LendingClub_Project_Final_Submission

April 20, 2021

Lending Club Loan Data Analysis

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
plt.rcParams['figure.figsize'] = (10,10)
```

```
Using TensorFlow backend.
```

Step 1. Import Data File

[2]:
```
loan_df = pd.read_csv('loan_data.csv')
loan_df.shape
```

[2]:
```
(9578, 14)
```

Step 2. Data Preprocessing

2.1 Check Top 5 Rows

[3]:
```
loan_df.head()
```

[3]:
```
   credit.policy             purpose  int.rate  installment  log.annual.inc  \
0              1  debt_consolidation    0.1189       829.10       11.350407
1              1         credit_card    0.1071       228.22       11.082143
2              1  debt_consolidation    0.1357       366.86       10.373491
3              1  debt_consolidation    0.1008       162.34       11.350407
4              1         credit_card    0.1426       102.92       11.299732
```

	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths
0	19.48	737	5639.958333	28854	52.1	0
1	14.29	707	2760.000000	33623	76.7	0
2	11.63	682	4710.000000	3511	25.6	1
3	8.10	712	2699.958333	33667	73.2	1
4	14.97	667	4066.000000	4740	39.5	0

	delinq.2yrs	pub.rec	not.fully.paid
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	1	0	0

2.2 Check Data Stats

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

[4]:

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

	fico	days.with.cr.line	revol.bal	revol.util
count	9578.000000	9578.000000	9.578000e+03	9578.000000
mean	710.846314	4560.767197	1.691396e+04	46.799236
std	37.970537	2496.930377	3.375619e+04	29.014417
min	612.000000	178.958333	0.000000e+00	0.000000
25%	682.000000	2820.000000	3.187000e+03	22.600000
50%	707.000000	4139.958333	8.596000e+03	46.300000
75%	737.000000	5730.000000	1.824950e+04	70.900000
max	827.000000	17639.958330	1.207359e+06	119.000000

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

2.3 Check for Null Values

[5]:
```
loan_df.isna().sum().sum()
#No Null Values Present in Dataset
```

[5]: 0

2.4 Check for Unique Values in Columns

[6]:
```
for col in loan_df.columns:
    print("Unique values in {} Column are".format(col), loan_df[col].unique())
    print("-----------------------------------------------------------------")
```

```
Unique values in credit.policy Column are [1 0]
-----------------------------------------------------------------
Unique values in purpose Column are ['debt_consolidation' 'credit_card'
'all_other' 'home_improvement'
 'small_business' 'major_purchase' 'educational']
-----------------------------------------------------------------
Unique values in int.rate Column are [0.1189 0.1071 0.1357 0.1008 0.1426 0.0788
0.1496 0.1114 0.1134 0.1221
 0.1347 0.1324 0.0859 0.0714 0.0863 0.1103 0.1317 0.0894 0.1039 0.1513
 0.08   0.1355 0.1229 0.0901 0.0743 0.1375 0.0807 0.1028 0.087  0.1122
 0.0996 0.0933 0.0838 0.0775 0.1059 0.1596 0.1154 0.1343 0.1249 0.0964
 0.1186 0.1501 0.128  0.1091 0.1217 0.1533 0.0712 0.1438 0.1565 0.1467
 0.1312 0.147  0.1407 0.1014 0.1046 0.133  0.0983 0.1393 0.092  0.1236
 0.1362 0.1078 0.1583 0.1109 0.1141 0.1267 0.1204 0.0951 0.1172 0.1299
 0.1488 0.152  0.1425 0.1836 0.1615 0.06   0.0832 0.1261 0.0945 0.1197
 0.1387 0.0976 0.1292 0.0737 0.0768 0.1166 0.1418 0.1545 0.1482 0.1703
 0.145  0.1671 0.1576 0.1608 0.164  0.1734 0.1051 0.157  0.1222 0.1273
 0.1379 0.1253 0.1128 0.1286 0.1287 0.097  0.1001 0.1538 0.1191 0.1254
 0.1159 0.138  0.1096 0.1064 0.1349 0.1033 0.1475 0.1601 0.1507 0.1412
 0.1633 0.1696 0.1146 0.1304 0.1272 0.1209 0.1083 0.1178 0.1241 0.1588
 0.0907 0.102  0.1336 0.1557 0.0938 0.1493 0.1462 0.1367 0.0963 0.1126
 0.1442 0.1148 0.1399 0.1525 0.143  0.1392 0.1904 0.1872 0.162  0.1715
 0.1568 0.0988 0.1062 0.1746 0.0932 0.1411 0.1505 0.1316 0.16   0.1158
 0.1284 0.1095 0.1695 0.1474 0.1537 0.1632 0.0751 0.1422 0.1218 0.1663
 0.1726 0.1853 0.1348 0.1531 0.1635 0.179  0.1758 0.1843 0.1821 0.1183
 0.074  0.1682 0.0774 0.1322 0.2086 0.1461 0.1311 0.1916 0.1884 0.1607
 0.2011 0.167  0.1979 0.1739 0.1704 0.1913 0.1774 0.0705 0.1878 0.1809
 0.2017 0.1982 0.1947 0.2121 0.1459 0.1385 0.1025 0.1099 0.1136 0.2052
 0.1719 0.0639 0.1645 0.0676 0.1793 0.209  0.2016 0.183  0.1941 0.1756
 0.1691 0.1754 0.1722 0.1628 0.1786 0.1659 0.1741 0.1709 0.1457 0.1804
 0.1646 0.1551 0.1772 0.1829 0.1861 0.1797 0.1766 0.1854 0.1665 0.1791
 0.1886 0.1759 0.1443 0.1728 0.1936 0.1683 0.1778 0.2164 0.1867]
-----------------------------------------------------------------
Unique values in installment Column are [829.1  228.22 366.86 … 161.01 257.7
853.43]
```

```
----------------------------------------------------------------
Unique values in log.annual.inc Column are [11.35040654 11.08214255 10.37349118
… 12.29225034 10.99909533
 10.11047245]
----------------------------------------------------------------
Unique values in dti Column are [19.48 14.29 11.63 … 10.31 23.74 24.05]
----------------------------------------------------------------
Unique values in fico Column are [737 707 682 712 667 727 722 677 662 767 747
702 672 797 772 782 802 812
 742 692 777 762 757 787 717 752 792 627 687 697 732 822 632 807 817 827
 642 647 652 657 637 612 617 622]
----------------------------------------------------------------
Unique values in days.with.cr.line Column are [ 5639.958333  2760.        4710.
…  3423.041667  5916.
 10474.      ]
----------------------------------------------------------------
Unique values in revol.bal Column are [28854 33623  3511 …   184 10036 37879]
----------------------------------------------------------------
Unique values in revol.util Column are [ 52.1   76.7   25.6   … 104.3  106.4
69.14]
----------------------------------------------------------------
Unique values in inq.last.6mths Column are [ 0  1  2  3  4  5  6  8  7 33  9 18
14 15 13 12 10 19 11 16 20 27 25 28
 31 24 17 32]
----------------------------------------------------------------
Unique values in delinq.2yrs Column are [ 0  1  2  4  3  5  6 13  7  8 11]
----------------------------------------------------------------
Unique values in pub.rec Column are [0 1 2 3 4 5]
----------------------------------------------------------------
Unique values in not.fully.paid Column are [0 1]
----------------------------------------------------------------
```

Training Data Observation with above command * Credit Policy and Not Fully Paid columns are binary columns * Purpose Columns is the only Categorical Column * All the other columns has numerical values * Target variable is binary - 0's and 1's

2.5 Perform Label Encoding for Categorical Columns

[7]:
```
loan_df_unchanged = loan_df.copy()
```

[8]:
```
df_dtypes = loan_df.dtypes.reset_index()
df_dtypes.columns=['count', 'dtype']
grpd_df = df_dtypes.groupby('dtype').aggregate('count').reset_index()
grpd_df
```

[8]:

	dtype	count
0	int64	7
1	float64	6

```
2   object      1
```

[9]:
```python
encoding_col_list = df_dtypes[df_dtypes.dtype==object]['count'].values
encoding_col_list
```

[9]:
```
array(['purpose'], dtype=object)
```

[10]:
```python
le = LabelEncoder()
```

[11]:
```python
for column in encoding_col_list:
    loan_df[column] = le.fit_transform(loan_df[column])
    print("Label Encoding for {} column Completed Successfully".format(column))
```

```
Label Encoding for purpose column Completed Successfully
```

[12]:
```python
loan_df.head(3)
```

[12]:
```
   credit.policy  purpose  int.rate  installment  log.annual.inc    dti  fico  \
0              1        2    0.1189       829.10       11.350407  19.48   737
1              1        1    0.1071       228.22       11.082143  14.29   707
2              1        2    0.1357       366.86       10.373491  11.63   682

   days.with.cr.line  revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
0        5639.958333      28854        52.1               0            0
1        2760.000000      33623        76.7               0            0
2        4710.000000       3511        25.6               1            0

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

2.6 Check if the Dataset is Balanced

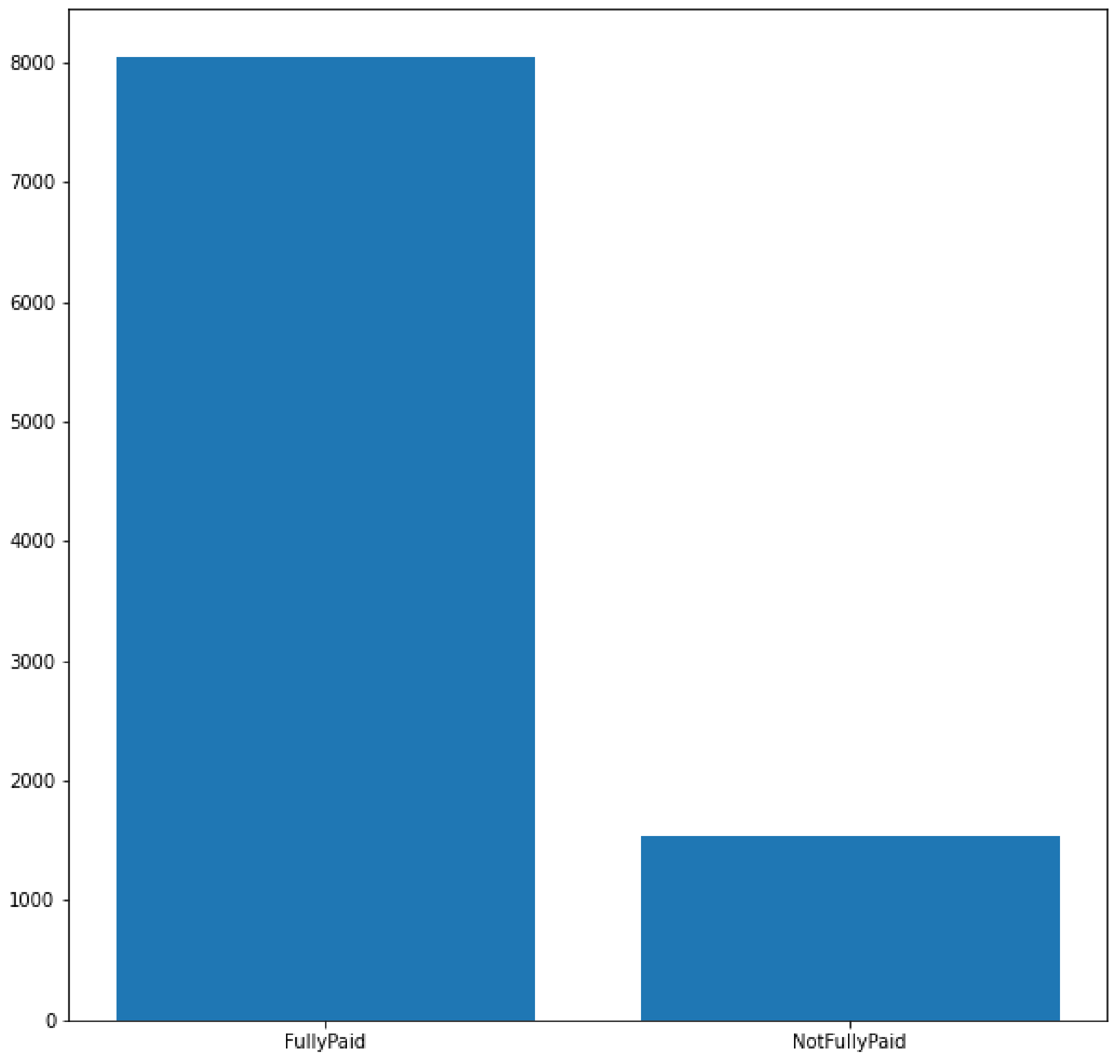
Here "NotFullyPaid" is the Dependent Feature

[13]:
```python
loan_df['not.fully.paid'].value_counts()
```

[13]:
```
0    8045
1    1533
Name: not.fully.paid, dtype: int64
```

[14]:
```python
# fullyPaid = loan_df['not.fully.paid'].value_counts()[0]
# notFullyPaid = loan_df['not.fully.paid'].value_counts()[1]
plt.bar(("FullyPaid", "NotFullyPaid"), loan_df['not.fully.paid'].value_counts())
```

[14]:
```
<BarContainer object of 2 artists>
```



Dataset is clearly found to be imbalanced from above graph

out of 9578 - 8045 people has paid the loan fully and only 1533 people haven't paid fully

So, Oversampling is required inorder to remove the biasness by algorithm towards the FullyPaid customer (notfullypaid=0)

2.7 Assign Features and label

[15]:
```
X = loan_df.drop('not.fully.paid', axis=1)
X.head()
```

[15]:
```
   credit.policy  purpose  int.rate  installment  log.annual.inc    dti  fico  \
0              1        2    0.1189       829.10       11.350407  19.48   737
1              1        1    0.1071       228.22       11.082143  14.29   707
2              1        2    0.1357       366.86       10.373491  11.63   682
```

```
3              1        2    0.1008       162.34       11.350407    8.10    712
4              1        1    0.1426       102.92       11.299732   14.97    667

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

   pub.rec
0        0
1        0
2        0
3        0
4        0
```

[16]:
```
y = loan_df['not.fully.paid']
y.head()
```

[16]:
```
0    0
1    0
2    0
3    0
4    0
Name: not.fully.paid, dtype: int64
```

2.8 Perform Oversampling using RandomOverSampler

[17]:
```
sampler = RandomOverSampler(random_state=42)
```

[18]:
```
X_over_sampled, y_over_sampled = sampler.fit_sample(X, y)
```

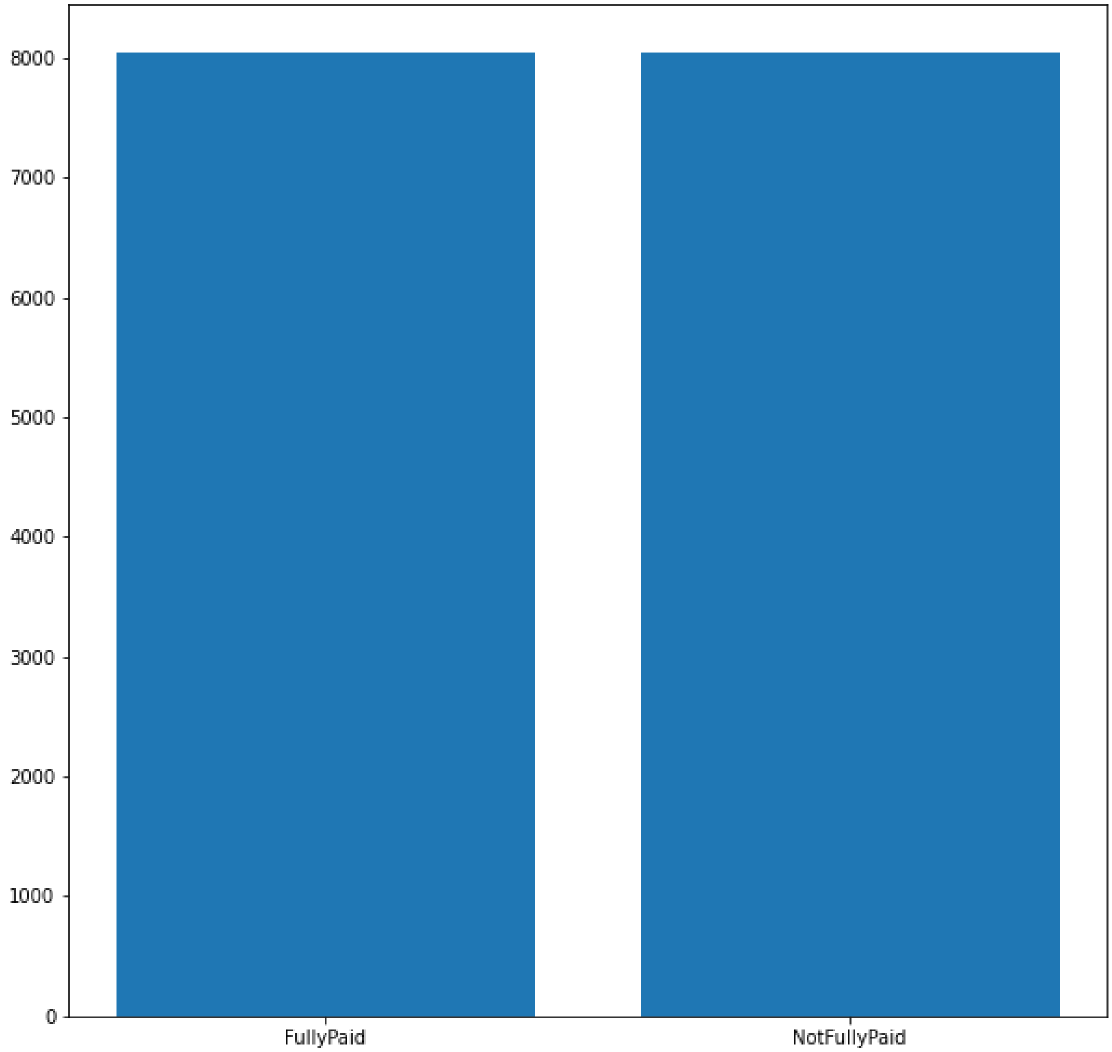
[19]:
```
print(X_over_sampled.shape)
print(y_over_sampled.shape)
```

```
(16090, 13)
(16090,)
```

[20]:
```
print(y_over_sampled.value_counts())
plt.bar(("FullyPaid", "NotFullyPaid"), y_over_sampled.value_counts())
```

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

[20]:
```
<BarContainer object of 2 artists>
```



[21]:
```
print("The shape of Original dataset is", X.shape)
print("The shape of Oversampled dataset is", X_over_sampled.shape)
```

```
The shape of Original dataset is (9578, 13)
The shape of Oversampled dataset is (16090, 13)
```

2.8 Detect Outliers in the Oversampled Dataset

2.8.1 To Detect Outliers - rejoin the X_oversampled and y_oversampled data

[22]:
```
sampled_df = pd.concat([X_over_sampled, y_over_sampled], axis=1)
print("Shape of Sampled Dataframe is: ", sampled_df.shape)
sampled_df.head()
```

```
Shape of Sampled Dataframe is:  (16090, 14)
```

[22]:
```
   credit.policy  purpose  int.rate  installment  log.annual.inc    dti  fico  \
0              1        2    0.1189       829.10       11.350407  19.48   737
1              1        1    0.1071       228.22       11.082143  14.29   707
2              1        2    0.1357       366.86       10.373491  11.63   682
3              1        2    0.1008       162.34       11.350407   8.10   712
4              1        1    0.1426       102.92       11.299732  14.97   667

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

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

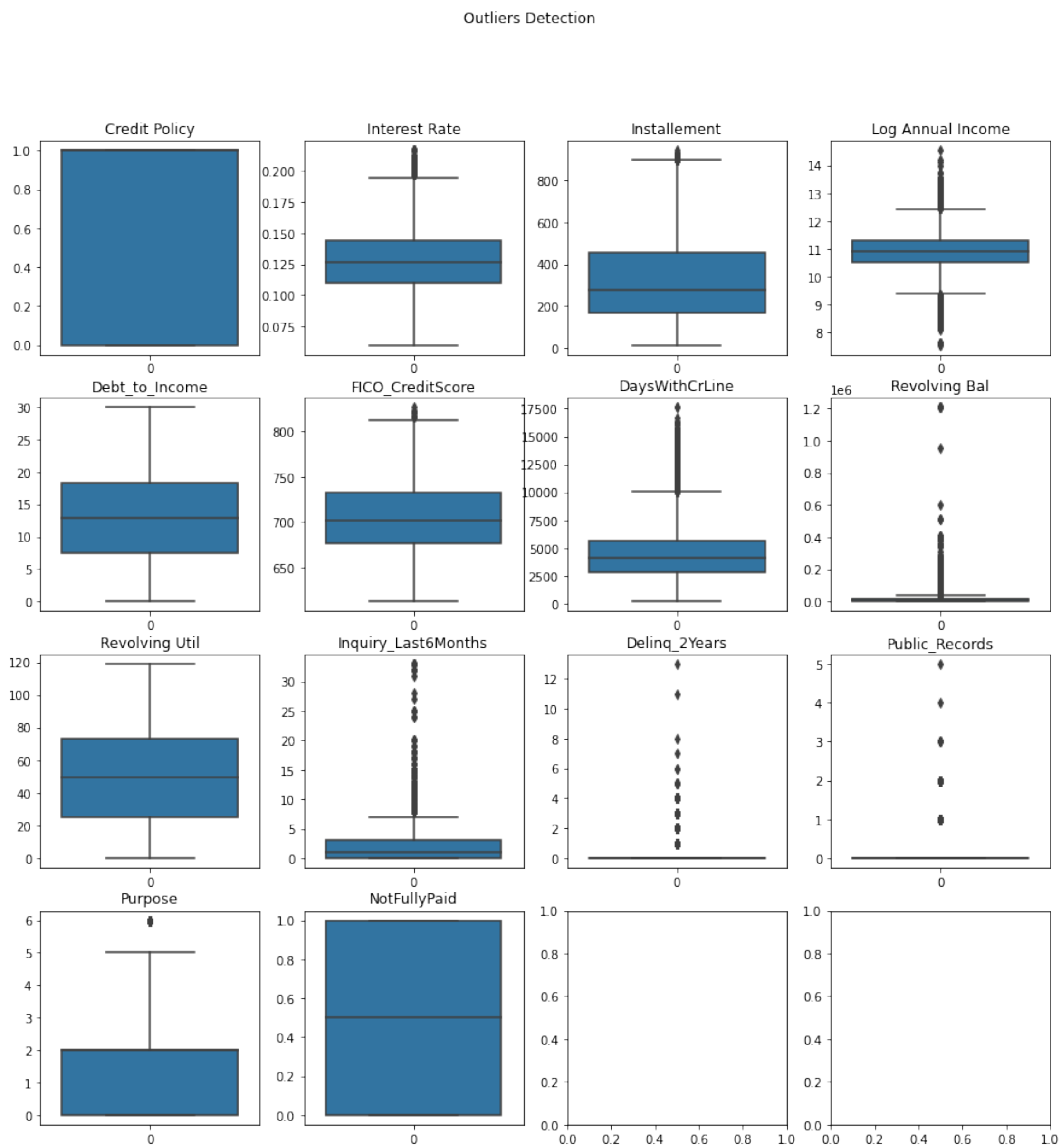
[23]:
```
sampled_df_copy = sampled_df.copy()
```

2.8.2 Check and remove Outliers using boxplot

[24]:
```
fig, axes = plt.subplots(4, 4, figsize=(15, 15))
fig.suptitle('Outliers Detection')
sns.boxplot(ax=axes[0, 0], data = sampled_df['credit.policy']).
 ↪set_title("Credit Policy")
sns.boxplot(ax=axes[0, 1], data = sampled_df['int.rate']).set_title("Interest␣
 ↪Rate")
sns.boxplot(ax=axes[0, 2], data = sampled_df['installment']).
 ↪set_title("Installement")
sns.boxplot(ax=axes[0, 3], data = sampled_df['log.annual.inc']).set_title("Log␣
 ↪Annual Income")
sns.boxplot(ax=axes[1, 0], data = sampled_df['dti']).set_title("Debt_to_Income")
sns.boxplot(ax=axes[1, 1], data = sampled_df['fico']).
 ↪set_title("FICO_CreditScore")
sns.boxplot(ax=axes[1, 2], data = sampled_df['days.with.cr.line']).
 ↪set_title("DaysWithCrLine")
sns.boxplot(ax=axes[1, 3], data = sampled_df['revol.bal']).set_title("Revolving␣
 ↪Bal")
sns.boxplot(ax=axes[2, 0], data = sampled_df['revol.util']).
 ↪set_title("Revolving Util")
sns.boxplot(ax=axes[2, 1], data = sampled_df['inq.last.6mths']).
 ↪set_title("Inquiry_Last6Months")
```

```
sns.boxplot(ax=axes[2, 2], data = sampled_df['delinq.2yrs']).
↪set_title("Delinq_2Years")
sns.boxplot(ax=axes[2, 3], data = sampled_df['pub.rec']).
↪set_title("Public_Records")
sns.boxplot(ax=axes[3, 0], data = sampled_df['purpose']).set_title("Purpose")
sns.boxplot(ax=axes[3, 1], data = sampled_df['not.fully.paid']).
↪set_title("NotFullyPaid")
```

[24]: Text(0.5, 1.0, 'NotFullyPaid')



Treating the Outliers

[25]:
```
def outlier_treatment(datacolumn):
    sorted(datacolumn)
    Q1,Q3 = np.percentile(datacolumn,[25,75])
    IQR = Q3-Q1
    lower_range = Q1 - (1.5*IQR)
    upper_range = Q3 + (1.5*IQR)
    return lower_range,upper_range
```

Outliers Treatment has no much Impact, delinq.2yrs and pub.rec are becoming NAN after treament. So, Not treating the outliers

[26]:
```
# l,u = outlier_treatment(sampled_df["int.rate"])
# sampled_df[(sampled_df["int.rate"] > u) | (sampled_df["int.rate"] < l)]
# sampled_df.drop(sampled_df[(sampled_df["int.rate"] > u) | (sampled_df["int.
↪rate"] < l)].index,inplace=True)
# #139 rows
# l,u = outlier_treatment(sampled_df["installment"])
# sampled_df[(sampled_df["installment"] > u) | (sampled_df["installment"] < l)]
# sampled_df.drop(sampled_df[(sampled_df["installment"] > u) |␣
↪(sampled_df["installment"] < l)].index,inplace=True)
# # 100 rows
# l,u = outlier_treatment(sampled_df["log.annual.inc"])
# sampled_df[(sampled_df["log.annual.inc"] > u) | (sampled_df["log.annual.inc"]␣
↪< l)]
# sampled_df.drop(sampled_df[(sampled_df["log.annual.inc"] > u) |␣
↪(sampled_df["log.annual.inc"] < l)].index,inplace=True)
# # 362 rows
# l,u = outlier_treatment(sampled_df["fico"])
# sampled_df[(sampled_df["fico"] > u) | (sampled_df["fico"] < l)].shape
# sampled_df.drop(sampled_df[(sampled_df["fico"] > u) | (sampled_df["fico"] <␣
↪l)].index,inplace=True)
# # 18 rows
# l,u = outlier_treatment(sampled_df["days.with.cr.line"])
# sampled_df[(sampled_df["days.with.cr.line"] > u) | (sampled_df["days.with.cr.
↪line"] < l)].shape
# sampled_df.drop(sampled_df[(sampled_df["days.with.cr.line"] > u) |␣
↪(sampled_df["days.with.cr.line"] < l)].index,inplace=True)
# # 555 rows
# l,u = outlier_treatment(sampled_df["revol.bal"])
# sampled_df[(sampled_df["revol.bal"] > u) | (sampled_df["revol.bal"] < l)].
↪shape
# sampled_df.drop(sampled_df[(sampled_df["revol.bal"] > u) | (sampled_df["revol.
↪bal"] < l)].index,inplace=True)
# # 1416 rows
# l,u = outlier_treatment(sampled_df["inq.last.6mths"])
# sampled_df[(sampled_df["inq.last.6mths"] > u) | (sampled_df["inq.last.6mths"]␣
↪< l)].shape
```

```
# sampled_df.drop(sampled_df[(sampled_df["inq.last.6mths"] > u) |␣
↪(sampled_df["inq.last.6mths"] < l)].index,inplace=True)
# 527 rows
# l,u = outlier_treatment(sampled_df["delinq.2yrs"])
# sampled_df[(sampled_df["delinq.2yrs"] > u) | (sampled_df["delinq.2yrs"] < l)].
↪shape
# sampled_df.drop(sampled_df[(sampled_df["delinq.2yrs"] > u) |␣
↪(sampled_df["delinq.2yrs"] < l)].index,inplace=True)
# # 1963 rows
# l,u = outlier_treatment(sampled_df["pub.rec"])
# sampled_df[(sampled_df["pub.rec"] > u) | (sampled_df["pub.rec"] < l)].shape
# sampled_df.drop(sampled_df[(sampled_df["pub.rec"] > u) | (sampled_df["pub.
↪rec"] < l)].index,inplace=True)
# # 1145 rows
# l,u = outlier_treatment(sampled_df["purpose"])
# sampled_df[(sampled_df["purpose"] > u) | (sampled_df["purpose"] < l)].shape
# sampled_df.drop(sampled_df[(sampled_df["purpose"] > u) |␣
↪(sampled_df["purpose"] < l)].index,inplace=True)
# # 1337 rows
```

[27]:
```
sampled_df.shape
```

[27]:
```
(16090, 14)
```

2.9 Assign the dependent label and Independent features again from sampled and outlier treated data

[28]:
```
X_over_sampled = sampled_df.drop('not.fully.paid', axis=1)
y_over_sampled = sampled_df['not.fully.paid']
```

[29]:
```
print(X_over_sampled.shape)
print(y_over_sampled.shape)
```

```
(16090, 13)
(16090,)
```

2.10 Check if the dataset is imbalanced after the outliers are removed from sampled data

[30]:
```
# print(y_over_sampled.value_counts())
# plt.bar(("FullyPaid", "NotFullyPaid"), y_over_sampled.value_counts())
```

Dataset is not imbalanced (As there is no significant difference) after the Outliers are treated

[ ]:
```
```

2.11 Perform Train Test Split of the data

[31]:
```
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(X_over_sampled,y_over_sampled, test_size=0.30,␣
 ↪random_state=42)
```

[32]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(11263, 13)
(4827, 13)
(11263,)
(4827,)
```

2.12 Perform Standardization using Standard Scaler

[33]:
```
from sklearn.preprocessing import StandardScaler
```

[34]:
```
scaler = StandardScaler()
```

Applying fit on Train Data

[35]:
```
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
```

Applying Transform without fit for Test Data

[36]:
```
X_test_scaled = scaler.transform(X_test)
```

[37]:
```
print(X_train_scaled.shape)
print(X_test_scaled.shape)
```

```
(11263, 13)
(4827, 13)
```

[ ]:
```

```

Step 3. Feature Engineering

3.1 Feature Engineering using Correlation

[38]:
```
sampled_df.corr()
```

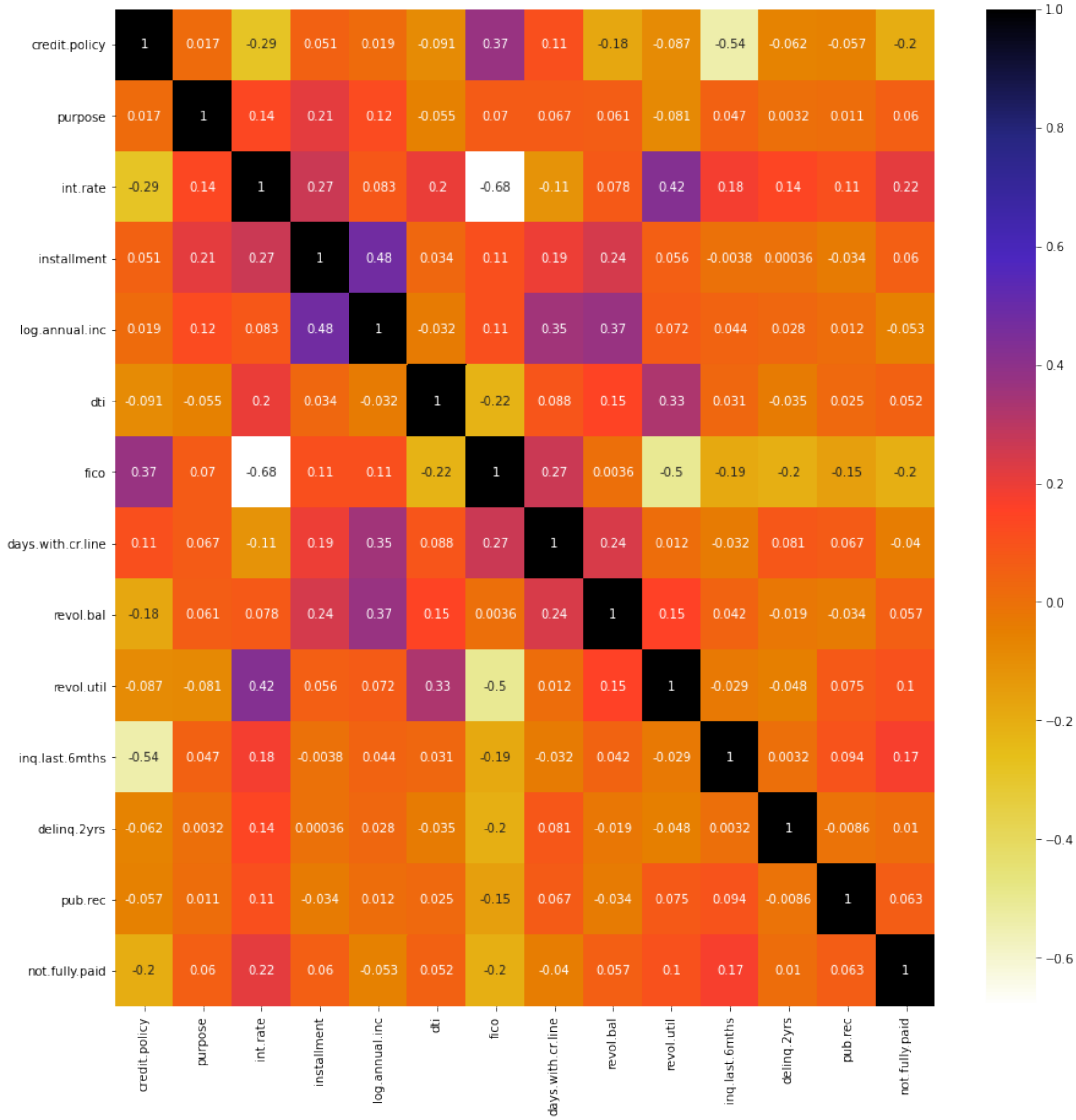
[38]:
```
                credit.policy   purpose  int.rate  installment  \
credit.policy        1.000000  0.016951 -0.286761     0.050919
purpose              0.016951  1.000000  0.138771     0.211918
int.rate            -0.286761  0.138771  1.000000     0.269195
installment          0.050919  0.211918  0.269195     1.000000
log.annual.inc       0.019125  0.124596  0.082992     0.479625
```

```
dti                  -0.090719 -0.054517  0.202168     0.033582
fico                  0.371264  0.069898 -0.682194     0.109620
days.with.cr.line     0.109544  0.067324 -0.112010     0.194459
revol.bal            -0.176510  0.061231  0.077827     0.244933
revol.util           -0.087100 -0.080684  0.423886     0.055947
inq.last.6mths       -0.540411  0.047405  0.183928    -0.003804
delinq.2yrs          -0.062096  0.003205  0.144630     0.000359
pub.rec              -0.057081  0.010630  0.107155    -0.034339
not.fully.paid       -0.197046  0.059746  0.215874     0.059783

                   log.annual.inc       dti      fico  days.with.cr.line  \
credit.policy            0.019125 -0.090719  0.371264           0.109544
purpose                  0.124596 -0.054517  0.069898           0.067324
int.rate                 0.082992  0.202168 -0.682194          -0.112010
installment              0.479625  0.033582  0.109620           0.194459
log.annual.inc           1.000000 -0.032237  0.107600           0.351122
dti                     -0.032237  1.000000 -0.221844           0.088069
fico                     0.107600 -0.221844  1.000000           0.265073
days.with.cr.line        0.351122  0.088069  0.265073           1.000000
revol.bal                0.374387  0.146485  0.003571           0.240694
revol.util               0.071787  0.326387 -0.501385           0.011517
inq.last.6mths           0.043998  0.030524 -0.188331          -0.031798
delinq.2yrs              0.028143 -0.034776 -0.203227           0.081435
pub.rec                  0.011696  0.025163 -0.154982           0.067153
not.fully.paid          -0.053177  0.051964 -0.204929          -0.040445

                   revol.bal  revol.util  inq.last.6mths  delinq.2yrs  \
credit.policy      -0.176510   -0.087100       -0.540411    -0.062096
purpose             0.061231   -0.080684        0.047405     0.003205
int.rate            0.077827    0.423886        0.183928     0.144630
installment         0.244933    0.055947       -0.003804     0.000359
log.annual.inc      0.374387    0.071787        0.043998     0.028143
dti                 0.146485    0.326387        0.030524    -0.034776
fico                0.003571   -0.501385       -0.188331    -0.203227
days.with.cr.line   0.240694    0.011517       -0.031798     0.081435
revol.bal           1.000000    0.154683        0.042413    -0.019313
revol.util          0.154683    1.000000       -0.029071    -0.047815
inq.last.6mths      0.042413   -0.029071        1.000000     0.003208
delinq.2yrs        -0.019313   -0.047815        0.003208     1.000000
pub.rec            -0.034489    0.075139        0.093896    -0.008630
not.fully.paid      0.056710    0.101486        0.172950     0.010338

                    pub.rec  not.fully.paid
credit.policy     -0.057081       -0.197046
purpose            0.010630        0.059746
int.rate           0.107155        0.215874
installment       -0.034339        0.059783
```

```
log.annual.inc     0.011696       -0.053177
dti                0.025163        0.051964
fico              -0.154982       -0.204929
days.with.cr.line  0.067153       -0.040445
revol.bal         -0.034489        0.056710
revol.util         0.075139        0.101486
inq.last.6mths     0.093896        0.172950
delinq.2yrs       -0.008630        0.010338
pub.rec            1.000000        0.062588
not.fully.paid     0.062588        1.000000
```

[39]:
```python
plt.figure(figsize=(15,15))
sns.heatmap(sampled_df.corr(), annot=True, cmap = plt.cm.CMRmap_r)
```

[39]:
```
<AxesSubplot:>
```



[ ]:

[40]:
```
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if(corr_matrix.iloc[i,j] > threshold):
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

Apply the above correlation check on Training data only as we should not perform correlaton check on test data

[41]:
```
corr_features = correlation(X_train, 0.7)
len(set(corr_features))
```

[41]: 0

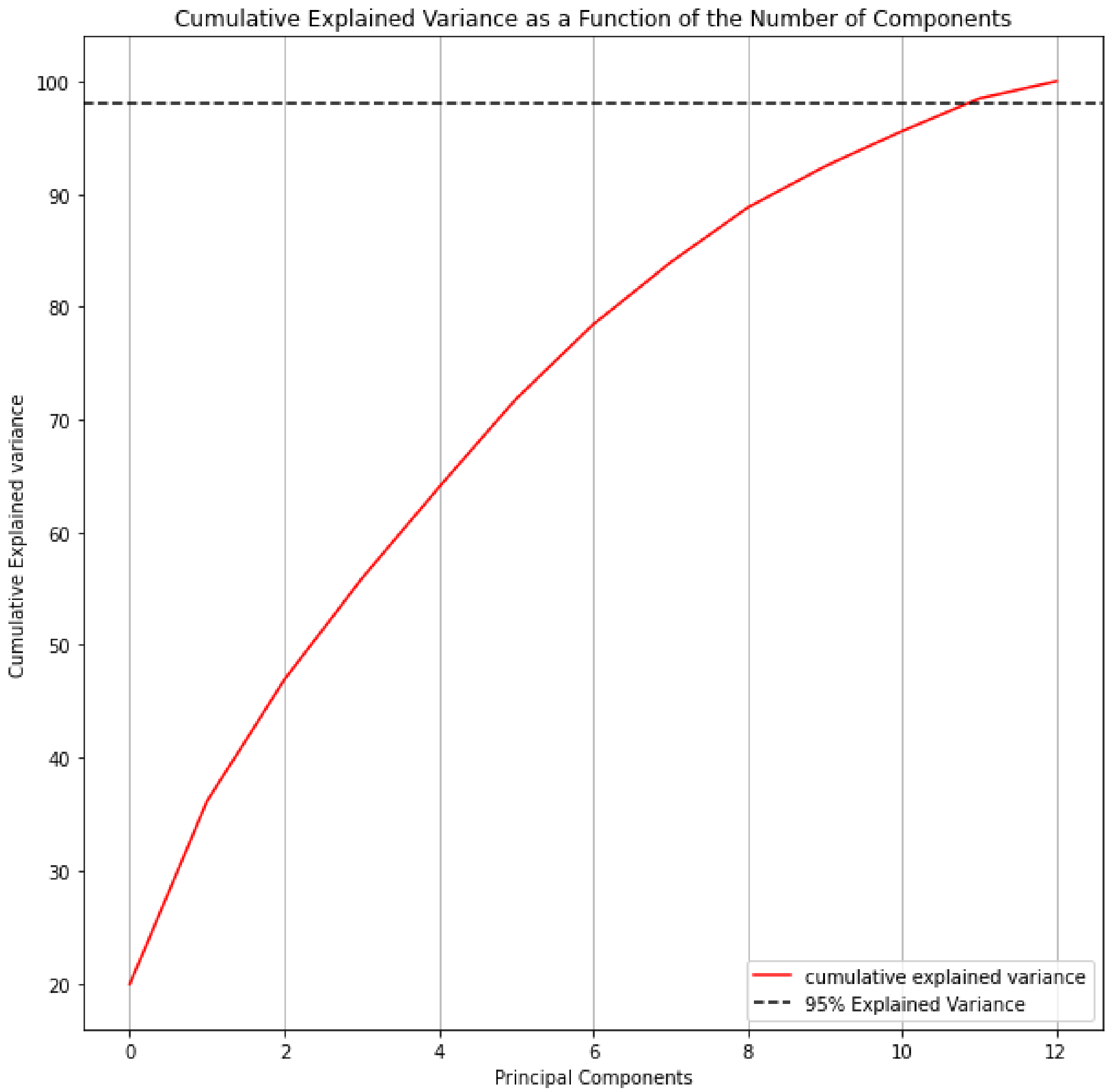
Correlation below 0.3 are considered to be weak; 0.3-0.7 are moderate; >0.7 are strong

There are no strongly correlated features in the data

3.2 Feature selection using PCA

[42]:
```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(X_train_scaled)
cumsum = np.cumsum(pca.explained_variance_ratio_)*100
d = [n for n in range(len(cumsum))]
plt.figure(figsize=(10, 10))
plt.plot(d,cumsum, color = 'red',label='cumulative explained variance')
plt.title('Cumulative Explained Variance as a Function of the Number of␣
 ↪Components')
plt.ylabel('Cumulative Explained variance')
plt.grid(axis='x')
plt.xlabel('Principal Components')
plt.axhline(y = 98, color='k', linestyle='--', label = '95% Explained Variance')
plt.legend(loc='best')
```

[42]: <matplotlib.legend.Legend at 0x7ff91c2fd4d0>



[43]:
```
pca = PCA(n_components=0.98)
pca.fit(X_train_scaled)
print(pca.explained_variance_ratio_.size)
X_train_scaled = pca.transform(X_train_scaled)
print(X_train_scaled.shape)
X_test_scaled = pca.transform(X_test_scaled)
print(X_test_scaled.shape)
```

```
12
(11263, 12)
(4827, 12)
```

[ ]:

Step 4. Build Models

Build Machine Learning Models before a Deep Learning Model to compare the accuracy

#1: Apply Logistic Regression

[44]:
```
model_lr = LogisticRegression()
```

[45]:
```
model_lr.fit(X_train_scaled, y_train)
```

[45]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[46]:
```
y_pred_lr = model_lr.predict(X_test_scaled)
```

[47]:
```
accuracy_score(y_test, y_pred_lr)
```

[47]:
```
0.620261031696706
```

#2 Apply K Nearest Neighbors Classifier Algorithm

[48]:
```
model_knn = KNeighborsClassifier(n_neighbors=5)
```

[49]:
```
model_knn.fit(X_train_scaled,y_train)
```

[49]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

[50]:
```
y_pred_knn = model_knn.predict(X_test_scaled)
```

[51]:
```
accuracy_score(y_test, y_pred_knn)
```

[51]:
```
0.7470478558110628
```

#3 Apply Decision Tree Classification Algorithm

[52]:
```
from sklearn.tree import DecisionTreeClassifier
```

[53]:
```
model_dtr = DecisionTreeClassifier(random_state=2)
```

[54]:
```
model_dtr.fit(X_train_scaled,y_train)
```

[54]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
```

```
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=2, splitter='best')
```

[55]:
```
y_pred_dtr = model_dtr.predict(X_test_scaled)
```

[56]:
```
accuracy_score(y_test, y_pred_dtr)
```

[56]:
```
0.8875077688004972
```

[ ]:
```

```

Step 5. Ensemble Techniques

#1 Apply Random Forest Regressor Algorithm

[57]:
```
from sklearn.ensemble import RandomForestClassifier
```

[58]:
```
model_rfr = RandomForestClassifier(n_estimators=40, random_state=2)
```

[59]:
```
model_rfr.fit(X_train_scaled,y_train)
```

[59]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=40,
                       n_jobs=None, oob_score=False, random_state=2, verbose=0,
                       warm_start=False)
```

[60]:
```
y_pred_rfr = model_rfr.predict(X_test_scaled)
```

[61]:
```
accuracy_score(y_test, y_pred_rfr)
```

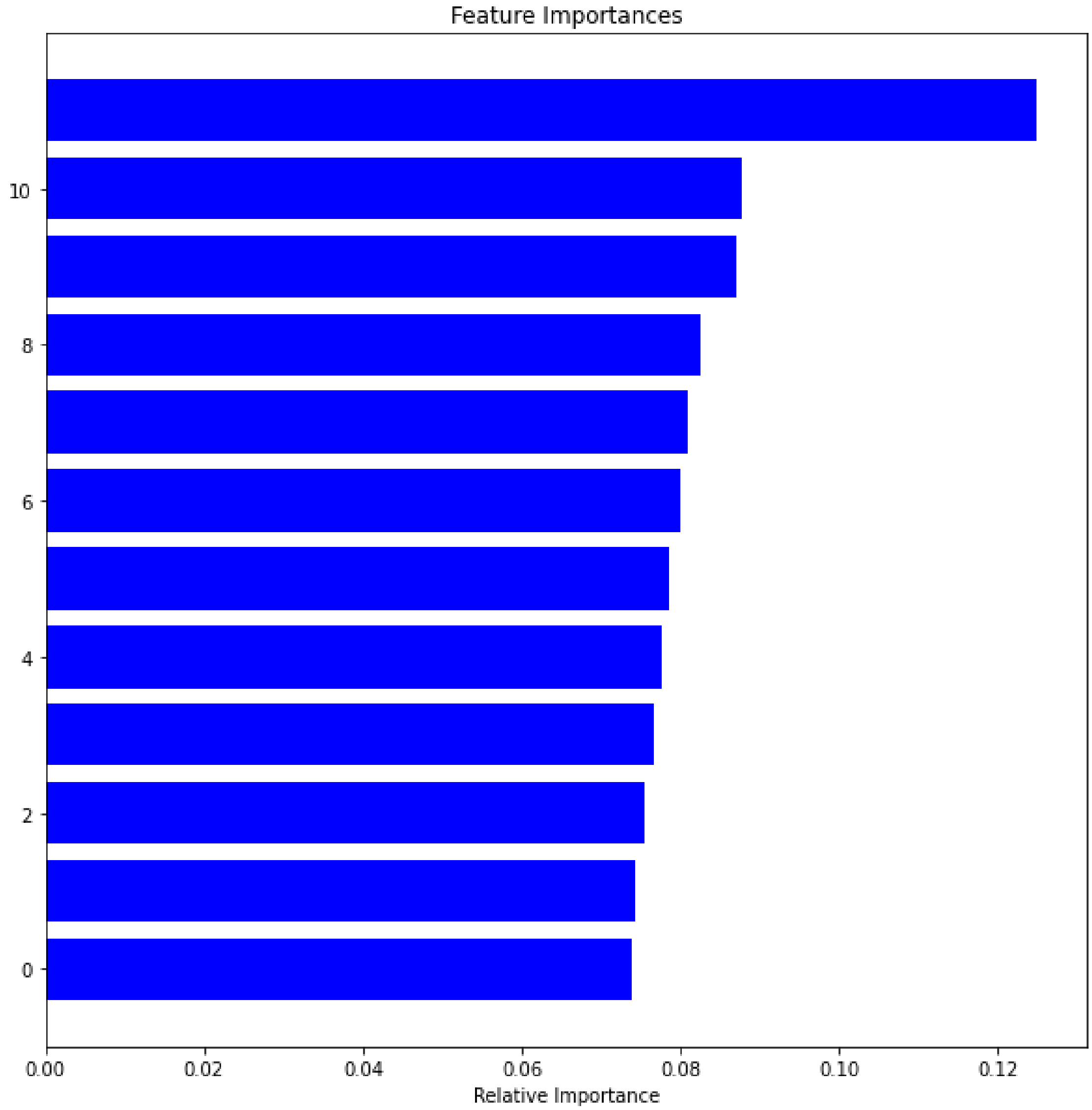
[61]:
```
0.9585663973482494
```

[ ]:
```

```

Importance of Features for Random Forest Classifier

[62]:
```
# features = X_train_scaled.columns   --- Which columns are picked in PCA,␣
 ↪order of columns, we are not aware - So not plotting Features Names
importances = model_rfr.feature_importances_
indices = np.argsort(importances)[-50:]  # top 50 features
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
# plt.yticks(range(len(indices)), [features[i] for i in indices])
```

```
plt.xlabel('Relative Importance')
plt.show()
```



[63]:
```
X_train_scaled.shape[1]
```

[63]: 12

Deep Learning Model using Keras with Tensor Flow backend

[64]:
```
model = Sequential()
model.add(Dense(60, input_dim=X_train_scaled.shape[1], activation='relu'))
model.add(Dropout(0.2))
```

```
model.add(Dense(120, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(60, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
model.fit(X_train_scaled, y_train, epochs=400, batch_size=256,␣
 ↪validation_data=(X_test_scaled, y_test))
loss_df = pd.DataFrame(model.history.history)
print(loss_df.head())
loss_df[['val_loss', 'loss']].plot()
```

```
Train on 11263 samples, validate on 4827 samples
Epoch 1/400
11263/11263 [==============================] - 0s 40us/step - loss: 0.6685 -
accuracy: 0.5918 - val_loss: 0.6498 - val_accuracy: 0.6265
Epoch 2/400
11263/11263 [==============================] - 0s 17us/step - loss: 0.6512 -
accuracy: 0.6177 - val_loss: 0.6422 - val_accuracy: 0.6317
Epoch 3/400
11263/11263 [==============================] - 0s 17us/step - loss: 0.6454 -
accuracy: 0.6273 - val_loss: 0.6397 - val_accuracy: 0.6337
Epoch 4/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6408 -
accuracy: 0.6312 - val_loss: 0.6362 - val_accuracy: 0.6317
Epoch 5/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6390 -
accuracy: 0.6354 - val_loss: 0.6347 - val_accuracy: 0.6385
Epoch 6/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6377 -
accuracy: 0.6342 - val_loss: 0.6338 - val_accuracy: 0.6399
Epoch 7/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.6356 -
accuracy: 0.6424 - val_loss: 0.6330 - val_accuracy: 0.6385
Epoch 8/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6360 -
accuracy: 0.6388 - val_loss: 0.6322 - val_accuracy: 0.6341
Epoch 9/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6317 -
accuracy: 0.6432 - val_loss: 0.6301 - val_accuracy: 0.6391
Epoch 10/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6316 -
accuracy: 0.6409 - val_loss: 0.6288 - val_accuracy: 0.6433
Epoch 11/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.6291 -
accuracy: 0.6483 - val_loss: 0.6280 - val_accuracy: 0.6443
```

```
Epoch 12/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6284 -
accuracy: 0.6458 - val_loss: 0.6265 - val_accuracy: 0.6437
Epoch 13/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6256 -
accuracy: 0.6498 - val_loss: 0.6253 - val_accuracy: 0.6445
Epoch 14/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6261 -
accuracy: 0.6508 - val_loss: 0.6244 - val_accuracy: 0.6420
Epoch 15/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6250 -
accuracy: 0.6517 - val_loss: 0.6230 - val_accuracy: 0.6453
Epoch 16/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.6231 -
accuracy: 0.6522 - val_loss: 0.6216 - val_accuracy: 0.6466
Epoch 17/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6209 -
accuracy: 0.6524 - val_loss: 0.6223 - val_accuracy: 0.6439
Epoch 18/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.6209 -
accuracy: 0.6488 - val_loss: 0.6192 - val_accuracy: 0.6459
Epoch 19/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6187 -
accuracy: 0.6534 - val_loss: 0.6168 - val_accuracy: 0.6472
Epoch 20/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6177 -
accuracy: 0.6560 - val_loss: 0.6166 - val_accuracy: 0.6522
Epoch 21/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6173 -
accuracy: 0.6569 - val_loss: 0.6149 - val_accuracy: 0.6511
Epoch 22/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6140 -
accuracy: 0.6641 - val_loss: 0.6134 - val_accuracy: 0.6542
Epoch 23/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6160 -
accuracy: 0.6588 - val_loss: 0.6120 - val_accuracy: 0.6567
Epoch 24/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6112 -
accuracy: 0.6660 - val_loss: 0.6107 - val_accuracy: 0.6584
Epoch 25/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6090 -
accuracy: 0.6628 - val_loss: 0.6097 - val_accuracy: 0.6542
Epoch 26/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6123 -
accuracy: 0.6650 - val_loss: 0.6075 - val_accuracy: 0.6569
Epoch 27/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6058 -
accuracy: 0.6679 - val_loss: 0.6062 - val_accuracy: 0.6621
```

```
Epoch 28/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6071 -
accuracy: 0.6686 - val_loss: 0.6039 - val_accuracy: 0.6627
Epoch 29/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.6049 -
accuracy: 0.6690 - val_loss: 0.6026 - val_accuracy: 0.6640
Epoch 30/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.6042 -
accuracy: 0.6673 - val_loss: 0.6026 - val_accuracy: 0.6654
Epoch 31/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.6048 -
accuracy: 0.6722 - val_loss: 0.6043 - val_accuracy: 0.6642
Epoch 32/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5990 -
accuracy: 0.6718 - val_loss: 0.6009 - val_accuracy: 0.6677
Epoch 33/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.5988 -
accuracy: 0.6770 - val_loss: 0.5973 - val_accuracy: 0.6679
Epoch 34/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5947 -
accuracy: 0.6785 - val_loss: 0.5961 - val_accuracy: 0.6679
Epoch 35/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5956 -
accuracy: 0.6712 - val_loss: 0.5953 - val_accuracy: 0.6644
Epoch 36/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5967 -
accuracy: 0.6733 - val_loss: 0.5963 - val_accuracy: 0.6729
Epoch 37/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5932 -
accuracy: 0.6755 - val_loss: 0.5922 - val_accuracy: 0.6758
Epoch 38/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5922 -
accuracy: 0.6786 - val_loss: 0.5908 - val_accuracy: 0.6714
Epoch 39/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5896 -
accuracy: 0.6789 - val_loss: 0.5926 - val_accuracy: 0.6733
Epoch 40/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5923 -
accuracy: 0.6805 - val_loss: 0.5907 - val_accuracy: 0.6712
Epoch 41/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5856 -
accuracy: 0.6887 - val_loss: 0.5882 - val_accuracy: 0.6801
Epoch 42/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5870 -
accuracy: 0.6841 - val_loss: 0.5861 - val_accuracy: 0.6845
Epoch 43/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5860 -
accuracy: 0.6863 - val_loss: 0.5860 - val_accuracy: 0.6797
```

```
Epoch 44/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5882 -
accuracy: 0.6784 - val_loss: 0.5858 - val_accuracy: 0.6888
Epoch 45/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5841 -
accuracy: 0.6871 - val_loss: 0.5838 - val_accuracy: 0.6861
Epoch 46/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5834 -
accuracy: 0.6876 - val_loss: 0.5827 - val_accuracy: 0.6783
Epoch 47/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5795 -
accuracy: 0.6903 - val_loss: 0.5808 - val_accuracy: 0.6895
Epoch 48/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5797 -
accuracy: 0.6876 - val_loss: 0.5810 - val_accuracy: 0.6878
Epoch 49/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5758 -
accuracy: 0.6912 - val_loss: 0.5773 - val_accuracy: 0.6938
Epoch 50/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5736 -
accuracy: 0.6939 - val_loss: 0.5748 - val_accuracy: 0.6880
Epoch 51/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5773 -
accuracy: 0.6908 - val_loss: 0.5762 - val_accuracy: 0.6955
Epoch 52/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5741 -
accuracy: 0.6947 - val_loss: 0.5733 - val_accuracy: 0.6946
Epoch 53/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5747 -
accuracy: 0.6930 - val_loss: 0.5733 - val_accuracy: 0.6924
Epoch 54/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5724 -
accuracy: 0.6913 - val_loss: 0.5764 - val_accuracy: 0.6932
Epoch 55/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5703 -
accuracy: 0.6966 - val_loss: 0.5713 - val_accuracy: 0.6963
Epoch 56/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5723 -
accuracy: 0.6940 - val_loss: 0.5746 - val_accuracy: 0.6969
Epoch 57/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5669 -
accuracy: 0.7004 - val_loss: 0.5707 - val_accuracy: 0.6996
Epoch 58/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5693 -
accuracy: 0.6986 - val_loss: 0.5713 - val_accuracy: 0.6977
Epoch 59/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5689 -
accuracy: 0.6958 - val_loss: 0.5704 - val_accuracy: 0.7058
```

```
Epoch 60/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.5621 -
accuracy: 0.7071 - val_loss: 0.5689 - val_accuracy: 0.6988
Epoch 61/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5659 -
accuracy: 0.6990 - val_loss: 0.5650 - val_accuracy: 0.7025
Epoch 62/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5653 -
accuracy: 0.7017 - val_loss: 0.5657 - val_accuracy: 0.6996
Epoch 63/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5607 -
accuracy: 0.7050 - val_loss: 0.5625 - val_accuracy: 0.7019
Epoch 64/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5580 -
accuracy: 0.7024 - val_loss: 0.5606 - val_accuracy: 0.7004
Epoch 65/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5623 -
accuracy: 0.7014 - val_loss: 0.5636 - val_accuracy: 0.7106
Epoch 66/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.5579 -
accuracy: 0.7064 - val_loss: 0.5599 - val_accuracy: 0.7015
Epoch 67/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5584 -
accuracy: 0.7081 - val_loss: 0.5568 - val_accuracy: 0.7071
Epoch 68/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5574 -
accuracy: 0.7070 - val_loss: 0.5565 - val_accuracy: 0.7077
Epoch 69/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.5579 -
accuracy: 0.7066 - val_loss: 0.5553 - val_accuracy: 0.7075
Epoch 70/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5505 -
accuracy: 0.7128 - val_loss: 0.5531 - val_accuracy: 0.7118
Epoch 71/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5536 -
accuracy: 0.7096 - val_loss: 0.5555 - val_accuracy: 0.7071
Epoch 72/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5451 -
accuracy: 0.7212 - val_loss: 0.5524 - val_accuracy: 0.7075
Epoch 73/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5475 -
accuracy: 0.7141 - val_loss: 0.5521 - val_accuracy: 0.7029
Epoch 74/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5485 -
accuracy: 0.7124 - val_loss: 0.5476 - val_accuracy: 0.7145
Epoch 75/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5495 -
accuracy: 0.7138 - val_loss: 0.5490 - val_accuracy: 0.7178
```

```
Epoch 76/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5509 -
accuracy: 0.7144 - val_loss: 0.5463 - val_accuracy: 0.7205
Epoch 77/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5466 -
accuracy: 0.7165 - val_loss: 0.5476 - val_accuracy: 0.7191
Epoch 78/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5469 -
accuracy: 0.7129 - val_loss: 0.5460 - val_accuracy: 0.7230
Epoch 79/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5501 -
accuracy: 0.7151 - val_loss: 0.5442 - val_accuracy: 0.7249
Epoch 80/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5425 -
accuracy: 0.7190 - val_loss: 0.5411 - val_accuracy: 0.7197
Epoch 81/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5410 -
accuracy: 0.7205 - val_loss: 0.5403 - val_accuracy: 0.7299
Epoch 82/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5439 -
accuracy: 0.7170 - val_loss: 0.5420 - val_accuracy: 0.7286
Epoch 83/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5404 -
accuracy: 0.7179 - val_loss: 0.5395 - val_accuracy: 0.7311
Epoch 84/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5426 -
accuracy: 0.7204 - val_loss: 0.5406 - val_accuracy: 0.7282
Epoch 85/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5385 -
accuracy: 0.7264 - val_loss: 0.5406 - val_accuracy: 0.7243
Epoch 86/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5412 -
accuracy: 0.7228 - val_loss: 0.5370 - val_accuracy: 0.7284
Epoch 87/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5408 -
accuracy: 0.7182 - val_loss: 0.5383 - val_accuracy: 0.7296
Epoch 88/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5381 -
accuracy: 0.7262 - val_loss: 0.5384 - val_accuracy: 0.7251
Epoch 89/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5416 -
accuracy: 0.7234 - val_loss: 0.5374 - val_accuracy: 0.7317
Epoch 90/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5372 -
accuracy: 0.7243 - val_loss: 0.5361 - val_accuracy: 0.7325
Epoch 91/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5365 -
accuracy: 0.7231 - val_loss: 0.5359 - val_accuracy: 0.7267
```

```
Epoch 92/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5329 -
accuracy: 0.7252 - val_loss: 0.5308 - val_accuracy: 0.7249
Epoch 93/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5307 -
accuracy: 0.7246 - val_loss: 0.5303 - val_accuracy: 0.7315
Epoch 94/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5304 -
accuracy: 0.7319 - val_loss: 0.5322 - val_accuracy: 0.7296
Epoch 95/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5264 -
accuracy: 0.7329 - val_loss: 0.5315 - val_accuracy: 0.7305
Epoch 96/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5307 -
accuracy: 0.7271 - val_loss: 0.5290 - val_accuracy: 0.7404
Epoch 97/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5288 -
accuracy: 0.7295 - val_loss: 0.5268 - val_accuracy: 0.7367
Epoch 98/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5289 -
accuracy: 0.7322 - val_loss: 0.5254 - val_accuracy: 0.7381
Epoch 99/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5247 -
accuracy: 0.7342 - val_loss: 0.5254 - val_accuracy: 0.7383
Epoch 100/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.5256 -
accuracy: 0.7295 - val_loss: 0.5233 - val_accuracy: 0.7319
Epoch 101/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5282 -
accuracy: 0.7283 - val_loss: 0.5254 - val_accuracy: 0.7354
Epoch 102/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5282 -
accuracy: 0.7284 - val_loss: 0.5260 - val_accuracy: 0.7346
Epoch 103/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5268 -
accuracy: 0.7310 - val_loss: 0.5238 - val_accuracy: 0.7392
Epoch 104/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5196 -
accuracy: 0.7347 - val_loss: 0.5235 - val_accuracy: 0.7369
Epoch 105/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5202 -
accuracy: 0.7372 - val_loss: 0.5198 - val_accuracy: 0.7354
Epoch 106/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5220 -
accuracy: 0.7367 - val_loss: 0.5227 - val_accuracy: 0.7396
Epoch 107/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5163 -
accuracy: 0.7373 - val_loss: 0.5197 - val_accuracy: 0.7394
```

```
Epoch 108/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5210 -
accuracy: 0.7357 - val_loss: 0.5214 - val_accuracy: 0.7423
Epoch 109/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5255 -
accuracy: 0.7304 - val_loss: 0.5197 - val_accuracy: 0.7437
Epoch 110/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5182 -
accuracy: 0.7371 - val_loss: 0.5189 - val_accuracy: 0.7415
Epoch 111/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5134 -
accuracy: 0.7409 - val_loss: 0.5176 - val_accuracy: 0.7398
Epoch 112/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5156 -
accuracy: 0.7393 - val_loss: 0.5173 - val_accuracy: 0.7435
Epoch 113/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5150 -
accuracy: 0.7388 - val_loss: 0.5154 - val_accuracy: 0.7431
Epoch 114/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5181 -
accuracy: 0.7413 - val_loss: 0.5159 - val_accuracy: 0.7460
Epoch 115/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5086 -
accuracy: 0.7472 - val_loss: 0.5114 - val_accuracy: 0.7470
Epoch 116/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5147 -
accuracy: 0.7383 - val_loss: 0.5170 - val_accuracy: 0.7415
Epoch 117/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5156 -
accuracy: 0.7402 - val_loss: 0.5152 - val_accuracy: 0.7446
Epoch 118/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5106 -
accuracy: 0.7427 - val_loss: 0.5103 - val_accuracy: 0.7508
Epoch 119/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5135 -
accuracy: 0.7426 - val_loss: 0.5134 - val_accuracy: 0.7499
Epoch 120/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5094 -
accuracy: 0.7454 - val_loss: 0.5084 - val_accuracy: 0.7514
Epoch 121/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.5025 -
accuracy: 0.7512 - val_loss: 0.5065 - val_accuracy: 0.7487
Epoch 122/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.5071 -
accuracy: 0.7389 - val_loss: 0.5063 - val_accuracy: 0.7539
Epoch 123/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5091 -
accuracy: 0.7387 - val_loss: 0.5076 - val_accuracy: 0.7473
```

```
Epoch 124/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5118 -
accuracy: 0.7413 - val_loss: 0.5055 - val_accuracy: 0.7493
Epoch 125/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5104 -
accuracy: 0.7413 - val_loss: 0.5026 - val_accuracy: 0.7462
Epoch 126/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5065 -
accuracy: 0.7455 - val_loss: 0.5075 - val_accuracy: 0.7446
Epoch 127/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5079 -
accuracy: 0.7454 - val_loss: 0.5066 - val_accuracy: 0.7504
Epoch 128/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5031 -
accuracy: 0.7515 - val_loss: 0.5025 - val_accuracy: 0.7551
Epoch 129/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5023 -
accuracy: 0.7437 - val_loss: 0.5006 - val_accuracy: 0.7502
Epoch 130/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4950 -
accuracy: 0.7497 - val_loss: 0.5025 - val_accuracy: 0.7535
Epoch 131/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5046 -
accuracy: 0.7486 - val_loss: 0.4974 - val_accuracy: 0.7531
Epoch 132/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5055 -
accuracy: 0.7476 - val_loss: 0.4993 - val_accuracy: 0.7508
Epoch 133/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.5038 -
accuracy: 0.7493 - val_loss: 0.4994 - val_accuracy: 0.7576
Epoch 134/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5085 -
accuracy: 0.7431 - val_loss: 0.4974 - val_accuracy: 0.7570
Epoch 135/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5043 -
accuracy: 0.7494 - val_loss: 0.5027 - val_accuracy: 0.7545
Epoch 136/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4985 -
accuracy: 0.7533 - val_loss: 0.4960 - val_accuracy: 0.7545
Epoch 137/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.5002 -
accuracy: 0.7494 - val_loss: 0.4988 - val_accuracy: 0.7533
Epoch 138/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4975 -
accuracy: 0.7501 - val_loss: 0.4965 - val_accuracy: 0.7520
Epoch 139/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4935 -
accuracy: 0.7588 - val_loss: 0.4985 - val_accuracy: 0.7526
```

```
Epoch 140/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4992 -
accuracy: 0.7488 - val_loss: 0.4996 - val_accuracy: 0.7522
Epoch 141/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4970 -
accuracy: 0.7487 - val_loss: 0.4932 - val_accuracy: 0.7553
Epoch 142/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4938 -
accuracy: 0.7556 - val_loss: 0.4928 - val_accuracy: 0.7568
Epoch 143/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4983 -
accuracy: 0.7557 - val_loss: 0.4906 - val_accuracy: 0.7557
Epoch 144/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4999 -
accuracy: 0.7519 - val_loss: 0.4944 - val_accuracy: 0.7601
Epoch 145/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4919 -
accuracy: 0.7554 - val_loss: 0.4908 - val_accuracy: 0.7593
Epoch 146/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4975 -
accuracy: 0.7515 - val_loss: 0.4902 - val_accuracy: 0.7636
Epoch 147/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4961 -
accuracy: 0.7509 - val_loss: 0.4926 - val_accuracy: 0.7597
Epoch 148/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4964 -
accuracy: 0.7534 - val_loss: 0.4885 - val_accuracy: 0.7591
Epoch 149/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4873 -
accuracy: 0.7613 - val_loss: 0.4896 - val_accuracy: 0.7578
Epoch 150/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4948 -
accuracy: 0.7510 - val_loss: 0.4911 - val_accuracy: 0.7562
Epoch 151/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4953 -
accuracy: 0.7499 - val_loss: 0.4879 - val_accuracy: 0.7618
Epoch 152/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4948 -
accuracy: 0.7554 - val_loss: 0.4886 - val_accuracy: 0.7568
Epoch 153/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4962 -
accuracy: 0.7525 - val_loss: 0.4868 - val_accuracy: 0.7613
Epoch 154/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4918 -
accuracy: 0.7550 - val_loss: 0.4867 - val_accuracy: 0.7599
Epoch 155/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4931 -
accuracy: 0.7538 - val_loss: 0.4917 - val_accuracy: 0.7634
```

```
Epoch 156/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4849 -
accuracy: 0.7602 - val_loss: 0.4855 - val_accuracy: 0.7649
Epoch 157/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4846 -
accuracy: 0.7611 - val_loss: 0.4854 - val_accuracy: 0.7601
Epoch 158/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4901 -
accuracy: 0.7594 - val_loss: 0.4860 - val_accuracy: 0.7640
Epoch 159/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4844 -
accuracy: 0.7603 - val_loss: 0.4892 - val_accuracy: 0.7613
Epoch 160/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4844 -
accuracy: 0.7620 - val_loss: 0.4826 - val_accuracy: 0.7727
Epoch 161/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4862 -
accuracy: 0.7581 - val_loss: 0.4836 - val_accuracy: 0.7684
Epoch 162/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4802 -
accuracy: 0.7642 - val_loss: 0.4831 - val_accuracy: 0.7686
Epoch 163/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4859 -
accuracy: 0.7618 - val_loss: 0.4834 - val_accuracy: 0.7622
Epoch 164/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4860 -
accuracy: 0.7584 - val_loss: 0.4829 - val_accuracy: 0.7678
Epoch 165/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4836 -
accuracy: 0.7606 - val_loss: 0.4818 - val_accuracy: 0.7707
Epoch 166/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4868 -
accuracy: 0.7567 - val_loss: 0.4780 - val_accuracy: 0.7736
Epoch 167/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4832 -
accuracy: 0.7652 - val_loss: 0.4800 - val_accuracy: 0.7719
Epoch 168/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4840 -
accuracy: 0.7589 - val_loss: 0.4813 - val_accuracy: 0.7711
Epoch 169/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4823 -
accuracy: 0.7603 - val_loss: 0.4782 - val_accuracy: 0.7700
Epoch 170/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4831 -
accuracy: 0.7605 - val_loss: 0.4723 - val_accuracy: 0.7717
Epoch 171/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4877 -
accuracy: 0.7595 - val_loss: 0.4809 - val_accuracy: 0.7657
```

```
Epoch 172/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4828 -
accuracy: 0.7651 - val_loss: 0.4826 - val_accuracy: 0.7655
Epoch 173/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4812 -
accuracy: 0.7585 - val_loss: 0.4789 - val_accuracy: 0.7707
Epoch 174/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4777 -
accuracy: 0.7619 - val_loss: 0.4781 - val_accuracy: 0.7719
Epoch 175/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4775 -
accuracy: 0.7646 - val_loss: 0.4771 - val_accuracy: 0.7740
Epoch 176/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4861 -
accuracy: 0.7609 - val_loss: 0.4817 - val_accuracy: 0.7721
Epoch 177/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4790 -
accuracy: 0.7621 - val_loss: 0.4778 - val_accuracy: 0.7703
Epoch 178/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4809 -
accuracy: 0.7589 - val_loss: 0.4735 - val_accuracy: 0.7758
Epoch 179/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4799 -
accuracy: 0.7643 - val_loss: 0.4778 - val_accuracy: 0.7744
Epoch 180/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4786 -
accuracy: 0.7589 - val_loss: 0.4740 - val_accuracy: 0.7707
Epoch 181/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4769 -
accuracy: 0.7650 - val_loss: 0.4750 - val_accuracy: 0.7758
Epoch 182/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4736 -
accuracy: 0.7684 - val_loss: 0.4718 - val_accuracy: 0.7754
Epoch 183/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4771 -
accuracy: 0.7652 - val_loss: 0.4749 - val_accuracy: 0.7694
Epoch 184/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4816 -
accuracy: 0.7675 - val_loss: 0.4723 - val_accuracy: 0.7740
Epoch 185/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4748 -
accuracy: 0.7682 - val_loss: 0.4723 - val_accuracy: 0.7721
Epoch 186/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4707 -
accuracy: 0.7694 - val_loss: 0.4711 - val_accuracy: 0.7698
Epoch 187/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4785 -
accuracy: 0.7678 - val_loss: 0.4721 - val_accuracy: 0.7711
```

```
Epoch 188/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4746 -
accuracy: 0.7682 - val_loss: 0.4696 - val_accuracy: 0.7719
Epoch 189/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4788 -
accuracy: 0.7678 - val_loss: 0.4688 - val_accuracy: 0.7769
Epoch 190/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4721 -
accuracy: 0.7728 - val_loss: 0.4723 - val_accuracy: 0.7769
Epoch 191/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4712 -
accuracy: 0.7719 - val_loss: 0.4663 - val_accuracy: 0.7738
Epoch 192/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4724 -
accuracy: 0.7675 - val_loss: 0.4676 - val_accuracy: 0.7752
Epoch 193/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4675 -
accuracy: 0.7719 - val_loss: 0.4676 - val_accuracy: 0.7773
Epoch 194/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4675 -
accuracy: 0.7727 - val_loss: 0.4643 - val_accuracy: 0.7775
Epoch 195/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4655 -
accuracy: 0.7767 - val_loss: 0.4648 - val_accuracy: 0.7794
Epoch 196/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4735 -
accuracy: 0.7707 - val_loss: 0.4661 - val_accuracy: 0.7816
Epoch 197/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4636 -
accuracy: 0.7750 - val_loss: 0.4627 - val_accuracy: 0.7823
Epoch 198/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4748 -
accuracy: 0.7637 - val_loss: 0.4632 - val_accuracy: 0.7819
Epoch 199/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4648 -
accuracy: 0.7721 - val_loss: 0.4656 - val_accuracy: 0.7814
Epoch 200/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4705 -
accuracy: 0.7691 - val_loss: 0.4607 - val_accuracy: 0.7796
Epoch 201/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4682 -
accuracy: 0.7741 - val_loss: 0.4622 - val_accuracy: 0.7781
Epoch 202/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4740 -
accuracy: 0.7718 - val_loss: 0.4596 - val_accuracy: 0.7831
Epoch 203/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4659 -
accuracy: 0.7733 - val_loss: 0.4607 - val_accuracy: 0.7773
```

```
Epoch 204/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4644 -
accuracy: 0.7728 - val_loss: 0.4600 - val_accuracy: 0.7812
Epoch 205/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4657 -
accuracy: 0.7750 - val_loss: 0.4645 - val_accuracy: 0.7819
Epoch 206/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4657 -
accuracy: 0.7743 - val_loss: 0.4670 - val_accuracy: 0.7790
Epoch 207/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4646 -
accuracy: 0.7705 - val_loss: 0.4641 - val_accuracy: 0.7843
Epoch 208/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4626 -
accuracy: 0.7781 - val_loss: 0.4622 - val_accuracy: 0.7850
Epoch 209/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4701 -
accuracy: 0.7703 - val_loss: 0.4612 - val_accuracy: 0.7781
Epoch 210/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4629 -
accuracy: 0.7775 - val_loss: 0.4581 - val_accuracy: 0.7833
Epoch 211/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4657 -
accuracy: 0.7757 - val_loss: 0.4604 - val_accuracy: 0.7872
Epoch 212/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4623 -
accuracy: 0.7764 - val_loss: 0.4598 - val_accuracy: 0.7829
Epoch 213/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4695 -
accuracy: 0.7719 - val_loss: 0.4592 - val_accuracy: 0.7841
Epoch 214/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4560 -
accuracy: 0.7779 - val_loss: 0.4593 - val_accuracy: 0.7831
Epoch 215/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4657 -
accuracy: 0.7728 - val_loss: 0.4512 - val_accuracy: 0.7916
Epoch 216/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4640 -
accuracy: 0.7740 - val_loss: 0.4551 - val_accuracy: 0.7858
Epoch 217/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4613 -
accuracy: 0.7748 - val_loss: 0.4542 - val_accuracy: 0.7926
Epoch 218/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4602 -
accuracy: 0.7768 - val_loss: 0.4566 - val_accuracy: 0.7885
Epoch 219/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4676 -
accuracy: 0.7685 - val_loss: 0.4563 - val_accuracy: 0.7908
```

```
Epoch 220/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4611 -
accuracy: 0.7771 - val_loss: 0.4516 - val_accuracy: 0.7889
Epoch 221/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4631 -
accuracy: 0.7780 - val_loss: 0.4566 - val_accuracy: 0.7841
Epoch 222/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4650 -
accuracy: 0.7742 - val_loss: 0.4531 - val_accuracy: 0.7845
Epoch 223/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4650 -
accuracy: 0.7735 - val_loss: 0.4608 - val_accuracy: 0.7833
Epoch 224/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4635 -
accuracy: 0.7750 - val_loss: 0.4588 - val_accuracy: 0.7858
Epoch 225/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4630 -
accuracy: 0.7746 - val_loss: 0.4561 - val_accuracy: 0.7901
Epoch 226/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4611 -
accuracy: 0.7718 - val_loss: 0.4592 - val_accuracy: 0.7916
Epoch 227/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4616 -
accuracy: 0.7812 - val_loss: 0.4553 - val_accuracy: 0.7874
Epoch 228/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4655 -
accuracy: 0.7741 - val_loss: 0.4537 - val_accuracy: 0.7935
Epoch 229/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4615 -
accuracy: 0.7760 - val_loss: 0.4547 - val_accuracy: 0.7883
Epoch 230/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4637 -
accuracy: 0.7771 - val_loss: 0.4549 - val_accuracy: 0.7883
Epoch 231/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4570 -
accuracy: 0.7789 - val_loss: 0.4542 - val_accuracy: 0.7914
Epoch 232/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4549 -
accuracy: 0.7785 - val_loss: 0.4514 - val_accuracy: 0.7901
Epoch 233/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4572 -
accuracy: 0.7783 - val_loss: 0.4516 - val_accuracy: 0.7932
Epoch 234/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4572 -
accuracy: 0.7795 - val_loss: 0.4516 - val_accuracy: 0.7910
Epoch 235/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4498 -
accuracy: 0.7821 - val_loss: 0.4490 - val_accuracy: 0.7945
```

```
Epoch 236/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4562 -
accuracy: 0.7792 - val_loss: 0.4518 - val_accuracy: 0.7945
Epoch 237/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4524 -
accuracy: 0.7866 - val_loss: 0.4512 - val_accuracy: 0.7926
Epoch 238/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4619 -
accuracy: 0.7795 - val_loss: 0.4504 - val_accuracy: 0.7949
Epoch 239/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4547 -
accuracy: 0.7820 - val_loss: 0.4536 - val_accuracy: 0.7895
Epoch 240/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4588 -
accuracy: 0.7792 - val_loss: 0.4532 - val_accuracy: 0.7930
Epoch 241/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4552 -
accuracy: 0.7821 - val_loss: 0.4461 - val_accuracy: 0.7955
Epoch 242/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4567 -
accuracy: 0.7806 - val_loss: 0.4478 - val_accuracy: 0.7947
Epoch 243/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4523 -
accuracy: 0.7835 - val_loss: 0.4504 - val_accuracy: 0.7920
Epoch 244/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4584 -
accuracy: 0.7835 - val_loss: 0.4499 - val_accuracy: 0.7897
Epoch 245/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4506 -
accuracy: 0.7810 - val_loss: 0.4452 - val_accuracy: 0.7912
Epoch 246/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4562 -
accuracy: 0.7841 - val_loss: 0.4476 - val_accuracy: 0.7910
Epoch 247/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4576 -
accuracy: 0.7824 - val_loss: 0.4495 - val_accuracy: 0.7970
Epoch 248/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4533 -
accuracy: 0.7827 - val_loss: 0.4466 - val_accuracy: 0.7920
Epoch 249/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4481 -
accuracy: 0.7867 - val_loss: 0.4456 - val_accuracy: 0.7910
Epoch 250/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4435 -
accuracy: 0.7906 - val_loss: 0.4438 - val_accuracy: 0.7949
Epoch 251/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4517 -
accuracy: 0.7843 - val_loss: 0.4436 - val_accuracy: 0.7968
```

```
Epoch 252/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4444 -
accuracy: 0.7880 - val_loss: 0.4469 - val_accuracy: 0.7930
Epoch 253/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4563 -
accuracy: 0.7783 - val_loss: 0.4422 - val_accuracy: 0.7982
Epoch 254/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4501 -
accuracy: 0.7833 - val_loss: 0.4435 - val_accuracy: 0.7932
Epoch 255/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4488 -
accuracy: 0.7854 - val_loss: 0.4457 - val_accuracy: 0.7974
Epoch 256/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4565 -
accuracy: 0.7780 - val_loss: 0.4446 - val_accuracy: 0.7978
Epoch 257/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4478 -
accuracy: 0.7850 - val_loss: 0.4417 - val_accuracy: 0.7941
Epoch 258/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4517 -
accuracy: 0.7834 - val_loss: 0.4435 - val_accuracy: 0.7961
Epoch 259/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4474 -
accuracy: 0.7877 - val_loss: 0.4426 - val_accuracy: 0.7961
Epoch 260/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4504 -
accuracy: 0.7873 - val_loss: 0.4386 - val_accuracy: 0.7988
Epoch 261/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4439 -
accuracy: 0.7868 - val_loss: 0.4373 - val_accuracy: 0.8017
Epoch 262/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4486 -
accuracy: 0.7825 - val_loss: 0.4376 - val_accuracy: 0.7990
Epoch 263/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4460 -
accuracy: 0.7872 - val_loss: 0.4423 - val_accuracy: 0.7961
Epoch 264/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4406 -
accuracy: 0.7934 - val_loss: 0.4405 - val_accuracy: 0.7945
Epoch 265/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4470 -
accuracy: 0.7894 - val_loss: 0.4409 - val_accuracy: 0.7978
Epoch 266/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4491 -
accuracy: 0.7864 - val_loss: 0.4454 - val_accuracy: 0.8032
Epoch 267/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4460 -
accuracy: 0.7892 - val_loss: 0.4406 - val_accuracy: 0.7943
```

```
Epoch 268/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4480 -
accuracy: 0.7861 - val_loss: 0.4408 - val_accuracy: 0.7955
Epoch 269/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4491 -
accuracy: 0.7868 - val_loss: 0.4358 - val_accuracy: 0.8048
Epoch 270/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4466 -
accuracy: 0.7864 - val_loss: 0.4365 - val_accuracy: 0.8009
Epoch 271/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4483 -
accuracy: 0.7818 - val_loss: 0.4359 - val_accuracy: 0.8007
Epoch 272/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4451 -
accuracy: 0.7905 - val_loss: 0.4373 - val_accuracy: 0.8007
Epoch 273/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4497 -
accuracy: 0.7873 - val_loss: 0.4349 - val_accuracy: 0.8019
Epoch 274/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4468 -
accuracy: 0.7875 - val_loss: 0.4353 - val_accuracy: 0.7982
Epoch 275/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4462 -
accuracy: 0.7890 - val_loss: 0.4367 - val_accuracy: 0.7978
Epoch 276/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4482 -
accuracy: 0.7876 - val_loss: 0.4386 - val_accuracy: 0.7970
Epoch 277/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4499 -
accuracy: 0.7839 - val_loss: 0.4401 - val_accuracy: 0.8011
Epoch 278/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4419 -
accuracy: 0.7925 - val_loss: 0.4420 - val_accuracy: 0.7988
Epoch 279/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4474 -
accuracy: 0.7867 - val_loss: 0.4370 - val_accuracy: 0.8034
Epoch 280/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4508 -
accuracy: 0.7862 - val_loss: 0.4419 - val_accuracy: 0.7982
Epoch 281/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4435 -
accuracy: 0.7908 - val_loss: 0.4386 - val_accuracy: 0.8015
Epoch 282/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4502 -
accuracy: 0.7874 - val_loss: 0.4358 - val_accuracy: 0.8055
Epoch 283/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4474 -
accuracy: 0.7858 - val_loss: 0.4410 - val_accuracy: 0.8032
```

```
Epoch 284/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4485 -
accuracy: 0.7850 - val_loss: 0.4342 - val_accuracy: 0.8100
Epoch 285/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4361 -
accuracy: 0.7964 - val_loss: 0.4402 - val_accuracy: 0.8007
Epoch 286/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4394 -
accuracy: 0.7930 - val_loss: 0.4365 - val_accuracy: 0.8053
Epoch 287/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4466 -
accuracy: 0.7829 - val_loss: 0.4378 - val_accuracy: 0.8003
Epoch 288/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4409 -
accuracy: 0.7916 - val_loss: 0.4370 - val_accuracy: 0.8042
Epoch 289/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4438 -
accuracy: 0.7867 - val_loss: 0.4352 - val_accuracy: 0.7978
Epoch 290/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4481 -
accuracy: 0.7897 - val_loss: 0.4337 - val_accuracy: 0.8073
Epoch 291/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4367 -
accuracy: 0.7977 - val_loss: 0.4346 - val_accuracy: 0.8030
Epoch 292/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4396 -
accuracy: 0.7889 - val_loss: 0.4296 - val_accuracy: 0.8042
Epoch 293/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4417 -
accuracy: 0.7905 - val_loss: 0.4307 - val_accuracy: 0.8077
Epoch 294/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4354 -
accuracy: 0.7939 - val_loss: 0.4344 - val_accuracy: 0.8046
Epoch 295/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4465 -
accuracy: 0.7842 - val_loss: 0.4338 - val_accuracy: 0.8065
Epoch 296/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4399 -
accuracy: 0.7900 - val_loss: 0.4293 - val_accuracy: 0.8075
Epoch 297/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4362 -
accuracy: 0.7948 - val_loss: 0.4313 - val_accuracy: 0.8057
Epoch 298/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4372 -
accuracy: 0.7920 - val_loss: 0.4328 - val_accuracy: 0.7999
Epoch 299/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4413 -
accuracy: 0.7898 - val_loss: 0.4345 - val_accuracy: 0.8055
```

```
Epoch 300/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4394 -
accuracy: 0.7918 - val_loss: 0.4284 - val_accuracy: 0.8104
Epoch 301/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4310 -
accuracy: 0.7989 - val_loss: 0.4305 - val_accuracy: 0.8036
Epoch 302/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4388 -
accuracy: 0.7923 - val_loss: 0.4292 - val_accuracy: 0.8102
Epoch 303/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4427 -
accuracy: 0.7917 - val_loss: 0.4320 - val_accuracy: 0.8057
Epoch 304/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4372 -
accuracy: 0.7949 - val_loss: 0.4310 - val_accuracy: 0.8051
Epoch 305/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4425 -
accuracy: 0.7897 - val_loss: 0.4313 - val_accuracy: 0.8088
Epoch 306/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4250 -
accuracy: 0.7967 - val_loss: 0.4316 - val_accuracy: 0.7997
Epoch 307/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4382 -
accuracy: 0.7865 - val_loss: 0.4325 - val_accuracy: 0.8046
Epoch 308/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4411 -
accuracy: 0.7890 - val_loss: 0.4326 - val_accuracy: 0.8092
Epoch 309/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4423 -
accuracy: 0.7890 - val_loss: 0.4303 - val_accuracy: 0.8042
Epoch 310/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4346 -
accuracy: 0.7977 - val_loss: 0.4267 - val_accuracy: 0.8121
Epoch 311/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4316 -
accuracy: 0.7982 - val_loss: 0.4286 - val_accuracy: 0.8036
Epoch 312/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4368 -
accuracy: 0.7942 - val_loss: 0.4310 - val_accuracy: 0.8086
Epoch 313/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4342 -
accuracy: 0.7967 - val_loss: 0.4260 - val_accuracy: 0.8142
Epoch 314/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4355 -
accuracy: 0.7920 - val_loss: 0.4253 - val_accuracy: 0.8125
Epoch 315/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4346 -
accuracy: 0.7971 - val_loss: 0.4260 - val_accuracy: 0.8094
```

```
Epoch 316/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4324 -
accuracy: 0.7941 - val_loss: 0.4259 - val_accuracy: 0.8084
Epoch 317/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4323 -
accuracy: 0.7969 - val_loss: 0.4223 - val_accuracy: 0.8090
Epoch 318/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4296 -
accuracy: 0.7984 - val_loss: 0.4226 - val_accuracy: 0.8104
Epoch 319/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4333 -
accuracy: 0.7969 - val_loss: 0.4209 - val_accuracy: 0.8150
Epoch 320/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4296 -
accuracy: 0.8035 - val_loss: 0.4275 - val_accuracy: 0.8090
Epoch 321/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4365 -
accuracy: 0.7938 - val_loss: 0.4257 - val_accuracy: 0.8125
Epoch 322/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4357 -
accuracy: 0.7953 - val_loss: 0.4219 - val_accuracy: 0.8121
Epoch 323/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4362 -
accuracy: 0.7959 - val_loss: 0.4248 - val_accuracy: 0.8113
Epoch 324/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4319 -
accuracy: 0.7936 - val_loss: 0.4249 - val_accuracy: 0.8109
Epoch 325/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4356 -
accuracy: 0.7972 - val_loss: 0.4233 - val_accuracy: 0.8121
Epoch 326/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4302 -
accuracy: 0.7986 - val_loss: 0.4236 - val_accuracy: 0.8150
Epoch 327/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4271 -
accuracy: 0.7992 - val_loss: 0.4187 - val_accuracy: 0.8175
Epoch 328/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4274 -
accuracy: 0.7997 - val_loss: 0.4153 - val_accuracy: 0.8185
Epoch 329/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4267 -
accuracy: 0.7984 - val_loss: 0.4163 - val_accuracy: 0.8160
Epoch 330/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4387 -
accuracy: 0.7950 - val_loss: 0.4144 - val_accuracy: 0.8175
Epoch 331/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4295 -
accuracy: 0.8007 - val_loss: 0.4176 - val_accuracy: 0.8150
```

```
Epoch 332/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4319 -
accuracy: 0.8009 - val_loss: 0.4194 - val_accuracy: 0.8125
Epoch 333/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4377 -
accuracy: 0.7929 - val_loss: 0.4214 - val_accuracy: 0.8106
Epoch 334/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4326 -
accuracy: 0.7945 - val_loss: 0.4246 - val_accuracy: 0.8119
Epoch 335/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4298 -
accuracy: 0.7995 - val_loss: 0.4142 - val_accuracy: 0.8125
Epoch 336/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4260 -
accuracy: 0.7996 - val_loss: 0.4179 - val_accuracy: 0.8094
Epoch 337/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4300 -
accuracy: 0.7974 - val_loss: 0.4228 - val_accuracy: 0.8104
Epoch 338/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4358 -
accuracy: 0.7990 - val_loss: 0.4175 - val_accuracy: 0.8210
Epoch 339/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4261 -
accuracy: 0.8041 - val_loss: 0.4217 - val_accuracy: 0.8138
Epoch 340/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4297 -
accuracy: 0.7998 - val_loss: 0.4170 - val_accuracy: 0.8177
Epoch 341/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4344 -
accuracy: 0.7931 - val_loss: 0.4193 - val_accuracy: 0.8127
Epoch 342/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4271 -
accuracy: 0.8015 - val_loss: 0.4193 - val_accuracy: 0.8129
Epoch 343/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4298 -
accuracy: 0.7953 - val_loss: 0.4181 - val_accuracy: 0.8175
Epoch 344/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4353 -
accuracy: 0.7934 - val_loss: 0.4179 - val_accuracy: 0.8144
Epoch 345/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4325 -
accuracy: 0.7984 - val_loss: 0.4146 - val_accuracy: 0.8181
Epoch 346/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4248 -
accuracy: 0.8017 - val_loss: 0.4179 - val_accuracy: 0.8200
Epoch 347/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4279 -
accuracy: 0.8009 - val_loss: 0.4187 - val_accuracy: 0.8196
```

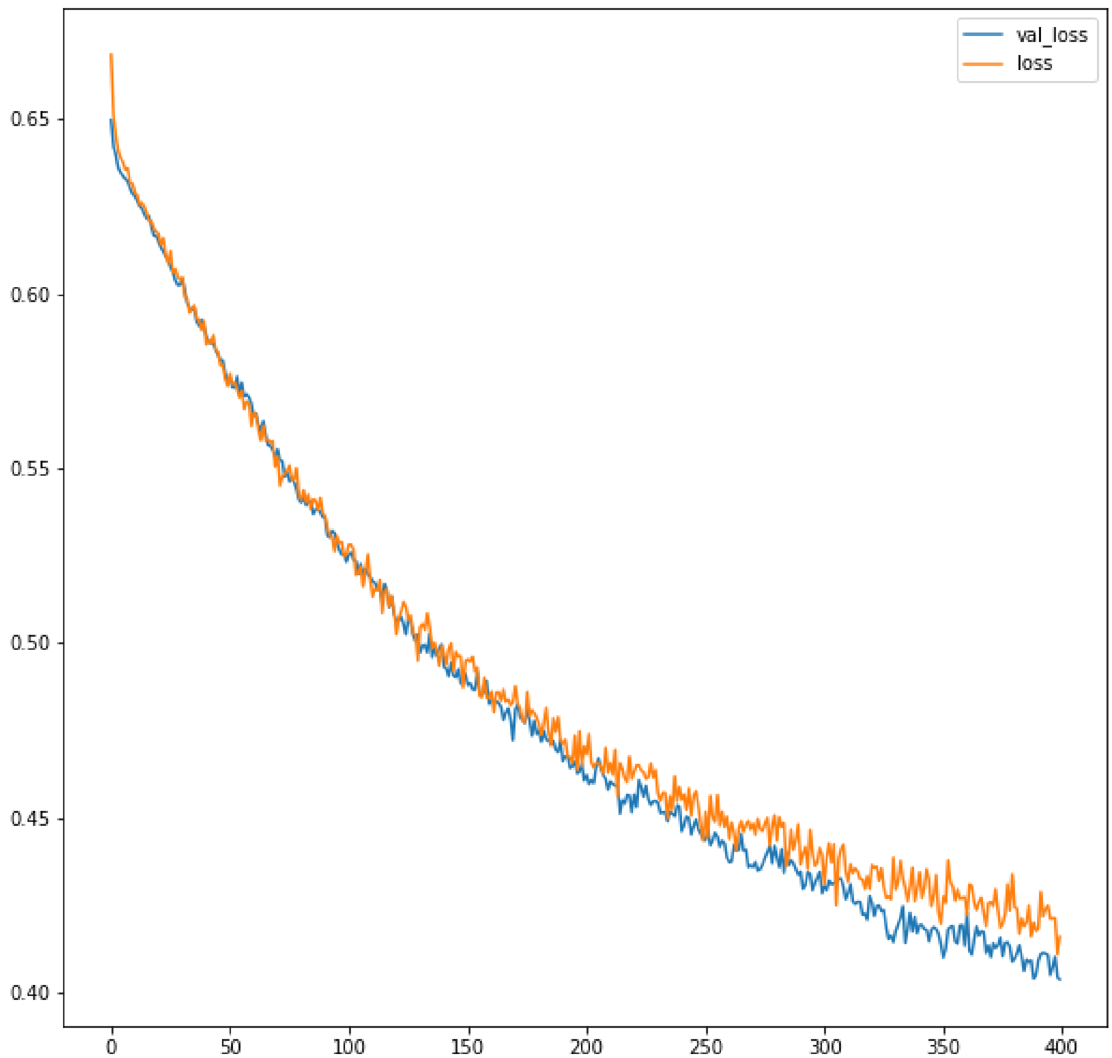
```
Epoch 348/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4336 -
accuracy: 0.7953 - val_loss: 0.4182 - val_accuracy: 0.8160
Epoch 349/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4231 -
accuracy: 0.8010 - val_loss: 0.4168 - val_accuracy: 0.8187
Epoch 350/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4197 -
accuracy: 0.8032 - val_loss: 0.4137 - val_accuracy: 0.8189
Epoch 351/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4274 -
accuracy: 0.7995 - val_loss: 0.4100 - val_accuracy: 0.8239
Epoch 352/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4257 -
accuracy: 0.7989 - val_loss: 0.4125 - val_accuracy: 0.8214
Epoch 353/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4378 -
accuracy: 0.7924 - val_loss: 0.4181 - val_accuracy: 0.8162
Epoch 354/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4316 -
accuracy: 0.7999 - val_loss: 0.4188 - val_accuracy: 0.8158
Epoch 355/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4304 -
accuracy: 0.7988 - val_loss: 0.4190 - val_accuracy: 0.8175
Epoch 356/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4263 -
accuracy: 0.7993 - val_loss: 0.4146 - val_accuracy: 0.8220
Epoch 357/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4295 -
accuracy: 0.7997 - val_loss: 0.4140 - val_accuracy: 0.8218
Epoch 358/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4268 -
accuracy: 0.7994 - val_loss: 0.4191 - val_accuracy: 0.8160
Epoch 359/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4271 -
accuracy: 0.7968 - val_loss: 0.4192 - val_accuracy: 0.8154
Epoch 360/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4272 -
accuracy: 0.8009 - val_loss: 0.4134 - val_accuracy: 0.8204
Epoch 361/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4221 -
accuracy: 0.8062 - val_loss: 0.4228 - val_accuracy: 0.8133
Epoch 362/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4309 -
accuracy: 0.7970 - val_loss: 0.4119 - val_accuracy: 0.8212
Epoch 363/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4305 -
accuracy: 0.7991 - val_loss: 0.4108 - val_accuracy: 0.8216
```

```
Epoch 364/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4258 -
accuracy: 0.8011 - val_loss: 0.4152 - val_accuracy: 0.8214
Epoch 365/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4236 -
accuracy: 0.8036 - val_loss: 0.4188 - val_accuracy: 0.8187
Epoch 366/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4258 -
accuracy: 0.7970 - val_loss: 0.4163 - val_accuracy: 0.8239
Epoch 367/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4271 -
accuracy: 0.8000 - val_loss: 0.4175 - val_accuracy: 0.8148
Epoch 368/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4227 -
accuracy: 0.8042 - val_loss: 0.4172 - val_accuracy: 0.8179
Epoch 369/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4189 -
accuracy: 0.8014 - val_loss: 0.4112 - val_accuracy: 0.8183
Epoch 370/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4255 -
accuracy: 0.7979 - val_loss: 0.4135 - val_accuracy: 0.8177
Epoch 371/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4259 -
accuracy: 0.8030 - val_loss: 0.4102 - val_accuracy: 0.8206
Epoch 372/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4247 -
accuracy: 0.8003 - val_loss: 0.4141 - val_accuracy: 0.8169
Epoch 373/400
11263/11263 [==============================] - 0s 13us/step - loss: 0.4295 -
accuracy: 0.8020 - val_loss: 0.4127 - val_accuracy: 0.8196
Epoch 374/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4234 -
accuracy: 0.8062 - val_loss: 0.4135 - val_accuracy: 0.8179
Epoch 375/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4186 -
accuracy: 0.8076 - val_loss: 0.4158 - val_accuracy: 0.8177
Epoch 376/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4196 -
accuracy: 0.8072 - val_loss: 0.4104 - val_accuracy: 0.8202
Epoch 377/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4226 -
accuracy: 0.7970 - val_loss: 0.4141 - val_accuracy: 0.8146
Epoch 378/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4310 -
accuracy: 0.8008 - val_loss: 0.4144 - val_accuracy: 0.8187
Epoch 379/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4234 -
accuracy: 0.8049 - val_loss: 0.4131 - val_accuracy: 0.8191
```

```
Epoch 380/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4337 -
accuracy: 0.7985 - val_loss: 0.4088 - val_accuracy: 0.8204
Epoch 381/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4244 -
accuracy: 0.8022 - val_loss: 0.4097 - val_accuracy: 0.8216
Epoch 382/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4241 -
accuracy: 0.7985 - val_loss: 0.4113 - val_accuracy: 0.8235
Epoch 383/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4168 -
accuracy: 0.8062 - val_loss: 0.4135 - val_accuracy: 0.8200
Epoch 384/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4212 -
accuracy: 0.8036 - val_loss: 0.4102 - val_accuracy: 0.8206
Epoch 385/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4189 -
accuracy: 0.8099 - val_loss: 0.4062 - val_accuracy: 0.8262
Epoch 386/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4201 -
accuracy: 0.8048 - val_loss: 0.4094 - val_accuracy: 0.8220
Epoch 387/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4249 -
accuracy: 0.8023 - val_loss: 0.4087 - val_accuracy: 0.8212
Epoch 388/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4160 -
accuracy: 0.8052 - val_loss: 0.4092 - val_accuracy: 0.8193
Epoch 389/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4200 -
accuracy: 0.8054 - val_loss: 0.4040 - val_accuracy: 0.8254
Epoch 390/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4176 -
accuracy: 0.8064 - val_loss: 0.4049 - val_accuracy: 0.8239
Epoch 391/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4183 -
accuracy: 0.8064 - val_loss: 0.4092 - val_accuracy: 0.8185
Epoch 392/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4288 -
accuracy: 0.7991 - val_loss: 0.4109 - val_accuracy: 0.8196
Epoch 393/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4217 -
accuracy: 0.8029 - val_loss: 0.4114 - val_accuracy: 0.8241
Epoch 394/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4239 -
accuracy: 0.8040 - val_loss: 0.4111 - val_accuracy: 0.8241
Epoch 395/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4249 -
accuracy: 0.8026 - val_loss: 0.4107 - val_accuracy: 0.8200
```

```
Epoch 396/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4210 -
accuracy: 0.8049 - val_loss: 0.4049 - val_accuracy: 0.8229
Epoch 397/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4213 -
accuracy: 0.8019 - val_loss: 0.4072 - val_accuracy: 0.8227
Epoch 398/400
11263/11263 [==============================] - 0s 14us/step - loss: 0.4212 -
accuracy: 0.8029 - val_loss: 0.4102 - val_accuracy: 0.8167
Epoch 399/400
11263/11263 [==============================] - 0s 16us/step - loss: 0.4110 -
accuracy: 0.8091 - val_loss: 0.4043 - val_accuracy: 0.8216
Epoch 400/400
11263/11263 [==============================] - 0s 15us/step - loss: 0.4160 -
accuracy: 0.8098 - val_loss: 0.4037 - val_accuracy: 0.8187
   val_loss  val_accuracy      loss  accuracy
0  0.649788      0.626476  0.668521  0.591849
1  0.642175      0.631655  0.651196  0.617686
2  0.639743      0.633727  0.645354  0.627275
3  0.636164      0.631655  0.640819  0.631182
4  0.634715      0.638492  0.638979  0.635355
```

[64]: <AxesSubplot:>



[65]:
```
predictions = model.predict_classes(X_test_scaled)
print("With the above hyperparameter tuning I'm able to acheive {}% accuracy".
 ↪format(np.round((accuracy_score(y_test, predictions)*100), 2)))
```

With the above hyperparameter tuning I'm able to acheive 81.87% accuracy

[ ]:
```
```

[66]:
```
# 0.7663144810441268 - 128 batch size

# 0.7418686554795939 - 256 batch size

# 6 layers + 2 dropouts - 0.8537393826393205
```

```
#2 layers + 0 dropouts - 0.8844002486016159
```

[67]:
```
#With 150 Epochs - accuracy is 0.9051170499274912

# With 400 Epochs also we are not seeing any improvement in accuracy
```

Computation time for the below Stratified KFold is too high and % accuracy is very less

[68]:
```
# from sklearn.model_selection import StratifiedKFold
# kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
# cvscores = []
# y_over_sampled = pd.DataFrame(y_over_sampled, columns=['not.fully.paid'])
# for train, test in kfold.split(X_over_sampled, y_over_sampled):
#     X_K_train = X_over_sampled.loc[(train),:]
#     y_K_train = y_over_sampled.loc[(train),:]
#     X_K_test = X_over_sampled.loc[(test),:]
#     y_K_test = y_over_sampled.loc[(test),:]
#     #create model
#     model = Sequential()
#     model.add(Dense(32, input_dim=13, activation='relu'))
#     model.add(Dense(16, activation='relu'))
#     model.add(Dense(1, activation='sigmoid'))
#     # Compile model
#     model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
#     # Fit the model
#     model.fit(X_K_train, y_K_train, epochs=150, batch_size=10, verbose=0)
#     # evaluate the model
#     scores = model.evaluate(X_K_test, y_K_test, verbose=0)
#     print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
#     cvscores.append(scores[1] * 100)
```

[69]:
```
# Output:

# accuracy: 50.59%
# accuracy: 50.03%
# accuracy: 49.97%
# accuracy: 50.03%
# accuracy: 49.97%
# accuracy: 51.52%
# accuracy: 50.34%
# accuracy: 50.47%
# accuracy: 50.22%
# accuracy: 50.16%
```

[70]:
```
# print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

[71]:
```
# Output: 50.33% (+/- 0.45%)
```

[ ]:
```
```

[ ]:
```
```

[ ]:
```
```