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 Scenario:

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:

1. Feature Transformation

```
Transform categorical values into numerical values (discrete)
```

2. Exploratory data analysis of different factors of the dataset.
3. 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
```

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
```

Import necessary modules

In [170…

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

pd.options.display.max_columns = 999
```

Load Data

In [171…

```
import os
for dirname, _, filenames in os.walk('.\input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
.\input\loan_data.csv
```

In [172…

```
file=r'.\input\loan_data.csv'
```

In [173…

```
df=pd.read_csv(file)
print("Input Data has ", df.shape[0],"Rows and ",df.shape[1], "Columns", )
```

```
Input Data has  9578 Rows and  14 Columns
```

In [174…

```
df.sample(5)
```

Out[174…

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths
4451	1	credit_card	0.1183	106.03	10.910186	20.96	702	3630.000000	16948	83.5	0
7315	1	debt_consolidation	0.1136	789.87	11.502875	19.79	732	4650.041667	21459	52.0	0
3419	1	all_other	0.1284	84.05	10.275051	17.17	692	2940.000000	1687	39.2	0
4048	1	all_other	0.1284	268.95	10.308953	4.28	687	4949.958333	1602	76.3	0
2921	1	debt_consolidation	0.1474	345.37	11.736101	9.34	677	5762.000000	12403	39.8	2

In [175…

```
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 [176…

```
df.describe(include='all')
```

Out[176…

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal
count	9578.000000	9578	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9.578000e+03
unique	NaN	7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	debt_consolidation	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	3957	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	0.804970	NaN	0.122640	319.089413	10.932117	12.606679	710.846314	4560.767197	1.691396e+04
std	0.396245	NaN	0.026847	207.071301	0.614813	6.883970	37.970537	2496.930377	3.375619e+04
min	0.000000	NaN	0.060000	15.670000	7.547502	0.000000	612.000000	178.958333	0.000000e+00
25%	1.000000	NaN	0.103900	163.770000	10.558414	7.212500	682.000000	2820.000000	3.187000e+03
50%	1.000000	NaN	0.122100	268.950000	10.928884	12.665000	707.000000	4139.958333	8.596000e+03
75%	1.000000	NaN	0.140700	432.762500	11.291293	17.950000	737.000000	5730.000000	1.824950e+04
max	1.000000	NaN	0.216400	940.140000	14.528354	29.960000	827.000000	17639.958330	1.207359e+06

In [177…

```
df.describe().transpose()
```

Out[177…

	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

Check for missing values or null values

In [178…

```
df.isnull().sum().sum()
```

Out[178…

```
0
```

In [179…

```
NAs= pd.concat([df.isnull().sum()],axis=1,keys=['Count of NANs'])
NAs.sample(14)
```

Out[179…

	Count of NANs
revol.util	0
fico	0
installment	0
delinq.2yrs	0
not.fully.paid	0
pub.rec	0
inq.last.6mths	0
days.with.cr.line	0
log.annual.inc	0
revol.bal	0
credit.policy	0
int.rate	0
purpose	0
dti	0

In [180…

```
NAs[NAs.sum(axis=1)>0]
```

Out[180…

Count of NANs

It is good that there is no missing values or no null values

Now let us have a look at the data type of all the variables present in the input dataset

In [181…

```
# Now let us have a look at the data type of all the variables present in the training dataset
dtype_data=df.dtypes.reset_index()
dtype_data.columns = ["Count", "Column Type"]
dtype_data.groupby("Column Type").aggregate('count').reset_index()
```

Out[181…

	Column Type	Count
0	int64	7
1	float64	6
2	object	1

So 7 columns are integers with 1 object columns and 6 float column

Analyze the column types of input data

In [182…

```
# posible data types in pandas
numerics=['int16','int32','int64','float16','float32','float64']  # numeric
objects = ['O']

df_num = df.select_dtypes(include=numerics)
df_cat = df.select_dtypes(include=objects)

print(df_cat.columns)
print(df_num.columns)
```

```
Index(['purpose'], dtype='object')
Index(['credit.policy', 'int.rate', 'installment', 'log.annual.inc', 'dti',
       'fico', 'days.with.cr.line', 'revol.bal', 'revol.util',
       'inq.last.6mths', 'delinq.2yrs', 'pub.rec', 'not.fully.paid'],
      dtype='object')
```

Check the label "no.fully.paid" distribution in the dataset.

In [183…

```
df['not.fully.paid'].isnull().mean()
df.groupby('not.fully.paid')['not.fully.paid'].count()/len(df)
```

Out[183…

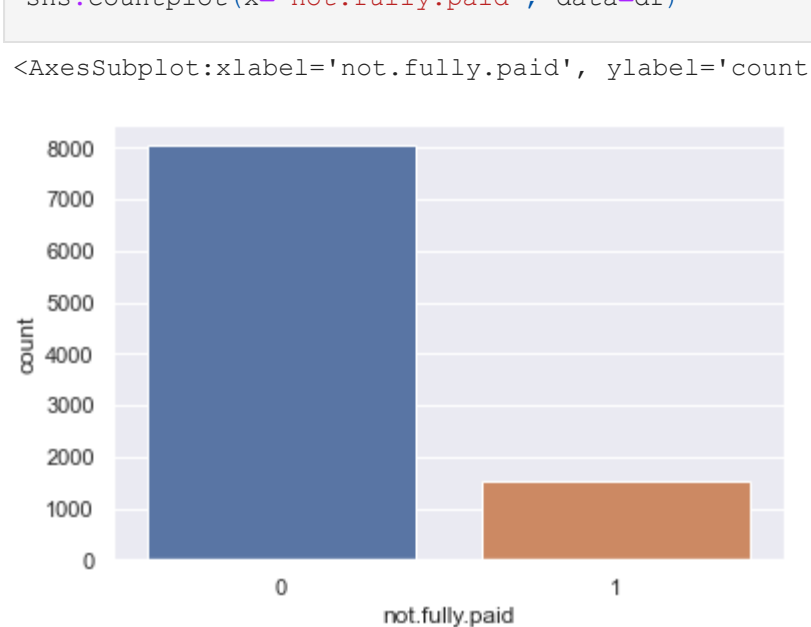
```
not.fully.paid
0    0.839946
1    0.160054
Name: not.fully.paid, dtype: float64
```

In [184…

```
sns.set_style('darkgrid')
sns.countplot(x='not.fully.paid', data=df)
```

Out[184…

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



The above shows, This dataset is highly imbalanced and includes features that make this problem more challenging. If we do model training with this data, the prediction will be biased since the "not.fully.paid =0 " has 83.9% filled, and only 16% is the "not.fully.paid=1"

To handle the imbalance of target variable "not.fully.paid" **Oversampling** is used when the quantity of data is insufficient.It tries to balance the dataset by increasing the size of rare samples.

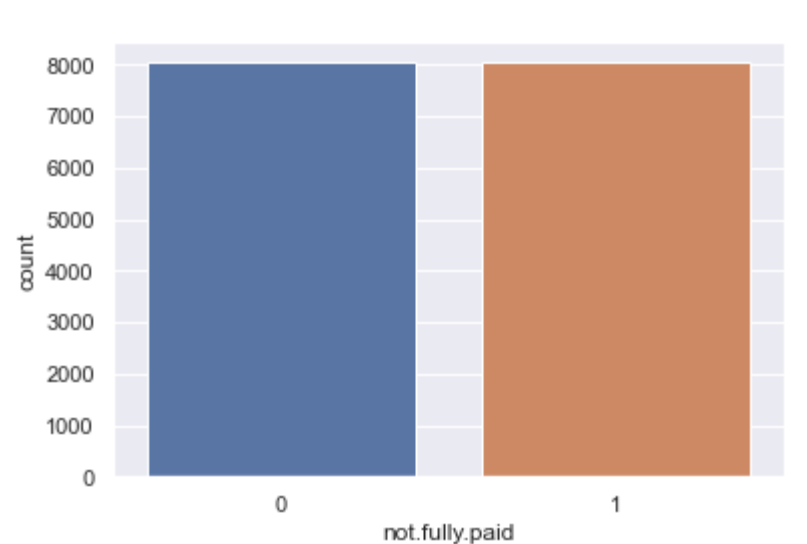
Over-sample the minority class with or w/o replacement by making the number of positive and negative examples equal. We'll add around 6500 samples from the training data set with this strategy. It's a lot more computationally expensive than under-sampling. Also, it's more prune to overfitting due to repeated examples.

In [185…

```
count_class_0, count_class_1 = df['not.fully.paid'].value_counts()
df_0 = df[df['not.fully.paid'] == 0]
df_1 = df[df['not.fully.paid'] == 1]
df_1_over = df_1.sample(count_class_0, replace=True)
df_over = pd.concat([df_0, df_1_over], axis=0)

print('Random over-sampling:')
print(df_over['not.fully.paid'].value_counts())

sns.set_style('darkgrid')
sns.countplot(x='not.fully.paid', data=df_over)
```

```
Random over-sampling:
0    8045
1    8045
Name: not.fully.paid, dtype: int64
<AxesSubplot:xlabel='not.fully.paid', ylabel='count'>
```

Out[185…



In [186…

```
df_over = df_over.reset_index()
df_over[df_over.index.duplicated()]
```

Out[186…

index	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs

Categorical Variables:

In [187…

```
dtype_data.loc[1:1,:]
```

Out[187…

	Count	Column Type
1	purpose	object

So purpose is the only object column

In [188…

```
df_over[['purpose']].sample(10)
```

Out[188…

	purpose
6533	all_other
6275	credit_card
12066	all_other
6034	debt_consolidation
10562	major_purchase
1563	small_business
281	credit_card
13139	all_other
8030	debt_consolidation
2897	all_other

The column purpose is categorical

Looking into categorical feature

In [189…

```
# display distinct value for each categorical feature
df_cat = df_over.select_dtypes(include=objects)
for col_name in df_cat.columns:
    print('The unique values in ' +col_name+ ' are: ', df_cat[col_name].nunique())
    print(df_cat[col_name].unique())
```

```
The unique values in purpose are:  7
['debt_consolidation' 'credit_card' 'home_improvement' 'small_business'
 'major_purchase' 'all_other' 'educational']
```

In [190…

```
df_over['purpose'].value_counts()
```

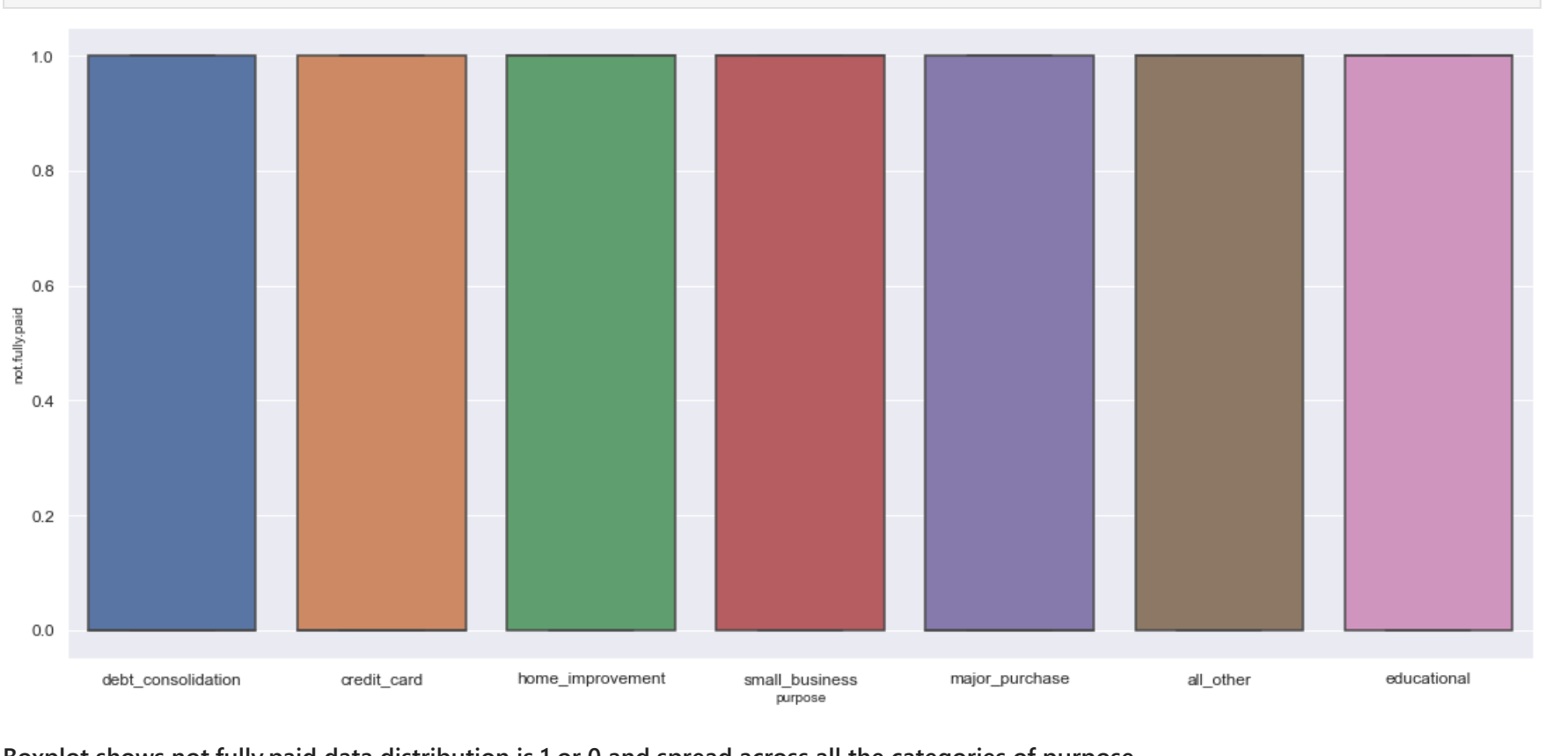
Out[190…

```
debt_consolidation    6460
all_other             4037
credit_card           1902
small_business        1352
home_improvement      1062
major_purchase         643
educational            634
Name: purpose, dtype: int64
```

Analyze the not.fully.paid - for cat column purpose
* To assess the usefulness of categorical feature we shall use boxplot

In [192…

```
# value of credit.policy(target variable) change for categorical column purpose .....
cols=['purpose']

for col in cols:
    plt.figure(figsize=(18.7,8.27))
    sns.boxplot(x=col,y='not.fully.paid',data=df_over)
    plt.xlabel(col,fontsize=10)
    plt.ylabel('not.fully.paid',fontsize=10)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
```

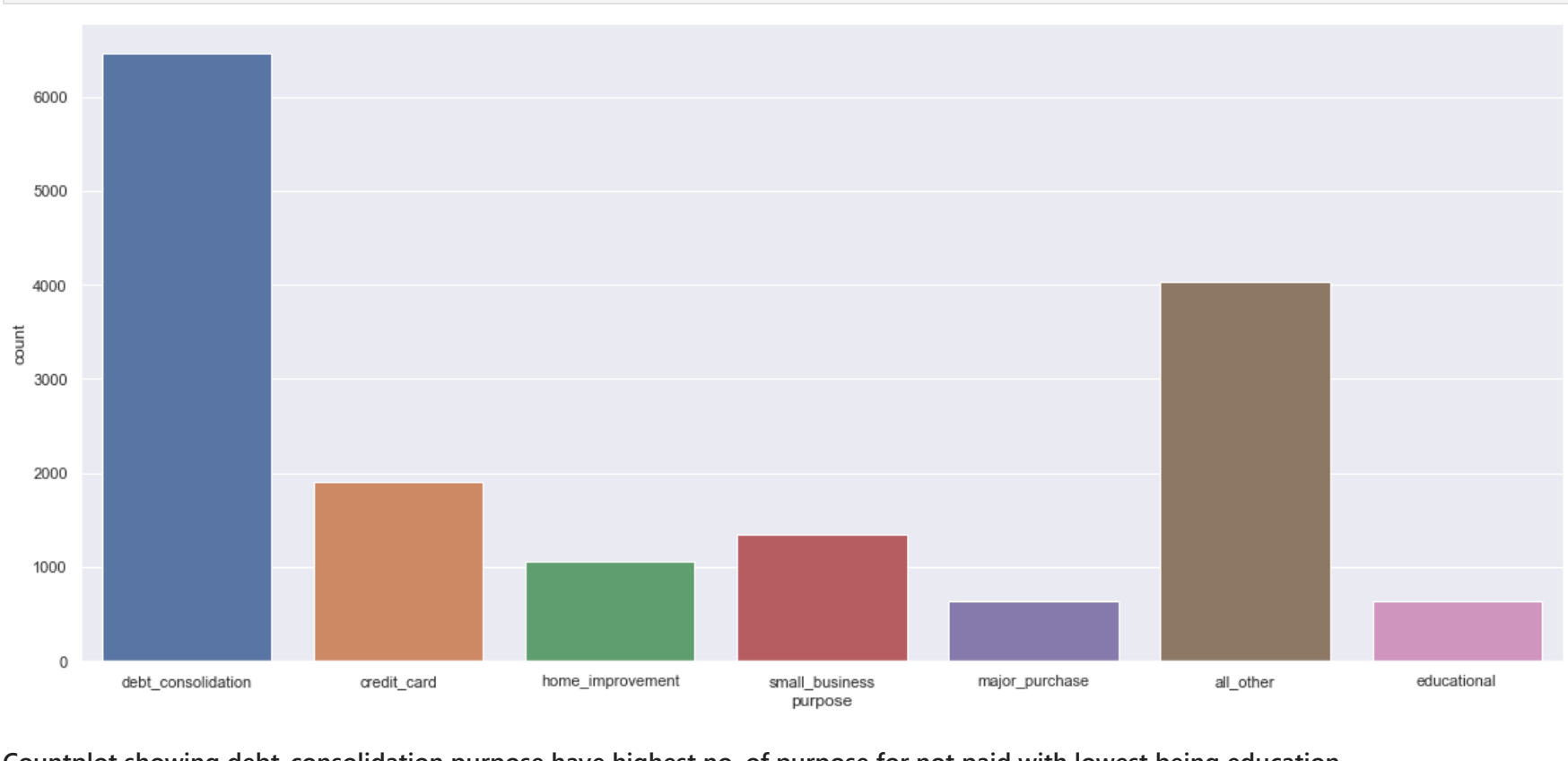


Boxplot shows not.fully.paid data distribution is 1 or 0 and spread across all the categories of purpose.

Let us see the count of different purpose

In [193…

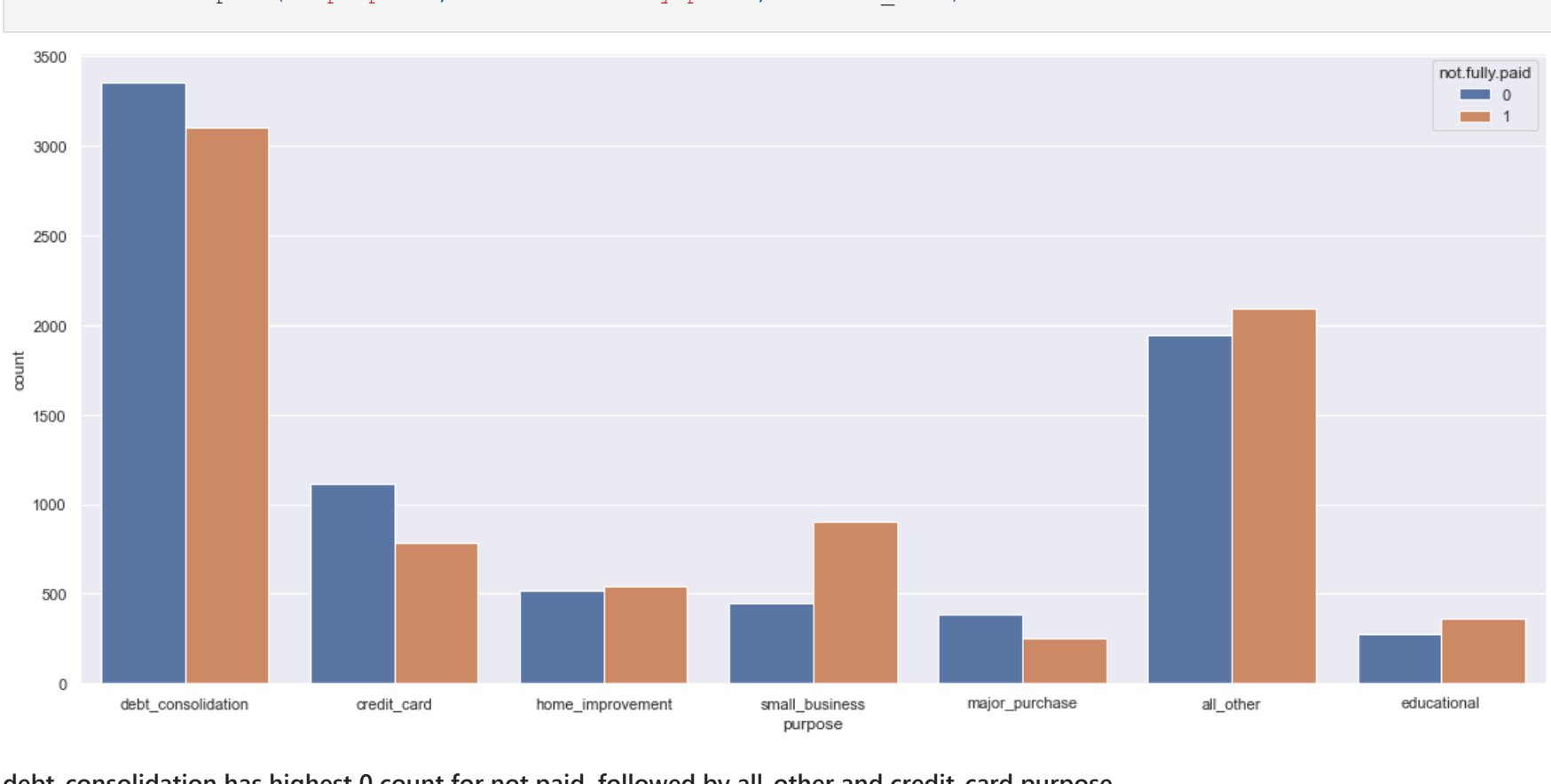
```
sns.set_theme(style="darkgrid")
sns.set(rc={'figure.figsize':(18.7,8.27)})
ax = sns.countplot(x="purpose",  data=df_over)
```



Countplot showing debt_consolidation purpose have highest no. of purpose for not paid with lowest being education

In [219…

```
sns.set_theme(style="darkgrid")
sns.set(rc={'figure.figsize':(18.7,8.27)})
ax = sns.countplot(x="purpose",  hue="not.fully.paid", data=df_over)
```



debt_consolidation has highest 0 count for not paid, followed by all_other and credit_card purpose

Observation:
The categorical column Purpose is having random distribution for every category and would contribute to the final credit policy target

Feature Transformation

Transform categorical values into numerical values (discrete)

In [203…

```
col_fea = ['purpose']
final_data = pd.get_dummies(df_over,columns=col_fea,drop_first=True)
final_data = final_data.drop(["index"], axis=1)
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16090 entries, 0 to 16089
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: 1.7 MB
```

Exploratory data analysis of different factors of the dataset.

Create a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.

In [196…

```
plt.figure(figsize=(10,6))
df_over[df_over['credit.policy']==1]['fico'].hist(alpha=0.5,color='blue',bins=30,label='Credit.Policy=1')
df_over[df_over['credit.policy']==0]['fico'].hist(alpha=0.5,color='red',bins=30,label='Credit.Policy=0')
plt.legend()
plt.xlabel('FICO')
```

Out[196…

```
Text(0.5, 0, 'FICO')
```

Let's see a similar chart for "not.fully.paid" column.

In [197…

```
plt.figure(figsize=(10,6))
df_over[df_over['not.fully.paid']==1]['fico'].hist(alpha=0.5,color='blue',bins=30,label='not.fully.paid=1')
df_over[df_over['not.fully.paid']==0]['fico'].hist(alpha=0.5,color='red',bins=30,label='not.fully.paid=0')
plt.legend()
plt.xlabel('FICO')
```

Out[197…

```
Text(0.5, 0, 'FICO')
```

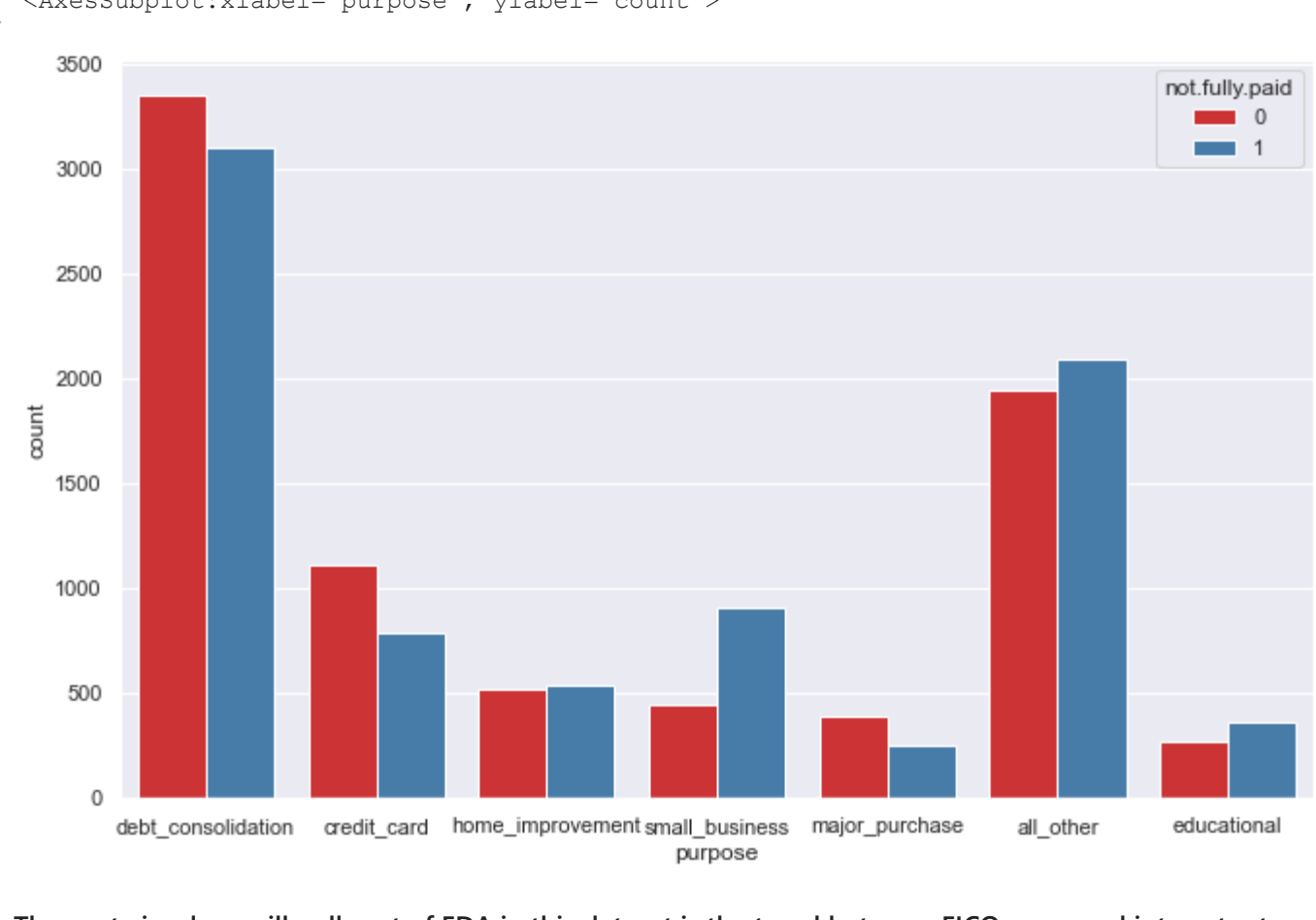
Now, check the dataset group by loan purpose. Create a countplot with the color hue defined by not.fully.paid.

In [198…

```
plt.figure(figsize=(11,7))
sns.countplot(x='purpose',hue='not.fully.paid',data=df_over,palette='Set1')
```

Out[198…

```
<AxesSubplot:xlabel='purpose', ylabel='count'>
```



The next visual we will pull part of EDA in this dataset is the trend between FICO score and interest rate.

In [199…

```
sns.jointplot(x='fico',y='int.rate',data=df_over,color='purple')
```

Out[199…

```
<seaborn.axisgrid.JointGrid at 0x1fff448c490>
```

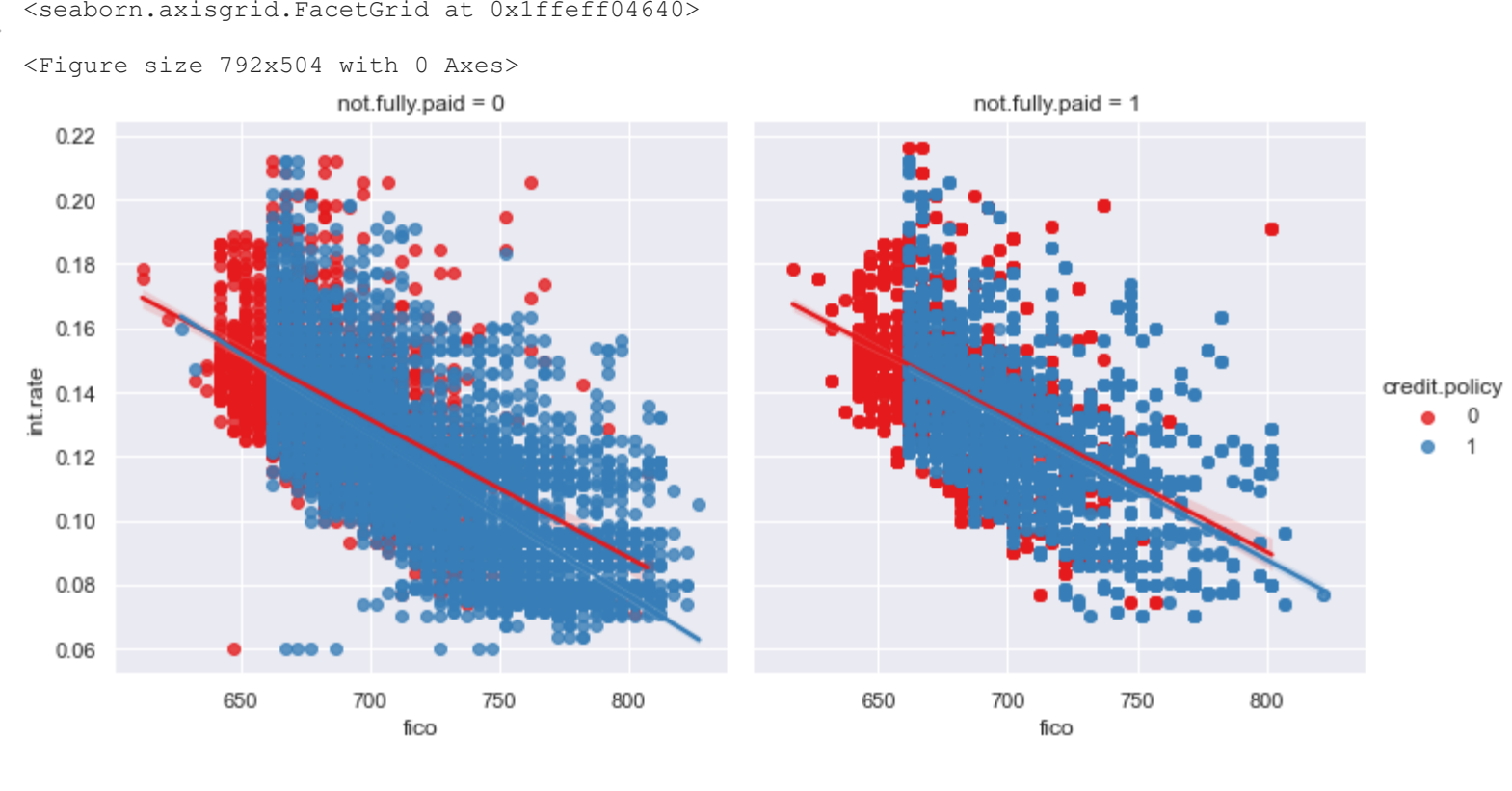
To compare the trend between not.fully.paid and credit.policy, create seaborn implot.

In [200…

```
plt.figure(figsize=(11,7))
sns.lmplot(y='int.rate',x='fico',data=df_over,hue='credit.policy',
           col='not.fully.paid',palette='Set1')
```

Out[200…

```
<seaborn.axisgrid.FacetGrid at 0x1ffeff04640>
<Figure size 792x504 with 0 Axes>
```



Additional Feature Engineering

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

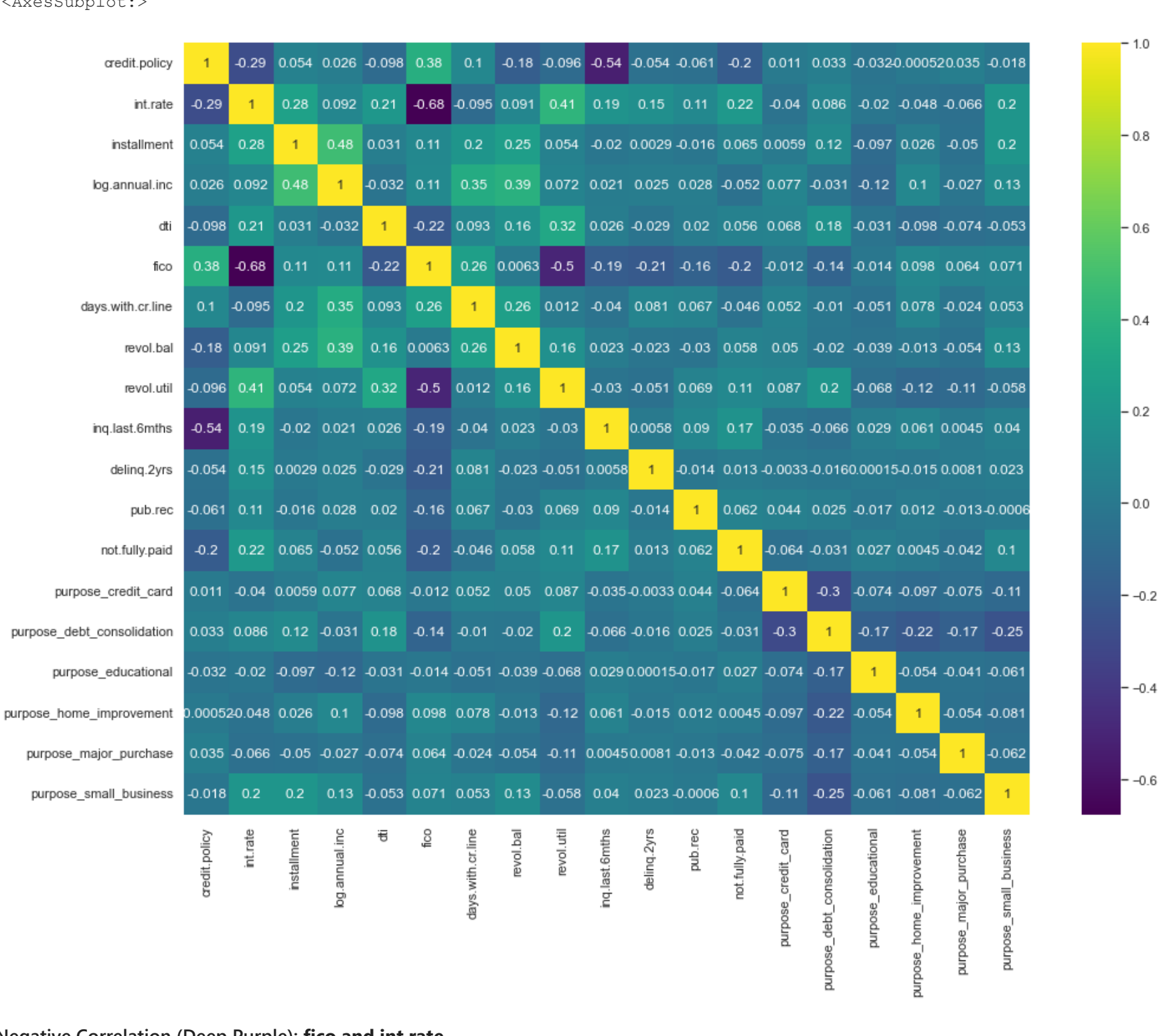
This will help reduce the number of features and will leave you with the most relevant features

In [205…

```
final_data.corr()
plt.figure(
        figsize=[16,12]
)
sns.heatmap(
        data=final_data.corr(),
        cmap='viridis',
        annot=True,
        fmt='.2g'
)
```

Out[205…

```
<AxesSubplot:>
```



Negative Correlation (Deep Purple): **fico and int.rate**

Positive Correlation (Light Green): **credit.policy and fico int.rate and revol.util installment and log.annual.inc**

log.annual.inc and revol.bal,days.with.cr.line,installment

After comparing the correlation and checking the feature description, **we can drop:'revol.bal', 'days.with.cr.line', 'installment', 'revol.util'.**

revol.bal, day.with.cr.line, installment can be represented by log.annual.inc. revol.util can represent by int.rate.

In [206…

```
features_to_drop=['revol.bal', 'days.with.cr.line', 'installment','revol.util']
final_data_fe=final_data.drop(features_to_drop,axis=1)
final_data_fe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16090 entries, 0 to 16089
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   credit.policy               16090 non-null  int64  
 1   int.rate                    16090 non-null  float64
 2   log.annual.inc              16090 non-null  float64
 3   dti                         16090 non-null  float64
 4   fico                        16090 non-null  int64  
 5   inq.last.6mths              16090 non-null  int64  
 6   delinq.2yrs                 16090 non-null  int64  
 7   pub.rec                     16090 non-null  int64  
 8   not.fully.paid              16090 non-null  int64  
 9   purpose_credit_card         16090 non-null  uint8  
 10  purpose_debt_consolidation  16090 non-null  uint8  
 11  purpose_educational         16090 non-null  uint8  
 12  purpose_home_improvement    16090 non-null  uint8  
 13  purpose_major_purchase      16090 non-null  uint8  
 14  purpose_small_business      16090 non-null  uint8  
dtypes: float64(3), int64(6), uint8(6)
memory usage: 1.2 MB
```

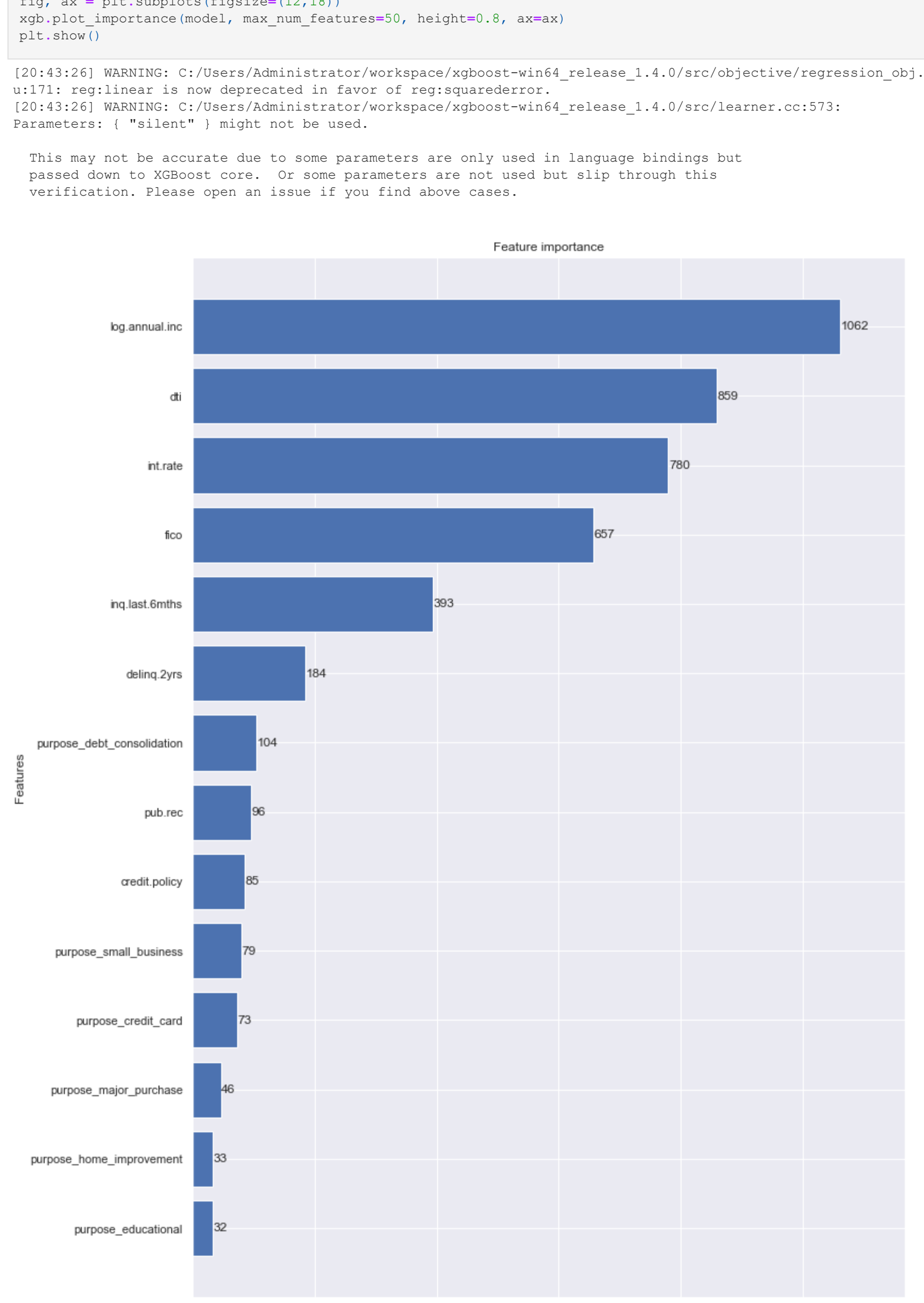
Important Variables:

Now let us run and xgboost model to get the important variables.

In [208…

```python
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
import xgboost as xgb
color = sns.color_palette()

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

def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)

xgb_params = {
    'eta': 0.05,
    'max_depth': 6,
    'subsample': 0.7,
    'colsample_bytree': 0.7,
    'objective': 'reg:linear',
    'silent': 1
}
dtrain = xgb.DMatrix(X, y, feature_names=X.columns.values)
model = xgb.train(dict(xgb_params, silent=0), dtrain, num_boost_round=100, feval=xgb_r2_score, maximize=True)

# plot the important features #
fig, ax = plt.subplots(figsize=(12,18))
xgb.plot_importance(model, max_num_features=50, height=0.8, ax=ax)
plt.show()
```

```
[20:43:26] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/objective/regression_obj.c
u:171: reg:linear is now deprecated in favor of reg:squarederror.
[20:43:26] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:573:
Parameters: { "silent" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.
```

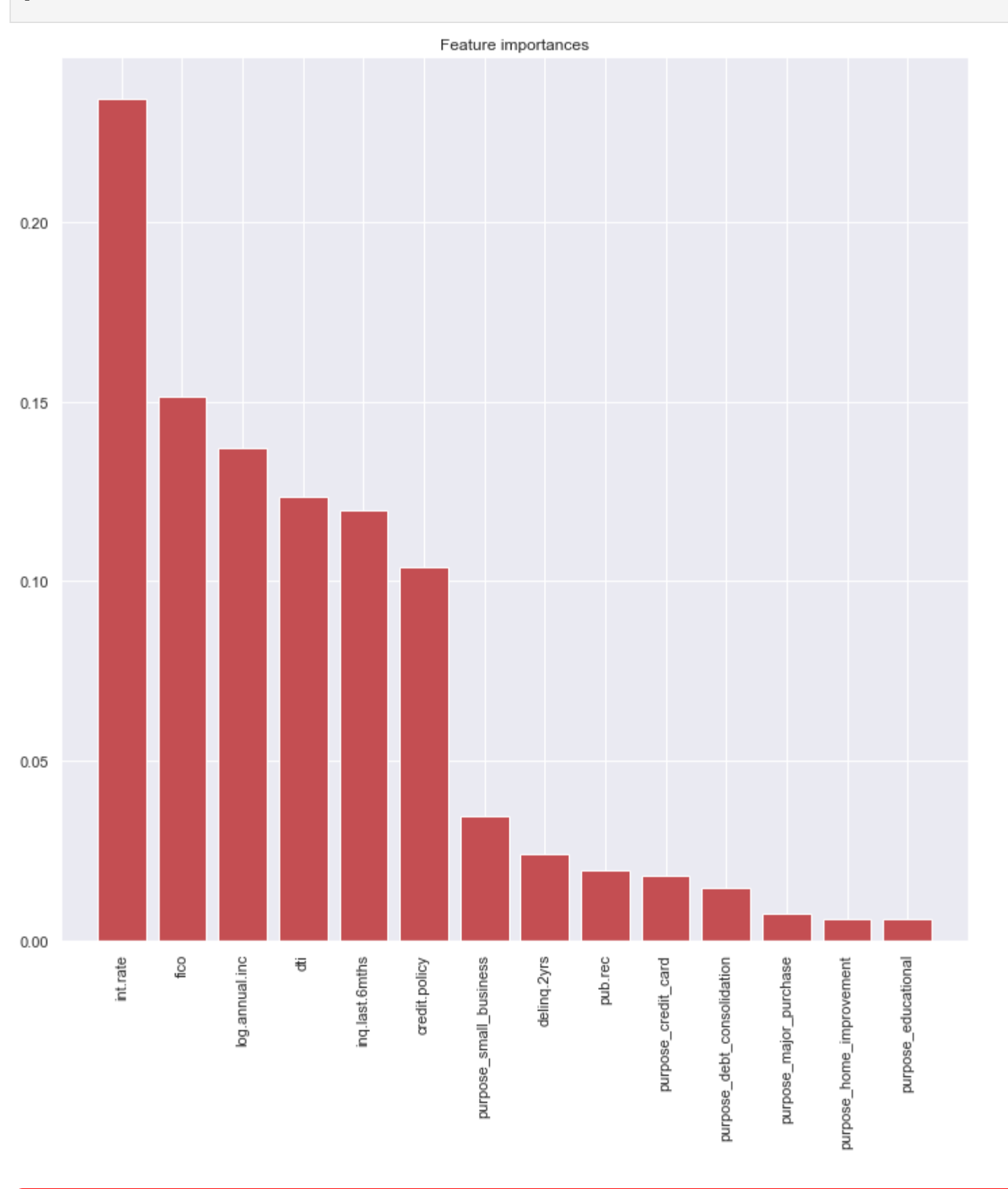


</h1></center>

Let us also build a Random Forest model and check the important variables.

In [209…

```python
from sklearn import ensemble
model = ensemble.RandomForestRegressor(n_estimators=200, max_depth=10, min_samples_leaf=4, max_features=0.2, n_
model.fit(X, y)
feat_names = X.columns.values

## plot the importances ##
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0)
indices = np.argsort(importances)[::-1][:20]

plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(len(indices)), importances[indices], color="r", align="center")
plt.xticks(range(len(indices)), feat_names[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
```



Deep Learning Model Implementation

Do the train test split and fit the model with the data shape we created above. since there are now 14 features, the first layer of the neural network is created with 14 nodes.

In [110…

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
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 sklearn.metrics import confusion_matrix, classification_report
```

In [210…

```python
to_train = final_data_fe[final_data_fe['not.fully.paid'].isin([0,1])]

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_state = 101)

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

model = Sequential()

model.add(
        Dense(14, 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']
)

early_stop = EarlyStopping(
        monitor='val_loss',
        mode='min',
        verbose=1,
        patience=25
)

model.fit(
        X_train,
        y_train,
        epochs=200,
        batch_size=256,
        validation_data=(X_test, y_test),
         callbacks=[early_stop]
)
```

```
Epoch 1/200
44/44 [==============================] - 1s 9ms/step - loss: 0.6902 - accuracy: 0.5420 - val_loss: 0.6857 - val
_accuracy: 0.5823
Epoch 2/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6804 - accuracy: 0.5987 - val_loss: 0.6747 - val
_accuracy: 0.6004
Epoch 3/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6684 - accuracy: 0.6065 - val_loss: 0.6651 - val
_accuracy: 0.6012
Epoch 4/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6593 - accuracy: 0.6109 - val_loss: 0.6591 - val
_accuracy: 0.6035
Epoch 5/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6547 - accuracy: 0.6150 - val_loss: 0.6565 - val
_accuracy: 0.6058
Epoch 6/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6515 - accuracy: 0.6181 - val_loss: 0.6555 - val
_accuracy: 0.6111
Epoch 7/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6497 - accuracy: 0.6202 - val_loss: 0.6549 - val
_accuracy: 0.6099
Epoch 8/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6477 - accuracy: 0.6188 - val_loss: 0.6532 - val
_accuracy: 0.6062
Epoch 9/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6461 - accuracy: 0.6212 - val_loss: 0.6526 - val
_accuracy: 0.6124
Epoch 10/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6450 - accuracy: 0.6240 - val_loss: 0.6519 - val
_accuracy: 0.6114
Epoch 11/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6439 - accuracy: 0.6242 - val_loss: 0.6524 - val
_accuracy: 0.6178
Epoch 12/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6436 - accuracy: 0.6229 - val_loss: 0.6508 - val
_accuracy: 0.6085
Epoch 13/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6426 - accuracy: 0.6253 - val_loss: 0.6510 - val
_accuracy: 0.6167
Epoch 14/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6422 - accuracy: 0.6258 - val_loss: 0.6507 - val
_accuracy: 0.6157
Epoch 15/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6419 - accuracy: 0.6281 - val_loss: 0.6500 - val
_accuracy: 0.6145
Epoch 16/200
44/44 [==============================] - 0s 5ms/step - loss: 0.6415 - accuracy: 0.6247 - val_loss: 0.6502 - val
_accuracy: 0.6126
Epoch 17/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6411 - accuracy: 0.6257 - val_loss: 0.6499 - val
_accuracy: 0.6138
Epoch 18/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6400 - accuracy: 0.6302 - val_loss: 0.6494 - val
_accuracy: 0.6132
Epoch 19/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6398 - accuracy: 0.6298 - val_loss: 0.6495 - val
_accuracy: 0.6114
Epoch 20/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6394 - accuracy: 0.6282 - val_loss: 0.6496 - val
_accuracy: 0.6114
Epoch 21/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6392 - accuracy: 0.6291 - val_loss: 0.6495 - val
_accuracy: 0.6107
Epoch 22/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6388 - accuracy: 0.6294 - val_loss: 0.6491 - val
_accuracy: 0.6107
Epoch 23/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6386 - accuracy: 0.6297 - val_loss: 0.6499 - val
_accuracy: 0.6165
Epoch 24/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6387 - accuracy: 0.6263 - val_loss: 0.6494 - val
_accuracy: 0.6149
Epoch 25/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6385 - accuracy: 0.6302 - val_loss: 0.6487 - val
_accuracy: 0.6116
Epoch 26/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6386 - accuracy: 0.6271 - val_loss: 0.6485 - val
_accuracy: 0.6134
Epoch 27/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6378 - accuracy: 0.6308 - val_loss: 0.6484 - val
_accuracy: 0.6134
Epoch 28/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6378 - accuracy: 0.6304 - val_loss: 0.6501 - val
_accuracy: 0.6159
Epoch 29/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6384 - accuracy: 0.6256 - val_loss: 0.6484 - val
_accuracy: 0.6116
Epoch 30/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6370 - accuracy: 0.6294 - val_loss: 0.6494 - val
_accuracy: 0.6107
Epoch 31/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6372 - accuracy: 0.6297 - val_loss: 0.6478 - val
_accuracy: 0.6130
Epoch 32/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6368 - accuracy: 0.6280 - val_loss: 0.6480 - val
_accuracy: 0.6147
Epoch 33/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6369 - accuracy: 0.6312 - val_loss: 0.6486 - val
_accuracy: 0.6201
Epoch 34/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6370 - accuracy: 0.6306 - val_loss: 0.6477 - val
_accuracy: 0.6153
Epoch 35/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6375 - accuracy: 0.6276 - val_loss: 0.6479 - val
_accuracy: 0.6147
Epoch 36/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6369 - accuracy: 0.6311 - val_loss: 0.6474 - val
_accuracy: 0.6136
Epoch 37/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6359 - accuracy: 0.6330 - val_loss: 0.6488 - val
_accuracy: 0.6174
Epoch 38/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6358 - accuracy: 0.6303 - val_loss: 0.6475 - val
_accuracy: 0.6172
Epoch 39/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6357 - accuracy: 0.6314 - val_loss: 0.6477 - val
_accuracy: 0.6188
Epoch 40/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6355 - accuracy: 0.6302 - val_loss: 0.6473 - val
_accuracy: 0.6163
Epoch 41/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6352 - accuracy: 0.6311 - val_loss: 0.6471 - val
_accuracy: 0.6209
Epoch 42/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6347 - accuracy: 0.6295 - val_loss: 0.6473 - val
_accuracy: 0.6203
Epoch 43/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6350 - accuracy: 0.6306 - val_loss: 0.6495 - val
_accuracy: 0.6163
Epoch 44/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6346 - accuracy: 0.6311 - val_loss: 0.6468 - val
_accuracy: 0.6184
Epoch 45/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6348 - accuracy: 0.6289 - val_loss: 0.6469 - val
_accuracy: 0.6118
Epoch 46/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6350 - accuracy: 0.6332 - val_loss: 0.6469 - val
_accuracy: 0.6188
Epoch 47/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6344 - accuracy: 0.6328 - val_loss: 0.6471 - val
_accuracy: 0.6232
Epoch 48/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6350 - accuracy: 0.6309 - val_loss: 0.6471 - val
_accuracy: 0.6248
Epoch 49/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6342 - accuracy: 0.6332 - val_loss: 0.6467 - val
_accuracy: 0.6180
Epoch 50/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6344 - accuracy: 0.6316 - val_loss: 0.6484 - val
_accuracy: 0.6201
Epoch 51/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6340 - accuracy: 0.6322 - val_loss: 0.6469 - val
_accuracy: 0.6242
Epoch 52/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6339 - accuracy: 0.6337 - val_loss: 0.6461 - val
_accuracy: 0.6205
Epoch 53/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6333 - accuracy: 0.6336 - val_loss: 0.6463 - val
_accuracy: 0.6157
Epoch 54/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6334 - accuracy: 0.6331 - val_loss: 0.6462 - val
_accuracy: 0.6132
Epoch 55/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6334 - accuracy: 0.6334 - val_loss: 0.6466 - val
_accuracy: 0.6219
Epoch 56/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6326 - accuracy: 0.6346 - val_loss: 0.6463 - val
_accuracy: 0.6207
Epoch 57/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6322 - accuracy: 0.6361 - val_loss: 0.6465 - val
_accuracy: 0.6132
Epoch 58/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6330 - accuracy: 0.6341 - val_loss: 0.6461 - val
_accuracy: 0.6203
Epoch 59/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6324 - accuracy: 0.6374 - val_loss: 0.6459 - val
_accuracy: 0.6167
Epoch 60/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6322 - accuracy: 0.6331 - val_loss: 0.6461 - val
_accuracy: 0.6180
Epoch 61/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6326 - accuracy: 0.6373 - val_loss: 0.6465 - val
_accuracy: 0.6203
Epoch 62/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6325 - accuracy: 0.6350 - val_loss: 0.6459 - val
_accuracy: 0.6238
Epoch 63/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6326 - accuracy: 0.6337 - val_loss: 0.6456 - val
_accuracy: 0.6161
Epoch 64/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6317 - accuracy: 0.6366 - val_loss: 0.6458 - val
_accuracy: 0.6217
Epoch 65/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6319 - accuracy: 0.6362 - val_loss: 0.6458 - val
_accuracy: 0.6221
Epoch 66/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6313 - accuracy: 0.6362 - val_loss: 0.6456 - val
_accuracy: 0.6103
Epoch 67/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6311 - accuracy: 0.6391 - val_loss: 0.6467 - val
_accuracy: 0.6246
Epoch 68/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6313 - accuracy: 0.6344 - val_loss: 0.6460 - val
_accuracy: 0.6283
Epoch 69/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6311 - accuracy: 0.6355 - val_loss: 0.6463 - val
_accuracy: 0.6296
Epoch 70/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6317 - accuracy: 0.6384 - val_loss: 0.6455 - val
_accuracy: 0.6201
Epoch 71/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6309 - accuracy: 0.6370 - val_loss: 0.6452 - val
_accuracy: 0.6261
Epoch 72/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6308 - accuracy: 0.6386 - val_loss: 0.6466 - val
_accuracy: 0.6252
Epoch 73/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6307 - accuracy: 0.6346 - val_loss: 0.6456 - val
_accuracy: 0.6263
Epoch 74/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6308 - accuracy: 0.6373 - val_loss: 0.6450 - val
_accuracy: 0.6169
Epoch 75/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6308 - accuracy: 0.6356 - val_loss: 0.6458 - val
_accuracy: 0.6265
Epoch 76/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6298 - accuracy: 0.6364 - val_loss: 0.6456 - val
_accuracy: 0.6306
Epoch 77/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6300 - accuracy: 0.6396 - val_loss: 0.6458 - val
_accuracy: 0.6196
Epoch 78/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6301 - accuracy: 0.6366 - val_loss: 0.6450 - val
_accuracy: 0.6261
Epoch 79/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6302 - accuracy: 0.6400 - val_loss: 0.6455 - val
_accuracy: 0.6267
Epoch 80/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6294 - accuracy: 0.6405 - val_loss: 0.6446 - val
_accuracy: 0.6184
Epoch 81/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6294 - accuracy: 0.6381 - val_loss: 0.6450 - val
_accuracy: 0.6244
Epoch 82/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6295 - accuracy: 0.6398 - val_loss: 0.6449 - val
_accuracy: 0.6232
Epoch 83/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6290 - accuracy: 0.6403 - val_loss: 0.6447 - val
_accuracy: 0.6227
Epoch 84/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6290 - accuracy: 0.6381 - val_loss: 0.6450 - val
_accuracy: 0.6300
Epoch 85/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6288 - accuracy: 0.6391 - val_loss: 0.6442 - val
_accuracy: 0.6288
Epoch 86/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6293 - accuracy: 0.6382 - val_loss: 0.6454 - val
_accuracy: 0.6265
Epoch 87/200
44/44 [==============================] - 0s 5ms/step - loss: 0.6289 - accuracy: 0.6433 - val_loss: 0.6442 - val
_accuracy: 0.6288
Epoch 88/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6287 - accuracy: 0.6426 - val_loss: 0.6455 - val
_accuracy: 0.6273
Epoch 89/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6294 - accuracy: 0.6363 - val_loss: 0.6456 - val
_accuracy: 0.6256
Epoch 90/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6291 - accuracy: 0.6407 - val_loss: 0.6448 - val
_accuracy: 0.6267
Epoch 91/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6284 - accuracy: 0.6388 - val_loss: 0.6455 - val
_accuracy: 0.6259
Epoch 92/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6291 - accuracy: 0.6357 - val_loss: 0.6462 - val
_accuracy: 0.6248
Epoch 93/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6291 - accuracy: 0.6389 - val_loss: 0.6442 - val
_accuracy: 0.6248
Epoch 94/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6286 - accuracy: 0.6380 - val_loss: 0.6439 - val
_accuracy: 0.6269
Epoch 95/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6279 - accuracy: 0.6432 - val_loss: 0.6449 - val
_accuracy: 0.6238
Epoch 96/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6276 - accuracy: 0.6405 - val_loss: 0.6435 - val
_accuracy: 0.6244
Epoch 97/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6276 - accuracy: 0.6423 - val_loss: 0.6443 - val
_accuracy: 0.6213
Epoch 98/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6276 - accuracy: 0.6404 - val_loss: 0.6440 - val
_accuracy: 0.6265
Epoch 99/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6273 - accuracy: 0.6435 - val_loss: 0.6448 - val
_accuracy: 0.6205
Epoch 100/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6276 - accuracy: 0.6418 - val_loss: 0.6443 - val
_accuracy: 0.6244
Epoch 101/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6275 - accuracy: 0.6432 - val_loss: 0.6435 - val
_accuracy: 0.6269
Epoch 102/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6270 - accuracy: 0.6394 - val_loss: 0.6432 - val
_accuracy: 0.6310
Epoch 103/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6273 - accuracy: 0.6413 - val_loss: 0.6429 - val
_accuracy: 0.6263
Epoch 104/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6273 - accuracy: 0.6423 - val_loss: 0.6431 - val
_accuracy: 0.6327
Epoch 105/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6264 - accuracy: 0.6428 - val_loss: 0.6439 - val
_accuracy: 0.6227
Epoch 106/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6265 - accuracy: 0.6439 - val_loss: 0.6437 - val
_accuracy: 0.6283
Epoch 107/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6263 - accuracy: 0.6442 - val_loss: 0.6442 - val
_accuracy: 0.6306
Epoch 108/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6270 - accuracy: 0.6401 - val_loss: 0.6431 - val
_accuracy: 0.6263
Epoch 109/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6262 - accuracy: 0.6454 - val_loss: 0.6444 - val
_accuracy: 0.6215
Epoch 110/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6264 - accuracy: 0.6435 - val_loss: 0.6438 - val
_accuracy: 0.6285
Epoch 111/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6264 - accuracy: 0.6412 - val_loss: 0.6432 - val
_accuracy: 0.6232
Epoch 112/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6262 - accuracy: 0.6434 - val_loss: 0.6431 - val
_accuracy: 0.6298
Epoch 113/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6262 - accuracy: 0.6428 - val_loss: 0.6428 - val
_accuracy: 0.6292
Epoch 114/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6257 - accuracy: 0.6441 - val_loss: 0.6429 - val
_accuracy: 0.6275
Epoch 115/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6256 - accuracy: 0.6431 - val_loss: 0.6425 - val
_accuracy: 0.6256
Epoch 116/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6257 - accuracy: 0.6435 - val_loss: 0.6429 - val
_accuracy: 0.6290
Epoch 117/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6252 - accuracy: 0.6437 - val_loss: 0.6433 - val
_accuracy: 0.6232
Epoch 118/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6250 - accuracy: 0.6432 - val_loss: 0.6423 - val
_accuracy: 0.6283
Epoch 119/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6252 - accuracy: 0.6441 - val_loss: 0.6429 - val
_accuracy: 0.6292
Epoch 120/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6250 - accuracy: 0.6464 - val_loss: 0.6424 - val
_accuracy: 0.6248
Epoch 121/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6252 - accuracy: 0.6432 - val_loss: 0.6433 - val
_accuracy: 0.6269
Epoch 122/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6249 - accuracy: 0.6429 - val_loss: 0.6430 - val
_accuracy: 0.6279
Epoch 123/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6249 - accuracy: 0.6435 - val_loss: 0.6425 - val
_accuracy: 0.6261
Epoch 124/200
44/44 [==============================] - 0s 6ms/step - loss: 0.6247 - accuracy: 0.6457 - val_loss: 0.6428 - val
_accuracy: 0.6279
Epoch 125/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6254 - accuracy: 0.6441 - val_loss: 0.6432 - val
_accuracy: 0.6298
Epoch 126/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6244 - accuracy: 0.6436 - val_loss: 0.6420 - val
_accuracy: 0.6279
Epoch 127/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6247 - accuracy: 0.6425 - val_loss: 0.6423 - val
_accuracy: 0.6267
Epoch 128/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6244 - accuracy: 0.6457 - val_loss: 0.6425 - val
_accuracy: 0.6292
Epoch 129/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6245 - accuracy: 0.6452 - val_loss: 0.6451 - val
_accuracy: 0.6244
Epoch 130/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6242 - accuracy: 0.6446 - val_loss: 0.6421 - val
_accuracy: 0.6285
Epoch 131/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6242 - accuracy: 0.6454 - val_loss: 0.6415 - val
_accuracy: 0.6277
Epoch 132/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6239 - accuracy: 0.6462 - val_loss: 0.6479 - val
_accuracy: 0.6163
Epoch 133/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6249 - accuracy: 0.6420 - val_loss: 0.6420 - val
_accuracy: 0.6281
Epoch 134/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6235 - accuracy: 0.6462 - val_loss: 0.6419 - val
_accuracy: 0.6314
Epoch 135/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6237 - accuracy: 0.6432 - val_loss: 0.6419 - val
_accuracy: 0.6259
Epoch 136/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6234 - accuracy: 0.6447 - val_loss: 0.6417 - val
_accuracy: 0.6265
Epoch 137/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6233 - accuracy: 0.6454 - val_loss: 0.6418 - val
_accuracy: 0.6279
Epoch 138/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6235 - accuracy: 0.6459 - val_loss: 0.6426 - val
_accuracy: 0.6294
Epoch 139/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6232 - accuracy: 0.6443 - val_loss: 0.6422 - val
_accuracy: 0.6296
Epoch 140/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6236 - accuracy: 0.6437 - val_loss: 0.6426 - val
_accuracy: 0.6256
Epoch 141/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6236 - accuracy: 0.6418 - val_loss: 0.6426 - val
_accuracy: 0.6248
Epoch 142/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6235 - accuracy: 0.6440 - val_loss: 0.6414 - val
_accuracy: 0.6254
Epoch 143/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6235 - accuracy: 0.6449 - val_loss: 0.6416 - val
_accuracy: 0.6283
Epoch 144/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6237 - accuracy: 0.6434 - val_loss: 0.6413 - val
_accuracy: 0.6294
Epoch 145/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6231 - accuracy: 0.6446 - val_loss: 0.6411 - val
_accuracy: 0.6306
Epoch 146/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6226 - accuracy: 0.6465 - val_loss: 0.6420 - val
_accuracy: 0.6273
Epoch 147/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6227 - accuracy: 0.6461 - val_loss: 0.6430 - val
_accuracy: 0.6223
Epoch 148/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6230 - accuracy: 0.6432 - val_loss: 0.6417 - val
_accuracy: 0.6230
Epoch 149/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6231 - accuracy: 0.6445 - val_loss: 0.6430 - val
_accuracy: 0.6219
Epoch 150/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6233 - accuracy: 0.6462 - val_loss: 0.6414 - val
_accuracy: 0.6273
Epoch 151/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6227 - accuracy: 0.6424 - val_loss: 0.6417 - val
_accuracy: 0.6254
Epoch 152/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6225 - accuracy: 0.6441 - val_loss: 0.6423 - val
_accuracy: 0.6252
Epoch 153/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6222 - accuracy: 0.6443 - val_loss: 0.6412 - val
_accuracy: 0.6244
Epoch 154/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6220 - accuracy: 0.6456 - val_loss: 0.6416 - val
_accuracy: 0.6240
Epoch 155/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6227 - accuracy: 0.6439 - val_loss: 0.6410 - val
_accuracy: 0.6275
Epoch 156/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6222 - accuracy: 0.6444 - val_loss: 0.6415 - val
_accuracy: 0.6244
Epoch 157/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6218 - accuracy: 0.6467 - val_loss: 0.6409 - val
_accuracy: 0.6271
Epoch 158/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6219 - accuracy: 0.6462 - val_loss: 0.6427 - val
_accuracy: 0.6259
Epoch 159/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6229 - accuracy: 0.6454 - val_loss: 0.6410 - val
_accuracy: 0.6248
Epoch 160/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6216 - accuracy: 0.6458 - val_loss: 0.6422 - val
_accuracy: 0.6254
Epoch 161/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6219 - accuracy: 0.6448 - val_loss: 0.6412 - val
_accuracy: 0.6300
Epoch 162/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6219 - accuracy: 0.6449 - val_loss: 0.6416 - val
_accuracy: 0.6306
Epoch 163/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6219 - accuracy: 0.6463 - val_loss: 0.6412 - val
_accuracy: 0.6256
Epoch 164/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6220 - accuracy: 0.6446 - val_loss: 0.6408 - val
_accuracy: 0.6238
Epoch 165/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6215 - accuracy: 0.6470 - val_loss: 0.6408 - val
_accuracy: 0.6281
Epoch 166/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6223 - accuracy: 0.6455 - val_loss: 0.6409 - val
_accuracy: 0.6230
Epoch 167/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6211 - accuracy: 0.6451 - val_loss: 0.6436 - val
_accuracy: 0.6238
Epoch 168/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6215 - accuracy: 0.6450 - val_loss: 0.6411 - val
_accuracy: 0.6254
Epoch 169/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6216 - accuracy: 0.6447 - val_loss: 0.6406 - val
_accuracy: 0.6267
Epoch 170/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6209 - accuracy: 0.6463 - val_loss: 0.6407 - val
_accuracy: 0.6240
Epoch 171/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6217 - accuracy: 0.6457 - val_loss: 0.6410 - val
_accuracy: 0.6244
Epoch 172/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6209 - accuracy: 0.6464 - val_loss: 0.6417 - val
_accuracy: 0.6256
Epoch 173/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6223 - accuracy: 0.6434 - val_loss: 0.6413 - val
_accuracy: 0.6259
Epoch 174/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6210 - accuracy: 0.6452 - val_loss: 0.6407 - val
_accuracy: 0.6298
Epoch 175/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6210 - accuracy: 0.6465 - val_loss: 0.6405 - val
_accuracy: 0.6279
Epoch 176/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6216 - accuracy: 0.6460 - val_loss: 0.6410 - val
_accuracy: 0.6236
Epoch 177/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6210 - accuracy: 0.6464 - val_loss: 0.6422 - val
_accuracy: 0.6256
Epoch 178/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6204 - accuracy: 0.6452 - val_loss: 0.6413 - val
_accuracy: 0.6252
Epoch 179/200
44/44 [==============================] - ETA: 0s - loss: 0.6200 - accuracy: 0.64 - 0s 3ms/step - loss: 0.6206 - 
accuracy: 0.6473 - val_loss: 0.6414 - val_accuracy: 0.6242
Epoch 180/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6204 - accuracy: 0.6467 - val_loss: 0.6417 - val
_accuracy: 0.6283
Epoch 181/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6202 - accuracy: 0.6473 - val_loss: 0.6402 - val
_accuracy: 0.6302
Epoch 182/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6205 - accuracy: 0.6450 - val_loss: 0.6398 - val
_accuracy: 0.6277
Epoch 183/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6203 - accuracy: 0.6440 - val_loss: 0.6403 - val
_accuracy: 0.6292
Epoch 184/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6206 - accuracy: 0.6438 - val_loss: 0.6396 - val
_accuracy: 0.6288
Epoch 185/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6200 - accuracy: 0.6502 - val_loss: 0.6417 - val
_accuracy: 0.6259
Epoch 186/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6209 - accuracy: 0.6449 - val_loss: 0.6399 - val
_accuracy: 0.6279
Epoch 187/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6195 - accuracy: 0.6450 - val_loss: 0.6399 - val
_accuracy: 0.6273
Epoch 188/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6197 - accuracy: 0.6471 - val_loss: 0.6406 - val
_accuracy: 0.6261
Epoch 189/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6201 - accuracy: 0.6473 - val_loss: 0.6402 - val
_accuracy: 0.6250
Epoch 190/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6197 - accuracy: 0.6473 - val_loss: 0.6405 - val
_accuracy: 0.6250
Epoch 191/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6197 - accuracy: 0.6459 - val_loss: 0.6410 - val
_accuracy: 0.6259
Epoch 192/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6192 - accuracy: 0.6465 - val_loss: 0.6400 - val
_accuracy: 0.6277
Epoch 193/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6194 - accuracy: 0.6479 - val_loss: 0.6411 - val
_accuracy: 0.6219
Epoch 194/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6192 - accuracy: 0.6462 - val_loss: 0.6412 - val
_accuracy: 0.6273
Epoch 195/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6195 - accuracy: 0.6463 - val_loss: 0.6397 - val
_accuracy: 0.6292
Epoch 196/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6196 - accuracy: 0.6497 - val_loss: 0.6404 - val
_accuracy: 0.6254
Epoch 197/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6189 - accuracy: 0.6481 - val_loss: 0.6398 - val
_accuracy: 0.6288
Epoch 198/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6189 - accuracy: 0.6469 - val_loss: 0.6395 - val
_accuracy: 0.6308
Epoch 199/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6188 - accuracy: 0.6476 - val_loss: 0.6397 - val
_accuracy: 0.6312
Epoch 200/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6185 - accuracy: 0.6473 - val_loss: 0.6395 - val
_accuracy: 0.6277
```

Out[210…

```
<keras.callbacks.History at 0x1fff0ae2d90>
```

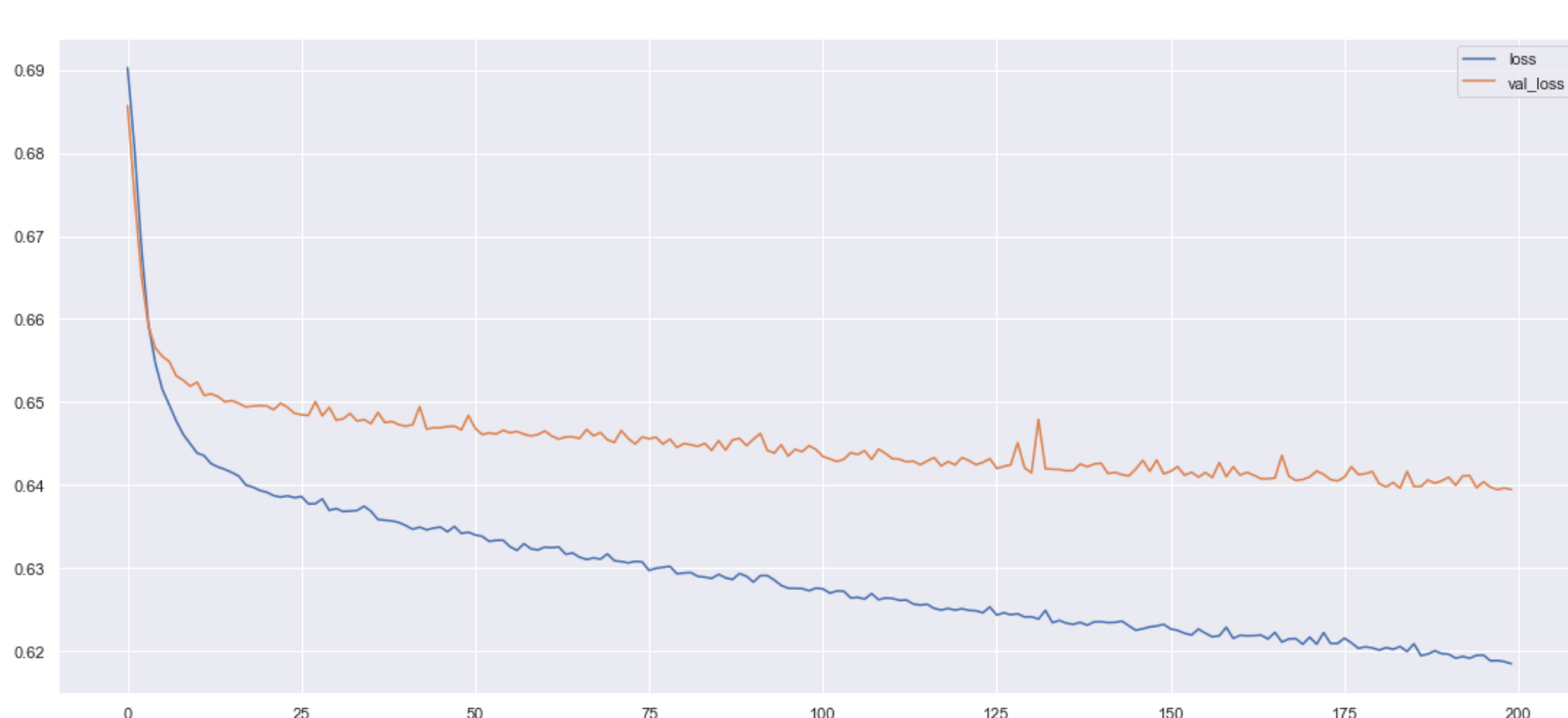
Model Evaluation and Validation

In [211…

```python
pd.DataFrame(model.history.history)[['loss','val_loss']].plot()
```

Out[211…

```
<AxesSubplot:>
```



This validation result, the Loss plot, shows us the model is overfitted.

In [212…

```python
y_pred = model.predict(X_test)
threshold=0.5
y_pred=np.where(y_pred > threshold, 1,0)
y_pred
```

Out[212…

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [1],
       [1]])
```

In [213…

```python
print(
        confusion_matrix(y_test,y_pred),
        '\n',
        classification_report(y_test,y_pred)
)
```

```
[[1591  846]
 [ 951 1439]]
              precision    recall  f1-score   support

           0       0.63      0.65      0.64      2437
           1       0.63      0.60      0.62      2390

    accuracy                           0.63      4827
   macro avg       0.63      0.63      0.63      4827
weighted avg       0.63      0.63      0.63      4827
```

The model's overall f1-score for accuracy is 0.63. Still, there are type 2 errors (951) in the prediction.

Model Refinement

Two ways of refining the model we will try here.

Add Dropout layers to bring down the overfitting OR Lower the cut-off line in binary prediction to reduce the Type 2 error, at the cost of increasing Type 1 error. In the LendingClub case, Type 2 error is the more serious problem because it devastates its balance sheet, while Type 1 error is not a very big deal.

In [215…

```python
model_new = Sequential()

model_new.add(
        Dense(14, 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_accuracy']
)


model_new.fit(
        X_train,
        y_train,
        epochs=200,
        batch_size=256,
        validation_data=(X_test, y_test),
         callbacks=[early_stop]
)
```

```
Epoch 1/200
44/44 [==============================] - 1s 9ms/step - loss: 0.6930 - binary_accuracy: 0.5075 - val_loss: 0.690
9 - val_binary_accuracy: 0.5797
Epoch 2/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6904 - binary_accuracy: 0.5340 - val_loss: 0.686
7 - val_binary_accuracy: 0.6029
Epoch 3/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6846 - binary_accuracy: 0.5577 - val_loss: 0.678
2 - val_binary_accuracy: 0.5993
Epoch 4/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6803 - binary_accuracy: 0.5605 - val_loss: 0.671
4 - val_binary_accuracy: 0.6024
Epoch 5/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6749 - binary_accuracy: 0.5760 - val_loss: 0.666
8 - val_binary_accuracy: 0.6068
Epoch 6/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6724 - binary_accuracy: 0.5792 - val_loss: 0.664
2 - val_binary_accuracy: 0.6031
Epoch 7/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6699 - binary_accuracy: 0.5871 - val_loss: 0.662
3 - val_binary_accuracy: 0.6095
Epoch 8/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6701 - binary_accuracy: 0.5880 - val_loss: 0.661
5 - val_binary_accuracy: 0.6051
Epoch 9/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6686 - binary_accuracy: 0.5926 - val_loss: 0.660
4 - val_binary_accuracy: 0.6091
Epoch 10/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6653 - binary_accuracy: 0.5985 - val_loss: 0.658
1 - val_binary_accuracy: 0.6124
Epoch 11/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6621 - binary_accuracy: 0.5982 - val_loss: 0.657
1 - val_binary_accuracy: 0.6103
Epoch 12/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6610 - binary_accuracy: 0.6011 - val_loss: 0.656
3 - val_binary_accuracy: 0.6124
Epoch 13/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6611 - binary_accuracy: 0.6052 - val_loss: 0.655
9 - val_binary_accuracy: 0.6099
Epoch 14/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6601 - binary_accuracy: 0.6054 - val_loss: 0.655
4 - val_binary_accuracy: 0.6043
Epoch 15/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6604 - binary_accuracy: 0.6024 - val_loss: 0.654
5 - val_binary_accuracy: 0.6085
Epoch 16/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6579 - binary_accuracy: 0.6082 - val_loss: 0.654
4 - val_binary_accuracy: 0.6080
Epoch 17/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6588 - binary_accuracy: 0.6089 - val_loss: 0.654
2 - val_binary_accuracy: 0.6078
Epoch 18/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6567 - binary_accuracy: 0.6075 - val_loss: 0.653
4 - val_binary_accuracy: 0.6085
Epoch 19/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6563 - binary_accuracy: 0.6087 - val_loss: 0.652
5 - val_binary_accuracy: 0.6087
Epoch 20/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6577 - binary_accuracy: 0.6092 - val_loss: 0.652
6 - val_binary_accuracy: 0.6099
Epoch 21/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6554 - binary_accuracy: 0.6089 - val_loss: 0.652
6 - val_binary_accuracy: 0.6095
Epoch 22/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6582 - binary_accuracy: 0.6097 - val_loss: 0.652
0 - val_binary_accuracy: 0.6145
Epoch 23/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6553 - binary_accuracy: 0.6100 - val_loss: 0.651
6 - val_binary_accuracy: 0.6099
Epoch 24/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6517 - binary_accuracy: 0.6130 - val_loss: 0.651
1 - val_binary_accuracy: 0.6114
Epoch 25/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6536 - binary_accuracy: 0.6123 - val_loss: 0.651
2 - val_binary_accuracy: 0.6085
Epoch 26/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6531 - binary_accuracy: 0.6141 - val_loss: 0.650
7 - val_binary_accuracy: 0.6145
Epoch 27/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6529 - binary_accuracy: 0.6191 - val_loss: 0.651
2 - val_binary_accuracy: 0.6095
Epoch 28/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6537 - binary_accuracy: 0.6172 - val_loss: 0.650
1 - val_binary_accuracy: 0.6118
Epoch 29/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6528 - binary_accuracy: 0.6146 - val_loss: 0.650
5 - val_binary_accuracy: 0.6062
Epoch 30/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6524 - binary_accuracy: 0.6124 - val_loss: 0.650
1 - val_binary_accuracy: 0.6064
Epoch 31/200
44/44 [==============================] - ETA: 0s - loss: 0.6511 - binary_accuracy: 0.622 - 0s 2ms/step - loss:
0.6523 - binary_accuracy: 0.6193 - val_loss: 0.6496 - val_binary_accuracy: 0.6085
Epoch 32/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6519 - binary_accuracy: 0.6186 - val_loss: 0.650
0 - val_binary_accuracy: 0.6095
Epoch 33/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6512 - binary_accuracy: 0.6202 - val_loss: 0.649
5 - val_binary_accuracy: 0.6114
Epoch 34/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6512 - binary_accuracy: 0.6189 - val_loss: 0.649
2 - val_binary_accuracy: 0.6114
Epoch 35/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6479 - binary_accuracy: 0.6230 - val_loss: 0.648
8 - val_binary_accuracy: 0.6091
Epoch 36/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6529 - binary_accuracy: 0.6108 - val_loss: 0.649
8 - val_binary_accuracy: 0.6097
Epoch 37/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6502 - binary_accuracy: 0.6174 - val_loss: 0.649
3 - val_binary_accuracy: 0.6114
Epoch 38/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6491 - binary_accuracy: 0.6215 - val_loss: 0.649
1 - val_binary_accuracy: 0.6053
Epoch 39/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6509 - binary_accuracy: 0.6205 - val_loss: 0.648
8 - val_binary_accuracy: 0.6095
Epoch 40/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6505 - binary_accuracy: 0.6217 - val_loss: 0.648
7 - val_binary_accuracy: 0.6095
Epoch 41/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6517 - binary_accuracy: 0.6133 - val_loss: 0.648
6 - val_binary_accuracy: 0.6099
Epoch 42/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6495 - binary_accuracy: 0.6203 - val_loss: 0.647
7 - val_binary_accuracy: 0.6116
Epoch 43/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6514 - binary_accuracy: 0.6195 - val_loss: 0.647
8 - val_binary_accuracy: 0.6128
Epoch 44/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6480 - binary_accuracy: 0.6241 - val_loss: 0.647
6 - val_binary_accuracy: 0.6140
Epoch 45/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6498 - binary_accuracy: 0.6215 - val_loss: 0.647
3 - val_binary_accuracy: 0.6124
Epoch 46/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6503 - binary_accuracy: 0.6205 - val_loss: 0.647
5 - val_binary_accuracy: 0.6072
Epoch 47/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6494 - binary_accuracy: 0.6200 - val_loss: 0.647
4 - val_binary_accuracy: 0.6103
Epoch 48/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6493 - binary_accuracy: 0.6185 - val_loss: 0.647
4 - val_binary_accuracy: 0.6103
Epoch 49/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6479 - binary_accuracy: 0.6228 - val_loss: 0.646
8 - val_binary_accuracy: 0.6132
Epoch 50/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6481 - binary_accuracy: 0.6224 - val_loss: 0.646
8 - val_binary_accuracy: 0.6157
Epoch 51/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6494 - binary_accuracy: 0.6205 - val_loss: 0.646
6 - val_binary_accuracy: 0.6082
Epoch 52/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6486 - binary_accuracy: 0.6191 - val_loss: 0.646
8 - val_binary_accuracy: 0.6116
Epoch 53/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6503 - binary_accuracy: 0.6176 - val_loss: 0.646
6 - val_binary_accuracy: 0.6109
Epoch 54/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6488 - binary_accuracy: 0.6231 - val_loss: 0.647
1 - val_binary_accuracy: 0.6186
Epoch 55/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6486 - binary_accuracy: 0.6193 - val_loss: 0.646
6 - val_binary_accuracy: 0.6132
Epoch 56/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6487 - binary_accuracy: 0.6203 - val_loss: 0.646
3 - val_binary_accuracy: 0.6172
Epoch 57/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6480 - binary_accuracy: 0.6177 - val_loss: 0.646
4 - val_binary_accuracy: 0.6155
Epoch 58/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6493 - binary_accuracy: 0.6202 - val_loss: 0.646
7 - val_binary_accuracy: 0.6143
Epoch 59/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6482 - binary_accuracy: 0.6221 - val_loss: 0.646
3 - val_binary_accuracy: 0.6145
Epoch 60/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6493 - binary_accuracy: 0.6185 - val_loss: 0.646
5 - val_binary_accuracy: 0.6203
Epoch 61/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6473 - binary_accuracy: 0.6245 - val_loss: 0.646
1 - val_binary_accuracy: 0.6198
Epoch 62/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6454 - binary_accuracy: 0.6240 - val_loss: 0.645
8 - val_binary_accuracy: 0.6151
Epoch 63/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6472 - binary_accuracy: 0.6211 - val_loss: 0.645
8 - val_binary_accuracy: 0.6186
Epoch 64/200
44/44 [==============================] - 0s 8ms/step - loss: 0.6446 - binary_accuracy: 0.6241 - val_loss: 0.645
7 - val_binary_accuracy: 0.6153
Epoch 65/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6466 - binary_accuracy: 0.6234 - val_loss: 0.645
6 - val_binary_accuracy: 0.6176
Epoch 66/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6451 - binary_accuracy: 0.6253 - val_loss: 0.645
5 - val_binary_accuracy: 0.6176
Epoch 67/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6478 - binary_accuracy: 0.6197 - val_loss: 0.645
6 - val_binary_accuracy: 0.6184
Epoch 68/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6454 - binary_accuracy: 0.6251 - val_loss: 0.645
5 - val_binary_accuracy: 0.6182
Epoch 69/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6465 - binary_accuracy: 0.6252 - val_loss: 0.645
3 - val_binary_accuracy: 0.6159
Epoch 70/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6449 - binary_accuracy: 0.6217 - val_loss: 0.645
9 - val_binary_accuracy: 0.6209
Epoch 71/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6440 - binary_accuracy: 0.6266 - val_loss: 0.645
2 - val_binary_accuracy: 0.6176
Epoch 72/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6461 - binary_accuracy: 0.6290 - val_loss: 0.645
0 - val_binary_accuracy: 0.6209
Epoch 73/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6445 - binary_accuracy: 0.6282 - val_loss: 0.645
5 - val_binary_accuracy: 0.6172
Epoch 74/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6443 - binary_accuracy: 0.6249 - val_loss: 0.644
9 - val_binary_accuracy: 0.6194
Epoch 75/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6457 - binary_accuracy: 0.6241 - val_loss: 0.644
9 - val_binary_accuracy: 0.6238
Epoch 76/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6435 - binary_accuracy: 0.6289 - val_loss: 0.644
6 - val_binary_accuracy: 0.6169
Epoch 77/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6472 - binary_accuracy: 0.6261 - val_loss: 0.645
4 - val_binary_accuracy: 0.6230
Epoch 78/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6448 - binary_accuracy: 0.6239 - val_loss: 0.644
4 - val_binary_accuracy: 0.6167
Epoch 79/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6472 - binary_accuracy: 0.6240 - val_loss: 0.645
0 - val_binary_accuracy: 0.6211
Epoch 80/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6462 - binary_accuracy: 0.6249 - val_loss: 0.644
8 - val_binary_accuracy: 0.6217
Epoch 81/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6474 - binary_accuracy: 0.6226 - val_loss: 0.645
0 - val_binary_accuracy: 0.6201
Epoch 82/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6453 - binary_accuracy: 0.6235 - val_loss: 0.644
1 - val_binary_accuracy: 0.6190
Epoch 83/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6463 - binary_accuracy: 0.6230 - val_loss: 0.644
1 - val_binary_accuracy: 0.6190
Epoch 84/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6469 - binary_accuracy: 0.6279 - val_loss: 0.644
8 - val_binary_accuracy: 0.6194
Epoch 85/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6446 - binary_accuracy: 0.6235 - val_loss: 0.643
9 - val_binary_accuracy: 0.6182
Epoch 86/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6425 - binary_accuracy: 0.6296 - val_loss: 0.643
8 - val_binary_accuracy: 0.6205
Epoch 87/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6434 - binary_accuracy: 0.6307 - val_loss: 0.644
2 - val_binary_accuracy: 0.6196
Epoch 88/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6434 - binary_accuracy: 0.6315 - val_loss: 0.643
9 - val_binary_accuracy: 0.6215
Epoch 89/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6464 - binary_accuracy: 0.6242 - val_loss: 0.644
3 - val_binary_accuracy: 0.6213
Epoch 90/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6454 - binary_accuracy: 0.6282 - val_loss: 0.643
9 - val_binary_accuracy: 0.6230
Epoch 91/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6449 - binary_accuracy: 0.6252 - val_loss: 0.644
1 - val_binary_accuracy: 0.6215
Epoch 92/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6437 - binary_accuracy: 0.6197 - val_loss: 0.643
7 - val_binary_accuracy: 0.6221
Epoch 93/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6438 - binary_accuracy: 0.6220 - val_loss: 0.644
0 - val_binary_accuracy: 0.6198
Epoch 94/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6443 - binary_accuracy: 0.6293 - val_loss: 0.643
6 - val_binary_accuracy: 0.6205
Epoch 95/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6457 - binary_accuracy: 0.6222 - val_loss: 0.644
0 - val_binary_accuracy: 0.6213
Epoch 96/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6439 - binary_accuracy: 0.6291 - val_loss: 0.644
4 - val_binary_accuracy: 0.6213
Epoch 97/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6432 - binary_accuracy: 0.6273 - val_loss: 0.644
1 - val_binary_accuracy: 0.6178
Epoch 98/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6440 - binary_accuracy: 0.6264 - val_loss: 0.643
8 - val_binary_accuracy: 0.6261
Epoch 99/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6437 - binary_accuracy: 0.6315 - val_loss: 0.643
7 - val_binary_accuracy: 0.6236
Epoch 100/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6455 - binary_accuracy: 0.6254 - val_loss: 0.643
6 - val_binary_accuracy: 0.6263
Epoch 101/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6428 - binary_accuracy: 0.6335 - val_loss: 0.643
7 - val_binary_accuracy: 0.6213
Epoch 102/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6422 - binary_accuracy: 0.6323 - val_loss: 0.644
0 - val_binary_accuracy: 0.6236
Epoch 103/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6430 - binary_accuracy: 0.6275 - val_loss: 0.643
3 - val_binary_accuracy: 0.6263
Epoch 104/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6433 - binary_accuracy: 0.6278 - val_loss: 0.643
7 - val_binary_accuracy: 0.6254
Epoch 105/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6446 - binary_accuracy: 0.6259 - val_loss: 0.643
5 - val_binary_accuracy: 0.6248
Epoch 106/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6431 - binary_accuracy: 0.6319 - val_loss: 0.643
3 - val_binary_accuracy: 0.6236
Epoch 107/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6429 - binary_accuracy: 0.6301 - val_loss: 0.643
4 - val_binary_accuracy: 0.6219
Epoch 108/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6434 - binary_accuracy: 0.6292 - val_loss: 0.643
5 - val_binary_accuracy: 0.6246
Epoch 109/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6426 - binary_accuracy: 0.6291 - val_loss: 0.643
2 - val_binary_accuracy: 0.6230
Epoch 110/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6410 - binary_accuracy: 0.6320 - val_loss: 0.643
3 - val_binary_accuracy: 0.6238
Epoch 111/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6445 - binary_accuracy: 0.6290 - val_loss: 0.643
4 - val_binary_accuracy: 0.6215
Epoch 112/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6440 - binary_accuracy: 0.6263 - val_loss: 0.643
5 - val_binary_accuracy: 0.6252
Epoch 113/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6425 - binary_accuracy: 0.6335 - val_loss: 0.643
1 - val_binary_accuracy: 0.6269
Epoch 114/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6447 - binary_accuracy: 0.6300 - val_loss: 0.643
2 - val_binary_accuracy: 0.6223
Epoch 115/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6433 - binary_accuracy: 0.6282 - val_loss: 0.643
6 - val_binary_accuracy: 0.6236
Epoch 116/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6442 - binary_accuracy: 0.6292 - val_loss: 0.643
3 - val_binary_accuracy: 0.6232
Epoch 117/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6448 - binary_accuracy: 0.6318 - val_loss: 0.643
5 - val_binary_accuracy: 0.6225
Epoch 118/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6430 - binary_accuracy: 0.6329 - val_loss: 0.643
2 - val_binary_accuracy: 0.6234
Epoch 119/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6399 - binary_accuracy: 0.6339 - val_loss: 0.643
2 - val_binary_accuracy: 0.6227
Epoch 120/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6423 - binary_accuracy: 0.6299 - val_loss: 0.643
2 - val_binary_accuracy: 0.6273
Epoch 121/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6402 - binary_accuracy: 0.6320 - val_loss: 0.642
6 - val_binary_accuracy: 0.6256
Epoch 122/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6413 - binary_accuracy: 0.6294 - val_loss: 0.642
1 - val_binary_accuracy: 0.6271
Epoch 123/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6414 - binary_accuracy: 0.6370 - val_loss: 0.642
5 - val_binary_accuracy: 0.6273
Epoch 124/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6413 - binary_accuracy: 0.6325 - val_loss: 0.642
5 - val_binary_accuracy: 0.6232
Epoch 125/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6437 - binary_accuracy: 0.6326 - val_loss: 0.642
7 - val_binary_accuracy: 0.6263
Epoch 126/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6433 - binary_accuracy: 0.6359 - val_loss: 0.642
6 - val_binary_accuracy: 0.6265
Epoch 127/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6419 - binary_accuracy: 0.6347 - val_loss: 0.642
4 - val_binary_accuracy: 0.6219
Epoch 128/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6434 - binary_accuracy: 0.6322 - val_loss: 0.642
6 - val_binary_accuracy: 0.6219
Epoch 129/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6420 - binary_accuracy: 0.6305 - val_loss: 0.642
7 - val_binary_accuracy: 0.6238
Epoch 130/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6425 - binary_accuracy: 0.6299 - val_loss: 0.642
2 - val_binary_accuracy: 0.6256
Epoch 131/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6414 - binary_accuracy: 0.6335 - val_loss: 0.642
5 - val_binary_accuracy: 0.6242
Epoch 132/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6415 - binary_accuracy: 0.6384 - val_loss: 0.642
5 - val_binary_accuracy: 0.6227
Epoch 133/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6421 - binary_accuracy: 0.6291 - val_loss: 0.642
2 - val_binary_accuracy: 0.6238
Epoch 134/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6409 - binary_accuracy: 0.6322 - val_loss: 0.642
0 - val_binary_accuracy: 0.6219
Epoch 135/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6411 - binary_accuracy: 0.6316 - val_loss: 0.642
0 - val_binary_accuracy: 0.6277
Epoch 136/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6412 - binary_accuracy: 0.6303 - val_loss: 0.641
8 - val_binary_accuracy: 0.6261
Epoch 137/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6419 - binary_accuracy: 0.6306 - val_loss: 0.642
5 - val_binary_accuracy: 0.6271
Epoch 138/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6414 - binary_accuracy: 0.6324 - val_loss: 0.641
9 - val_binary_accuracy: 0.6275
Epoch 139/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6422 - binary_accuracy: 0.6282 - val_loss: 0.641
4 - val_binary_accuracy: 0.6248
Epoch 140/200
44/44 [==============================] - 0s 4ms/step - loss: 0.6399 - binary_accuracy: 0.6284 - val_loss: 0.641
4 - val_binary_accuracy: 0.6223
Epoch 141/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6386 - binary_accuracy: 0.6313 - val_loss: 0.641
6 - val_binary_accuracy: 0.6256
Epoch 142/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6409 - binary_accuracy: 0.6394 - val_loss: 0.642
1 - val_binary_accuracy: 0.6267
Epoch 143/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6423 - binary_accuracy: 0.6310 - val_loss: 0.641
4 - val_binary_accuracy: 0.6279
Epoch 144/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6408 - binary_accuracy: 0.6306 - val_loss: 0.641
3 - val_binary_accuracy: 0.6271
Epoch 145/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6436 - binary_accuracy: 0.6320 - val_loss: 0.642
0 - val_binary_accuracy: 0.6263
Epoch 146/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6417 - binary_accuracy: 0.6322 - val_loss: 0.641
8 - val_binary_accuracy: 0.6232
Epoch 147/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6386 - binary_accuracy: 0.6355 - val_loss: 0.641
4 - val_binary_accuracy: 0.6279
Epoch 148/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6393 - binary_accuracy: 0.6312 - val_loss: 0.641
6 - val_binary_accuracy: 0.6230
Epoch 149/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6395 - binary_accuracy: 0.6307 - val_loss: 0.641
4 - val_binary_accuracy: 0.6259
Epoch 150/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6404 - binary_accuracy: 0.6317 - val_loss: 0.641
1 - val_binary_accuracy: 0.6259
Epoch 151/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6401 - binary_accuracy: 0.6346 - val_loss: 0.641
4 - val_binary_accuracy: 0.6279
Epoch 152/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6394 - binary_accuracy: 0.6338 - val_loss: 0.641
3 - val_binary_accuracy: 0.6254
Epoch 153/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6421 - binary_accuracy: 0.6307 - val_loss: 0.641
1 - val_binary_accuracy: 0.6242
Epoch 154/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6384 - binary_accuracy: 0.6371 - val_loss: 0.641
1 - val_binary_accuracy: 0.6269
Epoch 155/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6413 - binary_accuracy: 0.6346 - val_loss: 0.641
2 - val_binary_accuracy: 0.6225
Epoch 156/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6398 - binary_accuracy: 0.6334 - val_loss: 0.641
4 - val_binary_accuracy: 0.6273
Epoch 157/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6399 - binary_accuracy: 0.6328 - val_loss: 0.641
5 - val_binary_accuracy: 0.6271
Epoch 158/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6382 - binary_accuracy: 0.6335 - val_loss: 0.641
2 - val_binary_accuracy: 0.6269
Epoch 159/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6380 - binary_accuracy: 0.6334 - val_loss: 0.641
3 - val_binary_accuracy: 0.6283
Epoch 160/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6415 - binary_accuracy: 0.6334 - val_loss: 0.641
4 - val_binary_accuracy: 0.6267
Epoch 161/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6402 - binary_accuracy: 0.6336 - val_loss: 0.641
3 - val_binary_accuracy: 0.6246
Epoch 162/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6418 - binary_accuracy: 0.6319 - val_loss: 0.641
4 - val_binary_accuracy: 0.6263
Epoch 163/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6403 - binary_accuracy: 0.6309 - val_loss: 0.641
6 - val_binary_accuracy: 0.6275
Epoch 164/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6400 - binary_accuracy: 0.6356 - val_loss: 0.641
5 - val_binary_accuracy: 0.6248
Epoch 165/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6420 - binary_accuracy: 0.6276 - val_loss: 0.641
0 - val_binary_accuracy: 0.6232
Epoch 166/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6411 - binary_accuracy: 0.6302 - val_loss: 0.641
4 - val_binary_accuracy: 0.6207
Epoch 167/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6407 - binary_accuracy: 0.6342 - val_loss: 0.641
1 - val_binary_accuracy: 0.6292
Epoch 168/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6389 - binary_accuracy: 0.6355 - val_loss: 0.641
1 - val_binary_accuracy: 0.6271
Epoch 169/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6393 - binary_accuracy: 0.6359 - val_loss: 0.641
1 - val_binary_accuracy: 0.6292
Epoch 170/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6383 - binary_accuracy: 0.6331 - val_loss: 0.640
4 - val_binary_accuracy: 0.6283
Epoch 171/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6398 - binary_accuracy: 0.6342 - val_loss: 0.641
1 - val_binary_accuracy: 0.6252
Epoch 172/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6403 - binary_accuracy: 0.6335 - val_loss: 0.640
5 - val_binary_accuracy: 0.6221
Epoch 173/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6399 - binary_accuracy: 0.6325 - val_loss: 0.641
0 - val_binary_accuracy: 0.6263
Epoch 174/200
44/44 [==============================] - 0s 3ms/step - loss: 0.6376 - binary_accuracy: 0.6362 - val_loss: 0.640
1 - val_binary_accuracy: 0.6277
Epoch 175/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6389 - binary_accuracy: 0.6376 - val_loss: 0.641
1 - val_binary_accuracy: 0.6290
Epoch 176/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6397 - binary_accuracy: 0.6343 - val_loss: 0.640
9 - val_binary_accuracy: 0.6285
Epoch 177/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6414 - binary_accuracy: 0.6333 - val_loss: 0.642
3 - val_binary_accuracy: 0.6263
Epoch 178/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6408 - binary_accuracy: 0.6316 - val_loss: 0.640
8 - val_binary_accuracy: 0.6263
Epoch 179/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6401 - binary_accuracy: 0.6348 - val_loss: 0.641
4 - val_binary_accuracy: 0.6269
Epoch 180/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6393 - binary_accuracy: 0.6339 - val_loss: 0.640
2 - val_binary_accuracy: 0.6281
Epoch 181/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6375 - binary_accuracy: 0.6344 - val_loss: 0.640
0 - val_binary_accuracy: 0.6271
Epoch 182/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6399 - binary_accuracy: 0.6308 - val_loss: 0.640
1 - val_binary_accuracy: 0.6290
Epoch 183/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6376 - binary_accuracy: 0.6331 - val_loss: 0.640
9 - val_binary_accuracy: 0.6283
Epoch 184/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6384 - binary_accuracy: 0.6328 - val_loss: 0.639
7 - val_binary_accuracy: 0.6275
Epoch 185/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6374 - binary_accuracy: 0.6309 - val_loss: 0.640
7 - val_binary_accuracy: 0.6296
Epoch 186/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6402 - binary_accuracy: 0.6316 - val_loss: 0.640
0 - val_binary_accuracy: 0.6254
Epoch 187/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6383 - binary_accuracy: 0.6346 - val_loss: 0.640
4 - val_binary_accuracy: 0.6269
Epoch 188/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6380 - binary_accuracy: 0.6344 - val_loss: 0.640
3 - val_binary_accuracy: 0.6300
Epoch 189/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6380 - binary_accuracy: 0.6324 - val_loss: 0.640
4 - val_binary_accuracy: 0.6317
Epoch 190/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6381 - binary_accuracy: 0.6359 - val_loss: 0.640
0 - val_binary_accuracy: 0.6283
Epoch 191/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6383 - binary_accuracy: 0.6333 - val_loss: 0.640
4 - val_binary_accuracy: 0.6275
Epoch 192/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6401 - binary_accuracy: 0.6321 - val_loss: 0.640
8 - val_binary_accuracy: 0.6252
Epoch 193/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6394 - binary_accuracy: 0.6362 - val_loss: 0.639
8 - val_binary_accuracy: 0.6288
Epoch 194/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6361 - binary_accuracy: 0.6415 - val_loss: 0.640
1 - val_binary_accuracy: 0.6261
Epoch 195/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6397 - binary_accuracy: 0.6338 - val_loss: 0.639
4 - val_binary_accuracy: 0.6277
Epoch 196/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6377 - binary_accuracy: 0.6389 - val_loss: 0.639
5 - val_binary_accuracy: 0.6312
Epoch 197/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6366 - binary_accuracy: 0.6372 - val_loss: 0.639
5 - val_binary_accuracy: 0.6331
Epoch 198/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6380 - binary_accuracy: 0.6351 - val_loss: 0.639
0 - val_binary_accuracy: 0.6275
Epoch 199/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6378 - binary_accuracy: 0.6417 - val_loss: 0.639
7 - val_binary_accuracy: 0.6294
Epoch 200/200
44/44 [==============================] - 0s 2ms/step - loss: 0.6398 - binary_accuracy: 0.6330 - val_loss: 0.639
1 - val_binary_accuracy: 0.6300
```

Out[215…

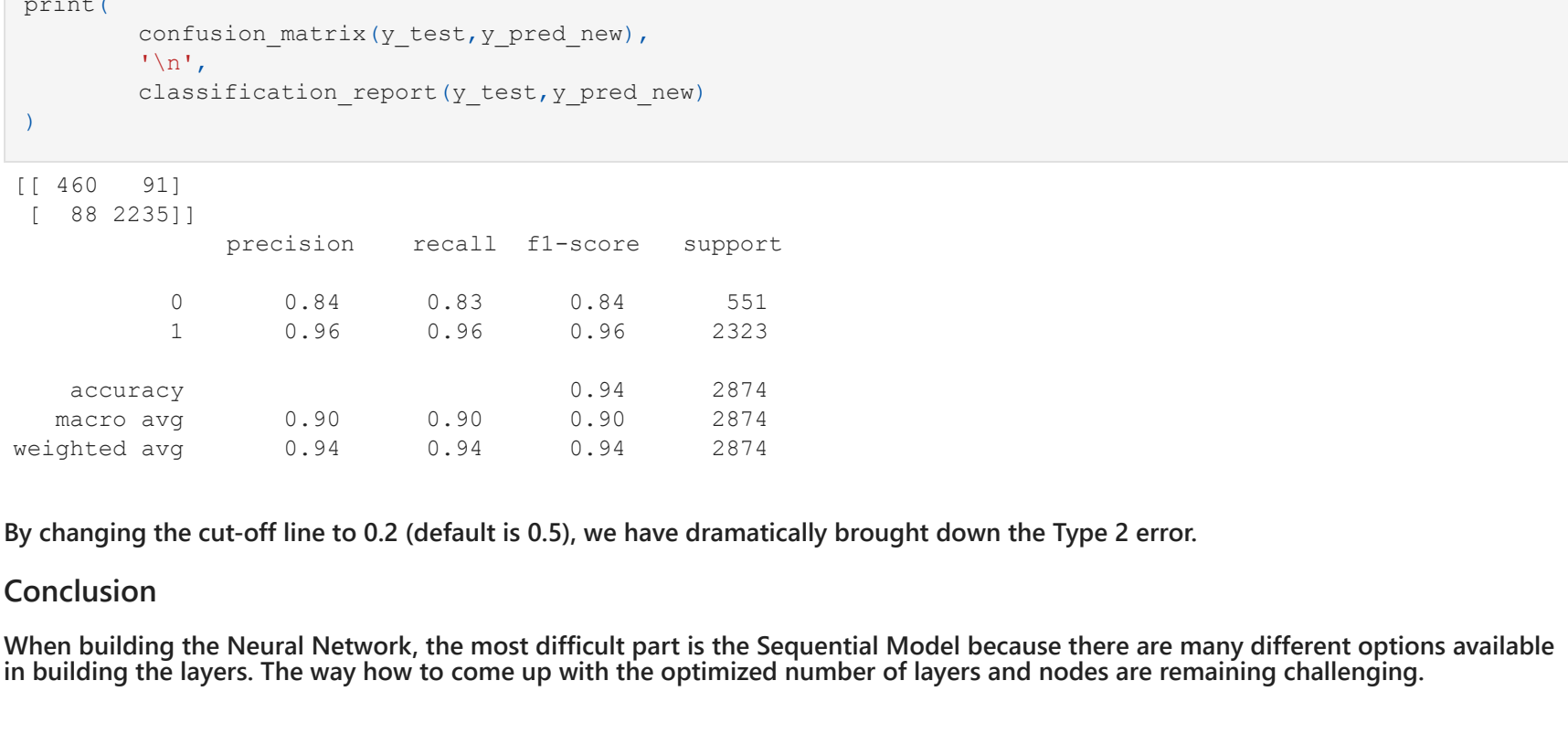
```
<keras.callbacks.History at 0x1fff4eae7c0>
```

In [216…

```python
pd.DataFrame(model_new.history.history)[['loss','val_loss']].plot()
```

Out[216…

```
<AxesSubplot:>
```



The graph shows that, by adding in Dropout layers, we have reduced the overfitting issue compared with the old model.

In [217…

```python
y_pred_new = model.predict(X_test)
threshold=0.2
y_pred_new=np.where(y_pred > threshold, 1,0)
y_pred_new
```

Out[217…

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [1],
       [1]])
```

In [109…

```python
print(
        confusion_matrix(y_test,y_pred_new),
        '\n',
        classification_report(y_test,y_pred_new)
)
```

```
[[ 460   91]
 [  88 2235]]
              precision    recall  f1-score   support

           0       0.84      0.83      0.84       551
           1       0.96      0.96      0.96      2323

    accuracy                           0.94      2874
   macro avg       0.90      0.90      0.90      2874
weighted avg       0.94      0.94      0.94      2874
```

By changing the cut-off line to 0.2 (default is 0.5), we have dramatically brought down the Type 2 error.

Conclusion

When building the Neural Network, the most difficult part is the Sequential Model because there are many different options available in building the layers. The way how to come up with the optimized number of layers and nodes are remaining challenging.