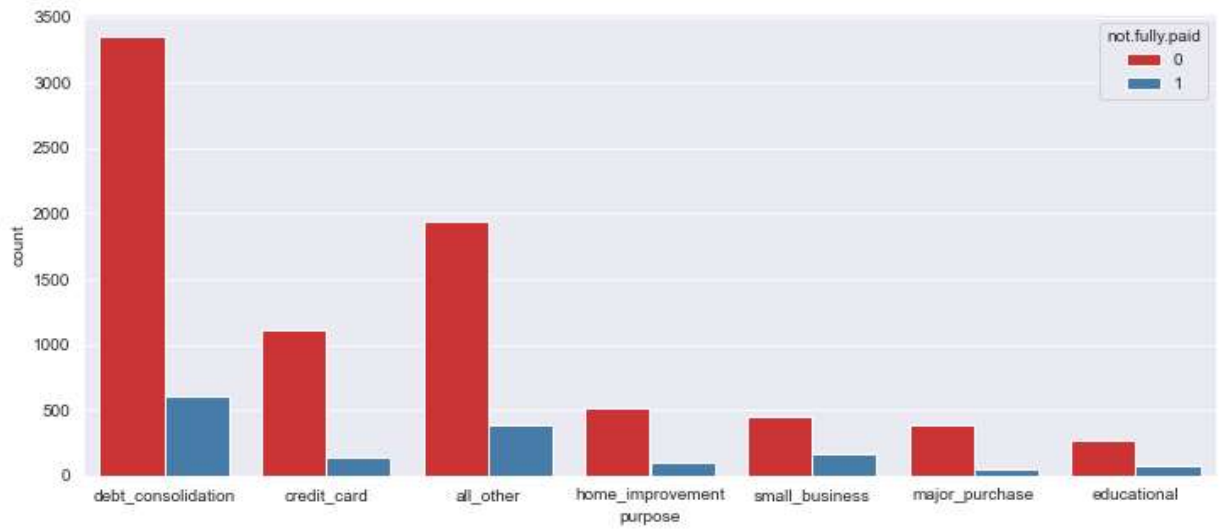
Out[42]:

	purpose
0	2
1	1
2	2
3	2
4	1

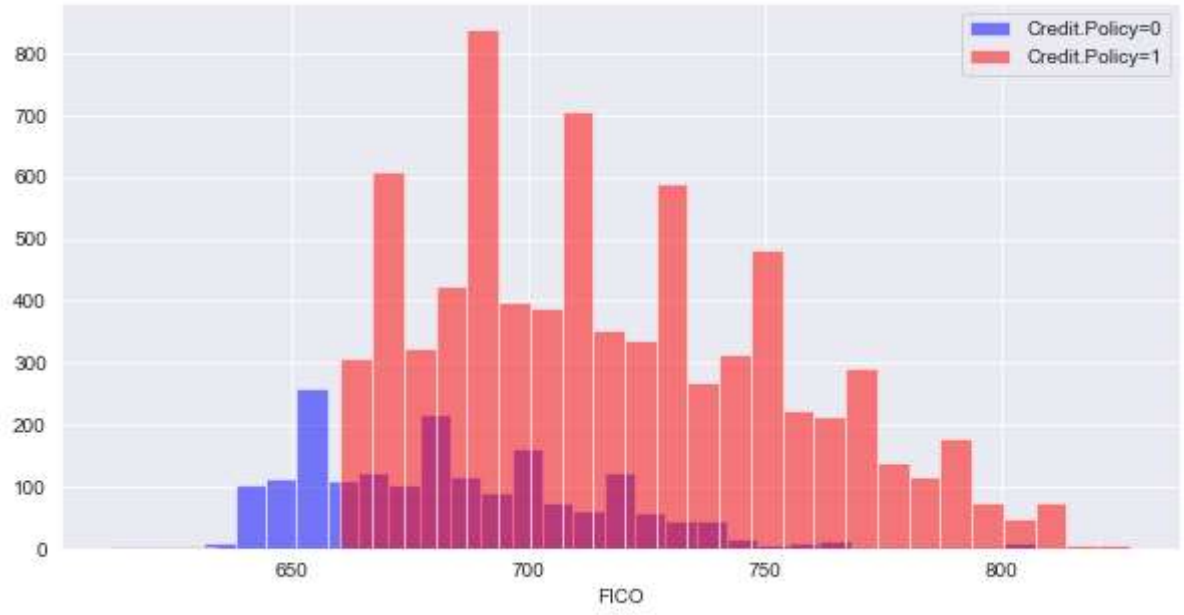
In [43]:
```
plt.figure(figsize=(10,5))
loan_df[loan_df['not.fully.paid']==0]['fico'].hist(alpha=0.5,color='blue',bins=30
loan_df[loan_df['not.fully.paid']==1]['fico'].hist(alpha=0.5,color='red',bins=30,
plt.legend()
plt.xlabel('FICO');
```



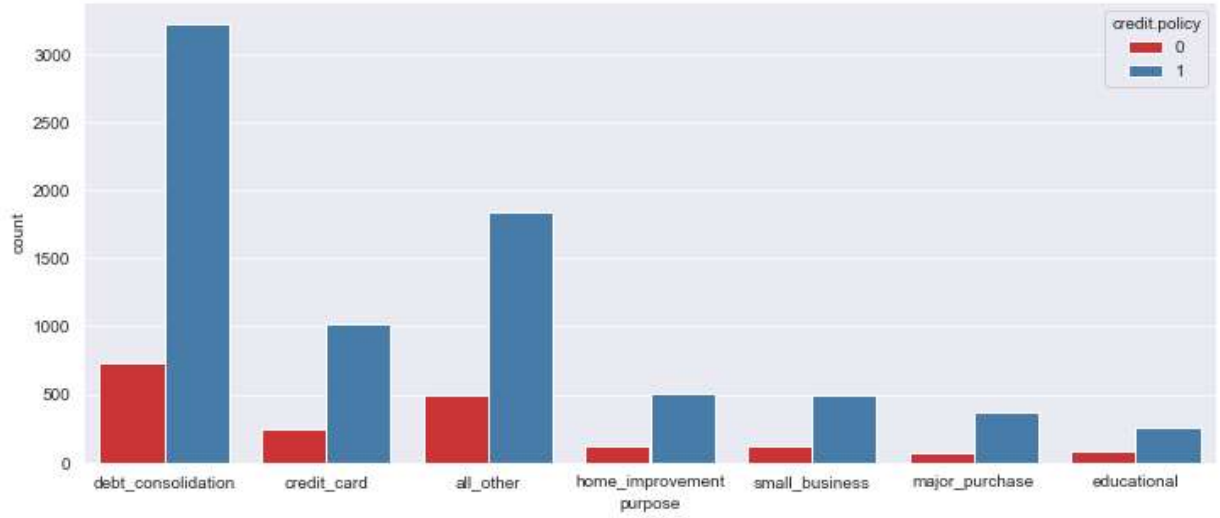
In [44]:
```
plt.figure(figsize=(12,5))
sns.countplot(x='purpose',hue='not.fully.paid',data=loan_df,palette='Set1');
```



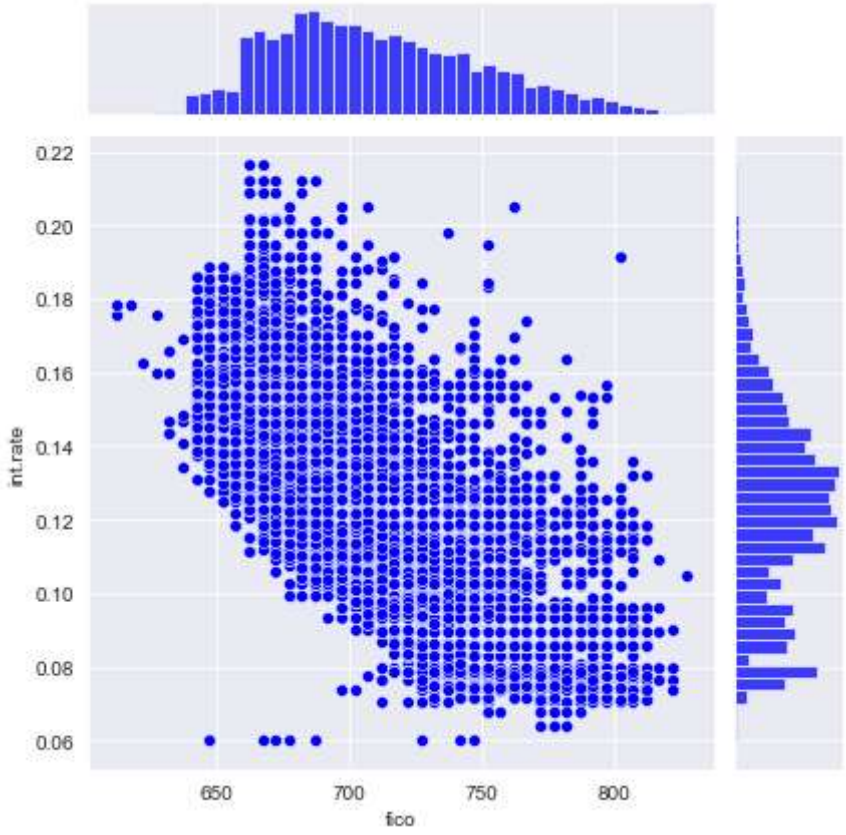
In [45]:
```
plt.figure(figsize=(10,5))
loan_df[loan_df['credit.policy']==0]['fico'].hist(alpha=0.5,color='blue',bins=30,
loan_df[loan_df['credit.policy']==1]['fico'].hist(alpha=0.5,color='red',bins=30,l
plt.legend()
plt.xlabel('FICO');
```



In [46]:
```
plt.figure(figsize=(12,5))
sns.countplot(x='purpose',hue='credit.policy',data=loan_df,palette='Set1');
```



In [47]:
```
sns.jointplot(x='fico',y='int.rate',data=loan_df,color='blue');
```

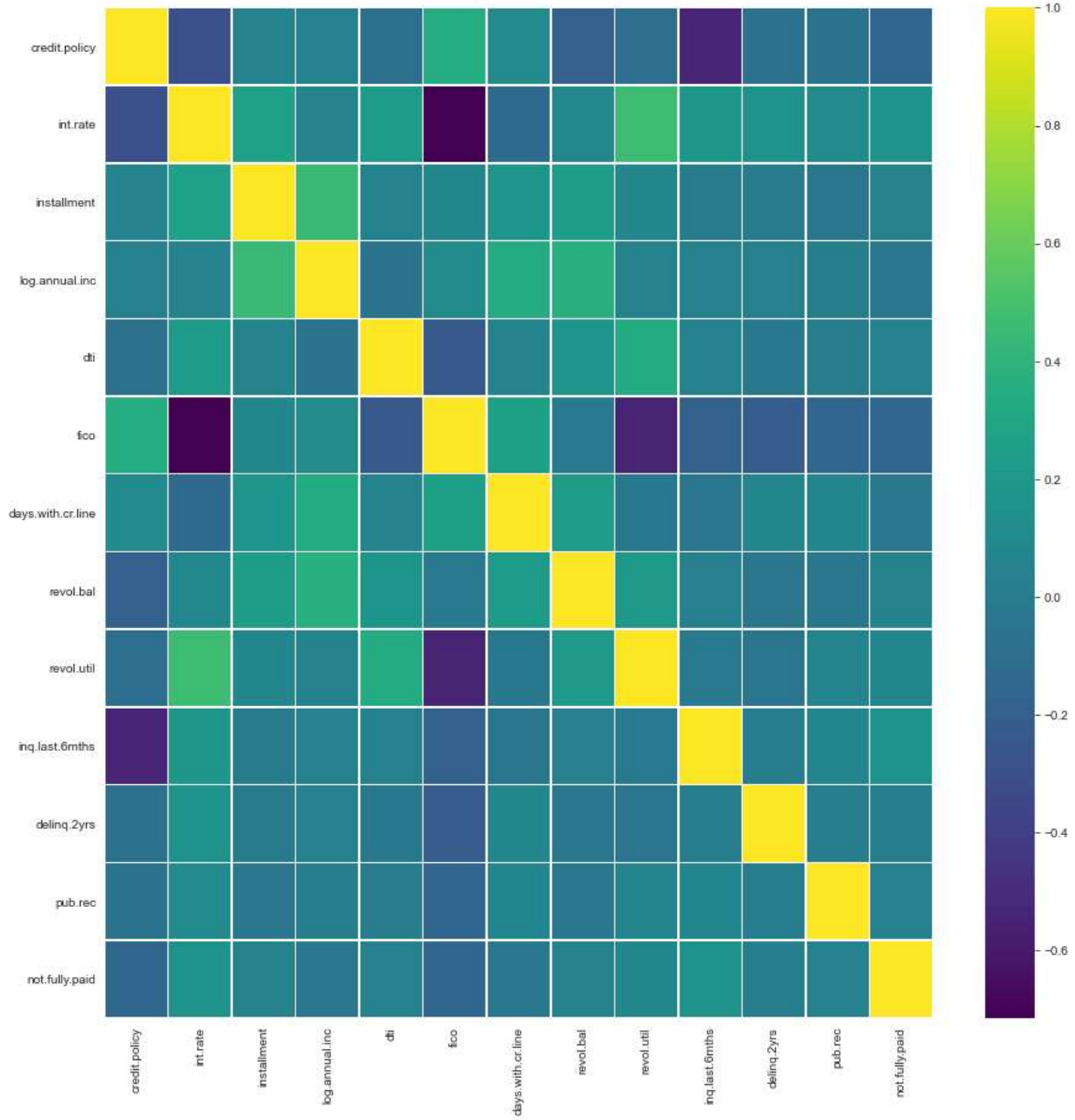


In [48]:
```
plt.figure(figsize=(11,7))
sns.lmplot(y='int.rate',x='fico',data=loan_df,hue='credit.policy',
           col='not.fully.paid',palette='Set2');
```

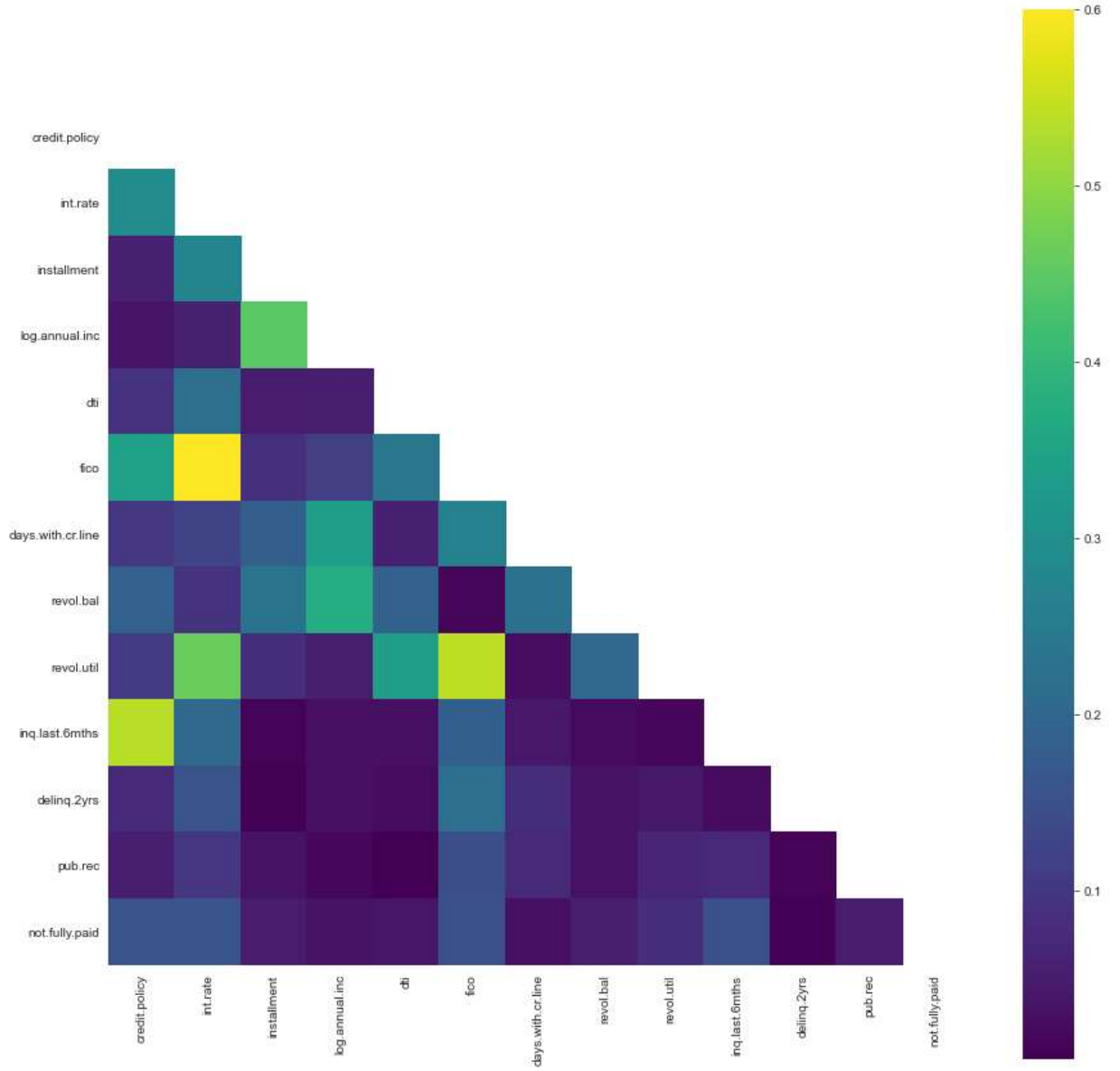
```
<Figure size 792x504 with 0 Axes>
```

In [49]:
```
#correlation
#loan_df1 = loan_df.drop('not.fully.paid', axis=1).values
#cor_matrix = loan_df.drop('not.fully.paid', axis=1).corr().abs()
cor_matrix = loan_df.corr().abs()
#print(cor_matrix)
#print only upper triangular part of correlation matrix
#upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),k=1).astype(np.bo
#print(upper_tri)
```

In [50]:
```
plt.figure(figsize=[15,15])
sns.heatmap(
       #data=loan_df.drop('not.fully.paid', axis=1).corr(),
        data=loan_df.corr(),
        cmap='viridis',
        annot=False,
        fmt='.2g',  linewidths=.5
);
```



In [51]:
```
#Use a mask to plot only part of a matrix
#corr = np.corrcoef(loan_df)
#corr = loan_df.drop('not.fully.paid', axis=1).corr().abs()
corr = loan_df.corr().abs()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(15, 15))
    ax = sns.heatmap(corr, mask=mask, vmax=0.6, square=True, cmap='viridis')
```

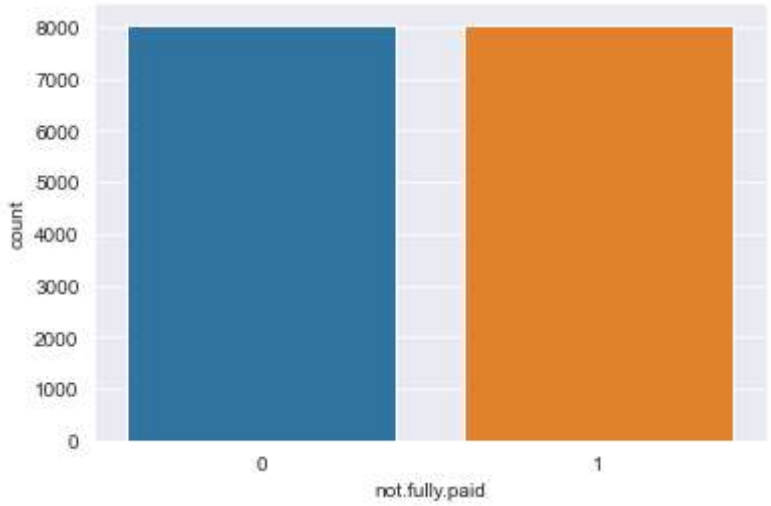



```
# There appears to be very high correlation between few pairs of feature
variables, such as:
# fico and int.rate, fico and revol.util, int.rate and revol.util,
credit.policy and inq.last.6mths, installment and log.annual.inc, etc.
# Dropping some of those features with strong correlation might help reduce the
number of features and proceed modeling with only the most relevant features
```

In [52]:
```
count_class_0, count_class_1 = loan_df['not.fully.paid'].value_counts()
loan_0 = loan_df[loan_df['not.fully.paid'] == 0]
loan_1 = loan_df[loan_df['not.fully.paid'] == 1]
loan_1_over = loan_1.sample(count_class_0, replace=True)
loan_test_over = pd.concat([loan_0, loan_1_over], axis=0)
print('Random over-sampling:')
print(loan_test_over['not.fully.paid'].value_counts())
```

```
Random over-sampling:
0    8045
1    8045
Name: not.fully.paid, dtype: int64
```

In [54]:
```
sns.set_style('darkgrid')
sns.countplot(x='not.fully.paid', data=loan_test_over);
```



In [55]:
```
cat_feature = ['purpose']
loan_df_final1 = pd.get_dummies(loan_test_over,columns=cat_feature,drop_first=Tru
loan_df_final1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16090 entries, 0 to 6840
Data columns (total 19 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   credit.policy               16090 non-null  int64
 1   int.rate                    16090 non-null  float64
 2   installment                 16090 non-null  float64
 3   log.annual.inc              16090 non-null  float64
 4   dti                         16090 non-null  float64
 5   fico                        16090 non-null  int64
 6   days.with.cr.line           16090 non-null  float64
 7   revol.bal                   16090 non-null  int64
 8   revol.util                  16090 non-null  float64
 9   inq.last.6mths              16090 non-null  int64
 10  delinq.2yrs                 16090 non-null  int64
 11  pub.rec                     16090 non-null  int64
 12  not.fully.paid              16090 non-null  int64
 13  purpose_credit_card         16090 non-null  uint8
 14  purpose_debt_consolidation  16090 non-null  uint8
 15  purpose_educational         16090 non-null  uint8
 16  purpose_home_improvement    16090 non-null  uint8
 17  purpose_major_purchase      16090 non-null  uint8
 18  purpose_small_business      16090 non-null  uint8
dtypes: float64(6), int64(7), uint8(6)
memory usage: 2.1 MB
```

In [72]:
```
loan_df_final1.head(10)
```

Out[72]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.u
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39
5	1	0.0788	125.13	11.904968	16.98	727	6120.041667	50807	51
8	1	0.1134	87.19	11.407565	17.25	682	3989.000000	69909	51
9	1	0.1221	84.12	10.203592	10.00	707	2730.041667	5630	23
10	1	0.1347	360.43	10.434116	22.09	677	6713.041667	13846	71
11	1	0.1324	253.58	11.835009	9.16	662	4298.000000	5122	18

In [74]:
```
to_train = loan_df_final1[loan_df_final1['not.fully.paid'].isin([0,1])]
to_pred = loan_df_final1[loan_df_final1['not.fully.paid'] == 2]
#to_pred.head()
```

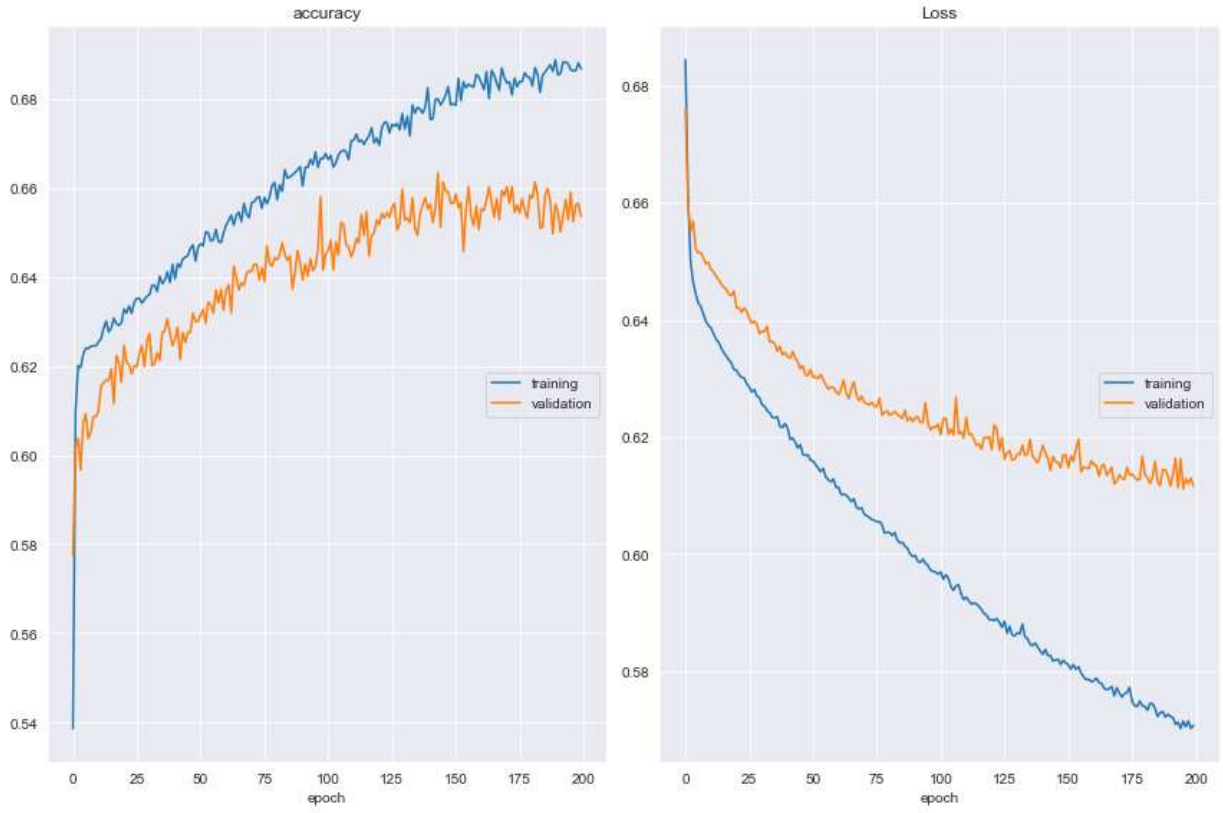
In [75]:
```
X = to_train.drop('not.fully.paid', axis=1).values
y = to_train['not.fully.paid'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

In [76]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [77]:
```
X_train.shape
```

Out[77]:
```
(11263, 18)
```

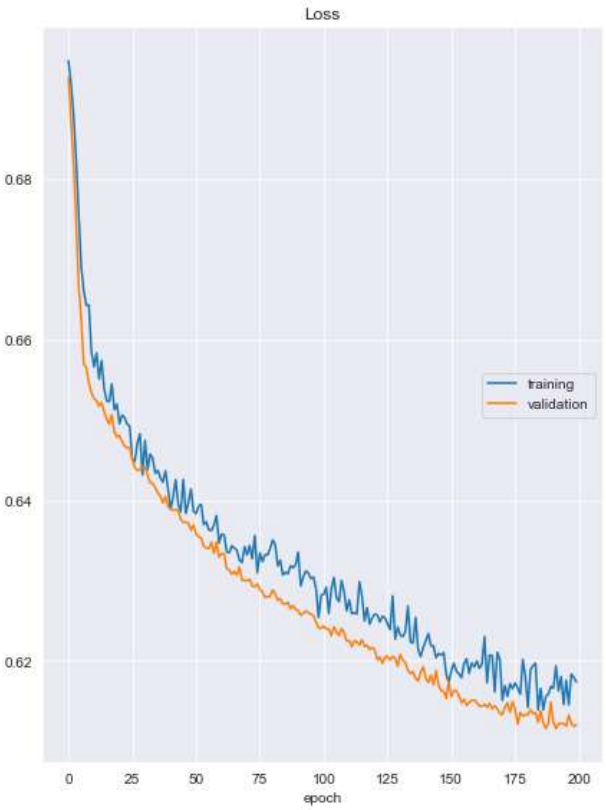
In [64]:
```
#Model without dropout

model = Sequential()

model.add(Dense(19, activation='relu'))

model.add(Dense(10, activation='relu'))


model.add(Dense(5, activation='relu'))

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

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=200, batch_size=256, validation_data=(X_test,
```




```
accuracy
        training                 (min:    0.539, max:    0.689, cur:    0.687)
        validation               (min:    0.577, max:    0.663, cur:    0.654)
Loss
        training                 (min:    0.570, max:    0.684, cur:    0.571)
        validation               (min:    0.611, max:    0.676, cur:    0.612)
44/44 [==============================] - 0s 9ms/step - loss: 0.5708 - accuracy:
0.6867 - val_loss: 0.6116 - val_accuracy: 0.6536
```

Out[64]:
```
<keras.callbacks.History at 0x17e43131970>
```

In [67]:
```
#Model with dropout
model_new = Sequential()

model_new.add(Dense(19, activation='relu'))

model_new.add(Dropout(0.2))

model_new.add(Dense(10, activation='relu'))

model_new.add(Dropout(0.2))

model_new.add(Dense(5, activation='relu'))

model_new.add(Dropout(0.2))

model_new.add(Dense(1, activation='sigmoid'))

model_new.compile(optimizer='adam', loss='binary_crossentropy', metrics=['binary_

model_new.fit(X_train, y_train, epochs=200, batch_size=256, validation_data=(X_te
```




```
binary_accuracy
        training                (min:    0.504, max:    0.661, cur:    0.65
6)
        validation              (min:    0.527, max:    0.658, cur:    0.65
1)
Loss
```

```
        training                 (min:    0.614, max:    0.695, cur:    0.61
7)
        validation               (min:    0.612, max:    0.693, cur:    0.61
2)
44/44 [==============================] - 0s 8ms/step - loss: 0.6174 - binary_
accuracy: 0.6563 - val_loss: 0.6120 - val_binary_accuracy: 0.6511
```

Out[67]:
```
<keras.callbacks.History at 0x17e42872910>
```

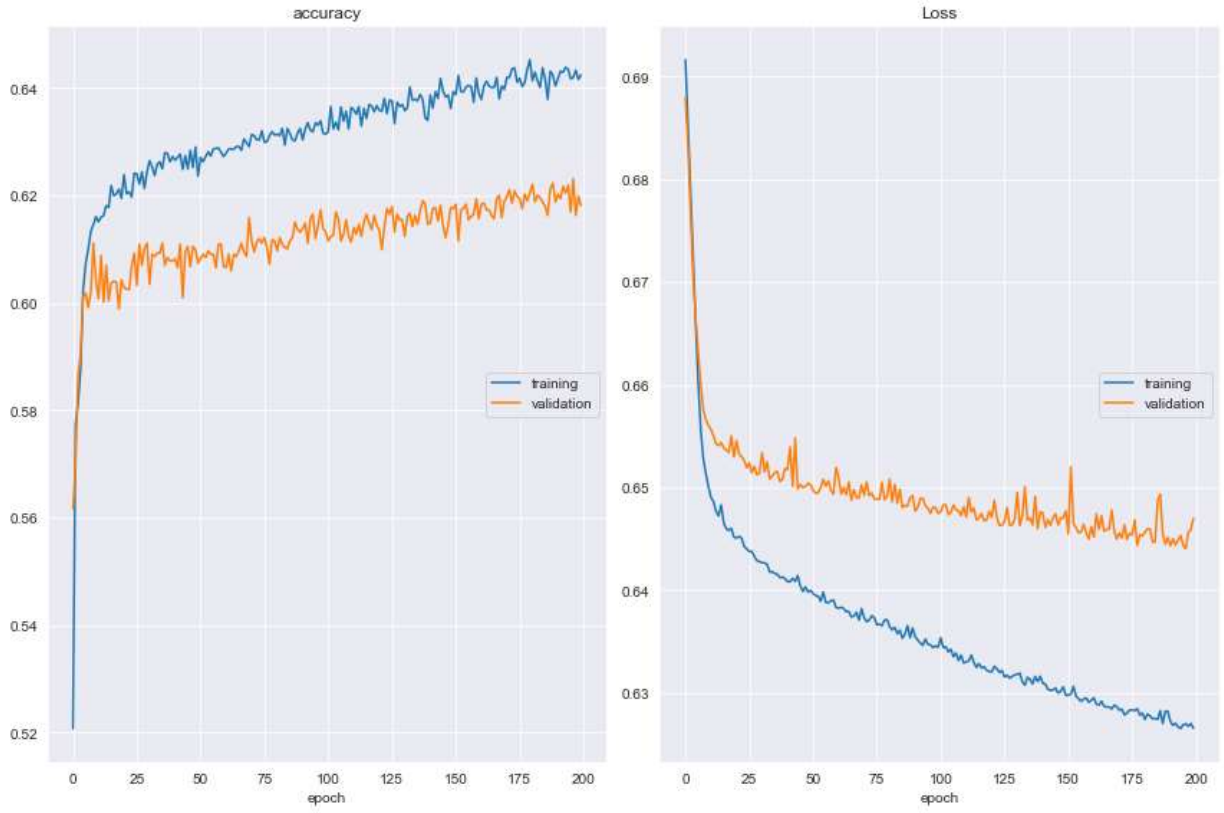
In [83]:
```
#Lets model after dropping some x variables that show strong correlation with oth
loan_df_final2 = loan_df_final1.drop(['int.rate', 'revol.util', 'inq.last.6mths',
```

In [84]:
```
to_train = loan_df_final2[loan_df_final2['not.fully.paid'].isin([0,1])]
#to_pred = loan_df_final2[loan_df_final2['not.fully.paid'] == 2]
#to_pred.head()

X = to_train.drop('not.fully.paid', axis=1).values
y = to_train['not.fully.paid'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [85]:
```
X_train.shape
```

Out[85]:
```
(11263, 14)
```

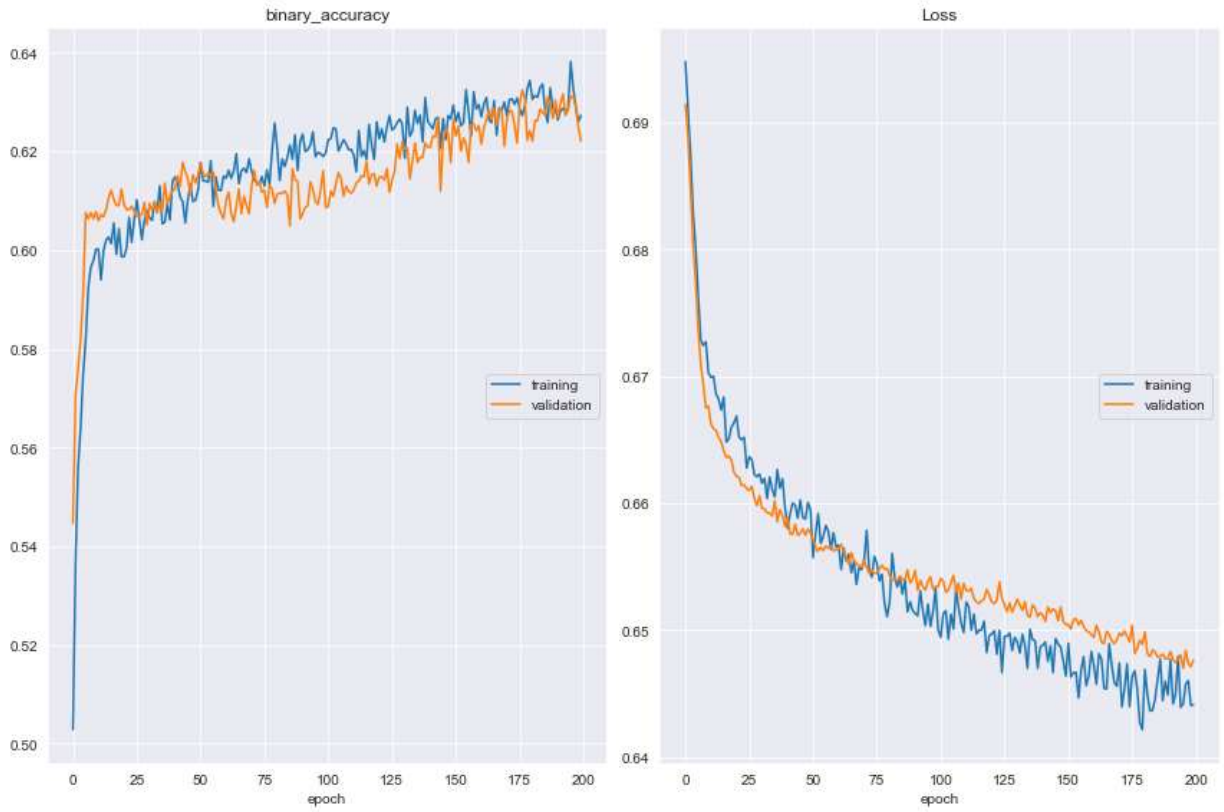
In [86]:
```
#Model without dropout after removing few correlated features (rfcf)

model_rfcf = Sequential()

model_rfcf.add(Dense(15, activation='relu'))

model_rfcf.add(Dense(10, activation='relu'))


model_rfcf.add(Dense(5, activation='relu'))

model_rfcf.add(Dense(1, activation='sigmoid'))

model_rfcf.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accura

model_rfcf.fit(X_train, y_train, epochs=200, batch_size=256, validation_data=(X_t
```




```
accuracy
        training                 (min:    0.521, max:    0.645, cur:    0.642)
        validation               (min:    0.562, max:    0.623, cur:    0.618)
Loss
        training                 (min:    0.627, max:    0.692, cur:    0.627)
        validation               (min:    0.644, max:    0.688, cur:    0.647)
44/44 [==============================] - 0s 8ms/step - loss: 0.6266 - accuracy:
0.6425 - val_loss: 0.6470 - val_accuracy: 0.6182
```

Out[86]: <keras.callbacks.History at 0x17e4f6f95e0>

In [87]:
```
#Model with dropout after removing few correlated features (rfcf)

model_new = Sequential()

model_new.add(Dense(15, activation='relu'))

model_new.add(Dropout(0.2))

model_new.add(Dense(10, activation='relu'))

model_new.add(Dropout(0.2))

model_new.add(Dense(5, activation='relu'))

model_new.add(Dropout(0.2))

model_new.add(Dense(1, activation='sigmoid'))

model_new.compile(optimizer='adam', loss='binary_crossentropy', metrics=['binary_

model_new.fit(X_train, y_train, epochs=200, batch_size=256, validation_data=(X_te
```




```
binary_accuracy
        training                 (min:    0.503, max:    0.638, cur:    0.627)
        validation               (min:    0.545, max:    0.632, cur:    0.622)
Loss
        training                 (min:    0.642, max:    0.695, cur:    0.644)
        validation               (min:    0.647, max:    0.691, cur:    0.648)
44/44 [==============================] - 0s 9ms/step - loss: 0.6441 - binary_ac
curacy: 0.6273 - val_loss: 0.6476 - val_binary_accuracy: 0.6221
```

Out[87]: <keras.callbacks.History at 0x17e4315e130>

In [ ]: