RECOMMENDATION SYSTEM

Athens University of Economics and Business
Machine Learning and Content Analytics

Authored by: G.Vlassi p2822001
S. Vretteas p2822003
A. Ntetsika p2822014
D.Mentakis p2822024

Professors: H. Papageorgiou, G. Perakis

September 2021

Table of Contents

Abstract

The data of this report refers to the viewership of a pay television, including almost 1.5 million data of 15 features of the month December. The dataset contains demographic data like age and gender of the user, content based data like the genre description (e.g., Adventure, Crime, Romance), asset type and category, the name of the channel, the rating of the user and the viewership duration. After completing the data analysis, the clustering of the user's features will be implemented to describe the data. We use these data to build a recommendation system by using the Python programming language. Basic machine learning models are created, such as Long Short Term Memory (LSTM), which is a special version of Recurrent Neural Network (RNN) model and Collaborative filtering recommendation system. The predictions and weights of each model will be further assessed.

1. Introduction

The recommendation systems are becoming increasingly known and important. After the enormous impact that the recommendation system of Netflix has on its audience, Greek pay television tries to harmonize with it. Cosmote TV, Nova and Ertflix are a few examples of movie/series recommendation systems. The main scope of this assignment is to create a recommendation system, which can be sold and used by companies with viewership content.

The dataset contains 1.697.179 records of 15 features each. These features describe both the user and the content of the pay television. There are categorical variables, such us user's sex, asset's type and category, content's genre and view's day, which will help us both in the clustering part and in our machine learning models. Furthermore, we have the numeric variables, as the user's age, the duration of the program and the duration of the view. The two latter features will be used to calculate the percentage of the viewership in minutes. The rating of the user, which is an ordinal, will be used in the Collaborative filtering model, which will be appraised later.

To cluster our data K-Means algorithm is used alongside the PCA method for dimensionality reduction. Initially, we use PCA for the dataset with the 91 variables created after converting some features to categories and dummies in a scale of 0 to 1. The goal of PCA is to find a low - dimensional representation of the observation that explains a good fraction of the variance, in our case 95% of the dataset variation. As a result, we end up with 36 dimensions that sufficiently explain our dataset. Using these 36 variables and by implementing the Elbow method, we end up the optimal value of k in k-means is 3. As a result, we construct three clusters to find homogeneities among the observations. These subgroups will be further assessed.

In terms of the recommendation system, four models were constructed. In the first model we predict the sex of the user (male - female) from his/her username. The stacked LSTM architecture is used to produce this model. By constructing this model, we want to define if the input credentials of the user are real or if purposely have provided false details for himself. In the second model we use the sequences of the user's views with window size 5 in row and try to predict the next type of the content. The aim of this model is to understand the viewership behavior of the user and predict the next genre that he probably will watch (e.g. Kids, Christmas). The Long short-term memory architecture is used for this model too. The third model is an Item-Based Collaborative filtering model, which finds similarity between items and recommends similar items. More specifically, the Pearson similarity is used in this model and by the input that the user gives to the system, it returns the more correlated movies/series of the input. Our final model is a Multi- Model, which has as input the sequences of the user's views, as in the second model, and the user's gender. Similarly, the aim of this is an enhanced prediction of the next genre content. Last but not least, for all the models we use embedding methodology in the first layer, in order to improve accuracy.

2. Descriptive Analysis

Our dataset consists of real data from an Oracle's Telecommunication company client. It represents a real business case in an actual environment that meets real business needs. As a result, the data is not cleaned and some manipulations and transformations should be implemented. We have to clarify that the columns that have null values and need updates are the user's name, the source's genre, the channel's name and the asset's name. As all these columns are connected with each other, the transformations should be horizontal, in order to preserve as much information as possible.

Although, there are null values at the above features, we will not proceed in dropping them, still we will update them. The sample with null values at the user's name will be updated to 'UNKNOWN', because it is very small and will not affect the results of the first model, where it will be used. Similarly, the same process will be used in the rest of the variables.

Regarding the feature asset's category there are a lot of values marked as 'Not Available', a fact that will affect our outcomes. To handle this misalignment, we found such key words at the asset's name variable and update the asset's category respectively. For instance, where the asset's name is inside the list of Animation/Cartoon, Children, Lonney, SpongeBob, Mickey, Scooby, Tales, Jerry, Tom, OZ, Disney or Ducktales, the asset's category is updated to 'KIDS'. Moreover, categories, for which the asset's name is either Kardashians, Masterchef, Survivor or Murphy's updated to 'SHOW'. Likewise, have manipulated different categories, in order to reduce the nonexistent values. Implementing the above we have created a better handling dataset for our analysis.

3. Explanatory Data Analysis

A few diagrams were created to better understand our data and their distribution. In Figure 1 a Word Cloud has been constructed. The Word Cloud is a representation of words – tags. We use them to show the word frequency in a text, interview, document or dataset in our case. This type of visualization can assist evaluators with exploratory analysis. Observing the below figure, we can understand that the Kids, Comedy and Sports-Live are the most frequent genres with the most views in our dataset.

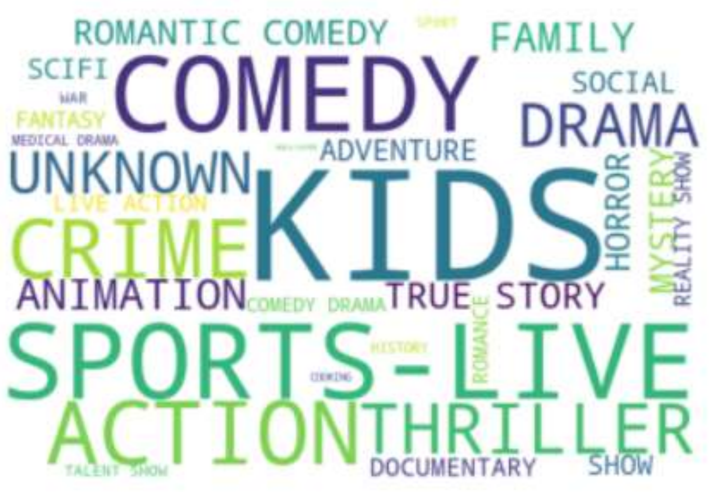


Figure 1 Word Cloud with genres

A very informative way for the distribution of our data are the pie charts. The pie charts are used to show percentages of a whole. Observing the first pie chart in the Figure 2 we can see how the three types of asset are distributed. We should clarify that the data collected from Video on Demand (VOD), which provide users with access to bundles of video entertainment content and individual movies and shows, from Live TV, also, where the user can stream sport games and live shows, and finally Catchup TV, where the user can view a program that he missed. The biggest sample belongs to VOD as it happens in pay television. Also, another pie chart is used to show the percentage distribution of the categories, with movies occupying most of the pie.

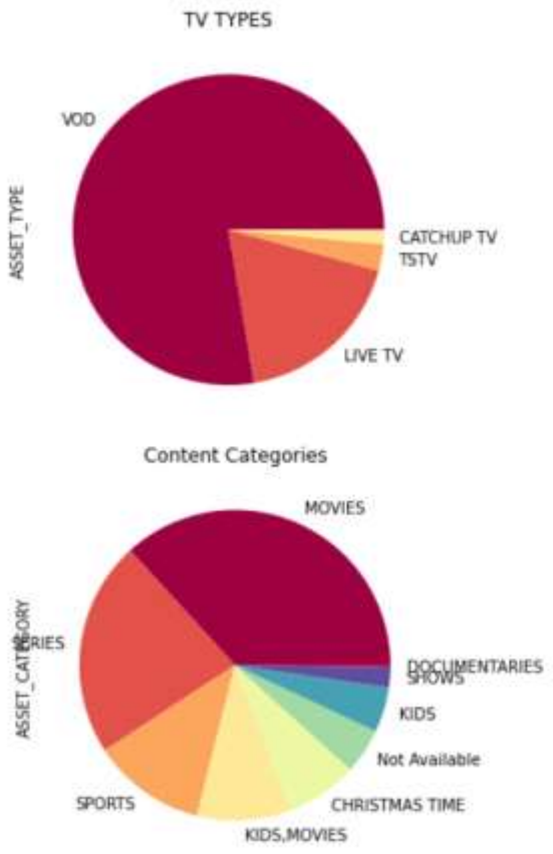


Figure 2 Pie charts of asset's characteristics

The user's gender is an important feature of the dataset , which will be used in our first model. In Figure 3, we can see that the genders are equally, as a result the prediction of the sex will be spontaneous and inviolable.

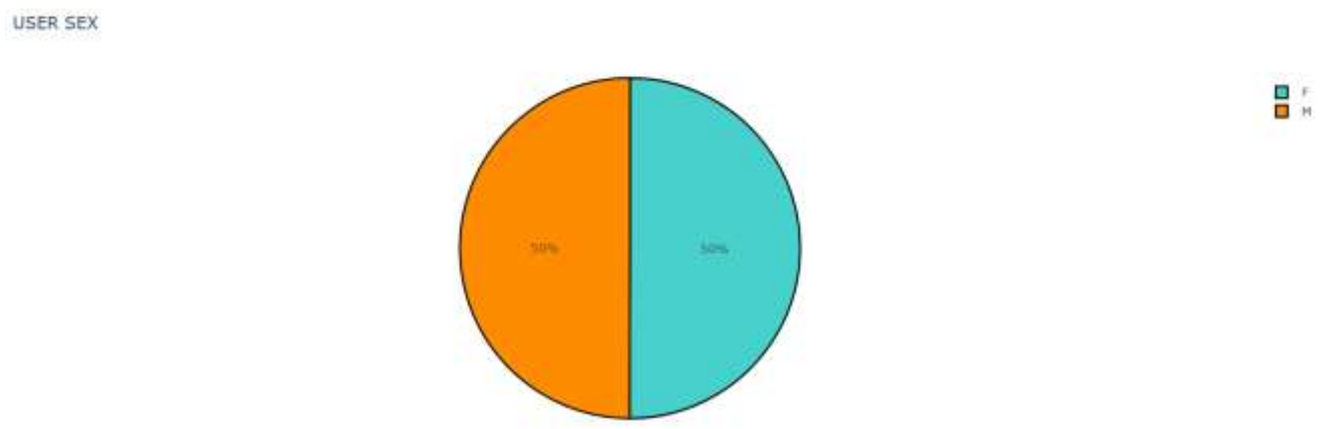


Figure 3 Pie chart of user's gender

In order to better understand the relationship among the most important numeric variables, a correlation plot has been drawn. The correlation plot is designed to give more emphasis to the relationships among attributes. If the orthogonal tends towards the yellow color, is a positive correlation and if it tends to the mauve, is a negative correlation. As shown to the correlation plot below in Figure 4, the highly correlated variables are the view's duration with the rating of the user.

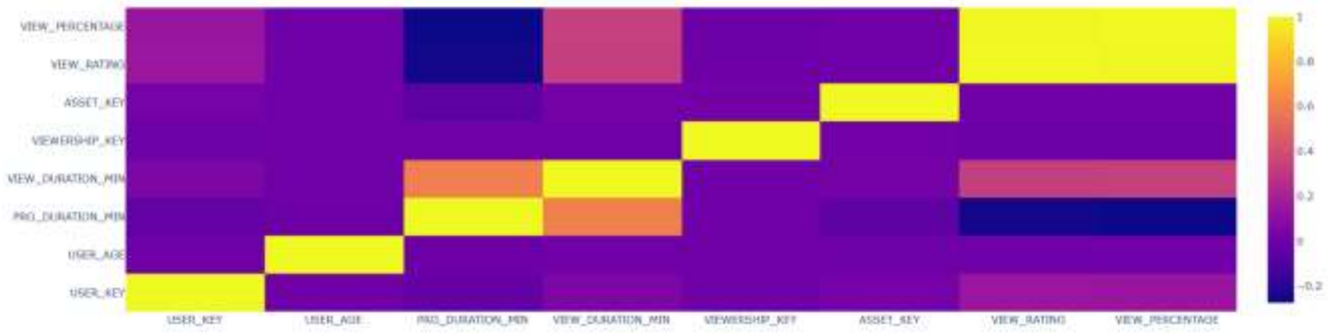


Figure 4 Correlation matrix among numeric features

4. Data Clustering

For our cluster analysis or clustering we need to group our objects in such a way that objects in the same group are more similar to each other than to those in other groups (clusters). Before implementing the clustering, we have to prepare our dataset and the variables that will be used. Initially, we create the dummy variables of the following categorical features user's sex, asset's type and category, source's genre and view day. Then, we use min-max scaler for the numerical features: user's age, program's and view's duration, as well the ordinal variable rating. For all the above groups of variables, we check for their normality with the use of the Shapiro test.

By performing the above, we end up with 91 components, including all the dummies created. Our analysis starts by using Principal Component Analysis (PCA) before K-means clustering. It is an approach to reduce the dimensions and decrease the computation cost. The PCA is an unsupervised learning algorithm, which uses a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. After implementing the PCA method we find the lower-dimensional surface, which in our case is 36 as shown in the Figure 5.

Having reduced the dimension of the dataset the heuristic Elbow method is used to determine the number of clusters in a data set. This method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use. The 'knee of the curve' in our study is the 3 clusters as is shown in the Figure 6.

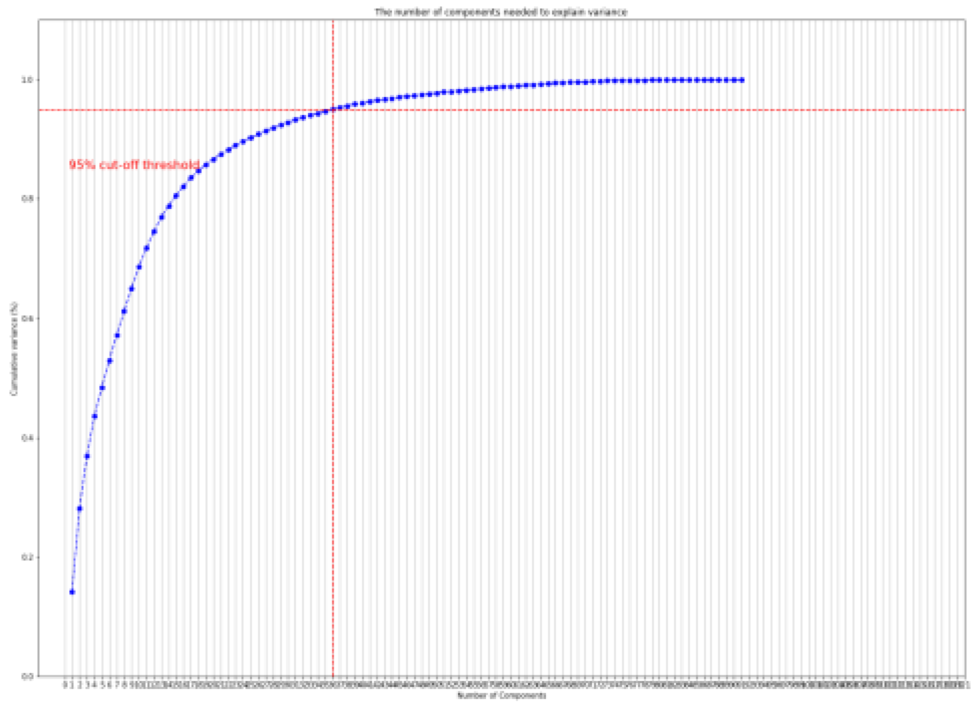


Figure 5 PCA - Number of Components

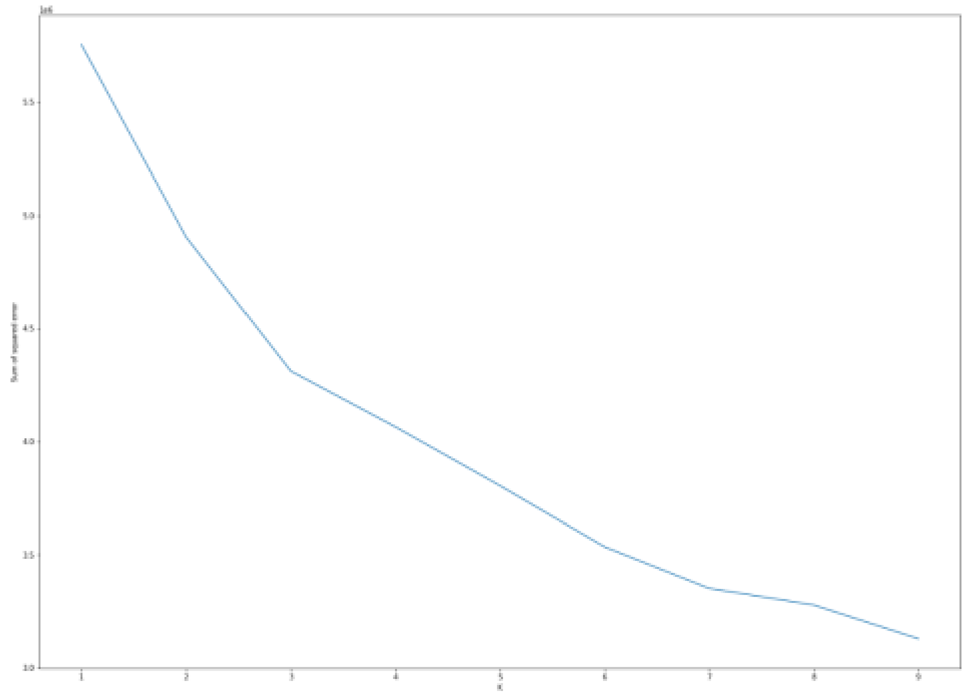


Figure 6 Elbow method

Having calculated the appropriate number of the clusters, we proceed in implementing an unsupervised machine learning algorithm called k-means clustering. We define a target number k equal to 3, which refers to the number of centroids we need in our dataset. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. The clusters are presented at the below Figure 7.

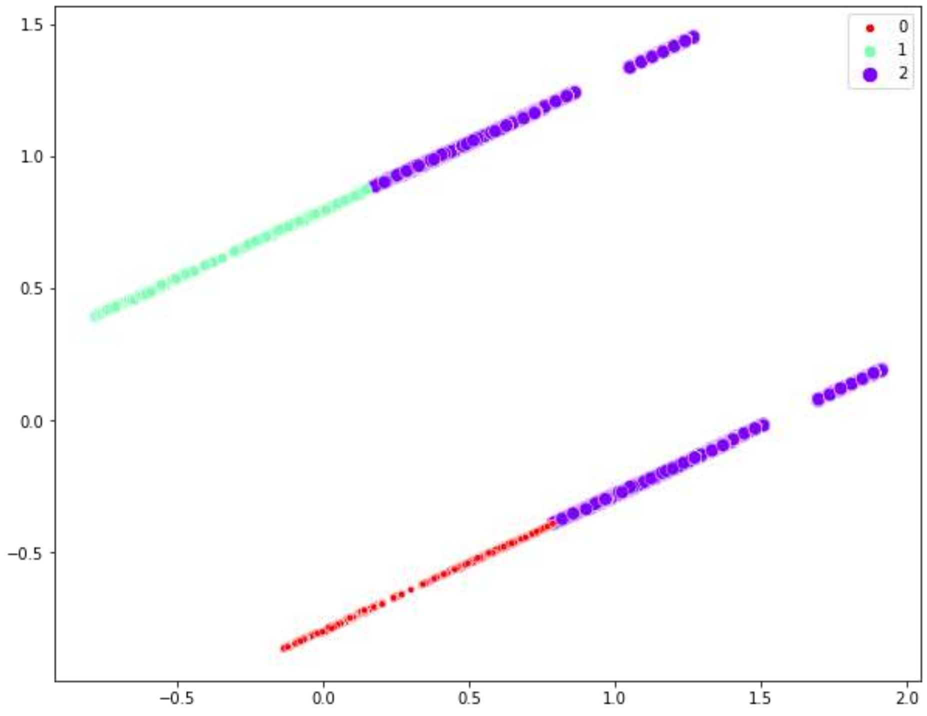


Figure 7 K-Means clustering

Every record is allocated in one of the three clusters based on its characteristics. Comparing the results of the allocation there are 692.708 records at the first cluster, 687.909 records at the second and 316.562 at the third. Analyzing the clusters, we observe that in the first one there are only female users, compared to the second, where there are only males. In the third cluster the genders are equally distributed. Regarding the age of the users in the clusters the average is 57 years old. The mean rating at the first two clusters is 4 and 2 at the last one. Furthermore, we observe that characteristics relevant to viewership’s duration are very close at the clusters one and two.

The k-means clustering was chosen because it consists only of two steps. These steps are the cluster assignment step and the move centroid step. As a result, it is easy and can handle large datasets, as in our study. Moreover, in cases where the k is small, K-Means clustering may be computationally faster than the hierarchical clustering. Completing our data manipulation and analysis we have prepared the sample for building our machine learning models.

5. Machine Learning Models

5.1 Model 1: Predict User's Gender from Username

In this section we will explain the first model. This tasks' purpose is to produce a model in order to predict user's sex derived from the username the subscriber registers. This can be very helpful in order to understand if the gender that the user declared is true. Many users, for various reasons, do not declare their true details when they register on an online platform. Either because they are afraid of the integrity of the platform (GDPR reasons), or they declare false details by purpose and act under false identity. The username a user declares, can be an indicator of their gender. Under the hypothesis that most of the users declare their true gender, with the help of deep learning, we will try to produce a model that can "learn" from the usernames of various users and correctly predict their gender.

A good fit for our task will be to implement a RNN (Recurrent Neural Network) model, as this architecture of neural networks is learning from sequences. In our case, the sequences will be the characters of each user's username. Furthermore, we are going to use the LSTM (Long Short-Term Memory) architecture in order to bypass the "problems" of the traditional RNN's.

First, we shall try to understand what the above deep learning architectures do, with simple examples. Humans like us, do not start thinking about everything from scratch every second. Our understanding comes from previous facts. When reading this text, our understanding comes from the previous words, so our thoughts have persistence over time. The traditional neural networks cannot do this unlike the RNN's which address this issue. RNN's are networks with loops, so they can "learn" from the past sequences, thus these loops allow the information fed to them to persist. Such a network can be thought of as multiple copies of the same network, each passing a message to a successor. Their chain-like reaction makes them the best fit for sequential data and lists. For better understanding, below is a visualization of an unrolled RNN.

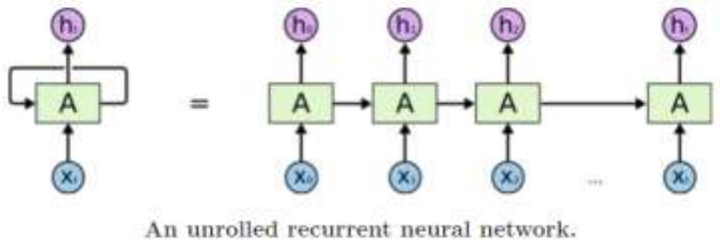


Figure 8 Unrolled RNN

Unfortunately, the drawback of the RNNs is that it fails to store information for a long period of time. Supposedly, we want to predict the next word in a sentence of four words then the predicted output would be quite obvious and would be easy to predict. If the gap is larger, let's say we want to predict the ninth (9th) word in a sentence, then the results would not be so good. This is called the problem of Long-Term Dependencies.

In order to face this, LSTM model architectures were introduced by Hochreiter & Schmidhuber (1997) and designed to avoid the long-term dependency problem. Their natural behavior is to “remember information” over long periods of time. LSTMs also have the chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer like the traditional RNN’s, in LSTM there are four, interacting in a very special way. With this structure, it is very easy for information to just flow along as it is unchanged and persists over time.

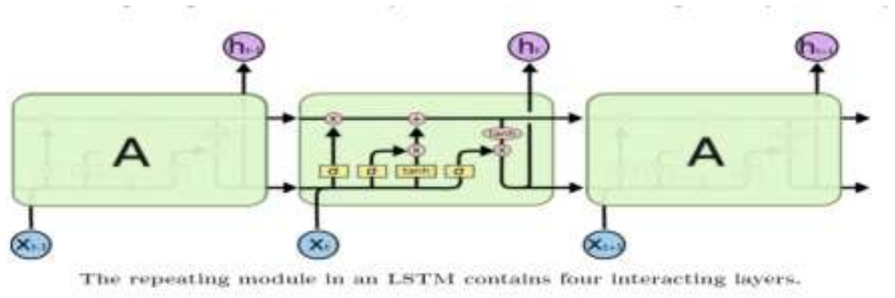

A
A
tanh
The repeating module in an LSTM contains four interacting layers.

Figure 9 LSTM architecture

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

Having briefly introduced the theory behind our model, is time to give details regarding the model. Our problem is to predict correctly if the given username indicates a male or female user. For this purpose, we will use a sequence of characters as our X variable and our Y variable will be the binary value “M” for Male and “F” for female respectively. We will build two models. The first will contain sequences of all letters and the other will contain only lower cases to see if there will be any difference. Thus, we implement the below procedures in parallel, one for the model with both capital and lower letters and the other for only lower case letters.

For the implementation, the first step was to create a vocabulary of the unique characters and create sequential data of the same length in order to feed them in our first embedding layer. We will use the embedding layer as the first layer, in order to achieve better results than the simple one-hot encoding. We exclude usernames with length of two (2) characters and usernames with length greater than twenty characters (20). Thus, we create vectors with length of twenty (20) elements/characters. Each vector contains integers with the mapping of the username from the vocabulary we created. To achieve the required length of twenty (20) we append the value of zero at each vector. This technique is called padding. In the final step, since the response variable is categorical, we are going to one-hot encode it. We assign the value 0 for the Female label and the value 1 for the Male label respectively.

Then, we split our dataset into train, validation and test datasets. First, we split the dataset with the stratify technique, in order to have homogenous datasets, into rest and test datasets with the ratio of 80% percent of the dataset for the rest and 20% percent for the test dataset. This test dataset will serve for evaluation and testing reasons later on. These will be our unseen data and will not participate in the training of the model. The test dataset consists of 20.506 rows.

Then, we split the rest into a train and validation dataset with ratio 75% and 25% respectively. These two will be used for the training of our model. The training dataset consists of 61.517 rows and the validation dataset consists of 20.506 rows.

Having done the above data manipulation, we are going to build our model. Our model will consist of an embedding layer with a vocabulary of sixty-five (65) characters , embedding size vectors of fifty (50) and input sequences of fixed length twenty (20). Then a stacked LSTM and a final dense layer with sigmoid activation will follow. We use after each LSTM layer a 20% percent dropout layer in order to avoid overfitting and noised data. Our loss function is binary_crossentropy (since our problem is a binary prediction) and we use, also, Adam optimizer with learning rate of 0.002. We train the model for 50 epochs with a batch size of 64. For metrics, we are going to use binary accuracy. Last but not least for the training session we are going to implement some callbacks in order to avoid overfitting. Specifically, we implement callbacks Early Stopping where we monitor the “validation loss” quantity with patience for seven (7) epochs. This means that if the validation loss does not improve for seven (7) epochs in a row then our model will stop training. In addition to this, we implement Model Checkpoint too, we monitor the "validation loss" quantity too and we save the best model according to this quantity after each epoch.

Below is the training and validation loss for each epoch for model1

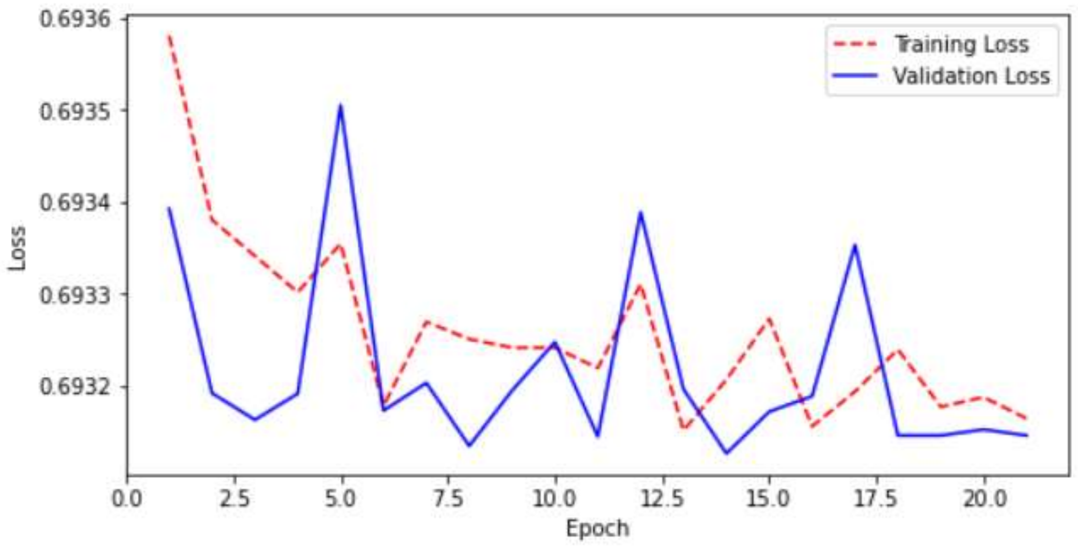


Figure 10 – Model1 Train & Validation Loss

Our model stopped training after twenty-one (21) epochs due to the Early Stopping callback as the "validation loss" quantity did not improve after seven consecutive epochs. This fact happened in order to prevent overfitting. As a result, our training stopped.

The evaluation of the model when tested at the unseen data showed that the loss in the test dataset was 0.6931 while the test accuracy was at 0.4999.

Regarding the second model which contains only the sequences with the lower-cased letters. The results are presented below.

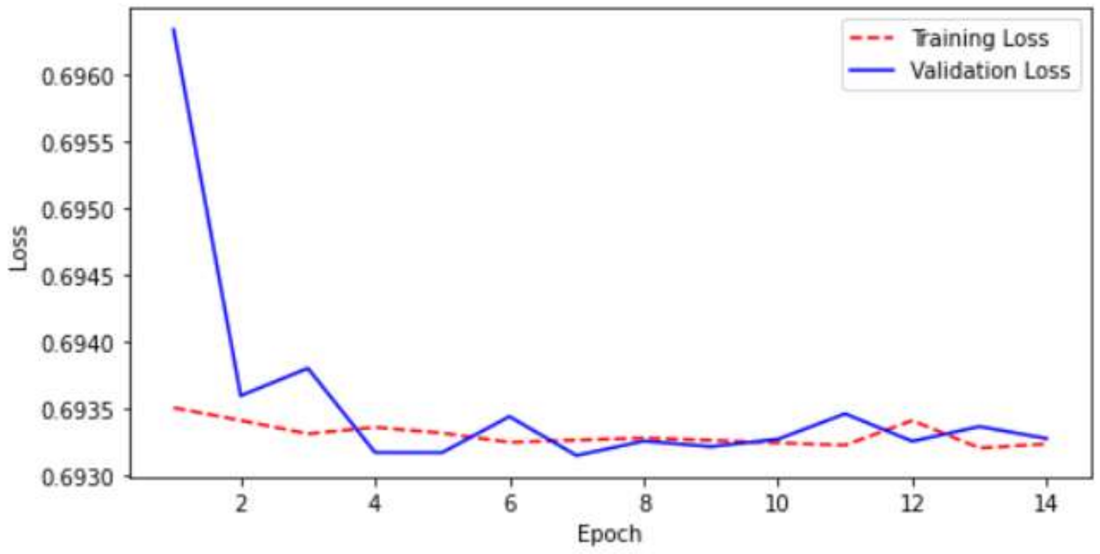


Figure 11 - Training - Validation Loss - Model with lower letters

The model1_lower as called stopped training after 14 epochs, as the “validation loss” quantity did not improve after seven (7) consecutive epochs too.

The evaluation of the model1_lower at the test dataset showed test loss of 0.6931 and test accuracy equal to 0.5021 slightly better than the previous version but there was not a big difference.

5.2 Model 2: What's Next Prediction

In this section, we will explain task2's model. This task's purpose is to create a model from the viewership behavior of all users in order to predict the next Genre the user will watch. For business purposes this will be used in terms of the recommendation system tool we introduce. Here our model will try to predict the next genre based on behavior data. The implementation is similar to a word text generator. Finally, alongside the model, we will create a function which will take as arguments, the model, the sequential data, the actual Y output and will return the predicted Y value.

For the purposes of this task, a good fit will be to use LSTM deep network architecture, a special category of the RNN (Recurrent Neural Network) as we introduced previously. Given five (5) sequences of Genres, our model will try to produce the sixth (6th). The unique structure of the LSTM architecture which allows the information to persist over a longer period of time than the traditional RNN's networks will be essential to our task.

The implementation of the data preparation is presented below. Firstly, we create a vocabulary with the unique value of the "SRC_GENRE_DESC" column. We tokenize the unique values and then we give each token an id in order to map it and feed the mapped sequences to our first embedding layer. Regarding the dataset, we sort the dataset chronologically in order to get the historical data correctly. In addition, we group the entire dataset by "USER_KEY", in order to get lists of the Genre's sequences. We will exclude sequences of length less than six (6) because we want to feed our model with sequences of length five (5) in order to predict the sixth. Afterwards, we will use the windowed method in order to iterate through the entire sequential data with a window size of length 5 and create the input of our equal length sequences. Alongside this iteration we take the sixth value for every window which will serve as our actual Y output value. Last, but not least in our data preparation we map the created sequences with their vocabulary token ids in order to have integer format inputs.

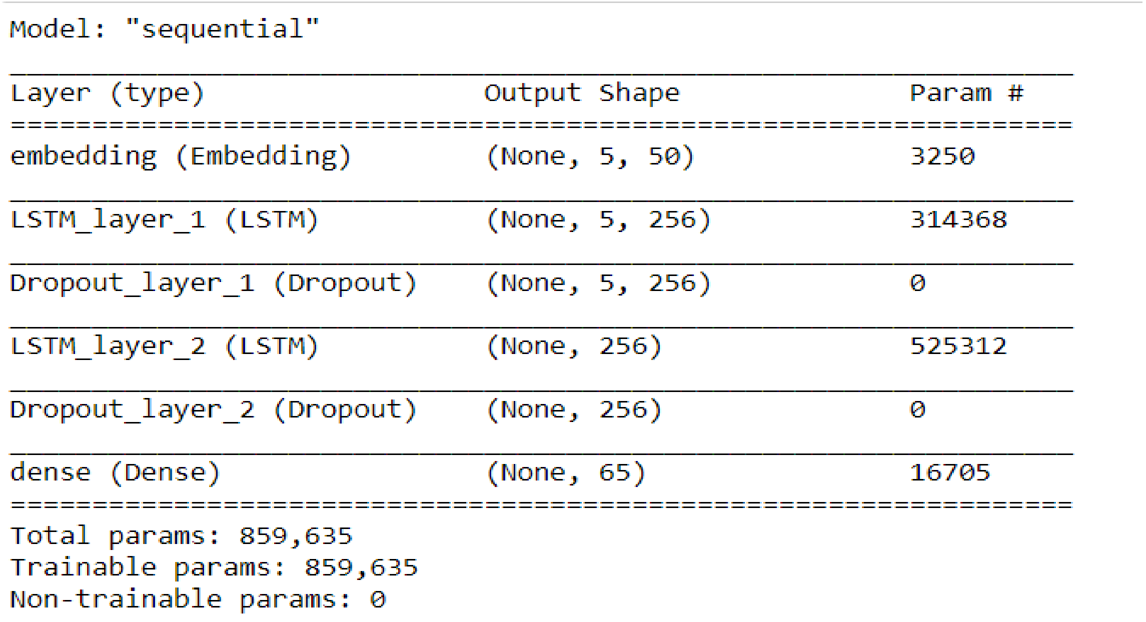
Before modelling, we will split the data into train, test and validation datasets using the stratify technique in order to get homogenous datasets.

For building the model, we use an embedding layer as our first layer. We use an embedding layer to compress the input feature space into a smaller one. An embedding tries to find the optimal mapping of each of the unique words to a vector of real numbers, the size of this vector in our case is equal to fifty

(50). This is an alternate to one-hot encoding technique along with dimensionality reduction. Embeddings will group commonly co-occurring items together in the representation space. Therefore, this approach is more efficient.

After the embedding layer, we will use a stacked LSTM architecture, a dropout of 30% in order to remove the noise data and prevent overfitting. For the final layer, since we face a classification problem, we use SoftMax activation function for getting the probabilities for each category.

In the Figure 12 we can see the architecture of the model.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        (None, 5, 50)             3250      
_________________________________________________________________
LSTM_layer_1 (LSTM)          (None, 5, 256)            314368    
_________________________________________________________________
Dropout_layer_1 (Dropout)    (None, 5, 256)            0         
_________________________________________________________________
LSTM_layer_2 (LSTM)          (None, 256)               525312    
_________________________________________________________________
Dropout_layer_2 (Dropout)    (None, 256)               0         
_________________________________________________________________
dense (Dense)                (None, 65)                16705     
=================================================================
Total params: 859,635
Trainable params: 859,635
Non-trainable params: 0
```

Figure 12- Model2 architecture

Furthermore, for metrics we will implement accuracy, top_k_categorical_accuracy with k=5 so as to check the accuracy of the model for the top 5 categories and auc metric which stands for area under curve.

For the training section we implement an early stopping callback. We monitor the “validation loss” quantity with patience five (5). If the “validation loss” does not improve after the iteration of five (5) epochs the training will stop. This is made to avoid overfitting. Finally, an Adam optimizer is used with a learning rate of 0.002.

Regarding the training of the model, is important to say that, when we trained the model for 100 epochs, we achieved an excellent accuracy of 97% but the validation accuracy was at 40%. This was a clear sign of an overfitting model. Thus, we add the early stopping callback, as mentioned before and change the monitored quantity from “training loss” to “validation loss”. The model stopped after eleven epochs of training since the validation loss quantity did not improve (from 2.34061). The results of the training process are presented below:

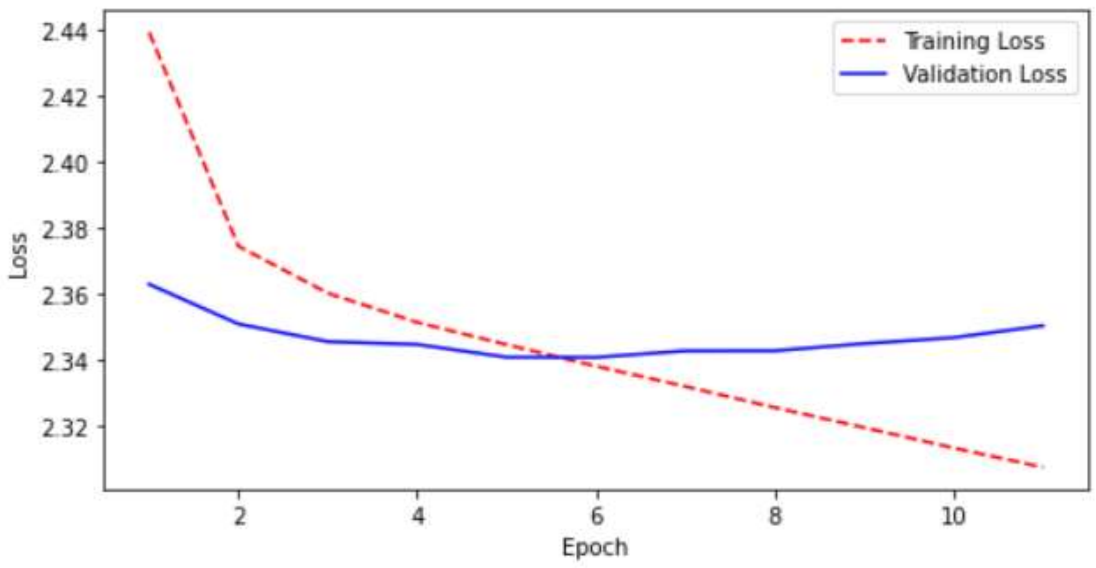


Figure 13- Training & Validation Loss

Regarding the evaluation of the model on unseen data, the model had the below results. Test loss at 2.3455, test accuracy at 0.4083 and regarding the top 5 categories the model performed quite well having a 0.6969 which means a rate similar to 70%.

Last, for business and visual purposes, we create a function which presents the input values, the actual sequence and the value that the model predicted. Below, are some indicative examples:

```
Input Values: ['KIDS', 'HORROR', 'HORROR', 'TRUE STORY', 'ACTION,CRIME']
Actual Value: DOCUMENTARY
Predicted Value: KIDS
```

5.3 Model 3: Collaborative Filtering

We all have used services like Netflix, Amazon and YouTube. These services are very sophisticated systems to recommend the best items to their users. But how do these platforms achieve this?

Collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources. Basically, it is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users.

Collaborative Filtering has the two following approaches:

- **Memory-Based:** It is an approach which finds similarity between users or between items to recommend similar items.
- **Model-Based:** In this approach we use different data mining, machine learning algorithms to predict users' rating of unrated items.

In this project, the first approach has been used and more specifically item based collaborative filtering was implemented. In this approach, we try finding movie's look-alike. Once we have movie's look-alike matrix, we can easily recommend alike movies to users who have rated any movie from the dataset. Before starting our analysis regarding the Collaborative filtering, we analyzed the mean ratings of each asset. The probability distribution of these mean ratings is shown in the Figure 14.

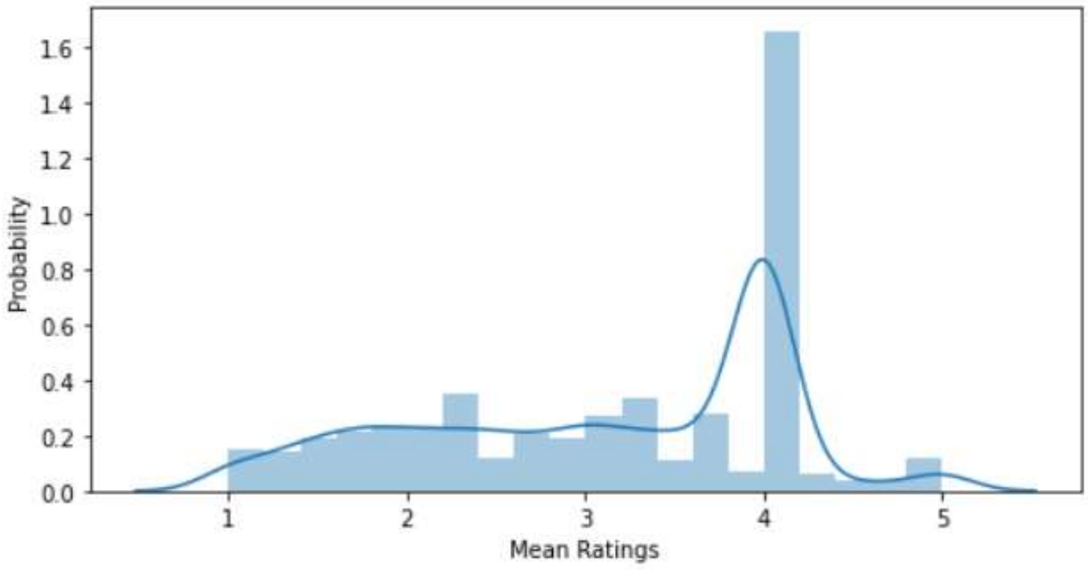
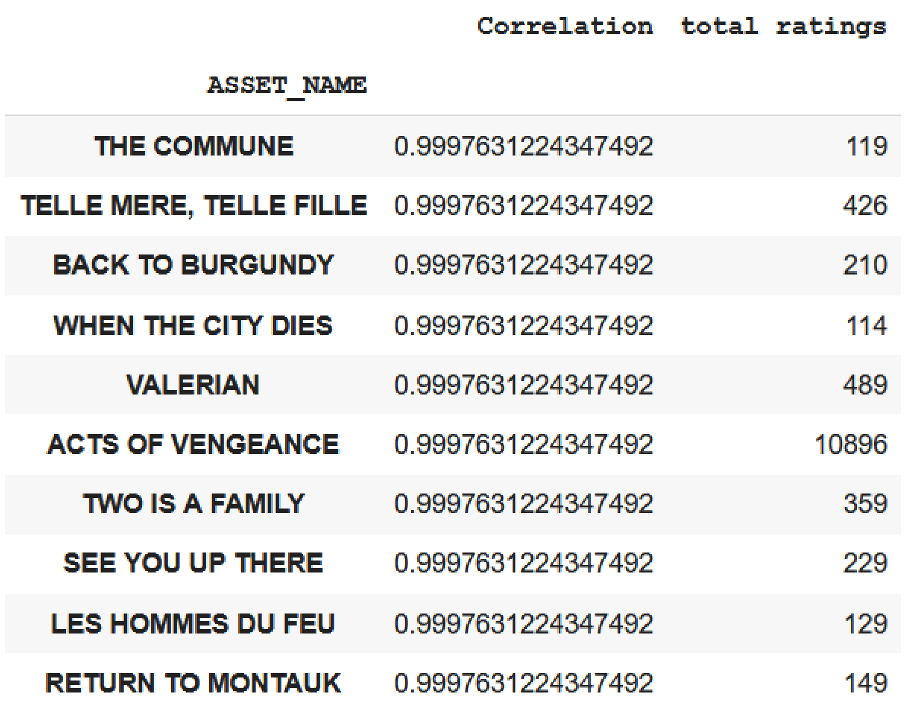


Figure 14 Mean Ratings with probabilities

Firstly, we create a dataset that has three columns: user's name, asset's name and view rating. After that, we create a pivot table in which the rows are Users and the columns are Movies. Each cell contains the View Rating of each user for each movie. In the following step, we calculate the correlations (Pearson Correlation) in order to find the correlations among movies. As a result, each movie has correlation equal to 1 with itself and high correlation with movies with relevant content. NaN values here, mean that a user has not seen a movie, so NaN values are replaced with the mean view rating. Finally, we enter the name of the movie we want, in order to recommend movies with similar content. We select the movies that have the biggest correlation with the chosen one and we return a list of them. The list contains the correlations and the total ratings, as shown in the Figure 15 for the movie 'The Commune'.



ASSET_NAME	Correlation	total ratings
THE COMMUNE	0.9997631224347492	119
TELLE MERE, TELLE FILLE	0.9997631224347492	426
BACK TO BURGUNDY	0.9997631224347492	210
WHEN THE CITY DIES	0.9997631224347492	114
VALERIAN	0.9997631224347492	489
ACTS OF VENGEANCE	0.9997631224347492	10896
TWO IS A FAMILY	0.9997631224347492	359
SEE YOU UP THERE	0.9997631224347492	229
LES HOMMES DU FEU	0.9997631224347492	129
RETURN TO MONTAUK	0.9997631224347492	149

Figure 15 Example with Correlation

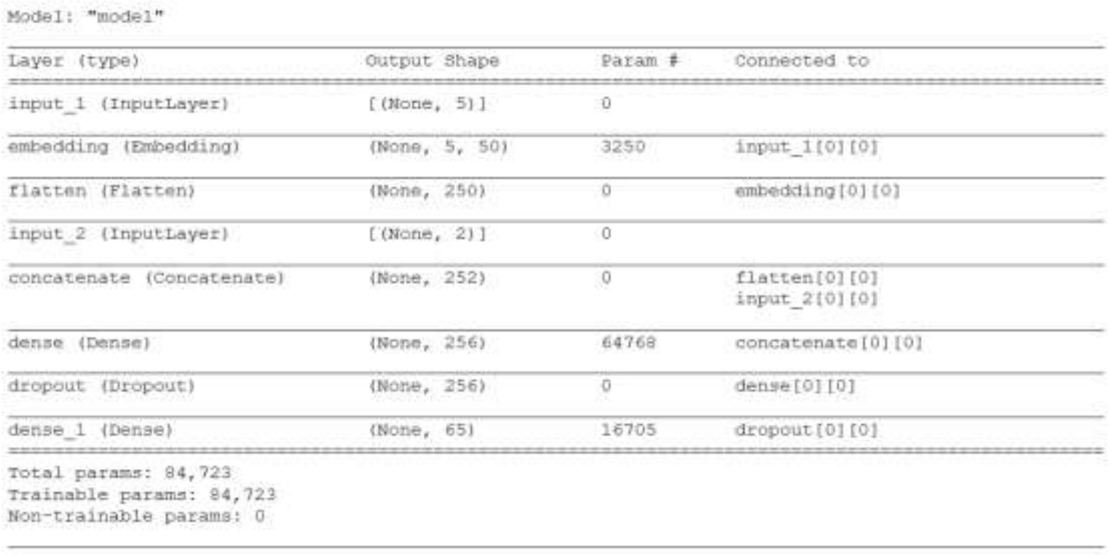
5.4 Model 4: Multi-Model

In this section, we will explain task4's model. For the implementation of this task, we are going to make a model containing multiple inputs. The basic difference from task 2 is that except the genres sequences as input we add one more, information about user's sex. Thus, from the viewership behavior of all users and the sex of them we are going to predict the next genre the user will watch.

For the business part, this model will enhance our recommendation system tool that we have already introduced. Furthermore, it will give to our model extra information for the input and thus the results we will get will be more enriched than task 2. As we did in task 2, we are going to use embeddings again and LSTM deep network architecture. We will flatten the output of the embedding layer in order to concatenate it with the output of the other layer which will contain the vectors containing information regarding sex. Thus, the first input is data that fed to the embedding layer (like model2 – vectors with fixed length of size five (5) which contain the viewership behavior) and the second input is vectors with length of size two (2). The position with index zero (0) specifies the female gender while the position indexed one (1) specifies the male gender.

The implementation of the data preparation is similar as we presented above. In task 2 we make the same implementation in parallel for both input features in order to not lose information. Firstly, we begin with the vocabulary by taking the unique value of the genre's description and of the user sex. We tokenize the unique values and then we give each token an id with the aim of mapping it and feed the fixed length vectors of mapped sequences to our first embedding layer. Then we create id's for both genre and user id towards to make integer our variables and thus we could feed our model. We continue with excluding sequences of length less than six (6) because we want to feed our model with sequences of length five (5) in order to predict the sixth. Afterwards, we will use the windowed method with the purpose to iterate through the entire sequential data with a window size of length 5 and create the input of our equal length sequences. Alongside this iteration, we take the sixth value for every window which will serve as our actual Y output value. For the user sex, we do not have to append sequence because the values will be in a "standard" sequence that will be mapped. Before training, we are going to split the data into train, test and validation datasets using the stratify technique to get homogenous datasets.

For building the model, we will use an embedding layer as our first layer. We use an embedding layer to compress the input feature space into a smaller one for the genre and alongside for the user sex. We add a dropout of 30% to exclude noise data. To finalize, we used the last layer based on vocabulary size in order to predict the probability of each category. Since we do classification, we use SoftMax activation function for getting the probabilities for each category. Below we present the model4 architecture.



```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 5)]          0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 5, 50)        3250        input_1[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 250)          0           embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 2)]          0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 252)          0           flatten[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          64768       concatenate[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 256)          0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 65)           16705       dropout[0][0]
==================================================================================================
Total params: 84,723
Trainable params: 84,723
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 16 - Model4 architecture

Furthermore, for metrics we will implement accuracy, top_k_categorical_accuracy with k=5 so as to check the accuracy of the model for the top 5 categories and ROC curve. For the training section we implement again an early stopping callback to prevent overfitting, we monitor the “val_loss” quantity with patience five (5), if the “val_loss” does not improve after the iteration of five (5) epochs the training will stop. Finally, an Adam optimizer is used with a learning rate of 0.002.

The model stopped its training after eleven (11) epochs since validation loss did not improve from value 2.36096 after five (5) consecutive times. Below is the training and validation plot for model4.

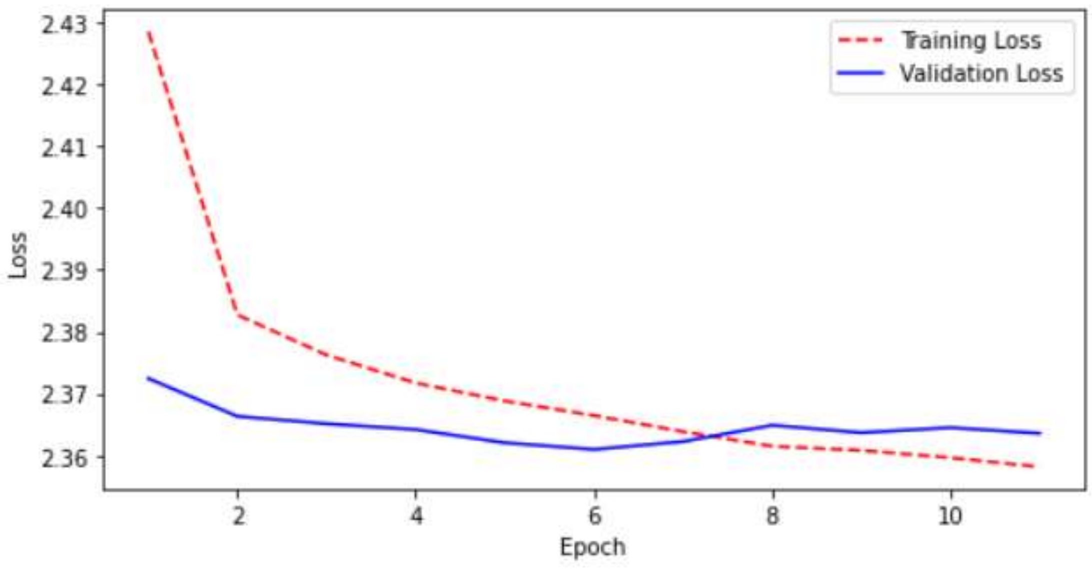


Figure 17 - Training & Validation loss

Regarding the evaluation of the model, the model performed as below at unseen data.
Test loss at 2.3638, test accuracy at 0.4078 and top-5 categorical accuracy at 0.6928 which is very good as it stands nearly at 70% rate.

6. Conclusion

In conclusion, for this project a real dataset from a client with a real business need is used. We tried to meet these needs and give solutions to real business problems. We implemented four models in order to categorize the users and analyze better their behavior. All the above, have not only the goal to build a recommendation system, which is used to predict relevant movies and series to every user, but also to give solutions and insights in various side business cases.