National College of Ireland

Configuration Manual

MSc Data Analytics
Customer Behaviour prediction

Nikhil Kulkarni
Student ID:x18201130

School of Computing
National College of Ireland

Supervisor: Manaz Kaleel

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nikhil Kulkarni ……
Student ID:	……x18201130…… …….……
Programme:	Msc Data Analytics …… …… **Year:2019-2020** ……..
Module:	Research Project … ……..……
Lecturer:	Manaz Kaleel ……..……
Submission Due Date:	28/09/2020 ……..……
Project Title:	Customer behaviour prediction ……..……
Word Count:	…… **Page Count:** ……..……..……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Nikhil Kulkarni ……
Date:	28/09/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nikhil Kulkarni
Student ID: x18201130

1 Pre-requisites

To implement this project google collab id is necessary, windows system with 16gb RAM and i5 processor with GPU. This project is developed in python 3.6 so a relevant version is needed.

2 Access Collab and Files Import

```
from google.colab import drive
drive.mount('/content/drive')
```

3 Import Libraries

```
import pandas as pd
import tensorflow as tf
import numpy as np
import keras
from keras.layers import Input, Dense, Embedding, Conv2D, MaxPool2D
from keras.layers import Reshape, Flatten, Dropout, Concatenate
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adam
from keras.models import Model
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
import datetime
import time
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import scipy.sparse as sp
from scipy.sparse import vstack
from scipy import sparse
from scipy.sparse.linalg import spsolve
from subprocess import check_output
from sklearn import metrics
```

4. Data Pre-processing

```
[4]  # Importing the dataset
     from google.colab import drive
     drive.mount('/content/drive')
     events_df = pd.read_csv('/content/drive/My Drive/project_data/events.csv')
     category_tree_df = pd.read_csv('/content/drive/My Drive/project_data/category_tree.csv')
     item_properties_1_df = pd.read_csv('/content/drive/My Drive/project_data/item_properties_part1.csv')
     item_properties_2_df = pd.read_csv('/content/drive/My Drive/project_data/item_properties_part2.csv')

     Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8q

     Enter your authorization code:
     ··········
     Mounted at /content/drive
```

Figure 2 Mount dataset on collab.

```
user_activity_count = dict()
for row in events.itertuples():
    if row.visitorid not in user_activity_count:
        user_activity_count[row.visitorid] = {'view':0 , 'addtocart':0, 'transaction':0};
    if row.event == 'addtocart':
        user_activity_count[row.visitorid]['addtocart'] += 1
    elif row.event == 'transaction':
        user_activity_count[row.visitorid]['transaction'] += 1
    elif row.event == 'view':
        user_activity_count[row.visitorid]['view'] += 1

d = pd.DataFrame(user_activity_count)
dataframe = d.transpose()
# Activity range
dataframe['activity'] = dataframe['view'] + dataframe['addtocart'] + dataframe['transaction']
# removing users with only a single view
cleaned_data = dataframe[dataframe['activity']!=1]
# all users contains the userids with more than 1 activity in the events (4lac)
all_users = set(cleaned_data.index.values)
all_items = set(events['itemid'])
# todo: we need to clear items which are only viewed once

visitorid_to_index_mapping  = {}
itemid_to_index_mapping  = {}
vid = 0
iid = 0
for row in events.itertuples():
    if row.visitorid in all_users and row.visitorid not in visitorid_to_index_mapping:
        visitorid_to_index_mapping[row.visitorid] = vid
        vid = vid + 1

    if row.itemid in all_items and row.itemid not in itemid_to_index_mapping:
        itemid_to_index_mapping[row.itemid] = iid
        iid = iid + 1
```

Figure 3 Data Pre-processing

```
events_df[events_df.visitorid == 102019].sort_values('timestamp')
```

	timestamp	visitorid	event	itemid	transactionid
19690	1433175714335	102019	view	49521	NaN
19501	1433175801314	102019	addtocart	49521	NaN
14842	1433175812596	102019	view	150318	NaN
19573	1433175871497	102019	view	49521	NaN
8701	1433175894837	102019	view	49521	NaN
19708	1433175945872	102019	view	150318	NaN
8740	1433176042269	102019	view	49521	NaN
814	1433176736375	102019	transaction	150318	13556.0
19724	1433176736422	102019	transaction	49521	13556.0

Figure 4 – Dataset

4 Logistic Regression-

```
[37] X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42, train_size = 0.7)

[38] logreg = LogisticRegression()

[39] logreg.fit(X_train, y_train)

     LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                        warm_start=False)

[40] #use the model to predict the test features
     y_pred_class = logreg.predict(X_test)

[41] print('accuracy = {:7.3f}'.format(metrics.accuracy_score(y_test, y_pred_class)))

     accuracy =   0.794
```

Figure 5 Logistic Regression model

5 LightFM Model-

```
    test_set = ratings.copy() # Make a copy of the original set to be the test set.
    test_set[test_set != 0] = 1 # Store the test set as a binary preference matrix
    training_set = ratings.copy() # Make a copy of the original data we can alter as our training set.
    nonzero_inds = training_set.nonzero() # Find the indices in the ratings data where an interaction exists
    nonzero_pairs = list(zip(nonzero_inds[0], nonzero_inds[1])) # Zip these pairs together of user,item index into list
    random.seed(0) # Set the random seed to zero for reproducibility
    num_samples = int(np.ceil(pct_test*len(nonzero_pairs))) # Round the number of samples needed to the nearest integer
    samples = random.sample(nonzero_pairs, num_samples) # Sample a random number of user-item pairs without replacement
    user_inds = [index[0] for index in samples] # Get the user row indices
    item_inds = [index[1] for index in samples] # Get the item column indices
    training_set[user_inds, item_inds] = 0 # Assign all of the randomly chosen user-item pairs to zero
    training_set.eliminate_zeros() # Get rid of zeros in sparse array storage after update to save space
    return training_set, test_set, list(set(user_inds)) # Output the unique list of user rows that were altered
```

+ Code + Markdown

```
X_train, X_test, item_users_altered = make_train(user_to_item_matrix, pct_test = 0.1)
```

```
no_comp, lr, ep = 100, 0.01, 10
model = LightFM(no_components=no_comp, learning_rate=lr, loss='warp')
model.fit_partial(
        X_train,
        item_features=item_to_property_matrix_sparse,
        epochs=ep,
        num_threads=4,
        verbose=True)
model.summary(accuracy)
```

Figure 6 LightFM model

6 Multilayer PerceptronModel

```
# Fitting classifier to the Training set
# Create your classifier here
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout

classifier = Sequential()

classifier.add(Dense(activation="relu", input_dim=11, units=1024, kernel_initializer="uniform"))
#classifier.add(Dropout(0.2))
classifier.add(Dense(activation="relu", units=512, kernel_initializer="uniform"))


classifier.add(Dense(activation="sigmoid", units=2, kernel_initializer="uniform"))



classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
classifier.summary()
classifier.fit(X_train,y_train,epochs=10,batch_size=32)
```

Figure7 Multilayer perceptron model