Get the data

In [ ]:
```
!gdown --id 1HVSazFk8m553VWPjFnZZ-YfJA_KecPea
!unzip translated_data_updated.zip
```

```
Downloading...
From: https://drive.google.com/uc?id=1HVSazFk8m553VWPjFnZZ-YfJA_KecPea
To: /content/translated_data_updated.zip
100% 122M/122M [00:00<00:00, 126MB/s]
Archive:  translated_data_updated.zip
   creating: data_translated/
  inflating: data_translated/coupon_visit_train.csv
  inflating: data_translated/coupon_list_train.csv
  inflating: data_translated/prefecture_locations.csv
  inflating: data_translated/coupon_area_test.csv
  inflating: data_translated/coupon_detail_train.csv
  inflating: data_translated/coupon_area_train.csv
  inflating: data_translated/user_list.csv
  inflating: data_translated/coupon_list_test.csv
```

In [ ]:
```
# imports
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

sns.set_theme(context='notebook', style='darkgrid')
mpl.rcParams['figure.figsize'] = (12, 10)
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
```

In [ ]:
```
# Important Note:
# Visits = browsing history in the training period. No test set available.
# Purchases = purchase history in the training period. No test set available.

df_users        = pd.read_csv('data_translated/user_list.csv')
df_c_list_train = pd.read_csv('data_translated/coupon_list_train.csv')
df_c_list_test  = pd.read_csv('data_translated/coupon_list_test.csv')
df_area_train   = pd.read_csv('data_translated/coupon_area_train.csv')
df_area_test    = pd.read_csv('data_translated/coupon_area_test.csv')
df_visit_train  = pd.read_csv('data_translated/coupon_visit_train.csv')
df_purch_train  = pd.read_csv('data_translated/coupon_detail_train.csv')
df_locations    = pd.read_csv('data_translated/prefecture_locations.csv')
```

Feature Engineering

User List

In [ ]:
```
# rename SEX_ID column, change to categorical value (0 Male, 1 Female)
df_users['SEX'] = df_users['SEX_ID'].replace('f', 1)
df_users['SEX'] = df_users['SEX'].replace('m', 0)
```

In [ ]:
```
# create a categorical variable for age group:
# 14-21, 22-35, 36-49, 50-65, 66-75, 76-90
def age_cat(age):
  if age <= 21:
    return 0
  elif age <= 35:
    return 1
  elif age <= 49:
    return 2
  elif age <= 65:
    return 3
  elif age <= 75:
    return 4
  elif age <= 90:
    return 5
  else:
    return 6

lbl_age_ranges = ['14-21', '22-35', '36-49', '50-65', '66-75', '76-90']

df_users['AGE_GROUP'] = [age_cat(a) for a in df_users['AGE']]
```

Data Preparation for Model Training

We will first train a simple model that takes the following features.

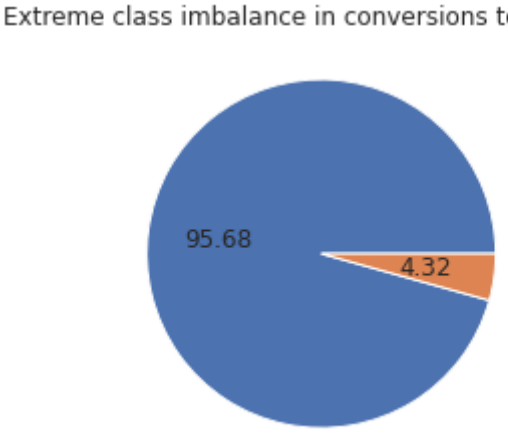
- A denormalized row of users's coupon purchases, including
  - User Gender
  - User Age
  - User Prefecture
  - Coupon Genre
  - Coupon Ken Name (Prefecture)
  - Price Rate
  - Catalog Price
  - Discount Rate
  - Ken Name

This is a model-based hybrid collaborative filtering approach.

In [ ]:
```
# Model Input Features
# For each user who purchased a coupon...

# Gender, Age, Prefecture, Coupon Genre, Coupon Prefecture, Price Rate, Catalog

############################
# BUILD DF_TRAIN DATAFRAME  #
############################
df_visit_train = df_visit_train.rename(columns={'VIEW_COUPON_ID_hash': 'COUPON_I
df_train = df_visit_train.join(df_users.set_index('USER_ID_hash'), on='USER_ID_h
df_train = df_train.join(df_c_list_train.set_index('COUPON_ID_hash'), on='COUPON
```

In [ ]:
```
# get a subset of the training columns and rename them
df_train = df_train[['AGE_GROUP', 'SEX', 'PREF_NAME_EN', 'KEN_NAME_EN', 'GENRE_N
df_train.columns = ['age_group', 'sex', 'user_prefecture', 'coupon_prefecture',
# NaN preprocessing
df_train = df_train.fillna(0)
```

In [ ]:
```
stats = df_train.groupby('purchased').size().to_frame().reset_index()
stats.columns = ['purchased', 'count']

fig, ax = plt.subplots()
ax.pie(stats['count'], autopct='%.2f')
plt.title('Extreme class imbalance in conversions to purchase')
plt.show()

pos = stats[stats['purchased'] == 1].agg('sum')['count']
neg = stats[stats['purchased'] == 0].agg('sum')['count']
total = pos + neg
print(f'Purchased: {pos} ({round(pos/total * 100, 2)}% of total)')
print(f"Visited but didn't purchase: {neg}")
```




```
Purchased: 122389 (4.32% of total)
Visited but didn't purchase: 2710791
```

In [ ]:
```
# one-hot encode training data categorical columns
df_train_enc = pd.get_dummies(df_train, columns=['age_group', 'sex', 'user_prefe
df_train_enc.head(5)
```

Out[ ]:

	discount_rate	discount_price	purchased	age_group_0	age_group_1	age_group_2	age_grou
0	78.0	1575.0	0	0	1	0	
1	78.0	1575.0	0	0	1	0	
2	66.0	1480.0	0	0	1	0	
3	80.0	1990.0	0	0	1	0	
4	85.0	4980.0	0	0	1	0	

5 rows × 146 columns

In [ ]:

```
```

In [ ]:

```
# Train test split with stratified sampling on the positive class.
# Make sure each set has the same (though very small) ratio of positive instance
df_training, df_test = train_test_split(df_train_enc, test_size=0.2, stratify=df
df_training, df_validation = train_test_split(df_training, test_size=0.2, strati

train_labels = np.array(df_training.pop('purchased'))
val_labels = np.array(df_validation.pop('purchased'))
test_labels = np.array(df_test.pop('purchased'))

train_features = np.array(df_training)
val_features = np.array(df_validation)
test_features = np.array(df_test)
```

In [ ]:

```
# Normalize the input features using StandardScaler to mitigate class imbalance
# Only *fit* the scaler on the training data, but transform the other two sets.
scaler = StandardScaler()
train_features = scaler.fit_transform(train_features)
val_features = scaler.transform(val_features)
test_features = scaler.transform(test_features)
```

In [ ]:

```
print(f'Train features shape: {train_features.shape}')
print(f'Validation features shape: {val_features.shape}')
print(f'Test features shape: {test_features.shape}')
print(f'Train labels shape: {train_labels.shape}')
print(f'Validation labels shape: {val_labels.shape}')
print(f'Test labels shape: {test_labels.shape}')

print(f'Train columns: {df_training.columns} (len: {len(df_training.columns)}')
```

```
Train features shape: (1813235, 145)
Validation features shape: (453309, 145)
Test features shape: (566636, 145)
Train labels shape: (1813235,)
Validation labels shape: (453309,)
Test labels shape: (566636,)
Train columns: Index(['discount_rate', 'discount_price', 'age_group_0', 'age_gro
up_1',
       'age_group_2', 'age_group_3', 'age_group_4', 'age_group_5', 'sex_0',
       'sex_1',
       ...
       'capsule_Lodge', 'capsule_Nail and eye salon', 'capsule_Other coupon',
       'capsule_Public inn', 'capsule_Relaxation', 'capsule_Resort inn',
       'capsule_Restaurant', 'capsule_Spa', 'capsule_Vacation rental',
       'capsule_Web service'],
      dtype='object', length=145) (len: 145
```

In [ ]:

```
# Build a binary classification model
METRICS = [
      keras.metrics.Precision(name='precision'),
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
```

```
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]

def make_model(metrics=METRICS, output_bias=None):
  if output_bias is not None:
    output_bias = tf.keras.initializers.Constant(output_bias)

  model = keras.Sequential([
      keras.layers.Dense(16, activation='relu',
                         input_shape=(train_features.shape[-1],)),
      keras.layers.Dropout(0.2),
      keras.layers.Dense(16, activation='relu'),
      keras.layers.Dropout(0.2),
      keras.layers.Dense(1, activation='sigmoid',
                         bias_initializer=output_bias),
  ])

  model.compile(
      optimizer=keras.optimizers.Adam(learning_rate=1e-3),
      loss=keras.losses.BinaryCrossentropy(),
      metrics=metrics)

  return model
```

In [ ]:

```
# build the untrained model
EPOCHS = 100
BATCH_SIZE = 2048

# early stopping
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_prc',
    verbose=1,
    patience=10,
    mode='max',
    restore_best_weights=True)

model = make_model()
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 16)                2336
_________________________________________________________________
dropout_6 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_10 (Dense)             (None, 16)                272
_________________________________________________________________
dropout_7 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 17
=================================================================
Total params: 2,625
Trainable params: 2,625
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
# test run with no training - these weights should be unrealistically high
model.predict(train_features[:10])
```

Out[ ]:
```
array([[0.78800064],
       [0.7660615 ],
       [0.7773887 ],
       [0.771608  ],
       [0.80279243],
       [0.72266585],
       [0.66574454],
       [0.4658518 ],
       [0.7539464 ],
       [0.73523104]], dtype=float32)
```

In [ ]:
```
results = model.evaluate(train_features, train_labels, batch_size=BATCH_SIZE, ve
print("Loss with naive bias initialization: {:0.4f}".format(results[0]))
```

```
Loss with naive bias initialization: 1.2691
```

In [ ]:
```
# set a slightly smarter initial bias to improve learning
# these weights should be much more reasonable
initial_bias = np.log([pos/neg])
model = make_model(output_bias=initial_bias)
model.predict(train_features[:10])
```

Out[ ]:
```
array([[0.11360571],
       [0.05527395],
       [0.08539504],
       [0.07177296],
       [0.10174784],
       [0.06554922],
       [0.04680771],
       [0.04310331],
       [0.08703664],
       [0.06663606]], dtype=float32)
```

In [ ]:
```
results = model.evaluate(train_features, train_labels, batch_size=BATCH_SIZE, ve
print("Loss with careful bias init (should be way better): {:0.4f}".format(resul
# save the initial weights for better training
model.save_weights('initial_weights')
```

```
Loss with careful bias init (should be way better): 0.1996
```

In [ ]:
```
# Compare model losses with and without bias initialization
def plot_loss(history, label, n):
  # Use a log scale on y-axis to show the wide range of values.
  plt.semilogy(history.epoch, history.history['loss'],
               color=colors[n], label='Train ' + label)
  plt.semilogy(history.epoch, history.history['val_loss'],
               color=colors[n], label='Val ' + label,
               linestyle="--")
  plt.legend()
  plt.xlabel('Epoch')
  plt.ylabel('Loss')
```

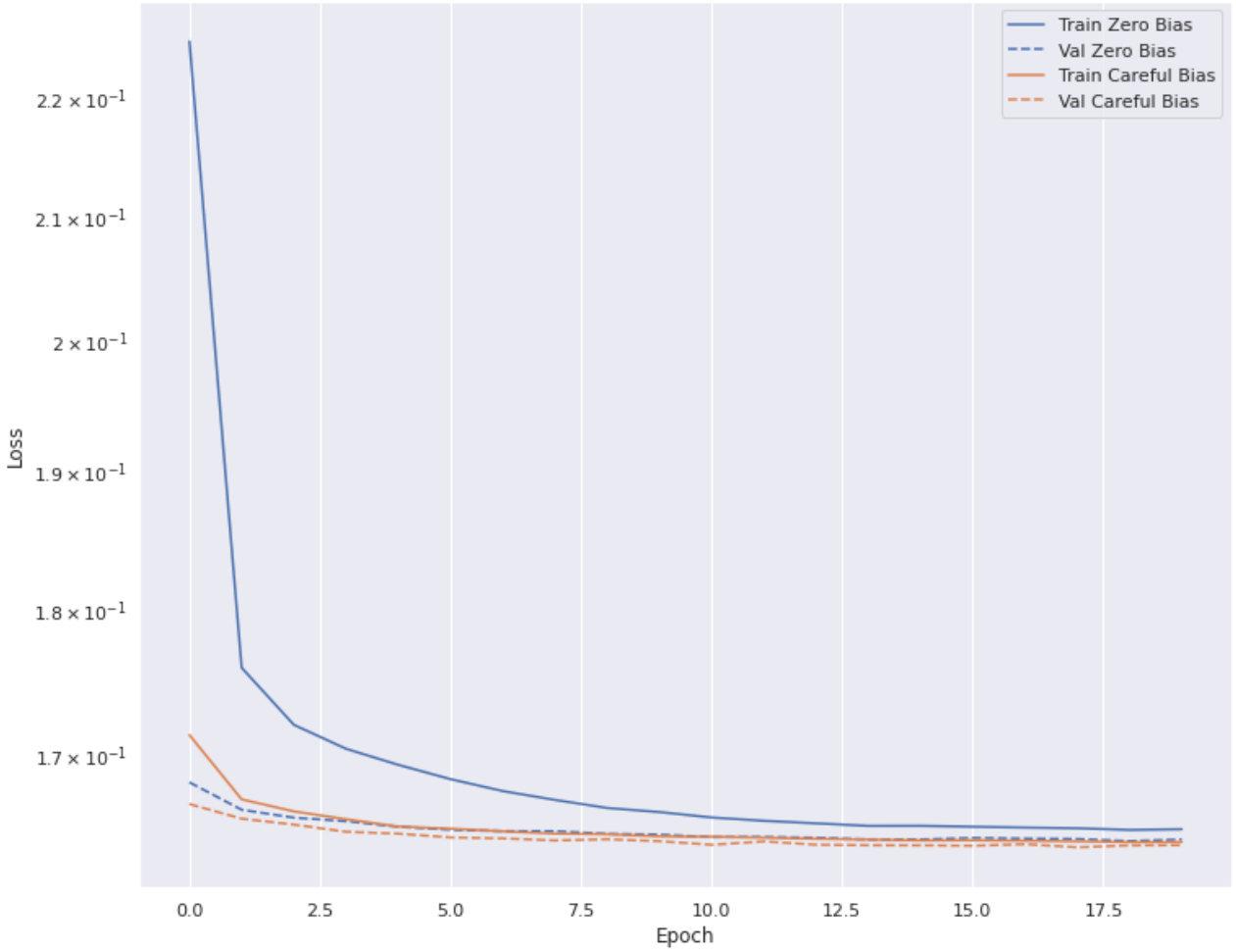
In [ ]:
```
# naive bias @ 20 epochs
model = make_model()
```

```
model.load_weights('initial_weights')
model.layers[-1].bias.assign([0.0])
zero_bias_history = model.fit(
    train_features,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=20,
    validation_data=(val_features, val_labels),
    verbose=0)

# careful bias @ 20 epochs
model = make_model()
model.load_weights('initial_weights')
careful_bias_history = model.fit(
    train_features,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=20,
    validation_data=(val_features, val_labels),
    verbose=0)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-42-f4d209eaf678> in <module>()
     33     verbose=0)
     34
---> 35 plot_loss(zero_bias_history, "Zero Bias", 0)
     36 plot_loss(careful_bias_history, "Careful Bias", 1)

<ipython-input-42-f4d209eaf678> in plot_loss(history, label, n)
      3   # Use a log scale on y-axis to show the wide range of values.
      4   plt.semilogy(history.epoch, history.history['loss'],
----> 5                color=colors[n], label='Train ' + label)
      6   plt.semilogy(history.epoch, history.history['val_loss'],
      7                color=colors[n], label='Val ' + label,

NameError: name 'colors' is not defined
```

In [ ]:

```
plot_loss(zero_bias_history, "Zero Bias", 0)
plot_loss(careful_bias_history, "Careful Bias", 1)
```



In [ ]:

```
# Weight the classes, so the model pays more attention to purchases
weight_neg = (1 / neg) * (total / 2.0)
weight_pos = (1 / pos) * (total / 2.0)

class_weight = {0: weight_neg, 1: weight_pos}

print('Weight for class 0: {:.2f}'.format(weight_neg))
print('Weight for class 1: {:.2f}'.format(weight_pos))
```

```
Weight for class 0: 0.52
Weight for class 1: 11.57
```

In [ ]:

```
# Train the final model with early stopping, class weights, initial bias weights

weighted_model = make_model()
weighted_model.load_weights('initial_weights')

weighted_history = weighted_model.fit(
    train_features,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    callbacks=[early_stopping],
    validation_data=(val_features, val_labels),
    # The class weights go here
    class_weight=class_weight)
```

```
Epoch 1/100
```

```
886/886 [==============================] - 13s 12ms/step - loss: 0.7300 - precis
ion: 0.0633 - tp: 38988.0000 - fp: 577235.0000 - tn: 1591398.0000 - fn: 58923.00
00 - accuracy: 0.7193 - recall: 0.3982 - auc: 0.5990 - prc: 0.0621 - val_loss:
0.5554 - val_precision: 0.0764 - val_tp: 11312.0000 - val_fp: 136729.0000 - val_
tn: 296998.0000 - val_fn: 8270.0000 - val_accuracy: 0.6801 - val_recall: 0.5777
- val_auc: 0.6955 - val_prc: 0.0908
Epoch 2/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6529 - precisi
on: 0.0663 - tp: 45560.0000 - fp: 641264.0000 - tn: 1093642.0000 - fn: 32769.000
0 - accuracy: 0.6283 - recall: 0.5816 - auc: 0.6608 - prc: 0.0748 - val_loss: 0.
5527 - val_precision: 0.0774 - val_tp: 11484.0000 - val_fp: 136979.0000 - val_t
n: 296748.0000 - val_fn: 8098.0000 - val_accuracy: 0.6800 - val_recall: 0.5865 -
val_auc: 0.7001 - val_prc: 0.0941
Epoch 3/100
886/886 [==============================] - 8s 9ms/step - loss: 0.6347 - precisio
n: 0.0685 - tp: 47869.0000 - fp: 651212.0000 - tn: 1083694.0000 - fn: 30460.0000
- accuracy: 0.6241 - recall: 0.6111 - auc: 0.6737 - prc: 0.0790 - val_loss: 0.56
30 - val_precision: 0.0773 - val_tp: 12167.0000 - val_fp: 145186.0000 - val_tn:
288541.0000 - val_fn: 7415.0000 - val_accuracy: 0.6634 - val_recall: 0.6213 - va
l_auc: 0.7052 - val_prc: 0.0966
Epoch 4/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6262 - precisi
on: 0.0705 - tp: 49164.0000 - fp: 648343.0000 - tn: 1086563.0000 - fn: 29165.000
0 - accuracy: 0.6264 - recall: 0.6277 - auc: 0.6833 - prc: 0.0827 - val_loss: 0.
5984 - val_precision: 0.0739 - val_tp: 13521.0000 - val_fp: 169474.0000 - val_t
n: 264253.0000 - val_fn: 6061.0000 - val_accuracy: 0.6128 - val_recall: 0.6905 -
val_auc: 0.7089 - val_prc: 0.0993
Epoch 5/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6214 - precisi
on: 0.0716 - tp: 49904.0000 - fp: 647269.0000 - tn: 1087637.0000 - fn: 28425.000
0 - accuracy: 0.6274 - recall: 0.6371 - auc: 0.6895 - prc: 0.0855 - val_loss: 0.
5862 - val_precision: 0.0761 - val_tp: 13001.0000 - val_fp: 157835.0000 - val_t
n: 275892.0000 - val_fn: 6581.0000 - val_accuracy: 0.6373 - val_recall: 0.6639 -
val_auc: 0.7112 - val_prc: 0.0993
Epoch 6/100
886/886 [==============================] - 9s 11ms/step - loss: 0.6180 - precisi
on: 0.0724 - tp: 50427.0000 - fp: 645849.0000 - tn: 1089057.0000 - fn: 27902.000
0 - accuracy: 0.6284 - recall: 0.6438 - auc: 0.6945 - prc: 0.0872 - val_loss: 0.
6071 - val_precision: 0.0735 - val_tp: 13696.0000 - val_fp: 172647.0000 - val_t
n: 261080.0000 - val_fn: 5886.0000 - val_accuracy: 0.6062 - val_recall: 0.6994 -
val_auc: 0.7117 - val_prc: 0.1007
Epoch 7/100
886/886 [==============================] - 8s 9ms/step - loss: 0.6151 - precisio
n: 0.0732 - tp: 51429.0000 - fp: 651145.0000 - tn: 1083761.0000 - fn: 26900.0000
- accuracy: 0.6261 - recall: 0.6566 - auc: 0.6992 - prc: 0.0892 - val_loss: 0.59
84 - val_precision: 0.0748 - val_tp: 13402.0000 - val_fp: 165878.0000 - val_tn:
267849.0000 - val_fn: 6180.0000 - val_accuracy: 0.6204 - val_recall: 0.6844 - va
l_auc: 0.7129 - val_prc: 0.1009
Epoch 8/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6134 - precis
ion: 0.0734 - tp: 51962.0000 - fp: 656241.0000 - tn: 1078665.0000 - fn: 26367.00
00 - accuracy: 0.6235 - recall: 0.6634 - auc: 0.7015 - prc: 0.0901 - val_loss:
0.6192 - val_precision: 0.0723 - val_tp: 14076.0000 - val_fp: 180531.0000 - val_
tn: 253196.0000 - val_fn: 5506.0000 - val_accuracy: 0.5896 - val_recall: 0.7188
- val_auc: 0.7135 - val_prc: 0.1004
Epoch 9/100
886/886 [==============================] - 8s 10ms/step - loss: 0.6118 - precisi
on: 0.0735 - tp: 52366.0000 - fp: 659732.0000 - tn: 1075174.0000 - fn: 25963.000
0 - accuracy: 0.6218 - recall: 0.6685 - auc: 0.7038 - prc: 0.0911 - val_loss: 0.
6252 - val_precision: 0.0715 - val_tp: 14204.0000 - val_fp: 184554.0000 - val_t
n: 249173.0000 - val_fn: 5378.0000 - val_accuracy: 0.5810 - val_recall: 0.7254 -
val_auc: 0.7126 - val_prc: 0.1011
Epoch 10/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6107 - precisi
on: 0.0738 - tp: 53095.0000 - fp: 666501.0000 - tn: 1068405.0000 - fn: 25234.000
```

```
0 - accuracy: 0.6185 - recall: 0.6778 - auc: 0.7055 - prc: 0.0921 - val_loss: 0.
5953 - val_precision: 0.0755 - val_tp: 13289.0000 - val_fp: 162712.0000 - val_t
n: 271015.0000 - val_fn: 6293.0000 - val_accuracy: 0.6272 - val_recall: 0.6786 -
val_auc: 0.7139 - val_prc: 0.1008
Epoch 11/100
886/886 [==============================] - 9s 11ms/step - loss: 0.6099 - precisi
on: 0.0737 - tp: 53520.0000 - fp: 673082.0000 - tn: 1061824.0000 - fn: 24809.000
0 - accuracy: 0.6151 - recall: 0.6833 - auc: 0.7066 - prc: 0.0930 - val_loss: 0.
5959 - val_precision: 0.0750 - val_tp: 13492.0000 - val_fp: 166318.0000 - val_t
n: 267409.0000 - val_fn: 6090.0000 - val_accuracy: 0.6197 - val_recall: 0.6890 -
val_auc: 0.7144 - val_prc: 0.1008
Epoch 12/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6093 - precisi
on: 0.0738 - tp: 53573.0000 - fp: 672300.0000 - tn: 1062606.0000 - fn: 24756.000
0 - accuracy: 0.6156 - recall: 0.6839 - auc: 0.7077 - prc: 0.0933 - val_loss: 0.
6220 - val_precision: 0.0729 - val_tp: 14033.0000 - val_fp: 178406.0000 - val_t
n: 255321.0000 - val_fn: 5549.0000 - val_accuracy: 0.5942 - val_recall: 0.7166 -
val_auc: 0.7149 - val_prc: 0.1025
Epoch 13/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6087 - precisi
on: 0.0740 - tp: 53772.0000 - fp: 672978.0000 - tn: 1061928.0000 - fn: 24557.000
0 - accuracy: 0.6153 - recall: 0.6865 - auc: 0.7083 - prc: 0.0940 - val_loss: 0.
6216 - val_precision: 0.0723 - val_tp: 14114.0000 - val_fp: 181226.0000 - val_t
n: 252501.0000 - val_fn: 5468.0000 - val_accuracy: 0.5882 - val_recall: 0.7208 -
val_auc: 0.7145 - val_prc: 0.1019
Epoch 14/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6080 - precis
ion: 0.0739 - tp: 54047.0000 - fp: 677026.0000 - tn: 1057880.0000 - fn: 24282.00
00 - accuracy: 0.6132 - recall: 0.6900 - auc: 0.7096 - prc: 0.0951 - val_loss:
0.5988 - val_precision: 0.0748 - val_tp: 13500.0000 - val_fp: 166983.0000 - val_
tn: 266744.0000 - val_fn: 6082.0000 - val_accuracy: 0.6182 - val_recall: 0.6894
- val_auc: 0.7154 - val_prc: 0.1017
Epoch 15/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6077 - precisi
on: 0.0741 - tp: 54311.0000 - fp: 678241.0000 - tn: 1056665.0000 - fn: 24018.000
0 - accuracy: 0.6127 - recall: 0.6934 - auc: 0.7104 - prc: 0.0952 - val_loss: 0.
5974 - val_precision: 0.0750 - val_tp: 13540.0000 - val_fp: 166900.0000 - val_t
n: 266827.0000 - val_fn: 6042.0000 - val_accuracy: 0.6185 - val_recall: 0.6915 -
val_auc: 0.7154 - val_prc: 0.1030
Epoch 16/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6076 - precisi
on: 0.0738 - tp: 54228.0000 - fp: 680140.0000 - tn: 1054766.0000 - fn: 24101.000
0 - accuracy: 0.6116 - recall: 0.6923 - auc: 0.7105 - prc: 0.0951 - val_loss: 0.
6000 - val_precision: 0.0743 - val_tp: 13748.0000 - val_fp: 171306.0000 - val_t
n: 262421.0000 - val_fn: 5834.0000 - val_accuracy: 0.6092 - val_recall: 0.7021 -
val_auc: 0.7159 - val_prc: 0.1032
Epoch 17/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6072 - precisi
on: 0.0738 - tp: 54771.0000 - fp: 687017.0000 - tn: 1047889.0000 - fn: 23558.000
0 - accuracy: 0.6081 - recall: 0.6992 - auc: 0.7110 - prc: 0.0953 - val_loss: 0.
6217 - val_precision: 0.0725 - val_tp: 14094.0000 - val_fp: 180342.0000 - val_t
n: 253385.0000 - val_fn: 5488.0000 - val_accuracy: 0.5901 - val_recall: 0.7197 -
val_auc: 0.7154 - val_prc: 0.1031
Epoch 18/100
886/886 [==============================] - 8s 10ms/step - loss: 0.6070 - precisi
on: 0.0739 - tp: 54634.0000 - fp: 685052.0000 - tn: 1049854.0000 - fn: 23695.000
0 - accuracy: 0.6091 - recall: 0.6975 - auc: 0.7113 - prc: 0.0959 - val_loss: 0.
6198 - val_precision: 0.0725 - val_tp: 14129.0000 - val_fp: 180664.0000 - val_t
n: 253063.0000 - val_fn: 5453.0000 - val_accuracy: 0.5894 - val_recall: 0.7215 -
val_auc: 0.7159 - val_prc: 0.1033
Epoch 19/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6068 - precisi
on: 0.0738 - tp: 54866.0000 - fp: 688602.0000 - tn: 1046304.0000 - fn: 23463.000
0 - accuracy: 0.6073 - recall: 0.7005 - auc: 0.7119 - prc: 0.0962 - val_loss: 0.
6178 - val_precision: 0.0725 - val_tp: 14123.0000 - val_fp: 180795.0000 - val_t
```

```
n: 252932.0000 - val_fn: 5459.0000 - val_accuracy: 0.5891 - val_recall: 0.7212 -
val_auc: 0.7159 - val_prc: 0.1035
Epoch 20/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6066 - precis
ion: 0.0737 - tp: 54949.0000 - fp: 690572.0000 - tn: 1044334.0000 - fn: 23380.00
00 - accuracy: 0.6063 - recall: 0.7015 - auc: 0.7120 - prc: 0.0964 - val_loss:
0.6082 - val_precision: 0.0732 - val_tp: 13945.0000 - val_fp: 176463.0000 - val_
tn: 257264.0000 - val_fn: 5637.0000 - val_accuracy: 0.5983 - val_recall: 0.7121
- val_auc: 0.7159 - val_prc: 0.1038
Epoch 21/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6063 - precisi
on: 0.0737 - tp: 55074.0000 - fp: 692437.0000 - tn: 1042469.0000 - fn: 23255.000
0 - accuracy: 0.6053 - recall: 0.7031 - auc: 0.7123 - prc: 0.0968 - val_loss: 0.
6116 - val_precision: 0.0730 - val_tp: 14129.0000 - val_fp: 179321.0000 - val_t
n: 254406.0000 - val_fn: 5453.0000 - val_accuracy: 0.5924 - val_recall: 0.7215 -
val_auc: 0.7161 - val_prc: 0.1038
Epoch 22/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6061 - precis
ion: 0.0734 - tp: 55551.0000 - fp: 701561.0000 - tn: 1033345.0000 - fn: 22778.00
00 - accuracy: 0.6005 - recall: 0.7092 - auc: 0.7126 - prc: 0.0970 - val_loss:
0.6092 - val_precision: 0.0734 - val_tp: 13925.0000 - val_fp: 175837.0000 - val_
tn: 257890.0000 - val_fn: 5657.0000 - val_accuracy: 0.5996 - val_recall: 0.7111
- val_auc: 0.7159 - val_prc: 0.1034
Epoch 23/100
886/886 [==============================] - 9s 11ms/step - loss: 0.6058 - precisi
on: 0.0737 - tp: 55503.0000 - fp: 697823.0000 - tn: 1037083.0000 - fn: 22826.000
0 - accuracy: 0.6026 - recall: 0.7086 - auc: 0.7131 - prc: 0.0973 - val_loss: 0.
6109 - val_precision: 0.0729 - val_tp: 14040.0000 - val_fp: 178516.0000 - val_t
n: 255211.0000 - val_fn: 5542.0000 - val_accuracy: 0.5940 - val_recall: 0.7170 -
val_auc: 0.7160 - val_prc: 0.1034
Epoch 24/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6053 - precisi
on: 0.0738 - tp: 55695.0000 - fp: 699259.0000 - tn: 1035647.0000 - fn: 22634.000
0 - accuracy: 0.6019 - recall: 0.7110 - auc: 0.7139 - prc: 0.0971 - val_loss: 0.
5961 - val_precision: 0.0739 - val_tp: 13851.0000 - val_fp: 173702.0000 - val_t
n: 260025.0000 - val_fn: 5731.0000 - val_accuracy: 0.6042 - val_recall: 0.7073 -
val_auc: 0.7166 - val_prc: 0.1044
Epoch 25/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6056 - precisi
on: 0.0736 - tp: 55732.0000 - fp: 701735.0000 - tn: 1033171.0000 - fn: 22597.000
0 - accuracy: 0.6005 - recall: 0.7115 - auc: 0.7135 - prc: 0.0978 - val_loss: 0.
6158 - val_precision: 0.0730 - val_tp: 14134.0000 - val_fp: 179545.0000 - val_t
n: 254182.0000 - val_fn: 5448.0000 - val_accuracy: 0.5919 - val_recall: 0.7218 -
val_auc: 0.7166 - val_prc: 0.1031
Epoch 26/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6053 - precisi
on: 0.0734 - tp: 55924.0000 - fp: 705730.0000 - tn: 1029176.0000 - fn: 22405.000
0 - accuracy: 0.5984 - recall: 0.7140 - auc: 0.7139 - prc: 0.0978 - val_loss: 0.
6006 - val_precision: 0.0739 - val_tp: 13792.0000 - val_fp: 172931.0000 - val_t
n: 260796.0000 - val_fn: 5790.0000 - val_accuracy: 0.6057 - val_recall: 0.7043 -
val_auc: 0.7165 - val_prc: 0.1037
Epoch 27/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6048 - precis
ion: 0.0738 - tp: 55901.0000 - fp: 701949.0000 - tn: 1032957.0000 - fn: 22428.00
00 - accuracy: 0.6005 - recall: 0.7137 - auc: 0.7146 - prc: 0.0989 - val_loss:
0.6110 - val_precision: 0.0722 - val_tp: 14239.0000 - val_fp: 183076.0000 - val_
tn: 250651.0000 - val_fn: 5343.0000 - val_accuracy: 0.5843 - val_recall: 0.7271
- val_auc: 0.7168 - val_prc: 0.1050
Epoch 28/100
886/886 [==============================] - 10s 12ms/step - loss: 0.6051 - precis
ion: 0.0735 - tp: 56062.0000 - fp: 707062.0000 - tn: 1027844.0000 - fn: 22267.00
00 - accuracy: 0.5978 - recall: 0.7157 - auc: 0.7144 - prc: 0.0982 - val_loss:
0.6155 - val_precision: 0.0730 - val_tp: 14072.0000 - val_fp: 178660.0000 - val_
tn: 255067.0000 - val_fn: 5510.0000 - val_accuracy: 0.5937 - val_recall: 0.7186
- val_auc: 0.7169 - val_prc: 0.1040
```

```
Epoch 29/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6051 - precisi
on: 0.0734 - tp: 55951.0000 - fp: 706657.0000 - tn: 1028249.0000 - fn: 22378.000
0 - accuracy: 0.5979 - recall: 0.7143 - auc: 0.7141 - prc: 0.0981 - val_loss: 0.
6034 - val_precision: 0.0733 - val_tp: 14043.0000 - val_fp: 177477.0000 - val_t
n: 256250.0000 - val_fn: 5539.0000 - val_accuracy: 0.5963 - val_recall: 0.7171 -
val_auc: 0.7170 - val_prc: 0.1043
Epoch 30/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6048 - precis
ion: 0.0735 - tp: 56059.0000 - fp: 706724.0000 - tn: 1028182.0000 - fn: 22270.00
00 - accuracy: 0.5980 - recall: 0.7157 - auc: 0.7145 - prc: 0.0986 - val_loss:
0.6079 - val_precision: 0.0729 - val_tp: 14133.0000 - val_fp: 179721.0000 - val_
tn: 254006.0000 - val_fn: 5449.0000 - val_accuracy: 0.5915 - val_recall: 0.7217
- val_auc: 0.7173 - val_prc: 0.1045
Epoch 31/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6047 - precis
ion: 0.0735 - tp: 56069.0000 - fp: 706732.0000 - tn: 1028174.0000 - fn: 22260.00
00 - accuracy: 0.5980 - recall: 0.7158 - auc: 0.7148 - prc: 0.0989 - val_loss:
0.6171 - val_precision: 0.0718 - val_tp: 14397.0000 - val_fp: 186082.0000 - val_
tn: 247645.0000 - val_fn: 5185.0000 - val_accuracy: 0.5781 - val_recall: 0.7352
- val_auc: 0.7169 - val_prc: 0.1044
Epoch 32/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6048 - precisi
on: 0.0734 - tp: 56105.0000 - fp: 707870.0000 - tn: 1027036.0000 - fn: 22224.000
0 - accuracy: 0.5974 - recall: 0.7163 - auc: 0.7146 - prc: 0.0990 - val_loss: 0.
6135 - val_precision: 0.0730 - val_tp: 14061.0000 - val_fp: 178539.0000 - val_t
n: 255188.0000 - val_fn: 5521.0000 - val_accuracy: 0.5940 - val_recall: 0.7181 -
val_auc: 0.7176 - val_prc: 0.1053
Epoch 33/100
886/886 [==============================] - 8s 9ms/step - loss: 0.6043 - precisio
n: 0.0738 - tp: 55879.0000 - fp: 701129.0000 - tn: 1033777.0000 - fn: 22450.0000
- accuracy: 0.6009 - recall: 0.7134 - auc: 0.7153 - prc: 0.0991 - val_loss: 0.61
67 - val_precision: 0.0718 - val_tp: 14358.0000 - val_fp: 185706.0000 - val_tn:
248021.0000 - val_fn: 5224.0000 - val_accuracy: 0.5788 - val_recall: 0.7332 - va
l_auc: 0.7171 - val_prc: 0.1051
Epoch 34/100
886/886 [==============================] - 8s 9ms/step - loss: 0.6048 - precisio
n: 0.0737 - tp: 55938.0000 - fp: 703440.0000 - tn: 1031466.0000 - fn: 22391.0000
- accuracy: 0.5997 - recall: 0.7141 - auc: 0.7146 - prc: 0.0986 - val_loss: 0.59
75 - val_precision: 0.0738 - val_tp: 13907.0000 - val_fp: 174543.0000 - val_tn:
259184.0000 - val_fn: 5675.0000 - val_accuracy: 0.6024 - val_recall: 0.7102 - va
l_auc: 0.7166 - val_prc: 0.1047
Epoch 35/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6043 - precisi
on: 0.0737 - tp: 56047.0000 - fp: 704929.0000 - tn: 1029977.0000 - fn: 22282.000
0 - accuracy: 0.5989 - recall: 0.7155 - auc: 0.7154 - prc: 0.0990 - val_loss: 0.
6058 - val_precision: 0.0732 - val_tp: 14060.0000 - val_fp: 177935.0000 - val_t
n: 255792.0000 - val_fn: 5522.0000 - val_accuracy: 0.5953 - val_recall: 0.7180 -
val_auc: 0.7166 - val_prc: 0.1042
Epoch 36/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6047 - precisi
on: 0.0735 - tp: 56073.0000 - fp: 707003.0000 - tn: 1027903.0000 - fn: 22256.000
0 - accuracy: 0.5978 - recall: 0.7159 - auc: 0.7151 - prc: 0.0992 - val_loss: 0.
6006 - val_precision: 0.0736 - val_tp: 13982.0000 - val_fp: 175940.0000 - val_t
n: 257787.0000 - val_fn: 5600.0000 - val_accuracy: 0.5995 - val_recall: 0.7140 -
val_auc: 0.7172 - val_prc: 0.1053
Epoch 37/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6044 - precisi
on: 0.0735 - tp: 56314.0000 - fp: 709555.0000 - tn: 1025351.0000 - fn: 22015.000
0 - accuracy: 0.5965 - recall: 0.7189 - auc: 0.7153 - prc: 0.0992 - val_loss: 0.
6209 - val_precision: 0.0717 - val_tp: 14356.0000 - val_fp: 185850.0000 - val_t
n: 247877.0000 - val_fn: 5226.0000 - val_accuracy: 0.5785 - val_recall: 0.7331 -
val_auc: 0.7167 - val_prc: 0.1042
Epoch 38/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6043 - precisi
```

```
on: 0.0735 - tp: 56206.0000 - fp: 708123.0000 - tn: 1026783.0000 - fn: 22123.000
0 - accuracy: 0.5973 - recall: 0.7176 - auc: 0.7153 - prc: 0.0992 - val_loss: 0.
6101 - val_precision: 0.0727 - val_tp: 14167.0000 - val_fp: 180612.0000 - val_t
n: 253115.0000 - val_fn: 5415.0000 - val_accuracy: 0.5896 - val_recall: 0.7235 -
val_auc: 0.7175 - val_prc: 0.1053
Epoch 39/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6044 - precisi
on: 0.0735 - tp: 56154.0000 - fp: 707328.0000 - tn: 1027578.0000 - fn: 22175.000
0 - accuracy: 0.5977 - recall: 0.7169 - auc: 0.7154 - prc: 0.0993 - val_loss: 0.
6165 - val_precision: 0.0725 - val_tp: 14186.0000 - val_fp: 181595.0000 - val_t
n: 252132.0000 - val_fn: 5396.0000 - val_accuracy: 0.5875 - val_recall: 0.7244 -
val_auc: 0.7173 - val_prc: 0.1057
Epoch 40/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6043 - precisi
on: 0.0734 - tp: 56490.0000 - fp: 712903.0000 - tn: 1022003.0000 - fn: 21839.000
0 - accuracy: 0.5948 - recall: 0.7212 - auc: 0.7154 - prc: 0.0993 - val_loss: 0.
6027 - val_precision: 0.0742 - val_tp: 13837.0000 - val_fp: 172662.0000 - val_t
n: 261065.0000 - val_fn: 5745.0000 - val_accuracy: 0.6064 - val_recall: 0.7066 -
val_auc: 0.7169 - val_prc: 0.1052
Epoch 41/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6040 - precis
ion: 0.0737 - tp: 56013.0000 - fp: 704220.0000 - tn: 1030686.0000 - fn: 22316.00
00 - accuracy: 0.5993 - recall: 0.7151 - auc: 0.7159 - prc: 0.0992 - val_loss:
0.6171 - val_precision: 0.0721 - val_tp: 14267.0000 - val_fp: 183543.0000 - val_
tn: 250184.0000 - val_fn: 5315.0000 - val_accuracy: 0.5834 - val_recall: 0.7286
- val_auc: 0.7173 - val_prc: 0.1044
Epoch 42/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6043 - precisi
on: 0.0733 - tp: 56158.0000 - fp: 709896.0000 - tn: 1025010.0000 - fn: 22171.000
0 - accuracy: 0.5963 - recall: 0.7170 - auc: 0.7155 - prc: 0.0992 - val_loss: 0.
6064 - val_precision: 0.0729 - val_tp: 14077.0000 - val_fp: 179129.0000 - val_t
n: 254598.0000 - val_fn: 5505.0000 - val_accuracy: 0.5927 - val_recall: 0.7189 -
val_auc: 0.7175 - val_prc: 0.1060
Epoch 43/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6042 - precisi
on: 0.0738 - tp: 56169.0000 - fp: 705421.0000 - tn: 1029485.0000 - fn: 22160.000
0 - accuracy: 0.5987 - recall: 0.7171 - auc: 0.7157 - prc: 0.0989 - val_loss: 0.
5989 - val_precision: 0.0726 - val_tp: 14168.0000 - val_fp: 181036.0000 - val_t
n: 252691.0000 - val_fn: 5414.0000 - val_accuracy: 0.5887 - val_recall: 0.7235 -
val_auc: 0.7177 - val_prc: 0.1063
Epoch 44/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6036 - precis
ion: 0.0736 - tp: 56422.0000 - fp: 710049.0000 - tn: 1024857.0000 - fn: 21907.00
00 - accuracy: 0.5963 - recall: 0.7203 - auc: 0.7162 - prc: 0.0996 - val_loss:
0.6048 - val_precision: 0.0734 - val_tp: 13993.0000 - val_fp: 176613.0000 - val_
tn: 257114.0000 - val_fn: 5589.0000 - val_accuracy: 0.5981 - val_recall: 0.7146
- val_auc: 0.7174 - val_prc: 0.1058
Epoch 45/100
886/886 [==============================] - 10s 11ms/step - loss: 0.6040 - precis
ion: 0.0736 - tp: 56433.0000 - fp: 710350.0000 - tn: 1024556.0000 - fn: 21896.00
00 - accuracy: 0.5962 - recall: 0.7205 - auc: 0.7159 - prc: 0.0996 - val_loss:
0.6157 - val_precision: 0.0717 - val_tp: 14335.0000 - val_fp: 185714.0000 - val_
tn: 248013.0000 - val_fn: 5247.0000 - val_accuracy: 0.5787 - val_recall: 0.7320
- val_auc: 0.7171 - val_prc: 0.1053
Epoch 46/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6042 - precisi
on: 0.0733 - tp: 56550.0000 - fp: 714526.0000 - tn: 1020380.0000 - fn: 21779.000
0 - accuracy: 0.5939 - recall: 0.7220 - auc: 0.7155 - prc: 0.0995 - val_loss: 0.
6059 - val_precision: 0.0730 - val_tp: 14094.0000 - val_fp: 178892.0000 - val_t
n: 254835.0000 - val_fn: 5488.0000 - val_accuracy: 0.5933 - val_recall: 0.7197 -
val_auc: 0.7176 - val_prc: 0.1052
Epoch 47/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6040 - precisi
on: 0.0734 - tp: 56402.0000 - fp: 712129.0000 - tn: 1022777.0000 - fn: 21927.000
0 - accuracy: 0.5952 - recall: 0.7201 - auc: 0.7157 - prc: 0.0994 - val_loss: 0.
```

```
6181 - val_precision: 0.0718 - val_tp: 14314.0000 - val_fp: 185093.0000 - val_t
n: 248634.0000 - val_fn: 5268.0000 - val_accuracy: 0.5801 - val_recall: 0.7310 -
val_auc: 0.7173 - val_prc: 0.1056
Epoch 48/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6039 - precisi
on: 0.0734 - tp: 56374.0000 - fp: 711242.0000 - tn: 1023664.0000 - fn: 21955.000
0 - accuracy: 0.5956 - recall: 0.7197 - auc: 0.7159 - prc: 0.0994 - val_loss: 0.
6085 - val_precision: 0.0731 - val_tp: 14022.0000 - val_fp: 177860.0000 - val_t
n: 255867.0000 - val_fn: 5560.0000 - val_accuracy: 0.5954 - val_recall: 0.7161 -
val_auc: 0.7173 - val_prc: 0.1049
Epoch 49/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6037 - precisi
on: 0.0736 - tp: 56261.0000 - fp: 707908.0000 - tn: 1026998.0000 - fn: 22068.000
0 - accuracy: 0.5974 - recall: 0.7183 - auc: 0.7164 - prc: 0.1000 - val_loss: 0.
6078 - val_precision: 0.0727 - val_tp: 14038.0000 - val_fp: 179097.0000 - val_t
n: 254630.0000 - val_fn: 5544.0000 - val_accuracy: 0.5927 - val_recall: 0.7169 -
val_auc: 0.7173 - val_prc: 0.1050
Epoch 50/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6037 - precisi
on: 0.0737 - tp: 56335.0000 - fp: 708111.0000 - tn: 1026795.0000 - fn: 21994.000
0 - accuracy: 0.5973 - recall: 0.7192 - auc: 0.7163 - prc: 0.0998 - val_loss: 0.
6022 - val_precision: 0.0734 - val_tp: 13977.0000 - val_fp: 176363.0000 - val_t
n: 257364.0000 - val_fn: 5605.0000 - val_accuracy: 0.5986 - val_recall: 0.7138 -
val_auc: 0.7176 - val_prc: 0.1056
Epoch 51/100
886/886 [==============================] - 8s 9ms/step - loss: 0.6035 - precisio
n: 0.0735 - tp: 56347.0000 - fp: 710194.0000 - tn: 1024712.0000 - fn: 21982.0000
- accuracy: 0.5962 - recall: 0.7194 - auc: 0.7166 - prc: 0.0994 - val_loss: 0.61
43 - val_precision: 0.0723 - val_tp: 14211.0000 - val_fp: 182276.0000 - val_tn:
251451.0000 - val_fn: 5371.0000 - val_accuracy: 0.5861 - val_recall: 0.7257 - va
l_auc: 0.7175 - val_prc: 0.1058
Epoch 52/100
886/886 [==============================] - 8s 9ms/step - loss: 0.6034 - precisio
n: 0.0734 - tp: 56425.0000 - fp: 712806.0000 - tn: 1022100.0000 - fn: 21904.0000
- accuracy: 0.5948 - recall: 0.7204 - auc: 0.7165 - prc: 0.0994 - val_loss: 0.61
24 - val_precision: 0.0719 - val_tp: 14300.0000 - val_fp: 184612.0000 - val_tn:
249115.0000 - val_fn: 5282.0000 - val_accuracy: 0.5811 - val_recall: 0.7303 - va
l_auc: 0.7175 - val_prc: 0.1057
Epoch 53/100
886/886 [==============================] - 9s 10ms/step - loss: 0.6035 - precisi
on: 0.0730 - tp: 56815.0000 - fp: 721135.0000 - tn: 1013771.0000 - fn: 21514.000
0 - accuracy: 0.5904 - recall: 0.7253 - auc: 0.7164 - prc: 0.0998 - val_loss: 0.
6034 - val_precision: 0.0729 - val_tp: 14087.0000 - val_fp: 179259.0000 - val_t
n: 254468.0000 - val_fn: 5495.0000 - val_accuracy: 0.5924 - val_recall: 0.7194 -
val_auc: 0.7176 - val_prc: 0.1040
Restoring model weights from the end of the best epoch.
Epoch 00053: early stopping
```

Mapping coupon predictions to users and packaging the submission

In [ ]:

```
# generate a row containing user metadata with a test coupon
def get_test_row(user, coupon):
  age_group         = user.AGE_GROUP
  sex               = user.SEX
  user_prefecture   = user.PREF_NAME_EN
  coupon_prefecture = coupon.KEN_NAME_EN
  genre             = coupon.GENRE_NAME_EN
  capsule           = coupon.CAPSULE_TEXT_EN
  discount_rate     = coupon.PRICE_RATE
  discount_price    = coupon.DISCOUNT_PRICE

  row = [age_group, sex, user_prefecture, coupon_prefecture, genre, capsule,
         discount_rate, discount_price]
```

```
  return row
```

In [ ]:

```
# Use the model to predict 10 coupons for every single user
from tqdm import tqdm # this will take a while

df_join_users = df_users.fillna(0)

results = []
for i, u in tqdm(df_join_users.iterrows(), total=df_join_users.shape[0]):
  coupon_rows = []
  coupon_ids = []
  for j, c in df_c_list_test.iterrows():
    coupon_rows.append(get_test_row(u, c))
    coupon_ids.append(c.COUPON_ID_hash)

  df_user_recommended = pd.DataFrame(coupon_rows)
  df_user_recommended.columns = ['age_group', 'sex', 'user_prefecture', 'coupon_
  df_user_recommended.fillna(0)

  # one-hot encode the categorical test data columns
  coupon_rows = pd.get_dummies(df_user_recommended, columns=['age_group', 'sex',
  coupon_rows = coupon_rows.reindex(columns=df_training.columns, fill_value=0)
  coupon_rows = np.array(coupon_rows)
  coupon_rows = scaler.fit_transform(coupon_rows)

  # predict
  preds = model.predict(coupon_rows)
  preds = preds.flatten()

  df_predictions = pd.DataFrame(zip(coupon_ids, preds), columns=['coupon', 'like
  df_predictions = df_predictions.sort_values(by='likelihood', ascending=False)
  df_predictions = df_predictions.head(10)
  ten_coupon_string = ' '.join(df_predictions['coupon']).strip()
  results.append({'USER_ID_hash': u.USER_ID_hash, 'PURCHASED_COUPONS': ten_coupo

df_submission = pd.DataFrame(results)
```

```
  0%|          | 0/22873 [00:00<?, ?it/s]
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_fun
ction.<locals>.predict_function at 0x7f333ea67c20> triggered tf.function retraci
ng. Tracing is expensive and the excessive number of tracings could be due to
(1) creating @tf.function repeatedly in a loop, (2) passing tensors with differe
nt shapes, (3) passing Python objects instead of tensors. For (1), please define
your @tf.function outside of the loop. For (2), @tf.function has experimental_re
lax_shapes=True option that relaxes argument shapes that can avoid unnecessary r
etracing. For (3), please refer to https://www.tensorflow.org/guide/function#con
trolling_retracing and https://www.tensorflow.org/api_docs/python/tf/function fo
r  more details.
100%|██████████| 22873/22873 [56:23<00:00,  6.76it/s]
```

In [ ]:

```
df_submission.info()
df_submission.to_csv('submission.csv', header=True, index=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22873 entries, 0 to 22872
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   USER_ID_hash        22873 non-null  object
 1   PURCHASED_COUPONS   22873 non-null  object
dtypes: object(2)
memory usage: 357.5+ KB
```

In [ ]:
```
model.save('bruteforce_weighted_log_reg_model.pb')
```

```
INFO:tensorflow:Assets written to: bruteforce_weighted_log_reg_model.pb/assets
```

In [ ]:
```
!zip bruteforce_weighted_log_reg_model.zip bruteforce_weighted_log_reg_model.pb/
```

```
  adding: bruteforce_weighted_log_reg_model.pb/ (stored 0%)
```

In [ ]:
```

```