Case Study 4:
Financial Delinquency Project: Bankruptcy Prediction

Kebur Fantahun, Eli Kravez, Halle Purdom

February 28, 2022

Introduction

When a company was analyzing their biggest losses, they found it to be companies going bankrupt for their various investment strategies. To prevent this, they want to use their historical data to try and predict if a company is going to go bankrupt in the future so they can divest ahead of time. Their financial department has reached out to data scientists to create this prediction model, and they have provided the historical data about the companies they have been invested in. In this report, a Random Forest classification model and an XGBoost classification model were built and their parameters tuned to predict whether or not a company would eventually go bankrupt.

Methods

Data Preparation

The data provided by the finance department for this project included 5 files over 5 years containing data of various companies and whether or not they eventually went bankrupt. Even though the data spanned a range of time, the financial department has specified they do not care about the over time aspect of the data. To read the files, the arff package from scipy.io was used to get the data from the text files. These files were read in and combined to form one dataset of 43,405 entries with 64 attributes and 1 target. The different attributes contain data relating to the client company's financial status, such as assets, profits, sales, liabilities, and related statistics. The target variable of bankruptcy was converted to a binary integer of 0 and 1 so it can be interpreted by the models, where 0 represents not going bankrupt and 1 represents bankruptcy. The target variable is not balanced since it is pretty rare for a company to go bankrupt. There are 41,314 (95%) companies that do not go bankrupt, and 2,091 (5%) do go bankrupt.

Some companies were also missing some attribute values. The column with the most missing values was attribute 37 with 43.7% missing, and the second most was attribute 21 with 13.5% missing. Attribute 37 represents the (current assets - inventories) / long-term liabilities and attribute 21 represents profit on operating activities / financial expenses. The rest of the columns containing missing data were all 6% and lower. The mean was used to replace all the missing data using SimpleImputer.

The data was split into an 80% training set and a 20% testing set. The training set was used to tune the models parameters and the testing set was used to evaluate the final model's performances in terms of accuracy, precision, and recall.

Model Development

The prediction models were tuned for **highest accuracy**, but output for precision and recall is also provided. Precision and recall are very important to take into consideration in this case because we have a largely unbalanced dataset.

For the models, random search methods were used to find the best parameter value for the model. When the best model ended up being the highest value of the parameter supplied, the range of parameters was increased to make sure the best parameter could be found from the given lists.

Random Forest Classification Model Development

First a Random Forest model was created using the training data. To find the best parameters for this model, RandomGridSearch was used. The parameters tuned for this model included the following:

Parameter	Values
max_depth	[9,10, 15, 20, 25, 30 , 35]
max_features	[10,15, 20, 25]
min_samples_leaf	[3, 4, 5, 7]
min_samples_split	[8, 10, 12]
n_estimators	[100]
class_weight	['balanced', 'balanced_subsample']

criterion	['gini', 'entropy']

As seen above, the n_estimators parameter was set at 100 for parameter tuning for time efficiency. Once the best parameters were found with the random grid search, the final model n_estimators value was increased to 1,000. This was done only with this parameter because a bigger value for n_estimators is always better in terms of model performance. The class_weight was tuned between balanced and balanced_subsample to address the problem of having an unbalanced target variable. The final Random Forest model is discussed below in Results.

XGBoost Classification Model Development

The XGBoost model was also created using the training data and random selection of parameters to optimize the model. The parameters tested include the following:

Parameter	Values
subsample	10 Randomly generated numbers between 0 and 1
colsample_bytree	10 Randomly generated numbers between 0 and 1
max_depth	[3, 5, 10, 15, 20, 40]
min_child_weight	[1, 5, 10]
gamma	[0.5, 1, 1.5, 2, 5]
booster	‘gbtree’
objective	‘multi:softmax’
num_class	[2]
eta	[0.05]

As seen above, each model used the objective function ‘multi:softmax’ for our multiclass classification problem. The gbtree booster was used to select the tree based model for XGBoost, and num_class was set to 2 because the data has two values for the target variable. The eta learning rate was set to 0.05 and the rest of the variables were tuned to the values seen in the above table and the final model was chosen by minimizing the mean of the log loss function on the testing data. The final XGBoost model is discussed below in Results.

Results

The final model and its parameters for the Random Forest model are as follows:

RandomForestClassifier(class_weight='balanced', criterion='entropy', max_depth=20, max_features=25, min_samples_leaf=7, min_samples_split=12, n_estimators=1000)

The final optimized model for XGBoost is as follows:

xgb.train(final_params, dtrain, num_round=1000, evallist, early_stopping_rounds=2)

final_params = {'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05, 'subsample': 0.626486971668659, 'colsample_bytree': 0.895655234205252, 'max_depth': 20, 'min_child_weight': 5, 'gamma': 1.5}

To visualize the XGBoost model over each round of learning, a graph showing the log loss over each boosting round can be seen in Figure 1 below.

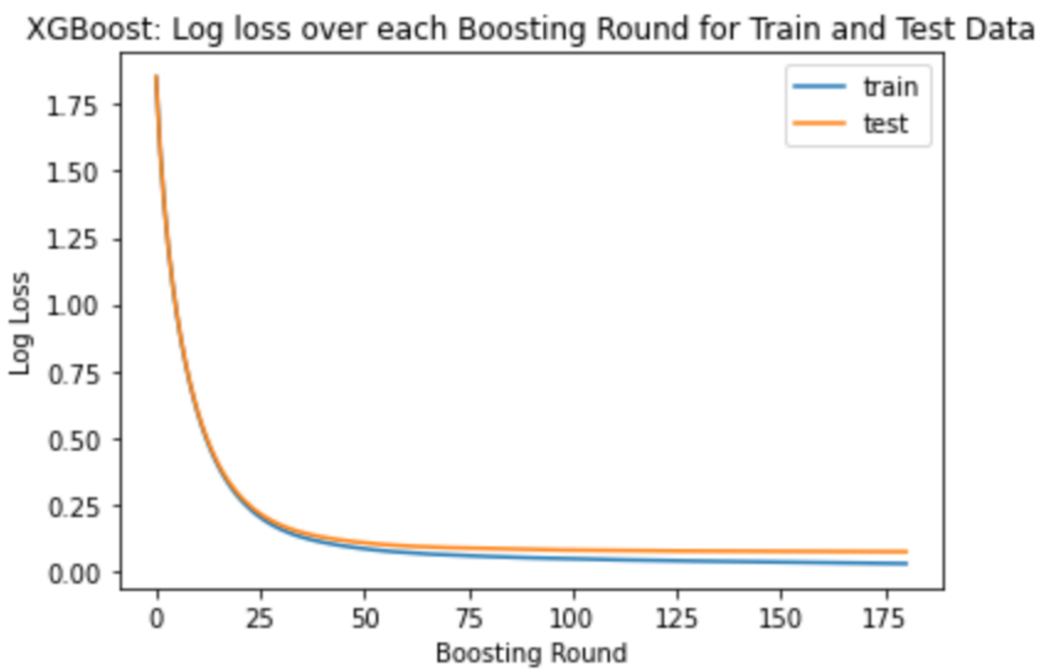


Figure 1: Log loss function over each boosting round for the XGBoost model. The log loss for the training data can be seen in blue and the log loss for the testing data can be seen in orange. As expected, the training data has a lower value for log loss than the testing. As seen in the graph, the log loss eventually becomes flat, meaning the training of the model is not increasing its performance by much after a certain point. This shows that the learning of the model is complete.

The final model's statistics can be seen below including accuracy, precision, and recall.

Model	Precision	Recall	**Accuracy**
Random Forest	0.84211	0.51232	**0.97270**
XGBoost	0.93416	0.55911	**0.97754**

As seen above, the XGBoost model outperforms the Random Forest model in terms of accuracy, precision, and recall. While accuracy is only a small increase in performance, precision and recall are much better. In terms of this unbalanced target these results show that the models are better at predicting the bankrupt class that has few data points to work from.

Conclusion

In conclusion, the XGBoost model produced a higher accuracy for predicting bankruptcy than the Random Forest model. The slightly better performance of the XGBoost model was expected because of the good performance of boosting algorithms, and this also confirmed that the parameter optimization for both models was successful. In this study only random search methods for parameter tuning were used to minimize the time needed to run the code. In future work, a more complete grid search could be conducted, which would further increase accuracy but would greatly increase time it takes to complete the search.

Case Study 4

From: Finance Department

To: Data Science Department

Subject: Financial Delinquency Project

We've collected our data. And we've noticed that you know what one of the biggest losses to our company is when companies go bankrupt and for our various investment strategies. So what we'd like to do is take a look at our historical data and see if there's any way that we can predict in the future that a company might go bankrupt, so that we can divest ourselves ahead of time.

From: Finance Department

To: Data Science Department

Subject: RE: Financial Delinquency Project

Oh, and just to clarify, this dataset is collected over five years, but we don't care the exact year that a company will go bankrupt, just whether or not they will go bankrupt at all, based on the data. Thanks!

Goal: Use Random Forest and XGBoost to accurately predict bankruptcy. Tune your models for maximum accuracy, but include precision and recall as summary metrics.

Dataset: https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import KFold
from sklearn.metrics import precision_score, recall_score, accuracy_score
import pandas as pd
from scipy.io import arff
from sklearn.ensemble import RandomForestClassifier
import missingno as msno
from numpy import nan
from numpy import isnan
from sklearn.impute import SimpleImputer
from xgboost import XGBClassifier
import xgboost as xgb
```

Data Preparation

In [ ]:

```
# Reading in files
# Change this folder: 'data 2/1year.arff' to folder containing datafiles
data1 = arff.loadarff('data 2/1year.arff')
df1 = pd.DataFrame(data1[0])
print(df1.shape)
```

```
data2 = arff.loadarff('data 2/2year.arff')
df2 = pd.DataFrame(data2[0])
print(df2.shape)

data3 = arff.loadarff('data 2/3year.arff')
df3 = pd.DataFrame(data3[0])
print(df3.shape)

data4 = arff.loadarff('data 2/4year.arff')
df4 = pd.DataFrame(data4[0])
print(df4.shape)

data5 = arff.loadarff('data 2/5year.arff')
df5 = pd.DataFrame(data5[0])
print(df5.shape)
```

```
(7027, 65)
(10173, 65)
(10503, 65)
(9792, 65)
(5910, 65)
```

In [ ]:

```
# Concatenating dataframes
dfs = [df1, df2, df3, df4, df5]
rawdata = pd.concat(dfs)
rawdata.shape
```

Out[ ]: (43405, 65)

In [ ]:

```
rawdata.head()
```

Out[ ]:

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	..
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	..
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	..
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	1.3090	0.30408	..
3	0.081483	0.30734	0.45879	2.4928	51.9520	0.14988	0.092704	1.86610	1.0571	0.57353	..
4	0.187320	0.61323	0.22960	1.4063	-7.3128	0.18732	0.187320	0.63070	1.1559	0.38677	..

5 rows × 65 columns

In [ ]:

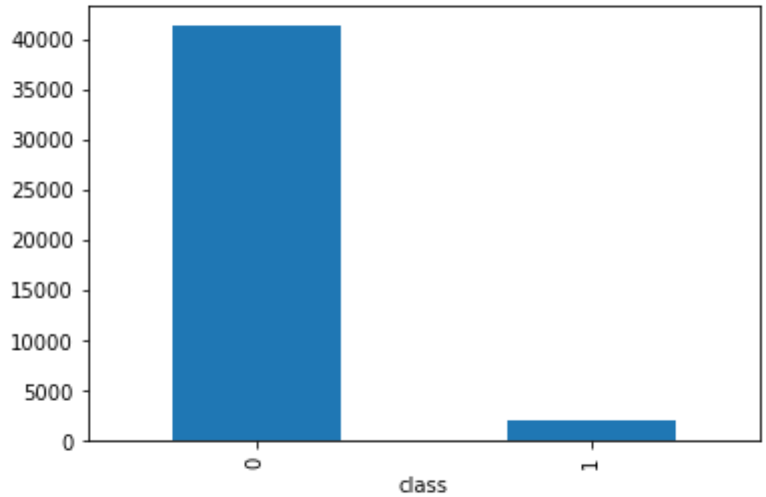
```
# One hot encode target
rawdata['class'] = pd.factorize(rawdata['class'])[0]
```

In [ ]:

```
# Look at target distribution
print(rawdata.groupby(by="class").size())
rawdata.groupby(by="class").size().plot.bar()
```

```
class
0    41314
1     2091
dtype: int64
```

```
Out[ ]: <AxesSubplot:xlabel='class'>
```

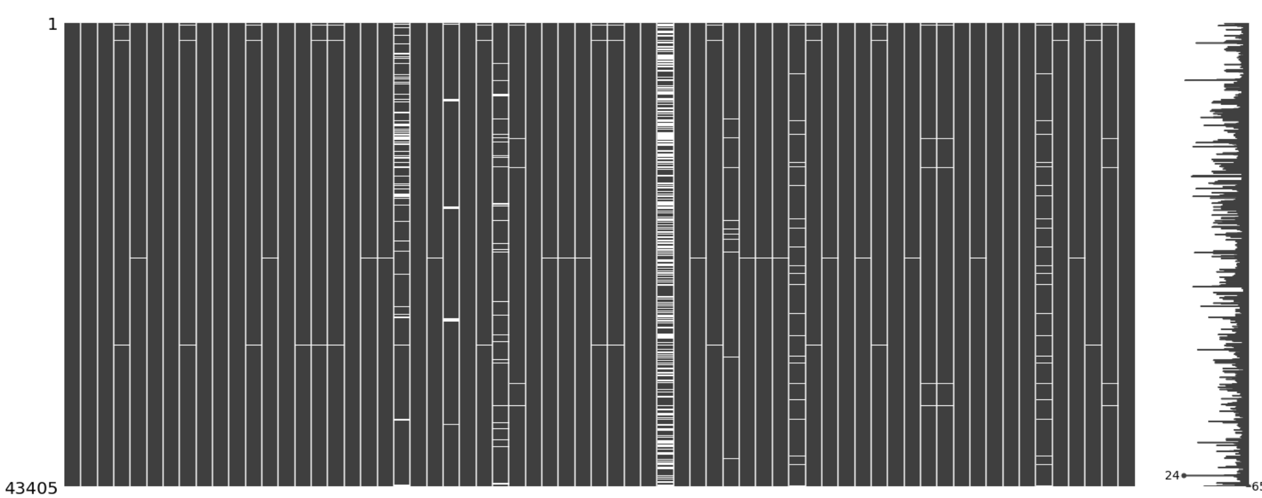



```
In [ ]: # Missing Data
        print(rawdata.isnull().sum())
```

```
Attr1       8
Attr2       8
Attr3       8
Attr4     134
Attr5      89
         ...
Attr61    102
Attr62    127
Attr63    134
Attr64    812
class       0
Length: 65, dtype: int64
```

```
In [ ]: # Missing Data Visualization
        msno.matrix(rawdata)
```

```
Out[ ]: <AxesSubplot:>
```




```
In [ ]: # Splitting target variable from attributes
        data = rawdata.iloc[:,0:64]
        target = rawdata['class']
```

```
# Check total missing values
values_pre = data.values
print('Missing: %d' % isnan(values_pre).sum())
```

```
Missing: 41322
```

In [ ]:

```
# Impute missing values
values = data.values

imputer = SimpleImputer(missing_values=nan, strategy='mean')

transformed_data = imputer.fit_transform(values)

print('Missing: %d' % isnan(transformed_data).sum())
```

```
Missing: 0
```

In [ ]:

```
# Transform back to dataframe
data = pd.DataFrame(transformed_data)
data.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	..
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	..
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	..
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	1.3090	0.30408	..
3	0.081483	0.30734	0.45879	2.4928	51.9520	0.14988	0.092704	1.86610	1.0571	0.57353	..
4	0.187320	0.61323	0.22960	1.4063	-7.3128	0.18732	0.187320	0.63070	1.1559	0.38677	..

5 rows × 64 columns

In [ ]:

```
# Train/test split
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2,
```

In [ ]:

```
X_train.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	
41259	0.041852	0.45276	0.273430	2.3347	15.2750	0.066363	0.043712	0.84156	1.0188	0.
40657	0.095317	0.37206	0.403710	2.2626	21.1340	-0.019148	0.117840	1.68770	1.5340	0.(
28231	0.132990	0.20781	0.735550	9.1921	155.5200	0.249450	0.163750	3.81200	1.2268	0.
811	0.383120	0.45246	0.083849	1.2908	-7.8546	0.442580	0.473600	1.19150	1.5371	0.
13488	0.023009	0.17168	0.330060	2.9226	66.9240	0.076127	0.030886	4.79520	0.9929	0.:

5 rows × 64 columns

Random Forest

In [ ]:

```
# Random Forest Parameter tuning
split = KFold(n_splits = 5, shuffle = True)

param_grid = {
    'max_depth': [9,10, 15, 20, 25, 30 , 35],
    'max_features': [10,15, 20, 25],
    'min_samples_leaf': [3, 4, 5, 7],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100],
    'class_weight':['balanced', 'balanced_subsample'],
    'criterion': ['gini', 'entropy']
}
# Create a based model
rf = RandomForestClassifier()
# Instantiate the grid search model
grid_search = RandomizedSearchCV(estimator = rf, param_distributions = param_gri
                          cv = split, n_jobs = 6, verbose = 2, scoring='accuracy
```

In [ ]:

```
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
```

Out[ ]:

```
RandomizedSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=True),
                   estimator=RandomForestClassifier(), n_iter=30, n_jobs=6,
                   param_distributions={'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [9, 10, 15, 20, 25, 30,
                                                      35],
                                        'max_features': [10, 15, 20, 25],
                                        'min_samples_leaf': [3, 4, 5, 7],
                                        'min_samples_split': [8, 10, 12],
                                        'n_estimators': [100]},
                   scoring='accuracy', verbose=2)
```

In [ ]:

```
# Best parameters
print(grid_search.best_estimator_)
print(grid_search.best_score_)
```

```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=20, max_features=25, min_samples_leaf=7,
                       min_samples_split=12)
0.9715758071151542
```

In [ ]:

```
# Add best params from above, make n_estimators greater=1000
rf = RandomForestClassifier(class_weight='balanced', criterion='entropy', max_de
rf.fit(X_train, y_train)
```

Out[ ]:

```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=20, max_features=25, min_samples_leaf=7,
                       min_samples_split=12, n_estimators=1000)
```

In [ ]:

```
# Performance statistics
preds = rf.predict(X_test)

precision = precision_score(y_true=y_test, y_pred=preds)
recall = recall_score(y_true=y_test, y_pred=preds)
accuracy = accuracy_score(y_true=y_test, y_pred=preds)
```

```
print(f"Accuracy: {accuracy:.5f}")
print(f"Precision: {precision:.5f}")
print(f"Recall: {recall:.5}")
```

```
Accuracy: 0.97270
Precision: 0.84211
Recall: 0.51232
```

XGBoost

In [ ]:

```
# Create dmatrices for xgboost
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)
evallist = [(dtest,'eval'), (dtrain,'train')]
num_round = 1000
```

In [ ]:

```
#Random search with xgboost package and dmatrices

params = {
    'booster':'gbtree',
    'objective':'multi:softmax',
    'num_class':2,
    'eta':0.05,
    'subsample':0.5,
    'colsample_bytree':0.5,
    'max_depth':3,
    'min_child_weight':1,
    'gamma': 0.5,
}

max_depth = [3,5,10,15,20,40]
sub_s = np.random.random(10)
cols = np.random.random(10)
md = np.random.randint(0,6,10)
m_child = [1, 5, 10]
mc = np.random.randint(0,3,10)
gam = [0.5, 1, 1.5, 2, 5]
g = np.random.randint(0,5,10)

for i in range(10):
    params['subsample']=sub_s[i]
    params['colsample_bytree']=cols[i]
    params['max_depth']=max_depth[md[i]]
    params['min_child_weight']=m_child[mc[i]]
    params['gamma']=gam[g[i]]
    tmp = xgb.cv(params, dtrain, num_boost_round=2000, nfold=5, stratified=False
    print('______________________done______________________')
    print(params)
    print(tmp.loc[tmp.shape[0]-1:,:])
    print("=================================================")
    tmp=0
```

```
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.5244252205385432, 'colsample_bytree': 0.20076858698063393, 'max_d
epth': 10, 'min_child_weight': 5, 'gamma': 0.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
```

```
323            0.023067            0.000342            0.093095

     test-mlogloss-std
323          0.004144
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.9432587885763094, 'colsample_bytree': 0.6370338166136749, 'max_de
pth': 20, 'min_child_weight': 5, 'gamma': 1.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
322            0.016427            0.000153             0.07373

     test-mlogloss-std
322          0.003277
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.47841206727338315, 'colsample_bytree': 0.013712710269699557, 'max
_depth': 40, 'min_child_weight': 1, 'gamma': 0.5}
    train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
86              0.10949             0.00059            0.180212

    test-mlogloss-std
86          0.004644
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.3332613435773485, 'colsample_bytree': 0.7987628866694357, 'max_de
pth': 40, 'min_child_weight': 5, 'gamma': 5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
391            0.053521            0.000282            0.080031

     test-mlogloss-std
391          0.002955
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.3571379741619207, 'colsample_bytree': 0.43080184902155905, 'max_d
epth': 10, 'min_child_weight': 10, 'gamma': 0.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
298             0.03616            0.000567            0.081087

     test-mlogloss-std
298          0.003049
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.4030339570851944, 'colsample_bytree': 0.10300620156631035, 'max_d
epth': 3, 'min_child_weight': 10, 'gamma': 1}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
802            0.077338            0.001235             0.10215

     test-mlogloss-std
802           0.00374
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.3428167871905941 6, 'colsample_bytree': 0.7910267248852602, 'max_d
epth': 20, 'min_child_weight': 10, 'gamma': 5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
508            0.056163            0.000296            0.080002

     test-mlogloss-std
508          0.003091
================================================
```

```
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.26429599604222465, 'colsample_bytree': 0.621176310277165 8, 'max_d
epth': 20, 'min_child_weight': 10, 'gamma': 1.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
268             0.047179              0.0005            0.082188

     test-mlogloss-std
268           0.003293
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.934175944416254, 'colsample_bytree': 0.04776423905882432, 'max_de
pth': 5, 'min_child_weight': 1, 'gamma': 0.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
462             0.069218            0.001321            0.123392

     test-mlogloss-std
462            0.00429
================================================
______________________done______________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.626486971668659, 'colsample_bytree': 0.895655234205252, 'max_dept
h': 20, 'min_child_weight': 5, 'gamma': 1.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
276              0.01905            0.000177            0.073181

     test-mlogloss-std
276           0.002771
================================================
```

In [ ]:

```
#Best model test-mlogloss-mean: 0.073181
final_params =  {'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class':
out_rs = xgb.cv(params=param, dtrain=dtrain, num_boost_round=2000, nfold=5, verb
```

```
[19:37:34] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-4
3e9a6c0910f/volume/xgboost-split_1619728204606/work/src/learner.cc:1061: Startin
g in XGBoost 1.3.0, the default evaluation metric used with the objective 'mult
i:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
[19:37:34] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-4
3e9a6c0910f/volume/xgboost-split_1619728204606/work/src/learner.cc:1061: Startin
g in XGBoost 1.3.0, the default evaluation metric used with the objective 'mult
i:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
[19:37:34] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-4
3e9a6c0910f/volume/xgboost-split_1619728204606/work/src/learner.cc:1061: Startin
g in XGBoost 1.3.0, the default evaluation metric used with the objective 'mult
i:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
[19:37:34] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-4
3e9a6c0910f/volume/xgboost-split_1619728204606/work/src/learner.cc:1061: Startin
g in XGBoost 1.3.0, the default evaluation metric used with the objective 'mult
i:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
[19:37:34] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-4
3e9a6c0910f/volume/xgboost-split_1619728204606/work/src/learner.cc:1061: Startin
g in XGBoost 1.3.0, the default evaluation metric used with the objective 'mult
i:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
[0]	train-mlogloss:1.85058+0.00096	test-mlogloss:1.85169+0.00106
[1]	train-mlogloss:1.57023+0.00111	test-mlogloss:1.57216+0.00137
[2]	train-mlogloss:1.36475+0.00112	test-mlogloss:1.36742+0.00156
[3]	train-mlogloss:1.20277+0.00112	test-mlogloss:1.20619+0.00165
```

```
[4]	train-mlogloss:1.07041+0.00127	test-mlogloss:1.07450+0.00183
[5]	train-mlogloss:0.95924+0.00105	test-mlogloss:0.96382+0.00197
[6]	train-mlogloss:0.86412+0.00109	test-mlogloss:0.86916+0.00204
[7]	train-mlogloss:0.78212+0.00119	test-mlogloss:0.78771+0.00213
[8]	train-mlogloss:0.71039+0.00120	test-mlogloss:0.71647+0.00226
[9]	train-mlogloss:0.64734+0.00131	test-mlogloss:0.65387+0.00233
[10]	train-mlogloss:0.59153+0.00129	test-mlogloss:0.59850+0.00228
[11]	train-mlogloss:0.54201+0.00126	test-mlogloss:0.54945+0.00237
[12]	train-mlogloss:0.49807+0.00123	test-mlogloss:0.50591+0.00245
[13]	train-mlogloss:0.45858+0.00117	test-mlogloss:0.46683+0.00241
[14]	train-mlogloss:0.42302+0.00125	test-mlogloss:0.43176+0.00250
[15]	train-mlogloss:0.39113+0.00111	test-mlogloss:0.40020+0.00254
[16]	train-mlogloss:0.36269+0.00107	test-mlogloss:0.37213+0.00264
[17]	train-mlogloss:0.33703+0.00102	test-mlogloss:0.34696+0.00267
[18]	train-mlogloss:0.31377+0.00114	test-mlogloss:0.32416+0.00271
[19]	train-mlogloss:0.29287+0.00113	test-mlogloss:0.30367+0.00266
[20]	train-mlogloss:0.27384+0.00098	test-mlogloss:0.28513+0.00267
[21]	train-mlogloss:0.25653+0.00110	test-mlogloss:0.26826+0.00254
[22]	train-mlogloss:0.24104+0.00107	test-mlogloss:0.25318+0.00255
[23]	train-mlogloss:0.22689+0.00101	test-mlogloss:0.23950+0.00260
[24]	train-mlogloss:0.21387+0.00081	test-mlogloss:0.22697+0.00262
[25]	train-mlogloss:0.20229+0.00078	test-mlogloss:0.21583+0.00263
[26]	train-mlogloss:0.19137+0.00084	test-mlogloss:0.20539+0.00256
[27]	train-mlogloss:0.18167+0.00099	test-mlogloss:0.19615+0.00247
[28]	train-mlogloss:0.17284+0.00094	test-mlogloss:0.18773+0.00246
[29]	train-mlogloss:0.16455+0.00099	test-mlogloss:0.17990+0.00244
[30]	train-mlogloss:0.15717+0.00088	test-mlogloss:0.17298+0.00251
[31]	train-mlogloss:0.15022+0.00094	test-mlogloss:0.16646+0.00263
[32]	train-mlogloss:0.14397+0.00093	test-mlogloss:0.16056+0.00263
[33]	train-mlogloss:0.13811+0.00100	test-mlogloss:0.15507+0.00257
[34]	train-mlogloss:0.13287+0.00098	test-mlogloss:0.15019+0.00267
[35]	train-mlogloss:0.12810+0.00097	test-mlogloss:0.14578+0.00274
[36]	train-mlogloss:0.12363+0.00097	test-mlogloss:0.14176+0.00281
[37]	train-mlogloss:0.11958+0.00090	test-mlogloss:0.13802+0.00282
[38]	train-mlogloss:0.11577+0.00090	test-mlogloss:0.13448+0.00281
[39]	train-mlogloss:0.11228+0.00089	test-mlogloss:0.13122+0.00272
[40]	train-mlogloss:0.10897+0.00090	test-mlogloss:0.12816+0.00276
[41]	train-mlogloss:0.10597+0.00089	test-mlogloss:0.12541+0.00279
[42]	train-mlogloss:0.10314+0.00083	test-mlogloss:0.12294+0.00280
[43]	train-mlogloss:0.10028+0.00077	test-mlogloss:0.12039+0.00310
[44]	train-mlogloss:0.09778+0.00089	test-mlogloss:0.11826+0.00310
[45]	train-mlogloss:0.09522+0.00103	test-mlogloss:0.11602+0.00323
[46]	train-mlogloss:0.09302+0.00088	test-mlogloss:0.11405+0.00314
[47]	train-mlogloss:0.09084+0.00093	test-mlogloss:0.11217+0.00333
[48]	train-mlogloss:0.08871+0.00079	test-mlogloss:0.11036+0.00330
[49]	train-mlogloss:0.08681+0.00092	test-mlogloss:0.10875+0.00344
[50]	train-mlogloss:0.08479+0.00092	test-mlogloss:0.10711+0.00352
[51]	train-mlogloss:0.08305+0.00094	test-mlogloss:0.10570+0.00350
[52]	train-mlogloss:0.08112+0.00089	test-mlogloss:0.10396+0.00328
[53]	train-mlogloss:0.07946+0.00082	test-mlogloss:0.10259+0.00331
[54]	train-mlogloss:0.07795+0.00085	test-mlogloss:0.10142+0.00335
[55]	train-mlogloss:0.07636+0.00082	test-mlogloss:0.10013+0.00327
[56]	train-mlogloss:0.07512+0.00089	test-mlogloss:0.09912+0.00334
[57]	train-mlogloss:0.07385+0.00077	test-mlogloss:0.09809+0.00339
[58]	train-mlogloss:0.07273+0.00088	test-mlogloss:0.09720+0.00339
[59]	train-mlogloss:0.07147+0.00087	test-mlogloss:0.09618+0.00346
[60]	train-mlogloss:0.07028+0.00094	test-mlogloss:0.09525+0.00336
[61]	train-mlogloss:0.06922+0.00093	test-mlogloss:0.09441+0.00326
[62]	train-mlogloss:0.06832+0.00078	test-mlogloss:0.09370+0.00319
[63]	train-mlogloss:0.06729+0.00081	test-mlogloss:0.09297+0.00311
[64]	train-mlogloss:0.06640+0.00089	test-mlogloss:0.09229+0.00307
[65]	train-mlogloss:0.06553+0.00088	test-mlogloss:0.09165+0.00315
[66]	train-mlogloss:0.06467+0.00090	test-mlogloss:0.09102+0.00309
[67]	train-mlogloss:0.06396+0.00096	test-mlogloss:0.09055+0.00302
[68]	train-mlogloss:0.06334+0.00090	test-mlogloss:0.09008+0.00298
```

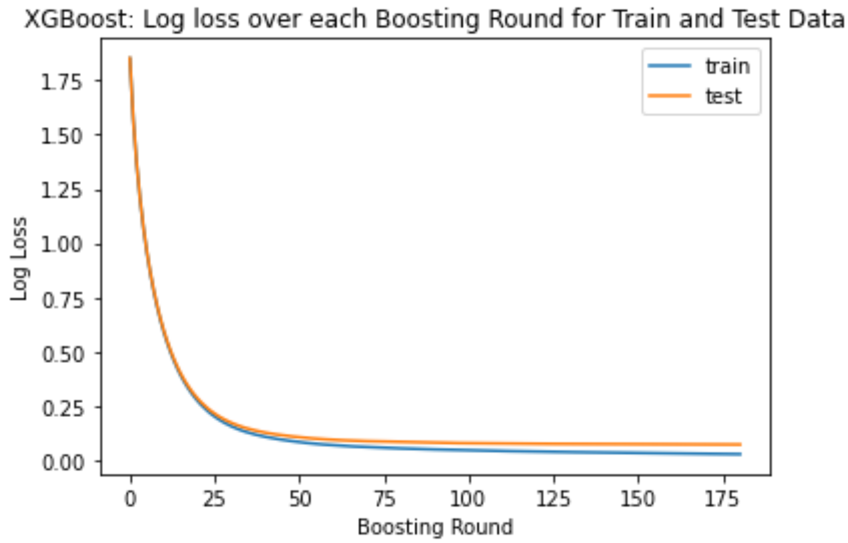
```
[69]	train-mlogloss:0.06257+0.00089	test-mlogloss:0.08953+0.00306
[70]	train-mlogloss:0.06188+0.00103	test-mlogloss:0.08907+0.00315
[71]	train-mlogloss:0.06132+0.00107	test-mlogloss:0.08871+0.00313
[72]	train-mlogloss:0.06063+0.00098	test-mlogloss:0.08820+0.00313
[73]	train-mlogloss:0.06003+0.00101	test-mlogloss:0.08778+0.00312
[74]	train-mlogloss:0.05939+0.00115	test-mlogloss:0.08734+0.00317
[75]	train-mlogloss:0.05874+0.00109	test-mlogloss:0.08699+0.00317
[76]	train-mlogloss:0.05811+0.00118	test-mlogloss:0.08654+0.00333
[77]	train-mlogloss:0.05752+0.00134	test-mlogloss:0.08616+0.00353
[78]	train-mlogloss:0.05681+0.00132	test-mlogloss:0.08563+0.00339
[79]	train-mlogloss:0.05628+0.00132	test-mlogloss:0.08536+0.00336
[80]	train-mlogloss:0.05571+0.00149	test-mlogloss:0.08492+0.00347
[81]	train-mlogloss:0.05514+0.00130	test-mlogloss:0.08453+0.00351
[82]	train-mlogloss:0.05457+0.00134	test-mlogloss:0.08408+0.00357
[83]	train-mlogloss:0.05395+0.00150	test-mlogloss:0.08371+0.00346
[84]	train-mlogloss:0.05350+0.00141	test-mlogloss:0.08341+0.00339
[85]	train-mlogloss:0.05301+0.00150	test-mlogloss:0.08313+0.00336
[86]	train-mlogloss:0.05257+0.00149	test-mlogloss:0.08288+0.00330
[87]	train-mlogloss:0.05214+0.00143	test-mlogloss:0.08265+0.00329
[88]	train-mlogloss:0.05173+0.00148	test-mlogloss:0.08242+0.00323
[89]	train-mlogloss:0.05129+0.00151	test-mlogloss:0.08217+0.00336
[90]	train-mlogloss:0.05094+0.00153	test-mlogloss:0.08200+0.00335
[91]	train-mlogloss:0.05055+0.00156	test-mlogloss:0.08177+0.00329
[92]	train-mlogloss:0.04997+0.00151	test-mlogloss:0.08142+0.00338
[93]	train-mlogloss:0.04958+0.00149	test-mlogloss:0.08125+0.00340
[94]	train-mlogloss:0.04902+0.00152	test-mlogloss:0.08085+0.00330
[95]	train-mlogloss:0.04862+0.00149	test-mlogloss:0.08062+0.00327
[96]	train-mlogloss:0.04830+0.00155	test-mlogloss:0.08044+0.00329
[97]	train-mlogloss:0.04804+0.00158	test-mlogloss:0.08030+0.00330
[98]	train-mlogloss:0.04766+0.00164	test-mlogloss:0.08013+0.00343
[99]	train-mlogloss:0.04736+0.00168	test-mlogloss:0.08000+0.00346
[100]	train-mlogloss:0.04701+0.00176	test-mlogloss:0.07980+0.00345
[101]	train-mlogloss:0.04665+0.00169	test-mlogloss:0.07959+0.00340
[102]	train-mlogloss:0.04640+0.00175	test-mlogloss:0.07952+0.00342
[103]	train-mlogloss:0.04614+0.00172	test-mlogloss:0.07940+0.00342
[104]	train-mlogloss:0.04572+0.00170	test-mlogloss:0.07913+0.00348
[105]	train-mlogloss:0.04547+0.00171	test-mlogloss:0.07900+0.00351
[106]	train-mlogloss:0.04518+0.00172	test-mlogloss:0.07885+0.00349
[107]	train-mlogloss:0.04478+0.00147	test-mlogloss:0.07864+0.00343
[108]	train-mlogloss:0.04439+0.00159	test-mlogloss:0.07842+0.00339
[109]	train-mlogloss:0.04409+0.00157	test-mlogloss:0.07833+0.00337
[110]	train-mlogloss:0.04379+0.00157	test-mlogloss:0.07816+0.00332
[111]	train-mlogloss:0.04348+0.00153	test-mlogloss:0.07800+0.00335
[112]	train-mlogloss:0.04325+0.00152	test-mlogloss:0.07789+0.00337
[113]	train-mlogloss:0.04295+0.00153	test-mlogloss:0.07776+0.00338
[114]	train-mlogloss:0.04274+0.00157	test-mlogloss:0.07770+0.00339
[115]	train-mlogloss:0.04247+0.00162	test-mlogloss:0.07753+0.00339
[116]	train-mlogloss:0.04218+0.00157	test-mlogloss:0.07739+0.00334
[117]	train-mlogloss:0.04196+0.00159	test-mlogloss:0.07733+0.00332
[118]	train-mlogloss:0.04165+0.00170	test-mlogloss:0.07718+0.00337
[119]	train-mlogloss:0.04137+0.00158	test-mlogloss:0.07705+0.00336
[120]	train-mlogloss:0.04111+0.00154	test-mlogloss:0.07699+0.00335
[121]	train-mlogloss:0.04080+0.00169	test-mlogloss:0.07684+0.00335
[122]	train-mlogloss:0.04057+0.00170	test-mlogloss:0.07674+0.00335
[123]	train-mlogloss:0.04031+0.00167	test-mlogloss:0.07664+0.00334
[124]	train-mlogloss:0.04011+0.00169	test-mlogloss:0.07662+0.00333
[125]	train-mlogloss:0.03988+0.00177	test-mlogloss:0.07660+0.00337
[126]	train-mlogloss:0.03962+0.00174	test-mlogloss:0.07654+0.00337
[127]	train-mlogloss:0.03938+0.00175	test-mlogloss:0.07646+0.00335
[128]	train-mlogloss:0.03913+0.00166	test-mlogloss:0.07635+0.00341
[129]	train-mlogloss:0.03893+0.00169	test-mlogloss:0.07627+0.00340
[130]	train-mlogloss:0.03870+0.00167	test-mlogloss:0.07621+0.00343
[131]	train-mlogloss:0.03853+0.00164	test-mlogloss:0.07618+0.00345
[132]	train-mlogloss:0.03831+0.00166	test-mlogloss:0.07609+0.00348
[133]	train-mlogloss:0.03809+0.00173	test-mlogloss:0.07605+0.00349
```

```
[134]	train-mlogloss:0.03786+0.00169	test-mlogloss:0.07598+0.00347
[135]	train-mlogloss:0.03769+0.00168	test-mlogloss:0.07595+0.00345
[136]	train-mlogloss:0.03750+0.00172	test-mlogloss:0.07587+0.00350
[137]	train-mlogloss:0.03732+0.00175	test-mlogloss:0.07586+0.00353
[138]	train-mlogloss:0.03703+0.00171	test-mlogloss:0.07569+0.00358
[139]	train-mlogloss:0.03682+0.00168	test-mlogloss:0.07565+0.00361
[140]	train-mlogloss:0.03664+0.00166	test-mlogloss:0.07563+0.00362
[141]	train-mlogloss:0.03647+0.00170	test-mlogloss:0.07559+0.00360
[142]	train-mlogloss:0.03629+0.00171	test-mlogloss:0.07558+0.00359
[143]	train-mlogloss:0.03616+0.00170	test-mlogloss:0.07560+0.00361
[144]	train-mlogloss:0.03595+0.00166	test-mlogloss:0.07553+0.00360
[145]	train-mlogloss:0.03569+0.00166	test-mlogloss:0.07539+0.00356
[146]	train-mlogloss:0.03545+0.00160	test-mlogloss:0.07534+0.00356
[147]	train-mlogloss:0.03527+0.00159	test-mlogloss:0.07529+0.00362
[148]	train-mlogloss:0.03504+0.00156	test-mlogloss:0.07516+0.00356
[149]	train-mlogloss:0.03471+0.00159	test-mlogloss:0.07496+0.00348
[150]	train-mlogloss:0.03443+0.00161	test-mlogloss:0.07481+0.00358
[151]	train-mlogloss:0.03428+0.00165	test-mlogloss:0.07477+0.00357
[152]	train-mlogloss:0.03412+0.00167	test-mlogloss:0.07475+0.00357
[153]	train-mlogloss:0.03392+0.00169	test-mlogloss:0.07473+0.00360
[154]	train-mlogloss:0.03372+0.00162	test-mlogloss:0.07466+0.00356
[155]	train-mlogloss:0.03348+0.00158	test-mlogloss:0.07453+0.00353
[156]	train-mlogloss:0.03334+0.00155	test-mlogloss:0.07450+0.00352
[157]	train-mlogloss:0.03307+0.00153	test-mlogloss:0.07444+0.00352
[158]	train-mlogloss:0.03292+0.00152	test-mlogloss:0.07445+0.00354
[159]	train-mlogloss:0.03269+0.00145	test-mlogloss:0.07440+0.00352
[160]	train-mlogloss:0.03243+0.00135	test-mlogloss:0.07430+0.00347
[161]	train-mlogloss:0.03234+0.00134	test-mlogloss:0.07428+0.00347
[162]	train-mlogloss:0.03213+0.00125	test-mlogloss:0.07418+0.00345
[163]	train-mlogloss:0.03199+0.00128	test-mlogloss:0.07414+0.00347
[164]	train-mlogloss:0.03182+0.00119	test-mlogloss:0.07408+0.00341
[165]	train-mlogloss:0.03163+0.00116	test-mlogloss:0.07406+0.00343
[166]	train-mlogloss:0.03146+0.00114	test-mlogloss:0.07401+0.00344
[167]	train-mlogloss:0.03135+0.00112	test-mlogloss:0.07399+0.00343
[168]	train-mlogloss:0.03121+0.00116	test-mlogloss:0.07401+0.00344
[169]	train-mlogloss:0.03105+0.00116	test-mlogloss:0.07393+0.00348
[170]	train-mlogloss:0.03088+0.00116	test-mlogloss:0.07392+0.00347
[171]	train-mlogloss:0.03076+0.00111	test-mlogloss:0.07391+0.00344
[172]	train-mlogloss:0.03056+0.00109	test-mlogloss:0.07391+0.00344
[173]	train-mlogloss:0.03043+0.00113	test-mlogloss:0.07389+0.00343
[174]	train-mlogloss:0.03024+0.00118	test-mlogloss:0.07388+0.00345
[175]	train-mlogloss:0.03008+0.00115	test-mlogloss:0.07387+0.00348
[176]	train-mlogloss:0.02993+0.00116	test-mlogloss:0.07386+0.00349
[177]	train-mlogloss:0.02980+0.00119	test-mlogloss:0.07385+0.00345
[178]	train-mlogloss:0.02964+0.00124	test-mlogloss:0.07382+0.00350
[179]	train-mlogloss:0.02947+0.00124	test-mlogloss:0.07379+0.00349
[180]	train-mlogloss:0.02929+0.00127	test-mlogloss:0.07373+0.00347
[181]	train-mlogloss:0.02909+0.00125	test-mlogloss:0.07374+0.00349
[182]	train-mlogloss:0.02895+0.00122	test-mlogloss:0.07374+0.00348
```

In [ ]:

```python
# Log loss plot

plt.plot(out_rs['train-mlogloss-mean'], label='train')
plt.plot(out_rs['test-mlogloss-mean'], label='test')
plt.legend()
plt.xlabel("Boosting Round")
plt.ylabel("Log Loss")
plt.title("XGBoost: Log loss over each Boosting Round for Train and Test Data")
plt.show()
```



In [ ]:

```
# Final model
model_rs = xgb.train(final_params, dtrain, num_round, evallist, early_stopping_r
preds = model_rs.predict(dtest)
```

```
[19:43:28] WARNING: /opt/concourse/worker/volumes/live/7a2b9f41-3287-451b-6691-4
3e9a6c0910f/volume/xgboost-split_1619728204606/work/src/learner.cc:1061: Startin
g in XGBoost 1.3.0, the default evaluation metric used with the objective 'mult
i:softmax' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
[0]	eval-mlogloss:0.64960	train-mlogloss:0.64912
[1]	eval-mlogloss:0.61019	train-mlogloss:0.60924
[2]	eval-mlogloss:0.57439	train-mlogloss:0.57298
[3]	eval-mlogloss:0.54156	train-mlogloss:0.53966
[4]	eval-mlogloss:0.51221	train-mlogloss:0.50976
[5]	eval-mlogloss:0.48467	train-mlogloss:0.48154
[6]	eval-mlogloss:0.45904	train-mlogloss:0.45556
[7]	eval-mlogloss:0.43552	train-mlogloss:0.43147
[8]	eval-mlogloss:0.41362	train-mlogloss:0.40911
[9]	eval-mlogloss:0.39341	train-mlogloss:0.38845
[10]	eval-mlogloss:0.37496	train-mlogloss:0.36946
[11]	eval-mlogloss:0.35744	train-mlogloss:0.35148
[12]	eval-mlogloss:0.34088	train-mlogloss:0.33454
[13]	eval-mlogloss:0.32547	train-mlogloss:0.31865
[14]	eval-mlogloss:0.31111	train-mlogloss:0.30384
[15]	eval-mlogloss:0.29835	train-mlogloss:0.29056
[16]	eval-mlogloss:0.28588	train-mlogloss:0.27763
[17]	eval-mlogloss:0.27416	train-mlogloss:0.26547
[18]	eval-mlogloss:0.26349	train-mlogloss:0.25433
[19]	eval-mlogloss:0.25309	train-mlogloss:0.24358
[20]	eval-mlogloss:0.24334	train-mlogloss:0.23345
[21]	eval-mlogloss:0.23398	train-mlogloss:0.22376
[22]	eval-mlogloss:0.22580	train-mlogloss:0.21511
[23]	eval-mlogloss:0.21789	train-mlogloss:0.20677
[24]	eval-mlogloss:0.21042	train-mlogloss:0.19883
[25]	eval-mlogloss:0.20333	train-mlogloss:0.19123
[26]	eval-mlogloss:0.19645	train-mlogloss:0.18401
[27]	eval-mlogloss:0.19012	train-mlogloss:0.17723
[28]	eval-mlogloss:0.18411	train-mlogloss:0.17076
[29]	eval-mlogloss:0.17835	train-mlogloss:0.16463
[30]	eval-mlogloss:0.17298	train-mlogloss:0.15888
[31]	eval-mlogloss:0.16822	train-mlogloss:0.15367
[32]	eval-mlogloss:0.16360	train-mlogloss:0.14859
[33]	eval-mlogloss:0.15925	train-mlogloss:0.14389
[34]	eval-mlogloss:0.15499	train-mlogloss:0.13924
```

```
[35]	eval-mlogloss:0.15101	train-mlogloss:0.13482
[36]	eval-mlogloss:0.14701	train-mlogloss:0.13049
[37]	eval-mlogloss:0.14343	train-mlogloss:0.12654
[38]	eval-mlogloss:0.14009	train-mlogloss:0.12275
[39]	eval-mlogloss:0.13681	train-mlogloss:0.11907
[40]	eval-mlogloss:0.13382	train-mlogloss:0.11566
[41]	eval-mlogloss:0.13093	train-mlogloss:0.11231
[42]	eval-mlogloss:0.12796	train-mlogloss:0.10901
[43]	eval-mlogloss:0.12524	train-mlogloss:0.10591
[44]	eval-mlogloss:0.12283	train-mlogloss:0.10310
[45]	eval-mlogloss:0.12041	train-mlogloss:0.10033
[46]	eval-mlogloss:0.11817	train-mlogloss:0.09767
[47]	eval-mlogloss:0.11600	train-mlogloss:0.09511
[48]	eval-mlogloss:0.11406	train-mlogloss:0.09278
[49]	eval-mlogloss:0.11210	train-mlogloss:0.09045
[50]	eval-mlogloss:0.11032	train-mlogloss:0.08823
[51]	eval-mlogloss:0.10858	train-mlogloss:0.08614
[52]	eval-mlogloss:0.10700	train-mlogloss:0.08425
[53]	eval-mlogloss:0.10540	train-mlogloss:0.08230
[54]	eval-mlogloss:0.10406	train-mlogloss:0.08051
[55]	eval-mlogloss:0.10265	train-mlogloss:0.07874
[56]	eval-mlogloss:0.10146	train-mlogloss:0.07719
[57]	eval-mlogloss:0.10021	train-mlogloss:0.07558
[58]	eval-mlogloss:0.09893	train-mlogloss:0.07394
[59]	eval-mlogloss:0.09759	train-mlogloss:0.07227
[60]	eval-mlogloss:0.09650	train-mlogloss:0.07084
[61]	eval-mlogloss:0.09531	train-mlogloss:0.06937
[62]	eval-mlogloss:0.09419	train-mlogloss:0.06795
[63]	eval-mlogloss:0.09329	train-mlogloss:0.06665
[64]	eval-mlogloss:0.09232	train-mlogloss:0.06532
[65]	eval-mlogloss:0.09144	train-mlogloss:0.06407
[66]	eval-mlogloss:0.09074	train-mlogloss:0.06297
[67]	eval-mlogloss:0.08980	train-mlogloss:0.06175
[68]	eval-mlogloss:0.08906	train-mlogloss:0.06070
[69]	eval-mlogloss:0.08838	train-mlogloss:0.05970
[70]	eval-mlogloss:0.08744	train-mlogloss:0.05856
[71]	eval-mlogloss:0.08675	train-mlogloss:0.05758
[72]	eval-mlogloss:0.08607	train-mlogloss:0.05663
[73]	eval-mlogloss:0.08544	train-mlogloss:0.05569
[74]	eval-mlogloss:0.08499	train-mlogloss:0.05492
[75]	eval-mlogloss:0.08437	train-mlogloss:0.05405
[76]	eval-mlogloss:0.08371	train-mlogloss:0.05316
[77]	eval-mlogloss:0.08325	train-mlogloss:0.05236
[78]	eval-mlogloss:0.08268	train-mlogloss:0.05155
[79]	eval-mlogloss:0.08225	train-mlogloss:0.05081
[80]	eval-mlogloss:0.08175	train-mlogloss:0.04998
[81]	eval-mlogloss:0.08141	train-mlogloss:0.04931
[82]	eval-mlogloss:0.08100	train-mlogloss:0.04860
[83]	eval-mlogloss:0.08063	train-mlogloss:0.04798
[84]	eval-mlogloss:0.08013	train-mlogloss:0.04724
[85]	eval-mlogloss:0.07961	train-mlogloss:0.04652
[86]	eval-mlogloss:0.07909	train-mlogloss:0.04587
[87]	eval-mlogloss:0.07877	train-mlogloss:0.04529
[88]	eval-mlogloss:0.07834	train-mlogloss:0.04464
[89]	eval-mlogloss:0.07787	train-mlogloss:0.04394
[90]	eval-mlogloss:0.07752	train-mlogloss:0.04337
[91]	eval-mlogloss:0.07731	train-mlogloss:0.04290
[92]	eval-mlogloss:0.07680	train-mlogloss:0.04232
[93]	eval-mlogloss:0.07631	train-mlogloss:0.04172
[94]	eval-mlogloss:0.07599	train-mlogloss:0.04113
[95]	eval-mlogloss:0.07562	train-mlogloss:0.04065
[96]	eval-mlogloss:0.07530	train-mlogloss:0.04018
[97]	eval-mlogloss:0.07488	train-mlogloss:0.03967
[98]	eval-mlogloss:0.07460	train-mlogloss:0.03922
[99]	eval-mlogloss:0.07439	train-mlogloss:0.03882
```

```
[100]	eval-mlogloss:0.07418	train-mlogloss:0.03841
[101]	eval-mlogloss:0.07405	train-mlogloss:0.03802
[102]	eval-mlogloss:0.07382	train-mlogloss:0.03764
[103]	eval-mlogloss:0.07360	train-mlogloss:0.03726
[104]	eval-mlogloss:0.07342	train-mlogloss:0.03693
[105]	eval-mlogloss:0.07322	train-mlogloss:0.03653
[106]	eval-mlogloss:0.07299	train-mlogloss:0.03615
[107]	eval-mlogloss:0.07272	train-mlogloss:0.03571
[108]	eval-mlogloss:0.07246	train-mlogloss:0.03534
[109]	eval-mlogloss:0.07226	train-mlogloss:0.03498
[110]	eval-mlogloss:0.07203	train-mlogloss:0.03462
[111]	eval-mlogloss:0.07192	train-mlogloss:0.03436
[112]	eval-mlogloss:0.07180	train-mlogloss:0.03404
[113]	eval-mlogloss:0.07164	train-mlogloss:0.03373
[114]	eval-mlogloss:0.07144	train-mlogloss:0.03341
[115]	eval-mlogloss:0.07130	train-mlogloss:0.03316
[116]	eval-mlogloss:0.07109	train-mlogloss:0.03284
[117]	eval-mlogloss:0.07077	train-mlogloss:0.03246
[118]	eval-mlogloss:0.07065	train-mlogloss:0.03217
[119]	eval-mlogloss:0.07038	train-mlogloss:0.03185
[120]	eval-mlogloss:0.07018	train-mlogloss:0.03156
[121]	eval-mlogloss:0.06997	train-mlogloss:0.03133
[122]	eval-mlogloss:0.06985	train-mlogloss:0.03105
[123]	eval-mlogloss:0.06964	train-mlogloss:0.03077
[124]	eval-mlogloss:0.06947	train-mlogloss:0.03053
[125]	eval-mlogloss:0.06933	train-mlogloss:0.03028
[126]	eval-mlogloss:0.06925	train-mlogloss:0.03005
[127]	eval-mlogloss:0.06907	train-mlogloss:0.02976
[128]	eval-mlogloss:0.06889	train-mlogloss:0.02952
[129]	eval-mlogloss:0.06883	train-mlogloss:0.02936
[130]	eval-mlogloss:0.06875	train-mlogloss:0.02914
[131]	eval-mlogloss:0.06872	train-mlogloss:0.02898
[132]	eval-mlogloss:0.06863	train-mlogloss:0.02878
[133]	eval-mlogloss:0.06857	train-mlogloss:0.02860
[134]	eval-mlogloss:0.06848	train-mlogloss:0.02837
[135]	eval-mlogloss:0.06842	train-mlogloss:0.02818
[136]	eval-mlogloss:0.06837	train-mlogloss:0.02806
[137]	eval-mlogloss:0.06832	train-mlogloss:0.02787
[138]	eval-mlogloss:0.06825	train-mlogloss:0.02772
[139]	eval-mlogloss:0.06815	train-mlogloss:0.02757
[140]	eval-mlogloss:0.06803	train-mlogloss:0.02739
[141]	eval-mlogloss:0.06799	train-mlogloss:0.02718
[142]	eval-mlogloss:0.06786	train-mlogloss:0.02702
[143]	eval-mlogloss:0.06770	train-mlogloss:0.02683
[144]	eval-mlogloss:0.06764	train-mlogloss:0.02668
[145]	eval-mlogloss:0.06752	train-mlogloss:0.02655
[146]	eval-mlogloss:0.06746	train-mlogloss:0.02641
[147]	eval-mlogloss:0.06737	train-mlogloss:0.02625
[148]	eval-mlogloss:0.06734	train-mlogloss:0.02609
[149]	eval-mlogloss:0.06731	train-mlogloss:0.02598
[150]	eval-mlogloss:0.06719	train-mlogloss:0.02581
[151]	eval-mlogloss:0.06714	train-mlogloss:0.02570
[152]	eval-mlogloss:0.06706	train-mlogloss:0.02558
[153]	eval-mlogloss:0.06702	train-mlogloss:0.02547
[154]	eval-mlogloss:0.06697	train-mlogloss:0.02537
[155]	eval-mlogloss:0.06694	train-mlogloss:0.02523
[156]	eval-mlogloss:0.06695	train-mlogloss:0.02510
[157]	eval-mlogloss:0.06687	train-mlogloss:0.02498
[158]	eval-mlogloss:0.06679	train-mlogloss:0.02486
[159]	eval-mlogloss:0.06674	train-mlogloss:0.02472
[160]	eval-mlogloss:0.06664	train-mlogloss:0.02458
[161]	eval-mlogloss:0.06655	train-mlogloss:0.02442
[162]	eval-mlogloss:0.06649	train-mlogloss:0.02430
[163]	eval-mlogloss:0.06644	train-mlogloss:0.02421
[164]	eval-mlogloss:0.06642	train-mlogloss:0.02413
```

```
[165]	eval-mlogloss:0.06636	train-mlogloss:0.02402
[166]	eval-mlogloss:0.06628	train-mlogloss:0.02393
[167]	eval-mlogloss:0.06617	train-mlogloss:0.02377
[168]	eval-mlogloss:0.06616	train-mlogloss:0.02363
[169]	eval-mlogloss:0.06615	train-mlogloss:0.02356
[170]	eval-mlogloss:0.06611	train-mlogloss:0.02347
[171]	eval-mlogloss:0.06607	train-mlogloss:0.02337
[172]	eval-mlogloss:0.06600	train-mlogloss:0.02325
[173]	eval-mlogloss:0.06594	train-mlogloss:0.02317
[174]	eval-mlogloss:0.06592	train-mlogloss:0.02307
[175]	eval-mlogloss:0.06585	train-mlogloss:0.02297
[176]	eval-mlogloss:0.06578	train-mlogloss:0.02288
[177]	eval-mlogloss:0.06575	train-mlogloss:0.02280
[178]	eval-mlogloss:0.06570	train-mlogloss:0.02274
[179]	eval-mlogloss:0.06566	train-mlogloss:0.02262
[180]	eval-mlogloss:0.06565	train-mlogloss:0.02259
[181]	eval-mlogloss:0.06565	train-mlogloss:0.02252
[182]	eval-mlogloss:0.06558	train-mlogloss:0.02246
[183]	eval-mlogloss:0.06552	train-mlogloss:0.02238
[184]	eval-mlogloss:0.06551	train-mlogloss:0.02232
[185]	eval-mlogloss:0.06546	train-mlogloss:0.02223
[186]	eval-mlogloss:0.06546	train-mlogloss:0.02220
[187]	eval-mlogloss:0.06541	train-mlogloss:0.02212
[188]	eval-mlogloss:0.06539	train-mlogloss:0.02208
[189]	eval-mlogloss:0.06537	train-mlogloss:0.02201
[190]	eval-mlogloss:0.06537	train-mlogloss:0.02196
[191]	eval-mlogloss:0.06534	train-mlogloss:0.02189
[192]	eval-mlogloss:0.06530	train-mlogloss:0.02184
[193]	eval-mlogloss:0.06524	train-mlogloss:0.02179
[194]	eval-mlogloss:0.06520	train-mlogloss:0.02173
[195]	eval-mlogloss:0.06515	train-mlogloss:0.02167
[196]	eval-mlogloss:0.06510	train-mlogloss:0.02161
[197]	eval-mlogloss:0.06505	train-mlogloss:0.02153
[198]	eval-mlogloss:0.06504	train-mlogloss:0.02149
[199]	eval-mlogloss:0.06499	train-mlogloss:0.02144
[200]	eval-mlogloss:0.06500	train-mlogloss:0.02139
[201]	eval-mlogloss:0.06497	train-mlogloss:0.02131
[202]	eval-mlogloss:0.06493	train-mlogloss:0.02125
[203]	eval-mlogloss:0.06489	train-mlogloss:0.02119
[204]	eval-mlogloss:0.06485	train-mlogloss:0.02113
[205]	eval-mlogloss:0.06478	train-mlogloss:0.02105
[206]	eval-mlogloss:0.06477	train-mlogloss:0.02096
[207]	eval-mlogloss:0.06477	train-mlogloss:0.02091
[208]	eval-mlogloss:0.06478	train-mlogloss:0.02090
[209]	eval-mlogloss:0.06478	train-mlogloss:0.02085
[210]	eval-mlogloss:0.06472	train-mlogloss:0.02078
[211]	eval-mlogloss:0.06473	train-mlogloss:0.02073
[212]	eval-mlogloss:0.06475	train-mlogloss:0.02066
[213]	eval-mlogloss:0.06475	train-mlogloss:0.02063
[214]	eval-mlogloss:0.06474	train-mlogloss:0.02057
[215]	eval-mlogloss:0.06473	train-mlogloss:0.02053
[216]	eval-mlogloss:0.06472	train-mlogloss:0.02048
[217]	eval-mlogloss:0.06471	train-mlogloss:0.02042
[218]	eval-mlogloss:0.06467	train-mlogloss:0.02036
[219]	eval-mlogloss:0.06463	train-mlogloss:0.02030
[220]	eval-mlogloss:0.06462	train-mlogloss:0.02024
[221]	eval-mlogloss:0.06461	train-mlogloss:0.02018
[222]	eval-mlogloss:0.06456	train-mlogloss:0.02012
[223]	eval-mlogloss:0.06452	train-mlogloss:0.02008
[224]	eval-mlogloss:0.06449	train-mlogloss:0.02002
[225]	eval-mlogloss:0.06445	train-mlogloss:0.01995
[226]	eval-mlogloss:0.06444	train-mlogloss:0.01992
[227]	eval-mlogloss:0.06439	train-mlogloss:0.01986
[228]	eval-mlogloss:0.06437	train-mlogloss:0.01982
[229]	eval-mlogloss:0.06434	train-mlogloss:0.01979
```

```
[230]	eval-mlogloss:0.06434	train-mlogloss:0.01974
[231]	eval-mlogloss:0.06434	train-mlogloss:0.01971
[232]	eval-mlogloss:0.06432	train-mlogloss:0.01966
[233]	eval-mlogloss:0.06426	train-mlogloss:0.01962
[234]	eval-mlogloss:0.06423	train-mlogloss:0.01959
[235]	eval-mlogloss:0.06419	train-mlogloss:0.01956
[236]	eval-mlogloss:0.06416	train-mlogloss:0.01952
[237]	eval-mlogloss:0.06417	train-mlogloss:0.01950
[238]	eval-mlogloss:0.06414	train-mlogloss:0.01947
[239]	eval-mlogloss:0.06414	train-mlogloss:0.01942
[240]	eval-mlogloss:0.06413	train-mlogloss:0.01937
[241]	eval-mlogloss:0.06410	train-mlogloss:0.01932
[242]	eval-mlogloss:0.06411	train-mlogloss:0.01928
[243]	eval-mlogloss:0.06408	train-mlogloss:0.01924
[244]	eval-mlogloss:0.06409	train-mlogloss:0.01923
[245]	eval-mlogloss:0.06407	train-mlogloss:0.01919
[246]	eval-mlogloss:0.06405	train-mlogloss:0.01916
[247]	eval-mlogloss:0.06402	train-mlogloss:0.01913
[248]	eval-mlogloss:0.06403	train-mlogloss:0.01908
[249]	eval-mlogloss:0.06403	train-mlogloss:0.01904
[250]	eval-mlogloss:0.06403	train-mlogloss:0.01900
[251]	eval-mlogloss:0.06401	train-mlogloss:0.01897
[252]	eval-mlogloss:0.06398	train-mlogloss:0.01892
[253]	eval-mlogloss:0.06394	train-mlogloss:0.01887
[254]	eval-mlogloss:0.06389	train-mlogloss:0.01883
[255]	eval-mlogloss:0.06388	train-mlogloss:0.01883
[256]	eval-mlogloss:0.06386	train-mlogloss:0.01879
[257]	eval-mlogloss:0.06384	train-mlogloss:0.01878
[258]	eval-mlogloss:0.06385	train-mlogloss:0.01876
[259]	eval-mlogloss:0.06386	train-mlogloss:0.01872
[260]	eval-mlogloss:0.06387	train-mlogloss:0.01868
[261]	eval-mlogloss:0.06386	train-mlogloss:0.01863
[262]	eval-mlogloss:0.06386	train-mlogloss:0.01861
[263]	eval-mlogloss:0.06385	train-mlogloss:0.01860
[264]	eval-mlogloss:0.06384	train-mlogloss:0.01859
[265]	eval-mlogloss:0.06383	train-mlogloss:0.01857
[266]	eval-mlogloss:0.06382	train-mlogloss:0.01854
[267]	eval-mlogloss:0.06380	train-mlogloss:0.01854
[268]	eval-mlogloss:0.06379	train-mlogloss:0.01852
[269]	eval-mlogloss:0.06376	train-mlogloss:0.01850
[270]	eval-mlogloss:0.06377	train-mlogloss:0.01847
[271]	eval-mlogloss:0.06376	train-mlogloss:0.01843
[272]	eval-mlogloss:0.06375	train-mlogloss:0.01841
[273]	eval-mlogloss:0.06375	train-mlogloss:0.01841
[274]	eval-mlogloss:0.06377	train-mlogloss:0.01837
[275]	eval-mlogloss:0.06376	train-mlogloss:0.01837
[276]	eval-mlogloss:0.06375	train-mlogloss:0.01835
[277]	eval-mlogloss:0.06375	train-mlogloss:0.01832
[278]	eval-mlogloss:0.06374	train-mlogloss:0.01831
[279]	eval-mlogloss:0.06373	train-mlogloss:0.01827
[280]	eval-mlogloss:0.06371	train-mlogloss:0.01824
[281]	eval-mlogloss:0.06370	train-mlogloss:0.01823
[282]	eval-mlogloss:0.06368	train-mlogloss:0.01821
[283]	eval-mlogloss:0.06367	train-mlogloss:0.01819
[284]	eval-mlogloss:0.06367	train-mlogloss:0.01816
[285]	eval-mlogloss:0.06366	train-mlogloss:0.01813
[286]	eval-mlogloss:0.06369	train-mlogloss:0.01811
[287]	eval-mlogloss:0.06367	train-mlogloss:0.01808
[288]	eval-mlogloss:0.06366	train-mlogloss:0.01806
[289]	eval-mlogloss:0.06364	train-mlogloss:0.01805
[290]	eval-mlogloss:0.06364	train-mlogloss:0.01802
[291]	eval-mlogloss:0.06363	train-mlogloss:0.01800
[292]	eval-mlogloss:0.06361	train-mlogloss:0.01798
[293]	eval-mlogloss:0.06361	train-mlogloss:0.01795
[294]	eval-mlogloss:0.06360	train-mlogloss:0.01794
```

```
[295]	eval-mlogloss:0.06359	train-mlogloss:0.01792
[296]	eval-mlogloss:0.06357	train-mlogloss:0.01789
[297]	eval-mlogloss:0.06354	train-mlogloss:0.01785
[298]	eval-mlogloss:0.06354	train-mlogloss:0.01783
[299]	eval-mlogloss:0.06356	train-mlogloss:0.01780
[300]	eval-mlogloss:0.06355	train-mlogloss:0.01778
[301]	eval-mlogloss:0.06352	train-mlogloss:0.01777
[302]	eval-mlogloss:0.06350	train-mlogloss:0.01774
[303]	eval-mlogloss:0.06351	train-mlogloss:0.01774
[304]	eval-mlogloss:0.06346	train-mlogloss:0.01769
[305]	eval-mlogloss:0.06344	train-mlogloss:0.01767
[306]	eval-mlogloss:0.06344	train-mlogloss:0.01765
[307]	eval-mlogloss:0.06344	train-mlogloss:0.01763
[308]	eval-mlogloss:0.06344	train-mlogloss:0.01761
[309]	eval-mlogloss:0.06338	train-mlogloss:0.01758
[310]	eval-mlogloss:0.06339	train-mlogloss:0.01755
[311]	eval-mlogloss:0.06339	train-mlogloss:0.01752
[312]	eval-mlogloss:0.06337	train-mlogloss:0.01751
[313]	eval-mlogloss:0.06338	train-mlogloss:0.01751
[314]	eval-mlogloss:0.06337	train-mlogloss:0.01751
[315]	eval-mlogloss:0.06337	train-mlogloss:0.01749
[316]	eval-mlogloss:0.06337	train-mlogloss:0.01747
[317]	eval-mlogloss:0.06338	train-mlogloss:0.01746
[318]	eval-mlogloss:0.06337	train-mlogloss:0.01742
[319]	eval-mlogloss:0.06337	train-mlogloss:0.01740
[320]	eval-mlogloss:0.06336	train-mlogloss:0.01737
[321]	eval-mlogloss:0.06336	train-mlogloss:0.01737
[322]	eval-mlogloss:0.06333	train-mlogloss:0.01735
[323]	eval-mlogloss:0.06333	train-mlogloss:0.01734
[324]	eval-mlogloss:0.06332	train-mlogloss:0.01732
[325]	eval-mlogloss:0.06331	train-mlogloss:0.01732
[326]	eval-mlogloss:0.06332	train-mlogloss:0.01730
[327]	eval-mlogloss:0.06331	train-mlogloss:0.01729
[328]	eval-mlogloss:0.06331	train-mlogloss:0.01727
[329]	eval-mlogloss:0.06330	train-mlogloss:0.01726
[330]	eval-mlogloss:0.06330	train-mlogloss:0.01725
[331]	eval-mlogloss:0.06330	train-mlogloss:0.01723
[332]	eval-mlogloss:0.06331	train-mlogloss:0.01721
[333]	eval-mlogloss:0.06330	train-mlogloss:0.01719
[334]	eval-mlogloss:0.06329	train-mlogloss:0.01718
[335]	eval-mlogloss:0.06328	train-mlogloss:0.01717
[336]	eval-mlogloss:0.06325	train-mlogloss:0.01716
[337]	eval-mlogloss:0.06325	train-mlogloss:0.01713
[338]	eval-mlogloss:0.06325	train-mlogloss:0.01712
[339]	eval-mlogloss:0.06322	train-mlogloss:0.01710
[340]	eval-mlogloss:0.06321	train-mlogloss:0.01710
```

In [ ]:

```python
# Performance metrics
precision = precision_score(y_true=y_test, y_pred=preds)
recall = recall_score(y_true=y_test, y_pred=preds)
accuracy = accuracy_score(y_true=y_test, y_pred=preds)

print(f"Accuracy: {accuracy:.5f}")
print(f"Precision: {precision:.5f}")
print(f"Recall: {recall:.5}")
```

```
Accuracy: 0.97754
Precision: 0.93416
Recall: 0.55911
```