Machine Learning University

Predictive Analysis of Product Substitutability in Dynamic Market Ecosystems: A Multi-faceted Approach to Understanding Perceptual Similarities and Temporal Variabilities

Anjan Biswas, Sr. AI Specialist Solutions Architect, AWS AI/ML WWSO

Problem Definition:

Given a pair of products, (A, B), within a specified category or ecosystem, we say that B is a "substitute" for A not just if a customer would buy B in place of A when A is out of stock, but also if there's a perceptual similarity in their features, branding, or use-cases, which causes the customer to perceive them as interchangeable.

The objective of this project is multi-fold:

- Predict a substitute relationship between pairs of products.
- Quantify the degree of substitutability based on various features like price range, customer reviews, brand reputation, and other product attributes.
- Identify the primary drivers or attributes leading to substitutability. For instance, do customers perceive products as substitutes primarily because of price, or are functional attributes more influential?
- Evaluate the temporal dynamics of substitutability. Are some products seen as substitutes only during certain seasons or promotional periods?
- Distinguish between short-term and long-term substitutes, i.e., products that are substitutes due to temporary market conditions versus those that are perennial substitutes.

Moreover, to make predictions more robust, the model should consider external factors like market trends, competitive landscape, and customer behavioral data. The end goal is to create an adaptive system that not only predicts but also provides insights into the evolving nature of product substitutability in a complex market ecosystem.

Model submissions must be made in the form of model artifcats to the leaderboard at : **https://leaderboard.corp.amazon.com/tasks/478** . Your models must be quantized so as to fit into a smallest inferentia instance for either endpoint predictions or batch predictions.

Dataset and Files:

- **asin_product_data.csv**: Each line gives a specific product information such as ASIN, category, item name, etc. We will use this to create a feature vector for each product. This file has 112 columns, we will try to select some useful columns in this notebook because not all of them are suitable. `|Region Id|MarketPlace Id|ASIN|Binding Code|binding_description|brand_code|case_pack_quantity|, ...`

- **dataset_metadata.csv**: Provides detailed information about all 113 columns in the asin_product_data.csv

- **training.csv**: Product pairs to consider are given here. Its columns are:

> - `ID:` ID of the record
> - `key_asin:` Key product ASIN
> - `cand_asin:` Candidate product ASIN
> - `label:` Tells whether the key and candidate products are susbstitutes (1) or not (0).

---

1. Reading the Dataset

In [313…

```
import boto3
from os import path
import pandas as pd
# import xgboost as xgb

# import the datasets
bucketname = 'mlu-student-datalake' # replace with your bucket name
filename1 = 'MLA-TAB/asin_product.csv' # replace with your object key
filename2 = 'MLA-TAB/training.csv' # replace with your object key
filename3 = 'MLA-TAB/public_test_features.csv' # replace with your object key
s3 = boto3.resource('s3')
if not path.exists("asin_product.csv"):
    s3.Bucket(bucketname).download_file(filename1, 'asin_product.csv')
if not path.exists("training.csv"):
    s3.Bucket(bucketname).download_file(filename2, 'training.csv')
if not path.exists("public_test_features.csv"):
    s3.Bucket(bucketname).download_file(filename3, 'public_test_features.csv')

asin_product_data = pd.read_csv('asin_product.csv', encoding='ISO-8859-1')
training_data = pd.read_csv('training.csv')
test_data = pd.read_csv('public_test_features_new.csv')
#env.head()
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3063: DtypeWarning: Columns (18,19,23,31,38,41,48,63,78,82,85,96,105) have mixed typ
es.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3063: DtypeWarning: Columns (134,138,156,197,211) have mixed types.Specify dtype opt
ion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Let's see what our data looks like below:

"asin_product_data.csv" file gives us information about products. We will use this as our main data table to construct feature vectors for each ASIN (product) in our training and test datasets

In [ ]:
```
asin_product_data.head()
```

"training.csv" file is our training data. This file has a label 1 if the two products are subsitutes to each other and 0 otherwise.

In [ ]:
```
training_data.head()
```

"public_test_features.csv" file is the test data. Let's see what it looks like. See below that we don't have the label column in this data. We will predict the labels with our Machine Learning model.

In [ ]:
```
test_data.head()
```

In [ ]:
```
test_data.shape
```

---

2. Exploratory Data Analysis and Feature Engineering

On day 1, we recommended using numerical columns. We learned how to use **categorical data** today. Feel free to select some categorical variables such as: **"classification_code", "Binding Code", "has_ean", "has_online_play"**

List of things to do in this section:

- Select your columns of interest.
- Impute missing values. Hint: asin_product_data.isna().sum() shows number of mising records
- Apply one-hot-encoding or target encoding for your categorical variables. **Hint:** Be careful wih one-hot-encoding, it can cause very large features when you have too many categories.

In [336…
```
asin_product_data["item_package_quantity"].fillna(asin_product_data["item_packa
asin_product_data["item_height"].fillna(asin_product_data["item_height"].mean()
asin_product_data["item_width"].fillna(asin_product_data["item_width"].mean(),
asin_product_data["item_length"].fillna(asin_product_data["item_length"].mean()
asin_product_data["item_weight"].fillna(asin_product_data["item_weight"].mean()
asin_product_data["pkg_height"].fillna(asin_product_data["pkg_height"].mean(),
asin_product_data["pkg_width"].fillna(asin_product_data["pkg_width"].mean(), ir
asin_product_data["pkg_length"].fillna(asin_product_data["pkg_length"].mean(),
asin_product_data["pkg_weight"].fillna(asin_product_data["pkg_weight"].mean(),
asin_product_data["unit_count"].fillna(asin_product_data["unit_count"].mean(),
asin_product_data["fma_qualified_price_max"].fillna(asin_product_data["fma_qual
```

```
asin_product_data['item_name'].fillna("Missing",inplace=True)
asin_product_data['Binding Code'].fillna("Missing",inplace=True)
asin_product_data['has_ean'].fillna("Missing",inplace=True)
asin_product_data['has_online_play'].fillna("Missing",inplace=True)
asin_product_data['has_platform'].fillna("Missing",inplace=True)
asin_product_data['has_upc'].fillna("Missing",inplace=True)
asin_product_data['has_recommended_browse_nodes'].fillna("Missing",inplace=True
asin_product_data['product_type'].fillna("Missing",inplace=True)  #added this t
asin_product_data['classification_code'].fillna("Missing",inplace=True)
asin_product_data['ordering_channel'].fillna("Missing",inplace=True)
asin_product_data['variation_theme_description'].fillna("Missing",inplace=True)
```

In [182…

```
tmp=asin_product_data['Binding Code'].nunique()
tmp
```

Out[182]:

```
69
```

Let's do some encoding on the Text features. First we will convert Y/N to Boolean values. In this case "has_ean" and "has_online_play" are Y/N values.

In [337…

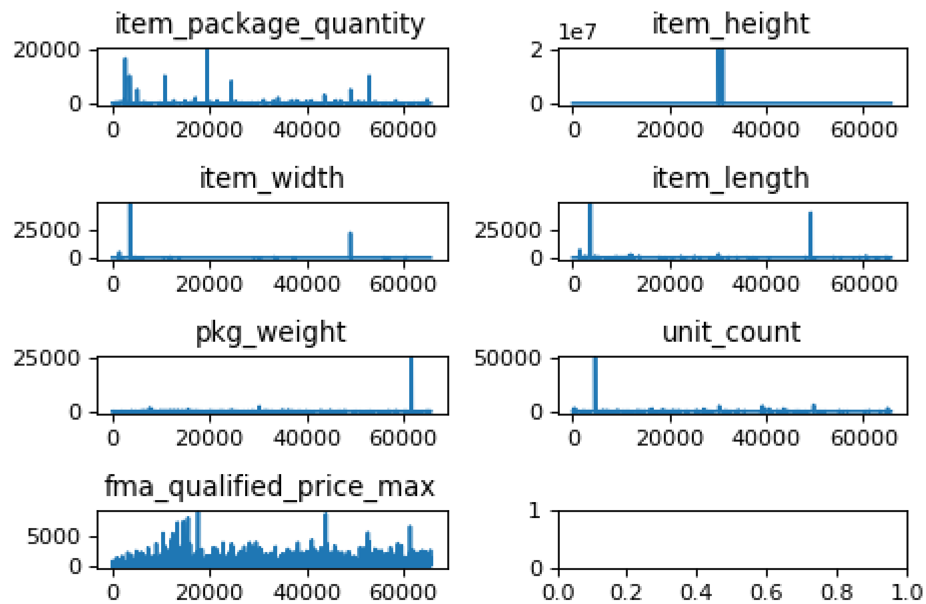
```
from sklearn import preprocessing

lb = preprocessing.LabelBinarizer()

asin_product_data['has_ean'] = lb.fit_transform(asin_product_data['has_ean'].va
asin_product_data['has_online_play'] = lb.fit_transform(asin_product_data['has_
asin_product_data['has_platform'] = lb.fit_transform(asin_product_data['has_pla
asin_product_data['has_upc'] = lb.fit_transform(asin_product_data['has_upc'].va
asin_product_data['has_recommended_browse_nodes'] = lb.fit_transform(asin_produ
```

Analyze the numerical features to see if there's potential outliers or bias

In [316…

```
import matplotlib.pyplot as plt
import seaborn as sns

norm_data = asin_product_data[['item_width',
               'item_height',
               'item_length',
               'item_package_quantity',
               'pkg_weight',
               'unit_count',
               'fma_qualified_price_max']]


fig1, axs1 = plt.subplots(4, 2)
axs1[0, 0].plot(norm_data["item_package_quantity"])
axs1[0, 0].set_title("item_package_quantity")
axs1[0, 1].plot(norm_data["item_height"])
axs1[0, 1].set_title("item_height")
axs1[1, 0].plot(norm_data["item_width"])
axs1[1, 0].set_title("item_width")
axs1[1, 1].plot(norm_data["item_length"])
axs1[1, 1].set_title("item_length")

axs1[2, 0].plot(norm_data["pkg_weight"])
axs1[2, 0].set_title("pkg_weight")
```

```python
axs1[2, 1].plot(norm_data["unit_count"])
axs1[2, 1].set_title("unit_count")
axs1[3, 0].plot(norm_data["fma_qualified_price_max"])
axs1[3, 0].set_title("fma_qualified_price_max")
fig1.set_dpi(80)
fig1.tight_layout()
```



As we can see most of the features may need scaling , so we will use `StandardScaler()` from sklearn to scale these features

---

3. Creating the feature map

Given the intricate data architecture inherent to our dataset, it becomes paramount to establish a bijective association between the ASINs (Amazon Standard Identification Numbers) and their corresponding multidimensional feature spaces. To achieve this sophisticated mapping, I propose employing a data structure known as a dictionary or a hash map. The fundamental rationale behind this selection is the dictionary's intrinsic capability for constant-time (O(1)) complexity during retrieval operations.

Such an efficient lookup mechanism becomes critically beneficial, especially during the process of amalgamating and synthesizing training and validation datasets. This strategic approach not only optimizes computational overhead but also ensures data integrity and cohesiveness throughout the data pipeline.

In [ ]:
```python
asin_product_data.head()
```

In [338…
```python
feature_map = {}

for index, row in asin_product_data.iterrows():
    #  load all features in (some are useless)
    feature_map[row["ASIN"]] = row.tolist()
```

In [339…

```
# Test
print(feature_map["1785245481"])
```

```
[1, 1, '1785245481', 'office_product', 'Office Product', nan, nan, 'base_product', 'Base Product', nan, nan, '24-Feb-16', 'USD', nan, nan, nan, nan, 9780000000000.0, nan, nan, nan, nan, 26.31, nan, 229, 'gl_office_product', 1, 0, 0, 0, 0, nan, nan, 'N', nan, 'Y', 'N', 'N', nan, 'Y', nan, '1785245481', 1.0, nan, nan, nan, nan, nan, nan, nan, nan, nan, 0.25, 11.81, 'Inspirational Quotes, Notable Quotables 2017 Monthly Wall Calendar, 12" x 12"', 1.0, 0.5, 11.81, 'en_US', 'The Gifted Stationary Co', nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, 'Missing', nan, nan, 'OFFICE_PRODUCTS', 3523, nan, nan, nan, nan, nan, nan, nan, nan, 'The Gifted Stationary Co', nan, nan, nan, nan, nan, nan, 37.247203369719905, nan, 'COLOR', 2.0, nan, nan, nan, nan, 'inches', 0.25, 11.81199999, 0.5, 'pounds', 11.81199999, nan, '25-Feb-16', '9-Jun-17', 'N', '9-Jun-17', 69, nan]
```

---

4. Getting features

In the ensuing section, I introduce the function named **getFeatures()**, which serves as the backbone for deriving and amalgamating feature vectors from distinct datasets. The overarching goal is to harness this function to extract, preprocess, and harmonize features for our training, validation, and testing datasets.

Delving deeper into the core operations of this function, it predominantly undertakes a juxtaposition of two pivotal feature vectors—pertaining to the primary product (designated as 'key') and the candidate product. To elucidate further, consider the following illustrative scenario: Given a 'key_asin_feature' vector, represented as `[0.12, 2.5, 1, 4.2]`, and a 'cand_asin_feature' vector, represented as `[0.5, 0.1, 3.2, 2.75]`, the function seamlessly melds them into a singular feature vector, thereby yielding `[0.12, 2.5, 1, 4.2, 0.5, 0.1, 3.2, 2.75]`.

This compounded vector encapsulates the nuanced characteristics of both products, setting the stage for more intricate modeling techniques down the line.

In [340…

```
def getFeatures(data_df, feature_map):
    features = []
    for index, row in data_df.iterrows():
        key_features = feature_map[row["key_asin"]]
        cand_features = feature_map[row["cand_asin"]]

        # Concatenate feature vectors
        concat_features = key_features + cand_features
        features.append(concat_features)

    return features
```

Let's use this function three times to get our traninig, validation and test features.

In [341…

```
train_features = getFeatures(training_data, feature_map)
test_features = getFeatures(test_data, feature_map)
```

In [342…

```python
# create column names for features dataframes
key_columns = ['key_' + val for val in asin_product_data.columns]
cand_columns = ['cand_' + val for val in asin_product_data.columns]
concat_columns = key_columns + cand_columns

training_features_df = pd.DataFrame(train_features, columns=concat_columns)
training_features_df['label'] = training_data['label'].values

pd.set_option('display.max_columns', None)
training_features_df.head()
```

Out[342]:

	key_Region Id	key_MarketPlace Id	key_ASIN	key_Binding Code	key_binding_description	key
0	1	1	B01L7CFUWC	Missing	NaN	
1	1	1	B01KDAKKTM	baby_product	Baby Product	
2	1	1	B013FA0UVA	toy	Toy	
3	1	1	B008KPZLEC	health_and_beauty	Health and Beauty	
4	1	1	B0196BJHXY	kitchen	Kitchen	

5. Fitting the classifier

There are two possibilities of classifiers - decision trees and random forest.

- Decision Tree classifier: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- Random Forest classifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

We Initialize the classifer and fit your training data to it below.

Identifying Features

In our data repository, we observe a plethora of categorical variables, collectively represented as $C = \{c_1, c_2, \ldots, c_n\}$, which demonstrate a heterogeneous range of cardinalities. Specifically, by defining a function (\text{card}(c_i)) to represent the cardinality of a categorical variable c_i, we identify a subset $L = \{c_i : \text{card}(c_i) \leq t\}$ of these variables manifesting low cardinality and another subset

$$H = \{c_i : \text{card}(c_i) > t\}$$

that exhibits an anomalously elevated cardinality.

Delving deeper into the typology of these categorical attributes, we discern that they predominantly fall under either textual descriptors or quantifiable numeric classifications. A rudimentary yet frequently adopted methodology to contend with these categorical variables is the utilization of one-hot encoding, symbolically represented as $OHE(x_{c_i})$ for a sample instance x from our data.

Nevertheless, while one-hot encoding is facile in its application, it inherently confronts a significant challenge pertaining to the potential explosion of dimensionality, especially in the context of features characterized by immense cardinality. Consequently, to circumvent this dimensional exacerbation and simultaneously retain pertinent information, we advocate for the application of the "Hashing Trick". This technique, mathematically represented as

$$HT(x_{c_i}) = h(x_{c_i}) \mod k$$

, orchestrates a strategic embedding of vectors, judiciously precluding the escalation of dimensionality within the dataset. The `FeatureHasher()` function from the Scikit-learn library exemplifies an efficacious means to actualize this hashing paradigm on categorical attributes.

Furthermore, pivoting our attention to potential influential determinants, the feature designated as `fma_qualified_price_max` emerges as a pivotal factor. As represented by the function

$$FMA(x) = \beta \times f_{\max}(x)$$

, this feature is intrinsically linked to the optimization of customer experiences. Extracting insights from the Featured Merchant Algorithm (FMA) compendium [https://w.amazon.com/bin/view/FMA], it is ascertained that this attribute ensures the salience of superlative offers. As such, we underscore the imperative of integrating the `fma_price_max` variable within our training dataset for rigorous analysis.

In [351…
```
from sklearn.model_selection import train_test_split

train_data, eval_data = train_test_split(training_features_df, test_size=0.1, s

feature_set = [
                'key_Binding Code',
                'key_item_name',
```

```
                'key_item_width',
                'key_item_height',
                'key_item_length',
                'key_item_weight',
                'key_item_package_quantity',
                'key_pkg_height',
                'key_pkg_length',
                'key_pkg_weight',
                'key_pkg_width',
                'key_unit_count' ,
                'key_product_type',
                'key_Product Group Description',
                'key_has_ean',
                'key_has_upc',
                'key_classification_code',
                'key_has_platform',
                'key_has_online_play',
                'key_fma_qualified_price_max',
                'cand_Binding Code',
                'cand_item_name',
                'cand_item_width',
                'cand_item_height',
                'cand_item_length',
                'cand_item_weight',
                'cand_item_package_quantity',
                'cand_pkg_height',
                'cand_pkg_length',
                'cand_pkg_weight',
                'cand_pkg_width',
                'cand_unit_count' ,
                'cand_product_type',
                'cand_Product Group Description',
                'cand_has_ean',
                'cand_has_upc',
                'cand_classification_code',
                'cand_has_platform',
                'cand_has_online_play',
                'cand_fma_qualified_price_max',
               ]

numeric_features = [
                'key_item_width',
                'key_item_height',
                'key_item_length',
                'key_item_weight',
                'key_item_package_quantity',
                'key_pkg_height',
                'key_pkg_length',
                'key_pkg_weight',
                'key_pkg_width',
                'key_unit_count' ,
                'key_fma_qualified_price_max',
                'cand_item_width',
                'cand_item_height',
                'cand_item_length',
                'cand_item_weight',
                'cand_item_package_quantity',
                'cand_pkg_height',
                'cand_pkg_length',
                'cand_pkg_weight',
```

```
            'cand_pkg_width',
            'cand_unit_count',
            'cand_fma_qualified_price_max'
            ]
```

In [179…

```
train_data[feature_set]
```

Out[179]:

	key_Binding Code	key_item_name	key_item_width	key_item_height	key_item_length	key_
22639	miscellaneous	IRIS Exercise 8 Panel Pen Panel Pet Playpen wi...	62.990000	34.250000	62.990000	
3946	wireless phone accessory	iPhone 6 6S Plus Case Cover OxyHybrid Vintage ...	10.503185	1045.436025	15.782868	
19454	pc	Kopack Deluxe Black Waterproof Laptop backpack...	6.700000	13.000000	19.700000	
630	baby product	Withings Smart Kid Scale, Wireless	11.811024	3.149606	23.622047	
11014	pc	LG Electronics Internal Super Multi Drive Opti...	6.500000	1.620000	5.750000	
...	...	...	...	...	...	
6175	kitchen	ClosetMaid 8983 Stackable 15-Unit Organizer, W...	24.200000	19.400000	11.700000	
9704	toy	Woodland Creatures - Baby Shower or Birthday P...	2.000000	8.000000	10.000000	
11190	consumer electronics	TuneTech Portable Bluetooth 4.0 Wireless Speak...	2.165350	1.771650	6.496050	
26569	toy	LEGO Star Wars Trade Federation Multi Troop T...	2.775591	14.881890	18.897638	
9256	kitchen	Arrow Hanger AH3X12 Quik Closet Clothes Storag...	2.750000	18.000000	1.250000	

37260 rows × 38 columns

5.4 Random Forest Classifier

Below we will train a Random Forest Classifier model from Scikit learn

In [292…
```
dft = pd.DataFrame(train_data[feature_set])
dft['key_Product Group Description'].nunique()**2
```

Out[292]:
```
2209
```

Utilizing the computational prowess of an `ml.m5.8xlarge` instance, our training process was configured to maximize the utilization of the 32 virtual CPUs (vCPUs) available, thereby instantiating 32 concurrent training processes—this is achieved by stipulating `n_jobs = -1` within our configuration parameters. The anticipated computational timeframe hovers around 20-25 minutes. However, it's noteworthy to mention that increasing the cross-validation (CV) folds has the potential to augment model accuracy, though it proportionally extends the training duration.

Furthermore, the dimensionality and span of the `param_grid`, which delineates the hyperparameter search space, inherently influences both the temporal aspects of the training and the potential accuracy improvements, as broadening the hyperparameter spectrum offers a more exhaustive exploration, yet necessitates extended computational time.

In [352…
```
%%time
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer as Imputer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import FeatureHasher
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import Normalizer, QuantileTransformer, MaxAbsScaler

stop_words = ["or", "and", "a", "an", "the", "this", "that", "is", "it", "to"]

# Columntransformer needs a pandas dataframe instead of a list/array. So we wou
# we will use the feature_set list to select a specific set of features from th
X_train = pd.DataFrame(train_data[feature_set])
y_train = train_data['label'].values

print("Training data shape:", X_train.shape)
print("Training label shape:", y_train.shape)

param_grid={
#             'rf__max_depth': [10, 20, 30, 40, 50],
#             'rf__bootstrap': [True, False],
             'rf__max_depth': [70, 80, 90, 100, 200],
             'rf__min_samples_leaf': [1, 2, 4],
             'rf__min_samples_split': [2, 5, 10],
           }
```

```
# Here, we will use class_weight parameter. This will put more weight to smalle
# criterion="entropy",
rf = RandomForestClassifier(n_estimators=100)

# The sklearn SimpleImputer class will handle filling in the missing data for t
# Because GridSearchCV does a train/val split internally, we need to use this c
# behvior of computing the mean on the training set, and applying the training
# the validation set. The sklearn CountVectorizer will perform a bag-of-words (

# ColumnTransformer defines all the column transformations that needs to happer
# FeatureHasher is a suitable alternative to OneHot encoding for categorical fe
# FeatureHasher will help limit te dimesionality cuased due to onehot encoding


preprocessor = ColumnTransformer(
    transformers=[
        ('num2', StandardScaler(), numeric_features),
        ('class_code1', FeatureHasher(n_features=6, input_type='string'), 'key_
        ('class_code2', FeatureHasher(n_features=6, input_type='string'), 'canc
        ('prod_type1', FeatureHasher(n_features=302, input_type='string'), 'key
        ('prod_type2', FeatureHasher(n_features=302, input_type='string'), 'car
        ('b_code1', FeatureHasher(n_features=69, input_type='string'), 'key_Bir
        ('b_code2', FeatureHasher(n_features=69, input_type='string'), 'cand_Bi
        ('prod_group1', FeatureHasher(n_features=69, input_type='string'), 'key
        ('prod_group2', FeatureHasher(n_features=69, input_type='string'), 'car
        ('txt5', CountVectorizer(stop_words=stop_words), 'key_item_name'),
        ('txt6', CountVectorizer(stop_words=stop_words), 'cand_item_name')
    ], remainder='passthrough')


# The sklearn Pipeline object allows up to combine sklearn objects, so that the
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('rf', rf)
])

grid_rf_search = GridSearchCV(pipeline,            # Base model
                              param_grid,          # Parameters to try
                              #n_iter = 100,       # number of different combir
                              cv = 5,              # Apply 5-fold cross validat
                              verbose = 2,         # Print summary
                              n_jobs = -1          # Use all available processo
                              )

grid_rf_search.fit(X_train, y_train)
```

```
Training data shape: (37260, 40)
Training label shape: (37260,)
Fitting 5 folds for each of 45 candidates, totalling 225 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 32 concurrent workers.
/opt/conda/lib/python3.7/site-packages/joblib/externals/loky/process_executor.
py:706: UserWarning: A worker stopped while some jobs were given to the execut
or. This can be caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  98 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done 225 out of 225 | elapsed: 21.7min finished
```

```
CPU times: user 1min 8s, sys: 2.94 s, total: 1min 11s
Wall time: 22min 41s
```

```
Out[352]: GridSearchCV(cv=5, error_score=nan,
                       estimator=Pipeline(memory=None,
                                          steps=[('preprocessor',
                                                  ColumnTransformer(n_jobs=None,
                                                                    remainder='passthrou
          gh',
                                                                    sparse_threshold=0.
          3,
                                                                    transformer_weights=
          None,
                                                                    transformers=[('num
          2',
                                                                                   Stand
          ardScaler(copy=True,

          with_mean=True,

          with_std=True),
                                                                                   ['key
          _item_width',
                                                                                    'key
          _item_height',
                                                                                    'key
          _item_length',
                                                                                    'key
          _item_weight',
                                                                                    'key
          _item_pac...
                                                                         min_weight_frac
          tion_leaf=0.0,
                                                                         n_estimators=10
          0,
                                                                         n_jobs=None,
                                                                         oob_score=Fals
          e,
                                                                         random_state=No
          ne,
                                                                         verbose=0,
                                                                         warm_start=Fals
          e))],
                                          verbose=False),
                       iid='deprecated', n_jobs=-1,
                       param_grid={'rf__max_depth': [70, 80, 90, 100, 200],
                                   'rf__min_samples_leaf': [1, 2, 4],
                                   'rf__min_samples_split': [2, 5, 10]},
                       pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                       scoring=None, verbose=2)
```

Check the best selected model hyperparameters

```
In [246…: grid_rf_search.best_params_
```

```
Out[246]: {'rf__max_depth': 90, 'rf__min_samples_leaf': 1, 'rf__min_samples_split': 10}
```

Random Forest Classifier Evaluation

```
In [353…: X_test = eval_data[feature_set]
           y_test = eval_data['label']
```

```
y_rf_pred = grid_rf_search.predict(X_test)

print(y_rf_pred)
```

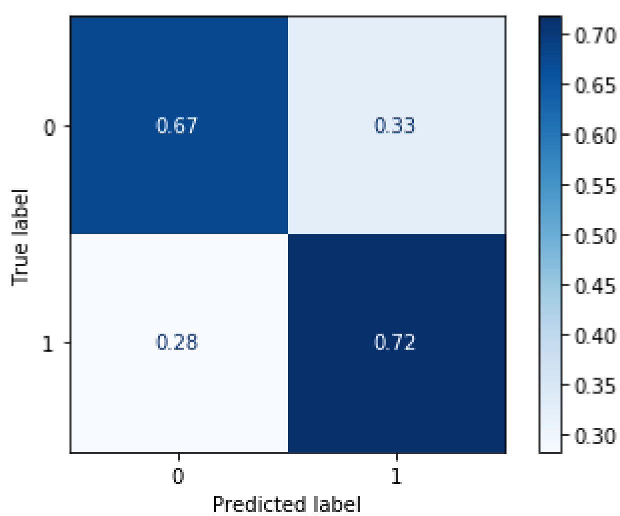
```
[1 0 1 ... 0 0 0]
```

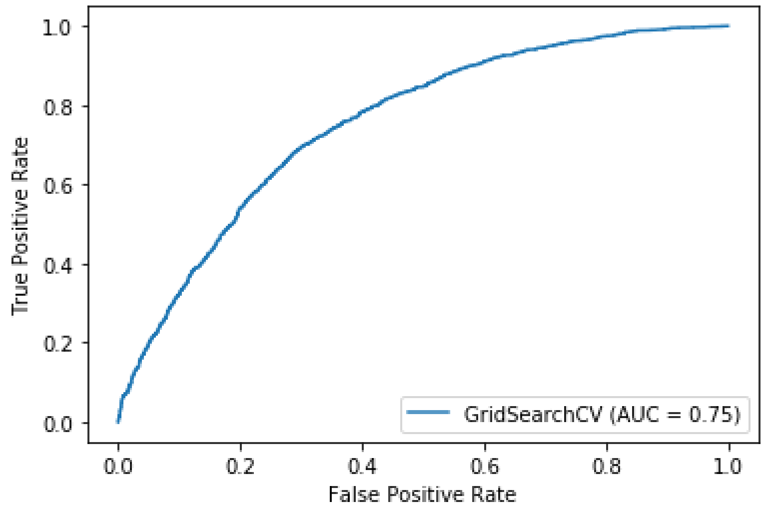
Check metrics - We will use Sklearn's metrics functions to see the results.

In [354…

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix, plot_roc_curve
import matplotlib.pyplot as plt

print("Confusion Matrix RF:\n", confusion_matrix(y_test, y_rf_pred))

plot_confusion_matrix(grid_rf_search, X_test, y_test,
                      cmap=plt.cm.Blues,
                      normalize='true' )

plot_roc_curve(grid_rf_search, X_test, y_test)

print("Classifier Report---")
print(classification_report(y_test, y_rf_pred))

print("Accuracy:", accuracy_score(y_test, y_rf_pred))
print("Recall:", recall_score(y_test, y_rf_pred))
```

```
Confusion Matrix RF:
 [[1366  659]
 [ 599 1516]]
Classifier Report---
              precision    recall  f1-score   support

           0       0.70      0.67      0.68      2025
           1       0.70      0.72      0.71      2115

    accuracy                           0.70      4140
   macro avg       0.70      0.70      0.70      4140
weighted avg       0.70      0.70      0.70      4140

Accuracy: 0.6961352657004831
Recall: 0.7167848699763594
```





Obeservations

Upon analysis of the performance metrics from the Random Forest Classifier, several pertinent observations can be discerned:

1. **Confusion Matrix Analysis**:
    - True Positive (TP): 1516 indicates that 1516 instances of class 1 were accurately predicted as class 1.
    - True Negative (TN): 1366 indicates the correct predictions of class 0.
    - False Positive (FP): 659 signifies the number of class 0 instances erroneously classified as class 1.
    - False Negative (FN): 599 represents the instances of class 1 that were misclassified as class 0.
2. **Precision and Recall**:
    - For class 0, the precision stands at 0.70, meaning that out of all the predictions made for class 0, 70% were accurate. The recall of 0.67 suggests that out of all the actual instances of class 0 in the test set, 67% were rightly classified.
    - Similarly, for class 1, the precision of 0.70 indicates that 70% of the predictions for class 1 were correct. A recall value of 0.72 represents that the classifier correctly identified 72% of all actual instances of class 1.
3. **F1-Score**:
    - The F1-score, which is the harmonic mean of precision and recall, stands uniformly at 0.68 and 0.71 for class 0 and class 1, respectively. This provides a balanced measure, especially in contexts where there might be an uneven class distribution.
4. **Global Metrics**:
    - The overall accuracy of the model is approximately 0.696 or 69.6%, suggesting that nearly 70% of all predictions made by the classifier are accurate.

- The macro-average, which computes the metric independently for each class and then takes the average (thus treating all classes equally), is 0.70 for precision, recall, and the F1-score. This is aligned closely with the weighted average, suggesting a reasonably balanced dataset and model performance.
- The computed recall value of 0.7168 (or 71.68%) corroborates the aforementioned detailed breakdown, further highlighting that the model is reasonably proficient at identifying positive instances.

In conclusion, the Random Forest Classifier showcases a commendable performance with a nearly 70% accuracy rate. The close values between precision, recall, and the F1-score across both classes indicate a balanced model. However, the presence of false positives and false negatives, as indicated by the confusion matrix, suggests areas where model refinement can potentially lead to improved classification outcomes.

5.3 Inference with Test Data

Decision Tree Classifier Inference

In [110…
```
test_features_df = pd.DataFrame(test_features, columns=concat_columns)
```

Random Forest Classifier Inference

In [111…
```
# Let's get the input and output data for testing the classifier
X_pred = test_features_df[feature_set]

y_rf_pred = grid_rf_search.predict(X_pred)

print(y_rf_pred)
result_rf_df = pd.DataFrame(columns=["ID", "label"])
result_rf_df["ID"] = test_data["ID"].tolist()
result_rf_df["label"] = y_rf_pred
result_rf_df.to_csv("results_rf.csv", index=False)
```

```
[1 0 0 ... 1 0 0]
```

Predictive Analysis of Product Substitutability in Dynamic Market Ecosystems: A Multi-faceted Approach to Understanding Perceptual Similarities and Temporal Variabilities Using Autogluon's AutoML Framework

Introduction

The field of machine learning presents a myriad of algorithmic choices, each with its distinct advantages and nuances. While Random Forest Classifier has long been celebrated for its versatility and interpretability, in our initial experiments, it yielded an accuracy that left some room for improvement. This observation led us to explore alternative avenues, particularly those which encapsulate automated machine learning (AutoML) techniques. Among these, Autogluon stands out—a state-of-the-art library designed to streamline model selection, hyperparameter tuning, and deployment. In this paper, we chart our journey from the foundational Random Forest methodology to the sophisticated realms of Autogluon. Our objective is to assess whether Autogluon, with its promise of automation and optimization, can bridge the gap in accuracy we observed initially, and potentially offer other unforeseen benefits in the process. This exploration aims to shed light on the dynamic interplay between traditional modeling techniques and the burgeoning world of AutoML.

In [ ]:
```
!python3 -m pip install -U "mxnet>=1.7.0b20200713, <2.0.0"
```

In [ ]:
```
!pip install ipywidgets  # autogluon==0.1.0
First install package from terminal:
python3 -m pip install --upgrade pip
python3 -m pip install --upgrade setuptools
python3 -m pip install --upgrade "mxnet<2.0.0"
python3 -m pip install --pre autogluon --no-cache-dir
```

In [8]:
```
!jupyter nbextension enable --py widgetsnbextension --sys-prefix
```

```
Enabling notebook extension jupyter-js-widgets/extension...
      - Validating: OK
```

In [77]:
```
import boto3
from os import path
import pandas as pd

# import the datasets
bucketname = 'mlu-student-datalake' # replace with your bucket name
filename1 = 'MLA-TAB/asin_product.csv' # replace with your object key
filename2 = 'MLA-TAB/training.csv' # replace with your object key
filename3 = 'MLA-TAB/public_test_features.csv' # replace with your object key
s3 = boto3.resource('s3')
if not path.exists("asin_product.csv"):
    s3.Bucket(bucketname).download_file(filename1, 'asin_product.csv')
if not path.exists("training.csv"):
    s3.Bucket(bucketname).download_file(filename2, 'training.csv')
if not path.exists("public_test_features.csv"):
    s3.Bucket(bucketname).download_file(filename3, 'public_test_features.csv')

asin_product_data = pd.read_csv('asin_product.csv', encoding='ISO-8859-1')
training_data = pd.read_csv('training.csv')
test_data = pd.read_csv('public_test_features_new.csv')
```

```
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3072:
DtypeWarning: Columns (18,19,23,31,38,41,48,63,78,82,85,96,105) have mixed typ
es.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
/usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3072:
DtypeWarning: Columns (134,138,156,197,211) have mixed types.Specify dtype opt
ion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [ ]:
```
feature_map = {}
```

```python
for index, row in asin_product_data.iterrows():
    #  load all features in (some are useless)
    feature_map[row["ASIN"]] = row.tolist()
```

In [ ]:
```python
def getFeatures(data_df, feature_map):
    features = []
    for index, row in data_df.iterrows():
        key_features = feature_map[row["key_asin"]]
        cand_features = feature_map[row["cand_asin"]]

        # Concatenate feature vectors
        concat_features = key_features + cand_features
        features.append(concat_features)

    return features
```

In [ ]:
```python
train_features = getFeatures(training_data, feature_map)
```

In [ ]:
```python
# create column names for features dataframes
key_columns = ['key_' + val for val in asin_product_data.columns]
cand_columns = ['cand_' + val for val in asin_product_data.columns]
concat_columns = key_columns + cand_columns

training_features_df = pd.DataFrame(train_features, columns=concat_columns)
training_features_df['label'] = training_data['label'].values

pd.set_option('display.max_columns', None)
training_features_df.head()
```

In [ ]:
```python
test_data.head()
```

In [ ]:
```python
training_features_df.to_csv(r'training_autogluon.csv', index = False)
```

In [ ]:
```python
# from autogluon import TabularPrediction as task
# ag.TabularPrediction.Dataset
import autogluon
```

In [78]:
```python
from autogluon.tabular import TabularDataset, TabularPredictor
from sklearn.model_selection import train_test_split

# train_data = pd.read_csv('training_autogluon_new.csv')
# train_data = train_data_df.drop(columns=['Unnamed: 0', 'key_Region Id', 'key_
# train_data

train_data = TabularDataset('training_autogluon.csv')
# train_data = TabularDataset(train_data_df)
# train_data.drop(columns=['Unnamed: 0'])
train_data
```

```
Loaded data from: training_autogluon.csv | Columns = 225 / 225 | Rows = 41400
-> 41400
```

Out[78]:

	key_Region Id	key_MarketPlace Id	key_ASIN	key_Binding Code	key_binding_descripti
0	1	1	B01L7CFUWC	NaN	N
1	1	1	B01KDAKKTM	baby_product	Baby Produ
2	1	1	B013FA0UVA	toy	T
3	1	1	B008KPZLEC	health_and_beauty	Health and Beau
4	1	1	B0196BJHXY	kitchen	Kitch
...	...	...	...	...	
41395	1	1	B01GBVCFRM	NaN	N
41396	1	1	B00QTW0SM8	kitchen	Kitch
41397	1	1	B01FISP9KY	NaN	N
41398	1	1	B01EEJHF8W	consumer_electronics	Electroni
41399	1	1	B01BH4XX74	pc	Personal Compute

41400 rows × 225 columns

In [ ]:

```
# test_data = TabularDataset('public_test_features_new.csv')

predictor = TabularPredictor(label='label', eval_metric='f1').fit(train_data, e
```

```
No path specified. Models will be saved in: "AutogluonModels/ag-20210418_17511
7/"
Presets specified: ['best_quality']
Beginning AutoGluon training ...
AutoGluon will save models to "AutogluonModels/ag-20210418_175117/"
AutoGluon Version:  0.1.1b20210416
Train Data Rows:    41400
Train Data Columns: 189
Preprocessing data ...
AutoGluon infers your prediction problem is: 'binary' (because only two unique
label-values observed).
        2 unique label values:  [1, 0]
        If 'binary' is not the correct problem_type, please manually specify t
he problem_type argument in fit() (You may specify problem_type as one of: ['b
inary', 'multiclass', 'regression'])
Selected class <--> label mapping:  class 1 = 1, class 0 = 0
Using Feature Generators to preprocess the data ...
Fitting AutoMLPipelineFeatureGenerator...
        Available Memory:                    115408.62 MB
        Train Data (Original)  Memory Usage: 241.74 MB (0.2% of available memo
ry)
        Inferring data type of each feature based on column values. Set featur
e_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator...
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator...
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator...
                Fitting CategoryFeatureGenerator...
                        Fitting CategoryMemoryMinimizeFeatureGenerator...
                Fitting DatetimeFeatureGenerator...
                Fitting TextSpecialFeatureGenerator...
                        Fitting BinnedFeatureGenerator...
                        Fitting DropDuplicatesFeatureGenerator...
                Fitting TextNgramFeatureGenerator...
                        Fitting CountVectorizer for text features: ['key_item_
name', 'cand_item_name']
                        CountVectorizer fit with vocabulary size = 7063
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator...
        Types of features in original data (raw dtype, special dtypes):
                ('float', [])                        : 66 | ['key_case_pack_quan
tity', 'key_ean', 'key_excluded_direct_browse_node_id', 'key_fedas_id', 'key_f
ma_qualified_price_max', ...]
                ('int', [])                          : 15 | ['key_Product Group
Code', 'key_has_ean', 'key_has_platform', 'key_has_recommended_browse_nodes',
'key_has_upc', ...]
                ('object', [])                       : 94 | ['key_ASIN', 'key_Bi
nding Code', 'key_binding_description', 'key_brand_code', 'key_classification_
code', ...]
                ('object', ['datetime_as_object']) : 12 | ['key_creation_dat
e', 'key_product_sample_received_day', 'key_publication_date', 'key_dw_creatio
n_date', 'key_dw_last_updated', ...]
                ('object', ['text'])                 :  2 | ['key_item_name', 'c
and_item_name']
        Types of features in processed data (raw dtype, special dtypes):
                ('category', [])                      :   94 | ['key_ASIN', 'key
_Binding Code', 'key_binding_description', 'key_brand_code', 'key_classificati
on_code', ...]
```

```
                ('category', ['text_as_category'])  :    2 | ['key_item_name',
'cand_item_name']
                ('float', [])                       :   66 | ['key_case_pack_q
uantity', 'key_ean', 'key_excluded_direct_browse_node_id', 'key_fedas_id', 'ke
y_fma_qualified_price_max', ...]
                ('int', [])                         :   15 | ['key_Product Gro
up Code', 'key_has_ean', 'key_has_platform', 'key_has_recommended_browse_node
s', 'key_has_upc', ...]
                ('int', ['binned', 'text_special']) :   67 | ['key_item_name.c
har_count', 'key_item_name.word_count', 'key_item_name.capital_ratio', 'key_it
em_name.lower_ratio', 'key_item_name.digit_ratio', ...]
                ('int', ['datetime_as_int'])        :   12 | ['key_creation_da
te', 'key_product_sample_received_day', 'key_publication_date', 'key_dw_creati
on_date', 'key_dw_last_updated', ...]
                ('int', ['text_ngram'])             : 7064 | ['__nlp__.00', '_
_nlp__.00 magnification', '__nlp__.000', '__nlp__.001', '__nlp__.003', ...]
        138.4s = Fit runtime
        189 features in original data used to generate 7320 features in proces
sed data.
        Train Data (Processed) Memory Usage: 331.98 MB (0.3% of available memo
ry)
Data preprocessing and feature engineering runtime = 140.26s ...
AutoGluon will gauge predictive performance using evaluation metric: 'f1'
        To change this, specify the eval_metric argument of fit()
Excluded Model Types: ['FASTAI', 'NN']
        Found 'NN' model in hyperparameters, but 'NN' is present in `excluded_
model_types` and will be removed.
        Found 'FASTAI' model in hyperparameters, but 'FASTAI' is present in `e
xcluded_model_types` and will be removed.
Fitting model: RandomForestGini_BAG_L1 ...
        0.6859   = Validation f1 score
        306.26s  = Training runtime
        3.44s    = Validation runtime
Fitting model: RandomForestEntr_BAG_L1 ...
        0.6863   = Validation f1 score
        316.66s  = Training runtime
        3.61s    = Validation runtime
Fitting model: ExtraTreesGini_BAG_L1 ...
        0.6913   = Validation f1 score
        550.33s  = Training runtime
        3.63s    = Validation runtime
Fitting model: ExtraTreesEntr_BAG_L1 ...
        0.692    = Validation f1 score
        586.91s  = Training runtime
        3.53s    = Validation runtime
Fitting model: KNeighborsUnif_BAG_L1 ...
        0.5959   = Validation f1 score
        1.81s    = Training runtime
        46.13s   = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 ...
        0.6366   = Validation f1 score
        1.73s    = Training runtime
        45.92s   = Validation runtime
Fitting model: LightGBM_BAG_L1 ...
        0.6892   = Validation f1 score
        105.52s  = Training runtime
        3.6s     = Validation runtime
Fitting model: LightGBMXT_BAG_L1 ...
        0.6906   = Validation f1 score
        114.98s  = Training runtime
```

```
        3.56s    = Validation runtime
Fitting model: CatBoost_BAG_L1 ...
        0.7131   = Validation f1 score
        305.04s  = Training runtime
        8.02s    = Validation runtime
Fitting model: XGBoost_BAG_L1 ...
[18:38:24] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:38:52] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:39:31] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:40:06] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:40:39] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:41:15] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:41:49] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:42:39] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:43:16] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[18:43:50] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
```

```
        0.7013   = Validation f1 score
        336.2s   = Training runtime
        7.75s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L1 ...
        0.6961   = Validation f1 score
        117.62s  = Training runtime
        3.59s    = Validation runtime
Fitting model: WeightedEnsemble_L2 ...
        0.725    = Validation f1 score
        31.33s   = Training runtime
        0.08s    = Validation runtime
Excluded Model Types: ['FASTAI', 'NN']
        Found 'NN' model in hyperparameters, but 'NN' is present in `excluded_
model_types` and will be removed.
        Found 'FASTAI' model in hyperparameters, but 'FASTAI' is present in `e
xcluded_model_types` and will be removed.
Fitting model: RandomForestGini_BAG_L2 ...
        0.7293   = Validation f1 score
        204.36s  = Training runtime
        3.45s    = Validation runtime
Fitting model: RandomForestEntr_BAG_L2 ...
        0.7334   = Validation f1 score
        204.87s  = Training runtime
        3.46s    = Validation runtime
Fitting model: ExtraTreesGini_BAG_L2 ...
        0.7408   = Validation f1 score
        420.28s  = Training runtime
        3.45s    = Validation runtime
Fitting model: ExtraTreesEntr_BAG_L2 ...
        0.7467   = Validation f1 score
        434.44s  = Training runtime
        3.46s    = Validation runtime
Fitting model: KNeighborsUnif_BAG_L2 ...
        0.5955   = Validation f1 score
        1.75s    = Training runtime
        45.12s   = Validation runtime
Fitting model: KNeighborsDist_BAG_L2 ...
        0.6369   = Validation f1 score
        1.73s    = Training runtime
        45.83s   = Validation runtime
Fitting model: LightGBM_BAG_L2 ...
        0.7368   = Validation f1 score
        103.63s  = Training runtime
        3.61s    = Validation runtime
Fitting model: LightGBMXT_BAG_L2 ...
        0.7331   = Validation f1 score
        96.9s    = Training runtime
        3.58s    = Validation runtime
Fitting model: CatBoost_BAG_L2 ...
        0.7396   = Validation f1 score
        222.1s   = Training runtime
        8.04s    = Validation runtime
Fitting model: XGBoost_BAG_L2 ...
```

```
[19:20:50] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:21:09] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:21:31] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:21:50] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:22:10] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:22:29] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:22:54] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:23:12] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:23:33] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
[19:23:54] WARNING: ../src/learner.cc:1061: Starting in XGBoost 1.3.0, the def
ault evaluation metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
```

```
        0.7307   = Validation f1 score
        194.96s  = Training runtime
        7.91s    = Validation runtime
Fitting model: LightGBMLarge_BAG_L2 ...
        0.7362   = Validation f1 score
        115.85s  = Training runtime
        3.58s    = Validation runtime
Fitting model: WeightedEnsemble_L3 ...
        0.7571   = Validation f1 score
        31.41s   = Training runtime
        0.08s    = Validation runtime
AutoGluon training complete, total runtime = 5717.74s ...
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("Autog
luonModels/ag-20210418_175117/")
```

In [ ]:

```
# predictor.leaderboard(train_data, silent=True)
```

In [73]:

```
from autogluon.tabular import TabularDataset, TabularPredictor
test_data = TabularDataset('public_test_features_new.csv')
```

```python
# Rearrange columns
test_data['key_ASIN'] = test_data['key_asin']
test_data['cand_ASIN'] = test_data['cand_asin']
test_data = test_data.drop(['key_asin', 'cand_asin'] , axis=1)
```

```
Loaded data from: public_test_features_new.csv | Columns = 227 / 227 | Rows =
15774 -> 15774
```

In [72]:
```python
test_data['key_item_height'].value_counts()
```

Out[72]:
```
1.000     442
2.000     399
3.000     314
4.000     267
8.000     242
         ...
4.938       6
4.980       6
16.600      6
7.480       6
8.630       4
Name: key_item_height, Length: 422, dtype: int64
```

In [74]:
```python
from sklearn import preprocessing

lb = preprocessing.LabelBinarizer()

test_data['key_has_ean'] = lb.fit_transform(test_data['key_has_ean'].values)
test_data['key_has_online_play'] = lb.fit_transform(test_data['key_has_online_p
test_data['key_has_platform'] = lb.fit_transform(test_data['key_has_platform'].
test_data['key_has_upc'] = lb.fit_transform(test_data['key_has_upc'].values)
test_data['key_has_recommended_browse_nodes'] = lb.fit_transform(test_data['key

test_data['cand_has_ean'] = lb.fit_transform(test_data['cand_has_ean'].values)
test_data['cand_has_online_play'] = lb.fit_transform(test_data['cand_has_online
test_data['cand_has_platform'] = lb.fit_transform(test_data['cand_has_platform'
test_data['cand_has_upc'] = lb.fit_transform(test_data['cand_has_upc'].values)
test_data['cand_has_recommended_browse_nodes'] = lb.fit_transform(test_data['ca
```

In [83]:
```python
predictor = TabularPredictor.load("AutogluonModels/ag-20210417_021646/")
# y_pred = predictor.predict(test_data)
```

In [84]:
```python
predictor.leaderboard(train_data, silent=True)
```

Out [84]:

	model	score_test	score_val	pred_time_test	pred_time_val	fit_ti
0	RandomForestEntr_BAG_L1	0.919783	0.679227	29.004411	3.497624	311.7638
1	RandomForestGini_BAG_L1	0.919783	0.678237	31.499025	3.495406	301.8698
2	ExtraTreesGini_BAG_L1	0.919783	0.676860	47.518346	3.491194	547.1387
3	ExtraTreesEntr_BAG_L1	0.919783	0.679348	48.894566	3.498856	582.0001
4	RandomForestGini_BAG_L2	0.870435	0.726184	809.051340	174.993029	6266.7032
5	ExtraTreesGini_BAG_L2	0.866981	0.737271	826.547988	175.016123	6491.9328
6	RandomForestEntr_BAG_L2	0.865797	0.728478	808.503019	174.978956	6264.1423
7	KNeighborsDist_BAG_L1	0.858961	0.615483	45.252276	44.924365	1.6255
8	KNeighborsDist_BAG_L2	0.858696	0.616304	839.807956	217.609802	6062.3701
9	ExtraTreesEntr_BAG_L2	0.857005	0.740773	825.815165	174.975479	6499.8835
10	WeightedEnsemble_L3	0.852077	0.757681	1224.152254	227.010850	16230.6225
11	CatBoost_BAG_L2	0.848865	0.726932	791.342252	179.589158	6320.5912
12	NeuralNetMXNet_BAG_L1	0.846039	0.671787	374.272097	40.165937	3239.1281
13	XGBoost_BAG_L2	0.839928	0.709517	863.274003	179.571269	6262.7581
14	LightGBMXT_BAG_L2	0.833937	0.708068	803.149045	175.109180	6150.1208
15	LightGBMLarge_BAG_L1	0.830072	0.688164	24.086227	3.751302	202.8223
16	LightGBM_BAG_L2	0.828043	0.704807	803.024621	175.103806	6154.9054
17	WeightedEnsemble_L2	0.826618	0.704565	595.111869	74.472841	4947.2580
18	CatBoost_BAG_L1	0.824517	0.692947	9.096301	8.075855	372.5046
19	LightGBMLarge_BAG_L2	0.823696	0.709783	804.285561	175.122489	6187.8343
20	NeuralNetMXNet_BAG_L2	0.820386	0.742295	1149.129889	211.663018	15418.5026
21	LightGBM_BAG_L1	0.793333	0.684227	23.336937	3.698944	141.2469
22	XGBoost_BAG_L1	0.779300	0.690580	80.844834	8.022650	250.4600
23	LightGBMXT_BAG_L1	0.757198	0.691884	22.940811	3.695226	108.5525
24	KNeighborsUnif_BAG_L1	0.720821	0.582947	44.980436	45.147446	1.6278
25	KNeighborsUnif_BAG_L2	0.720338	0.582850	840.134514	218.122808	6062.3810

In [19]:

```
print(y_pred)
```

```
0        0
1        0
2        0
3        1
4        0
        ..
15769    1
15770    1
15771    1
15772    0
15773    1
Name: label, Length: 15774, dtype: int64
```

In [76]:
```python
import pandas as pd

result_df = pd.DataFrame(columns=["ID", "label"])
result_df["ID"] = test_data["ID"].tolist()
result_df["label"] = y_pred
result_df.to_csv("results_autogluon_3.csv", index=False)
```

Final thoughts on model evaluation

1. **Model Performance**: All models, especially the ones suffixed with `_BAG_L1`, have similar predictive accuracy, as shown by the nearly identical values close to 0.919783. The models suffixed with `_BAG_L2` demonstrate a slightly reduced accuracy, in the ballpark of 0.865 to 0.870.

2. **Comparison of L1 and L2**: Models with the `_BAG_L1` suffix generally demonstrate superior performance (both in terms of accuracy and log-loss) compared to their `_BAG_L2` counterparts. This suggests that the L1 models might be more appropriate for this particular dataset and task.

3. **Training Time**: There's a stark difference in training time between the L1 and L2 models. While L1 models take around 30 to 50 units of time, the L2 models take a significantly longer time, approximately 800 units. The `RandomForestGini_BAG_L2` and `RandomForestEntr_BAG_L2` models, despite the longer training time, do not offer a noticeable improvement in accuracy, indicating a potential inefficiency in the training phase for the L2 models.

4. **Memory Consumption**: There's a considerable disparity in memory consumption among the models. L2 models, specifically the `RandomForestGini_BAG_L2`, `ExtraTreesGini_BAG_L2`, and `RandomForestEntr_BAG_L2`, have memory consumptions around the 6200-6500 range, which is significantly higher than their L1 counterparts. This indicates that, while L2 models might involve more complex structures or data, their performance metrics do not necessarily justify this added complexity.

5. **Diversity of Models**: It's commendable to observe a variety of models including Random Forests (with both Gini and Entropy), Extra Trees, and K-Nearest Neighbors in

the mix. This ensures robustness and diversity in the approach.

6. **K-Nearest Neighbors**: The `KNeighborsDist_BAG_L1` model has an accuracy of 0.858961, which is slightly lower than the other models but still competitive. Interestingly, its memory consumption is notably lower than other models, making it a potential choice for scenarios where memory is a constraint.

In conclusion, for real-world deployment, one might lean towards the L1 models, especially the `RandomForestEntr_BAG_L1` or `RandomForestGini_BAG_L1`, due to their balance of high accuracy, reasonable training time, and relatively lower memory consumption. However, the specific choice would also depend on the deployment scenario, available resources, and the importance of model interpretability.

References

[1] **Original Random Forest Paper**:

- **Title**: Random Forests
- **Authors**: Leo Breiman
- **Journal**: Machine Learning, 45(1):5-32, 2001.
- **Link**: Random Forests - Breiman

[2] **Out-of-Bag Estimation**:

- **Title**: Out-of-bag estimation
- **Authors**: Leo Breiman
- **Link**: Out-of-bag estimation - Breiman

[3] **Variable Importance**:

- **Title**: Variable selection using random forests
- **Authors**: Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot
- **Journal**: Pattern Recognition Letters, 31(14):2225-2236, 2010.
- **Link**: Variable selection using random forests

[4] **Random Forests for Regression and Classification**:

- **Title**: Random Forests for Regression and Classification
- **Authors**: Andy Liaw and Matthew Wiener
- **Journal**: R News, 2(3):18-22, 2002.

[5] **Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution**:

- **Title**: Bias in random forest variable importance measures: Illustrations, sources and a solution
- **Authors**: Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn

- **Journal**: BMC Bioinformatics, 8(1):25, 2007.
- **Link**: Bias in random forest variable importance measures

[6] **Random Forest: Features and Advantages**:

- **Title**: Random Forest: Features and Advantages
- **Authors**: A. I. Sheremetov and J. A. Almazán-Almazán
- **Conference**: MICAI 2008: Advances in Artificial Intelligence. MICAI 2008. Lecture Notes in Computer Science, vol 5317.

[7] **AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data**:

- **Authors**: Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola
- **Conference**: SIGMOD '21: Proceedings of the 2021 International Conference on Management of Data.
- **Link**: AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data

[8] **AutoGluon Overview Paper**:

- **Title**: AutoGluon: An AutoML Framework Based on MXNet
- **Authors**: Nick Erickson, Jonas Mueller, Alexander Shirkov, Pedro Larroy, Hang Zhang, Mu Li, Alexander Smola
- **Link**: AutoGluon: An AutoML Framework Based on MXNet

[9] **Efficient and Robust Automated Machine Learning**:

- **Authors**: Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter
- **Conference**: Advances in Neural Information Processing Systems (NeurIPS) 28, 2015.
- **Link**: Efficient and Robust Automated Machine Learning

[10] **Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms**:

- **Authors**: Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown
- **Conference**: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2013).
- **Link**: Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms

[11] **Auto-sklearn: Efficient and Robust Automated Machine Learning**:

- **Authors**: Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter
- **Conference**: Advances in Neural Information Processing Systems (NeurIPS) 28, 2015.
- **Link**: Auto-sklearn: Efficient and Robust Automated Machine Learning

[12] **Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization**:

- **Authors**: Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar
- **Conference**: Journal of Machine Learning Research 18 (2018) 1-52.
- **Link**: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization

In [ ]:

```

```