National College of Ireland

Improving the Click Prediction for Online Advertisement with the Integration of Recommended System using Neural Network Architecture

MSc Research Project
Data Analytics

Jayalakshmi Jayachandran
X20213557

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

1. Introduction

In this case, the configuration manual serves as a concise summary of the device specification that was employed along with a thorough description of the programming language that was used to implement the concept. Additionally, it provides an explanation of the libraries and packages used in the creation of our topic:

Improving the Click Prediction for Online Advertisement with the Integration of Recommended System using Neural Network Architecture

This manual procedure will be showing how the data has been uploaded, cleaned, and pre-processed, and then how it is implemented on the suitable models.

2. System Configuration

In this section the system configuration which has required for the implementation of the model.

2.1 Hardware specification

For the implementation of the whole idea of the project, system configuration which is required in the respective processes is given in the figure 1:

System	RAM 32 GB
Processor	Intel I5
Speed	1.90 GHz
Software	Jupyter Notebook
Programming Language	Python 3
Python libraries	Python libraries

2.2 Software Specification

A few programming tools with various packages are utilized to put the theory into practice. Python has been utilized throughout the framework for coding, and Jupyter Notebook is the platform used to carry out the concept.

2.3 Python

The current version which has been used in the idea is 3.6.9 for the development of algorithm structure.

Libraries

1. Pandas- For handling structure data.
2. Numpy- For linear algebra and mathematics.
3. Keras- for development and evaluating deep learning models.
4. Tensorflow- is used for fast numerical computing.
5. Scikit Learn- For machine learning
6. Pmdarima- For ARIMA model.
7. Seaborn- For data visualization.
8. Matpotlib- For data plotting and visualization.
9. Compare Models- For metrics plotting followed by their comparison

3. Data Sources

3.1 Dataset

The advertisement click data has been accumulated from the Kaggle website i.e., the Avazu click dataset (Click-Through Rate Prediction | Kaggle, 2022). This data contains four million rows and twenty-five columns. These columns contain information such as time stamp, banner position, site category, app category, click, etc. The data for the recommendation system is also collected from the Kaggle which is an Amazon product review dataset (Recommender System Using Amazon Reviews, 2022). This data contains seven million rows and four columns. These columns are accountable for product id, user id, timestamp, and ratings. All values in this data are in numerical format. The size of click datasets is about 6 GB and the Size of Amazon review dataset is 300 MB.

Dataset links:
Amazon review Dataset: Recommender System Using Amazon Reviews | Kaggle
Click-Through Rate Prediction: Click-Through Rate Prediction | Kaggle

---

https://pandas.pydata.org/
https://numpy.org/
https://keras.io/
https://www.tensorflow.org/
https://scikit-learn.org/stable/
https://pypi.org/project/pmdarima/
https://seaborn.pydata.org/
https://matplotlib.org/
https://pycaret.org/compare-models/

4. Project Implementation

After the selection of dataset, the data set has been imported into the python environment on the jupyter notebook.

```
In [8]:  1 data.head()      # getting first five rows from the dataset
```

Out[8]:

	Unnamed: 0	id	click	hour	C1	banner_pos	site_id	site_domain	site_category	app_id	app_domain	app_category	devi
0	0	10004765361151096125	1	14102100	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8d9	07d7df22	a99
1	1	10157267783354901009	0	14102100	1005	1	43d6df75	27e3c518	28905ebd	ecad2386	7801e8d9	07d7df22	a99
2	2	10204774106542702658	1	14102100	1005	1	77bfdd7b	bfa24f16	3e814130	ecad2386	7801e8d9	07d7df22	a99
3	3	10351076848552612232	0	14102100	1005	1	5b4d2eda	16a36ef3	f028772b	ecad2386	7801e8d9	07d7df22	a99
4	4	10408083230449973220	0	14102100	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8d9	07d7df22	a99

Fig 1

```
In [25]:  1 data2.head()  # getting first five rows of data
```

Out[25]:

	userId	productId	Rating	timestamp
0	AKM1MP6P0OYPR	0132793040	5.0	1365811200
1	A2CX7LUOHB2NDG	0321732944	5.0	1341100800
2	A2NWSAGRHCP8N5	0439886341	1.0	1367193600
3	A2WNBOD3WNDNKT	0439886341	3.0	1374451200
4	A1GI0U4ZRJA8WN	0439886341	1.0	1334707200

Fig2

Above fig1 shows the head data of Avazu click data and fig2 shows head data recommendation system.

5. Data Pre-Processing

After accessing the data from the authenticate source next step performed is the pre-processing of the data. Since both datasets contain a large number of rows therefore random samples of 100 thousand rows are taken from both datasets. In this step processes like data cleaning, removal of null values, changing of data types of columns, and data normalization are executed. Click data contains categorical columns, which are converted into the required numerical format by deploying a label encoder.

```
In [9]:  1 data.shape    # checking number of rows and columns of data
Out[9]: (100000, 25)

In [10]:  1 data.isnull().sum()    # checking data for null values
Out[10]: Unnamed: 0          0
         id                  0
         click               0
         hour                0
         C1                  0
         banner_pos          0
         site_id             0
         site_domain         0
         site_category       0
         app_id              0
         app_domain          0
         app_category        0
         device_id           0
         device_ip           0
         device_model        0
         device_type         0
         device_conn_type    0
         C14                 0
         C15                 0
```

Fig3

```
In [28]:  1 data2.isnull().sum()   # checking for null values in dataset
Out[28]: userId       0
         productId    0
         Rating       0
         timestamp    0
         dtype: int64
```

Fig4

Now, in fig3 and fig4 we can see that all the NA values has been dropped. Also, NAN values has been eliminated, that's why we have added data cleaning step for the implementation of the processes.

I. Feature Extraction

The stages of feature extraction and feature selection are crucial for this research since they help to overcome the problem data dimensionality and shorten training times.

```
In [43]:
data=data.drop(['date','id','site_id','app_id','device_id','device_ip','site_domain','app_domain','device_conn_type'],ax
new_data=data
data.head()
```

Out[43]:

	click	C1	banner_pos	site_category	app_category	device_model	device_type	C14	C15	C16	C17	C18	C19	C20	C21	hour_of_day	pro
0	1	1005	0	28905ebd	07d7df22	7fdd04d2	1	15701	320	50	1722	0	35	-1	79	0	B00(
1	0	1005	1	28905ebd	07d7df22	3bd9e8e7	1	15701	320	50	1722	0	35	-1	79	0	B00(
2	1	1005	1	3e814130	07d7df22	900981af	1	20596	320	50	2161	0	35	100034	157	0	B000
3	0	1005	1	f028772b	07d7df22	d787e91b	1	19950	320	50	1800	3	167	100075	23	0	B000B
4	0	1005	0	28905ebd	07d7df22	c6263d8a	1	15701	320	50	1722	0	35	-1	79	0	B00(

Fig5

Above image fig5, shows unnecessary columns such as date, and id are dropped first and new columns are added to the existing data which contains cumulative values of other columns. Data is scaled using a standard scalar for the standardization of data. This step results in a reduction in the size of the feature vector from 26 to 19. Click feature is allocated as the target variable while the remaining variables are assigned as feature variables.

II. Modelling and Evaluation

In this step we have applied three deep learning models which are capable in click prediction based on a recommendation system using the advertisement and product review dataset.

i. LSTM model

LSTM Model- Hyperparameter tuning using keras tuner

```
[ ]:
def build_model(hp):
    model = Sequential()
    model.add(LSTM(hp.Int('input_unit',min_value=32,max_value=512,step=32),return_sequences=True, input_shape=(x_train_.
    for i in range(hp.Int('n_layers', 1, 4)):
        model.add(LSTM(hp.Int(f'lstm_{i}_units',min_value=32,max_value=512,step=32),return_sequences=True))
    model.add(LSTM(hp.Int('layer_2_neurons',min_value=32,max_value=512,step=32)))
    model.add(Dropout(hp.Float('Dropout_rate',min_value=0,max_value=0.5,step=0.1)))
    model.add(Dense(hp.Int('layer_3_neurons',min_value=32,max_value=512,step=32), activation=hp.Choice('dense_activation
    model.add(Dropout(hp.Float('Dropout_rate',min_value=0,max_value=0.5,step=0.1)))
    model.add(Dense(1, activation=hp.Choice('dense_activation',values=['sigmoid'])))
    model.compile(loss='binary_crossentropy', optimizer = "adam", metrics=['accuracy'])
    return model
```

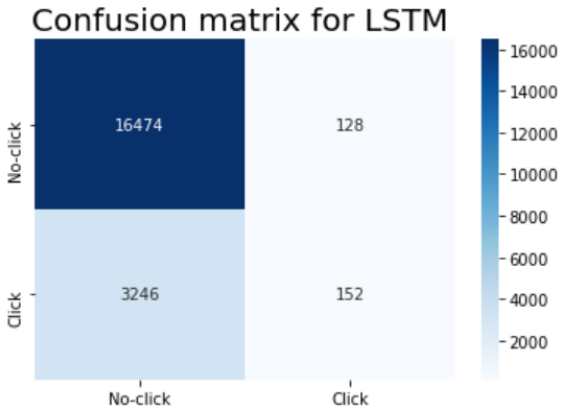
```
[ ]:
tuner= RandomSearch(
        build_model,
        objective='val_accuracy',
        max_trials=2,
        executions_per_trial=1
        )
```

Fig6

After implementation of the LSTM model, we got the below confusion matrix.

In [73]:

```
# plotting confusion mtrix
cm = confusion_matrix(y_test, y_pred1)
class_label = ["No-click", "Click"]
df_cm = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df_cm, annot = True, fmt = "d", cmap='Blues')
plt.title('Confusion matrix for LSTM ', fontsize = 20); # title with fontsize 20
```



ii. CNN model

CNN can also be implemented on text data by using different embedding layers provided by Keras.

CNN Model

```
model2=Sequential()
# Conv Block 1
model2.add(Conv1D(250, kernel_size=3,input_shape=(x_train_.shape[1],1),activation='linear'))   # adding conv layer with
model2.add(MaxPooling1D(pool_size=3,padding = 'same'))       # adding max pooling layer with same padding
model2.add(BatchNormalization()) # adding batch normalization layer

# Conv Block 2
model2.add(Conv1D(80, kernel_size=2,padding='same',activation='linear'))
model2.add(MaxPooling1D(pool_size=2,padding = 'same'))
model2.add(BatchNormalization())

# Flattening
model2.add(Flatten())   # adding flatten layer

# Dense Layer 1
model2.add(Dense(10,activation='linear'))
model2.add(Dropout(0.2))

#Output Layer
model2.add(Dense(1,activation='sigmoid'))     # adding dense output layer
# Compiling Model
model2.compile(loss='binary_crossentropy', optimizer = "adam", metrics=['accuracy'])

# Training Model
history = model2.fit(x_train_,y_train,epochs=10, validation_data=(x_test_,y_test), batch_size=64, shuffle=True)
model2.summary()
```

Fig7

Similarly the fig7 is representing the CNN convolution neural network model and their evaluation parameter.

iii. RNN

```python
# Initialising the RNN ( with two layers)
model3 = Sequential()
function = "relu"

# Adding the first RNN layer and some Dropout regularization
model3.add(SimpleRNN(units = 100, activation=function, return_sequences=True,input_shape = (x_train_.shape[1], 1)))
model3.add(Dropout(0.2))

# Adding a second RNN layer and some Dropout regularization
model3.add(SimpleRNN(units = 80))
model3.add(Dropout(0.2))
model3.add(Dense(units = 1,activation='sigmoid'))     # adding dense output layer withn 1 neuron

# Compiling the RNN
model3.compile(loss='binary_crossentropy', optimizer = "adam", metrics=['accuracy'])

# Fitting the RNN to the Training set
history=model3.fit(x_train_, y_train, epochs = 10, batch_size = 64, validation_data=(x_test_,y_test),shuffle=True)
model3.summary()
```

```
Epoch 1/10
1250/1250 [==============================] - 24s 18ms/step - loss: 0.4423 - accuracy: 0.8294 - val_loss: 0.4309 - val_acc
uracy: 0.8296
Epoch 2/10
1250/1250 [==============================] - 21s 17ms/step - loss: 0.4336 - accuracy: 0.8307 - val_loss: 0.4309 - val_acc
uracy: 0.8301
Epoch 3/10
1250/1250 [==============================] - 21s 17ms/step - loss: 0.4307 - accuracy: 0.8305 - val_loss: 0.4314 - val_acc
```

Fig 8

Where in fig 8 the RNN model is representing and the evaluation parameters which will shows the performance of the model.

III. Performance comparison of machine learning models.

The model which achieves the highest value of MCC score, accuracy, f1-score, precision and recall will be selected as the finest model for heading prediction basis. Bar plots are plotted to visualize the comparison among the implemented models.

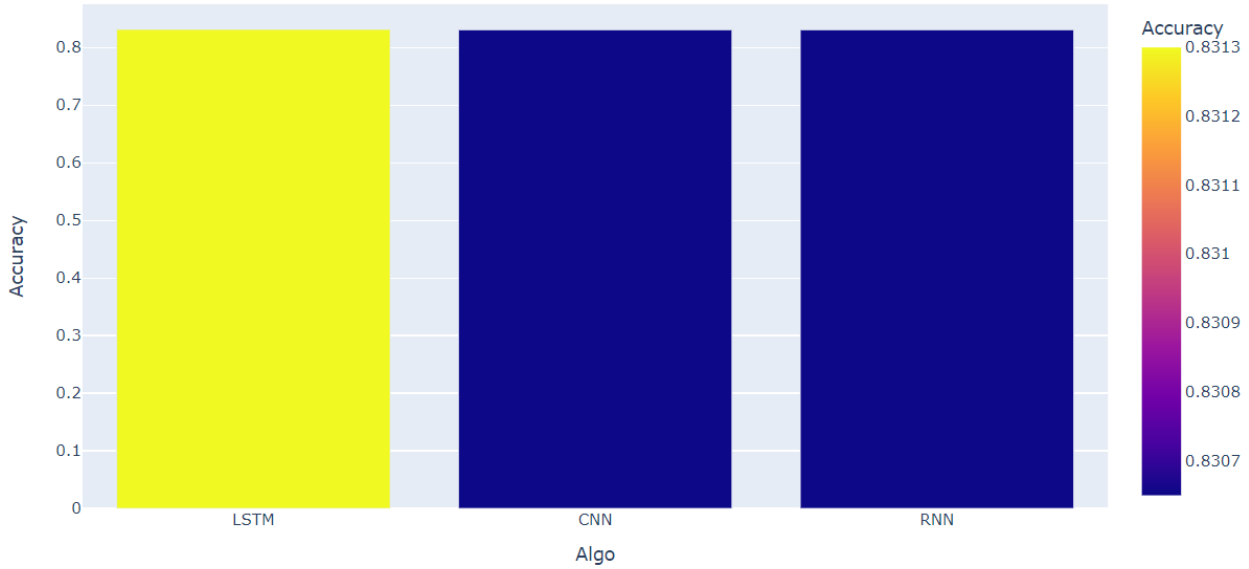


Fig 9

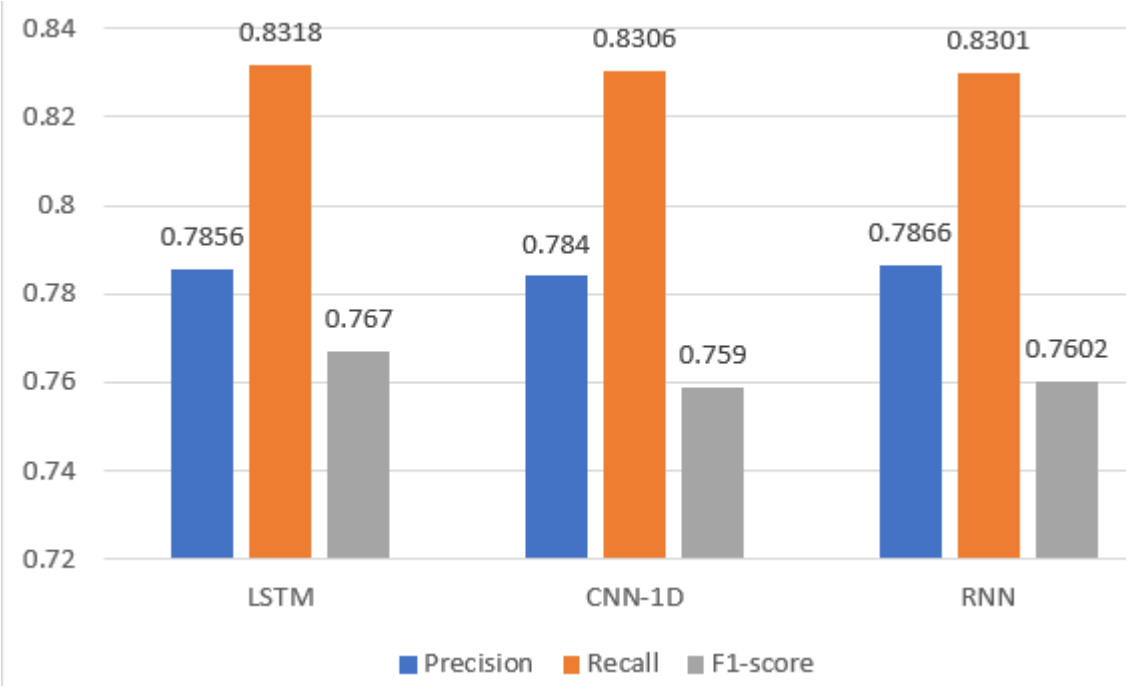


Fig 10

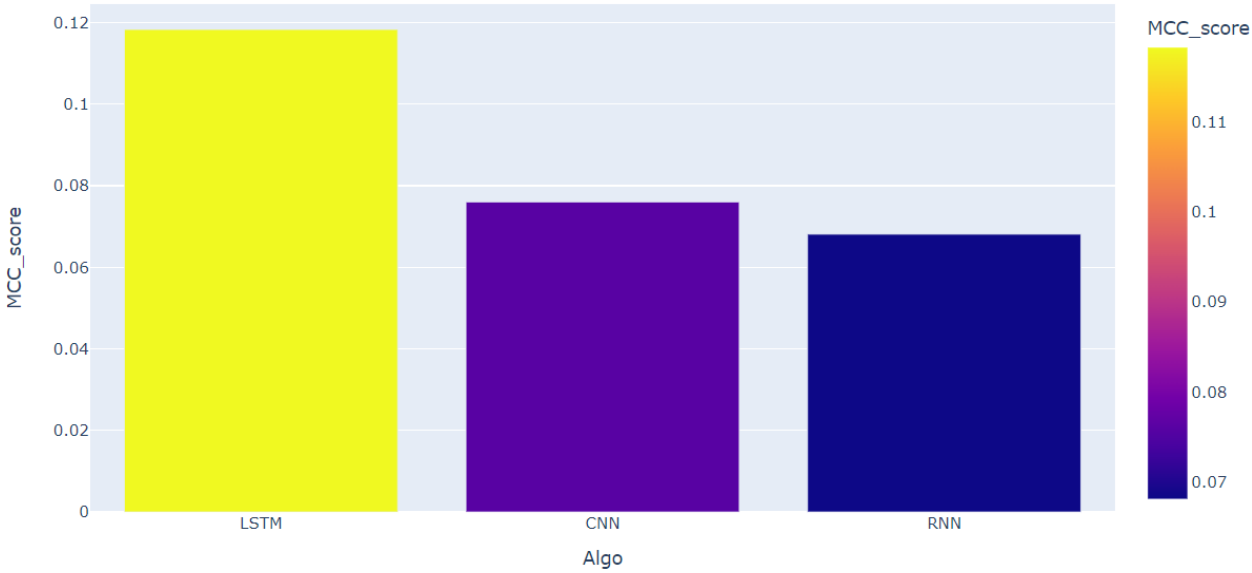


Fig11

Now, in the fig 9, 10 and 11 we can compare and see that which machine learning models has better performance in terms of predicting the clicks on the online advertisements.