POLITECNICO DI MILANO
School of Industrial and Information Engineering
Department of Electronics, Information and Bioengineering
Master of Science in Telecommunications Engineering

Machine Learning based QoE Prediction in Mobile Cellular Networks

Supervisor: Prof. Alessandro E. C. Redondi
Co-Supervisor: Andrea Pimpinella, Ph.D.

Tesina of:
Jaya Ram Kollipara, 896254

Academic Year 2018-2019

I dedicate my work to my family and loved ones.

Acknowledgements

I would like to take this opportunity to express my profound gratitude and deep regards to my supervisor Prof.Alessandro E.C. Redondi for the continuous support of my thesis and for his patience and motivation. Also, special thanks to Mr. Andrea Pimpinella, Ph.D. for insightful comments and encouragement through out my research period. And I am grateful to our Institution Politecnico di Milano for providing me this opportunity.

Finally, I would like to thank the people who contribute the most to my life, my family. Also, to my friends for their love and support. Without them, none of this work would have been possible.

Sommario

Gli operatori di rete sono interessati a monitorare continuamente la soddisfazione dei propri clienti per ridurre al minimo il tasso di abbandono: tuttavia, raccogliere i feed-back degli utenti attraverso i sondaggi è un compito scomodo. In questo lavoro esploriamo la possibilità di prevedere la soddisfazione dell'utente a lungo termine relativamente alla copertura della rete a partire solo dalle misurazioni della rete lato utente. Utilizziamo i set di dati nazionali per ingegnerizzare le funzionalità che vengono poi utilizzate per addestrare modelli di apprendimento automatico supervisionati come le reti neurali. I risultati ottenuti suggeriscono che, sebbene una certa correlazione sia visibile e possa essere sfruttata dai classificatori, la previsione della soddisfazione degli utenti a lungo termine dalle misurazioni della rete è un compito molto impegnativo: pertanto, indichiamo possibili punti di azione da implementare per migliorare i risultati di previsione.

Abstract

Network operators are interested in continuously monitoring the satisfaction of their customers to minimise the churn rate: however, collecting user feed backs through surveys is a cumbersome task. In this work we explore the possibility of predicting the long-term user satisfaction relative to network coverage starting from user-side network measurements only. We leverage country-wide data sets to engineer features which are then used to train Supervised Machine learning Models such as Neural Networks. The obtained results suggest that, although some correlation is visible and could be exploited by the classifiers, long-term user satisfaction prediction from network measurements is a very challenging task: we therefore point out possible action points to be implemented to improve the prediction results.

Contents

List of Figures

Chapter 1

Introduction

According to recent Cisco estimates, by 2021 mobile cellular networks will connect more than 11 billion mobile devices and will be responsible for more than one fifth of the total IP traffic generated worldwide. Aware of these facts, mobile network operators are constantly monitoring and improving their access networks in order to give the best possible service to users and reduce failures, with the final goal of generating profit. This goal can be reached on the one hand by attracting as many new customers as possible and on the other hand trying to minimise the churn rate, i.e., the percentage of customers that, due to an unsatisfactory service, stop their subscriptions and move to a different operator.

In order to monitor the level of satisfaction of their customers, network operators often rely on surveys and questionnaires. Standard tools exist to capture the level of satisfaction of users through general questions: as an example the Net Promoter Score (NPS) survey asks users to indicate the likelihood of recommending the network operator to a friend or colleague on a scale from 0 to 10. In addition to such a generic survey, operators often ask customers to reply on very specific questions related to the user satisfaction relative to certain mobile network services (network coverage, voice and video quality, etc.), which can better highlight possible problems in the network. Based on the results of such surveys, operators have some clues on which services should be boosted up and possibly where: as an example, the operator can invest money to increase the bandwidth or the output power available at a certain base station. Unfortunately, performing customer feedback surveys is costly and cumbersome for operators, mainly due to the generic poor cooperative attitude of customers.

At the same time, network operators has several ways of gathering objective measurements from their customers: radio statistics and channel quality indicators can be obtained at the Radio Access Network (RAN), while advanced measurements such as throughput or latency can be measured with deep packet inspection (DPI) devices and network probes nowadays commonly installed at the network gateways (GGSN in 3G networks or PGW for 4G networks). Additionally, operators may obtain network measurements directly from the user terminals with specific applications running in background and installed under the user consent. Leveraging the renovated interest in machine learning and artificial intelligence techniques, operators may therefore attempt to predict the satisfaction of their customers starting from network measurements only.

This work is an extension to existent predictive models developed by Dip. Elettronica, Informazione e Bioingegneria, Politecnico di Milano Vodafone Group, Network Engineering and Delivery. We present the prediction of customer satisfaction relative to network coverage and video streaming using robust machine learning models like Neural Networks. We focus here on long term satisfaction. We base our study on country-wide data sets obtained from Vodafone cellular network, containing both user-side activity measurements and ground truth satisfaction feed backs.

We devise an optimal approach to develop a neural network model using tensor flow and perform hyper parameter tuning using Bayesian optimization technique. We also present some important concepts like cross-validation techniques, missing value treatment, variable or feature transformations.

Chapter 2

State of the Art

The prediction of user satisfaction is quite helpful for the network operators to provide better quality to its customers. In these scenarios we have very few feed backs from the user and also imbalance is the binary labels. In this chapter, we present few relevant work on the problem statement under consideration.

2.1 Estimate QoE of Video Transmission:

In this research [1], the authors proposed a deep learning-based QoE assessment approach with a large-scale QoE data set for mobile video transmission. Each QoE metric is only related to a minority of the 89 network parameters. Feature selection methods are applied to reduce the redundancy of the network parameters for QoE assessment. The box-plot method is applied to clean the raw data by removing outliers. Finally, a deep neural network (DNN) is developed to learn the relationships between the network parameters and the subjective QoE scores. Measurements are taken from the mobile app and viewers are invited to view pre-installed videos. Size of the database in 80k samples with 89 features. Score data set is Subjective scores range from 1 to 5. whereas, 1 being the lowest, 5 being the highest. Score set of size 80k x 4. Where Users rate for Video Quality, loading, stalling and Overall Quality.

Author presented bidirectional search to find the feature subset.Bidirectional search is a graph search algorithm that finds the most related feature from an initial feature set to a goal subset in a directed graph. The degree of quality is measured by Pearson Correlation coefficient and Spearman correlation coefficient, 16 highest scores are selected in the feature subset.

Proposed Neural Network architecture is a 3 layer topology with hidden units 12, 48, 48 respectively. A logarithmic based loss function is used. It is stated that the complex relationships between the data and the results can be learned based on deep learning. The interactions between feature parameters can be learned through full-connections of the hidden layers, for generating the final QoE scores.The model is trained with the Adam optimizer. The learning rate is set as 0.0001. The iteration number is set at 30000, and the batch size is set at 128.

There was no proper approach in finalizing the topology of the neural network, it is commented that ReLu activation function is chosen and the best results were when the number of units in the II and III layer are 3 times the I layer.

Remarks:

1. A huge database with ground truth values were populated by encouraging the user to use the mobile application and rate based on built-in videos.

2. Neural Networks perform better than SVM, Decision trees.

3. There is no detail on hyper-parameter tuning. They assume that, if Neural Network are tuned correctly for the specific case, can expect more improvements.

.

2.2 Youtube QoE in Cellular Networks:

In this study [2], a analysis of QoE in cellular networks is addressed by using QoE and distributed network measurements collected in real users smartphones via YoMoApp, a well-known tool for collecting YouTube smartphone measurements and QoE feedback in a crowd sourcing fashion.

The data set is built from measurements collected with the YoMoApp tool, an app that we have conceived in the past to monitor network and QoE-relevant metrics related to YouTube directly at the smartphone. YoMoApp is publicly available and can be directly installed through the Google Play Store.Using YoMoApp, Measurements collected related to more than 3000 YouTube sessions worldwide, streamed on 70 different cellular-network providers to more than 360 different customers, between 2014 and 2018. The YoMoApp tool passively gathers multiple QoE relevant metrics and network performance indicators related to YouTube, including measurements at the player side (e.g., stalling events, changes in video resolution, initial delay), the network side (throughput, downlink/uplink bytes, radio access technology, etc.), as well as at the user side, retrieving user feedback through QoE surveys displayed by the app after completion of a session.

The prediction of four QoE-relevant metrics, the prediction of the MOS scores and the prediction of the user engagement , the corresponding predictors are built using machine-learning models, treating each problem as a classification task, where targets are discretized. The targets are as follows:

- whether initial delays (ID) are above or below a predefined QoE-relevant threshold – based on previous work on initial delay tolerance, we set this value to 4 seconds
- Whether a video quality switch has occurred during the session or not.
- The number of stalling events (NS), considering three classes – zero-stalling, mild-stalling: one or two stalling events, and severe-stalling: more than two stallings
- The stalling frequency or re-buffering rate (RR)

The prediction of MOS Scores is considered as a binary classification task, by threshold the score data set at score level 4. The prediction of user engagement is composed into a three-class classification task, predicting whether a user has watched less than 50% of the video, between 50% and 70%, or more than 70%. A random forest model with 10 trees through 10-fold cross validation prediction model is adopted. Simple bootstrapping techniques are used to balance classes for learning purposes.

The full feature set encompasses 275 features, including information about the received signal strength, the number of handovers, the number of network switches, and multiple statistics about the incoming and outgoing traffic, aggregated at different time windows of 1, 5, 10, 30, and 60 seconds. The traffic is measured on three different levels: the total traffic transmitted/received by the device, the traffic captured over the mobile network, and the traffic sent/received by the application itself. Feature-selection techniques are used to identify the most relevant features for each target.About 30 features out of the 275 are filtered to obtain high accuracy.

For the prediction of user engagement and MOS scores, multiple models are evaluated besides random forests, such as a single decision tree (DT), SVM, k- nearest neighbors (KNN), and Naive Bayes(NB). Also considered ensemble learning approaches, covering the three basic paradigms available in the ensemble-learning domain: bagging, boosting (AdaBoost (ADA) and gradient boosting (GRAD)), and stacking.

Remarks:

1. Relying on very rich and fairly large data set of QoE measurements collected at users smartphones with YoMoApp monitoring framework.

2. Different machine learning models are presented especially, the performances are higher for random forests classifier and its is assumed that Neural Networks can deliver better results.

3. Multiple classifications are used to predict the QoE.

4. Measurements can be directly accessed through Andrioid API instead of accessing application level KPI.

2.3 Web Browsing QoE Monitoring System:

In this work [3], the author has presented the prediction of QoE using Machine learning algorithms or deep learning approach in web browsing scenario. This work proposes a novel QoE model based on ML algorithms, namely, WBQoEM that estimate a MOS in web browsing service and predict how it can be used to modify and adapt network parameters before sending a page to the client.

Some of the dominant metrics measured are session length(that indicates number of clicks on visited pages) per user, rate indicating that a user decides to leave website just after the first page visit.Download and Upload throughput, total duration of surfing. The subjective measures, MOS scores are collected from the users, ranging from 0 to 5.

User rates the video played on a VLC over mininet to emulate video streaming over the Software defined networks. Based on these MOS ratings, the ML algorithm estimates the MOS and changes the real time video parameters in order to deliver the best quality to clients. With this approach the author draws a relationship between QoE and Qos.

Log transformations are used to remove the skewness in the data. Large data set is constructed by emulating a Software defined network. Correlations with respect to the target variable are taken and subset of features are selected. Out of all the features, the down link throughput is the highly correlated with the target variable.User age, sex, level of education and employment are also considered but were not strong enough to add value to the prediction. But, it is noted that user with age 22 to 25 years and good level of education have contributed to the data set. Before, training the model, the data set was binarized with a threshold of 4, where people who rated above 4/5 are marked satisfied while the rest marked unsatisfied.

Multiple machine learning models experiments are presented such as, Gradient Boosting, Linear Discriminant analysis, Logistic Regression, Random Forest, K-nearest Neighbours, Neural Networks. The data set splitting ration is Training(60%), Validation(20%) and Test(20%) and models are trained on training set and validated with validation set and reported performances on the test set. Out of which a binary Neural Network classifier gave the best results. It is concluded that the neural network can show better results with more data.

Remarks:

1. Relation between QoE and QoS is presented. It is commented that a better prediction of QoE will help the Operator and the Service provider to modulate the QoS.
2. The target variable, MOS scores are binarizeed and resulting data set is imbalanced.
3. Results of Various Classifiers are presented. A simple Neural Network has given best performances in terms of F1 score.
4. It is stated that, there is possibility to investigate in architecture and hyper parameters of Neural Network for even more better performances.

2.4 Summary:

In all the presented studies, [1], [2] and [3]. The major focus was on short term prediction. Our approach takes into account the memory of the user. Complex Non linear classifier such as gradient boosting, random forest were explored.In all the works, the authors presented the choice of Deep Neural Networks is expected to do better. Predominantly, the focus for these problems was made on amount of data set. Incentive based crowd sourcing platforms were used to attract larger set of customers to provide feed backs. In [3] it is clearly stated that, the performance of the classifiers will be better if more number of samples were used.

The important KPI's governing the prediction were discussed, out of which, CQI, SINR, Throughput features had high correlation with the target variable. In studies [2] and [1] the user satisfaction is viewed as a multi classification problem whereas in study [3], the MOS scores were threshold and treated as binary classification problem. Though the Neural Networks results were discussed by the authors in all the research papers. There is no substantial study on hyper parameter tuning and techniques to select the best Neural Network architecture.

In our current approach we present an approach to tune the hyper-parameters of Neural Network using Bayesian optimization. We also choose to use the long term behavior of a user within a cell of cellular network. As stated in the majority of the research papers, we also deal with the imbalanced data set and use F1 score, Precision , Recall and AUC as evaluation metrics for our performance.

Chapter 3

Datasets

This work uses two country-wide data sets coming from Vodafone, one of the major European mobile operators. A user side network measurements data set and a ground truth user satisfaction data set. Both data sets, in which users details are anonymised, are relative to a period of five months from May 2018 to November 2018

3.1 User Side Measurements Data Set

The first data set is obtained through a Vodafone-branded mobile application installed on a subset of the operator customers' equipment's and running in background under their consent. The application periodically logs several active and passive network measurements relative to low-level network indicators (e.g., average cell signal strength and channel quality indicators, daily time spent by the user in full or limited service conditions, etc.) as well as application level indicators (e.g., session downlink/uplink data volume, duration and throughput) of different applications run in foreground by the user. Beside the measurement itself, the application provides also additional information, such as the measurement timestamp or the ID and location of the base station to which the user is connected.
We are interested in measurements relative to (i) network coverage and (ii) video streaming. Regarding the former, the following measurements are avail-

able for each day d and only for 4G Radio Access Technology(RAT):

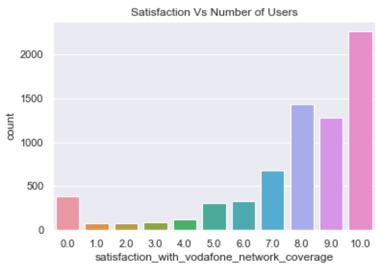
- Daily Full Service Time: The total time in seconds a user has reported full service in day d.
- Daily Limited Service Time : The total time in seconds a user has reported limited service (emergency service only) during the day d.
- Daily No Service Time : The total time in seconds a user has reported no service in day d.
- Signal to Noise Ratio (SNR)
- Reference Signal Received Quality (RSRQ)
- Channel Quality Index (CQI)

Regarding Video Streaming following measurements are considered:

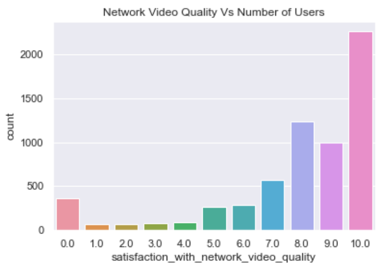
- Data Session Volume(DL)
- Data Session Duration (DL)
- Peak Data Session Throughput(DL)

3.2 User Satisfaction Data Set

The second data set contains ground truth feed backs of a subset of the operator customers on their satisfaction relative to different aspects of the received service (e.g. network coverage, video streaming quality, voice quality, data speed, etc.). The feed backs, collected by the operator through individual surveys, are reported in form of satisfaction grades on a scale from 0 (fully dissatisfied user) to 10 (fully satisfied user), and each user gives a different answer for each investigated network item. We extract from this data set only the feed backs relative to network coverage and video streaming, creating two distinct data sets QC and QV. Considering the five months period of analysis, QC contains 7045 survey responses for coverage and QV contains



(a) Network Coverage(QC)



(b) Video Streaming Quality(QV)

Figure 3.1: Satisfaction Vs Number of Users Before Thresholding

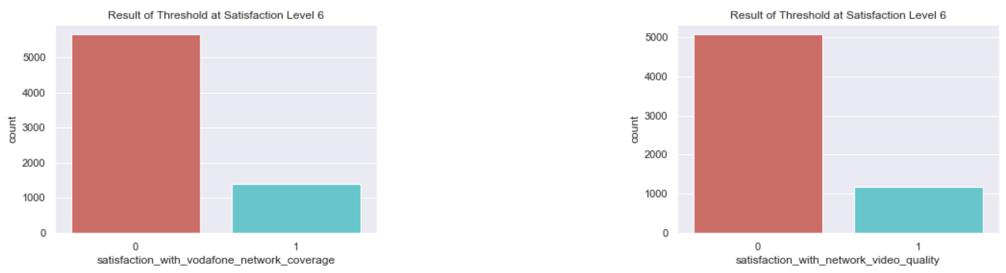
6264 survey responses for video streaming, 3.1a and 3.1b shows the distribution of satisfaction grades for the two considered services. As one can see, both distributions are highly skewed, with the majority of users reporting positive feed backs.

3.3 Binarize User Satisfaction Levels

It is possible to discretise the grades into two classes, with respect to a predefined satisfaction threshold T : users whose vote is less or equal than T are grouped together as Unsatisfied users, while the opposite happens for Satisfied users. According to [4], the best value of T is 6. Fig. 3.2a and 3.2b shows the results of binarization. In both the services the percentage of users unsatisfied is roughly 19%Ṫhe data set under test is Unbalanced data set.

3.4 Feature Description

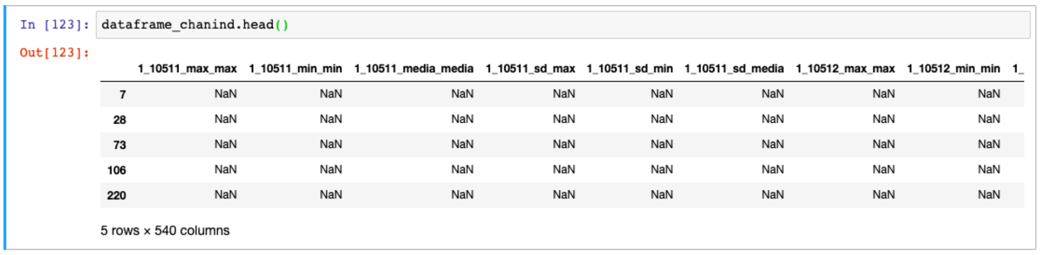
From [4], to make the long term prediction, a memory length of 30 days is taken. All the measurement per day is a new feature for our computations. Inheriting the data sets from [4], we have the following features computed on the raw data set. For the subset Network Coverage service measurement (CQI, RSRQ, SNR):



(a) Network Coverage(QC) (b) Video Streaming Quality(QV)

Figure 3.2: Satisfaction Vs Number of Users After Threshold

- max_max: Maximum of the hourly maximum measures
- min_min: Minimum of the hourly minimum measures
- media_media: Median of the hourly median measures
- sd_max: Maximum in the hourly variations measures
- sd_min: Minimum in the houlry variations measures
- sd_media: Meadian of the hourly variations measures



```
In [123]: dataframe_chanind.head()
Out[123]:
```

	1_10511_max_max	1_10511_min_min	1_10511_media_media	1_10511_sd_max	1_10511_sd_min	1_10511_sd_media	1_10512_max_max	1_10512_min_min	1_
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
73	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
106	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
220	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 540 columns

Figure 3.3: Data Frame imported using Pandas - Channel Related Measurements

There are 7045 entries of the users Network Coverage dataset. For each user we have 3 x 6 = 18 features. In order to make the long term prediction, we consider 30 days of all the measurements and we let the machine learning model to select the best inputs. Thus we have a total of 3 x 6 x 30 = 540

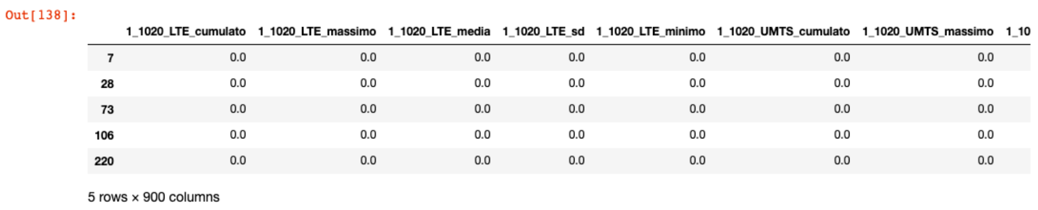
features. More details on Missing Value treatment will be discussed in the upcoming sections.
For the another subset of Network Coverage Measurements (Full/Lim/No), There are data with respect to following network access technologies:

- UMTS - Universal Mobile Telecommunication System(3G)
- LTE - Long Term Evolution(4G)

Similar to channel measurements, from this paper [**redondi**], following features are computed and made available for computations.

- cumulato: Cumulative Service time Ratio
- massimo: Maximum of the hourly Service time measures
- media: Median of the hourly Service time measures
- sd: Standdard Deviation in the hourly service time measures
- minimo: Minimum of the hourly Service time measures

For each user there are 3 x 2 x 5 x 30 = 900 features, where multiplication factor 3 corresponds to Full/Lim/No Service times. Factor 2 corresponds to Number of technologies i.e LTE + UMTS, factor 5 corresponds to type of measurements per day and factor 30 corresponds to memory of the long term prediction.



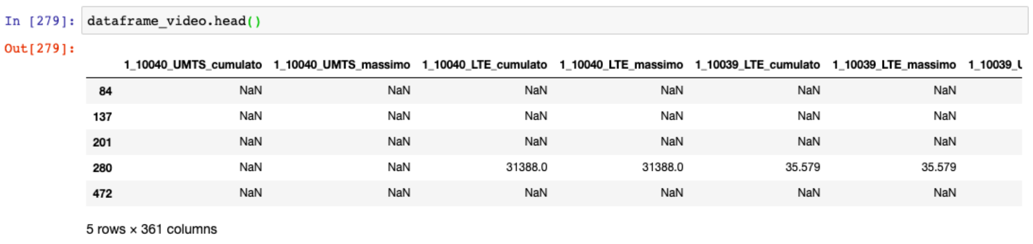
```
Out[138]:
      1_1020_LTE_cumulato  1_1020_LTE_massimo  1_1020_LTE_media  1_1020_LTE_sd  1_1020_LTE_minimo  1_1020_UMTS_cumulato  1_1020_UMTS_massimo  1_10
   7                  0.0                 0.0               0.0            0.0                0.0                   0.0                  0.0
  28                  0.0                 0.0               0.0            0.0                0.0                   0.0                  0.0
  73                  0.0                 0.0               0.0            0.0                0.0                   0.0                  0.0
 106                  0.0                 0.0               0.0            0.0                0.0                   0.0                  0.0
 220                  0.0                 0.0               0.0            0.0                0.0                   0.0                  0.0

5 rows × 900 columns
```

Figure 3.4: Data Frame imported using Pandas- Service Time

Beside network coverage, we focus also on predicting users satisfaction on the QoE of YouTube streaming sessions. We assume that the satisfaction of a user regarding video quality reported at day d is somehow correlated with RAT-dependent features (3G/4G) analysed within the previous 30 days. To give an example, it is reasonable to conjecture that a user that watched videos under 4G reports higher levels of satisfaction compared to a 3G-only user, since higher throughputs can be achieved with the former technology. Therefore, for a given user memory of length 30. We have the following features for 4G and 3G for the KPI's described above(DL Session time, Session Volume, Throughput).

- Cummulato
- massimo

Total Number of Samples: 6264
Total Number of Features: 360
For each user, 2 x 2 x 3 x 30 = 360 features, where factor 2 accounts for number of access technologies in consideration, another factor 2 for number of aggregate measurements i.e cumulato and massimo. Factor 3 corresponds to KPI's under consideration and factor 30 for memory of the prediction pipeline.



```
In [279]: dataframe_video.head()
Out[279]:
```

	1_10040_UMTS_cumulato	1_10040_UMTS_massimo	1_10040_LTE_cumulato	1_10040_LTE_massimo	1_10039_LTE_cumulato	1_10039_LTE_massimo	1_10039_
84	NaN	NaN	NaN	NaN	NaN	NaN	
137	NaN	NaN	NaN	NaN	NaN	NaN	
201	NaN	NaN	NaN	NaN	NaN	NaN	
280	NaN	NaN	31388.0	31388.0	35.579	35.579	
472	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 361 columns

Figure 3.5: Data Frame imported using Pandas - Video Streaming Data

3.5 Feature Engineering:

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process [5]. Though Feature engineering is the most important art in machine learning which creates the huge difference between a good model and a bad model, A optimal architecture of a neural network can internally come up with those combination of features.

In any way, we perform some evident feature transformations so as to increase performance of the neural networks.

3.5.1 Missing Values Treatment

In our raw dataset, there are approximately 80% of 'NaN' values. We consider one of the KPI's. We consider CQI feature columns for visualisation of presence of non-Null values.

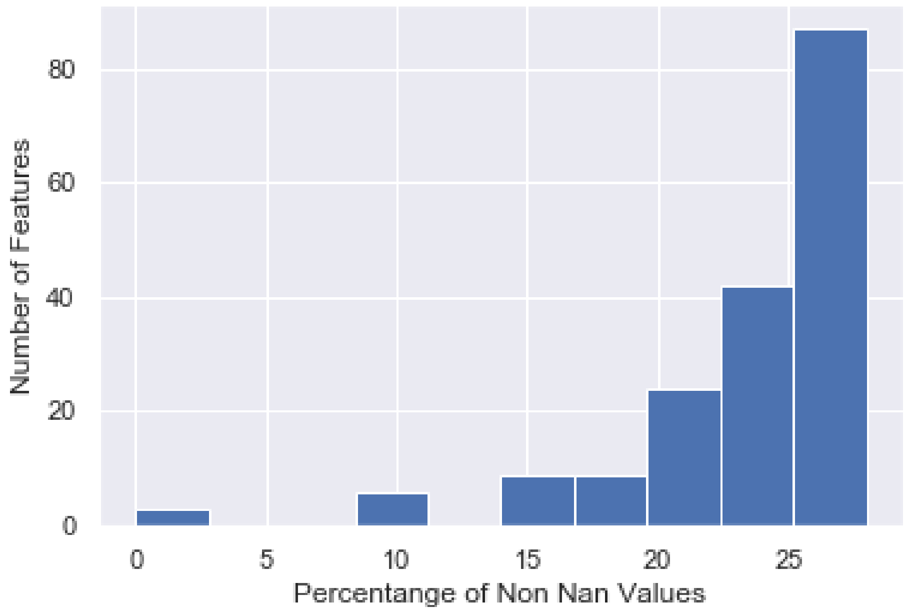
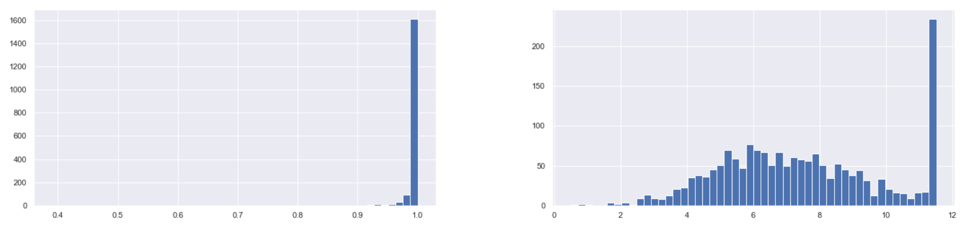


Figure 3.6: (%) of Non **NaN** Values per columns related to CQI

In Fig.3.6, we observe that only approx. 80 out of 180 features(related to CQI) corresponds to above 25% of non-null values out of 7045 samples. After deep inspection, the dataset corresponding to days 1 to 20 has most number



(a) Original (b) Log Transformation

Figure 3.7: Propability Density Function of F and Transformed F

of NaN values, Before we treat the missing values, we slice out dataset from 30 days to last 10 days. We pick the maximum feature column with non NaN values and perform missing value treatment. We drop all the rows corresponding to this feature column. In case of CQI, replacing 'NaN' with zeros or any other value is not the right choice, as the CQI is related to a network measurement we cannot infer anything from the corrupted value, where as in the case of Service times, if the user has not visited a cell generating a NaN value can be replaced with zero service time. This step reduces the dataset to 1789 samples. In the case of Video Streaming dataset, the NaN are safely replaced with zeros. In this scenario, No data reduction is done.

3.5.2 Log Transformation:

It is important to check the statistical distribution of each computed feature, as a large portion of machine learning methods assume that input features are characterised by a Gaussian distribution. We observe that the channel measurements features are already Gaussian distributed, while this is not true for the service time ratios. According to [6] Log of a variable is a common transformation method used to change the shape of distribution of the variable on a distribution plot. It is generally used for reducing right skewness of variables. Fig 3.7a represents the distribution of 'Full Service Time KPI' for LTE access type, We observe that the data is right skewed and probably a log transformation is helpful to remove the skewness. Fig

3.7b shows the transformed version of this feature. Similarly we transform all the Service time related features and insert to the dataframe.

3.5.3 Relative Service and Session Times:

We assume that computing the probabilities of service times can be effective. This information gives the model about the percentage of time the user in a cell experienced full/lim/No service times. We compute the following to find the cumulative full service time ratio(F_n)

$$F_n = \frac{\sum_{n=1}^{30} f_n}{\sum_{n=1}^{30}(f_n + l_n + z_n)}$$

Where f_n, l_n, z_n are the full, limited and No service times. Similarly, we compute the Cumulative Limited Service Time Ratio (L_n) and the Cumulative No Service Time Ratio (Z_n). These newly features are inserted to our dataframe. Similar approach is also applied on the Video Streaming data set. New feature columns are added with respect to relative download session times with respect to LTE and UMTS. Relative Session Time S_n is given by

$$S_n = \frac{\sum_{n=1}^{30} DL_{LTE}}{\sum_{n=1}^{30}(DL_{LTE} + DL_{UMTS})}$$

3.5.4 Average Throughput:

Features corresponding to download volume and time are transformed to compute the average throughput per access technology. The weighted average(THR_{avg}) is given by

$$THR_{avg} = \frac{\sum_{n=1}^{30}(THR_{LTE} * DL_{LTE}) + (THR_{UMTS} * DL_{UMTS})}{\sum_{n=1}^{30}(DL_{LTE} + DL_{UMTS})}$$

3.5.5 Discard Constant Features:

After performing all the processing steps. Standard deviations are measured in both row and columns wise. We find more number of features especially, features with respect to UMTS in video streaming data set (Q_V) are constant. We discard such columns. Users record with all zero information are also discarded. In this case, the data for Video Streaming is reduced to 1140 Samples and Features to 160.

3.5.6 Normalization:

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges. In our specific case, we have feature values corresponding to CQI, RSRQ, SNR in dBm, Service times in seconds, Session DL volume in kb, Peak Session Throuphput in kbps. Thus we normalize these feature vectors using mean and standard deviation. Consider X as the dataset, by normalization we have

$$X_{norm} = \frac{X - mean(X)}{\sqrt{}(Var(X))}$$

Chapter 4

Implementation

In this Chapter, we discuss about the improvements made to the previous work of [4]. We will propose an optimal Neural Network that is expected to give a better performance compared to other non-linear classifier. We use Keras and tensor flow python packages to build our Prediction pipeline with Neural networks.

4.1 Introduction to Neural Networks

The data set under consideration is non-linear and imbalanced. Neural Network can be used to represent convex regions. In fact, there is a theoretical finding from [7] that shows that a Neural Network with two hidden layers is sufficient for creating classification regions of any desired shape.

The multi-layer perceptron [8] is a feed forward neural network consisting of one input layer, at least one hidden layer, and one output layer. Feed forward means that data move from the input to the output layer. This type of network is trained by the back-propagation learning algorithm. MLP's are widely used for prediction, pattern classification, recognition, and estimation. MLP can resolve problems which are not separable linearly. The principal benefit of neural networks lies in an iterative learning process, in which the data set is loaded to the network one at a time, and the weights associated with the input values are changed each time.

During this learning stage, the network learns by calibrating the weights, which allows us to predict the proper outcome of input samples. Advantages of neural networks involve their adaptability to noisy data, as well as their capability to classify patterns on which they have not been trained.

In this work, we present the implementation of designing a Neural Network from scratch using tensor flow - keras API's, hyper parameter tuning for finding an optimal topology, Cross Validation techniques to train, validate and test the model and criterion for selecting the relevant evaluation metrics.

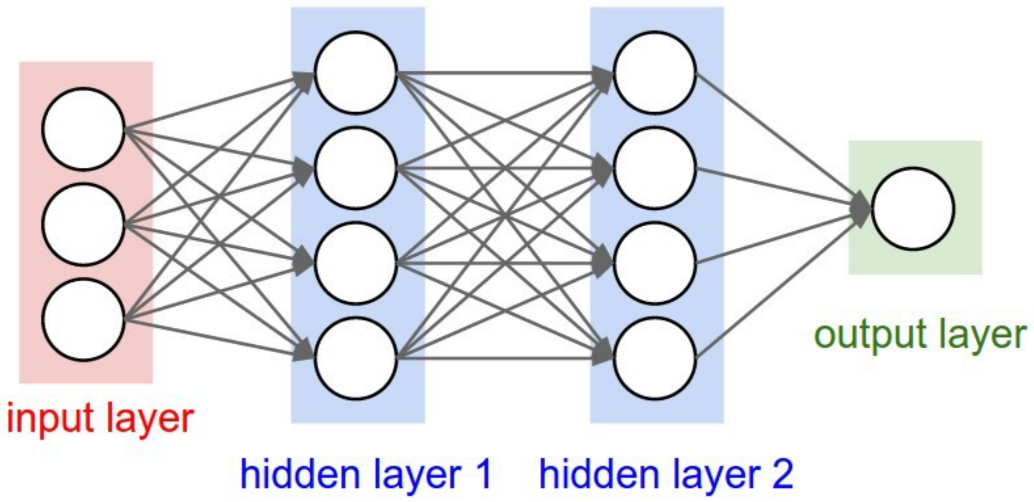


Figure 4.1: Example of a 2 Layer Neural Network

4.1.1 Building Block of Neural Network

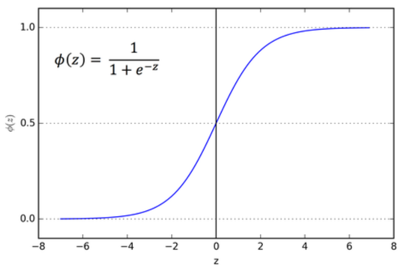
The most important block of a Neural Network that is closely related to the hypothesis space of the Neural Network classifier are Activation Functions. Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1. An additional aspect of activation functions is that they must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample. Modern neural networks use a technique called back propagation to train the model, which places an increased computational strain on the activation function, and its derivative function.

Role of Activation Functions:

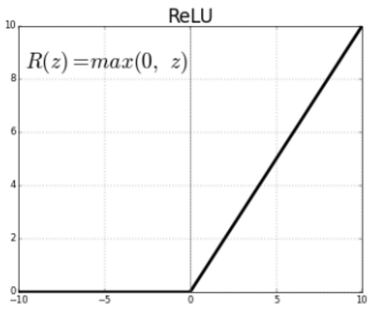
In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight, and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer. The activation function is a mathematical "gate" in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function. Increasingly, neural networks use non-linear activation functions, which can help the network learn complex data, compute and learn almost any function representing a question, and provide accurate predictions.

Non linear Activation Function:

Modern neural network models use non-linear activation functions. They allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or have high dimensional. Almost any process imaginable can be represented as a functional computation in a neural network, provided that the activation function is non-linear. Non-linear functions address the problems of a linear activation function: They allow back-propagation because they have a derivative function which is related to the inputs. They allow "stacking" of multiple layers of neurons to create a deep neural network. Multiple hidden layers of neurons are needed to learn complex data sets with high levels of accuracy.



(a) Sigmoid



(b) ReLu

Figure 4.2: Non Linear Activation Functions

Sigmoid:

Sigmoid funcion in 4.2a looks like a 'S-shape'. The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

Relu:

The ReLU in 4.2b is the most used activation function. Since, it is used in almost all the convolution neural networks or deep learning. The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. Range: [0 to infinity). The function and its derivative both are monotonic.

One of the problems with ReLu is that, negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately. LeakyRelu is an extension of ReLu to address the above mentioned problem.

4.1.2 Training a Neural Network:

The process of fine-tuning the weights and biases from the input data is known as training the Neural Network. Each iteration of the training process consists of the following steps:

- Calculating the predicted output ŷ, known as feed forward
- Updating the weights and biases, known as back propagation

Loss function estimate the prediction error and back propagate the parameters to modify the weights and biases.

Backpropagtion

This is the core step in functioning of neural network. After measuring the error of our prediction (loss), we need to find a way to propagate the error back, and to update our weights and biases. In order to know the appropriate amount to adjust the weights and biases by, we need to know the derivative of the loss function with respect to the weights and biases. Derivative of the function is simply the slope of the function. If we have the derivative, we can simply update the weights and biases by increasing/reducing with it. This is known as gradient descent. However, the derivative of the loss function can't be directly calculated, with respect to the weights and biases because the equation of the loss function does not contain the weights and biases. Therefore, we need the chain rule to help us calculate it. The process is shown in the Fig.4.3

$$Loss(y, \hat{y}) = \sum_{i=1}^{n} (y - \hat{y})^2$$

$$\frac{\partial\, Loss(y,\hat{y})}{\partial W} = \frac{\partial Loss(y,\hat{y})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial W} \qquad \text{where } z = Wx + b$$

$$= 2(y - \hat{y}) * \text{derivative of sigmoid function} * x$$

$$= 2(y - \hat{y}) * \text{z(1-z)} * x$$

Figure 4.3: Loss Function and its derivative - Chain Rule

Gradient Descent:

Gradient Descent is a process that occurs in the back-propagation phase where the goal is to continuously re-sample the gradient of the model's parameter in the opposite direction based on the weight w, updating consistently until we reach the global minimum of function J(w). Refer to Fig.4.4. The performance of the gradient descent depends on the initialization values, this algorithm do not guarantee global minimum. The keras API provides different optimizers that address this problem.

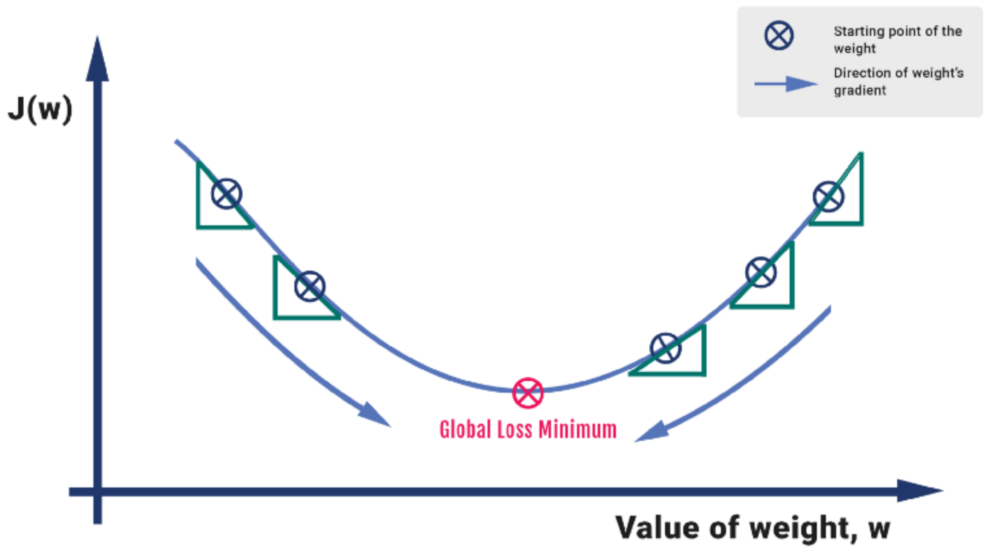


Figure 4.4: Gradient Descent

4.1.3 Hyper Parameters of a Neural Network:

A machine learning model is the definition of a mathematical formula with a number of parameters that need to be learned from the data. That is the crux of machine learning: fitting a model to the data. This is done through a process known as model training. However, there is another kind of parameters that cannot be directly learned from the regular training process. These parameters express "higher-level" properties of the model such as its complexity or how fast it should learn. They are called hyper parameters. Hyper parameters are usually fixed before the actual training process begins. Following are the hyper-parameters required to be tuned to maximize the performance of the Neural Network.

1. **Number of Layers:** It must be chosen wisely. As a very high number may introduce problems like over-fitting, vanishing and exploding gradient problems and a lower number may cause a model to have high bias and low potential.

2. **Number of Hidden Units per layer :** Choose reasonably to find a sweet spot between the high bias and variance.

3. **Activation Function:** Sigmoid/ReLu/Swish

4. **Optimizer:** It is the algorithm used by the model to update weights of every layer after every iteration. Popular choices are SGD, RMSProp and Adam. SGD works well for shallow networks but cannot escape saddle points and local minima in such cases RMSProp could be a better choice, AdaDelta/AdaGrad for sparse data whereas Adam is a general favorite and could be used to achieve faster convergence.

5. **Learning Rate:** It is responsible for the core learning characteristic and must be chosen in such a way that it is not too high wherein we are unable to converge to minima and not too low such that we are unable to speed up the learning process. Recommended values are in powers of 10, specifically 0.001, 0.01, 0.1, 1.

6. **Batch Size:** It is indicative of number of patterns shown to the network before the weight matrix is updated. If batch size is less, patterns would be less repeating and hence the weights would be all over the place and convergence would become difficult. If batch size is high learning would become slow as only after many iterations will the batch size change. It is recommend to try out batch sizes in powers of 2 (for better memory optimization) based on the data-size.

7. **Epochs:** The number of epochs is the number of times the entire training data is shown to the model. It plays an important role in how well does the model fit on the train data. High number of epochs may over-fit to the data and may have generalization problems on the test and validation set, also they could cause vanishing and exploding gradient problems. Lower number of epochs may limit the potential of the model.

8. **Drop Out Rate:** The keep-probability of the Dropout layer can be thought of hyper-parameter which could act as a regularizer to help us find the optimum bias-variance spot. It does so by removing certain connections every iteration therefore the hidden units cannot depend a

lot on any particular feature. The values it can take can be anywhere between 0–1 and it is solely based on how much is the model over-fitting

4.1.4 Loss Function:

In statistics, the mean squared error (MSE) or mean squared deviation (MSD) [9], of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and what is estimated. The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better.

The MSE is the second moment (about the origin) of the error, and thus incorporates both the variance of the estimator and its bias. For an unbiased estimator, the MSE is the variance of the estimator. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated.

$$MSE = \sum_{n=1}^{\infty} (y - y')^2$$

Where, y is the ground truth vales and y' are the predicted class.

4.1.5 Cross validation:

Cross-validation [10] is a re-sampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split. The general procedure is as follows:

- Shuffle the dataset randomly.
- Split the dataset into k groups.
- For each unique group:
  - Take the group as a hold out or test data set
  - Take the remaining groups as a training data set
  - Fit a model on the training set and evaluate it on the test set
  - Retain the evaluation score and discard the model
- Summarize the skill of the model using the sample of model evaluation scores

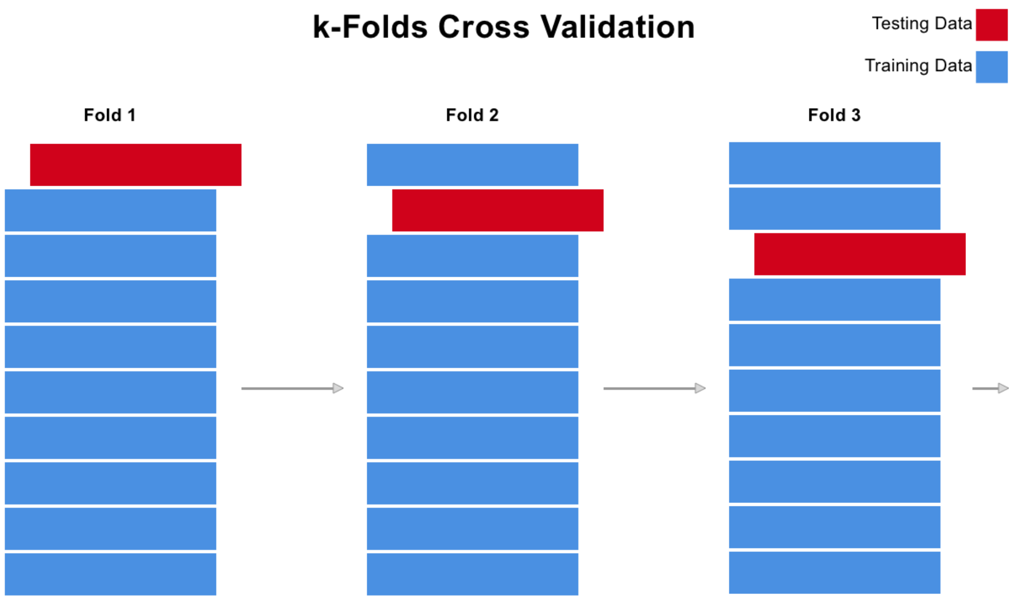


Figure 4.5: K fold Cross Validation Scheme

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

(a) Precision and Recall

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

(b) F1 Score

Figure 4.6: Evaluation Metrics

The results of a k-fold cross-validation run are often summarized with the mean of the model skill scores. Figure 4.5 shows the scheme of cross validation, for every fold the data set is split into training and test sets.

4.1.6 Evaluation Metrics:

Predictive accuracy [11], a popular choice for evaluating performance of a classifier, might not be appropriate when the data is imbalanced and/or the costs of different errors vary markedly

Reviewing both precision and recall is useful in cases where there is an imbalance in the observations between the two classes. Specifically, there are many examples of no event (class 0) and only a few examples of an event (class 1). The reason for this is that typically the large number of class 0 examples means we are less interested in the skill of the model at predicting class 0 correctly, e.g. high true negatives.

Key to the calculation of precision and recall is that the calculations do not make use of the true negatives. It is only concerned with the correct prediction of the minority class, class 1.

Precision is a ratio of the number of true positives divided by the sum of the true positives and false positives. It describes how good a model is at predicting the positive class. Precision is referred to as the positive predictive value.

Recall is calculated as the ratio of the number of true positives divided by the sum of the true positives and the false negatives. Recall is the same as sensitivity.

A composite score that attempts to summarize precision and recall is F1-score.(See 4.6a and 4.6b). In terms of model selection, F1 summarizes model skill for a specific probability threshold, whereas average precision and area under curve summarize the skill of a model across thresholds, like ROC AUC.

4.1.7 Neural Network in Practice

Keras [12] is a high-level API to build and train deep learning models. It's used for fast prototyping, advanced research, and production with three key advantages:

- *User friendly*
  Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
- *Modular and composable*
  Keras models are made by connecting configurable building blocks together, with few restrictions.
- *Easy to extend*
  Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models.

Sequential Model:

In Keras [13], its about assembling layers to build models. A model is (usually) a graph of layers. The most common type of model is a stack of layers: the tf.keras.Sequential model. To build a simple, fully-connected network (i.e. multi-layer perceptron), we use the following piece of python code.

```python
import tensorflow as tf
def keras_sequential_NN_model():
    model = Sequential()
    # Layer 1
    model.add(Dense(512, input_dim = input_features.shape[1]))
    model.add(Activation('relu'))
    model.add(Dropout(0.06))
    # Layer 2
    model.add(Dense(512))
    model.add(Activation('sigmoid'))
    model.add(Dropout(0.017))
    # Output Layer
    model.add(Dense(1))
    model.add(Activation('sigmoid'))
    # lr is learning rate
    adam = tf.keras.optimizers.Adam(lr=0.001)

    model.compile(optimizer='adam',
                  loss=tf.keras.losses.mean_squared_error,
                  metrics=['accuracy'])
    return model
```

Hyper-parameters Tuning Using Bayesian Optimization:

Hyper-parameters with the lowest validation loss is always a mundane task. Bayesian optimization is a probabilistic model based approach for finding the minimum of any function that returns a real-value metric. Bayesian optimization is a probabilistic model based approach for finding the minimum of any function that returns a real-value metric. This function may be as simple as $f(x) = x^2$, or it can be as complex as the validation error of a deep neural network with respect to hundreds of model architecture and hyper-parameter choices. Results from [14] suggest Bayesian hyper-parameter optimization of machine learning models is more efficient than manual, random, or grid search with:

- Better overall performance on the test set
- Less time required for optimization

Bayesian model-based optimization is intuitive: choose the next input values to evaluate based on the past results to concentrate the search on more promising values. The end outcome is a reduction in the total number of search iterations compared to uninformed random or grid search methods.

Bayesian Optimization with Hyperopt:

Hyperopt [15] provides algorithms and software infrastructure for carrying out hyper-parameter optimization for machine learning algorithms. Hyperopt provides an optimization interface that distinguishes a configuration space and an evaluation function that assigns real-valued loss values to points within the configuration space. Unlike the standard minimization interfaces provided by scientific programming libraries, Hyperopt's fmin interface requires users to specify the configuration space as a probability distribution. Specifying a probability distribution rather than just bounds and hard constraints allows domain experts to encode more of their intuitions regarding which values are plausible for various hyper parameters.

Like SciPy's optimize.minimize interface, Hyperopt makes the SMBO algorithm itself an interchangeable component so that any search algorithm can be applied to any search problem. We use algorithms Tree-of-Parzen-Estimators (TPE) algorithm introduced in [16]. Formulating an optimization problem in Hyperopt requires four parts:

- **Objective Function:** The objective function can be any function that returns a real value that we want to minimize. In Hyperopt, the objective function can take in any number of inputs but must return a single loss to minimize. We try to minimize (1-f1Score) as we target to find best model corresponding to best F1 Measure.

- **Domain Space:** The domain space is the input values over which we want to search. As a first try, we can use a uniform distribution over the range that our function is defined.

- **Optimization Algorithm:** We are using the Tree-structured Parzen Estimator model, and we can have Hyperopt configure it for us using the suggest method.

- **Trails:** This object saves the objective function loss , also supports saving extra information alongside the trial loss.

Find below the python usage of the hyperopt.

```python
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials
trials = Trials()
best_run = fmin(NN_model, search_space, algo=tpe.suggest,
    max_evals=35, trials=trials)
```

4.2 Prediction Pipeline

Neural classifier requires to optimize the hidden layer structure, by tuning the number of neurons. To tune such hyper-parameters, we proceed with a Bayesian optimized search on a set of candidate values as follows. First, according to a Stratified k-fold cross-validation strategy with k = 5, the original data set of 1780 observations and ground truth pairs is divided into five folds with splitting ratios 80% (Training set) and a 20% (Test set). Secondly, we focus on a given pair of Training Set and Test Set. We apply to the Training Set a further 10-fold cross-validation, such that it is divided into five folds with splitting ratios 80% (Sub-Training Set) and 20% (Validation Set). Each Sub-Training Set is then trained with each classifier's hyper-parameters candidate values.

Prediction performances are then evaluated on the corresponding Validation Set. At end of this (inner) cross-validation process, we can select the classifiers' best hyper-parameters (i.e. those maximising, per each classifier, the prediction results on the Validation Set) that are used to train each model in the outer cross-validation loop (i.e. the original Training Set). Finally, the trained models prediction performances are tested on the Test Set. Note that this procedure is repeated 5 times, one per each Training Set selected by the outer cross-validation loop. The results we will show correspond to the average results across the different Test Sets. In particular, for each observation of a given Test Set, the tested classifiers output the probability that the observation belongs to the Unsatisfied class.

In our case, We set the objective function of Hyperopt as '1-f1score' and minimize on it, there by finding a optimal search space with respect to the above loss for every model under evaluation. We evaluate 50 models per outer fold and for each model 10-Stratified fold f1-score is returned.

The Hyperopt selects the best performing model parameters out of 50 evaluated models and returns the corresponding parameters. The algorithm builds the model with best performing parameters and evaluate on Test Set. The model returns prediction class of probabilities. The algorithm, thresholds to 100 levels between 0 and 1 with step size of 0.01. For every level, Confusion matrix is calculated. True Positive rate, False positive rate, Precision and recall scores are recorded. Algorithm selects the best threshold ,that gives the maximum F1 score. The Receiver Operating Curve is also plotted and corresponding ROC-AUC is calculated. This process is repeated 5 times, one for each outer-fold.

4.2.1 Algorithm

Initialize: Normalize the data set, drop all the columns that has very low threshold of standard deviation and define search space for hyper parameters of our model.

1. Split the data using a stratified K fold split (n_spilts=5)
2. For every outer k fold split.
    - For every selection of hyper-parameters from search space using Bayesian optimization.
        - With max_evals=50
            - Initialize the model
            - Perform 10 fold Cross-validation
            - Evaluate F1 Score
            - Minimize 1-F1Score
        - Return the Best Set of Hyper-parameters from the 50 evaluated models.
    - Initialize a NN model using parameters from above step.
    - Fit the model to training set and predict on the test set.
    - Evaluate the results using F1 score and ROC-AUC

3. Store the Results

4.2.2 Summary:

The prediction pipeline returns 5 best performing models one for each outer-fold. Hyper parameters such as number of units per each layer, activation function, drop-out rate, learning rate, optimizer function and evaluation metrics like maximum F1 score, Average precision and recall scores, AUC(recall vs precision) and ROC-AUC in a .csv file. The end user can pick up an average model out of 5 proposed models or perform majority voting or pick a model with best F1 score or ROC-AUC or based on any evaluation metric under consideration.

Chapter 5

Results

The features computed in Section 3 are fed into the prediction pipeline proposed in Section 4. We compute True Positive Rate(TPR) and False Positive Rate(FPR). The TPR is defined as the fraction of correctly detected Unsatisfied users, while the FPR is the fraction of Satisfied users which are incorrectly labeled as Unsatisfied. Additionally, to summarise in a single value the performances of each classifier, the Area Under the Curve (AUC) is computed. Note that, for a random classifier, the AUC equals 0.5.

ROC curves help understand the performance of binary-classification mod- els at all classification thresholds and show the different false positive rates (FPRs) and true positive rates (TPRs).The ROC is for many different levels of thresholds and thus it has many F score values. F1 score is applicable for any particular point on the ROC curve. In the case of Unbalanced data set F1 captures the class 1 label predictions as it depends on precision and recall scores.

Before we present the results using proposed prediction pipeline. Feature Selection process was implemented using principal component analysis(PCA) [17] . In the case of Network coverage out of 480 features , 95 features capture the 99.99% of the variances in the data set. See fig 5.1.

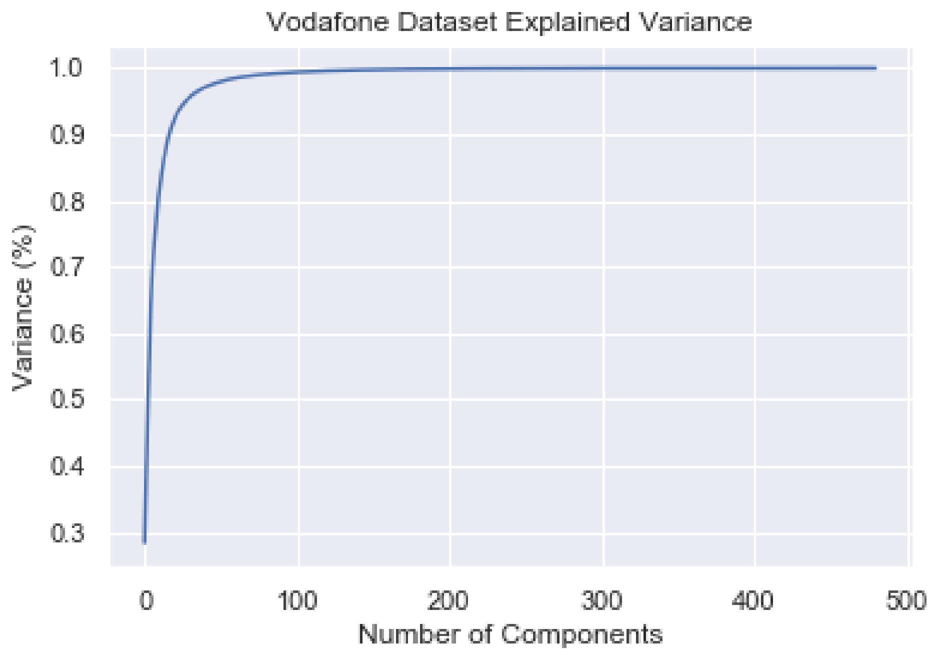


Figure 5.1: PCA on Network Coverage Data Set

We note that due to this feature reduction , there is drop in the F1 measure and ROC AUC in our experiments. So, we avoid this feature selection process and proceed to feed the entire data set to our proposed production pipeline.

5.1 Prediction results on Network Coverage:

We perform Grid search on the number of layers per model and Bayesian optimization on other hyper-parameters of the neural network. Below are the results of predictions made using 2-layered, 3-layered and 4-layered Neural Network model.

5.1.1 Observations

A total of 481 features with 1780 samples are fed into the 2 layered NN model. We plot best performing models and random classifier. We consider features related to LTE only. UMTS features are found to have the weakest correlations with the target variable.

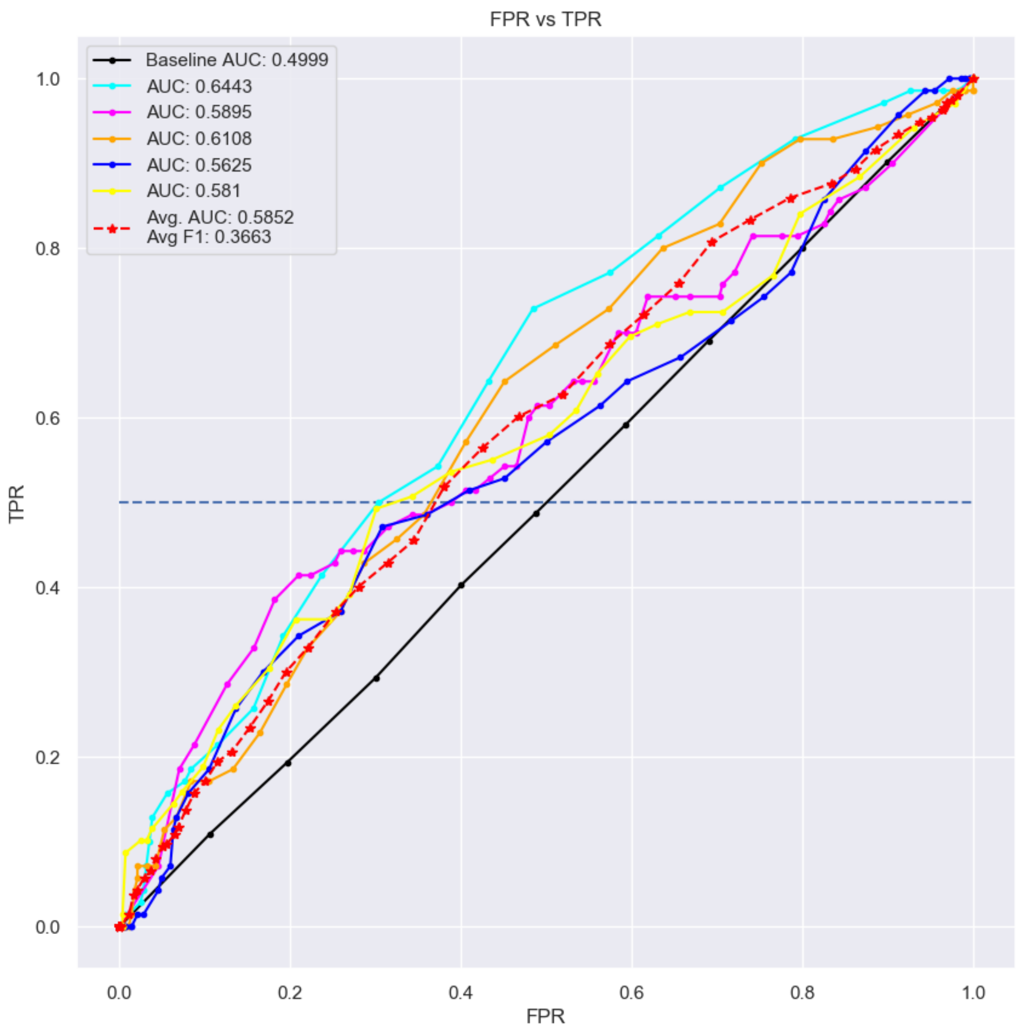
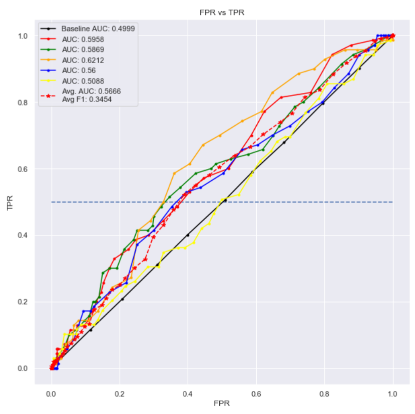


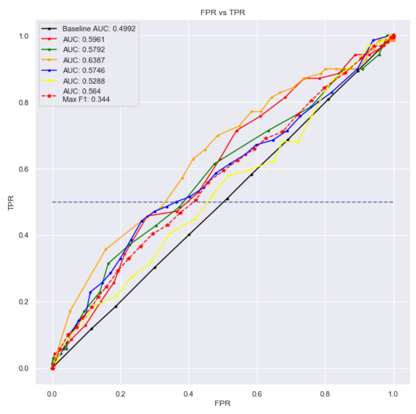
Figure 5.2: ROC-AUC for 2 Layered NN - LTE Only

In Fig.5.2 the red curve represents the performance of the average model. The Curves shown here are best performing model for each of the outer 5-fold split of data set. The points on the curve are True Positive Rate And False Positive rate computed by threshold the predicted probabilities between level 0 to 1 with step size of 1/100.

We observe that the curves are well above the random classifier with



(a) 3 Hidden Layers



(b) 4 Hidden Layers

Figure 5.3: ROC-AUC for 3 and 4 Layered NN - LTE Only

Average AUC of **58.5%** and Max F1 score of a working point on the average curve **36.62%**.

Similar results are observed for 3 and 4 hidden layered NN model. Refer to Fig.5.3a and 5.3b

5.1.2 Benchmark Results:

From [4] Various classifiers were used out of which the Random forest classifier was the best performing for the data set under observation. We take the performance of the random classifier as the benchmark and plot our results from our current work for better visualization.Results are shown in Fig. 5.4. The benchmark F1 score is **33%**.

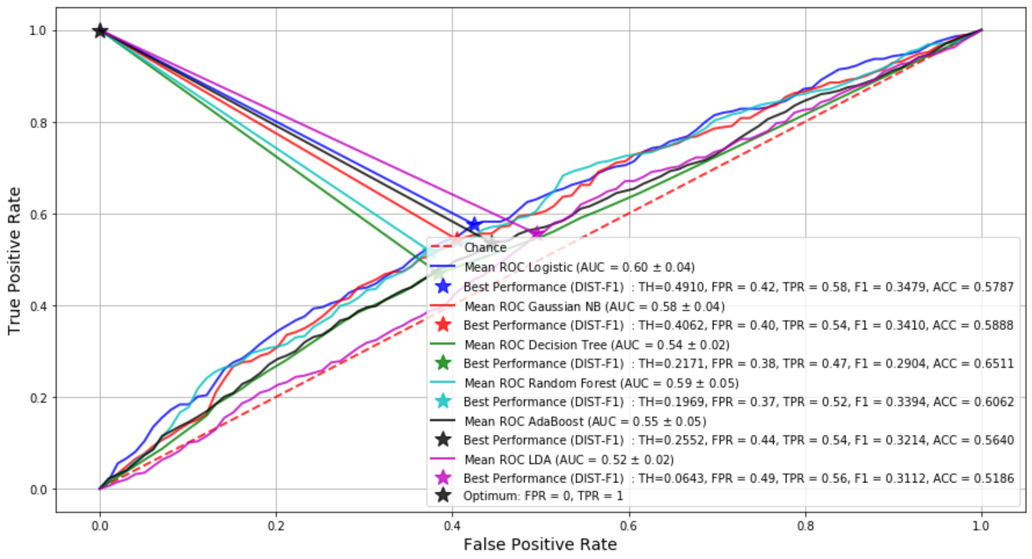
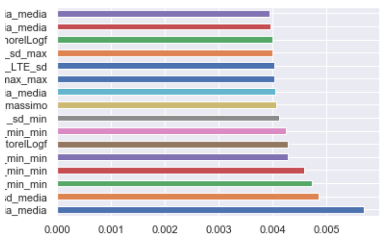


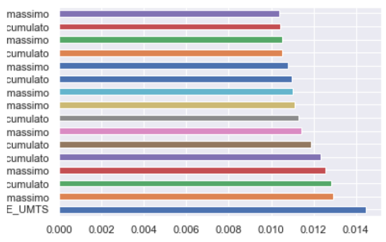
Figure 5.4: Bench marked Results Network Coverage

5.1.3 Consolidated Result

From Random Forest Classifier was the best performing one with F1 measure of **0.33** and AUC of **0.59**. In case of AUC there are no considerable improvements but in the case of F1 measure we have a improvement of approx **1%**. That means Neural Network classifiers are better in predicting the class 1 labels. Fig.5.6 shows the average performances of all the experiments and compared with the previous work (i.e Blue Curve). The performance of the Neural Network Model is also affected by the weak correlations with respect to the labels. In Fig.5.5a and 5.5b we show the best 10 features with high Pearson correlation coefficients, We observe low cross correlation scores on the engineered features as well.



(a) Relative to Network Coverage



(b) Relative to Video Quality

Figure 5.5: Pearson Correlation Scores

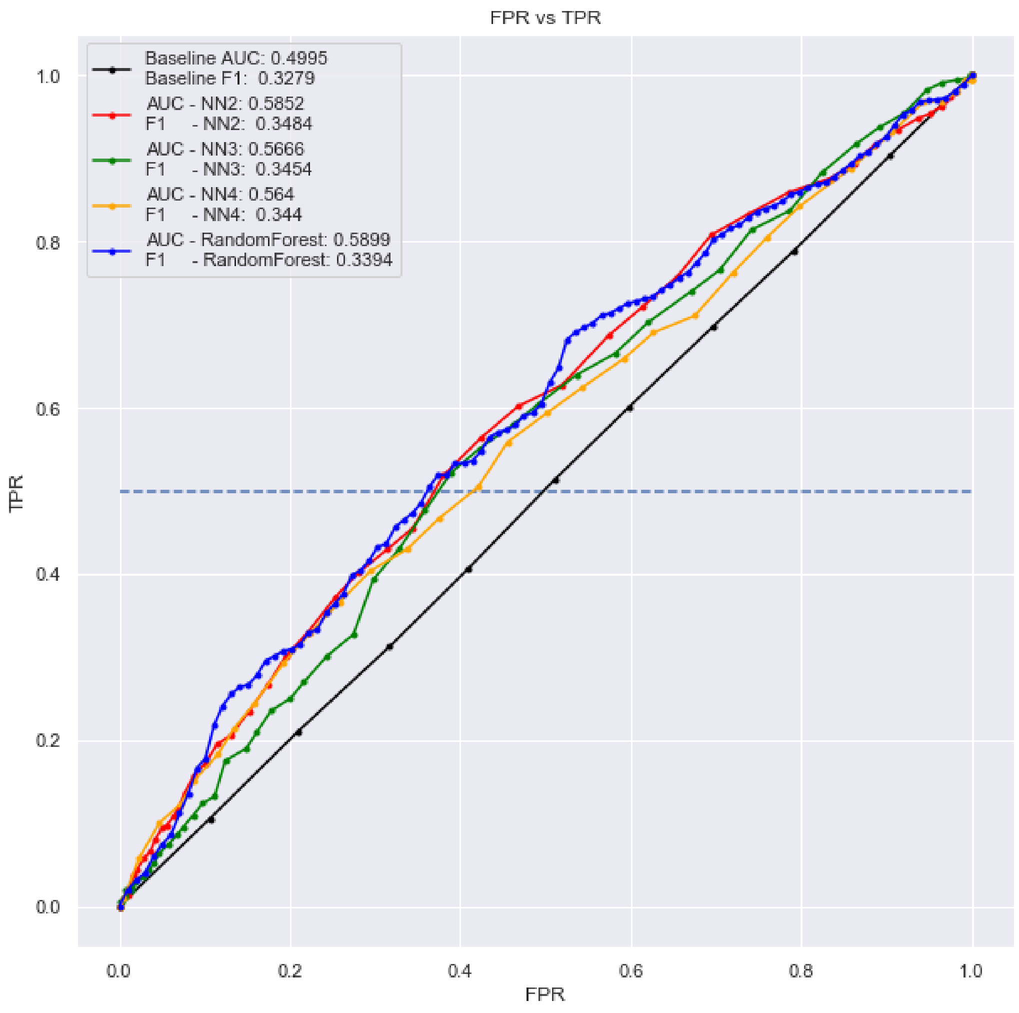


Figure 5.6: Performance Curve Past work Vs Present Work - Network Coverage

From 5.6 we observe that stacking more number of layers worsens the F1 score, this might be due to presence of less samples and at higher layer topology are prone to over fitting. In Fig. 5.7 for less number of samples there is over fitting problem, As the number of samples in the training increases, the train and test error tends to converge at higher loss.

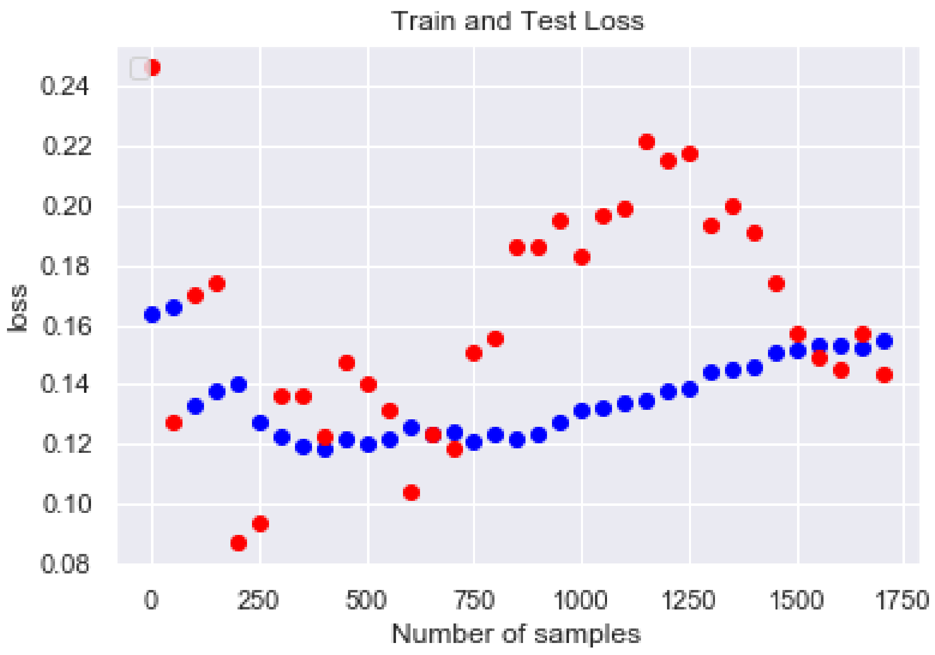


Figure 5.7: Learning Curve - Network Coverage

5.2 Prediction results for Video Streaming

Similar steps are applied on Video Streaming data set. We perform Grid search on the number of layers per model and Bayesian optimization on other hyper-parameters of the neural network. Below are the results of predictions made using 2-layered, 3-layered and 4-layered Neural Network model.

5.2.1 Observations

A total of 160 features with 1140 samples are fed into the 2 layered Neural Network model. We plot best performing models with respect to random classifier.

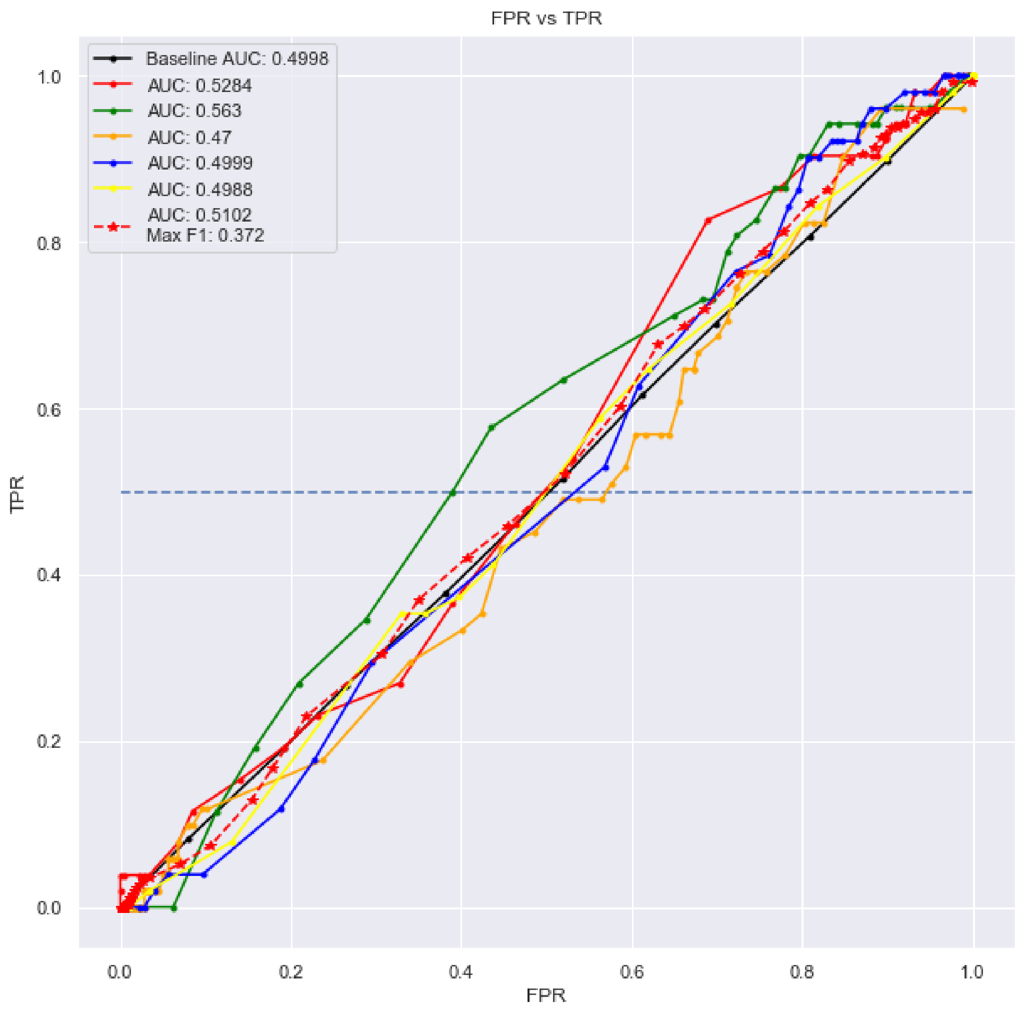
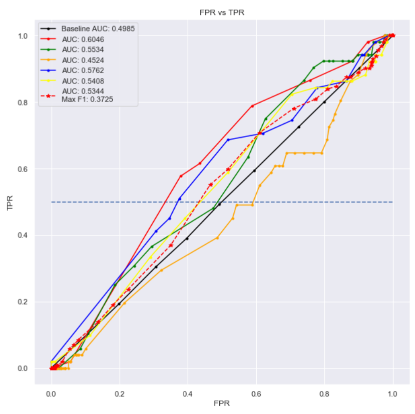
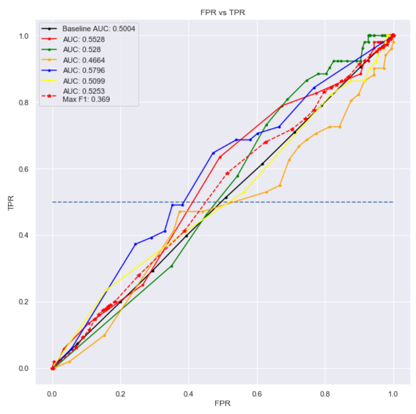


Figure 5.8: ROC-AUC for 2 Layered NN - Video Quality

In Fig.5.8, the 2 layer network shows poor results, The final 5 models are not stable and we do not notice reasonably decent performances. Two of the proposed best models falls below the random classifier. Recorded Average AUC of **0.51** and Best Working point F1 score is **0.3724**



(a) 3 Hidden Layers



(b) 4 Hidden Layers

Figure 5.9: ROC-AUC for 3 and 4 Layered NN

In the case of stacking more layers See Fig.5.9a and 5.9b, the stability in the performance model is relatively better, there is a slight increase in the overall AUC , but no improvements in the F1 score. It is observed that F1 score is slightly degraded. Average Curve AUC is **0.534** and **0.521** for 3 and 4 layers respectively.

Unlike the case of Network data set, we have more features to fit the model with lesser number of samples compared to previous case. The Learning curve experiments states that, the model is unstable and suffers with over fitting.Refer to figure 5.8.

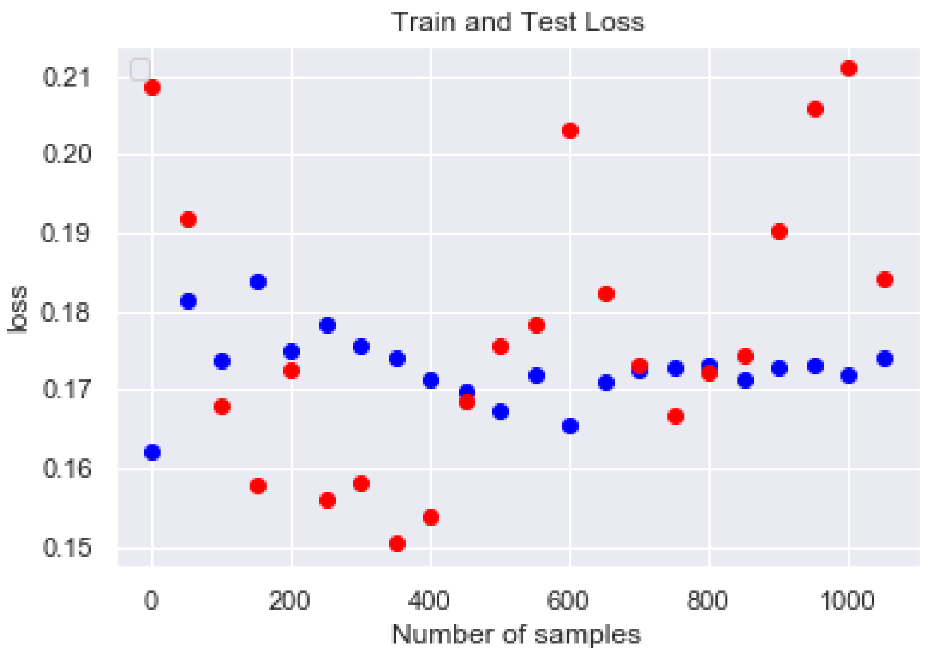


Figure 5.10: Learning Curve - Video Quality

5.2.2 Benchmark Results:

From[4] various classifiers were used. Out of all, the Random forest classifier was the best performing for the data set under observation. We take the performance of the random classifier as the benchmark and plot our results from our current work for better visualization.Results are shown in Fig 5.11. The benchmark AUC is **58%** and F1 score is **37%**.

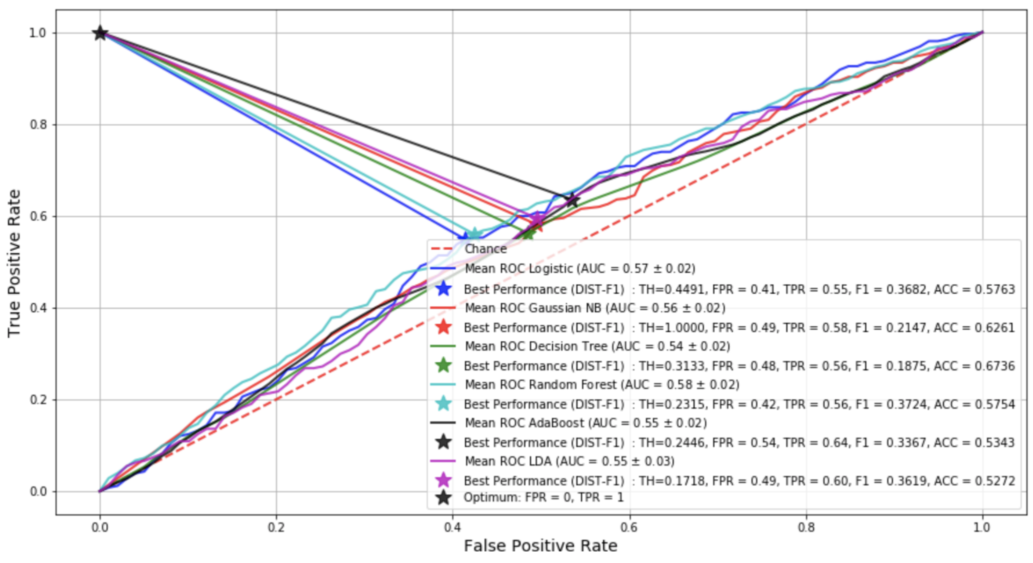


Figure 5.11: Bench marked Results Video Quality

5.2.3 Consolidated Results:

The results in Fig.5.12 shows that, In this particular scenario, the Neural Networks do not perform better than the bench marked classifiers. We do not record better performances. We state that, the possible reasons for these results are due to, weakly correlated features, Less number of samples to train the model.

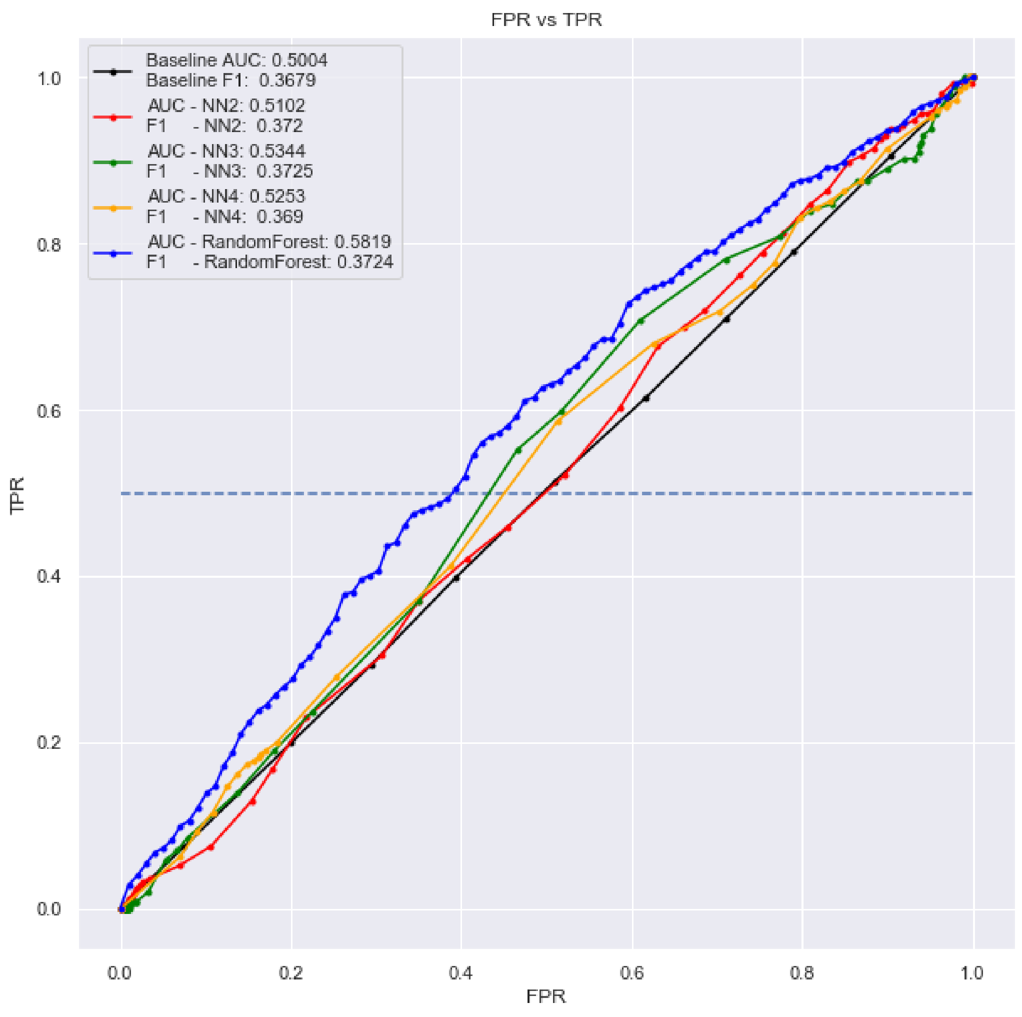


Figure 5.12: Performance Curve Past work Vs Present Work - Video Quality

Chapter 6

Conclusions and Future Work

In this work we have commented on the possibility of predicting the long-term coverage and video satisfaction starting from user-side network measurements. The results obtained demonstrate that the task is complex and challenging, as the most robust machine learning classifiers such as Neural Networks show quite poor performances.Nonetheless, a weak correlation between the engineered input features and user satisfaction feed backs could be exploited and can be used from the operator as a starting point to identify possible problems in the network. Some interesting points can be raised which might improve the performances

- Compared to short-term QoE estimation, long-term satisfaction prediction looks like a much more challenging task. The most direct explanation for this could lie in the way users reply to survey, which could be affected by many factors (e.g., value for money or other user-dependent standards) that network measurements alone cannot capture.

- Attracting more customers to provide feedback, implementation of incentive based approaches.

- A very less number of samples compared to number of features. Training a neural network causes over fitting. We observe that the developed Neural Networks performs better on the training set and not as good as on the test set.

- Using LSTM solutions as stated in[18], for long term prediction. This choice also decrease the computational complexity.
- We must investigate other solutions to tackle the problem of having a imbalanced smaller data set. Under Sampling and Over Sampling the data could one of the choices.
- Use more granular search space for hyper-parameter tuning ,investigate more on non linear activation functions and loss functions. Both feature engineering and Optimal Machine learning model are linked to get better performances.

Bibliography

[1] Xiaoming Tao et al. «Learning QoE of mobile video transmission with deep neural network: A data-driven approach». In: *IEEE Journal on Selected Areas in Communications* 37.6 (2019), pp. 1337–1348.

[2] Sarah Wassermann et al. «On the Analysis of YouTube QoE in Cellular Networks through in-Smartphone Measurements». In: 2019.

[3] Asma Ben Letaifa. «WBQoEMS: Web browsing QoE monitoring system based on prediction algorithms». In: *International Journal of Communication Systems* (2019), e4007.

[4] Alessandro Redondi Andrea Pimpinella. «Towards Long-Term Coverage and Video Users Satisfaction Prediction in Cellular Networks». In: *2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE. 2019.

[5] Ian H Witten et al. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[6] FENG Changyong et al. «Log-transformation and its implications for data analysis». In: *Shanghai archives of psychiatry* 26.2 (2014), p. 105.

[7] R. Lippmann. «An introduction to computing with neural nets». In: *IEEE ASSP Magazine* 4.2 (1987), pp. 4–22. ISSN: 0740-7467. DOI: 10.1109/MASSP.1987.1165576.

[8] Jaroslav Frnda et al. «A Hybrid QoS-QoE Estimation System for IPTV Service». In: *Electronics* 8.5 (2019), p. 585.

[9] Zhou Wang and Alan C Bovik. «Mean squared error: Love it or leave it? A new look at signal fidelity measures». In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[10] NA Diamantidis, Dimitris Karlis, and Emmanouel A Giakoumakis. «Unsupervised stratification of cross-validation for accuracy estimation». In: *Artificial Intelligence* 116.1-2 (2000), pp. 1–16.

[11] Nitesh V Chawla. «Data mining for imbalanced datasets: An overview». In: *Data mining and knowledge discovery handbook.* Springer, 2009, pp. 875–886.

[12] Keith Johnson. «Supervised Learning for Sequence Prediction Using Keras Sequential Models». PhD thesis. California State University, Northridge, 2019.

[13] Yeldar Toleubay and Alex Pappachen James. «Getting Started with TensorFlow Deep Learning». In: *Deep Learning Classifiers with Memristive Networks.* Springer, 2020, pp. 57–67.

[14] James Bergstra, Daniel Yamins, and David Daniel Cox. «Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures». In: (2013).

[15] James Bergstra, Dan Yamins, and David D Cox. «Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms». In: *Proceedings of the 12th Python in science conference.* Citeseer. 2013, pp. 13–20.

[16] James S Bergstra et al. «Algorithms for hyper-parameter optimization». In: *Advances in neural information processing systems.* 2011, pp. 2546–2554.

[17] Ian Jolliffe. *Principal component analysis.* Springer, 2011.

[18] Yuxiu Hua et al. «Deep Learning with Long Short-Term Memory for Time Series Prediction». In: *IEEE Communications Magazine* (2019).