FannieMaeCorrectInputArray

Collapse

By marty (http://cdsw2.lurie.biz/marty) — Python 2 Session (Base Image v6) — 5 hours ago for 108 minutes

Timeout

Fannie Mae failed loan prediction

Model to predict if a loan will have foreclosure costs

keras requires python 2 / get tensorflow import error see forked project debugging pairplot cast all variables to f choked on the decimal mixed with float limited the number of points in the query to 5000 - cuts way down on the

```
from __future__ import print_function
!echo $PYTHON_PATH

import os, sys
```

import path

```
from pyspark.sql import *
```

create spark sql session

```
myspark = SparkSession\
    .builder\
    .appName("LoanPredictionSparkML_LogisticRegression") \
    .getOrCreate()
sc = myspark.sparkContext
import time
print ( time.time())
```

```
1569613399.82
```

```
sc.setLogLevel("ERROR")
print ( myspark )
```

```
<pyspark.sql.session.SparkSession object at 0x7f668d851890>
```

make spark print text instead of octal

```
myspark.sql("SET spark.sql.parquet.binaryAsString=true")
```

```
DataFrame[key: string, value: string]
```

read in the data file from HDFS lots of sql cleanup, particularly removing NULL vals /user/hive/warehouse/fanniemae.db/fannie_harp_sql_normed_p

use train dataset traindata_p

```
loansdf = myspark.read.parquet ( "/user/hive/warehouse/fanniemae.db/traindata_p")
```

also read from s3 mydf = myspark.read.parquet (“s3a://impalas3a/sample_07_s3a_parquet”) print number of ro type of object

```
loansdf.cache()
```

```
loansdf.cache()
```

```
DataFrame[label: tinyint, loan_identifier: string, channel: decimal(1,1), seller_name: 
intrate: decimal(20,14), loanamt: decimal(21,11), loan2val: decimal(18,14), numborrowe
le, creditscore: double, property_state: string, origination_date: string, fcl_costs: 
(38,12)]
```

```
print ( loansdf.count() )
```

```
513866
```

```
print  ( loansdf )
```

```
DataFrame[label: tinyint, loan_identifier: string, channel: decimal(1,1), seller_name: 
intrate: decimal(20,14), loanamt: decimal(21,11), loan2val: decimal(18,14), numborrowe
le, creditscore: double, property_state: string, origination_date: string, fcl_costs: 
(38,12)]
```

```
loansdf.show(5)
```

```
+-----+---------------+-------+--------------------+---------------+-------------+---
-----+-------------------+------------------+--------------+----------------+----------
|label|loan_identifier|channel|         seller_name|        intrate|      loanamt|     
n2val|       numborrowers|       creditscore|property_state|origination_date|     fcl_
+-----+---------------+-------+--------------------+---------------+-------------+---
-----+-------------------+------------------+--------------+----------------+----------
|    1|   100004116882|    0.1|      PNC BANK, N.A.|0.50000000000000|0.06633380884|1.0
80412|0.16666666666666666|0.6247030878859857|            IL|         06/2014|0.0318460
|    0|   100006935001|    0.1|DITECH FINANCIAL LLC|0.54838709677419|0.11768901569|0.9
24742| 0.3333333333333333|0.9513064133016627|            IL|         05/2013|         
|    0|   100007666444|    0.1|WELLS FARGO BANK,...|0.69354838709677|0.29172610556|0.9
84536|0.16666666666666666| 0.828978622327791|            NY|         02/2010|         
|    0|   100015122571|    0.1|JPMORGAN CHASE BA...|0.43548387096774|0.04136947218|0.8
79381|0.16666666666666666|0.9382422802850356|            OH|         03/2013|         
|    0|   100016186842|    0.1|JPMORGAN CHASE BA...|0.54838709677419|0.06918687589|1.3
56701| 0.3333333333333333|0.9109263657957245|            IL|         01/2013|         
+-----+---------------+-------+--------------------+---------------+-------------+---
-----+-------------------+------------------+--------------+----------------+----------
only showing top 5 rows
```

create a table name to use for queries

```
loansdf.createOrReplaceTempView("loandata")
```

run a query data is already normalized foreclosure costs range from a value of 0 to 1

```
lndf=myspark.sql('select * from loandata where fcl_costs < .9 limit 500000')
lndf.show(5)
```

```
+-----+---------------+-------+--------------------+---------------+-------------+---
-----+-------------------+------------------+--------------+----------------+----------
|label|loan_identifier|channel|         seller_name|        intrate|      loanamt|     
n2val|       numborrowers|       creditscore|property_state|origination_date|     fcl_
+-----+---------------+-------+--------------------+---------------+-------------+---
-----+-------------------+------------------+--------------+----------------+----------
|    1|   100004116882|    0.1|      PNC BANK, N.A.|0.50000000000000|0.06633380884|1.0
80412|0.16666666666666666|0.6247030878859857|            IL|         06/2014|0.0318460
|    0|   100006935001|    0.1|DITECH FINANCIAL LLC|0.54838709677419|0.11768901569|0.9
```

```
24742| 0.3333333333333333|0.9513064133016627|            IL|         05/2013|         (
|    0|   100007666444|    0.1|WELLS FARGO BANK,...|0.69354838709677|0.29172610556|0.9(
84536|0.1666666666666666| 0.828978622327791|            NY|         02/2010|         (
|    0|   100015122571|    0.1|JPMORGAN CHASE BA...|0.43548387096774|0.04136947218|0.8
79381|0.1666666666666666|0.9382422802850356|            OH|         03/2013|         (
|    0|   100016186842|    0.1|JPMORGAN CHASE BA...|0.54838709677419|0.06918687589|1.3
56701| 0.3333333333333333|0.9109263657957245|            IL|         01/2013|         (
+-----+---------------+-------+--------------------+---------------+-------------+----
-----+-------------------+------------------+-------------+---------------+----------
only showing top 5 rows
```

```
lndf.count()
```

```
500000
```

pairplot to see what we have… not all cols are numeric, so we drop those for pair plot we will use string indexer l clean those up

```
import seaborn as sns
import pandas
```

debuging: convert all datatypes to float switch to python 3 note: case statement in impala will create a decimal f when a=b then 0.1 :-(

reduce total columns in the pair plot so it doesn't take too long to run

```
pplotdf1 =myspark.sql('select seller_name, float(channel),float( intrate) ,float(loanam
```

```
pplotdf1.show(3)
```

```
+--------------------+-------+---------+----------+---------+------------+-----------+
--+
|         seller_name|channel|  intrate|   loanamt| loan2val|numborrowers|creditscore|
ts|
+--------------------+-------+---------+----------+---------+------------+-----------+
--+
|      PNC BANK, N.A.|    0.1|      0.5|0.06633381|1.0824742|  0.16666667|  0.6247031|(
08|
|DITECH FINANCIAL LLC|    0.1|0.5483871|0.11768901|0.9484536|  0.33333334|  0.9513064|
0.0|
|WELLS FARGO BANK,...|    0.1|0.6935484| 0.2917261|0.9072165|  0.16666667|  0.8289786|
0.0|
+--------------------+-------+---------+----------+---------+------------+-----------+
--+
only showing top 3 rows
```

seaborn wants a pandas dataframe, not a spark dataframe so convert

```
pdsdf = pplotdf1.toPandas()
pdsdf.head()
```

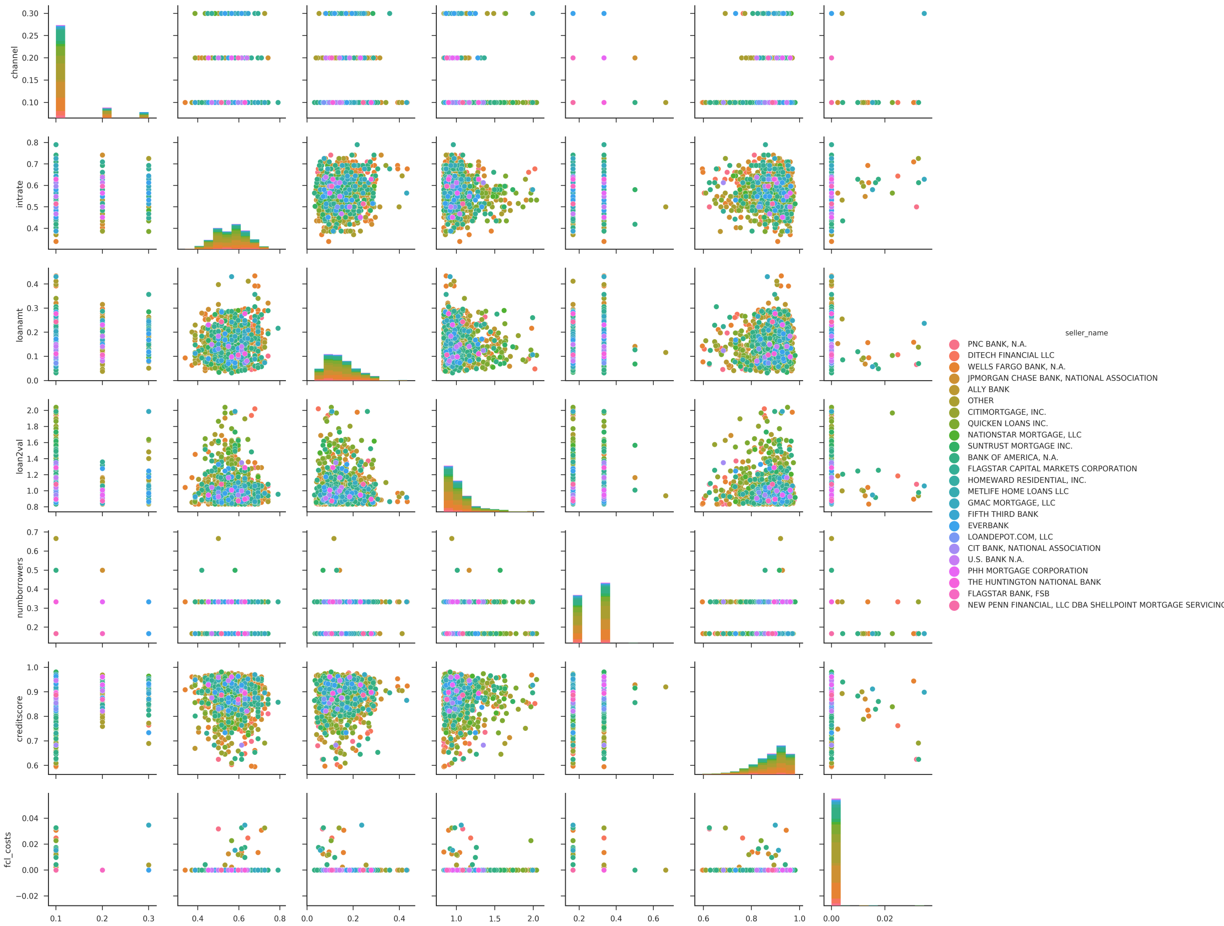
	seller_name	channel	intrate	loanamt	loan2val	numborrowers	creditscore
0	PNC BANK, N.A.	0.1	0.500000	0.066334	1.082474	0.166667	0.624703
1	DITECH FINANCIAL LLC	0.1	0.548387	0.117689	0.948454	0.333333	0.951306

2	WELLS FARGO BANK, N.A.	0.1	0.693548	0.291726	0.907216	0.166667	0.828979
3	JPMORGAN CHASE BANK, NATIONAL ASSOCIATION	0.1	0.435484	0.041369	0.876289	0.166667	0.938242
4	JPMORGAN CHASE BANK, NATIONAL ASSOCIATION	0.1	0.548387	0.069187	1.340206	0.333333	0.910926

```
sns.set(style="ticks" , color_codes=True)
```

this takes a long time to run: you can see it if you uncomment it

```
g = sns.pairplot(pdsdf,  hue="seller_name" )
```



Predict if a loan will have foreclosure costs

expand to all columns for machine learning

make the foreclosure costs binary a loan has “failed” if foreclosure > 0 in sql create column called “label” with 1

```
lndf = myspark.sql('select * from loandata ') #limit 500000')
```

lndf = myspark.sql(‘select * from loandata limit 500000’)

Impala did much of the normalization

show an example here for those who want to use pyspark

need to convert from text field to numeric this is a common requirement when using sparkML

```
from pyspark.ml.feature import StringIndexer
```

this will convert each unique string into a numeric

```
indexer = StringIndexer(inputCol="property_state", outputCol="loc_state")
indexed = indexer.fit(lndf).transform(lndf)
indexed.show(5)
```

```
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+--------------+----------------+----------
-------+
|label|loan_identifier|channel|         seller_name|         intrate|      loanamt|
n2val|       numborrowers|       creditscore|property_state|origination_date|     fcl_c
c_state|
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+--------------+----------------+----------
-------+
|    1|   100004116882|    0.1|      PNC BANK, N.A.|0.50000000000000|0.06633380884|1.08
80412|0.16666666666666666|0.6247030878859857|            IL|         06/2014|0.03184608
3.0|
|    0|   100006935001|    0.1|DITECH FINANCIAL LLC|0.54838709677419|0.11768901569|0.94
24742| 0.3333333333333333|0.9513064133016627|            IL|         05/2013|         0
3.0|
|    0|   100007666444|    0.1|WELLS FARGO BANK,...|0.69354838709677|0.29172610556|0.90
84536|0.16666666666666666| 0.828978622327791|            NY|         02/2010|         0
12.0|
|    0|   100015122571|    0.1|JPMORGAN CHASE BA...|0.43548387096774|0.04136947218|0.87
79381|0.16666666666666666|0.9382422802850356|            OH|         03/2013|         0
7.0|
|    0|   100016186842|    0.1|JPMORGAN CHASE BA...|0.54838709677419|0.06918687589|1.34
56701| 0.3333333333333333|0.9109263657957245|            IL|         01/2013|         0
3.0|
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+--------------+----------------+----------
-------+
only showing top 5 rows
```

First try a logistic regression

now we need to create a “label” and “features” input for using the sparkML library

This runs in the Cloudera Spark Cluster

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.linalg import Vectors
```

the debt to income col has nulls

```
assembler = VectorAssembler(
    inputCols=[ "intrate", "loanamt", "loan2val", \
    "numborrowers",  \
              "creditscore", "loc_state"],
    outputCol="features")
```

note the column headers - label and features are keywords

```
output = assembler.transform(indexed)
output.show(5)
```

```
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+-------------+---------------+----------
-------+--------------------+
|label|loan_identifier|channel|         seller_name|         intrate|      loanamt|
n2val|       numborrowers|       creditscore|property_state|origination_date|     fcl_c
c_state|            features|
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+-------------+---------------+----------
-------+--------------------+
|    1|   100004116882|    0.1|      PNC BANK, N.A.|0.50000000000000|0.06633380884|1.08
80412|0.16666666666666666|0.6247030878859857|           IL|         06/2014|0.03184608
3.0|[0.5,0.0663338088...|
|    0|   100006935001|    0.1|DITECH FINANCIAL LLC|0.54838709677419|0.11768901569|0.94
24742| 0.3333333333333333|0.9513064133016627|           IL|         05/2013|         6
3.0|[0.54838709677419...|
|    0|   100007666444|    0.1|WELLS FARGO BANK,...|0.69354838709677|0.29172610556|0.96
84536|0.16666666666666666| 0.828978622327791|           NY|         02/2010|         6
12.0|[0.69354838709677...|
|    0|   100015122571|    0.1|JPMORGAN CHASE BA...|0.43548387096774|0.04136947218|0.87
79381|0.16666666666666666|0.9382422802850356|           OH|         03/2013|         6
7.0|[0.43548387096774...|
|    0|   100016186842|    0.1|JPMORGAN CHASE BA...|0.54838709677419|0.06918687589|1.34
56701| 0.3333333333333333|0.9109263657957245|           IL|         01/2013|         6
3.0|[0.54838709677419...|
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+-------------+---------------+----------
-------+--------------------+
only showing top 5 rows
```

```
output.count()
```

```
513866
```

```
from pyspark.ml.classification import LogisticRegression
```

Create a LogisticRegression instance. This instance is an Estimator.

```
lr = LogisticRegression(maxIter=10, regParam=0.3)
```

Print out the parameters, documentation, and any default values.

```
print("LogisticRegression parameters:\n" + lr.explainParams() + "\n")
```

```
LogisticRegression parameters:
aggregationDepth: suggested depth for treeAggregate (>= 2). (default: 2)
elasticNetParam: the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the p
is an L2 penalty. For alpha = 1, it is an L1 penalty. (default: 0.0)
family: The name of family which is a description of the label distribution to be used
model. Supported options: auto, binomial, multinomial (default: auto)
featuresCol: features column name. (default: features)
fitIntercept: whether to fit an intercept term. (default: True)
labelCol: label column name. (default: label)
lowerBoundsOnCoefficients: The lower bounds on coefficients if fitting under bound cons
```

```
optimization. The bound matrix must be compatible with the shape (1, number of feature
inomial regression, or (number of classes, number of features) for multinomial regress
defined)
lowerBoundsOnIntercepts: The lower bounds on intercepts if fitting under bound constra
imization. The bounds vector size must beequal with 1 for binomial regression, or the 
flasses for multinomial regression. (undefined)
maxIter: max number of iterations (>= 0). (default: 100, current: 10)
predictionCol: prediction column name. (default: prediction)
probabilityCol: Column name for predicted class conditional probabilities. Note: Not a
s output well-calibrated probability estimates! These probabilities should be treated 
dences, not precise probabilities. (default: probability)
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPredict
regParam: regularization parameter (>= 0). (default: 0.0, current: 0.3)
standardization: whether to standardize the training features before fitting the model
lt: True)
threshold: Threshold in binary classification prediction, in range [0, 1]. If threshol
resholds are both set, they must match.e.g. if threshold is p, then thresholds must be
o [1-p, p]. (default: 0.5)
thresholds: Thresholds in multi-class classification to adjust the probability of pred
ach class. Array must have length equal to the number of classes, with values > 0, exc
hat at most one value may be 0. The class with largest value p/t is predicted, where p
original probability of that class and t is the class's threshold. (undefined)
tol: the convergence tolerance for iterative algorithms (>= 0). (default: 1e-06)
upperBoundsOnCoefficients: The upper bounds on coefficients if fitting under bound con
optimization. The bound matrix must be compatible with the shape (1, number of feature
inomial regression, or (number of classes, number of features) for multinomial regress
defined)
upperBoundsOnIntercepts: The upper bounds on intercepts if fitting under bound constra
imization. The bound vector size must be equal with 1 for binomial regression, or the 
f classes for multinomial regression. (undefined)
weightCol: weight column name. If this is not set or empty, we treat all instance weig
1.0. (undefined)
```

Learn a LogisticRegression model. This uses the parameters stored in lr.

```
model1 = lr.fit(output)
```

Major shortcut - no train and test data!!!

Since model1 is a Model (i.e., a transformer produced by an Estimator), we can view the parameters it used durir
This prints the parameter (name: value) pairs, where names are unique IDs for this LogisticRegression instance.

```
print("Model 1 was fit using parameters: ")
```

```
Model 1 was fit using parameters:
```

```
print(model1.extractParamMap())
```

```
{Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', name='featuresCol', doc='fea
lumn name'): 'features', Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', name=
cNetParam', doc='the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the 
is an L2 penalty. For alpha = 1, it is an L1 penalty'): 0.0, Param(parent=u'LogisticRe
_4515a8d50a7ae3d58d1d', name='rawPredictionCol', doc='raw prediction (a.k.a. confidenc
n name'): 'rawPrediction', Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', na
l', doc='the convergence tolerance for iterative algorithms (>= 0)'): 1e-06, Param(par
```

```
gisticRegression_4515a8d50a7ae3d58d1d', name='family', doc='The name of family which is
ription of the label distribution to be used in the model. Supported options: auto, bir
multinomial.'): 'auto', Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', name=
m', doc='regularization parameter (>= 0)'): 0.3, Param(parent=u'LogisticRegression_4515
ae3d58d1d', name='standardization', doc='whether to standardize the training features b
itting the model'): True, Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', name
bilityCol', doc='Column name for predicted class conditional probabilities. Note: Not a
ls output well-calibrated probability estimates! These probabilities should be treated
idences, not precise probabilities'): 'probability', Param(parent=u'LogisticRegression_
50a7ae3d58d1d', name='aggregationDepth', doc='suggested depth for treeAggregate (>= 2)
aram(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', name='predictionCol', doc='pred
column name'): 'prediction', Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', n
tIntercept', doc='whether to fit an intercept term'): True, Param(parent=u'LogisticRegr
4515a8d50a7ae3d58d1d', name='labelCol', doc='label column name'): 'label', Param(parent
sticRegression_4515a8d50a7ae3d58d1d', name='maxIter', doc='maximum number of iterations
0)'): 10, Param(parent=u'LogisticRegression_4515a8d50a7ae3d58d1d', name='threshold', do
shold in binary classification prediction, in range [0, 1]'): 0.5}
```

```
trainingSummary = model1.summary
```

Obtain the objective per iteration

```
objectiveHistory = trainingSummary.objectiveHistory
print("objectiveHistory:")
```

```
objectiveHistory:
```

```
for objective in objectiveHistory:
    print(objective)
```

```
0.103031331693
0.102837131521
0.102673807922
0.102534456359
0.102523328224
0.10252332116
0.102523320957
0.102523319599
0.102523318774
0.102523316697
0.102523309257
```

Obtain the receiver-operating characteristic as a dataframe and areaUnderROC. TPR true positive rate FPR false rate

```
trainingSummary.roc.show()
```

```
+--------------------+--------------------+
|                 FPR|                 TPR|
+--------------------+--------------------+
|                 0.0|                 0.0|
|0.009273767010522465|0.043136538110034726|
| 0.01876227819710334| 0.07667702430999818|
|0.028280614963692326| 0.10866386401023578|
|0.037838719170292134| 0.13891427526960337|
| 0.04740278849289356| 0.16898190458782672|
|  0.0569628810714939|  0.1991409248766222|
|  0.0665547876921029| 0.2278376896362639 |
```

```
|  0.0665547876021029| 0.2278376896362639 4|
| 0.07624014761673732| 0.25269603363187715|
| 0.08592749600337228| 0.27691464083348566|
|  0.0956466583420159|  0.2996709925059404|
| 0.10534991370465517| 0.32324986291354413|
| 0.11509293650730527|  0.3450923048802778|
| 0.12486777326196404|  0.3654724913178578|
| 0.13465851699262713|   0.385212940961433|
| 0.14448107467529886| 0.40358252604642664|
| 0.15429766724196897| 0.42186072016084813|
|  0.1640903993446326|   0.441326996892707|
|     0.1738970500513| 0.46015353683056115|
|  0.1836977356419658|  0.4791628587095595|
+--------------------+--------------------+
only showing top 20 rows
```

```
print("areaUnderROC: " + str(trainingSummary.areaUnderROC))
```

```
areaUnderROC: 0.737392082954
```

```
prediction = model1.transform(output)
prediction.show(10)
```

```
+-----+---------------+-------+--------------------+----------------+------------+---
-----+-------------------+------------------+--------------+----------------+---------
-------+--------------------+--------------------+--------------------+----------+
|label|loan_identifier|channel|         seller_name|         intrate|      loanamt|
n2val|       numborrowers|       creditscore|property_state|origination_date|     fcl_
c_state|            features|       rawPrediction|         probability|prediction|
+-----+---------------+-------+--------------------+----------------+------------+---
-----+-------------------+------------------+--------------+----------------+---------
-------+--------------------+--------------------+--------------------+----------+
|    1|   100004116882|    0.1|      PNC BANK, N.A.|0.50000000000000|0.06633380884|1.08
80412|0.16666666666666666|0.6247030878859857|            IL|         06/2014|0.03184608
3.0|[0.5,0.0663338088...|[3.73538045677095...|[0.97669213099883...|       0.0|
|    0|   100006935001|    0.1|DITECH FINANCIAL LLC|0.54838709677419|0.11768901569|0.94
24742| 0.3333333333333333|0.9513064133016627|            IL|         05/2013|         0
3.0|[0.54838709677419...|[3.88517236715389...|[0.97986928391875...|       0.0|
|    0|   100007666444|    0.1|WELLS FARGO BANK,...|0.69354838709677|0.29172610556|0.96
84536|0.16666666666666666| 0.828978622327791|            NY|         02/2010|         0
12.0|[0.69354838709677...|[3.74067522359799...|[0.97681236057540...|       0.0|
|    0|   100015122571|    0.1|JPMORGAN CHASE BA...|0.43548387096774|0.04136947218|0.87
79381|0.16666666666666666|0.9382422802850356|            OH|         03/2013|         0
7.0|[0.43548387096774...|[3.91715139298351...|[0.98049049928748...|       0.0|
|    0|   100016186842|    0.1|JPMORGAN CHASE BA...|0.54838709677419|0.06918687589|1.34
56701| 0.3333333333333333|0.9109263657957245|            IL|         01/2013|         0
3.0|[0.54838709677419...|[3.82782618604890...|[0.97870642176617...|       0.0|
|    0|   100018762402|    0.1|          ALLY BANK|0.54838709677419|0.12125534950|1.00
00000| 0.3333333333333333|0.6805225653206651|            MN|         12/2012|         0
9.0|[0.54838709677419...|[3.77040059184264...|[0.97747618185931...|       0.0|
|    0|   100022248239|    0.1|               OTHER|0.53225806451612|0.09843081312|1.02
70103|0.16666666666666666|0.9394299287410927|            MO|         05/2012|         0
21.0|[0.53225806451612...|[3.85371159813761...|[0.97923924569525...|       0.0|
|    0|   100026182562|    0.2|               OTHER|0.62903225806451|0.16048502139|0.86
44329| 0.3333333333333333|0.9453681710213777|            MO|         12/2009|         0
```

```
21.0|[0.62903225806451...|[3.84962970833864...|[0.97915609937942...|       0.0|
|    0|   100032945091|    0.1|  CITIMORTGAGE, INC.|0.51612903225806|0.09700427960|0.96
94845|0.16666666666666666|0.8087885985748219|            CA|         06/2013|         6
0.0|[0.51612903225806...|[3.81387121506068...|[0.97841364683026...|       0.0|
|    0|   100033142675|    0.1|JPMORGAN CHASE BA...|0.64516129032258|0.17118402282|1.11
85567| 0.3333333333333333|0.9489311163895487|            NV|         09/2011|         6
18.0|[0.64516129032258...|[3.81811618533423...|[0.97850312038437...|       0.0|
+-----+---------------+-------+--------------------+----------------+-------------+----
-----+-------------------+------------------+--------------+----------------+----------
-------+--------------------+--------------------+--------------------+----------+
only showing top 10 rows
```

```
result = prediction.select("label", "probability", "prediction") \
    .collect()
```

print(result)

```
true0=0
false0=0
true1=0
false1=0
i=0
for row in result:
   if ( row.label == 0 and  row.prediction ==0 ):
      true0=true0+1
   if ( row.label == 0 and  row.prediction ==1 ):
      false1=false1+1
   if ( row.label == 1 and  row.prediction ==1 ):
      true1=true1+1
   if ( row.label == 1 and  row.prediction ==0 ):
      false0=false0+1
```

print("label=%s, prob=%s, prediction=%s" \ % (row.label, row.probability, row.prediction)) comment: don't break th full error count if (i > 10): break

```
print ("true0=%i false0=%i true1=%i false1=%i"        % (true0 ,   false0 , true1 , fal
```

```
true0=502924 false0=10942 true1=0 false1=0
```

```
trainingSummary.roc.show()
```

```
+--------------------+--------------------+
|                 FPR|                 TPR|
+--------------------+--------------------+
|                 0.0|                 0.0|
|0.009273767010522465|0.043136538110034726|
| 0.01876227819710334| 0.07667702430999818|
|0.028280614963692326| 0.10866386401023578|
|0.037838719170292134| 0.13891427526960337|
| 0.04740278849289356| 0.16898190458782672|
|  0.0569628810714939|  0.1991409248766222|
|  0.0665547876021029| 0.22783768963626394|
| 0.07624014761673732| 0.25269603363187715|
| 0.08592749600337228| 0.27691464083348566|
|  0.0956466583420159|  0.2996709925059404|
```

```
| 0.1053499137046551 7| 0.3232498629135441 3|
| 0.11509293650730527|  0.3450923048802778|
| 0.12486777326196404|  0.3654724913178578|
| 0.13465851699262713|   0.385212940961433|
| 0.14448107467529886| 0.40358252604642664|
| 0.15429766724196897| 0.42186072016084813|
|  0.1640903993446326|   0.441326996892707|
|     0.1738970500513| 0.46015353683056115|
|  0.1836977356419658|  0.4791628587095595|
+--------------------+--------------------+
only showing top 20 rows
```

```
print("areaUnderROC: " + str(trainingSummary.areaUnderROC))
```

```
areaUnderROC: 0.737392082954
```

can we do better with a deep learning keras network?

https://medium.com/datadriveninvestor/building-neural-network-using-keras-for-classification-3a3656c726c1 (https://medium.com/datadriveninvestor/building-neural-network-using-keras-for-classification-3a3656c726c1)

This runs in the kubernetes-docker CDSW cluster

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

need dataframe for keras with only numerics

```
output.show(1)
```

```
+-----+---------------+-------+--------------+----------------+-------------+----------
-------------------+-------------------+--------------+----------------+--------------+-
-+--------------------+
|label|loan_identifier|channel|   seller_name|         intrate|      loanamt|         lo
numborrowers|       creditscore|property_state|origination_date|     fcl_costs|loc_stat
features|
+-----+---------------+-------+--------------+----------------+-------------+----------
-------------------+-------------------+--------------+----------------+--------------+-
-+--------------------+
|    1|   100004116882|    0.1|PNC BANK, N.A.|0.50000000000000|0.06633380884|1.08247422
0.16666666666666666|0.6247030878859857|            IL|         06/2014|0.031846080481|
0|[0.5,0.0663338088...|
+-----+---------------+-------+--------------+----------------+-------------+----------
-------------------+-------------------+--------------+----------------+--------------+-
-+--------------------+
only showing top 1 row
```

```
kerasinputdf=output.drop('property_state','features','origination_date', 'loan_identifi
```

```
kerasinputdf.show(1)
```

```
+-----+-------+----------------+-------------+----------------+------------------+---
-------+---------+
|label|channel|         intrate|      loanamt|        loan2val|       numborrowers|
itscore|loc_state|
+-----+-------+----------------+-------------+----------------+------------------+---
-------+---------+
|    1|    0.1|0.50000000000000|0.06633380884|1.08247422680412|0.16666666666666666|0.62
8859857|      3.0|
+-----+-------+----------------+-------------+----------------+------------------+---
-------+---------+
only showing top 1 row
```

```
kerasinputdf.count()
```

```
513866
```

```
kerasinputpsdf=kerasinputdf.toPandas()
kerasinputpsdf.head()
```

	label	channel	intrate	loanamt	loan2val	numborrowers	creditscore
0	1	0.1	0.50000000000000	0.06633380884	1.08247422680412	0.166667	0.624703
1	0	0.1	0.54838709677419	0.11768901569	0.94845360824742	0.333333	0.951306
2	0	0.1	0.69354838709677	0.29172610556	0.90721649484536	0.166667	0.828979
3	0	0.1	0.43548387096774	0.04136947218	0.87628865979381	0.166667	0.938242
4	0	0.1	0.54838709677419	0.06918687589	1.34020618556701	0.333333	0.910926

```
kerasinputpsdf.count()
```

```
label           513866
channel         513866
intrate         513866
loanamt         513866
loan2val        513866
numborrowers    513866
creditscore     513866
loc_state       513866
dtype: int64
```

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

	label	channel	intrate	loanamt	loan2val	numborro
count	513866.000000	513866	513866	513866	513866	513866.00
unique	NaN	3	1801	727	120	NaN
top	NaN	0.1	0.54838709677419	0.12125534950	0.97938144329896	NaN
freq	NaN	436097	42322	2563	20591	NaN
mean	0.021293	NaN	NaN	NaN	NaN	0.260434
std	0.144361	NaN	NaN	NaN	NaN	0.084680
min	0.000000	NaN	NaN	NaN	NaN	0.166667
25%	0.000000	NaN	NaN	NaN	NaN	0.166667

50%	0.000000	NaN	NaN	NaN	NaN	0.333333
75%	0.000000	NaN	NaN	NaN	NaN	0.333333
max	1.000000	NaN	NaN	NaN	NaN	1.333333

```
sns.heatmap(kerasinputpsdf.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6622cda310>
```

we should not have "peeked" at the full dataset :-)

traindataset at 80% of sample

train and test split

```
trainpdf=kerasinputpsdf.sample(frac=0.8,random_state=200)
testpdf=kerasinputpsdf.drop(trainpdf.index)
trainpdf
```

	label	channel	intrate	loanamt	loan2val	numborrowers	cred
65599	0	0.1	0.48387096774193	0.17546362339	0.90721649484536	0.333333	0.91
467918	0	0.1	0.59677419354838	0.09914407988	0.86597938144329	0.166667	0.77
89298	0	0.1	0.50000000000000	0.21255349500	1.00000000000000	0.166667	0.80
158714	0	0.3	0.61290322580645	0.23537803138	1.19587628865979	0.333333	0.81
313647	0	0.3	0.54838709677419	0.14835948644	0.86597938144329	0.333333	0.95
5718	0	0.2	0.61290322580645	0.21968616262	0.90721649484536	0.333333	0.95
286298	0	0.1	0.53225806451612	0.14122681883	1.65979381443298	0.166667	0.94
395366	0	0.1	0.50000000000000	0.25320970042	0.83505154639175	0.166667	0.90
494248	0	0.1	0.58064516129032	0.05064194008	1.83505154639175	0.166667	0.96

[illegible]	0	0.1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
359337	0	0.1	0.51612903225806	0.10199714693	1.14432989690721	0.166667	0.95
178143	0	0.1	0.43548387096774	0.13052781740	0.88659793814432	0.333333	0.76
503153	0	0.1	0.58064516129032	0.18188302425	1.31958762886597	0.333333	0.82
426179	0	0.1	0.61290322580645	0.09557774607	0.83505154639175	0.333333	0.86
154097	0	0.3	0.54838709677419	0.11840228245	1.14432989690721	0.166667	0.94
59076	0	0.1	0.66129032258064	0.23609129814	1.02061855670103	0.166667	0.92
28763	0	0.1	0.74193548387096	0.21398002853	1.21649484536082	0.333333	0.77
72786	0	0.1	0.66129032258064	0.13266761768	1.08247422680412	0.333333	0.72
10182	0	0.1	0.56451612903225	0.13980028530	1.32989690721649	0.166667	0.95
486280	0	0.2	0.46774193548387	0.26105563480	0.97938144329896	0.166667	0.89
153755	0	0.2	0.56451612903225	0.22467902995	0.92783505154639	0.333333	0.95
476383	0	0.1	0.53225806451612	0.27888730385	1.19587628865979	0.166667	0.82
365230	0	0.1	0.50000000000000	0.06847360912	0.95876288659793	0.166667	0.90
348758	0	0.1	0.59677419354838	0.08844507845	1.52577319587628	0.333333	0.95
220536	0	0.1	0.61290322580645	0.28174037089	0.84536082474226	0.333333	0.92
97003	1	0.1	0.74193548387096	0.10199714693	1.05154639175257	0.333333	0.76
199997	0	0.1	0.59677419354838	0.12838801711	0.86597938144329	0.166667	0.92
260303	0	0.1	0.51483870967741	0.13409415121	1.10309278350515	0.166667	0.93
356829	0	0.1	0.51483870967741	0.19044222539	1.08247422680412	0.166667	0.75
288482	0	0.1	0.59677419354838	0.10413694721	1.06185567010309	0.333333	0.82
132024	0	0.1	0.53225806451612	0.08559201141	0.84536082474226	0.166667	0.90
...	...	...	...	...	...	...	...
59471	0	0.1	0.46451612903225	0.11840228245	0.89690721649484	0.166667	0.94
510602	0	0.3	0.62903225806451	0.13266761768	1.51546391752577	0.333333	0.90
490487	0	0.1	0.66129032258064	0.04850213980	1.02061855670103	0.333333	0.83
273793	0	0.1	0.67741935483870	0.16262482168	1.35051546391752	0.166667	0.78
37768	0	0.1	0.56451612903225	0.17332382310	0.98969072164948	0.333333	0.85
307681	0	0.3	0.57935483870967	0.12910128388	0.97938144329896	0.166667	0.94
490969	0	0.1	0.54838709677419	0.21897289586	1.50515463917525	0.166667	0.83
438287	0	0.1	0.45161290322580	0.07631954350	0.90721649484536	0.166667	0.80
451014	0	0.1	0.61290322580645	0.24322396576	1.30927835051546	0.333333	0.84
77098	0	0.1	0.62903225806451	0.23038516405	1.08247422680412	0.333333	0.81
80431	0	0.1	0.48387096774193	0.12696148359	1.04123711340206	0.166667	0.95
112225	0	0.1	0.53225806451612	0.04707560627	0.91752577319587	0.166667	0.87
85348	0	0.1	0.45161290322580	0.07132667617	1.01030927835051	0.166667	0.87
192473	0	0.1	0.62903225806451	0.08059914407	1.02061855670103	0.166667	0.81

348204	0	0.1	0.51612903225806	0.17617689015	0.89690721649484	0.333333	0.95
162088	0	0.1	0.51612903225806	0.21683309557	0.90721649484536	0.333333	0.94
140724	0	0.1	0.54838709677419	0.25534950071	1.13402061855670	0.333333	0.76
362894	0	0.1	0.37096774193548	0.15905848787	0.87628865979381	0.333333	0.80
306627	0	0.1	0.58064516129032	0.25820256776	0.88659793814432	0.166667	0.94
183737	0	0.1	0.46774193548387	0.05777460770	0.86597938144329	0.333333	0.95
490581	0	0.1	0.48387096774193	0.21326676176	0.97938144329896	0.166667	0.91
409211	0	0.3	0.61290322580645	0.19900142653	0.97938144329896	0.333333	0.88
70446	0	0.1	0.58064516129032	0.11626248216	0.89690721649484	0.166667	0.84
69467	0	0.1	0.56451612903225	0.26676176890	0.95876288659793	0.333333	0.87
275809	0	0.1	0.53225806451612	0.26390870185	0.96907216494845	0.333333	0.94
506799	0	0.1	0.69354838709677	0.08273894436	0.83505154639175	0.166667	0.90
425744	0	0.2	0.54838709677419	0.23537803138	0.92783505154639	0.333333	0.95
65258	0	0.1	0.74193548387096	0.12054208273	1.01030927835051	0.333333	0.76
343146	0	0.1	0.46451612903225	0.09129814550	0.87628865979381	0.333333	0.89
383516	0	0.1	0.51612903225806	0.17831669044	1.09278350515463	0.166667	0.82

411093 rows × 8 columns

```
testpdf
```

	label	channel	intrate	loanamt	loan2val	numborrowers	cred
0	1	0.1	0.50000000000000	0.06633380884	1.08247422680412	0.166667	0.62
2	0	0.1	0.69354838709677	0.29172610556	0.90721649484536	0.166667	0.82
8	0	0.1	0.51612903225806	0.09700427960	0.96907216494845	0.166667	0.80
9	0	0.1	0.64516129032258	0.17118402282	1.11340206185567	0.333333	0.94
13	0	0.1	0.50000000000000	0.15121255349	0.96907216494845	0.333333	0.94
23	0	0.1	0.58064516129032	0.13980028530	1.73195876288659	0.333333	0.86
31	0	0.1	0.56451612903225	0.05349500713	0.90721649484536	0.333333	0.90
33	0	0.1	0.51612903225806	0.09272467902	0.93814432989690	0.166667	0.89
41	0	0.1	0.48387096774193	0.05706134094	0.83505154639175	0.166667	0.86
56	0	0.1	0.48387096774193	0.05206847360	1.25773195876288	0.166667	0.94
57	0	0.2	0.50000000000000	0.07203994293	1.08247422680412	0.333333	0.93
59	0	0.1	0.61290322580645	0.08630527817	0.95876288659793	0.166667	0.76
62	0	0.1	0.54838709677419	0.23965763195	0.97938144329896	0.166667	0.93
68	0	0.1	0.70967741935483	0.07631954350	0.94845360824742	0.333333	0.96
69	0	0.1	0.62903225806451	0.15121255349	0.90721649484536	0.333333	0.85
70	0	0.3	0.67741935483870	0.10841654778	1.02061855670103	0.333333	0.92
76	0	0.1	0.52903225806451	0.28102710413	0.98969072164948	0.166667	0.84

77	0	0.3	0.46774193548387	0.05135520684	1.06185567010309	0.333333	0.92
88	0	0.1	0.61290322580645	0.10912981455	1.03092783505154	0.333333	0.87
94	0	0.1	0.45161290322580	0.06419400855	0.90721649484536	0.333333	0.96
97	0	0.1	0.50000000000000	0.07631954350	0.88659793814432	0.166667	0.69
99	1	0.1	0.62903225806451	0.04921540656	1.25773195876288	0.166667	0.86
109	0	0.1	0.51612903225806	0.10342368045	0.94845360824742	0.333333	0.93
120	0	0.1	0.50000000000000	0.19044222539	1.03092783505154	0.333333	0.91
145	0	0.1	0.50000000000000	0.17403708987	1.07216494845360	0.166667	0.81
151	0	0.1	0.56451612903225	0.15406562054	1.03092783505154	0.333333	0.90
157	0	0.1	0.43548387096774	0.13837375178	0.91752577319587	0.166667	0.81
163	0	0.1	0.64516129032258	0.20470756062	1.11340206185567	0.166667	0.90
165	0	0.1	0.56451612903225	0.16476462196	0.93814432989690	0.166667	0.60
167	0	0.3	0.50000000000000	0.22895863052	0.89690721649484	0.333333	0.84
...	...	...	...	...	...	...	...
513696	0	0.3	0.58064516129032	0.12553495007	1.64948453608247	0.166667	0.92
513700	0	0.1	0.56451612903225	0.22111269614	1.18556701030927	0.166667	0.95
513712	0	0.1	0.66129032258064	0.03566333808	1.00000000000000	0.166667	0.84
513714	0	0.1	0.40322580645161	0.19044222539	1.10309278350515	0.333333	0.89
513716	0	0.1	0.56451612903225	0.11198288159	1.05154639175257	0.500000	0.94
513725	0	0.2	0.54838709677419	0.09985734664	1.69072164948453	0.166667	0.94
513726	0	0.1	0.58709677419354	0.15049928673	0.84536082474226	0.166667	0.83
513733	0	0.3	0.48387096774193	0.16975748930	0.85567010309278	0.166667	0.66
513734	0	0.1	0.53225806451612	0.17546362339	1.62886597938144	0.166667	0.81
513744	0	0.1	0.46774193548387	0.15263908701	0.84536082474226	0.166667	0.88
513759	0	0.1	0.61290322580645	0.06990014265	1.09278350515463	0.166667	0.85
513764	0	0.1	0.61290322580645	0.12339514978	1.22680412371134	0.333333	0.91
513769	0	0.1	0.54838709677419	0.13552068473	0.91752577319587	0.166667	0.74
513771	0	0.1	0.69354838709677	0.22681883024	1.00000000000000	0.333333	0.96
513782	0	0.1	0.70967741935483	0.10912981455	1.04123711340206	0.333333	0.93
513786	0	0.1	0.53225806451612	0.11198288159	0.86597938144329	0.333333	0.88
513789	0	0.1	0.59677419354838	0.11340941512	0.86597938144329	0.166667	0.83
513795	0	0.1	0.61290322580645	0.34593437945	0.88659793814432	0.166667	0.81
513797	0	0.1	0.38709677419354	0.07774607703	0.86597938144329	0.166667	0.96
513807	0	0.1	0.45161290322580	0.12767475035	0.83505154639175	0.333333	0.94
513809	0	0.1	0.56451612903225	0.28459343794	1.59793814432989	0.333333	0.92
513815	0	0.1	0.54838709677419	0.11055634807	0.90721649484536	0.333333	0.84

513821	0	0.1	0.61290322580645	0.14764621968	0.93814432989690	0.333333	0.96
513822	0	0.1	0.56451612903225	0.11768901569	1.60824742268041	0.166667	0.92
513838	0	0.1	0.51483870967741	0.22325249643	1.53608247422680	0.333333	0.88
513850	0	0.1	0.56451612903225	0.12624821683	0.87628865979381	0.166667	0.93
513855	0	0.1	0.67741935483870	0.09129814550	1.28865979381443	0.333333	0.86
513858	0	0.1	0.64516129032258	0.05563480741	0.94845360824742	0.166667	0.59
513860	0	0.1	0.54838709677419	0.11982881597	1.21649484536082	0.166667	0.83
513864	0	0.1	0.45161290322580	0.06704707560	0.85567010309278	0.166667	0.79

102773 rows × 8 columns

creating input features and target variables

```
X= trainpdf.iloc[:,1:7]
y= trainpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers	creditscore
65599	0.1	0.48387096774193	0.17546362339	0.90721649484536	0.333333	0.916865
467918	0.1	0.59677419354838	0.09914407988	0.86597938144329	0.166667	0.779097
89298	0.1	0.50000000000000	0.21255349500	1.00000000000000	0.166667	0.809976
158714	0.3	0.61290322580645	0.23537803138	1.19587628865979	0.333333	0.813539
313647	0.3	0.54838709677419	0.14835948644	0.86597938144329	0.333333	0.951306

```
y.head()
```

```
65599     0
467918    0
89298     0
158714    0
313647    0
Name: label, dtype: int8
```

from sklearn.cross_validation import train_test_split

import sklearn from sklearn.model_selection import train_test_split from sklearn import train_test_split X_train, X
y_train, y_test = train_test_split(X, y, test_size=0.3)

!pip install –upgrade –force-reinstall tensorflow !pip install –upgrade –force-reinstall keras

```
import tensorflow as tf
import keras as ks
```

```
Using TensorFlow backend.
```

```
from keras import Sequential
from keras.layers import Dense
from keras.callbacks import TensorBoard
```

tutorial from tensorflow.keras.callbacks import TensorBoard

```
classifier = Sequential()
```

First Hidden Layer

```
classifier.add(Dense(16, activation='relu', kernel_initializer='random_normal', input_d
```

Second Hidden Layer

```
classifier.add(Dense(20, activation='relu', kernel_initializer='random_normal'))
```

Output Layer

```
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))
```

Compiling the neural network

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

```
WARNING:tensorflow:From /home/cdsw/.local/lib/python2.7/site-packages/tensorflow/pythor
_impl.py:180: where (from tensorflow.python.ops.array_ops) is deprecated and will be re
n a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

tbcallback = tb(log_dir='/tmp/tblog', histogram_freq=0, write_graph=True, write_images=True)

upgrade dask to fix? fork first to save working project !pip install –upgrade –force-reinstall dask tensorboard = TensorBoard(log_dir="/tmp/tblogs")

Fitting the data to the training dataset classifier.fit(X,y, batch_size=10, epochs=50, verbose=1, callbacks=[tbcallba small batch size of 10 is hurting performance moving to 32

```
classifier.fit(X,y, batch_size=32, epochs=10, verbose=1)
```

```
ValueError: Error when checking input: expected dense_1_input to have shape (7,) but go
```

```
ValueErrorTraceback (most recent call last)
in engine
----> 1 classifier.fit(X,y, batch_size=32, epochs=10, verbose=1)

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in fit(self, x,
   1152             sample_weight=sample_weight,
   1153             class_weight=class_weight,
-> 1154             batch_size=batch_size)
   1155 
   1156         # Prepare validation data.

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in _standardize
    577             feed_input_shapes,
    578             check_batch_axis=False,  # Don't enforce the batch size.
--> 579             exception_prefix='input')
    580 
    581         if y is not None:

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training_utils.pyc in standa
    143                             ': expected ' + names[i] + ' to have shape ' +
    144                             str(shape) + ' but got array with shape ' +
--> 145                             str(data_shape))
    146     return data
    147 
```

```
ValueError: Error when checking input: expected dense_1_input to have shape (7,) but go
```

```
trainpdf.head
```

```
<bound method DataFrame.head of          label channel          intrate       loanamt
loan2val  \
65599       0     0.1  0.48387096774193  0.17546362339  0.90721649484536
467918      0     0.1  0.59677419354838  0.09914407988  0.86597938144329
89298       0     0.1  0.50000000000000  0.21255349500  1.00000000000000
158714      0     0.3  0.61290322580645  0.23537803138  1.19587628865979
313647      0     0.3  0.54838709677419  0.14835948644  0.86597938144329
5718        0     0.2  0.61290322580645  0.21968616262  0.90721649484536
286298      0     0.1  0.53225806451612  0.14122681883  1.65979381443298
395366      0     0.1  0.50000000000000  0.25320970042  0.83505154639175
494248      0     0.1  0.58064516129032  0.05064194008  1.83505154639175
359337      0     0.1  0.51612903225806  0.10199714693  1.14432989690721
178143      0     0.1  0.43548387096774  0.13052781740  0.88659793814432
503153      0     0.1  0.58064516129032  0.18188302425  1.31958762886597
426179      0     0.1  0.61290322580645  0.09557774607  0.83505154639175
154097      0     0.3  0.54838709677419  0.11840228245  1.14432989690721
59076       0     0.1  0.66129032258064  0.23609129814  1.02061855670103
28763       0     0.1  0.74193548387096  0.21398002853  1.21649484536082
72786       0     0.1  0.66129032258064  0.13266761768  1.08247422680412
10182       0     0.1  0.56451612903225  0.13980028530  1.32989690721649
486280      0     0.2  0.46774193548387  0.26105563480  0.97938144329896
153755      0     0.2  0.56451612903225  0.22467902995  0.92783505154639
476383      0     0.1  0.53225806451612  0.27888730385  1.19587628865979
365230      0     0.1  0.50000000000000  0.06847360912  0.95876288659793
348758      0     0.1  0.59677419354838  0.08844507845  1.52577319587628
220536      0     0.1  0.61290322580645  0.28174037089  0.84536082474226
97003       1     0.1  0.74193548387096  0.10199714693  1.05154639175257
199997      0     0.1  0.59677419354838  0.12838801711  0.86597938144329
260303      0     0.1  0.51483870967741  0.13409415121  1.10309278350515
356829      0     0.1  0.51483870967741  0.19044222539  1.08247422680412
288482      0     0.1  0.59677419354838  0.10413694721  1.06185567010309
132024      0     0.1  0.53225806451612  0.08559201141  0.84536082474226
...       ...     ...               ...            ...               ...
59471       0     0.1  0.46451612903225  0.11840228245  0.89690721649484
510602      0     0.3  0.62903225806451  0.13266761768  1.51546391752577
490487      0     0.1  0.66129032258064  0.04850213980  1.02061855670103
273793      0     0.1  0.67741935483870  0.16262482168  1.35051546391752
37768       0     0.1  0.56451612903225  0.17332382310  0.98969072164948
307681      0     0.3  0.57935483870967  0.12910128388  0.97938144329896
490969      0     0.1  0.54838709677419  0.21897289586  1.50515463917525
438287      0     0.1  0.45161290322580  0.07631954350  0.90721649484536
451014      0     0.1  0.61290322580645  0.24322396576  1.30927835051546
77098       0     0.1  0.62903225806451  0.23038516405  1.08247422680412
80431       0     0.1  0.48387096774193  0.12696148359  1.04123711340206
112225      0     0.1  0.53225806451612  0.04707560627  0.91752577319587
85348       0     0.1  0.45161290322580  0.07132667617  1.01030927835051
192473      0     0.1  0.62903225806451  0.08059914407  1.02061855670103
348204      0     0.1  0.51612903225806  0.17617689015  0.89690721649484
162088      0     0.1  0.51612903225806  0.21683309557  0.90721649484536
140724      0     0.1  0.54838709677419  0.25534950071  1.13402061855670
```

```
362894      0     0.1  0.37096774193548  0.15905848787  0.87628865979381
306627      0     0.1  0.58064516129032  0.25820256776  0.88659793814432
183737      0     0.1  0.46774193548387  0.05777460770  0.86597938144329
490581      0     0.1  0.48387096774193  0.21326676176  0.97938144329896
409211      0     0.3  0.61290322580645  0.19900142653  0.97938144329896
70446       0     0.1  0.58064516129032  0.11626248216  0.89690721649484
69467       0     0.1  0.56451612903225  0.26676176890  0.95876288659793
275809      0     0.1  0.53225806451612  0.26390870185  0.96907216494845
506799      0     0.1  0.69354838709677  0.08273894436  0.83505154639175
425744      0     0.2  0.54838709677419  0.23537803138  0.92783505154639
65258       0     0.1  0.74193548387096  0.12054208273  1.01030927835051
343146      0     0.1  0.46451612903225  0.09129814550  0.87628865979381
383516      0     0.1  0.51612903225806  0.17831669044  1.09278350515463

        numborrowers  creditscore  loc_state
65599       0.333333     0.916865       22.0
467918      0.166667     0.779097        2.0
89298       0.166667     0.809976       24.0
158714      0.333333     0.813539        2.0
313647      0.333333     0.951306       13.0
5718        0.333333     0.956057       29.0
286298      0.166667     0.945368        1.0
395366      0.166667     0.901425       10.0
494248      0.166667     0.960808        1.0
359337      0.166667     0.956057        5.0
178143      0.333333     0.763658       36.0
503153      0.333333     0.823040        1.0
426179      0.333333     0.863420        9.0
154097      0.166667     0.944181       14.0
59076       0.166667     0.921615       23.0
28763       0.333333     0.777910       12.0
72786       0.333333     0.722090        1.0
10182       0.166667     0.951306        3.0
486280      0.166667     0.895487       11.0
153755      0.333333     0.951306        8.0
476383      0.166667     0.824228        8.0
365230      0.166667     0.903800       21.0
348758      0.333333     0.954869        1.0
220536      0.333333     0.925178        3.0
97003       0.333333     0.767221        3.0
199997      0.166667     0.923990        8.0
260303      0.166667     0.931116       16.0
356829      0.166667     0.758907       19.0
288482      0.333333     0.824228        4.0
132024      0.166667     0.909739        7.0
...              ...          ...        ...
59471       0.166667     0.941805       16.0
510602      0.333333     0.901425        1.0
490487      0.333333     0.833729       21.0
273793      0.166667     0.785036        3.0
37768       0.333333     0.858670       12.0
307681      0.166667     0.948931        1.0
490969      0.166667     0.834917       11.0
438287      0.166667     0.802850        1.0
```

```
451014      0.333333     0.846793       11.0
77098       0.333333     0.811164       20.0
80431       0.166667     0.954869       12.0
112225      0.166667     0.875297       19.0
85348       0.166667     0.874109       19.0
192473      0.166667     0.818290        7.0
348204      0.333333     0.959620       23.0
162088      0.333333     0.940618       42.0
140724      0.333333     0.762470        4.0
362894      0.333333     0.802850       19.0
306627      0.166667     0.946556       12.0
183737      0.333333     0.953682        2.0
490581      0.166667     0.910926        8.0
409211      0.333333     0.888361       11.0
70446       0.166667     0.845606       15.0
69467       0.333333     0.875297       40.0
275809      0.333333     0.942993        5.0
506799      0.166667     0.904988        2.0
425744      0.333333     0.959620        9.0
65258       0.333333     0.768409        9.0
343146      0.333333     0.891924       21.0
383516      0.166667     0.823040       10.0

[411093 rows x 8 columns]>
```

creating input features and target variables

```
X= trainpdf.iloc[:,1:6]
y= trainpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers
65599	0.1	0.48387096774193	0.17546362339	0.90721649484536	0.333333
467918	0.1	0.59677419354838	0.09914407988	0.86597938144329	0.166667
89298	0.1	0.50000000000000	0.21255349500	1.00000000000000	0.166667
158714	0.3	0.61290322580645	0.23537803138	1.19587628865979	0.333333
313647	0.3	0.54838709677419	0.14835948644	0.86597938144329	0.333333

```
y.head()
```

```
65599     0
467918    0
89298     0
158714    0
313647    0
Name: label, dtype: int8
```

from sklearn.cross_validation import train_test_split

import sklearn from sklearn.model_selection import train_test_split from sklearn import train_test_split X_train, X
y_train, y_test = train_test_split(X, y, test_size=0.3)

!pip install –upgrade –force-reinstall tensorflow !pip install –upgrade –force-reinstall keras

```
import tensorflow as tf
```

```
import keras as ks
from keras import Sequential
from keras.layers import Dense
from keras.callbacks import TensorBoard
```

tutorial from tensorflow.keras.callbacks import TensorBoard

```
classifier = Sequential()
```

First Hidden Layer

```
classifier.add(Dense(16, activation='relu', kernel_initializer='random_normal', input_d
```

Second Hidden Layer

```
classifier.add(Dense(20, activation='relu', kernel_initializer='random_normal'))
```

Output Layer

```
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))
```

Compiling the neural network

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

tbcallback = tb(log_dir='/tmp/tblog', histogram_freq=0, write_graph=True, write_images=True)

upgrade dask to fix? fork first to save working project !pip install –upgrade –force-reinstall dask tensorboard = TensorBoard(log_dir="/tmp/tblogs")

Fitting the data to the training dataset classifier.fit(X,y, batch_size=10, epochs=50, verbose=1, callbacks=[tbcallb small batch size of 10 is hurting performance moving to 32

```
classifier.fit(X,y, batch_size=32, epochs=10, verbose=1)
ValueError: Error when checking input: expected dense_4_input to have shape (7,) but go
```

```
ValueErrorTraceback (most recent call last)
in engine
----> 1 classifier.fit(X,y, batch_size=32, epochs=10, verbose=1)

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in fit(self, x,
   1152             sample_weight=sample_weight,
   1153             class_weight=class_weight,
-> 1154             batch_size=batch_size)
   1155
   1156         # Prepare validation data.

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in _standardize
    577             feed_input_shapes,
    578             check_batch_axis=False,  # Don't enforce the batch size.
--> 579             exception_prefix='input')
    580
    581         if y is not None:

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training_utils.pyc in standa
    143                             ': expected ' + names[i] + ' to have shape ' +
    144                             str(shape) + ' but got array with shape ' +
--> 145                             str(data shape))
```

```
    return data

ValueError: Error when checking input: expected dense_4_input to have shape (7,) but go
```

creating input features and target variables

```
X= trainpdf.iloc[:,1:7]
y= trainpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers	creditscore
65599	0.1	0.48387096774193	0.17546362339	0.90721649484536	0.333333	0.916865
467918	0.1	0.59677419354838	0.09914407988	0.86597938144329	0.166667	0.779097
89298	0.1	0.50000000000000	0.21255349500	1.00000000000000	0.166667	0.809976
158714	0.3	0.61290322580645	0.23537803138	1.19587628865979	0.333333	0.813539
313647	0.3	0.54838709677419	0.14835948644	0.86597938144329	0.333333	0.951306

```
y.head()
```

```
65599     0
467918    0
89298     0
158714    0
313647    0
Name: label, dtype: int8
```

from sklearn.cross_validation import train_test_split

import sklearn from sklearn.model_selection import train_test_split from sklearn import train_test_split X_train, X
y_train, y_test = train_test_split(X, y, test_size=0.3)

!pip install –upgrade –force-reinstall tensorflow !pip install –upgrade –force-reinstall keras

```
import tensorflow as tf
import keras as ks
from keras import Sequential
from keras.layers import Dense
from keras.callbacks import TensorBoard
```

tutorial from tensorflow.keras.callbacks import TensorBoard

```
classifier = Sequential()
```

First Hidden Layer

```
classifier.add(Dense(16, activation='relu', kernel_initializer='random_normal', input_d
```

Second Hidden Layer

```
classifier.add(Dense(20, activation='relu', kernel_initializer='random_normal'))
```

Output Layer

```
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))
```

Compiling the neural network

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

tbcallback = tb(log_dir='/tmp/tblog', histogram_freq=0, write_graph=True, write_images=True)

upgrade dask to fix? fork first to save working project !pip install –upgrade –force-reinstall dask tensorboard = TensorBoard(log_dir="/tmp/tblogs")

Fitting the data to the training dataset classifier.fit(X,y, batch_size=10, epochs=50, verbose=1, callbacks=[tbcallba
small batch size of 10 is hurting performance moving to 32

```
classifier.fit(X,y, batch_size=32, epochs=10, verbose=1)
```

```
2019-09-27 19:51:08.214794: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your C
orts instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2019-09-27 19:51:08.222354: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] C
uency: 2591990000 Hz
2019-09-27 19:51:08.223124: I tensorflow/compiler/xla/service/service.cc:168] XLA servi
aab550 executing computations on platform Host. Devices:
2019-09-27 19:51:08.223188: I tensorflow/compiler/xla/service/service.cc:175]   StreamE
device (0): <undefined>, <undefined>
2019-09-27 19:51:08.365930: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412
time warning): Not using XLA:CPU for cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_g
it was not set.  If you want XLA:CPU, either set that envvar, or use experimental_jit_s
enable XLA:CPU.  To confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 
roper command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xla_hlo_pr
WARNING:tensorflow:From /home/cdsw/.local/lib/python2.7/site-packages/keras/backend/ten
_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.gl
riables instead.

Epoch 1/10
   13 14 14 14 14 15 15 159 16 16 17 17 17 17 18 18 189 19 19 20 20 20 21 21 21 21 22 2
23 23 24 24 25 25 26 26 26 26 27 27 27 27 28 28 28 28 28 28 29 29 30 30 30 31 31 321 32
33 337 34 34 34 35 35 35 35 35 36 36 36 36 37 37 37 38 38 38 39 39 39 40 40 40 41 41 41
42 43 43 44 44 44 45 45 45 46 46 47 47 47 47 48 48 48 48 49 49 50 50 50 50 51 51 51 52 
3 53 54 54 55 55 55 56 56 56 57 57 57 57 58 58 58 58 59 59 59 59 60 60 60 61 61 61 62 6
63 63 63 64 64 64 65 65 65 65 66 66 66 66 67 67 67 68 68 68 68 68 68 69 69 69 69 70 70 
1 71 72 72 73 73 74 74 75 75 75 76 76 76 77 77 77 78 78 78 78 78 79 79 79 79 80 80 80 8
80 81 81 81 81 81 81 82 82 82 82 83 83 83 83 84 84 84 85 85 85 85 85 86 86 86 86 87 87 
8 88 88 88 89 89 89 90 90 90 90 91 91 92 92 92 93 93 93 94 94 94 94 95 95 95 96 96 96 9
98 98 98 98 99 99 991001001001001011011011021021021021031031031041041041051051
510610610610710710710710810810810910910911011011111111111211211211311311411411511511511
117117118118118119119120120121121121122122123123124124124125125125126126127127127128128
291291301301311311311321321331331331341341351351361361361371371381381391391401401401
214214314314314414414514514614614714714814814914915015115115115215215315315415415415515
156157157158158159159159160160160161161162162163163164164164165165166166167167168168169
701711711711721721731731741751751761761771781781791791801811811821821821831831841841841
618618718718818818918919019019111911921921921931931941941941951951961961961971971981981
200200201202202203203204204205206206207207208208209209210210211211211212212212213213213
142152152162162172172172182182192192202202202212212212222222223223224224225225225226226
822822922923023123123223223323323423523523623623623723723823823923924024024024124124224
243243244244244244245245245246246247247247248248249249249250250250251251252252252252253
542552552562562562572572582582582592592602602612612612622622632632632642642642652652662
726726726826826826826926926927027027127127227227227327327327427427427527527527527627627
278278279279279280280281281282282283283283284284284285285286286287287288288289289289290
```

```
9129229229329329429429429529529629629729729729829829829929930030030030130130230230230 3
43043053053053063063073073083083093093103103103111312312313313313314314315315316316316 3
318319319320320320321321322322322323323324324325325325326326327327327328328328329329
3033033033133133233233233333333343343353353363363373373373383383393393400340340341341342
4334434434534534634634734734734834834934935035035035135135235235335335335435435535535535
6357357358358358359359359360360360361361362362363363363364364365365366366367367367368368
370370371371372372372373373374374375375375376376377377377378378379379379380380380381381
8238338338338338438438438438538538538638638638738738838838938938939039039039039139139 13
2393393394394395395396396396397397397398398399399400400401401401402402403403404404405405
40640740740840840840940941041041 1093/411093 [==============================] - 57s 138u
- loss: 0.1049 - accuracy: 0.9787
Epoch 2/10
  13 14 14 15 15 15 16 16 17 17 18 18 19 19 20 20 20 21 21 22 22 23 23 23 24 24 25 25 2
27 27 27 28 28 28 29 29 30 30 30 31 31 32 32 33 33 33 34 34 35 35 36 36 37 37 38 38 39
0 41 42 42 43 43 44 44 44 45 45 46 46 46 47 47 48 48 48 49 49 49 50 50 50 51 51 51 52 5
53 53 54 54 55 55 55 56 56 56 57 57 57 57 58 58 58 58 59 59 59 59 60 60 60 61 61 61 62
3 63 64 64 64 65 65 65 66 66 67 67 68 68 69 69 70 70 71 71 71 72 72 73 73 73 74 74 75 7
76 77 77 78 78 78 79 79 79 80 80 80 81 81 81 82 82 82 83 83 83 84 84 84 84 85 85 85 86
7 87 88 88 89 89 89 90 90 91 91 92 92 93 93 94 94 94 95 95 96 96 96 97 97 98 98 98 99 9
10110110210210210310310310410410410510510510510610610610710710710810810810910911011 0
11111111112112112112113113113114114115115115116116116117117118118119119119120120120120 1
11221221221221221221231231231241241241241251251261261261271271271271281281281291291291 2
13113113113213213213313313313313413413413413413513513513513613613613713713713713813813 8
39140140140141141142142142143143144144144145145146146147147148148148149149150150150151 1
11511521521521521531531531531541541541551551551551561561561571571571581581581581591591 5
16016116116116116216216316316416416416516516616616716716816816916916917017017117117117 2
73173174174175175176176176177177177178178178179179179180180180181181181182182182183183 1
41851851851861861871871881881881891891901901911911921921931931931941941951951951961961 9
19819819919920020020020120120220220320320320420420520520520620620620720720720820820820 8
09209210210210211211212212212213213214214214215215216216217217218218218218219219220220 2
22222232232242242252252252262262272272282282282292292292302302302312312312322322322332 3
23423423523523623623623723723723823823923924024024024124124224224324324424424524524524 6
47248248248248249249250250251251252252253254254255255256256257257257258258259259259260 2
12612622622622632632642642642652652652662662672672672682682682692692702712712722722732 7
27627627727727827827928028028128128228228328428428528628628728728828828929029029129129 2
94294295295296296297297297298298299300300301302302303303304304305306306307307308308308 3
03103113113123123123133143143153153163173173173183193193203213213223233243243253253263 2
32832932932933033133133233233233333333346335336336337337338339340340340341341342342343 4
34534534634634734734834935035035035135135235235335335435535535635635735835835935936036 0
61362363363364364365365366366367367368369369370370371372372373373374375375376377377378 3
03803813823833833843853863873883883893893903913913923933933943953953963963973973983993 9
40140140240240240240340340440440540540540640640740740840840940941041041 1093/411093 [===
====================] - 50s 123us/step - loss: 0.0959 - accuracy: 0.9787
Epoch 3/10
  14 14 15 15 15 16 16 16 17 17 17 18 18 19 19 19 20 20 20 20 20 21 21 21 22 22 22 23 2
24 25 25 25 25 25 26 26 26 26 27 27 27 27 27 28 28 28 28 28 29 29 29 30 30 30 31 31 31
2 32 32 33 33 33 34 34 34 34 35 35 35 35 35 35 35 36 36 36 36 36 36 36 36 36 36 36 37 3
37 37 37 37 38 38 38 38 38 38 38 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 40 40
0 40 40 40 40 40 40 41 41 41 41 42 42 42 42 42 42 42 42 43 43 43 43 43 43 44 44 44 44 4
45 45 45 45 45 45 45 46 46 46 46 46 46 46 46 46 47 47 47 47 47 47 47 47 47 48 48 48 48
9 49 49 49 49 49 49 49 49 50 50 50 50 50 50 50 50 51 51 51 51 51 51 51 51 51 51 5
52 52 52 52 52 52 52 52 52 52 53 53 53 53 53 53 53 53 53 54 54 54 54 54 54 54 54 54 54
5 55 55 55 55 55 55 55 55 56 56 56 56 56 56 56 56 56 56 57 57 57 57 57 57 57 57 58 5
58 58 58 58 58 59 59 59 59 59 59 59 59 59 60 60 60 60 60 60 60 60 61 61 61 61 61 61 61
```

```
1 62 62 62 62 62 62 63 63 63 63 63 63 64 64 64 64 64 64 65 65 65 65 65 65 65 65 65 65 
66 66 66 66 66 67 67 67 67 67 67 67 67 67 67 67 67 68 68 68 68 68 68 68 68 69 69 69 69 
0 70 70 70 70 71 71 71 71 71 71 71 71 71 72 72 72 72 72 72 72 73 73 73 73 73 73 73 73 
74 74 74 74 74 74 74 74 74 74 74 75 75 75 75 75 75 76 76 76 76 76 76 77 77 77 77 77 77 
8 78 78 78 78 78 79 79 79 79 79 79 79 80 80 80 80 80 80 81 81 81 81 82 82 82 82 83 83 
84 84 85 85 85 86 86 86 87 87 87 88 88 88 88 88 89 89 89 90 90 90 90 91 91 91 92 92 92 
3 93 93 94 94 94 94 94 95 95 95 95 96 96 96 97 97 97 98 98 98 99 991001001011011021021
102103103103103104104104105105105106106107107107108108108109109109110110111111112112113
141151151161161161171171181181191191191201201211211211221221221221231231241241241251251
612712712712812812812912913013013013113113113213213213313313413413413513513613613613613
138139139140140140141141141141141141142142142142142142143143143143143143144144144144144
451451461461461471471481491491501501511521521531541541551551561561571571581581581591601
016016116116216216316316416516516616716716816916917017017117191721731731741741751751761
717717717817817817917918018018118218218218318318318418418518618618718818818919019019119
192193193194194195195196197198198199200201201202202202203203203204204205205206207207208
102102102102112112122122132142142152162162172172172182192192202202212212222232232242252
722722822922923023023123123223323323423523623623723823923924024024024124124124224224224
244245245246246247248248249249250251251252253253254255255256257258258259260260261262263
652652662662662672672682682692692692702702712712722722732732742742742752752762762762772
827827927928028128128228328328428428528528528628628628728828828929029029129229229329429
296296296297297298298299300300301302302303304304305306306307308308309309310310311312312
143153153163173173183183193193203213223223233243243253263263273283283293303313313323333
533533633733733833833934034034134234334334334434434434534634634734834834935035035135135
235335335435435535535635635735735835935936036136236236336436436536636736736836937037037
373374374375376376377378378379379380381381382382383384384385385386387387388388389389389
913913923923933943943953953963973973983993994004014014024024034034044054054064064074084
0411411093/411093 [==============================] - 65s 157us/step - loss: 0.0954 - ac
0.9787
Epoch 4/10
  14 14 15 15 16 17 17 18 18 19 19 20 21 21 22 22 23 24 24 25 26 27 27 28 29 29 30 31 
33 34 34 35 36 36 37 38 39 39 40 41 41 42 42 43 43 44 45 45 46 46 47 47 48 49 50 51 51 
3 54 54 55 56 56 57 58 58 59 59 60 60 61 62 62 63 64 65 65 66 67 67 68 69 69 70 71 72 
74 75 76 76 77 78 79 80 80 81 82 83 83 84 85 86 86 87 88 89 90 90 91 92 92 93 94 95 95 
7 97 98 98 99 991001001011011021021031031041041051061061071081091091101111111121121131
115116117117118119120120121121122123123124124125125126126127127128128129129130130130
331331341341351351361361371371381391391391401401411411421421431441441451451451461461471
814814814814914915015015015015115115115215215315315315415515515615615715715815915915916
161161162162162163163164164165165166166167167168168169169170170171171172173173174174175
771771781781781781791791791791801801801811811821821831831831841841841851851861861871871
818918918919019019019119119219219319319419419419519519619619719719819919920020120120220
205205206207207208209209210210211211212212213214214215215216216217218219220220221222
242242252252262262272272282292292302302312312322322332332342352352352352362362372382382
024124124224224324424424524524624624724724824824924925025025025125125125225225325325425
256256256257257258259259260260260261261262262263263264265265266266267268268269270270271
732742742752762762772782782792802802812822822832842852852862862872882882892902902912922
329429529529529629629729829829829929930030130130230263033033043053053063073073083093093
231331331431531531631631731831831931932032132132232332332432532532632732732832932932933
331332332333334334335335336336337337338339339340340341342342343344344345346346347348349
513523523533543553563563573583593593603613613623633643643653663663673683683693703713713
337437537537537637737737837937938038038138138238238338338438438538538638738738738838838
391391392392393393394394394395395396396397397397398398399399400400401401402403403404404
054064064064074074074084084084094094094104104104104104111093/411093 [==================
====] - 38s 92us/step - loss: 0.0954 - accuracy: 0.9787
Epoch 5/10
 139 14 15 15 16 16 17 18 18 19 20 20 21 21 22 23 23 24 24 25 25 26 27 27 28 28 29 30 
```

```
32 33 34 34 35 35 36 37 38 38 39 39 40 41 41 42 43 44 44 45 46 47 47 48 48 49 49 50 50
2 52 53 53 54 55 55 56 56 57 57 58 58 59 59 60 60 61 61 62 62 62 63 63 64 65 65 66 66 6
69 70 70 71 72 72 73 73 74 74 75 76 76 77 77 78 79 79 80 81 82 82 83 84 84 85 86 86 87
9 90 91 92 92 93 94 95 95 96 97 97 98 99 991001001011021021031031041041051061061071071
1101111111121131131141151151161171181181191201201211221221231231241241251251261271271281
3013013113113213313313313413413513513613613713813813913914014014114114214314414414514
7148149149150151151152153153154155155156156157158159159160161161162162163164165165166161
1681691691701711711721721731731741741751751761771771781781791791791801801811811821831831
8618618718818818919019019111921921921931941951951961971971981992002002012012022032032042
5205206207207208208209210210211211212212212213213214214215216216217217218218219220221221
2232242252252262272272282292302302312312322332332342342352352362362372372382382392392401
41242242242243243244244245245245246246246247247248248672/411093 [=================
>...........] - ETA: 14s - loss:
```

```
Truncating text at 800000 characters to improve display performance.
Increase this limit with the environment variable 'MAX_TEXT_LENGTH'
```

```
classifier.summary()
```

```
Model: "sequential_3"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 16)                112
_________________________________________________________________
dense_8 (Dense)              (None, 20)                340
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 21
=================================================================
Total params: 473
Trainable params: 473
Non-trainable params: 0

_________________________________________________________________
```

print(classifier.metrics()

save the classifier to deploy

```
classifier.save("kerasLoanClassifier.keras.save")
eval_model=classifier.evaluate(X, y)
```

```
411093/411093 [==============================] - 17s 42us/step
```

```
eval_model
```

```
[0.09521508195356022, 0.9787396192550659]
```

what do the input features look like

```
X.head
```

```
<bound method DataFrame.head of        channel          intrate        loanamt
val  \
65599      0.1  0.48387096774193  0.17546362339  0.90721649484536
467918     0.1  0.59677419354838  0.09914407988  0.86597938144329
89298      0.1  0.50000000000000  0.21255349500  1.00000000000000
158714     0.3  0.61290322580645  0.23537803138  1.19587628865979
313647     0.3  0.54838709677419  0.14835948644  0.86597938144329
5718       0.2  0.61290322580645  0.21968616262  0.90721649484536
```

286298	0.1	0.53225806451612	0.14122681883	1.65979381443298
395366	0.1	0.50000000000000	0.25320970042	0.83505154639175
494248	0.1	0.58064516129032	0.05064194008	1.83505154639175
359337	0.1	0.51612903225806	0.10199714693	1.14432989690721
178143	0.1	0.43548387096774	0.13052781740	0.88659793814432
503153	0.1	0.58064516129032	0.18188302425	1.31958762886597
426179	0.1	0.61290322580645	0.09557774607	0.83505154639175
154097	0.3	0.54838709677419	0.11840228245	1.14432989690721
59076	0.1	0.66129032258064	0.23609129814	1.02061855670103
28763	0.1	0.74193548387096	0.21398002853	1.21649484536082
72786	0.1	0.66129032258064	0.13266761768	1.08247422680412
10182	0.1	0.56451612903225	0.13980028530	1.32989690721649
486280	0.2	0.46774193548387	0.26105563480	0.97938144329896
153755	0.2	0.56451612903225	0.22467902995	0.92783505154639
476383	0.1	0.53225806451612	0.27888730385	1.19587628865979
365230	0.1	0.50000000000000	0.06847360912	0.95876288659793
348758	0.1	0.59677419354838	0.08844507845	1.52577319587628
220536	0.1	0.61290322580645	0.28174037089	0.84536082474226
97003	0.1	0.74193548387096	0.10199714693	1.05154639175257
199997	0.1	0.59677419354838	0.12838801711	0.86597938144329
260303	0.1	0.51483870967741	0.13409415121	1.10309278350515
356829	0.1	0.51483870967741	0.19044222539	1.08247422680412
288482	0.1	0.59677419354838	0.10413694721	1.06185567010309
132024	0.1	0.53225806451612	0.08559201141	0.84536082474226
...	...	...	...	...
59471	0.1	0.46451612903225	0.11840228245	0.89690721649484
510602	0.3	0.62903225806451	0.13266761768	1.51546391752577
490487	0.1	0.66129032258064	0.04850213980	1.02061855670103
273793	0.1	0.67741935483870	0.16262482168	1.35051546391752
37768	0.1	0.56451612903225	0.17332382310	0.98969072164948
307681	0.3	0.57935483870967	0.12910128388	0.97938144329896
490969	0.1	0.54838709677419	0.21897289586	1.50515463917525
438287	0.1	0.45161290322580	0.07631954350	0.90721649484536
451014	0.1	0.61290322580645	0.24322396576	1.30927835051546
77098	0.1	0.62903225806451	0.23038516405	1.08247422680412
80431	0.1	0.48387096774193	0.12696148359	1.04123711340206
112225	0.1	0.53225806451612	0.04707560627	0.91752577319587
85348	0.1	0.45161290322580	0.07132667617	1.01030927835051
192473	0.1	0.62903225806451	0.08059914407	1.02061855670103
348204	0.1	0.51612903225806	0.17617689015	0.89690721649484
162088	0.1	0.51612903225806	0.21683309557	0.90721649484536
140724	0.1	0.54838709677419	0.25534950071	1.13402061855670
362894	0.1	0.37096774193548	0.15905848787	0.87628865979381
306627	0.1	0.58064516129032	0.25820256776	0.88659793814432
183737	0.1	0.46774193548387	0.05777460770	0.86597938144329
490581	0.1	0.48387096774193	0.21326676176	0.97938144329896
409211	0.3	0.61290322580645	0.19900142653	0.97938144329896
70446	0.1	0.58064516129032	0.11626248216	0.89690721649484
69467	0.1	0.56451612903225	0.26676176890	0.95876288659793
275809	0.1	0.53225806451612	0.26390870185	0.96907216494845
506799	0.1	0.69354838709677	0.08273894436	0.83505154639175
425744	0.2	0.54838709677419	0.23537803138	0.92783505154639
65258	0.1	0.74193548387096	0.12054208273	1.01030927835051
343146	0.1	0.46451612903225	0.09129814550	0.87628865979381

```
383516     0.1  0.51612903225806  0.17831669044  1.09278350515463

        numborrowers  creditscore
65599       0.333333     0.916865
467918      0.166667     0.779097
89298       0.166667     0.809976
158714      0.333333     0.813539
313647      0.333333     0.951306
5718        0.333333     0.956057
286298      0.166667     0.945368
395366      0.166667     0.901425
494248      0.166667     0.960808
359337      0.166667     0.956057
178143      0.333333     0.763658
503153      0.333333     0.823040
426179      0.333333     0.863420
154097      0.166667     0.944181
59076       0.166667     0.921615
28763       0.333333     0.777910
72786       0.333333     0.722090
10182       0.166667     0.951306
486280      0.166667     0.895487
153755      0.333333     0.951306
476383      0.166667     0.824228
365230      0.166667     0.903800
348758      0.333333     0.954869
220536      0.333333     0.925178
97003       0.333333     0.767221
199997      0.166667     0.923990
260303      0.166667     0.931116
356829      0.166667     0.758907
288482      0.333333     0.824228
132024      0.166667     0.909739
...              ...          ...
59471       0.166667     0.941805
510602      0.333333     0.901425
490487      0.333333     0.833729
273793      0.166667     0.785036
37768       0.333333     0.858670
307681      0.166667     0.948931
490969      0.166667     0.834917
438287      0.166667     0.802850
451014      0.333333     0.846793
77098       0.333333     0.811164
80431       0.166667     0.954869
112225      0.166667     0.875297
85348       0.166667     0.874109
192473      0.166667     0.818290
348204      0.333333     0.959620
162088      0.333333     0.940618
140724      0.333333     0.762470
362894      0.333333     0.802850
306627      0.166667     0.946556
183737      0.333333     0.953682
490581      0.166667     0.910926
```

```
409211      0.333333     0.888361
70446       0.166667     0.845606
69467       0.333333     0.875297
275809      0.333333     0.942993
506799      0.166667     0.904988
425744      0.333333     0.959620
65258       0.333333     0.768409
343146      0.333333     0.891924
383516      0.166667     0.823040

[411093 rows x 6 columns]>
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.65)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[402353      0]
 [  8740      0]]
```

```
def loanofficer(Z):
  return (classifier.predict(Z)>.65)
```

function we will use for “production deployment”

```
loanofficer(X.head(2))
```

```
array([[False],
       [False]])
```

need test data - have we just memorized the input data?

try a multiple output final stage? more layers? more cowbell?

testing

creating input features and target variables

```
X= testpdf.iloc[:,1:8]
y= testpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers	creditscore	loc
0	0.1	0.50000000000000	0.06633380884	1.08247422680412	0.166667	0.624703	3.0
2	0.1	0.69354838709677	0.29172610556	0.90721649484536	0.166667	0.828979	12.
8	0.1	0.51612903225806	0.09700427960	0.96907216494845	0.166667	0.808789	0.0
9	0.1	0.64516129032258	0.17118402282	1.11340206185567	0.333333	0.948931	18.
13	0.1	0.50000000000000	0.15121255349	0.96907216494845	0.333333	0.942993	5.0

```
y.head()
```

```
0     1
2     0
8     0
9     0
13    0
Name: label, dtype: int8
```

```
eval_model=classifier.evaluate(X, y)
```

```
ValueError: Error when checking input: expected dense_7_input to have shape (6,) but go
```

```
ValueErrorTraceback (most recent call last)
in engine
----> 1 eval_model=classifier.evaluate(X, y)

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in evaluate(sel
   1347             x, y,
   1348             sample_weight=sample_weight,
-> 1349             batch_size=batch_size)
   1350         # Prepare inputs, delegate logic to `test_loop`.
   1351         if self._uses_dynamic_learning_phase():

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in _standardize
    577             feed_input_shapes,
    578             check_batch_axis=False,  # Don't enforce the batch size.
--> 579             exception_prefix='input')
    580
    581         if y is not None:

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training_utils.pyc in standa
    143                             ': expected ' + names[i] + ' to have shape ' +
    144                             str(shape) + ' but got array with shape ' +
--> 145                             str(data_shape))
    146     return data
    147

ValueError: Error when checking input: expected dense_7_input to have shape (6,) but go
```

```
y_pred=classifier.predict(X)
```

```
ValueError: Error when checking input: expected dense_7_input to have shape (6,) but go
```

```
ValueErrorTraceback (most recent call last)
in engine
----> 1 y_pred=classifier.predict(X)

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in predict(self
   1439
   1440         # Case 2: Symbolic tensors or Numpy array-like.
-> 1441         x, _, _ = self._standardize_user_data(x)
   1442         if self.stateful:
   1443             if x[0].shape[0] > batch_size and x[0].shape[0] % batch_size != 0:

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training.pyc in _standardize
    577             feed_input_shapes,
    578             check_batch_axis=False,  # Don't enforce the batch size.
```

```
--> 579             exception_prefix='input')
    580
    581         if y is not None:

/home/cdsw/.local/lib/python2.7/site-packages/keras/engine/training_utils.pyc in standa
    143                            ': expected ' + names[i] + ' to have shape ' +
    144                            str(shape) + ' but got array with shape ' +
--> 145                            str(data_shape))
    146     return data
    147

ValueError: Error when checking input: expected dense_7_input to have shape (6,) but go
```

```
X= testpdf.iloc[:,1:7]
y= testpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers	creditscore
0	0.1	0.50000000000000	0.06633380884	1.08247422680412	0.166667	0.624703
2	0.1	0.69354838709677	0.29172610556	0.90721649484536	0.166667	0.828979
8	0.1	0.51612903225806	0.09700427960	0.96907216494845	0.166667	0.808789
9	0.1	0.64516129032258	0.17118402282	1.11340206185567	0.333333	0.948931
13	0.1	0.50000000000000	0.15121255349	0.96907216494845	0.333333	0.942993

```
y.head()
```

```
0     1
2     0
8     0
9     0
13    0
Name: label, dtype: int8
```

```
eval_model=classifier.evaluate(X, y)
```

```
102773/102773 [==============================] - 8s 81us/step
```

```
eval_model
```

```
[0.09580370603132549, 0.9785741567611694]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.65)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100571      0]
 [  2202      0]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.5)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100571      0]
 [  2202      0]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.4)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100571      0]
 [  2202      0]]
```

```
eval_model=classifier.evaluate(X, y)
```

```
102773/102773 [==============================] - 6s 62us/step
```

```
eval_model
```

```
[0.09580370603132549, 0.9785741567611694]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.3)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100571      0]
 [  2202      0]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.2)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100571      0]
 [  2202      0]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.1)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
```

```
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100539     32]
 [  2198      4]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.05)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[95479  5092]
 [ 1776   426]]
```

creating input features and target variables

```
X= trainpdf.iloc[:,1:7]
y= trainpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers	creditscore
65599	0.1	0.48387096774193	0.17546362339	0.90721649484536	0.333333	0.916865
467918	0.1	0.59677419354838	0.09914407988	0.86597938144329	0.166667	0.779097
89298	0.1	0.50000000000000	0.21255349500	1.00000000000000	0.166667	0.809976
158714	0.3	0.61290322580645	0.23537803138	1.19587628865979	0.333333	0.813539
313647	0.3	0.54838709677419	0.14835948644	0.86597938144329	0.333333	0.951306

```
y.head()
```

```
65599     0
467918    0
89298     0
158714    0
313647    0
Name: label, dtype: int8
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.05)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[381260  21093]
 [  7014   1726]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.1)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[402254     99]
 [  8718     22]]
```

```
classifier = Sequential()
```

First Hidden Layer

```
classifier.add(Dense(16, activation='relu', kernel_initializer='random_normal', input_d
```

Second Hidden Layer

```
classifier.add(Dense(20, activation='relu', kernel_initializer='random_normal'))
```

add another layer [[402254 99] [8718 22]] to try to beat the above

```
classifier.add(Dense(20, activation='relu', kernel_initializer='random_normal'))
```

Output Layer

```
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))
```

Compiling the neural network

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

tbcallback = tb(log_dir='/tmp/tblog', histogram_freq=0, write_graph=True, write_images=True)

upgrade dask to fix? fork first to save working project !pip install –upgrade –force-reinstall dask tensorboard = TensorBoard(log_dir="/tmp/tblogs")

Fitting the data to the training dataset classifier.fit(X,y, batch_size=10, epochs=50, verbose=1, callbacks=[tbcallb small batch size of 10 is hurting performance moving to 32

```
classifier.fit(X,y, batch_size=32, epochs=10, verbose=1)
```

```
Epoch 1/10
   41 13 14 14 14 15 15 15 16 16 17 17 17 18 18 19 19 19 20 20 20 21 21 21 21 22 22 22
3 24 24 25 25 25 26 26 27 27 27 27 28 28 28 29 29 29 30 30 30 31 31 32 32 32 33 337 34
5 35 35 36 36 36 37 37 37 38 38 38 39 39 40 40 40 41 41 42 42 42 42 42 43 43 43 44 44
45 46 46 46 47 47 47 48 48 48 49 49 49 50 50 50 50 51 51 51 52 52 52 53 53 53 53 54 54
5 55 55 56 56 56 57 57 57 58 58 58 58 59 59 59 59 60 60 60 61 61 61 62 62 62 63 63 63
65 65 65 66 66 66 67 67 67 68 68 68 69 69 69 70 70 70 71 71 71 71 72 72 72 72 73 73 73
4 75 75 75 75 76 76 76 76 77 77 77 78 78 78 78 79 79 79 80 80 81 81 81 82 82 82 83 83
85 85 85 86 86 87 87 87 88 88 88 89 89 90 90 90 91 91 92 92 92 93 93 94 94 94 95 95 96
7 97 97 98 98 98 99 99 99 9910010010010110110110210210210310310310410410510510510610610
107108108108109109109110110111111111112112113113113114114115115115116116116117117117
191191201201201211211221221221221231231231241241241241251251251261261261271271281281
913013013013113113113213213213313313413413513513513613613613713713813813813913913913
141141142142143143143144144144144145145145146146146146147147147148148148149149149150150
501511511511511511521521521521521531531531531541541541551551551561561571571571581581591
0160160161161162162162163163163164164164165165165166166166167167167168168169169169170
171172172173173174174174175175175176176176177177178178178179179180180181181181182182182
831841841841851851861861861871871881881881891891891901901911911911921921921921931931
4195195195196196196196197197197198198199199199199200200200201201201202202203203203204
20520620620620720720720820820820920921021021121121121221221221221321321421421421521
```

```
1621621621721721721721821821821821921921921921922022022022022022122122122122222222222222
3224224224225225225225225226226226226226227227227228228228228229229229230230230230231123
2322322322332332332332342342342352352352352362362362362362372372372372372372382382382382
3923923924024024024024024124124124224224224224224224224224224224224324324324324324424424442
5246246246246246246247247248248248249249249249250250250251251251252252252252253253254254254
2562562572572572572582582592592602602602612612612622622632632632632642642642652652652652
6626626726726726826826826826826826926926927027027027127127127227227227327327427427527527
6277277277278279279280280281281282282283283283284284285285286286287287288288289289290290
2912922922932932942942952952962962972972972982982992992993003003003013013013023023023030
0430430530530630730730830830830931031031131131231231331331431531531631631731831831931931
1322322323323324324324325325325326326326327327328328329329329330330330331331332332333333
3343353353363363373373383383393393403403413413423423433433443453453461346347347348348348
3513513523523533533543543553553563563573573583583593593603603603603613613613623623623636
6436536536636636736736836836936937037137137237237237337437437537537637637737837837837937
1381382382383383384384384384385385385385386386386386387387388388388389389389390390391139
3923933933933943953953963963973973983983993994004004004014014024024024034034040440440540
074074084084094094094104104110933/411093 [==============================] - 61s 148us/s
ss: 0.1028 - accuracy: 0.9787
Epoch 2/10
  14 14 14 15 16 16 17 17 17 18 18 19 19 20 21 21 22 22 23 23 24 24 25 25 26 27 27 28
30 30 31 31 32 32 33 33 34 34 34 35 35 36 36 37 37 38 38 39 39 40 40 41 41 42 42 43 43
5 45 46 47 47 48 48 49 49 50 50 51 51 52 52 53 54 54 55 55 56 56 56 57 57 57 57 58 58
60 60 61 61 62 62 62 63 63 64 64 64 65 65 65 66 66 66 67 67 67 68 68 69 69 70 70 70 71
2 73 73 74 74 75 75 76 77 77 78 78 79 79 80 80 81 81 82 83 83 84 84 85 85 86 86 87 87
89 89 90 91 91 91 92 92 93 93 94 94 95 96 96 97 97 98 98 99 99100100100101101102102102
4105105106107107108108109110110111111112112113113114115115116116117117117118118119119
1211221221231241241241251251261261271271281281291291301301311311321321331331341341351
3713813813913914014014114114214214314414414514514614714714814814914915015015115215215
5155156156157157158158159159160160161161162162163163164164165166166167167168168169169
1721721731731741751751761771771781791791801801811821821831831841841851851861871871881
9019119119219219319319419419519519619619719719819819920020020120120220320320420420520
7208209209210210211211212212213214214215216216217217218218219219220220221221222222222232
2252252252262262272272282282292292302302302312312322322332342342352352362362372372382
4024024124124224224324324424524524624624724724824824925025025125225225325425425525525
7258258259259260260261261262263263264264265265266266267268268269269270270271271272272
2752752762772772782782792802802812812822822832832842852852862872872882882892892902902
9229329329429429529529629629729829829929930030030130230230330330430430530630630730730
0310531131131231231331331431531531631631631731731831831931932032032132132232232332332
5326326327327328328329329330330331331332332333334334335336336337337338339339340340341
3433443443453463463473473483483493503503513513523523533533543543553553563563573573583
6036136136236236336336436436536536636636736836836936937037137137237337337437537537637
8379379380381381382382383384384385386386387387388389389389390390391391392392393393394
3963973973973983983993994004004014024024034034044054054064074074084084094094104114110
[==============================] - 40s 98us/step - loss: 0.0957 - accuracy: 0.9787
Epoch 3/10
  13 14 14 15 15 16 16 17 17 18 18 19 20 20 21 21 22 22 23 23 24 25 25 25 26 26 27 28
30 31 31 32 32 33 34 34 35 35 36 36 37 37 38 38 39 39 40 41 41 42 42 43 43 44 45 45 46
8 48 49 49 50 50 51 51 52 52 52 53 53 54 54 55 55 56 57 57 58 58 59 59 60 60 61 62 62
64 65 66 66 67 68 68 69 69 70 71 71 71 72 72 73 73 74 74 75 76 76 77 77 78 79 79 80 80
2 82 83 83 84 84 85 85 86 86 87 87 88 88 89 90 90 91 91 92 92 93 94 94 95 95 96 96 97
9910010110110210210310310410510510610610710710810910911011011111111211311311411411511
7117118118119119120120121121122122123123124124125126126127127128129129130130131132132
1351351361371371381381391401401411421421431431441451451461471471481481491501501511511
5315415415515515615615715715815915916016016116116216216316316416416516616616716716816
```

```
0171171172172173173174174175175176176177177178178178179180180180181181182182183183
185186186187187188188189189190190191191192192193194194195196196197198198199200200200
032032042042052062062072082082092092102112112122213213214214215215216216217217218218219
022122122222322322322422422522622622722722822822923023023123123223323323423423523523623
238239239240240241242242243244244244245246246247247248248249249249250251251252252253253
552552562562572572582592592602602612622622632632642652662662672682682692692702702712712
327327427427527527627627627727727727827827827927928028028128128228228228328328428428528
287287288288289289290291291292292293293294295295296296297297298298299299300301301302302
043043054306306307307308308309310310311311312312313313314314314315315316317317318318
203203213213223233233243243253253253263273273283283293293303313313323323333333343353353
733733833833934034034134134234334334434434534534634634734734834834934935035035135135235
354355355356356357358358359359360361361362362363363364364365366366367368368369370370371
723733743743753753763763773773783783793793803803813823823833833843843843853853863873873
939039039139139239339339439539539639639739739739839839940040040140140240240340440440540
4074084094094104104110 93/411093 [==============================] - 39s 94us/step - los
7 - accuracy: 0.9787
Epoch 4/10
  13 14 14 15 15 16 17 17 18 18 19 19 20 21 21 22 22 23 23 24 24 25 26 26 27 27 28 29
31 32 32 32 33 34 34 35 35 36 36 37 37 38 38 39 39 40 40 41 41 42 42 43 43 44 45 45 46
8 48 49 50 50 51 52 52 53 54 54 55 55 56 57 57 58 58 59 60 60 61 62 62 63 63 64 64 65
66 67 67 68 68 68 69 69 70 71 71 72 72 73 73 74 74 75 75 76 77 77 78 78 79 79 80 80 81
3 83 84 84 85 86 86 87 87 88 88 89 89 90 90 91 92 92 93 93 94 94 95 96 96 97 97 98 98
10010110110210210310310410410510510610610710710810810911011011111111211211311411411511
171181181191201201211221221231241241251261261271271281291291301301311311321321331331341
513613613613713713813813914014014114114214214314414414514514614614714814814914915015015
152153154154155156157157158158159160160161161162163163164164165165166166167167168168169
701711711721721731731731741741751751761761771771781781791801801811811821831831841841851
718718818818919019019119119219319319419419519619619719719819819919920020020120120220220
204205205206206207207208208209209210211211212212213214214215215216216217218218219219220
212222232232242242252262262272272282282292302302312322322332332342342352352362362372382
924024024124124224224324324424424524524624624724824824924925025125125225225325325425425
257257258258259259260260261261262262263263264264265265266266267267268268269270270271271
732732742742752752762762772782782782792802802812812822832832842842852852862862872882882
029029129229229329329429529529629629729729829929930030030130130130230230330330430483053
630730730830830931031031131131231231331331431531531631631731831831931932032132132232332
325326326327327328329329329330331331332333333333343343353353363373373383383393393403403411
433433443453453453463463473483483493493503503513513523523533543543553563563573573583593
036136236236336336436436536636736736836836936937037037137137237237337337337437437537637
378378379379380380381382382382383383383384384385385386387387388388389389390390391392392
943943953953963963973973983983993994004004004004014014014014024024024024024034034044044
54064064074074084084094094104104110 93/411093 [==============================] - 39s 95
- loss: 0.0955 - accuracy: 0.9787
Epoch 5/10
  14 14 14 15 16 16 16 17 17 17 18 18 19 19 20 20 21 21 22 22 22 23 23 24 24 24 25 25
27 27 28 28 29 29 30 30 31 31 32 32 33 33 34 34 35 35 36 36 37 37 38 39 39 40 40 41 41
3 43 44 44 45 45 46 46 47 47 48 48 49 49 49 50 50 51 51 52 52 53 53 54 54 55 55 56 56
59 59 60 60 61 62 62 63 63 64 65 65 66 66 67 67 68 69 69 70 70 71 72 72 73 73 74 75 75
7 78 78 78 78 78 79 80 80 81 81 81 82 82 83 84 84 84 85 85 86 87 87 87 88 88 89 89 90
92 92 93 93 94 94 95 95 96 96 97 97 98 98 99 9910010010110110210310310410410510510610 7
810910911011011111111211211311411411411511511611611711711811811911912012012112112212212
124125126126127128128129129130131131132133133134134135136136137137138139140140141141142
441441451461461471471481481491491501501511511521531531541541551551561561571571581581591
116216216316316416416516616616716816816917017017117117217217317317417417517617617717717
179180180180181181181181181182182183183184184184185185185186186186187187188188189189190
911921931931941941951951961971971981981992002002012012022022032042042052062062072082082
```

```
02102112112122122132132142142152152162162172172172182182192192202212212222222232232242
22622722722822822922923023123123223223323323423523523623623723723823823924024024124124
44244245245246246247247248248249249250250251251252252253254254255255256257257258258259
02612612622622632632642652652662672672682682692702702712722722732732742742752762772772
27928028028128128228228328328428428428528528528628628728828828928929029129129229229329
95295296297297298298299299300301301302302303304304304305306306307307308308309309310311
12313313314314315315316316316317317318318319319320320321321322323323324324325325326327
83293293303313313323333333343353353363363373383383393393403413413423423433433443443453
34634734734734834834934935035035135135235235235335435435535535635635635735835835935936
62362363363364364365366366367367368369369370370371372372373373374375375376376377377378
03803813813823823833833843843853853863873873883883893893903903913913923933933943953953
3983983993994004004014024024034034044054054064064074084084094104104111093/411093 [=====
=================] - 40s 97us/step - loss: 0.0955 - accuracy: 0.9787
Epoch 6/10
  13 14 15 15 16 17 17 18 18 19 20 20 21 21 22 23 23 24 24 24 25 26 26 27 28 28 29 30 3
32 33 34 34 35 35 36 36 37 37 38 38 39 39 40 40 41 41 42 42 43 43 44 44 45 45 46 47 47
9 49 50 51 51 52 52 53 53 54 54 55 56 56 57 58 59 59 60 61 61 62 62 63 64 64 65 65 66 6
68 69 69 70 70 71 71 72 72 73 73 73 74 75 75 75 76 76 77 78 78 78 79 79 80 81 81 82 82
4 85 85 86 86 87 88 88 89 89 90 90 91 92 92 93 93 94 95 95 96 96 97 97 98464/411093 [==
>..........
```

```
Truncating text at 800000 characters to improve display performance.
Increase this limit with the environment variable 'MAX_TEXT_LENGTH'
```

```
classifier.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 16)                112
_________________________________________________________________
dense_11 (Dense)             (None, 20)                340
_________________________________________________________________
dense_12 (Dense)             (None, 20)                420
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 21
=================================================================
Total params: 893
Trainable params: 893
Non-trainable params: 0
_________________________________________________________________
```

print(classifier.metrics()

save the classifier to deploy

```
classifier.save("kerasLoanClassifier.keras.save")
eval_model=classifier.evaluate(X, y)
```

```
411093/411093 [==============================] - 18s 43us/step
```

```
eval_model
```

```
[0.09516230860819683, 0.9787396192550659]
```

what do the input features look like

```
X.head
```

```
<bound method DataFrame.head of         channel          intrate        loanamt
val  \
65599      0.1  0.48387096774193  0.17546362339  0.90721649484536
467918     0.1  0.59677419354838  0.09914407988  0.86597938144329
89298      0.1  0.50000000000000  0.21255349500  1.00000000000000
158714     0.3  0.61290322580645  0.23537803138  1.19587628865979
313647     0.3  0.54838709677419  0.14835948644  0.86597938144329
5718       0.2  0.61290322580645  0.21968616262  0.90721649484536
286298     0.1  0.53225806451612  0.14122681883  1.65979381443298
395366     0.1  0.50000000000000  0.25320970042  0.83505154639175
494248     0.1  0.58064516129032  0.05064194008  1.83505154639175
359337     0.1  0.51612903225806  0.10199714693  1.14432989690721
178143     0.1  0.43548387096774  0.13052781740  0.88659793814432
503153     0.1  0.58064516129032  0.18188302425  1.31958762886597
426179     0.1  0.61290322580645  0.09557774607  0.83505154639175
154097     0.3  0.54838709677419  0.11840228245  1.14432989690721
59076      0.1  0.66129032258064  0.23609129814  1.02061855670103
28763      0.1  0.74193548387096  0.21398002853  1.21649484536082
72786      0.1  0.66129032258064  0.13266761768  1.08247422680412
10182      0.1  0.56451612903225  0.13980028530  1.32989690721649
486280     0.2  0.46774193548387  0.26105563480  0.97938144329896
153755     0.2  0.56451612903225  0.22467902995  0.92783505154639
476383     0.1  0.53225806451612  0.27888730385  1.19587628865979
365230     0.1  0.50000000000000  0.06847360912  0.95876288659793
348758     0.1  0.59677419354838  0.08844507845  1.52577319587628
220536     0.1  0.61290322580645  0.28174037089  0.84536082474226
97003      0.1  0.74193548387096  0.10199714693  1.05154639175257
199997     0.1  0.59677419354838  0.12838801711  0.86597938144329
260303     0.1  0.51483870967741  0.13409415121  1.10309278350515
356829     0.1  0.51483870967741  0.19044222539  1.08247422680412
288482     0.1  0.59677419354838  0.10413694721  1.06185567010309
132024     0.1  0.53225806451612  0.08559201141  0.84536082474226
...        ...               ...            ...               ...
59471      0.1  0.46451612903225  0.11840228245  0.89690721649484
510602     0.3  0.62903225806451  0.13266761768  1.51546391752577
490487     0.1  0.66129032258064  0.04850213980  1.02061855670103
273793     0.1  0.67741935483870  0.16262482168  1.35051546391752
37768      0.1  0.56451612903225  0.17332382310  0.98969072164948
307681     0.3  0.57935483870967  0.12910128388  0.97938144329896
490969     0.1  0.54838709677419  0.21897289586  1.50515463917525
438287     0.1  0.45161290322580  0.07631954350  0.90721649484536
451014     0.1  0.61290322580645  0.24322396576  1.30927835051546
77098      0.1  0.62903225806451  0.23038516405  1.08247422680412
80431      0.1  0.48387096774193  0.12696148359  1.04123711340206
112225     0.1  0.53225806451612  0.04707560627  0.91752577319587
85348      0.1  0.45161290322580  0.07132667617  1.01030927835051
192473     0.1  0.62903225806451  0.08059914407  1.02061855670103
348204     0.1  0.51612903225806  0.17617689015  0.89690721649484
162088     0.1  0.51612903225806  0.21683309557  0.90721649484536
140724     0.1  0.54838709677419  0.25534950071  1.13402061855670
362894     0.1  0.37096774193548  0.15905848787  0.87628865979381
306627     0.1  0.58064516129032  0.25820256776  0.88659793814432
183737     0.1  0.46774193548387  0.05777460770  0.86597938144329
```

490581	0.1	0.48387096774193	0.21326676176	0.97938144329896
409211	0.3	0.61290322580645	0.19900142653	0.97938144329896
70446	0.1	0.58064516129032	0.11626248216	0.89690721649484
69467	0.1	0.56451612903225	0.26676176890	0.95876288659793
275809	0.1	0.53225806451612	0.26390870185	0.96907216494845
506799	0.1	0.69354838709677	0.08273894436	0.83505154639175
425744	0.2	0.54838709677419	0.23537803138	0.92783505154639
65258	0.1	0.74193548387096	0.12054208273	1.01030927835051
343146	0.1	0.46451612903225	0.09129814550	0.87628865979381
383516	0.1	0.51612903225806	0.17831669044	1.09278350515463

	numborrowers	creditscore
65599	0.333333	0.916865
467918	0.166667	0.779097
89298	0.166667	0.809976
158714	0.333333	0.813539
313647	0.333333	0.951306
5718	0.333333	0.956057
286298	0.166667	0.945368
395366	0.166667	0.901425
494248	0.166667	0.960808
359337	0.166667	0.956057
178143	0.333333	0.763658
503153	0.333333	0.823040
426179	0.333333	0.863420
154097	0.166667	0.944181
59076	0.166667	0.921615
28763	0.333333	0.777910
72786	0.333333	0.722090
10182	0.166667	0.951306
486280	0.166667	0.895487
153755	0.333333	0.951306
476383	0.166667	0.824228
365230	0.166667	0.903800
348758	0.333333	0.954869
220536	0.333333	0.925178
97003	0.333333	0.767221
199997	0.166667	0.923990
260303	0.166667	0.931116
356829	0.166667	0.758907
288482	0.333333	0.824228
132024	0.166667	0.909739
...	...	...
59471	0.166667	0.941805
510602	0.333333	0.901425
490487	0.333333	0.833729
273793	0.166667	0.785036
37768	0.333333	0.858670
307681	0.166667	0.948931
490969	0.166667	0.834917
438287	0.166667	0.802850
451014	0.333333	0.846793
77098	0.333333	0.811164
80431	0.166667	0.954869
112225	0.166667	0.875297

```
85348       0.166667    0.874109
192473      0.166667    0.818290
348204      0.333333    0.959620
162088      0.333333    0.940618
140724      0.333333    0.762470
362894      0.333333    0.802850
306627      0.166667    0.946556
183737      0.333333    0.953682
490581      0.166667    0.910926
409211      0.333333    0.888361
70446       0.166667    0.845606
69467       0.333333    0.875297
275809      0.333333    0.942993
506799      0.166667    0.904988
425744      0.333333    0.959620
65258       0.333333    0.768409
343146      0.333333    0.891924
383516      0.166667    0.823040

[411093 rows x 6 columns]>
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.1)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[402353      0]
 [  8740      0]]
```

```
def loanofficer(Z):
  return (classifier.predict(Z)>.65)
```

function we will use for “production deployment”

```
loanofficer(X.head(2))
```

```
array([[False],
       [False]])
```

need test data - have we just memorized the input data?

try a multiple output final stage? more layers? more cowbell?

testing

creating input features and target variables

```
X= testpdf.iloc[:,1:7]
y= testpdf.iloc[:,0]
X.head()
```

	channel	intrate	loanamt	loan2val	numborrowers	creditscore
0	0.1	0.50000000000000	0.06623280884	1.08247422680412	0.166667	0.624702

0	0.1	0.50000000000000	0.06633380884	1.08247422680412	0.166667	0.624703
2	0.1	0.69354838709677	0.29172610556	0.90721649484536	0.166667	0.828979
8	0.1	0.51612903225806	0.09700427960	0.96907216494845	0.166667	0.808789
9	0.1	0.64516129032258	0.17118402282	1.11340206185567	0.333333	0.948931
13	0.1	0.50000000000000	0.15121255349	0.96907216494845	0.333333	0.942993

```
y.head()
```

```
0     1
2     0
8     0
9     0
13    0
Name: label, dtype: int8
```

```
eval_model=classifier.evaluate(X, y)
```

```
102773/102773 [==============================] - 4s 38us/step
```

```
eval_model
```

```
[0.09564311794400045, 0.9785741567611694]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.1)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[100571      0]
 [  2202      0]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.05)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[89422 11149]
 [ 1450   752]]
```

```
y_pred=classifier.predict(X)
y_pred =(y_pred>0.07)
```

confusion matrix - barely correct when true

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, y_pred)
print(cm)
```

```
[[97586  2985]
 [ 1917   285]]
```

```
 [ 1917   285]]
```

Console will exit automatically if it remains idle for another sixty seconds.

Engine exited with status 129.