In [0]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [0]:
```
loan_data = pd.read_csv('LendingClubDataSet/loan_data.csv')
input_data = pd.read_csv('LendingClubDataSet/input.csv')
output_data = pd.read_csv('LendingClubDataSet/output.csv')
```

In [9]:
```
print(loan_data.head())
print(input_data.head())
print(output_data.head())
```

```
   credit.policy             purpose  ...  pub.rec  not.fully.paid
0              1  debt_consolidation  ...        0               0
1              1         credit_card  ...        0               0
2              1  debt_consolidation  ...        0               0
3              1  debt_consolidation  ...        0               0
4              1         credit_card  ...        0               0

[5 rows x 14 columns]
   1  0.1189   829.1  11.35040654  19.48  737  ...  0.3  0.4  1.1
0.5  0.6  0.7
0  1  0.1071  228.22    11.082143  14.29  707  ...    0    1    0
0    0    0
1  1  0.1357  366.86    10.373491  11.63  682  ...    0    0    1
0    0    0
2  1  0.1008  162.34    11.350407   8.10  712  ...    0    0    1
0    0    0
3  1  0.1426  102.92    11.299732  14.97  667  ...    0    1    0
0    0    0
4  1  0.0788  125.13    11.904968  16.98  727  ...    0    1    0
0    0    0

[5 rows x 18 columns]
   0  1
0  0  1
1  0  1
2  0  1
3  0  1
4  0  1
```

In [0]:
```
print(input_data.shape)
print(output_data.shape)
```

```
(9577, 18)
(9577, 2)
```

In [10]:
```
loan_data.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 [11]:
```
loan_data.describe()
```

Out[11]:

| | credit.policy | int.rate | installment | log.annual.inc | dti | fico | da |
|---|---|---|---|---|---|---|---|
| **count** | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | |
| **mean** | 0.804970 | 0.122640 | 319.089413 | 10.932117 | 12.606679 | 710.846314 | |
| **std** | 0.396245 | 0.026847 | 207.071301 | 0.614813 | 6.883970 | 37.970537 | |
| **min** | 0.000000 | 0.060000 | 15.670000 | 7.547502 | 0.000000 | 612.000000 | |
| **25%** | 1.000000 | 0.103900 | 163.770000 | 10.558414 | 7.212500 | 682.000000 | |
| **50%** | 1.000000 | 0.122100 | 268.950000 | 10.928884 | 12.665000 | 707.000000 | |
| **75%** | 1.000000 | 0.140700 | 432.762500 | 11.291293 | 17.950000 | 737.000000 | |
| **max** | 1.000000 | 0.216400 | 940.140000 | 14.528354 | 29.960000 | 827.000000 | |

In [12]:
```
loan_data['purpose'].value_counts()
```

Out[12]:
```
debt_consolidation    3957
all_other             2331
credit_card           1262
home_improvement       629
small_business         619
major_purchase         437
educational            343
Name: purpose, dtype: int64
```

# 1. Feature Transformation

Transform categorical values into numerical values (discrete)

In [0]:
```
from sklearn.preprocessing import  LabelEncoder

le_purpose = LabelEncoder()
le_purpose.fit(loan_data['purpose'])
loan_data['purpose'] = le_purpose.transform(loan_data['purpose'])
```

In [14]:
```
loan_data.head(10)
```

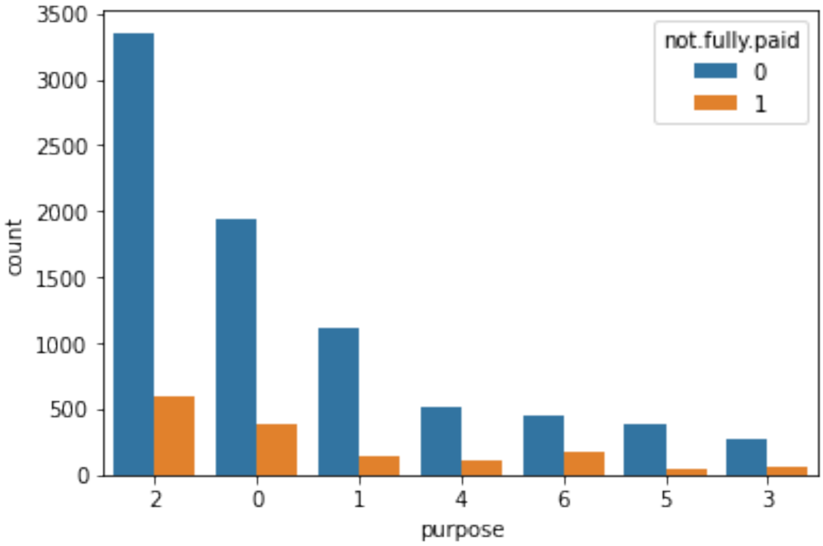
Out[14]:

| | credit.policy | purpose | int.rate | installment | log.annual.inc | dti | fico | days.with.cr.line |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 0.1189 | 829.10 | 11.350407 | 19.48 | 737 | 5639.958333 |
| **1** | 1 | 1 | 0.1071 | 228.22 | 11.082143 | 14.29 | 707 | 2760.000000 |
| **2** | 1 | 2 | 0.1357 | 366.86 | 10.373491 | 11.63 | 682 | 4710.000000 |
| **3** | 1 | 2 | 0.1008 | 162.34 | 11.350407 | 8.10 | 712 | 2699.958333 |
| **4** | 1 | 1 | 0.1426 | 102.92 | 11.299732 | 14.97 | 667 | 4066.000000 |
| **5** | 1 | 1 | 0.0788 | 125.13 | 11.904968 | 16.98 | 727 | 6120.041667 |
| **6** | 1 | 2 | 0.1496 | 194.02 | 10.714418 | 4.00 | 667 | 3180.041667 |
| **7** | 1 | 0 | 0.1114 | 131.22 | 11.002100 | 11.08 | 722 | 5116.000000 |
| **8** | 1 | 4 | 0.1134 | 87.19 | 11.407565 | 17.25 | 682 | 3989.000000 |
| **9** | 1 | 2 | 0.1221 | 84.12 | 10.203592 | 10.00 | 707 | 2730.041667 |

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

In [15]:
```
sns.countplot(x='purpose', data=loan_data, hue='not.fully.paid', or
der=loan_data.purpose.value_counts().index)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9aba55b9e8>




In [0]:
```
#print(pd.DataFrame(le_purpose.classes_[[2, 0, 1, 4, 6, 5, 3]], ind
ex=[2, 0, 1, 4, 6, 5, 3]))
```

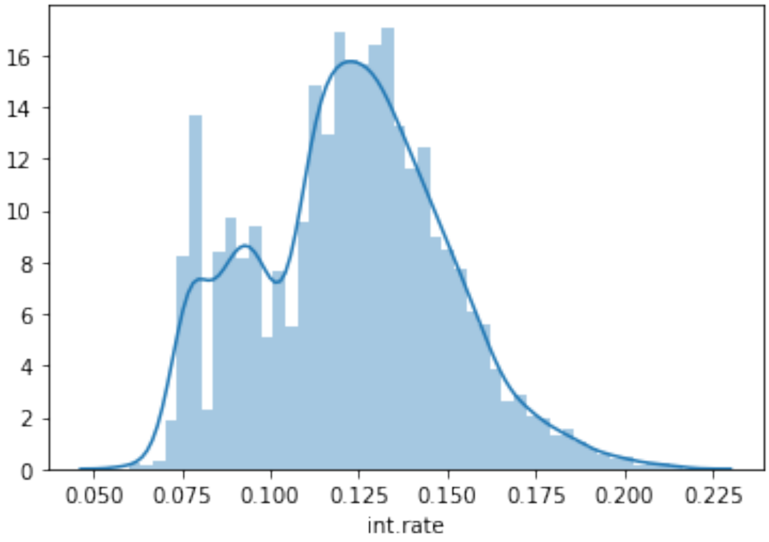
In [0]:

In [16]:
```
fg = sns.FacetGrid(data=loan_data, row='purpose', col='pub.rec', hu
e='not.fully.paid')
fg.map(sns.scatterplot, 'int.rate', 'log.annual.inc', alpha=0.7)
fg.add_legend()
```


purpose = 0 | pub.rec = 0
purpose = 0 | pub.rec = 1
purpose = 0 | pub.rec = 2
purpose = 0 | pub.rec = 3
purpose = 0 | pub.rec = 4
purpose = 0 | pub.rec = 5
log.annual.inc
int.rate
not.fully.paid
0
1

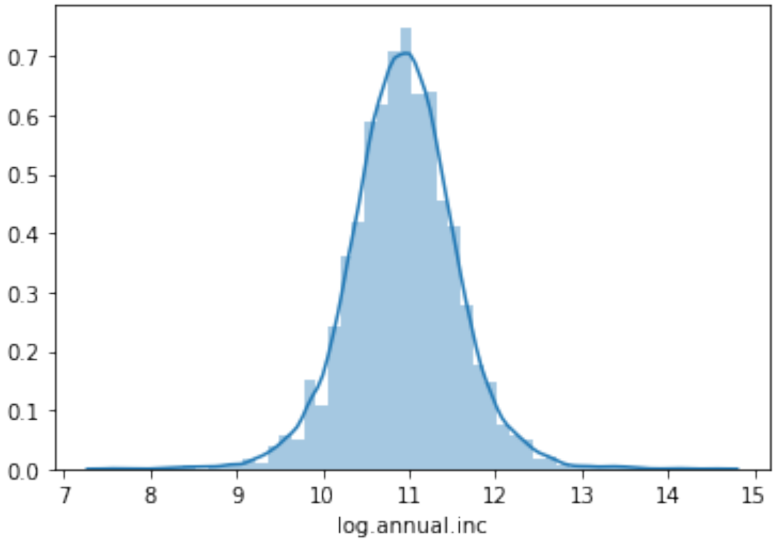
In [17]:
```
sns.distplot(loan_data['int.rate'])
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9ab591fb00>



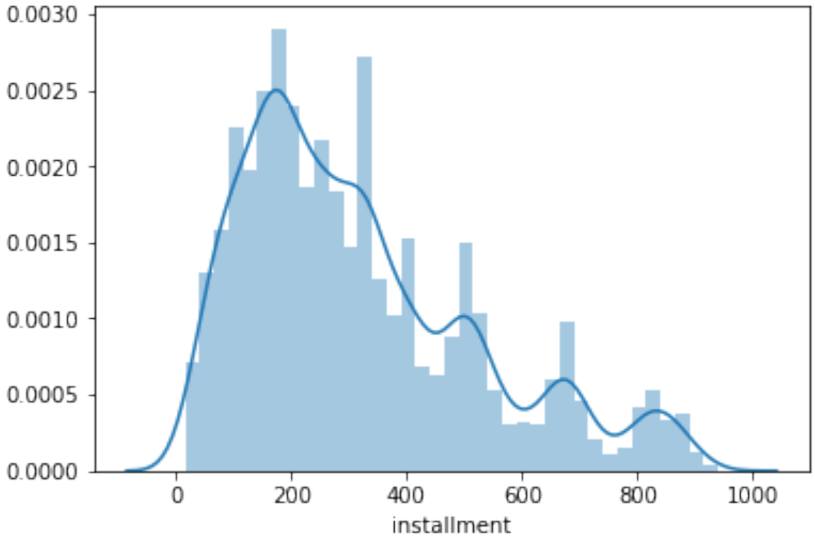

In [18]:
```
sns.distplot(loan_data['log.annual.inc'])
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9ab5a0b6a0>



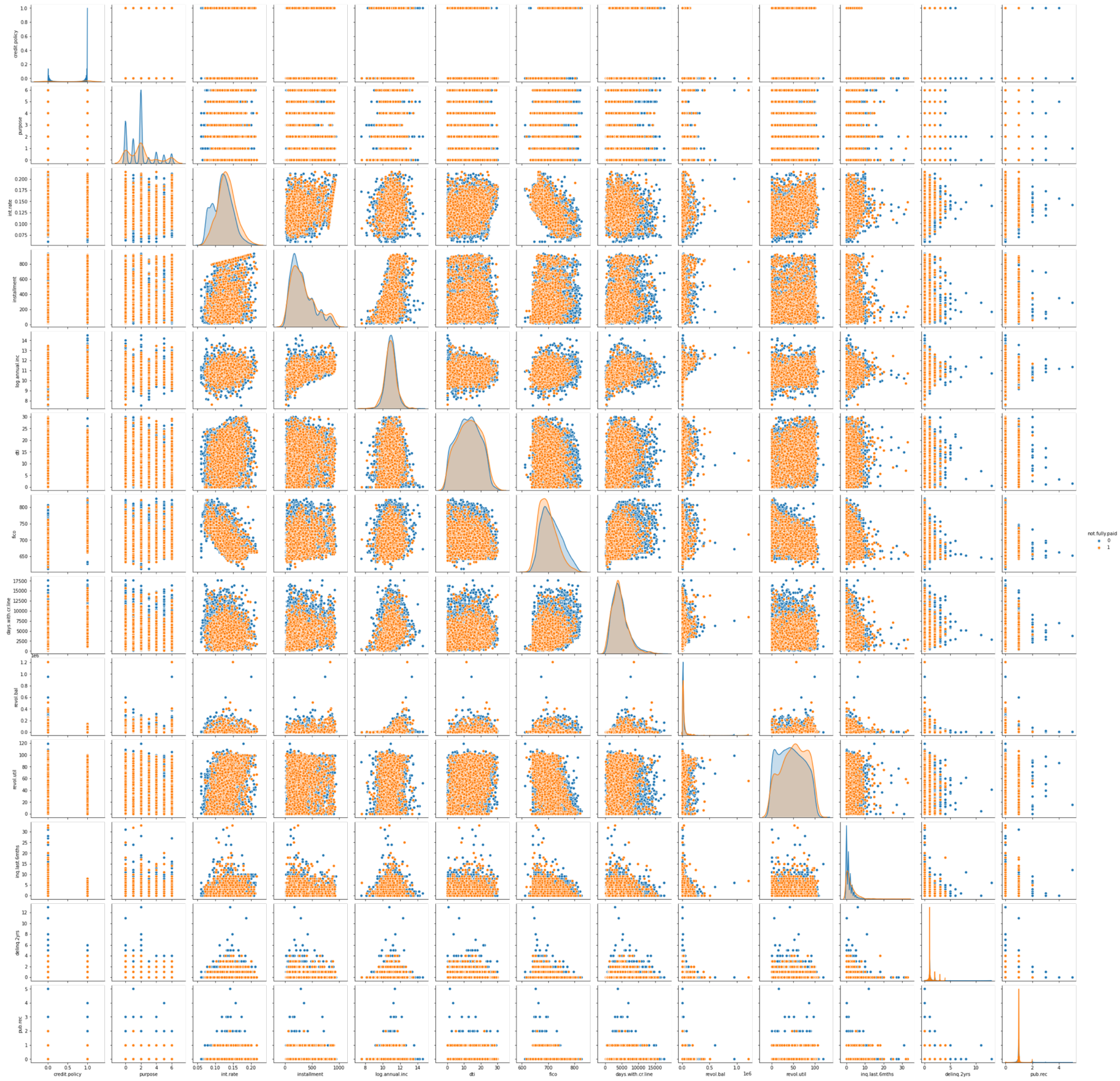
In [19]:
```
sns.distplot(loan_data['installment'])
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9ab3ed9860>




In [20]:
```
sns.pairplot(loan_data, hue='not.fully.paid')
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x7f9ab3e30dd8>



# 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**

In [0]:
```
data_corr = loan_data.corr()
```

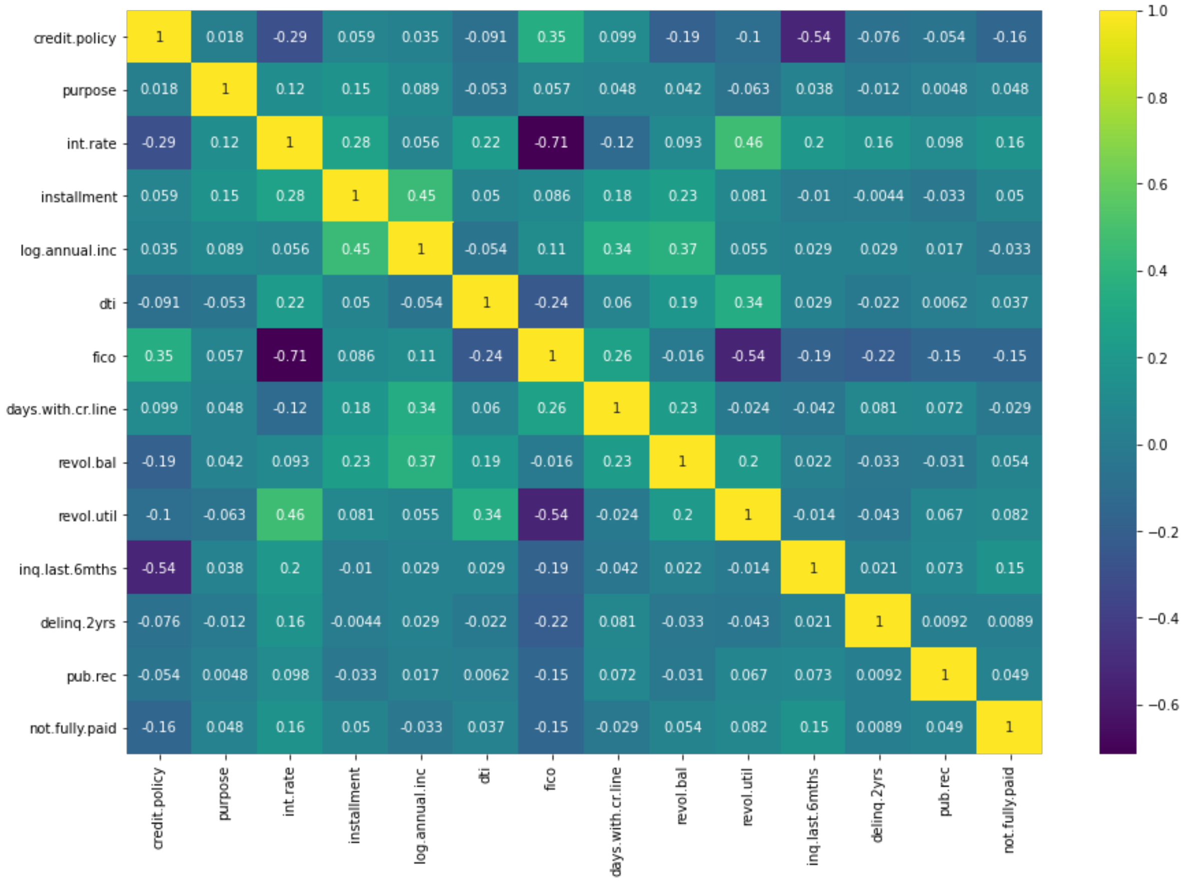
In [22]:
```
data_corr
```

Out[22]:

| | credit.policy | purpose | int.rate | installment | log.annual.inc | dti | |
|---|---|---|---|---|---|---|---|
| **credit.policy** | 1.000000 | 0.017569 | -0.294089 | 0.058770 | 0.034906 | -0.090901 | |
| **purpose** | 0.017569 | 1.000000 | 0.117067 | 0.154827 | 0.088958 | -0.053279 | |
| **int.rate** | -0.294089 | 0.117067 | 1.000000 | 0.276140 | 0.056383 | 0.220006 | - |
| **installment** | 0.058770 | 0.154827 | 0.276140 | 1.000000 | 0.448102 | 0.050202 | |
| **log.annual.inc** | 0.034906 | 0.088958 | 0.056383 | 0.448102 | 1.000000 | -0.054065 | |
| **dti** | -0.090901 | -0.053279 | 0.220006 | 0.050202 | -0.054065 | 1.000000 | - |
| **fico** | 0.348319 | 0.057337 | -0.714821 | 0.086039 | 0.114576 | -0.241191 | |
| **days.with.cr.line** | 0.099026 | 0.047526 | -0.124022 | 0.183297 | 0.336896 | 0.060101 | |
| **revol.bal** | -0.187518 | 0.042364 | 0.092527 | 0.233625 | 0.372140 | 0.188748 | - |
| **revol.util** | -0.104095 | -0.062947 | 0.464837 | 0.081356 | 0.054881 | 0.337109 | - |
| **inq.last.6mths** | -0.535511 | 0.037516 | 0.202780 | -0.010419 | 0.029171 | 0.029189 | - |
| **delinq.2yrs** | -0.076318 | -0.011701 | 0.156079 | -0.004368 | 0.029203 | -0.021792 | - |
| **pub.rec** | -0.054243 | 0.004793 | 0.098162 | -0.032760 | 0.016506 | 0.006209 | - |
| **not.fully.paid** | -0.158119 | 0.047907 | 0.159552 | 0.049955 | -0.033439 | 0.037362 | - |

In [23]:
```
plt.figure(figsize=(15, 10))
sns.heatmap(data_corr, cmap='viridis', annot=True)
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9ab06b60b8>



In [24]:
```
data_corr[np.sqrt(np.square(data_corr['not.fully.paid']))>0.1]['not.fully.paid']
```

Out[24]:
```
credit.policy     -0.158119
int.rate           0.159552
fico              -0.149666
inq.last.6mths     0.149452
not.fully.paid     1.000000
Name: not.fully.paid, dtype: float64
```

In [0]:

In [0]:
```
loan_data = pd.concat([pd.get_dummies(loan_data.purpose), loan_data.drop(['purpose'], axis=1)], axis=1)
```

In [26]:
```
loan_data.shape
```

Out[26]: (9578, 20)

# 4. Modeling

In [0]:
```
def getValue(data):
  return np.argmax(data)

X = input_data.values
Y  = np.array(list(map(getValue, output_data.values)))
```

## Splitting training and test data

In [0]:
```
from sklearn.model_selection import  train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size
=0.25, random_state=672)
```

In [0]:
```

```

## Training the model

In [0]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout


model = Sequential()


# input layer
model.add(Dense(18,  activation='relu'))

# hidden layer
model.add(Dense(18, activation='relu'))

# hidden layer
model.add(Dense(18, activation='relu'))


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

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

In [40]:
```
model.fit(
          x=x_train, y=y_train,
          epochs=30,
          batch_size=256
          )
```

```
Epoch 1/30
29/29 [==============================] - 0s 2ms/step - loss: 369.3
065 - accuracy: 0.4741
Epoch 2/30
29/29 [==============================] - 0s 1ms/step - loss: 43.29
62 - accuracy: 0.9390
Epoch 3/30
29/29 [==============================] - 0s 1ms/step - loss: 11.39
53 - accuracy: 0.8929
Epoch 4/30
29/29 [==============================] - 0s 1ms/step - loss: 4.917
2 - accuracy: 0.8995
Epoch 5/30
29/29 [==============================] - 0s 1ms/step - loss: 3.656
3 - accuracy: 0.9052
Epoch 6/30
29/29 [==============================] - 0s 2ms/step - loss: 3.548
2 - accuracy: 0.8946
Epoch 7/30
29/29 [==============================] - 0s 2ms/step - loss: 2.577
5 - accuracy: 0.8967
Epoch 8/30
29/29 [==============================] - 0s 1ms/step - loss: 3.108
8 - accuracy: 0.8942
Epoch 9/30
29/29 [==============================] - 0s 1ms/step - loss: 2.759
9 - accuracy: 0.8957
Epoch 10/30
29/29 [==============================] - 0s 1ms/step - loss: 1.615
8 - accuracy: 0.8972
Epoch 11/30
29/29 [==============================] - 0s 1ms/step - loss: 2.108
5 - accuracy: 0.9036
Epoch 12/30
29/29 [==============================] - 0s 1ms/step - loss: 2.339
7 - accuracy: 0.8865
Epoch 13/30
29/29 [==============================] - 0s 2ms/step - loss: 3.367
3 - accuracy: 0.8939
Epoch 14/30
29/29 [==============================] - 0s 1ms/step - loss: 3.080
6 - accuracy: 0.8876
Epoch 15/30
29/29 [==============================] - 0s 1ms/step - loss: 2.087
4 - accuracy: 0.9081
Epoch 16/30
29/29 [==============================] - 0s 2ms/step - loss: 0.997
```

```
3 - accuracy: 0.9106
Epoch 17/30
29/29 [==============================] - 0s 1ms/step - loss: 1.049
9 - accuracy: 0.9041
Epoch 18/30
29/29 [==============================] - 0s 1ms/step - loss: 1.337
1 - accuracy: 0.8896
Epoch 19/30
29/29 [==============================] - 0s 1ms/step - loss: 2.011
2 - accuracy: 0.8787
Epoch 20/30
29/29 [==============================] - 0s 1ms/step - loss: 4.239
7 - accuracy: 0.9062
Epoch 21/30
29/29 [==============================] - 0s 2ms/step - loss: 12.78
95 - accuracy: 0.8610
Epoch 22/30
29/29 [==============================] - 0s 1ms/step - loss: 5.425
8 - accuracy: 0.8965
Epoch 23/30
29/29 [==============================] - 0s 1ms/step - loss: 2.739
2 - accuracy: 0.8649
Epoch 24/30
29/29 [==============================] - 0s 1ms/step - loss: 3.100
9 - accuracy: 0.9176
Epoch 25/30
29/29 [==============================] - 0s 1ms/step - loss: 1.458
5 - accuracy: 0.8940
Epoch 26/30
29/29 [==============================] - 0s 1ms/step - loss: 1.971
0 - accuracy: 0.8910
Epoch 27/30
29/29 [==============================] - 0s 1ms/step - loss: 1.873
6 - accuracy: 0.8977
Epoch 28/30
29/29 [==============================] - 0s 1ms/step - loss: 2.665
6 - accuracy: 0.8954
Epoch 29/30
29/29 [==============================] - 0s 1ms/step - loss: 1.277
3 - accuracy: 0.8986
Epoch 30/30
29/29 [==============================] - 0s 1ms/step - loss: 1.274
8 - accuracy: 0.9103
```

Out[40]:
```
<tensorflow.python.keras.callbacks.History at 0x7f9a28564080>
```

In [0]:
```
predictions = model.predict_classes(x_test)
```

## Evaluation

In [0]:
```
from sklearn.metrics import  classification_report, confusion_matri
x
```

In [0]:
```

```

In [42]:
```
confusion_matrix(y_test, predictions)
```

Out[42]:
```
array([[  17,  164],
       [ 148, 2066]])
```

In [44]:
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.10      0.09      0.10       181
           1       0.93      0.93      0.93      2214

    accuracy                           0.87      2395
   macro avg       0.51      0.51      0.51      2395
weighted avg       0.86      0.87      0.87      2395
```

In [0]:
```

```

In [0]:
```

```

In [0]:
```

```

In [0]:
```

```

In [0]:
```

```