Case Study 7:
Economizing the Prediction of Class 0 and 1

Kebur Fantahun, Eli Kravez, Halle Purdom

April 11, 2022

Introduction

In this study, the customer presents private, anonymous data to create a model that will most accurately predict class 0 or 1. The goal is to minimize monetary loss, where every incorrect prediction loses either $100 or $250 depending on whether it was predicted as class 1 or class 0. Different binary classification models will be explored and tuned to find the one that best minimizes financial loss.

Methods

Data Preparation

The provided dataset has a shape of 160,000 rows and is 136MB in file size. There are 50 attributes and 1 target variable, ‘y’, which is a binary integer of 0 and 1. Fortyfive of the attributes were floats, with 5 categorical columns.

Looking further into these 5 attributes, column x37 represents some monetary value as seen by the dollar sign label. The ‘$’ character was removed from the column and it was converted to a float. The remaining four columns including 'x24', 'x29', 'x30', and 'x32' were categorical, and were one-hot encoded into dummy variables. By looking into the actual columns, these categorical attributes represented country, month, day, and percentages. The percentages column was kept as categorical because there were only 13 unique values in this column, however it could have been converted to numerical also.

The numerical attributes were all scaled with the StandardScaler from sklearn. This was done only to the numerical variables and not the dummy variables as they are already binary entries or 0 and 1 and it would over inflate their values.

There were 1,608 rows with missing data. Because these entries only comprised 1% of the overall data, they were dropped from the dataset.

Lastly, the data was split into 80% training dataset and 20% test data set. The training set will be used to train and tune the models, while the test set will be used solely for model performance evaluation.

Model Development

Accuracy was used to tune and optimize each model, and final evaluation for the models was based on minimizing monetary loss.

Feature Creation

In order to explore feature creation, the PolynomialFeatures function from sklearn was used on the attributes, creating over 1,000 features. Logistic regression was attempted to be used to narrow down these features, but the large size of the dataset alongside the increased number of features crashed the kernel and the computer when trying to narrow down which of these features was important through L1 regression. The ram of the system could not handle the memory of the large amount of columns. Since the huge amount of the interaction terms could not be narrowed down, they were not used in the models.

To reduce the number of added features introduced to the model, only the squared attributes were added to the dataset to see if this would increase performance of the models. These features were added to the dataset, then L1 was used to narrow down the most important features. Only the most important attributes were used in the data. This larger dataset was used to train and test all the models, however across the board the models increased monetary loss. Because of this, the squared features were not used in the final models, and instead only the original 79 attributes were used.

Random Forest Development

The first model created for this task was a Random Forest binary classification model. In order to optimize the model on the train data, 5-fold cross validation was used in a random search to find the model with the highest accuracy. The parameters tuned included max_depth, max_features, min_samples_leaf, min_samples_split, class_weight, and criterion. When the optimal parameter chosen for the model was the maximum value possible from the search, the parameter range was expanded to ensure no better value existed outside of the range. For the random search, n_estimators was set at 100, then scaled to 1,000 for the final model to minimize runtime.

XGBoost Development

For XGBoost, the train datasets were first converted to dmatrices in order to use the XGBoost package. In order to optimize the model on the train data, 5-fold cross validation was used in a manual random search to find the model with the lowest log loss for the validation data. The

parameters tuned included subsample, colsample_bytree, max_depth, min_child_weight, and gamma. Early stopping for the XGBoost model was set to two rounds, with num_round set to 1,000 but usually stopping around 750.

Neural Network Development

The neural network is built out of 7 layers, 1 input and 6 dense. Since the data has already been scaled with the standard scaler, BatchNormalization is not necessary. BatchNormalization was tested to see if accuracy would increase but it turned out that it did not help. The use of BatchNormalization caused the train and test loss to behave poorly. For early stopping, the monitored statistic was validation loss and patience was set to three. The validation set was created by splitting the training set into an 80% train and 20% validation set. Batch size and epochs were both set to 1,000, and early stopping usually only ran the neural net for about 10-15 epochs.

Ensemble Development

Many different forms of ensembles were tested, including different two level and three level ensembles. Models included in these levels were Random Forest, Logistic Regression, Neural Networks, and XGBoost Classifiers. The final level was tested between Linear Regressors and XGBoost Classifiers. The main issue with testing different ensembles is the extremely long runtimes. This greatly limited the number and variety of models that could be tested. The individual models within the layers were also tested to see if optimization would improve the overall ensemble, but in each case little to no improvement was seen for a cost of much runtime.

Results

Random Forest Final Model

The final Random Forest model scaled n_estimators up to 1,000, and its parameters can be seen below:

RandomForestClassifier(class_weight='balanced', criterion='entropy',max_depth=50, max_features=35, min_samples_leaf=4, min_samples_split=8, n_estimators=1000)

XGBoost Classifier Final Model

The final XGBoost Classifier model used the following parameters:

final_params = {'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05, 'subsample': 0.5792877172818175, 'colsample_bytree': 0.7680400747721351, 'max_depth': 20, 'min_child_weight': 5, 'gamma': 1.5}

For the final model, num_round was set to 1,000 and early stopping set to two. The model usually stops running at num_round 743.

Neural Network Final Model

The final Neural Network model uses the adam optimizer, binary cross entropy loss, a batch size and epochs set to 1,000, and early stopping patience set to 3 and monitor set to validation loss. The layers of the network can be seen below:

Layer	Neurons	Activation
Input	79	relu
Dense	500	relu
Dense	400	relu
Dense	300	relu
Dense	200	relu
Dense	100	relu
Dense (Output)	1	sigmoid

Ensemble Final Model

The best ensemble model was built with tuned Random Forest and tuned Logistic Regression models in layer 1, Random Forest and Logistic Regression models in layer 2, and an XGBoost Classifier for the final layer.

Model Performance Evaluation

Model	Monetary Loss per Prediction
Random Forest	12.8856
XGBoost	10.1866
Neural Network	8.4522
Ensemble	12.715

Conclusion

In conclusion, a collection of models (Random Forest, XGBoost, Neural Network, and an Ensemble of all previous) were built to minimize the incorrect predictions of the 2 given classes or to maximize the correct prediction of the 2 given classes. Monetary loss for the customer was minimized by optimizing the models to correctly predict the target classes. The model that successfully minimized the monetary loss the most was the Neural Network, with a loss of $8.45 per prediction.

Data Prep

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns


import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import cross_val_score, StratifiedKFold, KFold, cro
from sklearn.metrics import precision_score, recall_score, confusion_matrix, acc
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import plot_confusion_matrix

import random

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

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LinearRegression

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import accuracy_score
```

In [ ]:
```
data = pd.read_csv("/content/drive/MyDrive/SMU Work/Quantifying the World/Case S
# data = pd.read_csv("/Users/hallepurdom/Desktop/SMU/Classes/Quantifying the Wor
data.head()
```

In [ ]:

```
data.shape
```

In [ ]:
```
data.duplicated().value_counts()
```

In [ ]:
```
# Visualization function for the columns
def visualize_counts(col_name, df, f_heigh = 15, f_width=10, title= "", f_rotati
    plt.figure(figsize=(f_heigh, f_width))
    axis_font = {'fontname':'Arial', 'size':'24'}
    ax = sns.countplot(x=col_name, data=df)
    plt.xlabel(col_name, **axis_font)
    plt.ylabel("count", **axis_font)
    plt.title(title, **axis_font)
    plt.xticks(rotation = f_rotation)

    for label in (ax.get_xticklabels() + ax.get_yticklabels()):
        label.set_fontname('Arial')
        label.set_fontsize(18)

    plt.show()

visualize_counts('y', data, title='Target Distribution')
```

In [ ]:
```
# Missing Values, Column types
data_types = data.dtypes
missing_value_stats = ((data.isnull().sum()/len(data)*100))
unique_values = data.apply(lambda column: column.unique().shape[0])

basic_stats = pd.concat([data_types, unique_values, missing_value_stats], axis =
basic_stats.columns = ['type', 'unique values', 'missing percent']
basic_stats = basic_stats.sort_values('missing percent', ascending = False)
print(basic_stats)
```

In [ ]:
```
msno.matrix(data)
```

In [ ]:
```
data = data.dropna()
data.shape
```

In [ ]:
```
#Convert column x37 to numeric
data['x37'] = data['x37'].str[1:]
data['x37'] = data['x37'].astype('float')
data['x37']
```

In [ ]:
```
# reindex because concat function was combining incorrectly after missing values
data = data.reset_index()
data
```

In [ ]:
```
numerical_cols = data._get_numeric_data().columns
categorical_cols=  [i for i in data.columns if data.dtypes[i]=='object' ]
```

```
print(f"numerical columns: \n {numerical_cols}")
print(f"categorical columns: \n {categorical_cols}")
```

In [ ]:
```
# Normalize numeric variables with StandardScaler() -- W/OUT FEATURE CREATION
y = data['y']

numerical_cols_df = data[numerical_cols].copy().drop('y', axis=1)
scaler = StandardScaler()

numerical_cols_standard = scaler.fit_transform(numerical_cols_df)
numerical_cols_standard = pd.DataFrame(data = numerical_cols_standard, columns =

### FEATURE CREATION - squared features (no interaction terms bc ram limit)
# y = data['y']

# numerical_cols_df = data[numerical_cols].copy().drop('y', axis=1)

# for col in numerical_cols_df.columns:
#     numerical_cols_df[col + "_2"] = numerical_cols_df[col]*numerical_cols_df[c

# scaler = StandardScaler()

# numerical_cols_standard = scaler.fit_transform(numerical_cols_df)
# numerical_cols_standard = pd.DataFrame(data = numerical_cols_standard, columns
```

In [ ]:
```
# Create dummy variables for categorical features - One hot encoding
categorical_features = pd.get_dummies(data[categorical_cols])
```

In [ ]:
```
# Recombine dataset
data_processed = pd.concat([numerical_cols_standard, categorical_features], axis
```

In [ ]:
```
data_processed.shape
```

In [ ]:
```
# train test split
X_train, X_test, y_train, y_test = train_test_split(data_processed, y, test_size
```

Feature Creation and Logistic Regression

In [ ]:
```
# param = {'C': [10**-2,10**-1,10**0,10**1,10**2]}

# lr_model = LogisticRegression(penalty='l1', solver='liblinear')
# gs_model = GridSearchCV(estimator=lr_model, param_grid=param)
# gs_model.fit(X_train, y_train)

# # Train a LR model with best parameters
# model = LogisticRegression(**gs_model.best_params_, penalty='l1', solver='libl
# model.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression(C=1, penalty='l1', solver='liblinear')
```

In [ ]:
```
# Train a LR model with best parameters
# model = LogisticRegression(**gs_model.best_params_, penalty='l1', solver='libl
# model.fit(X_train, y_train)
```

Out[ ]:
```
LogisticRegression(C=1, penalty='l1', solver='liblinear')
```

In [ ]:
```
# set new relevant features for train/test

# coef = model.coef_[0]
# imp_features = pd.Series(data_processed.columns)[list(coef!=0)]
# X_train = X_train[imp_features]
# X_test = X_test[imp_features]
```

Random Forest

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

param_grid = {
    'max_depth': [35, 40, 50],# [9,10, 15, 20, 25, 30 , 35],
    'max_features': [25, 35, 50],# [10,15, 20, 25],
    'min_samples_leaf': [3, 4, 7], # [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 [ ]:
```
# Uncomment to run random search
# 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': [35, 40, 50],
                                        'max_features': [25, 35, 50],
                                        'min_samples_leaf': [3, 4, 7],
                                        'min_samples_split': [8, 10, 12],
                                        'n_estimators': [100]},
                   scoring='accuracy', verbose=2)
```

In [ ]:
```
# Uncomment if running random search
# Best parameters
# print(grid_search.best_estimator_)
# print(grid_search.best_score_)
```

```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=50, max_features=35, min_samples_leaf=4,
                       min_samples_split=8)
0.9275686192494458
```

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

Out[ ]:
```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=50, max_features=35, min_samples_leaf=4,
                       min_samples_split=8, n_estimators=1000)
```

In [ ]:
```
# Performance statistics 0.93172 accuracy
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: 0.93150
Precision: 0.92826
Recall: 0.90031
```

In [ ]:
```
confusion_matrix_rf = confusion_matrix(y_test, preds)
print(confusion_matrix_rf)
print((confusion_matrix_rf[0,1]*100 + confusion_matrix_rf[1,0]*250)/len(y_test))
```

```
[[17967   892]
 [ 1278 11542]]
12.901291076107201
```

RandomForestClassifier(class_weight='balanced_subsample',
criterion='entropy',max_depth=35, max_features=25,
min_samples_leaf=4,min_samples_split=10, n_estimators=1000)
Accuracy: 0.93077
Precision: 0.92696
Recall: 0.89984

RandomForestClassifier(class_weight='balanced', criterion='entropy',max_depth=50,
max_features=35, min_samples_leaf=4, min_samples_split=8, n_estimators=1000)
Accuracy: 0.93178
Precision: 0.92811
Recall: 0.90125

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 [ ]:

```
# Uncomment to run random search
# 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=Fal
#     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': 0.5629982928352992, 'colsample_bytree': 0.03643859527751492, 'max_d
epth': 3, 'min_child_weight': 5, 'gamma': 0.5}
      train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
1999              0.45507            0.002683            0.475331

      test-mlogloss-std
1999           0.003982
================================================
_____________________done_____________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.6254736809223108, 'colsample_bytree': 0.03169912253713203, 'max_d
epth': 10, 'min_child_weight': 10, 'gamma': 2}
      train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
1313             0.390581            0.003215            0.485198

      test-mlogloss-std
1313           0.005623
================================================
_____________________done_____________________
```

```
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.5792877172818175, 'colsample_bytree': 0.7680400747721351, 'max_de
pth': 20, 'min_child_weight': 5, 'gamma': 1.5}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
507             0.050257            0.000344            0.156774

     test-mlogloss-std
507           0.002705
================================================
_________________________done_____________________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.2154988709032628, 'colsample_bytree': 0.35378532017795794, 'max_d
epth': 15, 'min_child_weight': 10, 'gamma': 1}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
645             0.082731            0.000448            0.197044

     test-mlogloss-std
645           0.002947
================================================
_________________________done_____________________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.4380425538499748, 'colsample_bytree': 0.6255799676269455, 'max_de
pth': 40, 'min_child_weight': 5, 'gamma': 1}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
447              0.04278            0.000155            0.162993

     test-mlogloss-std
447            0.00327
================================================
_________________________done_____________________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.67426081615934, 'colsample_bytree': 0.6695681612228545, 'max_dept
h': 5, 'min_child_weight': 10, 'gamma': 1.5}
      train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
1031             0.172172            0.000798            0.214375

      test-mlogloss-std
1031           0.003184
================================================
_________________________done_____________________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.44028850038384393, 'colsample_bytree': 0.7428093558620357, 'max_d
epth': 40, 'min_child_weight': 10, 'gamma': 2}
     train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
465             0.073296            0.000212            0.164742

     test-mlogloss-std
465           0.002814
================================================
_________________________done_____________________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.4169617908893525, 'colsample_bytree': 0.1256871684760852, 'max_de
pth': 3, 'min_child_weight': 5, 'gamma': 0.5}
      train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
1999             0.297244            0.000528            0.319033

      test-mlogloss-std
1999           0.004782
================================================
_________________________done_____________________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.6087373301330863, 'colsample_bytree': 0.3344551174016209, 'max_de
pth': 3, 'min_child_weight': 10, 'gamma': 0.5}
      train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
1999             0.246245            0.001452            0.270635
```

```
      test-mlogloss-std
1999           0.002735
================================================
_________________________done_____________________
{'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05,
'subsample': 0.3272549518122969, 'colsample_bytree': 0.14665633169460623, 'max_d
epth': 5, 'min_child_weight': 1, 'gamma': 5}
      train-mlogloss-mean  train-mlogloss-std  test-mlogloss-mean  \
1999              0.25014            0.001454            0.265591

      test-mlogloss-std
1999           0.004463
================================================
```

In [ ]:

```
#Best model test-mlogloss-mean: 0.156774
final_params =  {'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class':

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

#32 min
# round 743
```

```
[21:35:12] 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:0.66345   train-mlogloss:0.66021
[1]     eval-mlogloss:0.63752   train-mlogloss:0.63079
[2]     eval-mlogloss:0.61255   train-mlogloss:0.60273
[3]     eval-mlogloss:0.58881   train-mlogloss:0.57636
[4]     eval-mlogloss:0.56908   train-mlogloss:0.55382
[5]     eval-mlogloss:0.54854   train-mlogloss:0.53092
[6]     eval-mlogloss:0.52963   train-mlogloss:0.50957
[7]     eval-mlogloss:0.51269   train-mlogloss:0.49019
[8]     eval-mlogloss:0.49507   train-mlogloss:0.47066
[9]     eval-mlogloss:0.48001   train-mlogloss:0.45351
[10]    eval-mlogloss:0.46456   train-mlogloss:0.43627
[11]    eval-mlogloss:0.45111   train-mlogloss:0.42087
[12]    eval-mlogloss:0.43836   train-mlogloss:0.40625
[13]    eval-mlogloss:0.42625   train-mlogloss:0.39245
[14]    eval-mlogloss:0.41441   train-mlogloss:0.37908
[15]    eval-mlogloss:0.40356   train-mlogloss:0.36671
[16]    eval-mlogloss:0.39291   train-mlogloss:0.35472
[17]    eval-mlogloss:0.38381   train-mlogloss:0.34401
[18]    eval-mlogloss:0.37463   train-mlogloss:0.33346
[19]    eval-mlogloss:0.36600   train-mlogloss:0.32350
[20]    eval-mlogloss:0.35789   train-mlogloss:0.31420
[21]    eval-mlogloss:0.35073   train-mlogloss:0.30575
[22]    eval-mlogloss:0.34368   train-mlogloss:0.29743
[23]    eval-mlogloss:0.33652   train-mlogloss:0.28913
[24]    eval-mlogloss:0.32981   train-mlogloss:0.28135
[25]    eval-mlogloss:0.32323   train-mlogloss:0.27381
[26]    eval-mlogloss:0.31708   train-mlogloss:0.26661
[27]    eval-mlogloss:0.31158   train-mlogloss:0.26011
[28]    eval-mlogloss:0.30571   train-mlogloss:0.25331
[29]    eval-mlogloss:0.30024   train-mlogloss:0.24690
[30]    eval-mlogloss:0.29544   train-mlogloss:0.24115
[31]    eval-mlogloss:0.29054   train-mlogloss:0.23533
[32]    eval-mlogloss:0.28579   train-mlogloss:0.22970
```

```
[33]    eval-mlogloss:0.28137   train-mlogloss:0.22452
[34]    eval-mlogloss:0.27666   train-mlogloss:0.21909
[35]    eval-mlogloss:0.27305   train-mlogloss:0.21460
[36]    eval-mlogloss:0.26911   train-mlogloss:0.20998
[37]    eval-mlogloss:0.26522   train-mlogloss:0.20549
[38]    eval-mlogloss:0.26152   train-mlogloss:0.20111
[39]    eval-mlogloss:0.25793   train-mlogloss:0.19695
[40]    eval-mlogloss:0.25519   train-mlogloss:0.19357
[41]    eval-mlogloss:0.25193   train-mlogloss:0.18971
[42]    eval-mlogloss:0.24888   train-mlogloss:0.18603
[43]    eval-mlogloss:0.24583   train-mlogloss:0.18241
[44]    eval-mlogloss:0.24311   train-mlogloss:0.17909
[45]    eval-mlogloss:0.24034   train-mlogloss:0.17578
[46]    eval-mlogloss:0.23784   train-mlogloss:0.17272
[47]    eval-mlogloss:0.23542   train-mlogloss:0.16974
[48]    eval-mlogloss:0.23306   train-mlogloss:0.16692
[49]    eval-mlogloss:0.23073   train-mlogloss:0.16401
[50]    eval-mlogloss:0.22872   train-mlogloss:0.16147
[51]    eval-mlogloss:0.22650   train-mlogloss:0.15883
[52]    eval-mlogloss:0.22429   train-mlogloss:0.15614
[53]    eval-mlogloss:0.22201   train-mlogloss:0.15351
[54]    eval-mlogloss:0.21978   train-mlogloss:0.15098
[55]    eval-mlogloss:0.21760   train-mlogloss:0.14842
[56]    eval-mlogloss:0.21592   train-mlogloss:0.14636
[57]    eval-mlogloss:0.21429   train-mlogloss:0.14434
[58]    eval-mlogloss:0.21272   train-mlogloss:0.14238
[59]    eval-mlogloss:0.21100   train-mlogloss:0.14035
[60]    eval-mlogloss:0.20933   train-mlogloss:0.13831
[61]    eval-mlogloss:0.20792   train-mlogloss:0.13646
[62]    eval-mlogloss:0.20623   train-mlogloss:0.13443
[63]    eval-mlogloss:0.20485   train-mlogloss:0.13268
[64]    eval-mlogloss:0.20353   train-mlogloss:0.13110
[65]    eval-mlogloss:0.20191   train-mlogloss:0.12928
[66]    eval-mlogloss:0.20066   train-mlogloss:0.12772
[67]    eval-mlogloss:0.19943   train-mlogloss:0.12617
[68]    eval-mlogloss:0.19819   train-mlogloss:0.12463
[69]    eval-mlogloss:0.19684   train-mlogloss:0.12305
[70]    eval-mlogloss:0.19579   train-mlogloss:0.12175
[71]    eval-mlogloss:0.19483   train-mlogloss:0.12050
[72]    eval-mlogloss:0.19371   train-mlogloss:0.11909
[73]    eval-mlogloss:0.19244   train-mlogloss:0.11758
[74]    eval-mlogloss:0.19141   train-mlogloss:0.11625
[75]    eval-mlogloss:0.19039   train-mlogloss:0.11497
[76]    eval-mlogloss:0.18937   train-mlogloss:0.11363
[77]    eval-mlogloss:0.18843   train-mlogloss:0.11249
[78]    eval-mlogloss:0.18755   train-mlogloss:0.11137
[79]    eval-mlogloss:0.18663   train-mlogloss:0.11036
[80]    eval-mlogloss:0.18592   train-mlogloss:0.10934
[81]    eval-mlogloss:0.18507   train-mlogloss:0.10823
[82]    eval-mlogloss:0.18422   train-mlogloss:0.10712
[83]    eval-mlogloss:0.18343   train-mlogloss:0.10618
[84]    eval-mlogloss:0.18260   train-mlogloss:0.10516
[85]    eval-mlogloss:0.18191   train-mlogloss:0.10423
[86]    eval-mlogloss:0.18137   train-mlogloss:0.10353
[87]    eval-mlogloss:0.18053   train-mlogloss:0.10245
[88]    eval-mlogloss:0.17984   train-mlogloss:0.10156
[89]    eval-mlogloss:0.17922   train-mlogloss:0.10074
[90]    eval-mlogloss:0.17832   train-mlogloss:0.09970
[91]    eval-mlogloss:0.17769   train-mlogloss:0.09890
[92]    eval-mlogloss:0.17722   train-mlogloss:0.09821
[93]    eval-mlogloss:0.17672   train-mlogloss:0.09754
[94]    eval-mlogloss:0.17613   train-mlogloss:0.09674
[95]    eval-mlogloss:0.17568   train-mlogloss:0.09604
[96]    eval-mlogloss:0.17526   train-mlogloss:0.09551
[97]    eval-mlogloss:0.17459   train-mlogloss:0.09476
```

```
[98]    eval-mlogloss:0.17397   train-mlogloss:0.09402
[99]    eval-mlogloss:0.17358   train-mlogloss:0.09356
[100]   eval-mlogloss:0.17321   train-mlogloss:0.09309
[101]   eval-mlogloss:0.17284   train-mlogloss:0.09257
[102]   eval-mlogloss:0.17245   train-mlogloss:0.09202
[103]   eval-mlogloss:0.17217   train-mlogloss:0.09167
[104]   eval-mlogloss:0.17193   train-mlogloss:0.09126
[105]   eval-mlogloss:0.17152   train-mlogloss:0.09073
[106]   eval-mlogloss:0.17114   train-mlogloss:0.09030
[107]   eval-mlogloss:0.17070   train-mlogloss:0.08972
[108]   eval-mlogloss:0.17034   train-mlogloss:0.08920
[109]   eval-mlogloss:0.17000   train-mlogloss:0.08871
[110]   eval-mlogloss:0.16975   train-mlogloss:0.08840
[111]   eval-mlogloss:0.16950   train-mlogloss:0.08807
[112]   eval-mlogloss:0.16928   train-mlogloss:0.08770
[113]   eval-mlogloss:0.16884   train-mlogloss:0.08708
[114]   eval-mlogloss:0.16848   train-mlogloss:0.08653
[115]   eval-mlogloss:0.16826   train-mlogloss:0.08611
[116]   eval-mlogloss:0.16796   train-mlogloss:0.08567
[117]   eval-mlogloss:0.16772   train-mlogloss:0.08538
[118]   eval-mlogloss:0.16748   train-mlogloss:0.08490
[119]   eval-mlogloss:0.16724   train-mlogloss:0.08451
[120]   eval-mlogloss:0.16703   train-mlogloss:0.08412
[121]   eval-mlogloss:0.16674   train-mlogloss:0.08370
[122]   eval-mlogloss:0.16662   train-mlogloss:0.08343
[123]   eval-mlogloss:0.16651   train-mlogloss:0.08319
[124]   eval-mlogloss:0.16621   train-mlogloss:0.08279
[125]   eval-mlogloss:0.16596   train-mlogloss:0.08239
[126]   eval-mlogloss:0.16591   train-mlogloss:0.08219
[127]   eval-mlogloss:0.16570   train-mlogloss:0.08188
[128]   eval-mlogloss:0.16557   train-mlogloss:0.08165
[129]   eval-mlogloss:0.16539   train-mlogloss:0.08130
[130]   eval-mlogloss:0.16531   train-mlogloss:0.08112
[131]   eval-mlogloss:0.16509   train-mlogloss:0.08077
[132]   eval-mlogloss:0.16489   train-mlogloss:0.08044
[133]   eval-mlogloss:0.16469   train-mlogloss:0.08011
[134]   eval-mlogloss:0.16459   train-mlogloss:0.07991
[135]   eval-mlogloss:0.16439   train-mlogloss:0.07962
[136]   eval-mlogloss:0.16433   train-mlogloss:0.07941
[137]   eval-mlogloss:0.16426   train-mlogloss:0.07922
[138]   eval-mlogloss:0.16411   train-mlogloss:0.07894
[139]   eval-mlogloss:0.16394   train-mlogloss:0.07862
[140]   eval-mlogloss:0.16376   train-mlogloss:0.07832
[141]   eval-mlogloss:0.16373   train-mlogloss:0.07814
[142]   eval-mlogloss:0.16371   train-mlogloss:0.07792
[143]   eval-mlogloss:0.16353   train-mlogloss:0.07764
[144]   eval-mlogloss:0.16329   train-mlogloss:0.07733
[145]   eval-mlogloss:0.16315   train-mlogloss:0.07711
[146]   eval-mlogloss:0.16315   train-mlogloss:0.07699
[147]   eval-mlogloss:0.16307   train-mlogloss:0.07676
[148]   eval-mlogloss:0.16302   train-mlogloss:0.07657
[149]   eval-mlogloss:0.16284   train-mlogloss:0.07624
[150]   eval-mlogloss:0.16275   train-mlogloss:0.07597
[151]   eval-mlogloss:0.16275   train-mlogloss:0.07579
[152]   eval-mlogloss:0.16257   train-mlogloss:0.07552
[153]   eval-mlogloss:0.16254   train-mlogloss:0.07547
[154]   eval-mlogloss:0.16241   train-mlogloss:0.07522
[155]   eval-mlogloss:0.16240   train-mlogloss:0.07509
[156]   eval-mlogloss:0.16238   train-mlogloss:0.07497
[157]   eval-mlogloss:0.16228   train-mlogloss:0.07479
[158]   eval-mlogloss:0.16227   train-mlogloss:0.07467
[159]   eval-mlogloss:0.16223   train-mlogloss:0.07453
[160]   eval-mlogloss:0.16219   train-mlogloss:0.07440
[161]   eval-mlogloss:0.16204   train-mlogloss:0.07412
[162]   eval-mlogloss:0.16204   train-mlogloss:0.07399
```

```
[163]   eval-mlogloss:0.16204   train-mlogloss:0.07386
[164]   eval-mlogloss:0.16202   train-mlogloss:0.07373
[165]   eval-mlogloss:0.16198   train-mlogloss:0.07356
[166]   eval-mlogloss:0.16185   train-mlogloss:0.07333
[167]   eval-mlogloss:0.16178   train-mlogloss:0.07315
[168]   eval-mlogloss:0.16169   train-mlogloss:0.07292
[169]   eval-mlogloss:0.16157   train-mlogloss:0.07270
[170]   eval-mlogloss:0.16155   train-mlogloss:0.07257
[171]   eval-mlogloss:0.16149   train-mlogloss:0.07240
[172]   eval-mlogloss:0.16143   train-mlogloss:0.07224
[173]   eval-mlogloss:0.16132   train-mlogloss:0.07203
[174]   eval-mlogloss:0.16111   train-mlogloss:0.07178
[175]   eval-mlogloss:0.16105   train-mlogloss:0.07157
[176]   eval-mlogloss:0.16097   train-mlogloss:0.07141
[177]   eval-mlogloss:0.16083   train-mlogloss:0.07118
[178]   eval-mlogloss:0.16075   train-mlogloss:0.07098
[179]   eval-mlogloss:0.16073   train-mlogloss:0.07084
[180]   eval-mlogloss:0.16069   train-mlogloss:0.07070
[181]   eval-mlogloss:0.16050   train-mlogloss:0.07045
[182]   eval-mlogloss:0.16050   train-mlogloss:0.07035
[183]   eval-mlogloss:0.16051   train-mlogloss:0.07021
[184]   eval-mlogloss:0.16032   train-mlogloss:0.06994
[185]   eval-mlogloss:0.16030   train-mlogloss:0.06985
[186]   eval-mlogloss:0.16019   train-mlogloss:0.06961
[187]   eval-mlogloss:0.16018   train-mlogloss:0.06950
[188]   eval-mlogloss:0.16015   train-mlogloss:0.06938
[189]   eval-mlogloss:0.15985   train-mlogloss:0.06903
[190]   eval-mlogloss:0.15977   train-mlogloss:0.06888
[191]   eval-mlogloss:0.15969   train-mlogloss:0.06871
[192]   eval-mlogloss:0.15959   train-mlogloss:0.06855
[193]   eval-mlogloss:0.15957   train-mlogloss:0.06844
[194]   eval-mlogloss:0.15948   train-mlogloss:0.06822
[195]   eval-mlogloss:0.15943   train-mlogloss:0.06807
[196]   eval-mlogloss:0.15927   train-mlogloss:0.06781
[197]   eval-mlogloss:0.15925   train-mlogloss:0.06771
[198]   eval-mlogloss:0.15922   train-mlogloss:0.06758
[199]   eval-mlogloss:0.15921   train-mlogloss:0.06746
[200]   eval-mlogloss:0.15918   train-mlogloss:0.06729
[201]   eval-mlogloss:0.15907   train-mlogloss:0.06711
[202]   eval-mlogloss:0.15896   train-mlogloss:0.06685
[203]   eval-mlogloss:0.15893   train-mlogloss:0.06671
[204]   eval-mlogloss:0.15888   train-mlogloss:0.06656
[205]   eval-mlogloss:0.15885   train-mlogloss:0.06646
[206]   eval-mlogloss:0.15880   train-mlogloss:0.06631
[207]   eval-mlogloss:0.15874   train-mlogloss:0.06613
[208]   eval-mlogloss:0.15873   train-mlogloss:0.06601
[209]   eval-mlogloss:0.15870   train-mlogloss:0.06588
[210]   eval-mlogloss:0.15866   train-mlogloss:0.06575
[211]   eval-mlogloss:0.15865   train-mlogloss:0.06566
[212]   eval-mlogloss:0.15851   train-mlogloss:0.06547
[213]   eval-mlogloss:0.15844   train-mlogloss:0.06534
[214]   eval-mlogloss:0.15843   train-mlogloss:0.06524
[215]   eval-mlogloss:0.15841   train-mlogloss:0.06513
[216]   eval-mlogloss:0.15830   train-mlogloss:0.06495
[217]   eval-mlogloss:0.15818   train-mlogloss:0.06478
[218]   eval-mlogloss:0.15817   train-mlogloss:0.06470
[219]   eval-mlogloss:0.15815   train-mlogloss:0.06462
[220]   eval-mlogloss:0.15810   train-mlogloss:0.06453
[221]   eval-mlogloss:0.15812   train-mlogloss:0.06448
[222]   eval-mlogloss:0.15803   train-mlogloss:0.06429
[223]   eval-mlogloss:0.15802   train-mlogloss:0.06418
[224]   eval-mlogloss:0.15801   train-mlogloss:0.06410
[225]   eval-mlogloss:0.15795   train-mlogloss:0.06397
[226]   eval-mlogloss:0.15788   train-mlogloss:0.06385
[227]   eval-mlogloss:0.15785   train-mlogloss:0.06374
```

```
[228]   eval-mlogloss:0.15779   train-mlogloss:0.06363
[229]   eval-mlogloss:0.15773   train-mlogloss:0.06348
[230]   eval-mlogloss:0.15770   train-mlogloss:0.06337
[231]   eval-mlogloss:0.15768   train-mlogloss:0.06323
[232]   eval-mlogloss:0.15766   train-mlogloss:0.06318
[233]   eval-mlogloss:0.15768   train-mlogloss:0.06309
[234]   eval-mlogloss:0.15768   train-mlogloss:0.06301
[235]   eval-mlogloss:0.15768   train-mlogloss:0.06291
[236]   eval-mlogloss:0.15761   train-mlogloss:0.06280
[237]   eval-mlogloss:0.15760   train-mlogloss:0.06271
[238]   eval-mlogloss:0.15759   train-mlogloss:0.06265
[239]   eval-mlogloss:0.15754   train-mlogloss:0.06254
[240]   eval-mlogloss:0.15753   train-mlogloss:0.06245
[241]   eval-mlogloss:0.15754   train-mlogloss:0.06239
[242]   eval-mlogloss:0.15754   train-mlogloss:0.06237
[243]   eval-mlogloss:0.15743   train-mlogloss:0.06222
[244]   eval-mlogloss:0.15743   train-mlogloss:0.06216
[245]   eval-mlogloss:0.15742   train-mlogloss:0.06209
[246]   eval-mlogloss:0.15729   train-mlogloss:0.06194
[247]   eval-mlogloss:0.15730   train-mlogloss:0.06186
[248]   eval-mlogloss:0.15730   train-mlogloss:0.06180
[249]   eval-mlogloss:0.15732   train-mlogloss:0.06174
[250]   eval-mlogloss:0.15728   train-mlogloss:0.06163
[251]   eval-mlogloss:0.15723   train-mlogloss:0.06153
[252]   eval-mlogloss:0.15722   train-mlogloss:0.06147
[253]   eval-mlogloss:0.15719   train-mlogloss:0.06134
[254]   eval-mlogloss:0.15717   train-mlogloss:0.06123
[255]   eval-mlogloss:0.15713   train-mlogloss:0.06112
[256]   eval-mlogloss:0.15712   train-mlogloss:0.06105
[257]   eval-mlogloss:0.15712   train-mlogloss:0.06099
[258]   eval-mlogloss:0.15709   train-mlogloss:0.06094
[259]   eval-mlogloss:0.15704   train-mlogloss:0.06085
[260]   eval-mlogloss:0.15703   train-mlogloss:0.06078
[261]   eval-mlogloss:0.15704   train-mlogloss:0.06072
[262]   eval-mlogloss:0.15699   train-mlogloss:0.06061
[263]   eval-mlogloss:0.15698   train-mlogloss:0.06054
[264]   eval-mlogloss:0.15697   train-mlogloss:0.06048
[265]   eval-mlogloss:0.15696   train-mlogloss:0.06041
[266]   eval-mlogloss:0.15694   train-mlogloss:0.06033
[267]   eval-mlogloss:0.15694   train-mlogloss:0.06027
[268]   eval-mlogloss:0.15692   train-mlogloss:0.06023
[269]   eval-mlogloss:0.15691   train-mlogloss:0.06015
[270]   eval-mlogloss:0.15689   train-mlogloss:0.06008
[271]   eval-mlogloss:0.15688   train-mlogloss:0.06000
[272]   eval-mlogloss:0.15686   train-mlogloss:0.05993
[273]   eval-mlogloss:0.15682   train-mlogloss:0.05980
[274]   eval-mlogloss:0.15681   train-mlogloss:0.05974
[275]   eval-mlogloss:0.15680   train-mlogloss:0.05967
[276]   eval-mlogloss:0.15678   train-mlogloss:0.05959
[277]   eval-mlogloss:0.15677   train-mlogloss:0.05951
[278]   eval-mlogloss:0.15670   train-mlogloss:0.05936
[279]   eval-mlogloss:0.15662   train-mlogloss:0.05926
[280]   eval-mlogloss:0.15659   train-mlogloss:0.05917
[281]   eval-mlogloss:0.15655   train-mlogloss:0.05905
[282]   eval-mlogloss:0.15654   train-mlogloss:0.05900
[283]   eval-mlogloss:0.15643   train-mlogloss:0.05882
[284]   eval-mlogloss:0.15634   train-mlogloss:0.05866
[285]   eval-mlogloss:0.15626   train-mlogloss:0.05856
[286]   eval-mlogloss:0.15621   train-mlogloss:0.05847
[287]   eval-mlogloss:0.15620   train-mlogloss:0.05838
[288]   eval-mlogloss:0.15613   train-mlogloss:0.05828
[289]   eval-mlogloss:0.15611   train-mlogloss:0.05818
[290]   eval-mlogloss:0.15611   train-mlogloss:0.05810
[291]   eval-mlogloss:0.15611   train-mlogloss:0.05806
[292]   eval-mlogloss:0.15611   train-mlogloss:0.05802
```

```
[293]   eval-mlogloss:0.15611   train-mlogloss:0.05796
[294]   eval-mlogloss:0.15610   train-mlogloss:0.05789
[295]   eval-mlogloss:0.15606   train-mlogloss:0.05778
[296]   eval-mlogloss:0.15606   train-mlogloss:0.05767
[297]   eval-mlogloss:0.15604   train-mlogloss:0.05762
[298]   eval-mlogloss:0.15604   train-mlogloss:0.05756
[299]   eval-mlogloss:0.15603   train-mlogloss:0.05749
[300]   eval-mlogloss:0.15599   train-mlogloss:0.05740
[301]   eval-mlogloss:0.15597   train-mlogloss:0.05734
[302]   eval-mlogloss:0.15594   train-mlogloss:0.05728
[303]   eval-mlogloss:0.15593   train-mlogloss:0.05722
[304]   eval-mlogloss:0.15586   train-mlogloss:0.05710
[305]   eval-mlogloss:0.15585   train-mlogloss:0.05708
[306]   eval-mlogloss:0.15585   train-mlogloss:0.05702
[307]   eval-mlogloss:0.15585   train-mlogloss:0.05697
[308]   eval-mlogloss:0.15585   train-mlogloss:0.05693
[309]   eval-mlogloss:0.15584   train-mlogloss:0.05686
[310]   eval-mlogloss:0.15583   train-mlogloss:0.05680
[311]   eval-mlogloss:0.15583   train-mlogloss:0.05678
[312]   eval-mlogloss:0.15584   train-mlogloss:0.05673
[313]   eval-mlogloss:0.15574   train-mlogloss:0.05661
[314]   eval-mlogloss:0.15571   train-mlogloss:0.05654
[315]   eval-mlogloss:0.15569   train-mlogloss:0.05649
[316]   eval-mlogloss:0.15569   train-mlogloss:0.05645
[317]   eval-mlogloss:0.15569   train-mlogloss:0.05641
[318]   eval-mlogloss:0.15564   train-mlogloss:0.05631
[319]   eval-mlogloss:0.15562   train-mlogloss:0.05625
[320]   eval-mlogloss:0.15561   train-mlogloss:0.05620
[321]   eval-mlogloss:0.15561   train-mlogloss:0.05613
[322]   eval-mlogloss:0.15559   train-mlogloss:0.05607
[323]   eval-mlogloss:0.15558   train-mlogloss:0.05603
[324]   eval-mlogloss:0.15556   train-mlogloss:0.05598
[325]   eval-mlogloss:0.15557   train-mlogloss:0.05592
[326]   eval-mlogloss:0.15556   train-mlogloss:0.05588
[327]   eval-mlogloss:0.15555   train-mlogloss:0.05582
[328]   eval-mlogloss:0.15553   train-mlogloss:0.05578
[329]   eval-mlogloss:0.15552   train-mlogloss:0.05572
[330]   eval-mlogloss:0.15550   train-mlogloss:0.05568
[331]   eval-mlogloss:0.15550   train-mlogloss:0.05563
[332]   eval-mlogloss:0.15545   train-mlogloss:0.05552
[333]   eval-mlogloss:0.15546   train-mlogloss:0.05549
[334]   eval-mlogloss:0.15539   train-mlogloss:0.05538
[335]   eval-mlogloss:0.15539   train-mlogloss:0.05532
[336]   eval-mlogloss:0.15539   train-mlogloss:0.05526
[337]   eval-mlogloss:0.15532   train-mlogloss:0.05518
[338]   eval-mlogloss:0.15532   train-mlogloss:0.05511
[339]   eval-mlogloss:0.15530   train-mlogloss:0.05506
[340]   eval-mlogloss:0.15526   train-mlogloss:0.05496
[341]   eval-mlogloss:0.15527   train-mlogloss:0.05492
[342]   eval-mlogloss:0.15527   train-mlogloss:0.05487
[343]   eval-mlogloss:0.15521   train-mlogloss:0.05476
[344]   eval-mlogloss:0.15521   train-mlogloss:0.05472
[345]   eval-mlogloss:0.15516   train-mlogloss:0.05463
[346]   eval-mlogloss:0.15516   train-mlogloss:0.05458
[347]   eval-mlogloss:0.15511   train-mlogloss:0.05451
[348]   eval-mlogloss:0.15509   train-mlogloss:0.05444
[349]   eval-mlogloss:0.15506   train-mlogloss:0.05437
[350]   eval-mlogloss:0.15503   train-mlogloss:0.05430
[351]   eval-mlogloss:0.15503   train-mlogloss:0.05426
[352]   eval-mlogloss:0.15500   train-mlogloss:0.05420
[353]   eval-mlogloss:0.15498   train-mlogloss:0.05416
[354]   eval-mlogloss:0.15496   train-mlogloss:0.05411
[355]   eval-mlogloss:0.15490   train-mlogloss:0.05402
[356]   eval-mlogloss:0.15489   train-mlogloss:0.05397
[357]   eval-mlogloss:0.15488   train-mlogloss:0.05393
```

```
[358]   eval-mlogloss:0.15482   train-mlogloss:0.05386
[359]   eval-mlogloss:0.15473   train-mlogloss:0.05375
[360]   eval-mlogloss:0.15473   train-mlogloss:0.05371
[361]   eval-mlogloss:0.15474   train-mlogloss:0.05366
[362]   eval-mlogloss:0.15474   train-mlogloss:0.05363
[363]   eval-mlogloss:0.15469   train-mlogloss:0.05355
[364]   eval-mlogloss:0.15464   train-mlogloss:0.05348
[365]   eval-mlogloss:0.15464   train-mlogloss:0.05345
[366]   eval-mlogloss:0.15465   train-mlogloss:0.05340
[367]   eval-mlogloss:0.15467   train-mlogloss:0.05337
[368]   eval-mlogloss:0.15468   train-mlogloss:0.05333
[369]   eval-mlogloss:0.15465   train-mlogloss:0.05326
[370]   eval-mlogloss:0.15465   train-mlogloss:0.05322
[371]   eval-mlogloss:0.15465   train-mlogloss:0.05318
[372]   eval-mlogloss:0.15463   train-mlogloss:0.05313
[373]   eval-mlogloss:0.15465   train-mlogloss:0.05310
[374]   eval-mlogloss:0.15464   train-mlogloss:0.05306
[375]   eval-mlogloss:0.15464   train-mlogloss:0.05303
[376]   eval-mlogloss:0.15462   train-mlogloss:0.05293
[377]   eval-mlogloss:0.15462   train-mlogloss:0.05289
[378]   eval-mlogloss:0.15462   train-mlogloss:0.05286
[379]   eval-mlogloss:0.15456   train-mlogloss:0.05279
[380]   eval-mlogloss:0.15457   train-mlogloss:0.05277
[381]   eval-mlogloss:0.15455   train-mlogloss:0.05270
[382]   eval-mlogloss:0.15455   train-mlogloss:0.05268
[383]   eval-mlogloss:0.15455   train-mlogloss:0.05263
[384]   eval-mlogloss:0.15454   train-mlogloss:0.05260
[385]   eval-mlogloss:0.15452   train-mlogloss:0.05253
[386]   eval-mlogloss:0.15450   train-mlogloss:0.05246
[387]   eval-mlogloss:0.15446   train-mlogloss:0.05242
[388]   eval-mlogloss:0.15446   train-mlogloss:0.05238
[389]   eval-mlogloss:0.15445   train-mlogloss:0.05235
[390]   eval-mlogloss:0.15445   train-mlogloss:0.05231
[391]   eval-mlogloss:0.15444   train-mlogloss:0.05227
[392]   eval-mlogloss:0.15445   train-mlogloss:0.05223
[393]   eval-mlogloss:0.15443   train-mlogloss:0.05217
[394]   eval-mlogloss:0.15443   train-mlogloss:0.05214
[395]   eval-mlogloss:0.15443   train-mlogloss:0.05211
[396]   eval-mlogloss:0.15444   train-mlogloss:0.05210
[397]   eval-mlogloss:0.15445   train-mlogloss:0.05207
[398]   eval-mlogloss:0.15446   train-mlogloss:0.05205
[399]   eval-mlogloss:0.15442   train-mlogloss:0.05199
[400]   eval-mlogloss:0.15443   train-mlogloss:0.05198
[401]   eval-mlogloss:0.15443   train-mlogloss:0.05194
[402]   eval-mlogloss:0.15444   train-mlogloss:0.05192
[403]   eval-mlogloss:0.15437   train-mlogloss:0.05185
[404]   eval-mlogloss:0.15430   train-mlogloss:0.05177
[405]   eval-mlogloss:0.15429   train-mlogloss:0.05175
[406]   eval-mlogloss:0.15425   train-mlogloss:0.05168
[407]   eval-mlogloss:0.15425   train-mlogloss:0.05164
[408]   eval-mlogloss:0.15422   train-mlogloss:0.05161
[409]   eval-mlogloss:0.15419   train-mlogloss:0.05157
[410]   eval-mlogloss:0.15420   train-mlogloss:0.05154
[411]   eval-mlogloss:0.15420   train-mlogloss:0.05151
[412]   eval-mlogloss:0.15419   train-mlogloss:0.05148
[413]   eval-mlogloss:0.15416   train-mlogloss:0.05142
[414]   eval-mlogloss:0.15416   train-mlogloss:0.05139
[415]   eval-mlogloss:0.15416   train-mlogloss:0.05136
[416]   eval-mlogloss:0.15415   train-mlogloss:0.05130
[417]   eval-mlogloss:0.15413   train-mlogloss:0.05127
[418]   eval-mlogloss:0.15413   train-mlogloss:0.05125
[419]   eval-mlogloss:0.15411   train-mlogloss:0.05121
[420]   eval-mlogloss:0.15410   train-mlogloss:0.05118
[421]   eval-mlogloss:0.15410   train-mlogloss:0.05115
[422]   eval-mlogloss:0.15409   train-mlogloss:0.05109
```

```
[423]   eval-mlogloss:0.15409   train-mlogloss:0.05105
[424]   eval-mlogloss:0.15407   train-mlogloss:0.05099
[425]   eval-mlogloss:0.15408   train-mlogloss:0.05094
[426]   eval-mlogloss:0.15409   train-mlogloss:0.05090
[427]   eval-mlogloss:0.15407   train-mlogloss:0.05086
[428]   eval-mlogloss:0.15407   train-mlogloss:0.05085
[429]   eval-mlogloss:0.15405   train-mlogloss:0.05081
[430]   eval-mlogloss:0.15406   train-mlogloss:0.05077
[431]   eval-mlogloss:0.15405   train-mlogloss:0.05074
[432]   eval-mlogloss:0.15405   train-mlogloss:0.05072
[433]   eval-mlogloss:0.15403   train-mlogloss:0.05067
[434]   eval-mlogloss:0.15403   train-mlogloss:0.05063
[435]   eval-mlogloss:0.15404   train-mlogloss:0.05060
[436]   eval-mlogloss:0.15404   train-mlogloss:0.05056
[437]   eval-mlogloss:0.15404   train-mlogloss:0.05053
[438]   eval-mlogloss:0.15404   train-mlogloss:0.05050
[439]   eval-mlogloss:0.15404   train-mlogloss:0.05048
[440]   eval-mlogloss:0.15404   train-mlogloss:0.05045
[441]   eval-mlogloss:0.15404   train-mlogloss:0.05044
[442]   eval-mlogloss:0.15404   train-mlogloss:0.05041
[443]   eval-mlogloss:0.15404   train-mlogloss:0.05038
[444]   eval-mlogloss:0.15402   train-mlogloss:0.05034
[445]   eval-mlogloss:0.15402   train-mlogloss:0.05031
[446]   eval-mlogloss:0.15402   train-mlogloss:0.05029
[447]   eval-mlogloss:0.15397   train-mlogloss:0.05023
[448]   eval-mlogloss:0.15397   train-mlogloss:0.05020
[449]   eval-mlogloss:0.15396   train-mlogloss:0.05015
[450]   eval-mlogloss:0.15397   train-mlogloss:0.05014
[451]   eval-mlogloss:0.15392   train-mlogloss:0.05008
[452]   eval-mlogloss:0.15391   train-mlogloss:0.05004
[453]   eval-mlogloss:0.15391   train-mlogloss:0.05001
[454]   eval-mlogloss:0.15388   train-mlogloss:0.04997
[455]   eval-mlogloss:0.15386   train-mlogloss:0.04994
[456]   eval-mlogloss:0.15386   train-mlogloss:0.04989
[457]   eval-mlogloss:0.15385   train-mlogloss:0.04985
[458]   eval-mlogloss:0.15386   train-mlogloss:0.04984
[459]   eval-mlogloss:0.15385   train-mlogloss:0.04980
[460]   eval-mlogloss:0.15385   train-mlogloss:0.04979
[461]   eval-mlogloss:0.15384   train-mlogloss:0.04976
[462]   eval-mlogloss:0.15384   train-mlogloss:0.04974
[463]   eval-mlogloss:0.15382   train-mlogloss:0.04968
[464]   eval-mlogloss:0.15377   train-mlogloss:0.04963
[465]   eval-mlogloss:0.15377   train-mlogloss:0.04960
[466]   eval-mlogloss:0.15376   train-mlogloss:0.04957
[467]   eval-mlogloss:0.15376   train-mlogloss:0.04952
[468]   eval-mlogloss:0.15375   train-mlogloss:0.04950
[469]   eval-mlogloss:0.15375   train-mlogloss:0.04948
[470]   eval-mlogloss:0.15375   train-mlogloss:0.04945
[471]   eval-mlogloss:0.15376   train-mlogloss:0.04942
[472]   eval-mlogloss:0.15377   train-mlogloss:0.04938
[473]   eval-mlogloss:0.15377   train-mlogloss:0.04933
[474]   eval-mlogloss:0.15375   train-mlogloss:0.04929
[475]   eval-mlogloss:0.15375   train-mlogloss:0.04927
[476]   eval-mlogloss:0.15374   train-mlogloss:0.04925
[477]   eval-mlogloss:0.15372   train-mlogloss:0.04921
[478]   eval-mlogloss:0.15371   train-mlogloss:0.04918
[479]   eval-mlogloss:0.15369   train-mlogloss:0.04914
[480]   eval-mlogloss:0.15369   train-mlogloss:0.04912
[481]   eval-mlogloss:0.15368   train-mlogloss:0.04908
[482]   eval-mlogloss:0.15369   train-mlogloss:0.04906
[483]   eval-mlogloss:0.15364   train-mlogloss:0.04900
[484]   eval-mlogloss:0.15362   train-mlogloss:0.04895
[485]   eval-mlogloss:0.15359   train-mlogloss:0.04891
[486]   eval-mlogloss:0.15358   train-mlogloss:0.04888
[487]   eval-mlogloss:0.15358   train-mlogloss:0.04886
```

```
[488]   eval-mlogloss:0.15357   train-mlogloss:0.04884
[489]   eval-mlogloss:0.15358   train-mlogloss:0.04882
[490]   eval-mlogloss:0.15357   train-mlogloss:0.04878
[491]   eval-mlogloss:0.15358   train-mlogloss:0.04875
[492]   eval-mlogloss:0.15360   train-mlogloss:0.04873
[493]   eval-mlogloss:0.15357   train-mlogloss:0.04868
[494]   eval-mlogloss:0.15356   train-mlogloss:0.04867
[495]   eval-mlogloss:0.15354   train-mlogloss:0.04863
[496]   eval-mlogloss:0.15354   train-mlogloss:0.04861
[497]   eval-mlogloss:0.15354   train-mlogloss:0.04860
[498]   eval-mlogloss:0.15354   train-mlogloss:0.04859
[499]   eval-mlogloss:0.15354   train-mlogloss:0.04856
[500]   eval-mlogloss:0.15350   train-mlogloss:0.04853
[501]   eval-mlogloss:0.15351   train-mlogloss:0.04851
[502]   eval-mlogloss:0.15349   train-mlogloss:0.04847
[503]   eval-mlogloss:0.15349   train-mlogloss:0.04845
[504]   eval-mlogloss:0.15345   train-mlogloss:0.04838
[505]   eval-mlogloss:0.15344   train-mlogloss:0.04837
[506]   eval-mlogloss:0.15341   train-mlogloss:0.04833
[507]   eval-mlogloss:0.15341   train-mlogloss:0.04832
[508]   eval-mlogloss:0.15341   train-mlogloss:0.04830
[509]   eval-mlogloss:0.15338   train-mlogloss:0.04825
[510]   eval-mlogloss:0.15336   train-mlogloss:0.04820
[511]   eval-mlogloss:0.15335   train-mlogloss:0.04817
[512]   eval-mlogloss:0.15335   train-mlogloss:0.04816
[513]   eval-mlogloss:0.15333   train-mlogloss:0.04812
[514]   eval-mlogloss:0.15331   train-mlogloss:0.04808
[515]   eval-mlogloss:0.15331   train-mlogloss:0.04805
[516]   eval-mlogloss:0.15332   train-mlogloss:0.04803
[517]   eval-mlogloss:0.15328   train-mlogloss:0.04797
[518]   eval-mlogloss:0.15328   train-mlogloss:0.04795
[519]   eval-mlogloss:0.15328   train-mlogloss:0.04792
[520]   eval-mlogloss:0.15328   train-mlogloss:0.04790
[521]   eval-mlogloss:0.15327   train-mlogloss:0.04788
[522]   eval-mlogloss:0.15328   train-mlogloss:0.04786
[523]   eval-mlogloss:0.15326   train-mlogloss:0.04783
[524]   eval-mlogloss:0.15324   train-mlogloss:0.04779
[525]   eval-mlogloss:0.15323   train-mlogloss:0.04776
[526]   eval-mlogloss:0.15323   train-mlogloss:0.04774
[527]   eval-mlogloss:0.15323   train-mlogloss:0.04772
[528]   eval-mlogloss:0.15323   train-mlogloss:0.04771
[529]   eval-mlogloss:0.15322   train-mlogloss:0.04768
[530]   eval-mlogloss:0.15322   train-mlogloss:0.04766
[531]   eval-mlogloss:0.15321   train-mlogloss:0.04764
[532]   eval-mlogloss:0.15322   train-mlogloss:0.04762
[533]   eval-mlogloss:0.15322   train-mlogloss:0.04760
[534]   eval-mlogloss:0.15322   train-mlogloss:0.04756
[535]   eval-mlogloss:0.15317   train-mlogloss:0.04750
[536]   eval-mlogloss:0.15312   train-mlogloss:0.04745
[537]   eval-mlogloss:0.15312   train-mlogloss:0.04744
[538]   eval-mlogloss:0.15306   train-mlogloss:0.04738
[539]   eval-mlogloss:0.15306   train-mlogloss:0.04738
[540]   eval-mlogloss:0.15307   train-mlogloss:0.04735
[541]   eval-mlogloss:0.15307   train-mlogloss:0.04731
[542]   eval-mlogloss:0.15307   train-mlogloss:0.04730
[543]   eval-mlogloss:0.15304   train-mlogloss:0.04726
[544]   eval-mlogloss:0.15302   train-mlogloss:0.04722
[545]   eval-mlogloss:0.15301   train-mlogloss:0.04719
[546]   eval-mlogloss:0.15301   train-mlogloss:0.04718
[547]   eval-mlogloss:0.15301   train-mlogloss:0.04717
[548]   eval-mlogloss:0.15300   train-mlogloss:0.04716
[549]   eval-mlogloss:0.15300   train-mlogloss:0.04713
[550]   eval-mlogloss:0.15300   train-mlogloss:0.04711
[551]   eval-mlogloss:0.15300   train-mlogloss:0.04710
[552]   eval-mlogloss:0.15300   train-mlogloss:0.04708
```

```
[553]   eval-mlogloss:0.15300   train-mlogloss:0.04706
[554]   eval-mlogloss:0.15299   train-mlogloss:0.04703
[555]   eval-mlogloss:0.15298   train-mlogloss:0.04701
[556]   eval-mlogloss:0.15296   train-mlogloss:0.04698
[557]   eval-mlogloss:0.15292   train-mlogloss:0.04692
[558]   eval-mlogloss:0.15291   train-mlogloss:0.04690
[559]   eval-mlogloss:0.15291   train-mlogloss:0.04689
[560]   eval-mlogloss:0.15292   train-mlogloss:0.04687
[561]   eval-mlogloss:0.15293   train-mlogloss:0.04684
[562]   eval-mlogloss:0.15292   train-mlogloss:0.04683
[563]   eval-mlogloss:0.15293   train-mlogloss:0.04680
[564]   eval-mlogloss:0.15292   train-mlogloss:0.04678
[565]   eval-mlogloss:0.15293   train-mlogloss:0.04675
[566]   eval-mlogloss:0.15291   train-mlogloss:0.04673
[567]   eval-mlogloss:0.15291   train-mlogloss:0.04671
[568]   eval-mlogloss:0.15291   train-mlogloss:0.04670
[569]   eval-mlogloss:0.15293   train-mlogloss:0.04668
[570]   eval-mlogloss:0.15292   train-mlogloss:0.04665
[571]   eval-mlogloss:0.15293   train-mlogloss:0.04664
[572]   eval-mlogloss:0.15293   train-mlogloss:0.04661
[573]   eval-mlogloss:0.15292   train-mlogloss:0.04660
[574]   eval-mlogloss:0.15293   train-mlogloss:0.04659
[575]   eval-mlogloss:0.15291   train-mlogloss:0.04656
[576]   eval-mlogloss:0.15287   train-mlogloss:0.04650
[577]   eval-mlogloss:0.15286   train-mlogloss:0.04648
[578]   eval-mlogloss:0.15285   train-mlogloss:0.04646
[579]   eval-mlogloss:0.15285   train-mlogloss:0.04644
[580]   eval-mlogloss:0.15283   train-mlogloss:0.04641
[581]   eval-mlogloss:0.15281   train-mlogloss:0.04638
[582]   eval-mlogloss:0.15281   train-mlogloss:0.04637
[583]   eval-mlogloss:0.15279   train-mlogloss:0.04632
[584]   eval-mlogloss:0.15279   train-mlogloss:0.04632
[585]   eval-mlogloss:0.15276   train-mlogloss:0.04627
[586]   eval-mlogloss:0.15276   train-mlogloss:0.04626
[587]   eval-mlogloss:0.15278   train-mlogloss:0.04624
[588]   eval-mlogloss:0.15277   train-mlogloss:0.04622
[589]   eval-mlogloss:0.15276   train-mlogloss:0.04619
[590]   eval-mlogloss:0.15276   train-mlogloss:0.04617
[591]   eval-mlogloss:0.15277   train-mlogloss:0.04615
[592]   eval-mlogloss:0.15276   train-mlogloss:0.04614
[593]   eval-mlogloss:0.15275   train-mlogloss:0.04611
[594]   eval-mlogloss:0.15276   train-mlogloss:0.04609
[595]   eval-mlogloss:0.15276   train-mlogloss:0.04609
[596]   eval-mlogloss:0.15276   train-mlogloss:0.04606
[597]   eval-mlogloss:0.15274   train-mlogloss:0.04605
[598]   eval-mlogloss:0.15274   train-mlogloss:0.04600
[599]   eval-mlogloss:0.15273   train-mlogloss:0.04598
[600]   eval-mlogloss:0.15273   train-mlogloss:0.04597
[601]   eval-mlogloss:0.15274   train-mlogloss:0.04595
[602]   eval-mlogloss:0.15271   train-mlogloss:0.04591
[603]   eval-mlogloss:0.15271   train-mlogloss:0.04589
[604]   eval-mlogloss:0.15271   train-mlogloss:0.04589
[605]   eval-mlogloss:0.15272   train-mlogloss:0.04587
[606]   eval-mlogloss:0.15272   train-mlogloss:0.04586
[607]   eval-mlogloss:0.15271   train-mlogloss:0.04582
[608]   eval-mlogloss:0.15270   train-mlogloss:0.04580
[609]   eval-mlogloss:0.15268   train-mlogloss:0.04578
[610]   eval-mlogloss:0.15267   train-mlogloss:0.04575
[611]   eval-mlogloss:0.15268   train-mlogloss:0.04572
[612]   eval-mlogloss:0.15266   train-mlogloss:0.04570
[613]   eval-mlogloss:0.15266   train-mlogloss:0.04568
[614]   eval-mlogloss:0.15266   train-mlogloss:0.04568
[615]   eval-mlogloss:0.15263   train-mlogloss:0.04563
[616]   eval-mlogloss:0.15263   train-mlogloss:0.04563
[617]   eval-mlogloss:0.15261   train-mlogloss:0.04559
```

```
[618]   eval-mlogloss:0.15260   train-mlogloss:0.04558
[619]   eval-mlogloss:0.15258   train-mlogloss:0.04555
[620]   eval-mlogloss:0.15257   train-mlogloss:0.04554
[621]   eval-mlogloss:0.15256   train-mlogloss:0.04552
[622]   eval-mlogloss:0.15256   train-mlogloss:0.04550
[623]   eval-mlogloss:0.15255   train-mlogloss:0.04549
[624]   eval-mlogloss:0.15256   train-mlogloss:0.04547
[625]   eval-mlogloss:0.15257   train-mlogloss:0.04546
[626]   eval-mlogloss:0.15257   train-mlogloss:0.04544
[627]   eval-mlogloss:0.15257   train-mlogloss:0.04544
[628]   eval-mlogloss:0.15257   train-mlogloss:0.04543
[629]   eval-mlogloss:0.15258   train-mlogloss:0.04541
[630]   eval-mlogloss:0.15258   train-mlogloss:0.04539
[631]   eval-mlogloss:0.15258   train-mlogloss:0.04539
[632]   eval-mlogloss:0.15257   train-mlogloss:0.04536
[633]   eval-mlogloss:0.15258   train-mlogloss:0.04534
[634]   eval-mlogloss:0.15257   train-mlogloss:0.04533
[635]   eval-mlogloss:0.15257   train-mlogloss:0.04531
[636]   eval-mlogloss:0.15258   train-mlogloss:0.04529
[637]   eval-mlogloss:0.15256   train-mlogloss:0.04527
[638]   eval-mlogloss:0.15257   train-mlogloss:0.04526
[639]   eval-mlogloss:0.15257   train-mlogloss:0.04524
[640]   eval-mlogloss:0.15258   train-mlogloss:0.04522
[641]   eval-mlogloss:0.15257   train-mlogloss:0.04521
[642]   eval-mlogloss:0.15255   train-mlogloss:0.04519
[643]   eval-mlogloss:0.15256   train-mlogloss:0.04516
[644]   eval-mlogloss:0.15256   train-mlogloss:0.04514
[645]   eval-mlogloss:0.15255   train-mlogloss:0.04513
[646]   eval-mlogloss:0.15254   train-mlogloss:0.04511
[647]   eval-mlogloss:0.15254   train-mlogloss:0.04510
[648]   eval-mlogloss:0.15255   train-mlogloss:0.04509
[649]   eval-mlogloss:0.15255   train-mlogloss:0.04507
[650]   eval-mlogloss:0.15255   train-mlogloss:0.04506
[651]   eval-mlogloss:0.15255   train-mlogloss:0.04503
[652]   eval-mlogloss:0.15249   train-mlogloss:0.04498
[653]   eval-mlogloss:0.15249   train-mlogloss:0.04497
[654]   eval-mlogloss:0.15249   train-mlogloss:0.04497
[655]   eval-mlogloss:0.15249   train-mlogloss:0.04494
[656]   eval-mlogloss:0.15249   train-mlogloss:0.04491
[657]   eval-mlogloss:0.15250   train-mlogloss:0.04489
[658]   eval-mlogloss:0.15248   train-mlogloss:0.04486
[659]   eval-mlogloss:0.15247   train-mlogloss:0.04484
[660]   eval-mlogloss:0.15246   train-mlogloss:0.04482
[661]   eval-mlogloss:0.15246   train-mlogloss:0.04481
[662]   eval-mlogloss:0.15246   train-mlogloss:0.04480
[663]   eval-mlogloss:0.15247   train-mlogloss:0.04480
[664]   eval-mlogloss:0.15246   train-mlogloss:0.04478
[665]   eval-mlogloss:0.15246   train-mlogloss:0.04476
[666]   eval-mlogloss:0.15246   train-mlogloss:0.04474
[667]   eval-mlogloss:0.15247   train-mlogloss:0.04472
[668]   eval-mlogloss:0.15246   train-mlogloss:0.04471
[669]   eval-mlogloss:0.15244   train-mlogloss:0.04467
[670]   eval-mlogloss:0.15244   train-mlogloss:0.04466
[671]   eval-mlogloss:0.15244   train-mlogloss:0.04464
[672]   eval-mlogloss:0.15242   train-mlogloss:0.04462
[673]   eval-mlogloss:0.15243   train-mlogloss:0.04460
[674]   eval-mlogloss:0.15241   train-mlogloss:0.04458
[675]   eval-mlogloss:0.15241   train-mlogloss:0.04457
[676]   eval-mlogloss:0.15241   train-mlogloss:0.04457
[677]   eval-mlogloss:0.15238   train-mlogloss:0.04453
[678]   eval-mlogloss:0.15238   train-mlogloss:0.04452
[679]   eval-mlogloss:0.15238   train-mlogloss:0.04451
[680]   eval-mlogloss:0.15238   train-mlogloss:0.04449
[681]   eval-mlogloss:0.15238   train-mlogloss:0.04447
[682]   eval-mlogloss:0.15238   train-mlogloss:0.04446
```

```
[683]   eval-mlogloss:0.15238   train-mlogloss:0.04444
[684]   eval-mlogloss:0.15238   train-mlogloss:0.04442
[685]   eval-mlogloss:0.15238   train-mlogloss:0.04441
[686]   eval-mlogloss:0.15238   train-mlogloss:0.04439
[687]   eval-mlogloss:0.15238   train-mlogloss:0.04439
[688]   eval-mlogloss:0.15239   train-mlogloss:0.04437
[689]   eval-mlogloss:0.15239   train-mlogloss:0.04435
[690]   eval-mlogloss:0.15238   train-mlogloss:0.04433
[691]   eval-mlogloss:0.15238   train-mlogloss:0.04432
[692]   eval-mlogloss:0.15239   train-mlogloss:0.04430
[693]   eval-mlogloss:0.15239   train-mlogloss:0.04429
[694]   eval-mlogloss:0.15239   train-mlogloss:0.04428
[695]   eval-mlogloss:0.15238   train-mlogloss:0.04426
[696]   eval-mlogloss:0.15238   train-mlogloss:0.04423
[697]   eval-mlogloss:0.15235   train-mlogloss:0.04419
[698]   eval-mlogloss:0.15235   train-mlogloss:0.04417
[699]   eval-mlogloss:0.15233   train-mlogloss:0.04416
[700]   eval-mlogloss:0.15233   train-mlogloss:0.04414
[701]   eval-mlogloss:0.15229   train-mlogloss:0.04411
[702]   eval-mlogloss:0.15230   train-mlogloss:0.04410
[703]   eval-mlogloss:0.15230   train-mlogloss:0.04407
[704]   eval-mlogloss:0.15229   train-mlogloss:0.04407
[705]   eval-mlogloss:0.15229   train-mlogloss:0.04405
[706]   eval-mlogloss:0.15230   train-mlogloss:0.04404
[707]   eval-mlogloss:0.15230   train-mlogloss:0.04403
[708]   eval-mlogloss:0.15231   train-mlogloss:0.04402
[709]   eval-mlogloss:0.15231   train-mlogloss:0.04401
[710]   eval-mlogloss:0.15230   train-mlogloss:0.04399
[711]   eval-mlogloss:0.15228   train-mlogloss:0.04395
[712]   eval-mlogloss:0.15227   train-mlogloss:0.04394
[713]   eval-mlogloss:0.15226   train-mlogloss:0.04393
[714]   eval-mlogloss:0.15226   train-mlogloss:0.04391
[715]   eval-mlogloss:0.15226   train-mlogloss:0.04390
[716]   eval-mlogloss:0.15224   train-mlogloss:0.04387
[717]   eval-mlogloss:0.15224   train-mlogloss:0.04386
[718]   eval-mlogloss:0.15224   train-mlogloss:0.04385
[719]   eval-mlogloss:0.15222   train-mlogloss:0.04382
[720]   eval-mlogloss:0.15222   train-mlogloss:0.04381
[721]   eval-mlogloss:0.15222   train-mlogloss:0.04381
[722]   eval-mlogloss:0.15222   train-mlogloss:0.04380
[723]   eval-mlogloss:0.15221   train-mlogloss:0.04378
[724]   eval-mlogloss:0.15221   train-mlogloss:0.04377
[725]   eval-mlogloss:0.15221   train-mlogloss:0.04376
[726]   eval-mlogloss:0.15222   train-mlogloss:0.04375
[727]   eval-mlogloss:0.15222   train-mlogloss:0.04374
[728]   eval-mlogloss:0.15222   train-mlogloss:0.04373
[729]   eval-mlogloss:0.15219   train-mlogloss:0.04370
[730]   eval-mlogloss:0.15220   train-mlogloss:0.04369
[731]   eval-mlogloss:0.15220   train-mlogloss:0.04368
[732]   eval-mlogloss:0.15220   train-mlogloss:0.04367
[733]   eval-mlogloss:0.15220   train-mlogloss:0.04366
[734]   eval-mlogloss:0.15220   train-mlogloss:0.04365
[735]   eval-mlogloss:0.15220   train-mlogloss:0.04364
[736]   eval-mlogloss:0.15220   train-mlogloss:0.04364
[737]   eval-mlogloss:0.15220   train-mlogloss:0.04364
[738]   eval-mlogloss:0.15220   train-mlogloss:0.04363
[739]   eval-mlogloss:0.15219   train-mlogloss:0.04362
[740]   eval-mlogloss:0.15219   train-mlogloss:0.04360
[741]   eval-mlogloss:0.15219   train-mlogloss:0.04358
[742]   eval-mlogloss:0.15219   train-mlogloss:0.04357
[743]   eval-mlogloss:0.15219   train-mlogloss:0.04357
[744]   eval-mlogloss:0.15219   train-mlogloss:0.04357
```

In [ ]:

```
# 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: 0.94567
Precision: 0.94279
Recall: 0.92168
```

In [ ]:

```
confusion_matrix_xgb = confusion_matrix(y_test, preds)
print(confusion_matrix_xgb)
print((confusion_matrix_xgb[0,1]*100 + confusion_matrix_xgb[1,0]*250)/len(y_test
```

```
[[18142   717]
 [ 1004 11816]]
10.186558919157802
```

In [ ]:

```
print(model_rs)
```

```
<xgboost.core.Booster object at 0x7faf2c64a4c0>
```

Best model test-mlogloss-mean: 0.156774

final_params = {'booster': 'gbtree', 'objective': 'multi:softmax', 'num_class': 2, 'eta': 0.05, 'subsample': 0.5792877172818175, 'colsample_bytree': 0.7680400747721351, 'max_depth': 20, 'min_child_weight': 5, 'gamma': 1.5}

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

Accuracy: 0.94567

Precision: 0.94279

Recall: 0.92168

NN

In [ ]:

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models

from time import time
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import TensorBoard
```

In [ ]:

```
X_train_nn, X_val_nn, y_train_nn, y_val_nn = train_test_split(X_train, y_train,
```

In [ ]:

```
X_train_nn.columns
```

Out[ ]:

```
Index(['index', 'x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9',
       'x10', 'x11', 'x12', 'x13', 'x14', 'x15', 'x16', 'x17', 'x18', 'x19',
```

```
       'x20', 'x21', 'x22', 'x23', 'x25', 'x26', 'x27', 'x28', 'x31', 'x33',
       'x34', 'x35', 'x36', 'x37', 'x38', 'x39', 'x40', 'x41', 'x42', 'x43',
       'x44', 'x45', 'x46', 'x47', 'x48', 'x49', 'x24_america', 'x24_asia',
       'x24_euorpe', 'x29_Apr', 'x29_Aug', 'x29_Dev', 'x29_Feb', 'x29_January',
       'x29_July', 'x29_Jun', 'x29_Mar', 'x29_May', 'x29_Nov', 'x29_Oct',
       'x29_sept.', 'x30_friday', 'x30_monday', 'x30_thurday', 'x30_tuesday',
       'x30_wednesday', 'x32_-0.0%', 'x32_-0.01%', 'x32_-0.02%', 'x32_-0.03%',
       'x32_-0.04%', 'x32_-0.05%', 'x32_0.0%', 'x32_0.01%', 'x32_0.02%',
       'x32_0.03%', 'x32_0.04%', 'x32_0.05%'],
      dtype='object')
```

In [ ]:
```
X_train_nn.shape
```

Out[ ]:
```
(101370, 79)
```

In [ ]:
```
from tensorflow.keras.layers import BatchNormalization
model_nn = Sequential()
model_nn.add(tf.keras.Input(shape=(79,)))
model_nn.add(tf.keras.layers.Dense(500, activation='relu'))
model_nn.add(tf.keras.layers.Dense(400, activation='relu'))
model_nn.add(tf.keras.layers.Dense(300, activation='relu'))
model_nn.add(tf.keras.layers.Dense(200, activation='relu'))
model_nn.add(tf.keras.layers.Dense(100, activation='relu'))
model_nn.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

In [ ]:
```
from tensorflow.keras.callbacks import EarlyStopping

safety = EarlyStopping(patience = 3, monitor='val_loss')
```

In [ ]:
```
# from tensorflow.keras.optimizers import SGD
# opt = SGD()
model_nn.compile(optimizer='adam', loss = tf.keras.losses.BinaryCrossentropy() ,
```

In [ ]:
```
# X_train, X_test, y_train, y_test
history = model_nn.fit(X_train_nn, y_train_nn, validation_data=(X_val_nn, y_val_
```

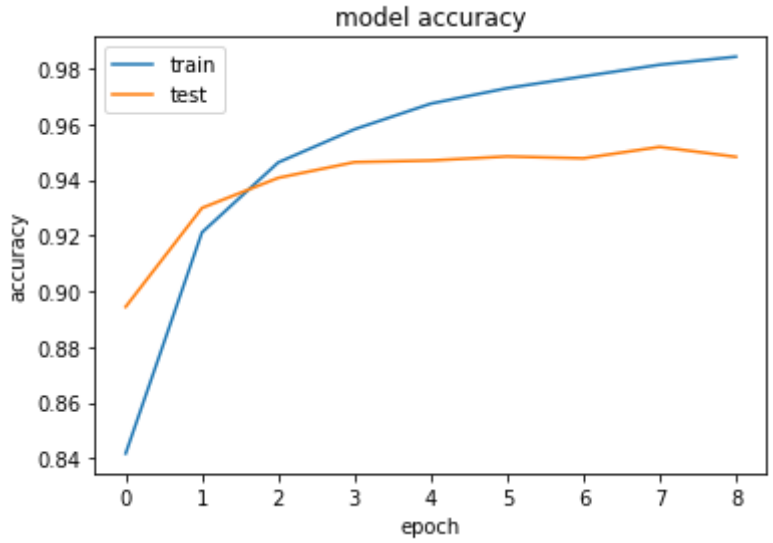
```
Epoch 1/1000
102/102 [==============================] - 9s 72ms/step - loss: 0.3498 - accurac
y: 0.8418 - false_negatives_1: 9513.0000 - false_positives_1: 6528.0000 - auc_1:
0.9204 - precision_1: 0.8265 - recall_1: 0.7657 - val_loss: 0.2541 - val_accurac
y: 0.8944 - val_false_negatives_1: 999.0000 - val_false_positives_1: 1677.0000 -
val_auc_1: 0.9611 - val_precision_1: 0.8447 - val_recall_1: 0.9013
Epoch 2/1000
102/102 [==============================] - 7s 65ms/step - loss: 0.2005 - accurac
y: 0.9211 - false_negatives_1: 4199.0000 - false_positives_1: 3799.0000 - auc_1:
0.9740 - precision_1: 0.9055 - recall_1: 0.8966 - val_loss: 0.1820 - val_accurac
y: 0.9299 - val_false_negatives_1: 1158.0000 - val_false_positives_1: 619.0000 -
val_auc_1: 0.9789 - val_precision_1: 0.9354 - val_recall_1: 0.8856
Epoch 3/1000
102/102 [==============================] - 7s 65ms/step - loss: 0.1464 - accurac
y: 0.9462 - false_negatives_1: 2897.0000 - false_positives_1: 2553.0000 - auc_1:
0.9853 - precision_1: 0.9366 - recall_1: 0.9287 - val_loss: 0.1598 - val_accurac
y: 0.9407 - val_false_negatives_1: 618.0000 - val_false_positives_1: 885.0000 -
val_auc_1: 0.9832 - val_precision_1: 0.9148 - val_recall_1: 0.9389
Epoch 4/1000
102/102 [==============================] - 7s 65ms/step - loss: 0.1193 - accurac
```

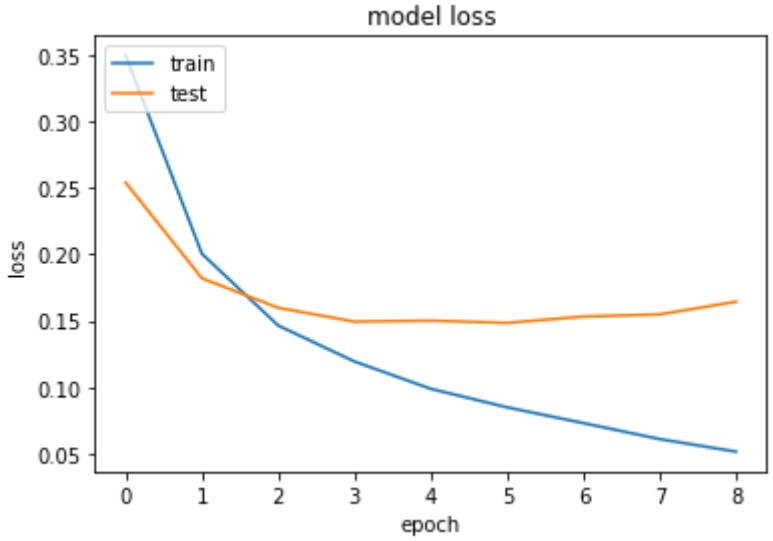
```
y: 0.9581 - false_negatives_1: 2244.0000 - false_positives_1: 1999.0000 - auc_1:
0.9895 - precision_1: 0.9505 - recall_1: 0.9447 - val_loss: 0.1494 - val_accurac
y: 0.9463 - val_false_negatives_1: 871.0000 - val_false_positives_1: 489.0000 -
val_auc_1: 0.9848 - val_precision_1: 0.9498 - val_recall_1: 0.9139
Epoch 5/1000
102/102 [==============================] - 7s 65ms/step - loss: 0.0988 - accurac
y: 0.9673 - false_negatives_1: 1731.0000 - false_positives_1: 1579.0000 - auc_1:
0.9921 - precision_1: 0.9610 - recall_1: 0.9574 - val_loss: 0.1501 - val_accurac
y: 0.9469 - val_false_negatives_1: 789.0000 - val_false_positives_1: 556.0000 -
val_auc_1: 0.9849 - val_precision_1: 0.9438 - val_recall_1: 0.9220
Epoch 6/1000
102/102 [==============================] - 7s 65ms/step - loss: 0.0848 - accurac
y: 0.9729 - false_negatives_1: 1459.0000 - false_positives_1: 1288.0000 - auc_1:
0.9937 - precision_1: 0.9681 - recall_1: 0.9641 - val_loss: 0.1484 - val_accurac
y: 0.9484 - val_false_negatives_1: 644.0000 - val_false_positives_1: 664.0000 -
val_auc_1: 0.9848 - val_precision_1: 0.9345 - val_recall_1: 0.9364
Epoch 7/1000
102/102 [==============================] - 7s 64ms/step - loss: 0.0729 - accurac
y: 0.9771 - false_negatives_1: 1221.0000 - false_positives_1: 1098.0000 - auc_1:
0.9949 - precision_1: 0.9729 - recall_1: 0.9699 - val_loss: 0.1532 - val_accurac
y: 0.9477 - val_false_negatives_1: 801.0000 - val_false_positives_1: 524.0000 -
val_auc_1: 0.9846 - val_precision_1: 0.9468 - val_recall_1: 0.9209
Epoch 8/1000
102/102 [==============================] - 7s 64ms/step - loss: 0.0609 - accurac
y: 0.9813 - false_negatives_1: 1001.0000 - false_positives_1: 894.0000 - auc_1:
0.9961 - precision_1: 0.9779 - recall_1: 0.9753 - val_loss: 0.1548 - val_accurac
y: 0.9518 - val_false_negatives_1: 758.0000 - val_false_positives_1: 463.0000 -
val_auc_1: 0.9850 - val_precision_1: 0.9529 - val_recall_1: 0.9251
Epoch 9/1000
102/102 [==============================] - 7s 64ms/step - loss: 0.0514 - accurac
y: 0.9842 - false_negatives_1: 860.0000 - false_positives_1: 743.0000 - auc_1:
0.9970 - precision_1: 0.9816 - recall_1: 0.9788 - val_loss: 0.1644 - val_accurac
y: 0.9482 - val_false_negatives_1: 695.0000 - val_false_positives_1: 617.0000 -
val_auc_1: 0.9831 - val_precision_1: 0.9386 - val_recall_1: 0.9313
```

In [ ]:

```
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'false_negatives_1', 'false_positives_1', 'auc_
1', 'precision_1', 'recall_1', 'val_loss', 'val_accuracy', 'val_false_negatives_
1', 'val_false_positives_1', 'val_auc_1', 'val_precision_1', 'val_recall_1'])
```





In [ ]:
```
nn_preds = model_nn.predict(X_test)
```

In [ ]:
```
nn_preds
```

Out[ ]:
```
array([[2.9492378e-04],
       [9.9992323e-01],
       [9.9677444e-01],
       ...,
       [1.1721969e-02],
       [1.3581514e-03],
       [7.9426467e-03]], dtype=float32)
```

In [ ]:
```
test_guess_nn = pd.DataFrame(nn_preds)
test_guess_nn['predict'] = 0
test_guess_nn.loc[test_guess_nn[0]>0.221,"predict"]=1
```

In [ ]:
```
precision = precision_score(test_guess_nn['predict'],y_test)
recall = recall_score(test_guess_nn['predict'],y_test)
accuracy = accuracy_score(test_guess_nn['predict'],y_test)

print(f"Accuracy: {accuracy:.5f}")
```

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

```
Accuracy: 0.94346
Precision: 0.95000
Recall: 0.91372
```

In [ ]:
```
confusion_matrix_nn = confusion_matrix(y_test, test_guess_nn['predict'])
print(confusion_matrix_nn)
print((confusion_matrix_nn[0,1]*100 + confusion_matrix_nn[1,0]*250)/len(y_test))
```

```
[[17709  1150]
 [  641 12179]]
8.688721234887465
```

Ensembling

In [ ]:
```
# Neural Net Wrapper Scikitlearn KerasClassifier
```

In [ ]:
```
from keras.wrappers.scikit_learn import KerasClassifier
```

In [ ]:
```
def create_model_3():
  my_model = Sequential()
  my_model.add(tf.keras.Input(shape=(3,)))
  my_model.add(tf.keras.layers.Dense(500, activation='relu'))
  my_model.add(tf.keras.layers.Dense(400, activation='relu'))
  my_model.add(tf.keras.layers.Dense(300, activation='relu'))
  my_model.add(tf.keras.layers.Dense(200, activation='relu'))
  my_model.add(tf.keras.layers.Dense(100, activation='relu'))
  my_model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

  # safety = EarlyStopping(patience = 3, monitor='val_loss')

  my_model.compile(optimizer='adam', loss = tf.keras.losses.BinaryCrossentropy()

  return my_model
```

In [ ]:
```
def create_model_79():
  my_model = Sequential()
  my_model.add(tf.keras.Input(shape=(79,)))
  my_model.add(tf.keras.layers.Dense(500, activation='relu'))
  my_model.add(tf.keras.layers.Dense(400, activation='relu'))
  my_model.add(tf.keras.layers.Dense(300, activation='relu'))
  my_model.add(tf.keras.layers.Dense(200, activation='relu'))
  my_model.add(tf.keras.layers.Dense(100, activation='relu'))
  my_model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

  # safety = EarlyStopping(patience = 3, monitor='val_loss')

  my_model.compile(optimizer='adam', loss = tf.keras.losses.BinaryCrossentropy()

  return my_model
```

In [ ]:
```
# my_model = KerasClassifier(build_fn=create_model, epochs=10, batch_size=1000,
```

Ensemble: 1

Level 1 - Random Forest and Logistic Regression

Level 2 - XGBoost

91.8% Train Accuracy

In [ ]:
```
rf_level1 = RandomForestClassifier()
lr_level1 = LogisticRegression(max_iter=500)

# Uncomment when you need test preds
# rf_level1.fit(X_train, y_train)
# lr_level1.fit(X_train, y_train)

rf1 = cross_val_predict(rf_level1, X_train, y_train, cv=3, method='predict_proba
lr1 = cross_val_predict(lr_level1, X_train, y_train, cv=3, method='predict_proba
```

In [ ]:
```
level2_data = pd.DataFrame(rf1[:,1])
level2_data['lr'] = lr1[:,1]
level2_data
```

Out[ ]:

	0	lr
0	0.28	0.374970
1	0.10	0.099207
2	0.74	0.515404
3	0.36	0.438831
4	0.83	0.595021
...	...	...
126708	0.39	0.780199
126709	0.17	0.071159
126710	0.14	0.166706
126711	0.38	0.569869
126712	0.37	0.619805

126713 rows × 2 columns

In [ ]:
```
final_model = GradientBoostingClassifier()

# Uncomment when you need test preds
# final_model.fit(level2_data, y_train)

final_preds = cross_val_predict(final_model, level2_data, y_train, cv=5)
```

In [ ]:
```
guess = pd.DataFrame(final_preds)
```

```
guess['predict'] = 0
guess.loc[guess[0]>0.5,"predict"]=1

accuracy_score(guess['predict'],y_train)
```

Out[ ]: 0.918074704252918

Ensemble 2

Level 1 - Tuned Random Forest, Tuned Logistic Regression

Level 2 - Random Forest, Logistic Regression

Level 3 - XGBoost

92.5% Train Accuracy

In [ ]:
```
rf_level1 = RandomForestClassifier(class_weight='balanced', criterion='entropy',
lr_level1 = LogisticRegression(C=0.01, max_iter=500)

# Uncomment when you need test predictions
rf_level1.fit(X_train, y_train)
lr_level1.fit(X_train, y_train)

rf1 = cross_val_predict(rf_level1, X_train, y_train, cv=3, method='predict_proba
lr1 = cross_val_predict(lr_level1, X_train, y_train, cv=4, method='predict_proba
```

In [ ]:
```
level2_data = pd.DataFrame(rf1[:,1])
level2_data['lr'] = lr1[:,1]
level2_data
```

Out[ ]:

	0	lr
0	0.329193	0.374303
1	0.084028	0.095674
2	0.717596	0.470001
3	0.518386	0.432468
4	0.921238	0.593786
...	...	...
126708	0.320996	0.773343
126709	0.032964	0.078576
126710	0.090867	0.170910
126711	0.430979	0.569288
126712	0.223400	0.616765

126713 rows × 2 columns

In [ ]:
```
rf_level2 = RandomForestClassifier()
lr_level2 = LogisticRegression()
```

```python
# Uncomment when you need test predictions
rf_level2.fit(level2_data, y_train)
lr_level2.fit(level2_data, y_train)

rf2 = cross_val_predict(rf_level2, level2_data, y_train, cv=3, method='predict_p
lr2 = cross_val_predict(lr_level2, level2_data, y_train, cv=3, method='predict_p
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

In [ ]:

```
level3_data = pd.DataFrame(rf2[:,1])
level3_data['lr'] = lr2[:,1]
level3_data
```

Out[ ]:

	0	lr
0	0.44	0.161453
1	0.00	0.024415
2	0.75	0.908471
3	0.50	0.561996
4	1.00	0.985386
...	...	...
126708	0.09	0.075824
126709	0.00	0.014855
126710	0.03	0.022748
126711	0.54	0.284001
126712	0.15	0.038034

126713 rows × 2 columns

In [ ]:

```
final_model = GradientBoostingClassifier()

# Uncomment when you need test predictions
final_model.fit(level3_data, y_train)

final_preds = cross_val_predict(final_model, level3_data, y_train, cv=5)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

In [ ]:

```
guess = pd.DataFrame(final_preds)
guess['predict'] = 0
guess.loc[guess[0]>0.5,"predict"]=1

accuracy_score(guess['predict'],y_train)
```

Out[ ]:

```
0.925098450829828
```

Ensemble 3

Level 1 - Random Forest, Logistic Regression, Neural Network

Level 2 - Random Forest, Logistic Regression, Neural Network

Level 3 - XGBoost

91.8% Train Accuracy

In [ ]:

```
rf_level1 = RandomForestClassifier()
lr_level1 = LogisticRegression(max_iter=500)
nn_level1 = KerasClassifier(build_fn=create_model_79, epochs=9, batch_size=1000,

# Uncomment when you need test predictions
# rf_level1.fit(X_train, y_train)
# lr_level1.fit(X_train, y_train)
# nn_level1.fit(X_train, y_train)

rf1 = cross_val_predict(rf_level1, X_train, y_train, cv=3, method='predict_proba
lr1 = cross_val_predict(lr_level1, X_train, y_train, cv=4, method='predict_proba
nn1 = cross_val_predict(lr_level1, X_train, y_train, cv=4, method='predict_proba

level2_data = pd.DataFrame(rf1[:,1])
level2_data['lr'] = lr1[:,1]
level2_data['nn'] = nn1[:,1]
level2_data
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarni
ng: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/sc
ikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for
```

```
help migrating.
  This is separate from the ipykernel package so we can avoid doing imports unti
l
```

Out[ ]:

	0	lr	nn
0	0.44	0.373100	0.373100
1	0.16	0.097436	0.097436
2	0.73	0.513791	0.513791
3	0.42	0.431478	0.431478
4	0.74	0.587620	0.587620
...	...	...	...
126708	0.46	0.777981	0.777981
126709	0.12	0.073421	0.073421
126710	0.14	0.167913	0.167913
126711	0.51	0.576890	0.576890
126712	0.29	0.620542	0.620542

126713 rows × 3 columns

In [ ]:

```
rf_level2 = RandomForestClassifier()
lr_level2 = LogisticRegression()
nn_level2 = KerasClassifier(build_fn=create_model_3, epochs=15, batch_size=1000,

# Uncomment when you need test predictions
# rf_level2.fit(level2_data, y_train)
# lr_level2.fit(level2_data, y_train)
# nn_level2.fit(level2_data, y_train)

rf2 = cross_val_predict(rf_level2, level2_data, y_train, cv=3, method='predict_p
lr2 = cross_val_predict(lr_level2, level2_data, y_train, cv=3, method='predict_p
nn2 = cross_val_predict(nn_level2, level2_data, y_train, cv=3, method='predict_p

level3_data = pd.DataFrame(rf2[:,1])
level3_data['lr'] = lr2[:,1]
level3_data['nn'] = nn2[:,1]
level3_data
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarni
ng: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/sc
ikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for
help migrating.
  This is separate from the ipykernel package so we can avoid doing imports unti
l
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
```

```
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

Out[ ]:

	0	lr	nn
0	0.43	0.445174	0.478795
1	0.00	0.040444	0.044240
2	0.99	0.961405	0.953613
3	0.66	0.345794	0.391266
4	0.99	0.958944	0.954714
...	...	...	...
126708	0.72	0.271237	0.316899
126709	0.00	0.025896	0.023975
126710	0.00	0.026349	0.022788
126711	0.22	0.547167	0.605917
126712	0.05	0.056640	0.072156

126713 rows × 3 columns

In [ ]:

```
final_model = GradientBoostingClassifier()

# Uncomment when you need test predictions
# final_model.fit(level3_data, y_train)

final_preds = cross_val_predict(final_model, level3_data, y_train, cv=5)

guess = pd.DataFrame(final_preds)
guess['predict'] = 0
guess.loc[guess[0]>0.5,"predict"]=1

accuracy_score(guess['predict'],y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

Out[ ]:

```
0.9178537324504984
```

Ensemble 4

Level 1 - Random Forest, Logistic Regression, Neural Network, Neural Network, XGBoost Classifier

Level 2 - XGBoost Classifier

91.8% Train Accuracy

In [ ]:

```
rf_level1 = RandomForestClassifier()
lr_level1 = LogisticRegression(max_iter=500)
nn_level1 = KerasClassifier(build_fn=create_model_79, epochs=9, batch_size=1000,
nn_level1_2 = KerasClassifier(build_fn=create_model_79, epochs=30, batch_size=10
xgb_level1 = GradientBoostingClassifier()

## Uncomment when test evaluation is needed
# rf_level1.fit(X_train, y_train)
# lr_level1.fit(X_train, y_train)
# nn_level1.fit(X_train, y_train)
# nn_level1_2.fit(X_train, y_train)
# xgb_level1.fit(X_train, y_train)

rf1 = cross_val_predict(rf_level1, X_train, y_train, cv=3, method='predict_proba
lr1 = cross_val_predict(lr_level1, X_train, y_train, cv=3, method='predict_proba
nn1 = cross_val_predict(lr_level1, X_train, y_train, cv=3, method='predict_proba
nn1_2 = cross_val_predict(nn_level1_2, X_train, y_train, cv=3, method='predict_p
xgb1 = cross_val_predict(xgb_level1, X_train, y_train, cv=3, method='predict_pro

level2_data = pd.DataFrame(rf1[:,1])
level2_data['lr'] = lr1[:,1]
level2_data['nn'] = nn1[:,1]
level2_data['nn_2'] = nn1_2[:,1]
level2_data['xgb'] = xgb1[:,1]
level2_data
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarni
ng: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/sc
ikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for
help migrating.
  This is separate from the ipykernel package so we can avoid doing imports unti
l
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DeprecationWarni
ng: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/sc
ikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for
help migrating.
  after removing the cwd from sys.path.
```

Out[ ]:

	0	lr	nn	nn_2	xgb
0	0.48	0.374840	0.374840	1.887556e-07	0.444457
1	0.15	0.099180	0.099180	5.177485e-06	0.072695
2	0.73	0.515659	0.515659	9.999931e-01	0.528277
3	0.36	0.438907	0.438907	9.998369e-01	0.342214
4	0.84	0.595057	0.595057	9.998915e-01	0.681795
...	...	...	...	...	...
126708	0.34	0.780217	0.780217	3.338272e-05	0.636786
126709	0.16	0.071169	0.071169	8.145187e-06	0.061157
126710	0.26	0.166705	0.166705	8.583481e-11	0.143517
126711	0.45	0.569767	0.569767	4.715800e-03	0.396046

	0	lr	nn	nn_2	xgb
126712	0.31	0.619666	0.619666	9.001749e-10	0.449983

126713 rows × 5 columns

In [ ]:

```
final_model = GradientBoostingClassifier()

# Uncomment when you need test predictions
# final_model.fit(level3_data, y_train)

final_preds = cross_val_predict(final_model, level3_data, y_train, cv=5)

guess = pd.DataFrame(final_preds)
guess['predict'] = 0
guess.loc[guess[0]>0.5,"predict"]=1

accuracy_score(guess['predict'],y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

Out[ ]: 0.9178379487503255

Test Eval

In [ ]:
```
test_rf = rf_level1.predict_proba(X_test)
test_lr = lr_level1.predict_proba(X_test)

test_level2_data = pd.DataFrame(test_rf[:,1])
test_level2_data['lr'] = test_lr[:,1]
```

In [ ]:
```
test_rf_2 = rf_level2.predict_proba(test_level2_data)
test_lr_2 = lr_level2.predict_proba(test_level2_data)

test_final_data = pd.DataFrame(test_rf_2[:,1])
test_final_data['lr'] = test_lr_2[:,1]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

In [ ]:
```
test_preds = final_model.predict(test_final_data)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureW
arning: Feature names only support names that are all strings. Got feature names
with dtypes: ['int', 'str']. An error will be raised in 1.2.
  FutureWarning,
```

In [ ]:
```
test_guess = pd.DataFrame(test_preds)
test_guess['predict'] = 0
test_guess.loc[test_guess[0]>0.5,"predict"]=1
```

In [ ]:
```
precision = precision_score(test_guess['predict'],y_test)
recall = recall_score(test_guess['predict'],y_test)
accuracy = accuracy_score(test_guess['predict'],y_test)

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

```
Accuracy: 0.92999
Precision: 0.90577
Recall: 0.91998
```

In [ ]:
```
confusion_matrix_ens = confusion_matrix(y_test, test_guess['predict'])
print(confusion_matrix_ens)
print((confusion_matrix_ens[0,1]*100 + confusion_matrix_ens[1,0]*250)/len(y_test
```

```
[[17849  1010]
 [ 1208 11612]]
12.721361154076833
```