title: "Starbuck_Project"
date: 2023-07-18

Section 1: Project Definition

Project Overview

This data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks.

Not all users receive the same offer, and that is the challenge to solve with this data set.

Your task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, you can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

You'll be given transactional data showing user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase. This transactional data also has a record for each offer that a user receives as well as a record for when a user actually views the offer. There are also records for when a user completes an offer.

Keep in mind as well that someone using the app might make a purchase through the app without having received an offer or seen an offer.

Problem Statement

Not all users receive the same offer, and that is the challenge to solve with this data set.

Predicting the offer to which a possible higher level of response or user actions like ‘offer received’, ‘offer viewed’, ‘transaction’ and ‘offer completed’ can be achieved based on attributes of the customer and the companies.

Metrics

We use ACCURACY as classification metric. Is an easily suited for binary as well a multiclass classification problem.

ACCURACY is the the result of: (TRUE_POSITIVE+TRUE_NEGATIVE) / (total cases)

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Section 2: Analysis

Data Exploration

The data is contained in three files:

* portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
* profile.json - demographic data for each customer
* transcript.json - records for transactions, offers received, offers viewed, and offers completed

Here is the schema and explanation of each variable in the files:

* portfolio.json

id (string) - offer id
offer_type (string) - type of offer ie BOGO, discount, informational
difficulty (int) - minimum required spend to complete an offer
reward (int) - reward given for completing an offer
duration (int) - time for offer to be open, in days
channels (list of strings)

	channels	difficulty	duration	id	offer_type	reward
0	[email, mobile, social]	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10

* profile.json

age (int) - age of the customer
became_member_on (int) - date when customer created an app account
gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
id (str) - customer id
income (float) - customer's income

	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN

* transcript.json

event (str) - record description (ie transaction, offer received, offer viewed, etc.)
person (str) - customer id
time (int) - time in hours since start of test. The data begins at time t=0
value - (dict of strings) - either an offer id or transaction amount depending on the record

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}

We need to clean this data to have a better visualization. So the dataframe resultant is:

Portfolio: Steps to clean it:

- copy of the original df
- rename 'id' to 'offer_id'
- convert the 'duration' to hours
- assign more readable offer ids
- explode 'channels' and remove 'channels'

	difficulty	duration	offer_id	offer_type	reward	web	email	mobile	social
0	10	168	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10	0	1	1	1
1	10	120	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10	1	1	1	1
2	0	96	3f207df678b143eea3cee63160fa8bed	informational	0	1	1	1	0
3	5	168	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5	1	1	1	0
4	20	240	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5	1	1	0	0

Profile: Steps to clean it:

- copy of the original df
- rename 'id' to 'person_id'
- Remove null values. In oor case, customers with age 118
- Assign each person the corresponding pseudonym
- replace 'became_member_on' with 'member_since' and set the format
- To have an easy manage, we assig per range

	age	gender	person_id	income	member_since	age_range	income_range
1	55	F	0610b486422d4921ae7d2bf64640c50b	112000.0	2017	50-59	110-119K

Transcript: Steps to clean it:

- copy of the original df
- rename 'id' to 'person_id'
- replace 'offer id' to 'offer_id' in the 'value' column
- convert 'value' column to actual dict
- split the 'value' column into colums ('offer_id','reward' and 'amount')

- fill the NAN as a NONE
- replace NONE to 0

	event	person_id	time	amount	offer_id	reward
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	0.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.0

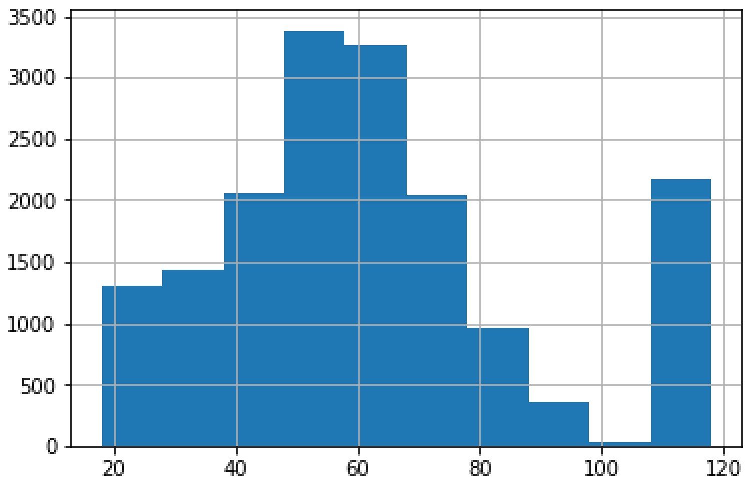
Once we have cleaned, we join them into one dataframe:

event	person_id	time	amount	offer_id	reward	difficulty	duration	offer_type	web	email	mobile	social	age	gender	income	member_since	age_range	income_range
offer received	78afa995795e4d85b5d9ceeca43f5fef	0	0.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.0	5.0	168.0	bogo	1.0	1.0	1.0	0.0	75	F	100000.0	2017	70-79	100-109K
offer viewed	78afa995795e4d85b5d9ceeca43f5fef	6	0.0	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.0	5.0	168.0	bogo	1.0	1.0	1.0	0.0	75	F	100000.0	2017	70-79	100-109K

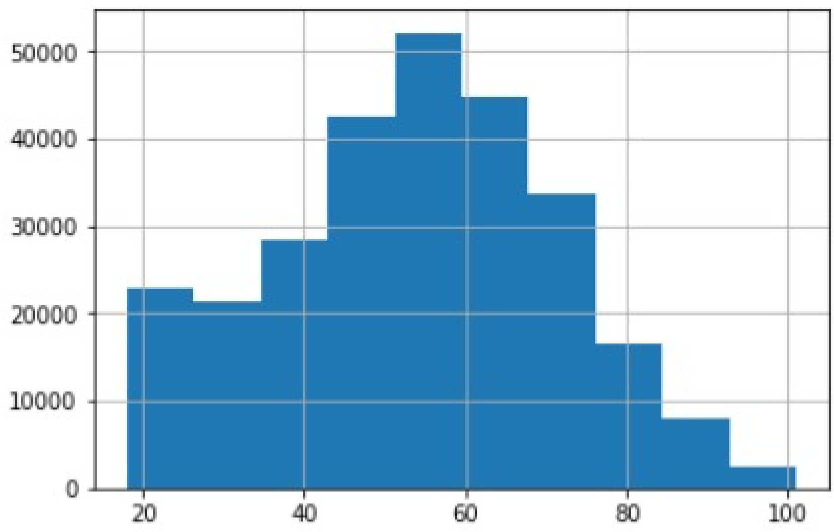
Data Visualization

Age group

	age
gender	
F	57.544950
M	52.116690
O	54.400943

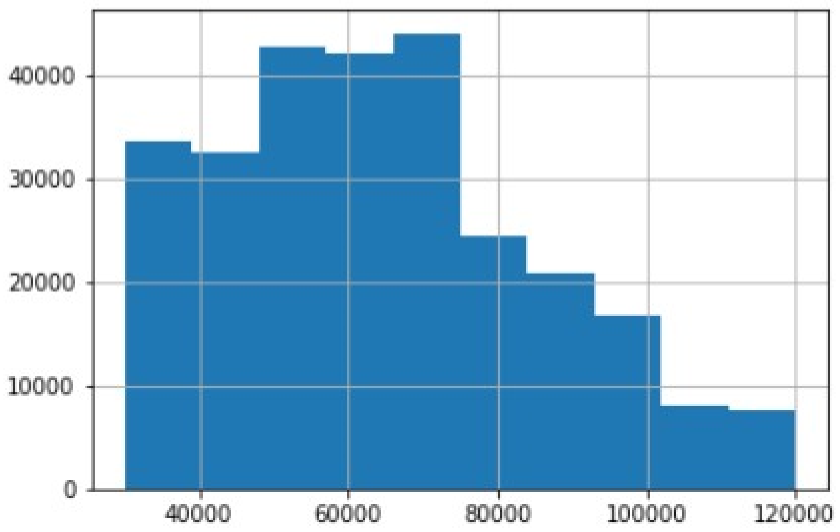


The age of more than 118 does not make sense. We remove i:



The average age of the users is: 50-60 years

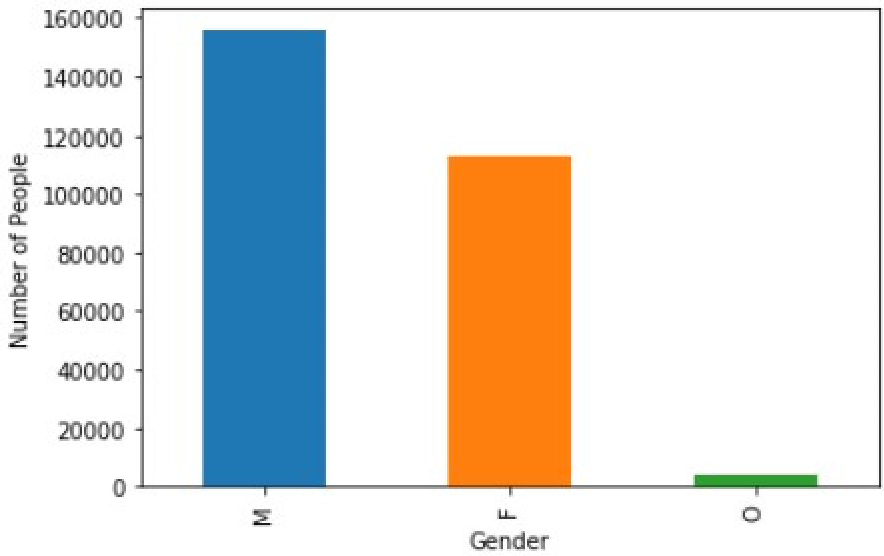
Income:



The average of the incomes is: 60000-70000

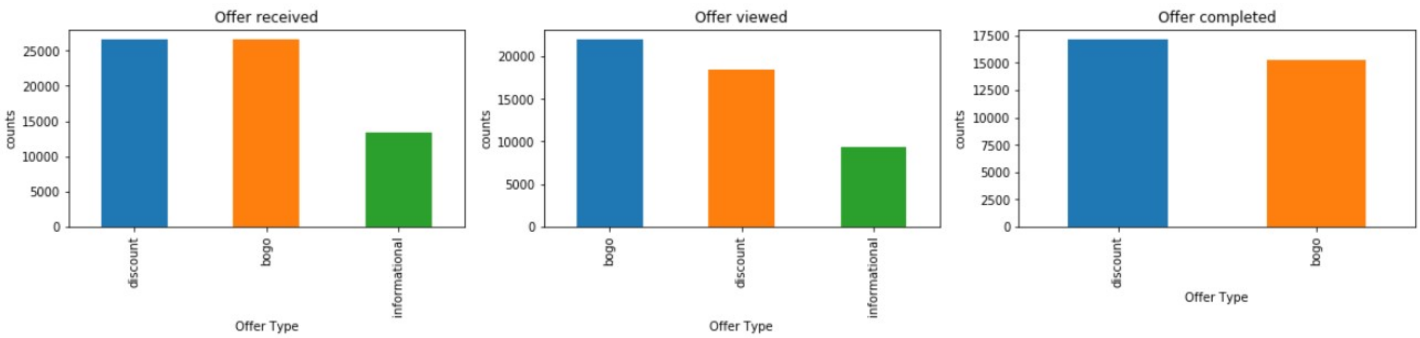
	income
gender	
F	71306.412139
M	61194.601603
O	63287.735849

Number of people per genre:

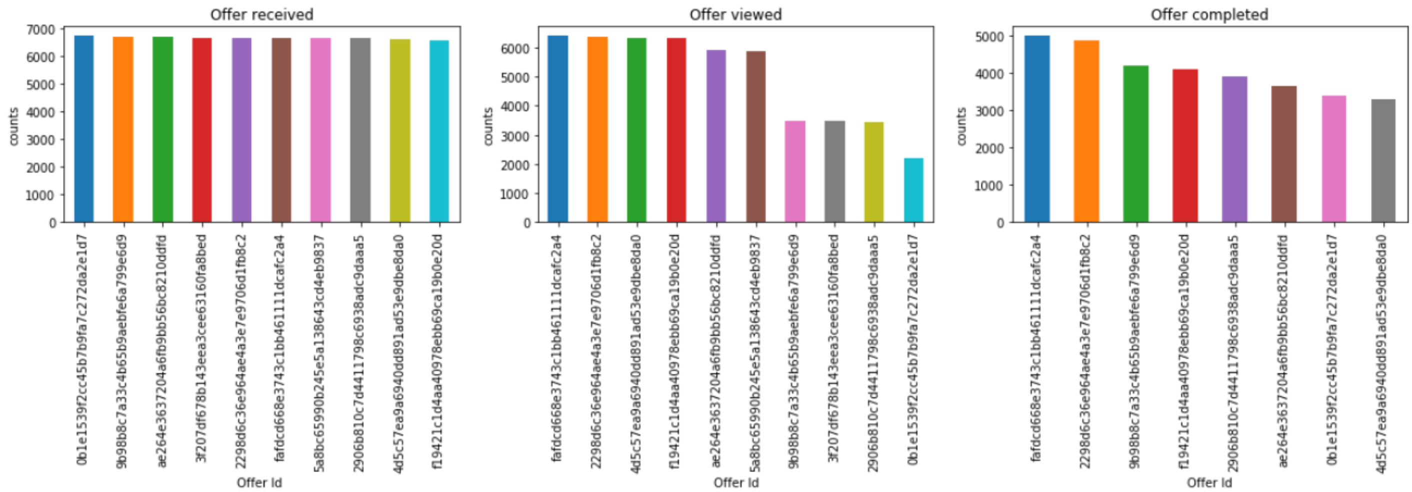



```
57.0790652657 % of Male
41.465086779 % of Female
1.45584795536 % of Others or Unknow
```

Offer type:



```
offer received
discount         26664
bogo             26537
informational    13300
Name: offer_type, dtype: int64
offer viewed
bogo             22039
discount         18461
informational     9360
Name: offer_type, dtype: int64
offer completed
discount     17186
bogo         15258
```



We can see that

- **offer received:** each Offer_id have the same number of offer received
- **offer viewed:** the ratio decreased for some offer_ids
- **Offer completed:** the ratio decreased for some offer_ids

BOGO

```
offer_received =  26537
offer_viewed =  22039
offer_completed =  17186
--------------------
offer_viewed / offer_received =  83.050081019 %
offer_completed / offer_received =  57.4970795493 %
```

DISCOUNT

```
offer_received =  26664
offer_viewed =  18461
offer_completed =  17186
--------------------
offer_viewed / offer_received =  69.2356735674 %
offer_completed / offer_received =  64.4539453945 %
```

To sum up:

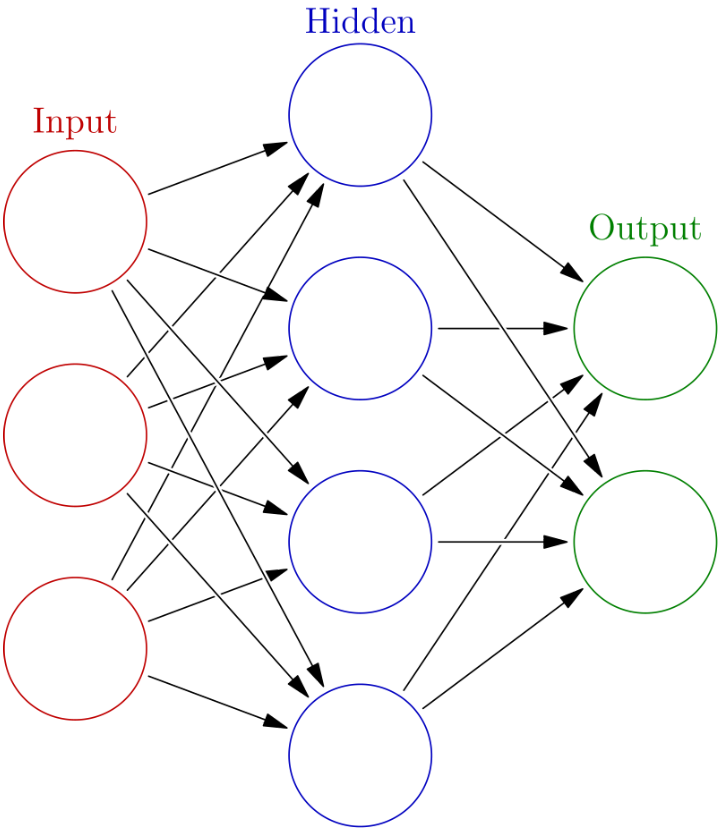
- Percentage of BOGO Offer viewer is 83%
- Percentage of DISCOUNT Offer viewer is 70%

Algorithms and Techniques

Artificial neural networks (ANNs, also shortened to neural networks (NNs) or neural nets) are a branch of machine learning models that are built using principles of neuronal organization discovered by connectionism in the biological neural networks constituting animal brains.[1][2]

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives signals then processes them and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold.

Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.



Inputs
Source data fed into the neural network, tith the goal of making a prediction

Training set
A set of inputs for which the correct outputs are known, used to train the neural network

Outputs
Neural networks generate their prediction in the form of a set of real values or boolean decision. Each output is generated by one of the neurons in the output layer.

Neuron/perceptron

ANNs are composed of artificial neurons which are conceptually derived from biological neurons. Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons.The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons. The outputs of the final output neurons of the neural net accomplish the task, such as recognizing an object in an image.

To find the output of the neuron we take the weighted sum of all the inputs, weighted by the weights of the connections from the inputs to the neuron. We add a bias term to this sum. This weighted sum is sometimes called the activation. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image.

Hyperparameters
A hyperparameter is a constant parameter whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyperparameters include learning rate, the number of hidden layers and batch size.The values of some hyperparameters can be dependent on those of other hyperparameters. For example, the size of some layers can depend on the overall number of layers.

Section 3: Methodology

Data Preprocessing

1. We use **One-hot-encoding.** In this case we encoding the gender and offer_type column

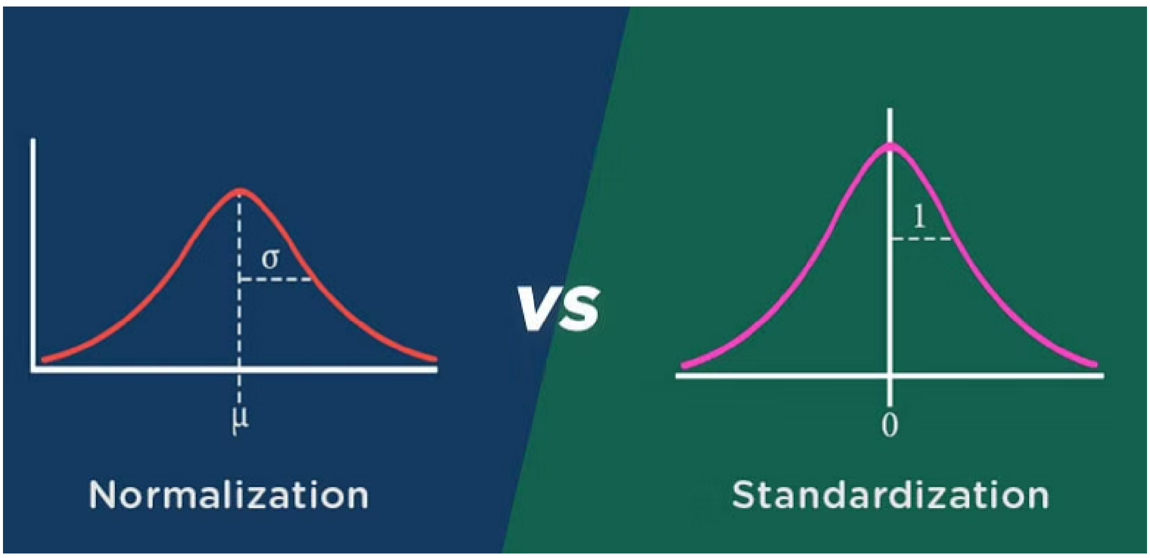
2. We split the dataset using **train_test_split** function, that we import:
```
from sklearn.model_selection import train_test_split
```
   We split training and testing.
   We use the training for build the model and we use the testing to evaluate the performance of the model.

3. Scaling: **normalization and standardization**:
   a. Standardization: Scaling technique, where the values are centered around the mean with a unit standard deviation.

$$z = \frac{x_i - \mu}{\sigma}$$

   b. Normalization: Scaling technique, where the values are shifted and rescaled.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$



4. We need to convert the pandas dataframe to np array
   Pandas dataframe is 2-D size mutable, potentially heterogeneous tabular data structure with labeled axes (col and raws). We can covert this data structure to Numpy with Dataframe.to_numpy() method.

Implementation

To build the model we use Sequential().

Layer 1 is the input layer
Layer 2 is the hidden layer

‘relu’ function is highly computationally efficient but is not able to process inputs that approach zero or negative.

Layer 3 is the output Layer
‘Softmax’ is a special activation function used for output neurons. It normalizes outputs for each class between 0 and 1, and returns the probability that the input belongs to a specific class.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 32)                256
_________________________________________________________________
dense_8 (Dense)              (None, 15)                495
_________________________________________________________________
dense_9 (Dense)              (None, 10)                160
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 66
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 28
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 30
_________________________________________________________________
dense_13 (Dense)             (None, 6)                 42
_________________________________________________________________
dense_14 (Dense)             (None, 4)                 28
=================================================================
Total params: 1,105
Trainable params: 1,105
Non-trainable params: 0
```

The optimizer that we use is Adam.

The **Adam optimization algorithm** is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

The loss that we use **sparse categorical cross entropy**.

sparse categorical cross entropy: When your classes are mutually exclusive

categorical cross entropy: When one sample can have multiple classes or labels are soft probabilities.

$$\text{Loss} = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where k is number of classes in the data

Refinement

To improve the Prediction Model:

The model suffers from underfitting. So, to overcome underfitting:

1. New dataframe with highly recommended features and dependent features:
    a. more layers and hidden units
2. Train the model longer
3. Advanced optimization algorithm

Section 4: Results

Model Evaluation and Validation

To evaluate the model we use a validation set:

```
Train on 190933 samples, validate on 81829 samples
Epoch 1/15
 - 15s - loss: nan - acc: 0.2433 - val_loss: nan - val_acc: 0.2448
Epoch 2/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 3/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 4/15
 - 11s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 5/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 6/15
 - 14s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 7/15
 - 14s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 8/15
 - 10s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 9/15
 - 10s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 10/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 11/15
 - 15s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 12/15
 - 16s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 13/15
 - 13s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 14/15
 - 14s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
Epoch 15/15
 - 16s - loss: nan - acc: 0.2434 - val_loss: nan - val_acc: 0.2448
```

```
model.evaluate(X_test , y_test)
```

```
81829/81829 [==============================] - 7s 86us/step

[nan, 0.2448154077419889]
```

To do that, we have followed this steps:

- We use One-hot-encoding. In this case we encoding the gender and offer_type column
- We split the dataset using train_test_split function, that we import,
- We split training and testing. We use the training for build the model and we use the testing to evaluate the performance of the model.

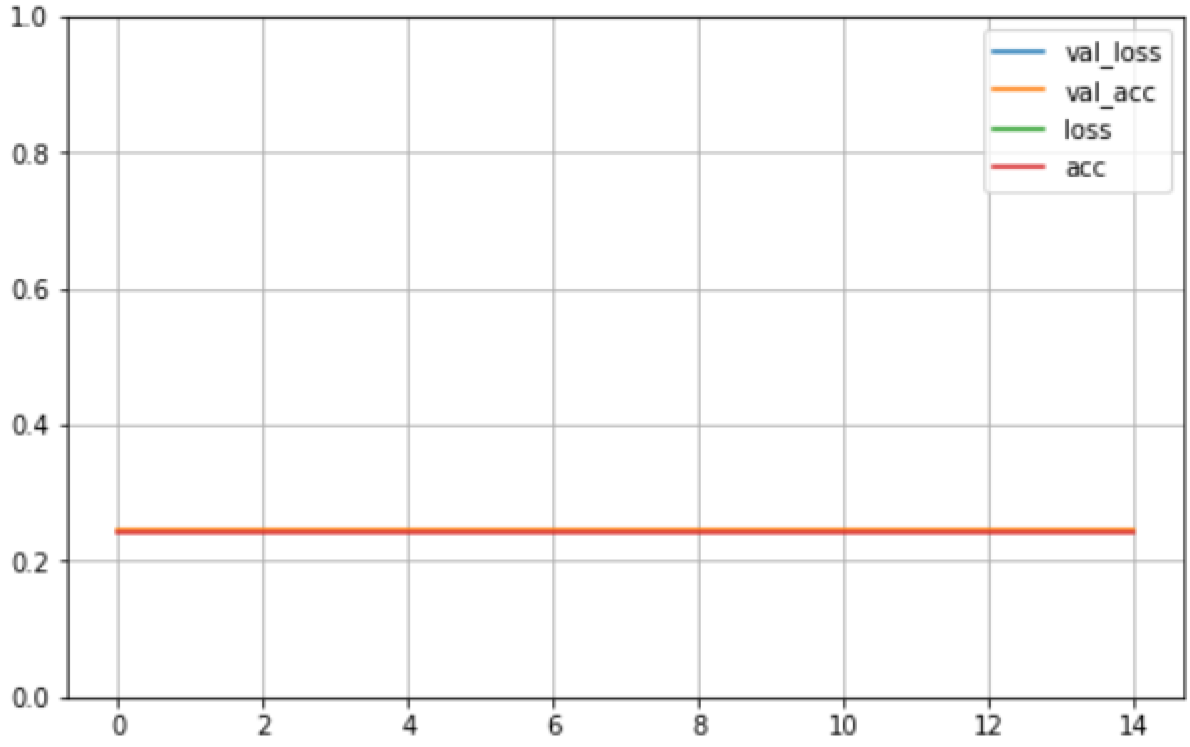
Scaling: normalization and standardization:

Standardization:Scaling technique, where the values are centered around the mean with a unit standard deviation.

Normalization: Scaling technique, where the values are shifted and rescaled.

- We need to convert the pandas dataframe to np array Pandas dataframe is 2-D size mutable, potentially heterogeneous tabular data structure with labeled axes (col and raws). We can covert this data structure to Numpy with Dataframe.to_numpy() method.

To evaluate the model we use a validation set where we can see that the accuracy is 0,244.



We can see that the improvement has similar accuracy.

Section 5: Justification

Underfitting techniques:

- Kernel Initializer : "Normal" The neural network needs to start with some weights and then iteratively update them to better values. The term kernel_initializer is a fancy term for which statistical distribution or function to use for initialising the weights. In case of statistical distribution, the library will generate numbers from that statistical distribution and use them as starting weights.

- Increase number of hidden layer and units

With these techniques we would have a better prediction model.

Section 6: Conclusion

Reflection

In my opinion this project has been very challenging, mainly because of the structure of the data, mainly in the transcript dataset.

The most occuring event is 'offer_received', so I tried to do a model with that, but the results of the model seems like not so good.

Major classes perform well but not minorities classes. This is because an imbalance dataset.

Main challenges and Potential improvement: Design, analysis and build deep learning model

My goal was to create a practical model to make choices more efficients, but the results are not good to do that. There is no change in rate of accuracy it keep constant.

Improvement

We use an imbalanced dataset, so we know that we don't have a great accuracy.

But this accuracy could be improved using deep neural networks or recommendation engines.

Potential improvement: Design, analysis and build deep learning model

REFERENCES

https://learn.udacity.com/my-programs?tab=Currently%2520Learning

https://pandas.pydata.org/docs/index.html

https://www.tensorflow.org/tutorials/keras/classification

https://forums.fast.ai/

https://stackoverflow.com/

https://www.appsloveworld.com/

https://scikit-learn.org/stable/

THANK YOU