Ensemble Methods for Image Quality Assessment using Individual or Group Ratings

Master Thesis
presented

by
Ali Asghar Marvi

at the

Universität Konstanz

Workgroup Multimedia Signal Processing
Department of Computer and Information Science

1. Evaluated by	Prof. Dr. Dietmar Saupe
2. Evaluated by	Prof. Dr. Bastian Goldlücke
3. Advised by	Dr. Vlad Hosu

Konstanz, 2023

Ali Asghar Marvi

Ensemble Methods for Image Quality Assessment using Individual or Group Ratings

Acknowledgements

I would first like to thank my thesis advisor Dr. Vlad Hosu and my thesis supervisors Prof. Dr. Dietmar Saupe and Prof. Dr. Bastian Goldlücke of the faculty of Computer Science at the University of Konstanz. My advisor Dr. Vlad ensured continuous assistance from his side in giving access to CCU to smoothly conduct experiments for this thesis. He also steered the direction of my thesis through his critical feedback and analysis.

My undergraduate degree was from a business school. As a result, I was not geared up to carry out academic research. Thanks to Prof. Dr. Dietmar Saupe, who is also the primary supervisor of my thesis, for guiding me through the formalities of carrying out this academic research. This thesis has been an excellent opportunity to equip me with research in academia and industry.

Lastly, I would like to especially mention my family, colleagues & friends for being my constant support and source of encouragement throughout this period. I can proudly say that I have had the privilege to work with some of the brightest minds out there. It was their encouragement during my master's that kept me undeterred in researching and writing my thesis.

Author

Ali Asghar Marvi

Abstract

In this thesis, I implemented the known ensemble learning methods on the KonIQ-10k dataset for Image Quality Assessment (IQA). The main idea here is to use ensemble methods like "Bagging", "Stacking", and "Boosting" on ratings from individual raters and groups of raters. These methods will be compared with the baseline models used within the implemented ensemble methods. The ensemble method consists of basic and less complicated models for each strategy, either individuals or groups. The base model for bagging is a Tensorflow implementation of Multi-Layer Perceptron (MLP). The base model used for the "Boosting" method is the Light Gradient Boosting Machine (LGBM), a memory-efficient and current state-of-the-art Gradient Boosting algorithm. For stacking, along with MLP, Light Gradient Boosting Machines (LGBM) and K-Nearest Neighbors (kNN) models are used to learn patterns. Linear regression is used as the meta-learner for the “stacking” method. For this thesis, the aforementioned ensemble methods are used to learn user rating patterns. The input features for the models in ensemble methods are the Multi-layered Spatially Pooled (MLSP) features. These features are extracted from the convolutional blocks of InceptionResNet architecture. The mean opinion scores (MOS) of predictions for test images are compared with the ground-truth MOS for analysis. The baseline stacking method with base models trained on all the data had the best performance with SRCC and PLCC values of **0.916** and **0.926**, respectively. The ensemble of worker clusters performed better compared to the ensemble of models of individual raters with SRCC and PLCC scores of **0.916** and **0.924**, respectively.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Image Quality Assessment (IQA) is an important area of study that has a broad range of applications in the domain of Image Processing [KS05], Computer Vision [CYKL21], Image Retrieval [YLKT14] and many more. Images are often distorted when these are collected and processed or are affected due to the quality of the camera. IQA is a technique to predict the perceptual quality of the image. Ideally, this perceptual quality of the image is assessed by conducting a user study by a group of experts who rate images on a discrete scale. Usually, this scale is in the range from 1 to 5, but this scale can also be on different ranges. These ratings are then averaged to obtain the Mean Opinion Score (MOS) (1) where R_n is the nth rating for an image and N is the total number of ratings received. This MOS is related to the ground truth but is not the same. The ground truth of an image's quality is typically obtained from the subjective assessment of images. The ratings received become the benchmark of the perceived image quality. This MOS, or the average of the ratings received, estimates the image's overall perceived quality, which is close to the ground truth. MOS plays a crucial role in any visual analysis task. The subjective experiments involving user groups can be expensive, and prone to significant errors caused by multiple factors acting on them, such as environmental, psychological, and others. Therefore, the ratings obtained are subject to higher variations and can produce MOS with significant deviations unless the number of raters is large.

$$\boldsymbol{MOS = \frac{\sum_{n=1}^{N} R_n}{N}} \tag{1}$$

IQA approaches are divided into two categories, objective and subjective IQA. Subjective IQA, as mentioned above, involves approaches in collecting ratings from human participants in either laboratory environments [CAY+20] or conducting crowd-sourcing experiments online [SHH+16]. These produce MOS, which becomes the estimate of the ground truth image quality. There can be a variety of subjective studies concerning various factors like environmental, demographic, psychological, and physical. Multiple benchmark datasets have been produced resulting from such subjective quality experiments. Some of these include, KonIQ-10k [HLSS20], LIVE in the Wild [GB16], LIVE [SSB06a], TID2013 [PJI+15]. These data sets mainly consist of images with specific resolutions and their corresponding MOS scores. Several standards for subjective quality assessment have been proposed for assessing image quality in a reliable way from human subjects. These are as follows:

- **ITU-R BT.500-11** [Cha95] - This standard proposes various methods for subjective quality assessment of television pictures. It is the most commonly used standard which contains information about viewing stimuli, instruction on carrying out experiments and presenting the results.
- **ITU-T P.910** [CJC$^+$07] - This standard proposes to assess digital video quality with a transmission rate below 1.5 Mbits/sec.
- **ITU-R BT.814-1** [BT] - This standard recommends setting a predefined brightness and contrast level on the display devices for viewing images or videos.
- **ITU-R BT.1129-2** [VQE] - This standard proposes a standard definition (SD) for assessing the quality of digital streams on television.
- **ITU-T P.913** [PJ14] - This standard proposes a method to assess the overall audiovisual quality of videos played on various multimedia devices. This method also incorporates the surrounding environment that can influence the overall evaluation of the video and audio.

Contrary to subjective IQA, objective IQA involves applying mathematical and computational methods to estimate the perceived quality of the image. This can involve complex machine-learning tasks or simple image-processing applications. Objective IQA can be classified into three categories:

- **FR-IQA** - Full-Reference image quality assessment where the quality of images, mainly distorted images, are assessed with their original reference image. It is a quantifiable method to measure the perceived differences between distorted and referenced images.
- **NR-IQA** - No-Reference image quality assessment where images' quality is assessed without an actual reference image. This can include utilizing numerical measures to assess distortions or any other type of noise in the image. NR-IQA is also called blind image quality assessment (BIQA). It is still a harder problem to solve and a challenging research area.
- **RR-IQA** - Reduced-Reference image quality assessment is somewhere between NR-IQA and FR-IQA. It uses a reduced set of features from reference images compared to noisy images. Therefore reduced reference methods are less susceptible to any other type of distortions that can affect other important image contents.

With time multiple standards and methods have been proposed to advance both subjective and objective image quality [WBSS04], [SSB06b], [WBL02] but creating a generalized version for all types of images remains an open challenge to date. In a similar approach to modeling user ratings, this thesis applies ensemble learning to learn a user or group of users' rating patterns on the KonIQ-10k dataset. Ensemble learning is a machine learning approach that combines predictions from multiple models to make final predictions. The ensemble methods, bagging, boosting, and stacking, are implemented using simpler base learners. The bagging approach utilizes Multi-Layer Perceptron (MLP) models as base learners. Predictions made on images from the test set by base learners in the bagging method are aggregated to produce final predictions. This is the predicted MOS evaluated against the MOS of images from the test set. The boosting approach is implemented by using the Light Gradient Boosting Machine (LGBM) [KMF$^+$17a] algorithm that uses the gradient boosting algorithm as recommended by Friedman et al. [Fri02]. Stacking is implemented using three diverse models: Light Gradient Boosting Machine (LGBM) [KMF$^+$17a], K-Nearest Neighbors, and MLP. The overall performance of these ensembles (both clusters of workers and single user ratings) is assessed against the performance of the models used as base learners. This means that the performance of the bagging method is compared with MLP used as the base learner in bagging, but it is trained on all the screened images from the KonIQ-10k data set. The performance of Boosting method is measured against the performance of the single LGBM model trained on the screened images and its ratings from the KonIQ-10k data set. For the Stacking method, the baseline stacking structure is the same as used to train on the subset of ratings, but it is trained on all the screened images and their ratings from the KonIQ-10k data set.

CHAPTER 2

Background

Multiple associations need to be discussed before diving deep into the explanations of experiments. For instance, the data set used, the features engineered, and the relevant research literature which conducted experiments on similar grounds as proposed in this thesis. In the first section of this chapter, I have mentioned the background of KonIQ-10k and its importance in IQA research. I also wrote about its use in Blind Image Quality Assessment (BIQA) and evaluation of the SRCC and PLCC scores obtained by testing on other IQA data sets. With time, multiple contributions have been made in introducing diverse data sets and respective objective IQA methods. These objective methods are introduced to improve the objective assessment of the images. Its assessment performance is generally measured by calculating SRCC and PLCC scores. In another section, the approach behind extracting features from KonIQ-10k images is discussed in detail. It briefly explains the feature engineering technique used in the AVA data set [MMP12]. The same approach was used to extract features from KonIQ-10k images. In the last section, relevant ensemble-based methods in IQA are discussed along with Spearman Rank-order Correlation Coefficient (SRCC) and Pearson Linear Correlation Coefficient (PLCC) scores. These scores are the common evaluation metrics to assess the performance of objective IQA methods.

2.1 KonIQ-10k

In 2020, the work by Hosu et al. [HLSS20] introduced an ecologically valid data set for IQA. It is considered to be one of the significant contributions in this area. The paper described a benchmark data set for the improved aesthetic evaluation of images and image quality prediction in Computer Vision. This IQA model was evaluated on the LIVE-in-the-Wild data set [GB16] and KonIQ-10k dataset. The IQA model, named *KonCept* trained on KonIQ-10k, was evaluated on the test set from KonIQ-10k (SROCC score of **0.921**) and on LIVE-in-the-Wild (LIVE-itw) with SROCC of **0.825**. Details about the proposed model are explained towards the end of this section.

The purpose behind introducing the data set was that the known IQA data sets until then were very much limited in scope and did not consider the diversity of real-world images. Conventional assessment methods involved artificially degrading the quality of images by introducing multiple types of distortions. Then a group of participants would be asked to assess the quality of the images. There are two fundamental drawbacks to this approach:

- The diversity of the image content is limited since all the distortions introduced come from the small set of pristine images.
- The combination of introduced distortions in the images is very limited. These distortions are not ecologically valid compared to, for example, any image viewed on the Internet.

To address this problem, the authors introduced an ecologically valid data set with reliable quality ratings. This data set consists of 10,073 images of the resolution **1024 × 768** pixels. 1459 workers rated these images in a crowd-sourcing experiment. Each user rated a different subset of the images producing 1.2 million ratings in total. This human data annotation had image ratings on a scale from 1 to 5. The ratings were then used to produce the quality MOS of the stimuli.

The data set creation was achieved by selecting images from a massive public data set called YFCC100m [TSF+16]. The initial data set for KonIQ-10k was created by randomly selecting ten million stimuli. Then it was further scrutinized in a two-stage process. The first stage required selecting images that can be reused having a Creative Commons license. It returned roughly 4.8 million images. A tag-based sampling procedure is introduced to select 1 million images out of 4.8 million. Tag-based sampling means selecting samples with specific tags associated with the images. Thus the selected images ensured that they covered a wide range of tags across the 4.8 million images. In the second stage, the images with a resolution higher than **1024 × 768** were used and re-scaled to **1024 × 768** pixels. The images were cropped to maintain the pixel-aspect ratio. A custom cropping method was used, which was based on Hou et al. [HHK12]. The images were checked manually for inappropriate content, and duplicates were removed. In the end, 10073 images remained for the consideration of the experiment.

The subjective experiment was conducted online, and workers or users were given instructions and explanations on what an image looks like concerning certain types of distortion. Some of these distortions are as follows:

- Noise
- JPEG artifacts
- Aliasing
- Lens and motion blur

- Over-sharpening
- Wrong Exposure
- Color Fringing
- Over-Saturation

To control the quality of the workers' answers, 240 test images were rated by experts who were photographers. Test questions were generated for the crowd-sourcing experiment from this set of images. The correct answers were based on the freelancer's MOS with $\pm$ one standard deviation, allowing for some margin of error in their ratings.

For the crowd-sourcing experiment, 2302 workers participated in the study. Workers were filtered to the final 1459 in steps explained as follows:

1. The workers gave a quiz consisting of 20 test questions. Workers who had 70% rating accuracy to the MOS obtained by expert users were considered for the next stage. There were 1749 users considered for this study.
2. Hidden test questions were presented to workers throughout the experiment to assess users' attentiveness. These hidden test questions were from the 240 test images. Workers whose ratings were above 70% were allowed to complete the remainder of the experiments. This left 1648 workers, and their ratings were used to produce preliminary MOS values.
3. Workers with very little agreement with the preliminary MOS were treated as outliers.
4. Line clickers (workers with unusually high frequency for a single answer choice) were detected such that score counts of workers were calculated. Then the ratio between the maximum count of a score and four other counts was calculated. Those workers who had a ratio of greater than 2.0 were removed.

After filtering, as mentioned above, ratings from 1459 workers were selected for the KonIQ-10k data set. This resulted in at least 120 ratings for each image producing 1.2 million sound ratings of the images.

The ratings from KonIQ-10k were then used to model a BIQA approach. The conventional approach requires domain experts to engineer feature vectors from images for regression models. To train a machine learning model on the KonIQ-10k dataset, the proposed approach by Hosu et al. was based on a deep convolutional neural network (DCNN) that would take images as input. DCNN was followed by Global Average Pooling (GAP). These layers are connected to four other Fully-Connected (FC) layers: These layers have 2048, 1024, and 256 units followed by one output for predicting MOS or five units to predict the distribution of the image ratings.

For all the fully connected layers, a drop-out layer of sizes 0.25, 0.25, and 0.5 was introduced in the order of fully connected layers. A dropout layer is a regularization parameter to avoid over-fitting. For the output layer, in the case of predicting MOS, the output layer was linear since only one output is expected. In the case of predicting the distribution of ratings, the soft-max activation was used. A softmax activation is normally used at the last layer to normalize outputs to a probability distribution.

Multiple CNN architectures were selected for this study:

- VGG16
- ResNet101
- InceptionV3
- IncepotionResNet V2
- NASNetMobile

Multiple loss functions were also used along with the proposed architecture. A loss function is an objective function that measures optimality in training a model. For the prediction of MOS, the loss functions tested are the following:

- Mean Absolute Error (MAE) Loss [KYLD14]
- Mean Squared Error (MSE) Loss [LPFY16]

For the prediction of the distribution of ratings, the following loss functions were used:

- Cross-entropy Loss
- Huber Loss [Hub92]
- Earth Mover's Distance (EMD) Loss [RTG00]

InceptionResNet-V2 architecture returned optimal results upon evaluating the KonIQ-10k test set and LIVE-itw data set. The best-performing model was named *KonCept512*, which included InceptionResNet-V2 at the base and the Mean Squared Error (MSE) as the loss function.

2.2 Multi Layer Spatially Pooled (MLSP) Features

In the work of Hosu et al. [HGS19], authors proposed to extract features based on convolutional block activations from convolutional neural networks (CNN). These are the Multi-Layered Spatially Pooled (MLSP) features extracted from all convolutional blocks of pre-trained InceptionResNet-v2 Deep Neural Network (DNN). Conventional approaches in reading the images into DNN, such as CNN, require a bunch of pre-processing like cropping and re-scaling. As a result, this causes slow processing due to small batch sizes. It is further aggravated when input images of higher resolution are used. As a remedy, these images are either down-sampled or up-sampled to make it coherent for a DNN to process. It means, that the input size for the DNN is less resource intensive for training. However, this causes a potential loss of information that this DNN could have learned. Hence images which are downsized or adjusted by artificial blurs may not be harmonious with quality as it was seen by the user. Hence, producing a flaw in the predictive task for the unseen stimuli.

The authors proposed a technique called the multi-layer spatially pooled features, which can extract features from the images using Inception networks. In summary, two approaches are discussed, including using a variation of multiple heads architecture. Multiple head architecture here means that the base network is used as is in Inception based networks, and a block or head is added to this base network. It can have multiple nuances, for instance, the use of more than one head with different variations of convolutional blocks, having multiple stacks of blocks before the final top blocks, and many more. This literature proposes two major techniques: Narrow MLSP features and Wide MLSP feature architectures.

The narrow MLSP feature of an image is obtained by pooling operation on the activation output of each convolutional block of the InceptionResNet-v2 architecture. The pooling operation resizes the output of each convolutional block to a fixed spatial resolution. This spatial resolution is $\mathbf{1 \times 1}$ obtained by Global Average Pooling (GAP) - the pooling operation used for resizing the kernels of the activation block. The kernels are the learnable filters of the convolutional block with fixed width and height. These kernels are reduced to $\mathbf{1 \times 1}$ in the case of narrow MLSP and $\mathbf{5 \times 5}$ in the case of Wide MLSP.

As can be seen from figure 1, the overall depiction of how narrow MLSP and wide MLSP features are extracted and used in the training pipeline. The features are extracted by combining the resized kernels obtained via the GAP operation in each convolutional block. The pooling output dimension is the fixed size ($\mathbf{1 \times 1}$ in case of narrow MLSP and $\mathbf{5 \times 5}$ in case of wide MLSP). The final dimensions of the MLSP feature of the image in KonIQ-10k are of the size $\mathbf{1 \times 1 \times 16928}$ for narrow MLSP and $\mathbf{5 \times 5 \times 16928}$ in the case of wide MLSP.

In this thesis, narrow MLSP features were used to train the models in the ensemble, with a target variable being the rating or MOS of the image. These features were made available in an HDF5 format file. HDF5 file format stores data in a

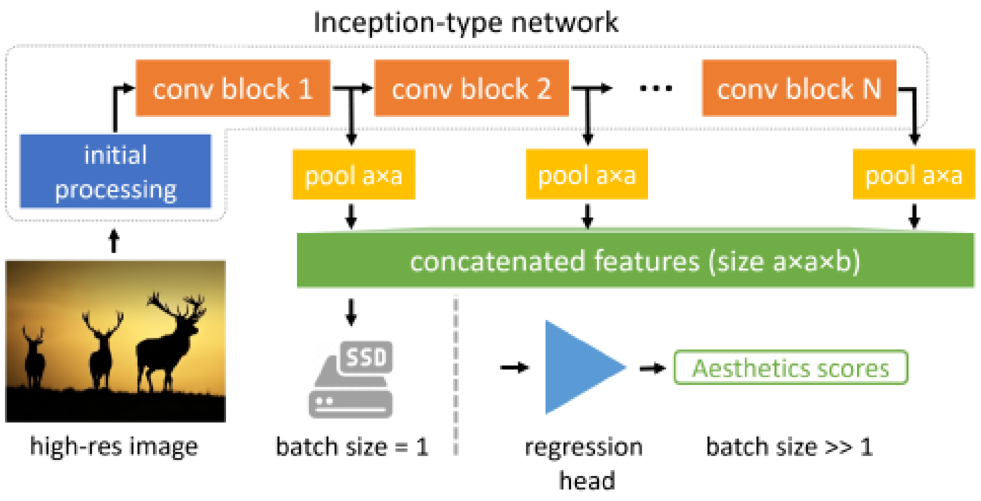

Figure 1. The feature extraction and the training pipeline utilizing MLSP. As can be seen that the image is fed as input into the architecture. The features from this image input are extracted at each activation block in multiple levels of the Inception network and concatenated to the size of $\boldsymbol{a} \times \boldsymbol{a} \times \boldsymbol{b}$. $\boldsymbol{a}$ is the output dimension of the pooling operation, and $\boldsymbol{b}$ is the total number of kernels available in the inception network. The training is carried out by feeding these features to an extraction head, such as the base model used in this thesis to predict the final output. (This image was re-used by the permission of the authors from their original literature.)

hierarchical format such that filenames are the keys and corresponding values are the vectors of dimension $\mathbf{1} \times \mathbf{1} \times \mathbf{16928}$.

2.3 Related Work

Various types of research have been carried out in IQA in the last few years, all in the area of Reduced Reference (RR) - IQA [DSS$^+$22], Full Reference (FR) - IQA [DMWS21] and No Reference (NR) - IQA, mainly utilizing CNNs [Var22]. For this thesis, I will mention literature that reviews IQA methods as part of the objective quality assessment of the images. The detailed evaluation of Image Quality Metrics (IQM) is described in the works of literature [KZG$^+$17], [GZY$^+$13], and [WB11]. Kim et al [KZG$^+$17] focus more on the IQMs involving CNNs. Gu et al. [GZY$^+$13] explain the general methodology of both subjective and objective IQA and Wang et al. [WB11] explain the objective assessment specifically for reduced and no reference IQA.

Over the years, ensemble learning has proven to be fruitful in better generalizing data and learning patterns from it. Ensemble learning combines predictions from diverse models to produce one final set of predictions. This is because each model learns patterns differently, compensating for unlearned features from other models. Details about the methods used in this research are discussed in Chapter 3 of this thesis. However, more details can be found in works of literature [D$^+$02], [SR18], and [WZ04].

Ensemble methods from machine learning have been widely applied in the area of IQA. Hammou et al. [HFH21] introduces an IQM that is built on aggregating average scores from an ensemble of different types of gradient boosting machines. The three boosting machines used are CatBoost [PGV$^+$18], XGBoost [CG16], and LightGBM [KMF$^+$17a]. Both XGBoost and LightGBM are the current state-of-the-art algorithms that optimize gradient boosting in terms of accuracy and training time, respectively. Catboost is another gradient-boosting algorithm that is optimized for handling categorical features. This metric performs significantly well on the PIPAL dataset [JG20] with state-of-the-art performance against other metrics with SROCC of **0.7758** and PLCC of **0.7752**.

This IQM metric performs FR-IQA by extracting features from reference and distorted images through convolutional neural networks. Then the ensemble of gradient-boosting machines is used to predict the image quality based on the similarity of features from reference images to those of distorted images. This metric, as mentioned earlier, was evaluated on the PIPAL dataset. The results outperformed state—of—the—art IQA metrics with respect to its correlation with human assessment.

In another paper by Lin et al. [LWK$^+$15], an ensemble-learning-based Video Quality Assessment (VQA) method is proposed. The proposed method extracts feature from multiple frames in videos. These features are then used to train several models to predict the quality of each frame. The final score is obtained by calculating the average of each frame's scores. This method incorporates dynamic IQA fusion (combining multiple IQA metrics to measure image quality) to consider the spatial complexity, temporal context of the frame, and strength among indices of

IQA. The spatial complexity measures the details visible in the frame. Temporal context measures how much each frame differs from the preceding frame. The IQA index strength measures the correlation of image quality with respect to human assessment.

Ma et al. [ML17] proposed an IQA method based on a boosting ensemble technique which is called Adaptive Boosting. This technique, like gradient boosting, learns weak learners with a weight. This weight value gets updated on each iteration, assigning higher weightage to high-error data points. The proposed algorithm in the literature is a hybrid of Random Forest and Adaptive Boosting (AdaBoost) which is applied to distorted images to carry out BIQA. It was tested on LIVEMD [JMMB12] and MDID2013 [GZYZ14] dataset. In comparison with other methods, this objective IQA returned (Spearman Rank Order Correlation Coefficient) SROCC and (Pearson Linear Correlation Coefficient) PLCC of **0.905** and **0.922** on the LIVEMD dataset. The SROCC and PLCC with MDID2013 were **0.860** and **0.869**, respectively. This comparison was made against the metrics DIIVINE [MB11], BLIINDE-II [SBC11], BRISQUE [MMB12], SISBLIM [GZYZ14], NIQE [MSB13], LQAF [LZW$^+$15] and AdaBoost-RF implementation outshined it all.

The application of IQA based on ensemble methods is also being implemented on domain-specific data. Pan et al. [PSZ$^+$17] apply an ensemble of an eye's left and right views to assess the 3D-IQA. Normally IQA is carried out, or 2D images, but Pan et al. introduced a Weighted Ensemble Deep Quality Network (WEDQN). This ensemble of deep learning networks is applied to assess symmetrically distorted stereoscopic images. The original architecture combines multiple convolutional, pooling, and fully connected layers. It takes input from both the left and right views and is processed by the network shown in Fig. 2.

Recent works have focused more on the type of ensembles built around various neural networks. These networks are mainly CNN-based, as discussed in [AA19]. Ahmed et al. [AA19] proposed an ensemble method of CNNs that utilizes a learning rate scheduler at regular intervals. The saved model checkpoints are treated as base learners of the ensemble. In this thesis, I have implemented the known ensemble methods utilizing conventional machine learning algorithms like MLP, Gradient Boosting, and k-NN to predict the perceptual quality of images and evaluate it against the original subjective ratings.

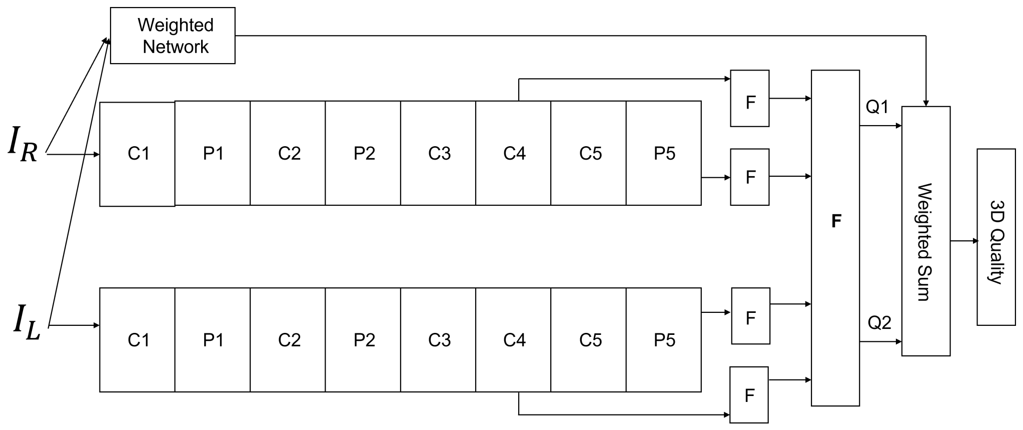

Figure 2. Pan et al. proposed an architecture that takes an input of two stereoscopic images. This network is designed in such a way that it learns the correlation between both the left view and the right view. The Bayesian deduction (a process of making inferences based on Bayesian probability) between the left and right views shows the impact factor on the overall 3D quality of the image. Hence, the network is designed to learn a single-view distribution. This architecture is an ensemble of various layers, and IQA from both views are merged to produce a final IQA figure.

CHAPTER 3

Machine Learning and Ensemble Methods

In this chapter, I will give an overview of the machine learning models used and the ensemble methods which uses these models as base learners of the ensemble. For this thesis, the machine learning models used were regression-based models. While there are different levels of perceived quality of an image (like excellent, good, fair, poor), IQA generally measures the overall perceived quality of an image on a continuous scale. However, some IQA datasets provide subjective quality scores on a discrete scale. Depending on such use case, this problem can be considered an ordinal regression problem. Ordinal regression is a type of regression analysis used for predicting an ordinal variable.

3.1 Models

For an ensemble, the machine learning models used should either be the same, like in bagging or be very different than each other, like in stacking. The idea here is to use diverse models such that their underlying structure differs, and each model compensates for the unlearned patterns from images by other models. This ensures better generalization and improves overall performance. In the following subsections, I will write about each model used and describe them in detail.

3.1.1 Multi-Layer Perceptron (MLP)

Multi-layer Perceptron (MLP) is a supervised learning algorithm in the family of Artificial Neural Networks (ANNs). This algorithm comprises multiple layers of neurons (also called perceptrons). A perceptron is a fundamental piece of the neural network which comprises of inputs, weights, biases, and activations. An MLP consists of several layers, but it can be classified into three types of layers; input, hidden, and output layer. The input layer takes an input of data and is always at the start. The input layer does not require any activations. Activations are the transformations applied to the output of neurons before it is fed to the next layer. The hidden layer is the first layer after the input layer, and it can have one or many fully connected layers. A fully connected layer means that each neuron in the previous layer is connected to all the other neurons in the next layer. The last layer is the output layer which outputs the predicted value from the network and is sometimes a probability distribution over the possible output classes (Fig. 3). Bias in a neuron or on a layer is a constant term used as an offset term in relation to weights in case the weight value converges to zero. On the other hand, weight is the unit value used to measure the strength of the connection between any two connected neurons in the layer.

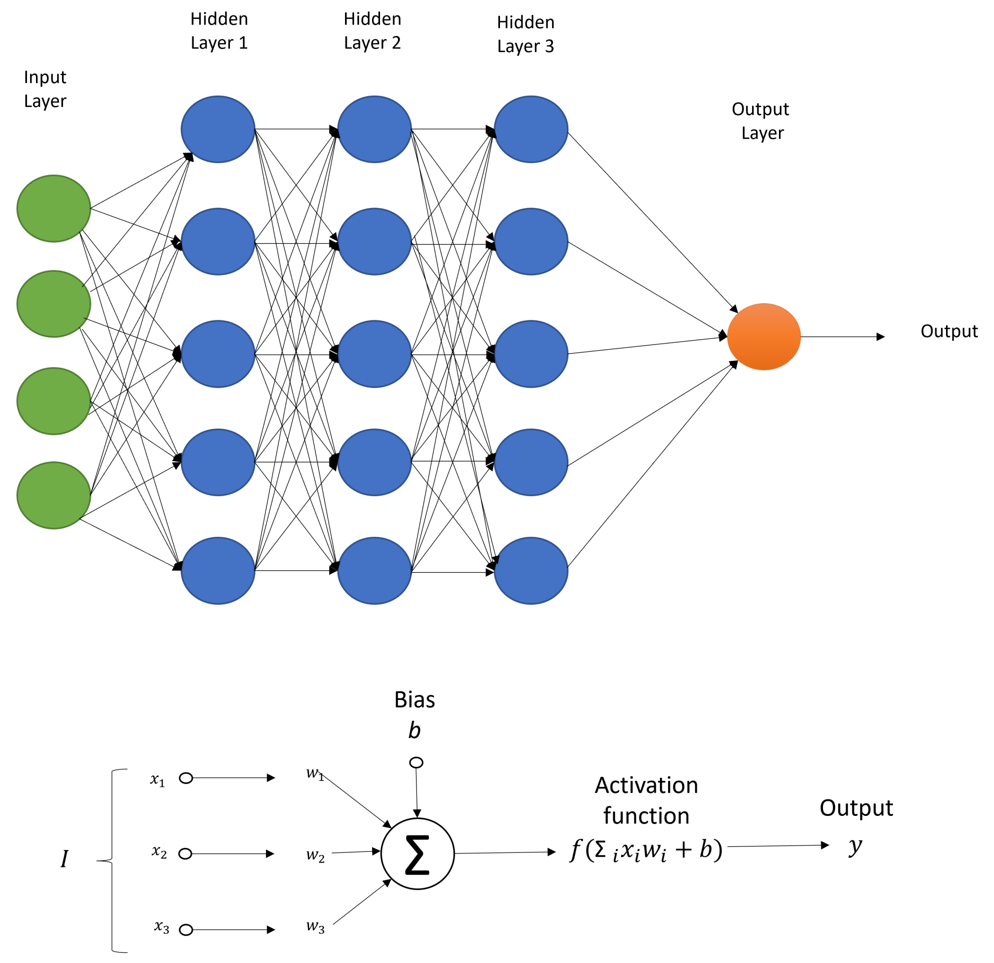

Figure 3. This figure illustrates the structure behind an MLP (top). It consists of fundamentally three portions - input layer, hidden layer/s, and output layer. The diagram below is an illustration of how fundamentally a single perceptron works. It consists of three parts; input value, weights for each input, bias, and the activation function applied to it to produce the output.

Within an MLP structure, a neuron, upon receiving input, computes a weighted sum of the inputs, which is passed to an activation function to produce the output of the neuron. Normally, activation functions used must be non-linear. This is important because it allows the model to learn complex non-linear relationships between the inputs and outputs. This makes it suitable for various applications, including image and speech recognition, natural language processing, financial analysis, etc. The output of the neuron can be expressed as (2):

$$\boldsymbol{y} = \boldsymbol{f}(\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{w_i x_i} + \boldsymbol{b}) \tag{2}$$

where $\boldsymbol{y}$ is the output, $\boldsymbol{f}$ is the activation function, $\boldsymbol{w_i}$ are the weights of the inputs $\boldsymbol{x_i}$ and $\boldsymbol{b}$ is the bias term.

Perceptrons or neurons are initialized with weights and biases that get updated during the training process by backpropagation. A gradient-based optimization process called "gradient-descent" computes the gradients of the loss function with respect to weights and biases, which are updated in a stochastic manner.

MLP has a wide range of applications, such as:

- **Computer Vision:** MLPs are widely used in Computer Vision (CV) tasks like image segmentation and object detection. Besides convolutional neural networks, MLPs are also considered in designing CV-based models, producing promising results in various CV-related tasks.

- **Natural Language Processing:** MLPs are used in the natural language domain to learn patterns from raw text. It is used commonly in tasks like sentiment prediction, text generation, text summarization, and others. MLPs can learn patterns in text with help from large labeled training data.

- **Medical Diagnosis:** Medical conditions can be diagnosed by understanding the symptoms, medical history, and test results. Training MLPs on medical-related data, like X-rays, will be able to learn patterns in the data and detect relevant medical conditions if there are any.

- **Robotics:** MLP has been used widely in the field of robotics to perform tasks like grabbing an object, working through tough environments, and working with precision. MLPs are also widely applied in the reinforcement learning domain where these are used in the underlying structure to learn crucial patterns concerning the respective environment.

MLP has also seen its implementation in IQA applications, involving learning image features and their relations with image quality scores. These are mainly centered around using CNN-based architectures combined with FC layers. FC layers are essentially MLPs in a generic sense having multiple layers of neurons all "fully-connected" with other neurons in adjacent layers.

3.1.2 Light Gradient Boosting Machines (LGBM)

Developed by Microsoft, Light Gradient Boosting Machines (LightGBM or LGBM) is a gradient boosting framework that utilizes tree-based learning algorithms. It was designed to be efficient in training speed and memory usage while returning high accuracy on predictions. LGBM has been widely used in academia and industry and has outperformed other machine learning algorithms in various areas.

LGBM adds weak learners, usually decision trees, which are generated iteratively. Upon each iteration, the algorithm trains a new tree and predicts the residuals errors which are adjusted on the next iteration until it converges. Residual errors or just "residuals" are the difference between the current approximation and the known target value. Predictions from all the trees are then aggregated to return an optimal value.

LGBM is different from the conventional gradient-boosting algorithm with respect to how it constructs the decision trees. Instead of building trees level-by-level it constructs trees using a leaf-wise approach. In this approach, the tree is grown by splitting the leaf on the largest gradient, rather than on the number of instances. This means that the tree is built using the best-first manner. This approach has proven to have faster convergence on residuals and returns better accuracy, especially for high-dimensional data.

LGBM also uses a histogram-based approach to convert continuous data to discrete data. It significantly reduces the usage of memory and the time for training as compared to conventional methods. The algorithm creates bins of feature values into intervals and then creates a histogram for each feature based on the frequency values from each bin. The algorithm then uses these histograms to calculate the gradient for each feature. This process is used in exclusive feature bundling discussed later in this section.

In principle, LGBM algorithm can be divided into four parts:

Objective function:

Like any other gradient boosting machine algorithm, LGBM also uses an objective function to compute loss on predicted values from true values. It is essentially the sum of the differentiable loss, basically the "gradient" and a regularization term (3):

$$\boldsymbol{\mathcal{L}(y, f(x)) = \sum_i l\left(y_i, f\left(x_i\right)\right) + \sum_k R\left(f_k\right)} \tag{3}$$

where $\boldsymbol{y}$ is vector for actual values, $\boldsymbol{f(x)}$ is the predictions, $\boldsymbol{l(y, f)}$ is the loss function, $\boldsymbol{R(f)}$ is the regularization term for avoiding overfitting and $\boldsymbol{k}$ is the index of the boosted tree.

Finding the split:

LGBM finds the best split for each feature by using a splitting algorithm. The algorithm considers all split points and selects the one which maximizes the gain (4).

$$\text{Gain } = \boldsymbol{\frac{1}{2}\left[\frac{\left(\sum_{i \in I_{\text{lett}}} g_i\right)^2}{\sum_{i \in I_{\text{left}}} h_i + \lambda} + \frac{\left(\sum_{i \in I_{\text{right}}} g_i\right)^2}{\sum_{i \in I_{\text{right}}} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda}\right]} - \gamma \tag{4}$$

where $\boldsymbol{I}$ is the set of values on the node being split, $\boldsymbol{I}_{\text{left}}$ and $\boldsymbol{I}_{\text{right}}$ are the set of values in the left and right child nodes respectively. $\boldsymbol{g_i}$ and $\boldsymbol{h_i}$ are the gradients and Hessians of the values $\boldsymbol{i}$, lastly, $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ are the regularization terms for better generalization of the model. The Hessians are essentially the weights assigned to child nodes of the decision trees and are known by the parameter `min_child_weight` which is passed during initialization in code.

Building the tree:

LGBM builds decision trees in a depth-wise fashion. The algorithm iteratively applies split on nodes, starting from the largest gain, and stops when the maximum tree depth has been achieved or when the number of values in the leaf node reaches a specified threshold. The depth is restricted and instead, a leaf-growth strategy is implemented to optimize the training procedure.

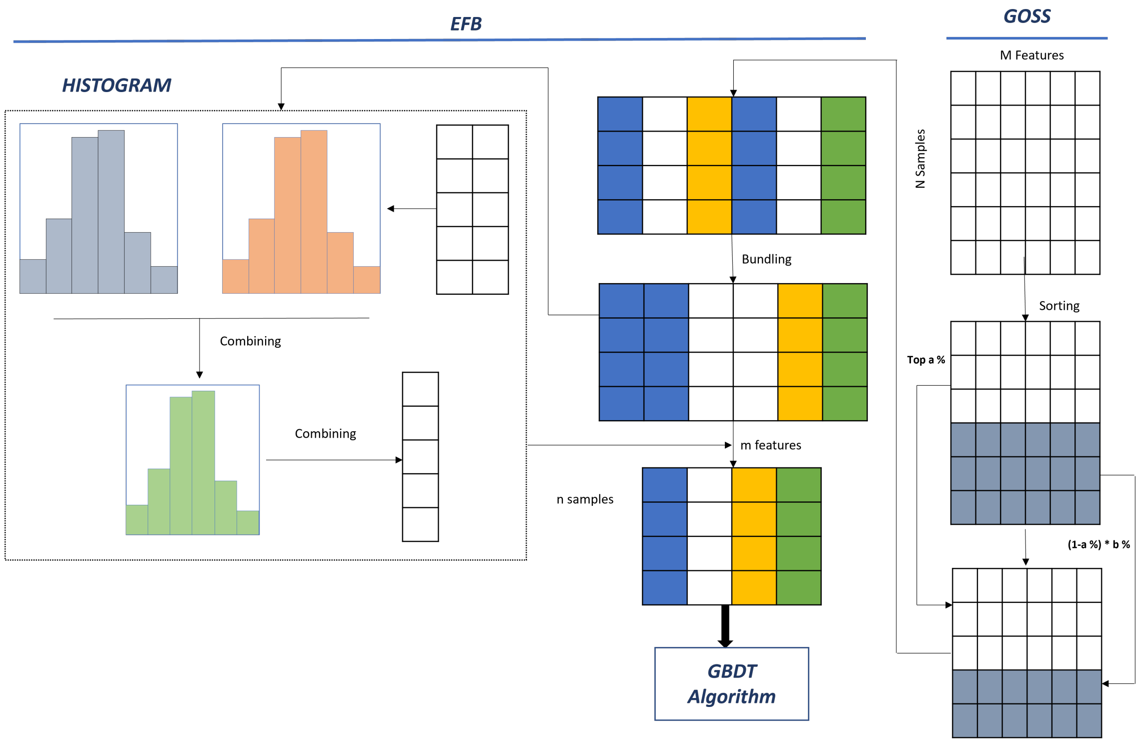

Figure 4. This figure illustrates how techniques like Gradient Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) are used together to handle a very large number of data samples and features to optimize the training process. As can be seen, the GOSS approach keeps samples with high gradients (unshaded regions) and does sampling with lower gradients (shaded regions). Through the EFB approach, the large number of features from these samples are used to bundle similar features to optimize training.

LGBM uses techniques [KMF+17b] like:

- **Gradient Based One-Side Sampling (GOSS):** GOSS optimizes the training process by selecting those points which have larger gradients and fewer points that have smaller gradients. High gradient points are generally high-error points.

- **Exclusive Feature Bundling (EFB):** EFB is the feature optimization strategy that bundles multiple features or "mutually exclusive" features. It then uses one feature to represent that bundle.

As it can be seen from Fig.15, how the above techniques work in harmony to optimize the training procedure. This algorithm used by LGBM is called Gradient Boosting Decision Tree (GBDT) and constructs trees using a leaf-growth strategy in parallel by restricting the depth of each tree.

To conclude LGBM is a state-of-the-art algorithm for gradient boosting with its implementations in C++, Python, R, and Java it is a widely used algorithm. It also supports a large range of loss functions and numerous evaluation metrics.

3.1.3 k-Nearest Neighbours (kNN)

k-Nearest Neighbors (k-NN) is a supervised learning algorithm used both for regression and classification-based modeling. It is the simplest model which is used in the domain of machine learning which bases its prediction only on distances.

The notion is that k-NN classifies objects which are close to each other in the feature space or are likely to have similar labels. In the machine learning context, the prediction is based on the labels from the nearest neighbors in the train set.

The generic algorithm for k-NN can be described as follows:

1. The distance between an unknown object and other objects from the training set is computed.
2. The k-NN is selected based on the distances which are calculated.
3. The label of this unknown object is predicted based on the most frequent label in this nearest neighbor's window.

From the algorithm explained above it is obvious that the main parameters in consideration for the k-NN are the distance metric and the value of "k". The distance metric helps in measuring the proximity of any values in the available feature space. The commonly used distance metrics are the Euclidean distance, Manhattan distance, and Cosine distance. The value of k defines the number of neighbors that are to be considered for the prediction of an unknown object. The choice of the value of k has its trade-offs; a smaller value of k can result in a smaller decision boundary while a larger value of k will return a smoother decision boundary but can cause overfitting.

To further the understanding of k-NN in detail I will discuss its application in both classification and regression tasks:

3.1.3.1 k-NN for Classification

For the given training set $\boldsymbol{m}$ with labeled objects $\boldsymbol{T = (x_1, y_1), (x_2, y_2), ..., (x_m, y_m)}$, where $\boldsymbol{x_i} \in \boldsymbol{R^n}$ is the feature vector of objects $\boldsymbol{i}$ and its label $\boldsymbol{y_i} \in \boldsymbol{1, 2, ..., K}$ - there is an unknown or unlabeled object $\boldsymbol{x_u}$ that has to be classified. The algorithm will work as follows:

1. The distance between $\boldsymbol{x_u}$ and all the data points in $\boldsymbol{T}$ is computed using a distance metric $\boldsymbol{d(x_i, x_j)}$.
2. Based on the distance calculated $\boldsymbol{k}$ nearest neighbours of $\boldsymbol{x_u}$ in $\boldsymbol{T}$ is selected.
3. The label that is most common in $\boldsymbol{k}$ nearest neighbours is assigned to $\boldsymbol{x_u}$.

The choice of $\boldsymbol{k}$ can be further scrutinized by means of a cross-validation strategy. In case the frequency of labels turns out to be equal in the k-NN window, then the predicted label can either be selected randomly or the label is selected based on some other criteria.

3.1.3.2 k-NN for Regression

As mentioned earlier, in the regression task the output variable is the continuous output variable rather than the categorical label. For the training set $\boldsymbol{m}$ with labeled objects $\boldsymbol{T = (x_1, y_1), (x_2, y_2), ..., (x_m, y_m)}$, where $\boldsymbol{x_i} \in \boldsymbol{R^n}$ is the feature vector of object $\boldsymbol{i}$ and $\boldsymbol{y_i} \in \boldsymbol{R}$ is the continuous variable. In order to predict the target continuous variable for an unknown object $\boldsymbol{x_u}$, regression works as follows:

1. The distance between $\boldsymbol{x_u}$ and all the data points in $\boldsymbol{T}$ is computed using a distance metric $\boldsymbol{d(x_i, x_j)}$.
2. Based on the distance calculated $\boldsymbol{k}$ nearest neighbours of $\boldsymbol{x_u}$ in $\boldsymbol{T}$ is selected.
3. The prediction of output variable for unknown object $\boldsymbol{x_u}$ is the average of all the variables in the $\boldsymbol{k}$ nearest neighbors space.

The average of all the variables is expressed as (5) where $\boldsymbol{N_k(x_u)}$ is the set of indices of $\boldsymbol{k}$ nearest neighbors of $\boldsymbol{x_u}$ in $\boldsymbol{T}$ and $\boldsymbol{N}$ is the total number of neighbors present in the nearest neighbor window.

$$\boldsymbol{y_n = \frac{1}{k} \sum_{i \in N_k(x_u)}^{N} y_i} \tag{5}$$

There are many pros and cons of using k-NN which should be considered for any problem at hand. Some of them are:

Pros:

- It is a simple and naive model.
- Distribution of data is not taken into account.
- Due to its non-parametric structure, no estimation is required on its parameters other than the value of $\boldsymbol{k}$.

Cons:

- Can be computationally expensive depending on the size of data.
- Memory intensive, since all of data has to be stored in memory.
- It can be sensitive to the choice of distance metric and the value of $\boldsymbol{k}$.
- Prone to be sensitive to outliers in the data.

Several improvements to k-NNs have been made over time to cater to the limitations of the algorithm and improve its performance. This involves:

- **Weighted k-NN**: Weights are assigned to neighbors in the nearest neighbor window the unknown object falls in. It is done by employing a weighted function that assigns varying weights to objects relative to the distance value from the unknown object.
- **Localized k-NN**: The value of $\boldsymbol{k}$ can be customized based on the density of points. For example, in sparse regions on the plane, a smaller value of $\boldsymbol{k}$ will be optimal, and a larger value of $\boldsymbol{k}$ otherwise. The dynamic selection of the value of $\boldsymbol{k}$ is ensured by implementing an appropriate function, such as "centroid estimation".
- **Ensemble Methods**: Combining with other algorithms to make predictions for improved predictions. It can be used as the level one learner in Stacking, as I have in this study, and combine its predictions with other models to better generalize the results.
- **Dimensionality Reduction**: The majority of distance-based algorithms like K-means and k-NNs are prone to the curse of dimensionality. This means that too many features can degrade the performance of the algorithm. There are many multiple techniques that have been introduced over time to reduce dimensions and projects dataset in the lower number of feature vector space. These algorithms include principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE) and Multidimensional Scaling (MDS). It can help to reduce the noise in the algorithm and avoid redundancy in data, in turn making k-NN an effective choice for predictions.

To conclude, the k-nn for classification is based on the majority of classes in the k-nn window, and for the regression task, the target variable will be the average of continuous values considered in the window. (Fig. 5).

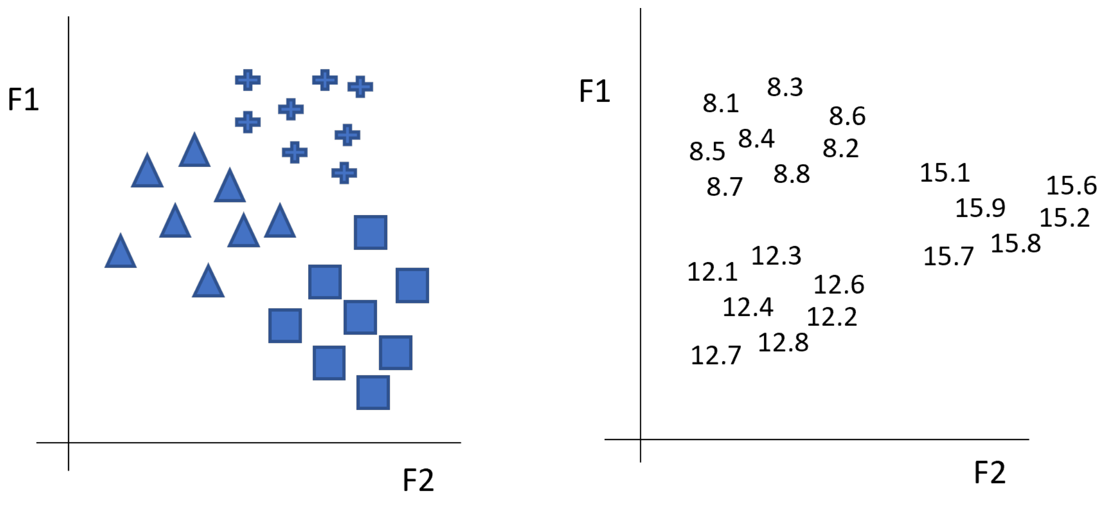

Figure 5. This figure illustrates how both k-NN classification (left) and k-NN regression (right) would look like for predicting the target variable. For the classification task, which is the shape in this example (left), the class will be decided based on the majority of type of shape in the nearest neighbors window. It will not be something in between. Similarly, for the regression task, the target variable is expected to be numeric. Therefore, the average of points taken in the nearest neighbors window will be the final output, and since it is a continuous value it **can** lie somewhere in between.

3.2 Ensemble Methods

Ensemble methods or commonly called "Ensembling" is an approach used in machine learning by combining predictions from multiple models to improve overall predictions. These methods are widely applied for both classification and regression-based tasks in different areas of machine learning like Computer Vision, Natural language Processing and Data Mining in general.

Fundamentally, ensemble methods combine the predictions of multiple models, which are also called "base learners" or "weak learners". These are then used to produce final predictions on the data set. The idea is to make models more robust by reducing bias and/or variance and producing more generalized predictions. In due process, the errors made in predictions by one model are compensated in predictions made by other ones.

Multiple techniques exist which will be looked at in detail in the following sections.

3.2.1 Bagging

Bagging or also known as bootstrap aggregating uses a sampling strategy from the training set of the dataset. This sampling is carried out randomly with replacement. Each sample is used to train a base learner and the final prediction is the aggregate of individual predictions made by each base learner. Bagging is, in general, useful when each base learner is essentially "weak". It is known to reduce variance against the true dataset.

Bagging can be explained mathematically in the following manner:

For the training dataset of size $\boldsymbol{m}$ with features $\boldsymbol{X_1, X_2, ..., X_m}$ and labels $\boldsymbol{y_1, y_2, ..., y_m}$ there are $\boldsymbol{B}$ bootstrap samples sets for each base learner $\boldsymbol{b = 1, 2, ..., B}$ where $\boldsymbol{B}$ is the total number of base learners. The bootstrap sample of size $\boldsymbol{n}$ is selected randomly from the original dataset with replacement. This bootstrap sample can be denoted as, $\boldsymbol{D_b}$. Base learner $\boldsymbol{b}$ is trained on the bootstrap sample $\boldsymbol{D_b}$ and learns a function which can be denoted as $\boldsymbol{h_b(X)}$. The generic structure behind bagging is described in Fig. 6.

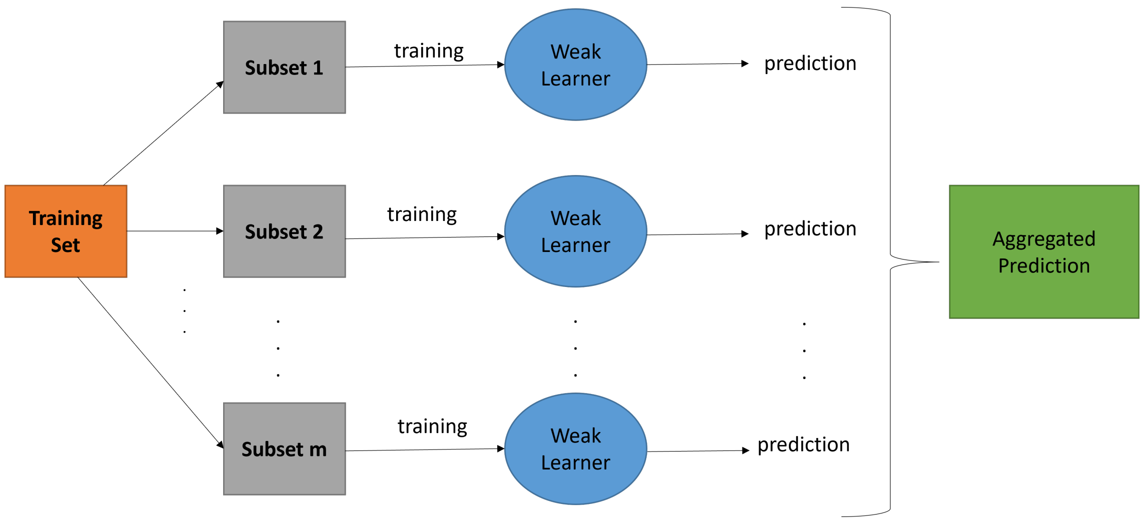

Figure 6. This figure illustrates the bagging process in a nutshell. As can be seen that multiple subsets of data are generated by sampling with replacement. Each subset is fed as training input to the corresponding weak learner. Each weak learner produces a set of predictions on the test data. These predictions are aggregated to produce a final prediction. In the case of the classification task, these predictions will be decided by the pre-defined voting strategy. In case of the regression task, the final predictions will be the average of all the predictions made by each weak learner.

The final predicted output is carried out by averaging the predictions of base learners (6).

$$\hat{y} = \frac{1}{\boldsymbol{B}} \sum_{\boldsymbol{b=1}}^{\boldsymbol{B}} \boldsymbol{h_b(X)} \tag{6}$$

The averaging over multiple base learners reduced the variance in predictions. It can be shown in the equation (7) where the first term is the variance of base learners and the second term is the covariance among the base learners. Ideally, the covariance term is small, since all the base learners are trained on separate bootstrap samples.

$$\boldsymbol{Var(\hat{y}) = \frac{1}{B^2}\sum_{b=1}^{B} Var(h_b(X)) + \frac{1}{B^2}\sum_{b\neq b'} Cov(h_b(X), h_{b'}(X))} \tag{7}$$

Another common approach in bagging is voting which involves combining predictions of base learners and the majority vote becomes the final prediction. The core difference between bagging and voting is the sampling strategy used. In voting, subsets of datasets are generated from a fixed percentage of the dataset. These subsamples also are generated with replacement. There are different kinds of voting strategies commonly used. Some of them are:

- **Hard Voting**: This is the actual name of the process where a majority vote is considered for final prediction. It is commonly used in classification tasks.
- **Soft Voting**: The final prediction is calculated by taking the average of the predicted probabilities of the base learners. Whatever class has the highest average of the probabilities is chosen to be the final prediction. It is again also commonly used in classification tasks where the output of base learners is the probabilities.
- **Weighted Voting**: In this voting strategy the predictions of all the base learners are combined by assigning weights to each learner. The choice of weights is assigned by measuring the performance of each individual learner. The strategy to calculate weights is usually needed when some base learners outperform or underperform the rest of the learners.

While each of its strategies has its pros and cons, the choice of voting method is subject to the problem at hand and the types of base models used.

3.2.2 Boosting

Boosting is another popular ensemble technique where models are trained in a sequence and errors from the previous models are adjusted. In simple terms, each model learns from the mistakes made by the predecessor model by assigning a higher weightage to the misclassified predictions. This process continues until misclassifications cannot be adjusted any further.

Various variants of Boosting algorithms have been introduced, such as AdaBoost, Gradient Boosting, XGBoost, and LightGBM. All of these use a cost function to measure the loss, learning rate, and decision trees as the base learners.

The algorithm can be described in the following steps and is also illustrated in Fig. 7:

1. **Initialization of weights:** To begin with, each row in the training set is assigned an equal weight. This is usually $\boldsymbol{w_i = \frac{1}{N}}$ for $\boldsymbol{i = 1, \ldots, N}$ where $\boldsymbol{N}$ is the total number of rows in the train set.

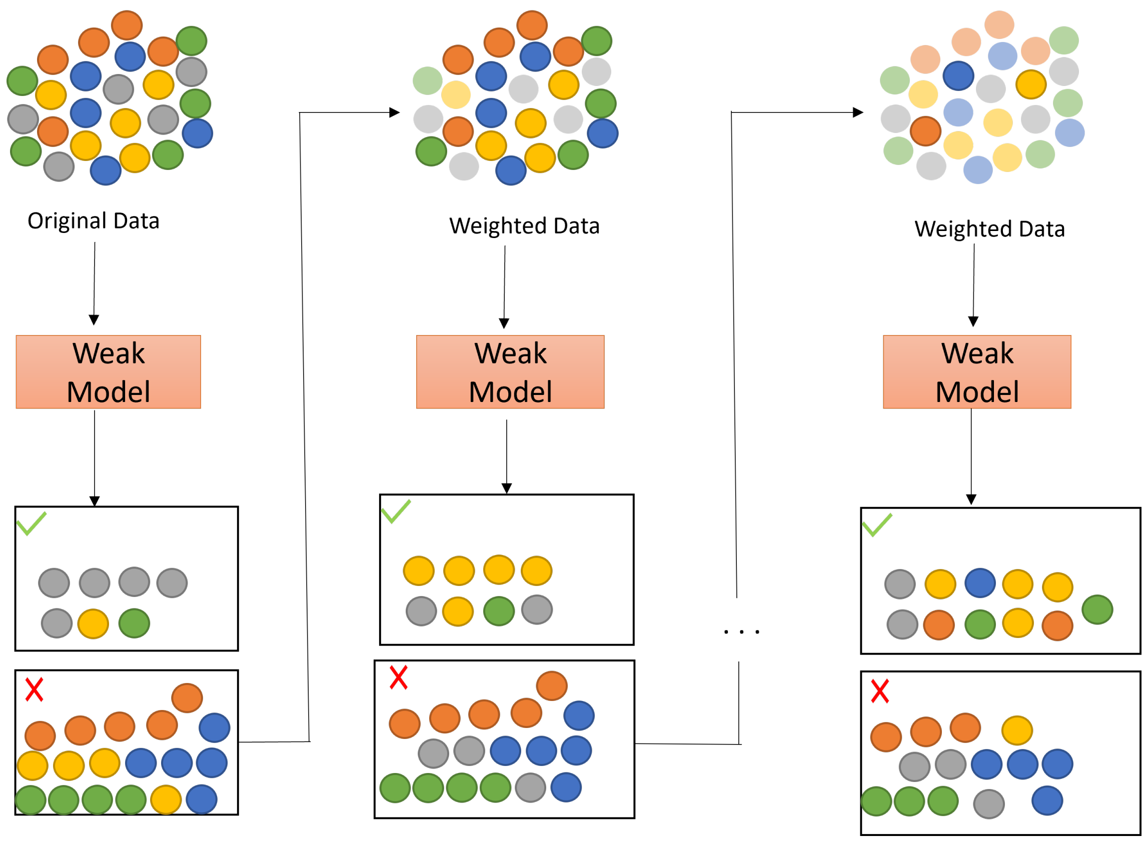

Figure 7. This figure illustrates the common approach in Boosting. In simple terms, for each iteration, the model, which is usually a decision tree, assigns higher weightage to misclassified predictions or predictions with higher error (in the case of regression). This data is retrained in the next iteration with the adjusted weights on the misclassified results or high error targets and continues to do so until it cannot further improve.

2. **Training Weak Learner:** The model is trained on a train set where the target variable is the average of the ground-truth values. This weak learner can be denoted as $\boldsymbol{h_m}$. The loss metric is used to find the predictions. These predictions are subtracted from the true values to produce residuals.

3. **Error Computation:** For the first model its performance is evaluated. The sum of the weights of the high-error or misclassified results are used to compute the error. If the value of the weight is high that means the number of misclassified results is high and requires more consideration during the training process. It can be denoted as $\boldsymbol{E_m} = \sum_{i=1}^{N} \boldsymbol{w_{i|y_i \neq h_m(x_i)}}$ where $\boldsymbol{E_m}$ is the error value and $\boldsymbol{w_i}$ are the weights of the misclassified results.

4. **Update the weights:** After computing the error, the weights of the data points are updated. It is done such that, the weights of correctly classified results are reduced and increased otherwise.

5. **Model training on next iteration:** The updated weights are used by the model in the next iteration to further minimize the number of misclassified or high-error data points.

6. **Repeat steps 3 to 5:** The process is repeated for a set number of iterations until the weights cannot be updated anymore and results cannot be improved. The final prediction is calculated by weighted aggregation of predictions from all weak learners.

Boosting can handle noise in data, works well with imbalanced classes, and can learn non-linear relationships in the data. Through recent advancements, it is also possible to control over-fitting with the help of regularization. The performance of boosting can be optimized by careful choice of the base learner, loss function, and the learning rate. Boosting can be a computationally expensive process and may take a lot of time on really large datasets.

3.2.3 Stacking

Another ensemble method used commonly for generalized predictions is called Stacking. It combines predictions from a set of diverse models and uses them as training samples for the meta-learner. The second-level model in Stacking learns the predictions from multiple models and produces the final set of predictions. This way the model generalizes better with respect to reduced bias or variance. In a nutshell, the basic Stacking process can be explained in the following steps:

1. Diverse set of models is used to make predictions. For example, Multi-Layer Perceptrons (MLP), Random Forest (RF) or Gradient Boosting (GB), Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), etc. This can be denoted as $\boldsymbol{L_i}$ where $\boldsymbol{i}$ is the $\boldsymbol{i}$th base model $\boldsymbol{L}$ in the structure.

2. Use the validation set to make the predictions $\boldsymbol{\hat{y}}$ by each model from step 1. Combine the predictions such that predictions from each model are stacked horizontally in the matrix.

3. Using the original target variables from the validation set. The predictions from Step 2 and target variables $\boldsymbol{y}$ from the validation set become the training set for the meta-learner. This training set can be denoted as $\boldsymbol{X}$.

4. The meta-learner is trained on set $\boldsymbol{X}$ to produce the function denoted as $\boldsymbol{f_{meta}}$. This learner is ideally a simple model, commonly a Linear Regression is used.

5. To test the model, the predictions are first made on the base models' test set. These predictions are combined to form the input for the f_{meta} function and final predictions are made.

The basic implementation of Stacking can be described in the equation (8):

$$\hat{y}_{test} = f_{meta}(\hat{y}_1, \hat{y}_2, \hat{y}_3, \ldots.., \hat{y}_L) \tag{8}$$

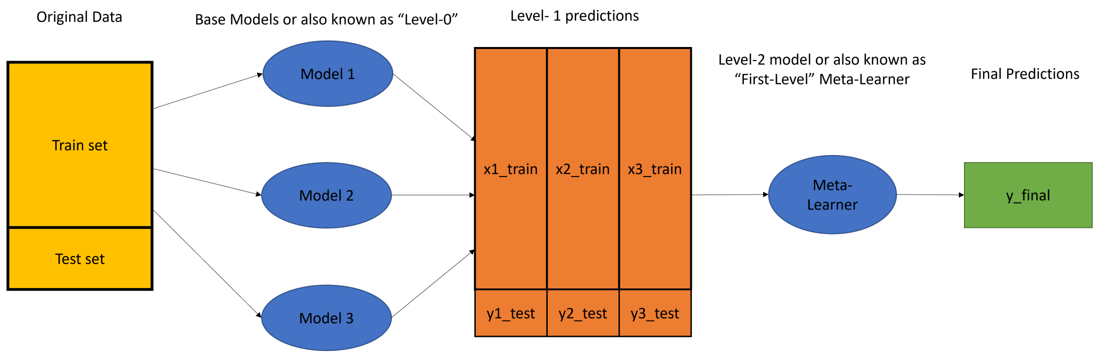

Figure 8. This figure describes the basic structure used in the Stacking ensemble. It can be seen that multiple base models are trained on the training set. The predictions of each are stacked together and become the train set for the first-level meta-learner. These stacked predictions are also called level-1 predictions. The meta-learner then makes the final set of predictions.

There are multiple nuances to Stacking and it can be more complex than the basic steps as defined above. Some of these nuances can be described as follows:

- **Feature Engineering:** Generate a set of predictions from the base model by using multiple strategies to generate features from the data. Use these predictions as input to the meta-learner. An alternate approach could be to use predictions from base learners and apply feature engineering techniques to them. This can include calculating mean or standard deviation, for example, from the base predictions.

- **K-Folds:** Using k-folds on the training set to create a training set and validation set for each kth fold. Combine predictions on the validation set from each fold. Repeat the process with multiple models. Horizontally stack all the predictions from different models and this will serve as the input to the meta-learner. It is similar to the basic steps described above but instead of using one validation set, multiple validation sets are used using the k-fold strategy.

- **Model Selection:** Use cross-validation on each base model to select the best model. This is done by evaluating the model on multiple subsets of data and selecting the best one.

- **Multiple Meta-Learners:** Using multiple meta-learners at first level. This can further make a better generalized and robust set of predictions. This strategy comes in handy when working with complex or high-dimensional data sets. An alternative to finding the average of predictions from first-level meta-learners is to create a second-level meta-learner too. This approach is also called "Hierarchical Stacking".
- **Model Fusion [RP07]:** Combining predictions from base learners at different levels of Stacking. For instance, combining predictions from the base model with predictions from the first-level meta-learner for the input at the second-level meta-learner.

The choice of a Stacking strategy is subject to the problem at hand. This can vary and can return different results. However, it is a complex choice of designing the entire Machine Learning pipeline. As a drawback, data leakage at different steps can lead to overfitting of the model. It's also hard to interpret the internal predictions within the entire structure, especially at the level where predictions for meta-learners are combined.

CHAPTER 4

Data Preparation

Any part of any machine learning application is data processing, and data preparation is one of the early steps taken. This chapter will discuss the complete approach to preparing data sets for single users and ensemble for clusters of users.

To begin with, a matrix $\boldsymbol{M_{ij}}$ is prepared such that on the rows $\boldsymbol{i}$ are the users or workers who rated the image and columns $\boldsymbol{j}$ are the images rated by the $\boldsymbol{ith}$ user. Since each user has rated a different subset of images, this matrix is the sparse matrix of the size $\mathbf{1714 \times 10073}$. The number of users is higher than the total number of users whose ratings were considered for the actual database. This is because workers being screened also provided ratings for the final images considered for KonIQ-10k data set. Note, that these are not sound ratings and were removed during the course of this thesis.

The train test split was 90 percent training and 10 percent for the test set. Out of 90 percent, 10 percent of training data is kept for the validation set. This split was carried out on the columns that are the images in the $\boldsymbol{M_{ij}}$ matrix. That way, the number of rows or worker IDs was the same.

The validation set is helpful in the training process since this is the set that the model uses to minimize the loss function. This means that if any predictions were made on the validation set by the model, the prediction accuracy would be high since the model based its loss on the available validation set. The training process will not have used the features available in the validation set. In this use case, validation helps validate the hyperparameters that are optimal for the data set the model validated on.

4.1 Single User

For this thesis, users who rated more than 500 images are considered. There were, in total 756 users considered for this thesis. The idea here is to consider only those users who produced sound ratings and a sufficient number of ratings that could produce meaningful MOS of the images.

The idea here is to create an ensemble such that each model corresponds to each user and is trained on the ratings the user provided. This way, the ensemble will comprise models that have learned authentic users' ratings. This will result in a meaningful predicted MOS from the ensemble for IQA on KonIQ-10k.

The comparison of the distribution of change in MOS of images before and after the filtering of users (keeping only users who rated at least 500 images) seems to be unchanged (Fig. 9). The images considered for figure 9 were all the screened images (10073 in total) from the KonIQ-10k dataset. A comparative analysis was done between the users before and after the filtering step to investigate this further. The distribution involved the total number of ratings given by each user. Fig 10 shows the distribution of these numbers; the left image shows the number of ratings made by each user before the filtering operation, and the right one indicates otherwise. The visible peak on the right of each figure shows that the number of raters rated 1800 images or more is 366, which is the highest in the database.

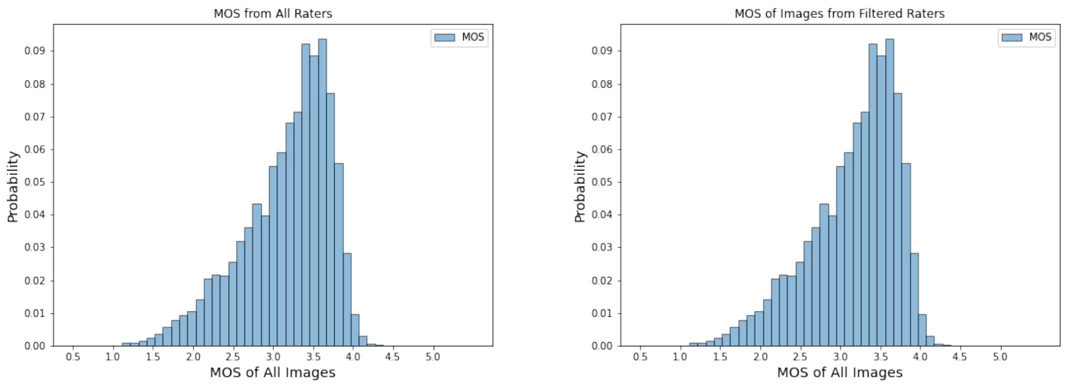

Figure 9. Figure on the left shows the rating distribution from all the raters, and the right shows the distribution of MOS from raters rated at least 500 images. It can be seen there is no visible difference in the distribution of MOS both before and after filtering the users.

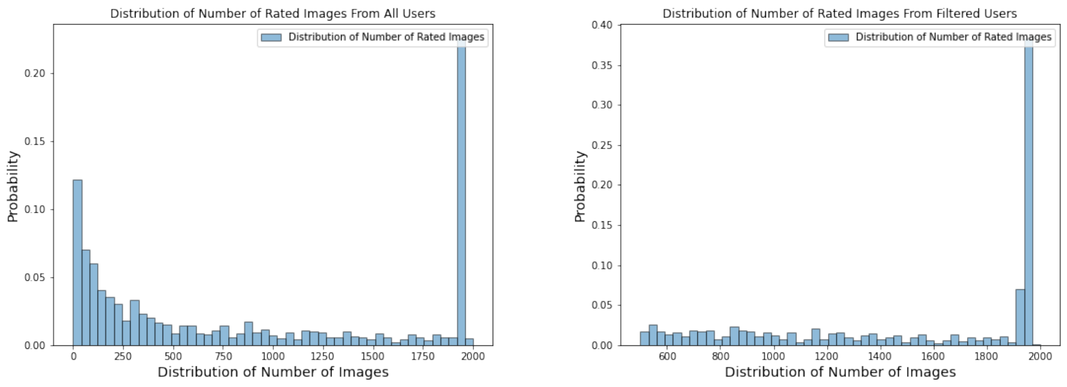

Figure 10. The figure on the left shows the distribution of the number of images rated by all the raters. The figure on the right shows the distribution of the number of images rated by raters who rated more than 500 images. In both cases, there is a visible peak towards the right of the figures; this is because 366 users rated more than 1800 images

4.2 Groups of Users

Another part of this thesis is to cluster raters or users into groups such that each group is unique. The idea here is to group users who gave similar ratings and test how an ensemble of models of each group of user's data performs compared to an ensemble of single users. Again, only users who rated 500 images or more are considered for finding unique groups of users.

The grouping of users who gave similar ratings is carried out by means of using similarity measures. The similarity metrics considered here are the correlation coefficients and co-variances. The correlation coefficient among users is computed from the sparse matrix, $\boldsymbol{M_{ij}}$. These coefficients are presented as a similarity matrix with a unit diagonal. The covariance and correlation values demonstrate the proximity of dissimilarity and similarity of vectors, respectively. Both correlation and co-variance can be regarded as the same values but on a different scale. It can be seen from equation (14) that this scaling factor is $\frac{1}{\sigma_X \sigma_Y}$.

Based on the law of cosines, the squared distance between two vectors is defined as $\boldsymbol{d_{12}^2 = h_1^2 + h_2^2 - 2h_1h_2\cos\phi}$. The $\boldsymbol{h_1^2}$ and $\boldsymbol{h_2^2}$ are the sum of squared coordinates, and $\boldsymbol{2h_1h_2\cos\phi}$ is the scalar product or the dot product of two vectors. For the correlation matrix, if the data is centered on an n-dimensional plane, then normalized $\boldsymbol{h^2}$'s become the variances of vectors and $\boldsymbol{\cos\phi}$ becomes the Pearson coefficient $\boldsymbol{r}$ (appendix A.1) (since the value will be within the range of 1 and -1). Therefore the dot product $\boldsymbol{h_1h_2\cos\phi}$ becomes the scalar product in the form $\boldsymbol{\sigma_1\sigma_2r_{12}}$, essentially the covariance. For a similarity matrix with 1 unit on the diagonal, the squared Euclidean distance matrix can be computed from the law of cosine formula above derived to $\boldsymbol{d^2 = 2(1-s)}$. The $\boldsymbol{2}$ becomes the factor for the Euclidean distance, and $\boldsymbol{d^2 = 1 - s}$ would suffice.

The cosine theorem is applied to the correlation coefficient matrix resulting in a dissimilarity (or euclidean distance) matrix. Hence, the clustering operation used to cluster unique users on this dissimilarity matrix returns a similar grouping of users with several clusters close to three or four.

The clustering carried out is agglomerative hierarchical clustering. This clustering generated the clusters using a bottom-up approach, meaning each data point starts off as a cluster, and pair of clusters are merged as it goes up in the hierarchy. For this clustering step, I used the "Ward" method [ML11], which is the minimum variance criterion and is the recommended criterion for the similarity to dissimilarity converted matrices [MAET15]. The minimum variance criterion minimizes the overall within-cluster variance. The idea here is that the minimum increase in the within-cluster is calculated at each step of hierarchical clustering. This increase can be deemed as a weighted squared distance between clusters. Therefore Ward's minimum variance criterion can be represented as in (9):

$$\boldsymbol{d_{ij} = d\left(\{X_i\}, \{X_j\}\right) = \|X_i - X_j\|^2} \tag{9}$$

Fig. 11 shows results from the clustering operation applied to the correlation matrix. The distinct clusters below the assigned threshold are still the same, i.e. 3.

The final set of clusters used in the experiment was obtained from the dissimilarity matrix converted from the correlation matrix. Fig. 12 shows the range of images rated by users belonging to each cluster. It can be seen that the variation in the median number is different. For cluster 1, the median range is around 1800, and for cluster 3 it is approximately 1500. However, for cluster 2, the median range is the upper quartile close to 1900. It can be seen that number of raters in cluster 2 was the lowest, with raters who rated over 1900 images (table 1).

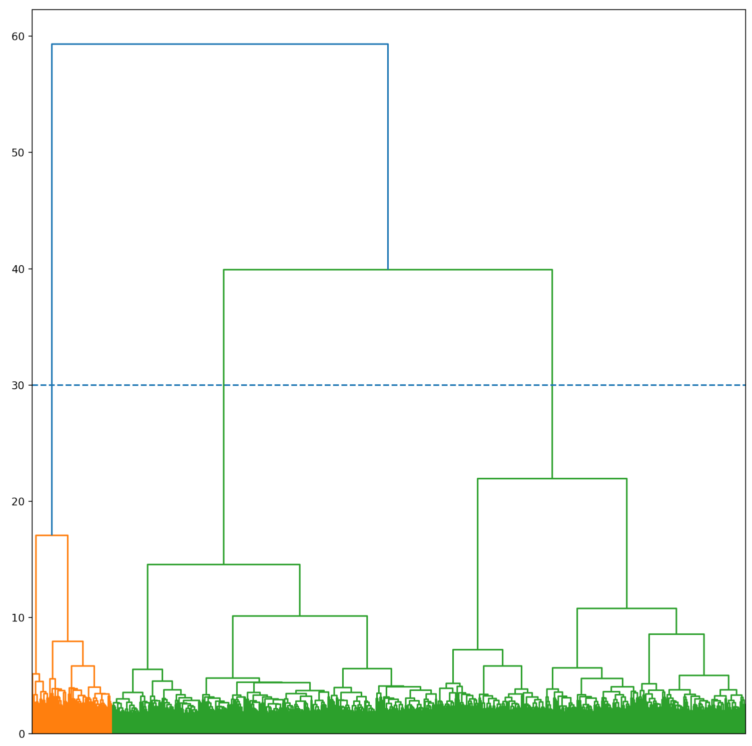

Figure 11. Figure shows the dendrogram plot of the correlation matrix obtained from users who rated at least 500 images. It can be seen from the plot that the number of distinct clusters visible below the threshold line is three.

Clusters	**Number of Users**
Cluster 1	327
Cluster 2	85
Cluster 3	344

Table 1. This table shows the number of raters from each cluster after carrying out clustering on the original correlation matrix from the list of filtered users.

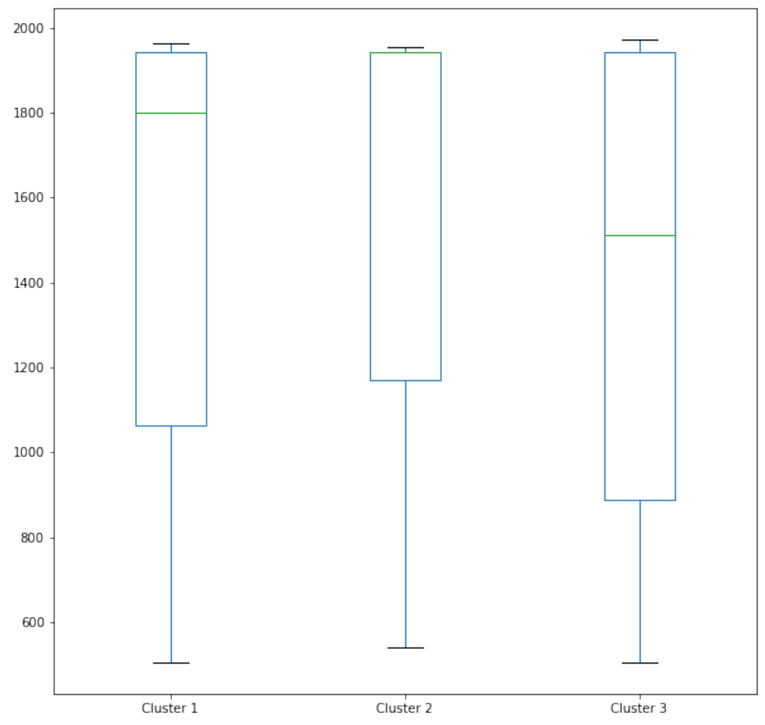

Figure 12. Box plot for the range of number of raters from each cluster

CHAPTER 5

Results and Analysis

This chapter gives an overview of the experiments conducted in this thesis and their results. The details discussed are discussed in separate sections - data preparation 5.1, creating baseline models 5.2, an ensemble of the clusters of workers 5.3, and an ensemble of models from ratings received by individual workers 5.4. The results obtained from each are visualized and also discussed in detail.

5.1 Data Preparation

As mentioned in chapter 4 the users considered for this experiment are those who rated at least 500 images. This is because there was no visible change in the distribution of the MOS obtained from all the users and users who had rated at least 500 images. The experiments were conducted in *python* programming language using *Anaconda's Jupyter Lab* environment. The core steps involved in this study were carried out in separate Jupyter files called *notebooks*.

Single Users Matrix:

As mentioned in chapter 4, using python's *pandas* library, the matrix $\boldsymbol{M_{ij}}$ was created such that the $\boldsymbol{i}$ is the worker on the horizontal axis and $\boldsymbol{j}$ are the images on the vertical axis. The train, validation, and test matrix were created by splitting the matrix $\boldsymbol{M_{ij}}$ on images. That way the number of workers for each set was the same and the images rated by the workers were allocated to separate sets. The train and test sets were split on a ratio of 90/10 and the validation set was created from 10% of images from the train set. Table 2 mentions the number of images belonging to each set.

Group of Users Matrix:

The clustering of users was carried out by using the hierarchical clustering method on the dissimilarity matrix as discussed in chapter 4. The distribution of workers in each cluster is re-mapped on the matrix $\boldsymbol{M}$ such that it is structured in the form $\boldsymbol{M_{(ki)j}}$, where $\boldsymbol{k}$ is the cluster, $\boldsymbol{i}$ are the workers belonging to that cluster and $\boldsymbol{j}$ are the images rated by the each $\boldsymbol{i}$th worker. The division of images in the data splits is the same as described in the previous sub-section (table 2).

Train	Validation	Test
8158	907	1008

Table 2. Train, Test and Validation split of 10073 stimuli.

5.2 Baseline Models

To carry out the experiment the baseline was established for each ensemble method implemented. This baseline was set by training models from all the user ratings on images from the $\boldsymbol{Mij}$ matrix, i.e. the users who had rated at least 500 images. The models trained and which ensemble method it is compared against are mentioned as follows:

- **Multi-Layer Perceptron (MLP):** This is the underlying model used in the bagging procedure of the ensemble for both the clusters of workers and the single worker's ensemble. The MLP was trained on both the individual ratings and the MOS of the images from the train set. The evaluation of the model was carried out on the individual ratings of images from the test set and the MOS of images from the test set, respectively. The details can be found in figure 13 and table 3. The parameters for the MLP were decided through trial and error, especially the size of the hidden layer in the model. The number of hidden layers was set to 1 only, in order to ensure the simplicity of the model (11). The MLP structure was implemented using *keras* library in Python programming language which uses *tensorflow* framework in backend. It was an optimal choice since it supports execution on GPU.

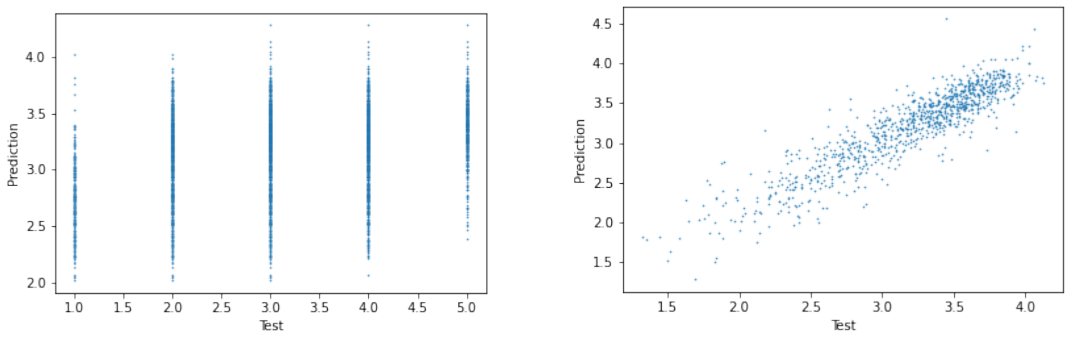

Figure 13. MLP Baseline for both discrete ratings (left) and MOS from the test set (right)

	SRCC	PLCC	MAE	RMSE
Individual Ratings	0.312	0.328	0.946	1.104
MOS	0.898	0.910	0.166	0.220

Table 3. This table shows the baseline scores from the MLP model.

- **Light Gradient boosting Machine (LGBM):** This is the underlying model used in the boosting ensemble method implemented for both clusters of workers and a single user's ensemble. As explained in chapter 3, the LGBM model is an ensemble model on its own. It uses decision trees as the base model and each tree is generated iteratively by assigning higher weightage to the high-error predictions from the previous iteration. The training of each LGBM was carried out on GPU since the tree generation process can be an expensive process despite being efficient as compared to other gradient boosting algorithms. Therefore the selection of parameters was set in regard to its optimal training on GPU. The parameters used are mentioned in the appendix (12). The baseline results are shown in figure 14 and table 4.

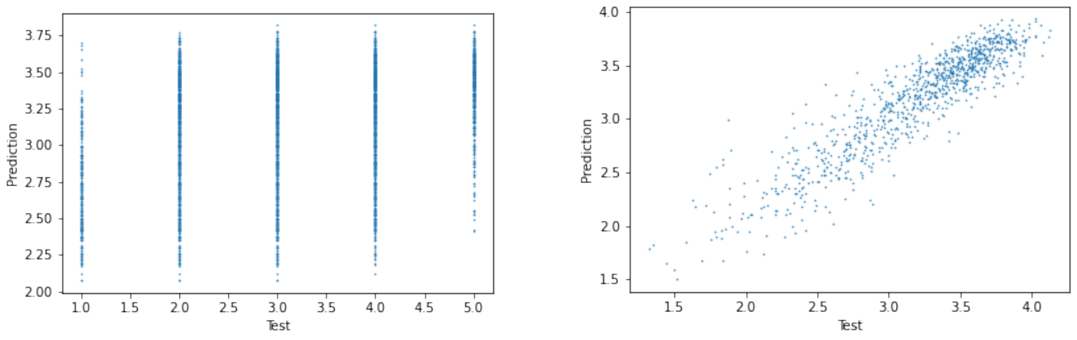

Figure 14. LGBM Baseline for both discrete ratings (left) and MOS from the test set (right).

	SRCC	PLCC	MAE	RMSE
Individual Ratings	0.312	0.322	0.949	1.102
MOS	0.906	0.917	0.158	0.210

Table 4. This table shows the baseline scores from the LGB model.

- **Stacking:** To set the baseline for stacking ensemble, the baseline stacking was implemented and trained on ratings from users who had rated at least 500 images from KonIQ-10k data set. The stacking ensemble as discussed in chapter 3 combines predictions from base models to train a meta-learner which makes the final set of predictions. As a rule of thumb, the choice of base models should be such that it is diverse and totally different from each other in terms of structure. Three base models were used to combine the predictions; MLP, LGBM and k-nearest Neighbors (k-NN). k-NN is a naive and basic model which computes predictions based on only distances and nothing else. No transformations or any other unique feature learning strategy is implemented under the hood but just the pre-defined distance metric.

  Another rule of thumb for Stacking is that the choice of meta-learner should be as simple as possible. Therefore a basic Linear Regression (LR) sufficed in this case. LR was trained on the predictions made by base models on the validation set. Predictions made on the validation set were used because data leakage from the base models is minimum with this approach. In the training of MLP and LGBM, the over-fitting was avoided by using the validation set. This means that model minimized its loss by evaluating loss on the validation set. Since no data points from the validation set were learned by the based models, therefore creating predictions on the validation set was an optimal choice for training the meta-learner. The evaluation scores from baseline stacking are demonstrated in figure 16 and table 5.

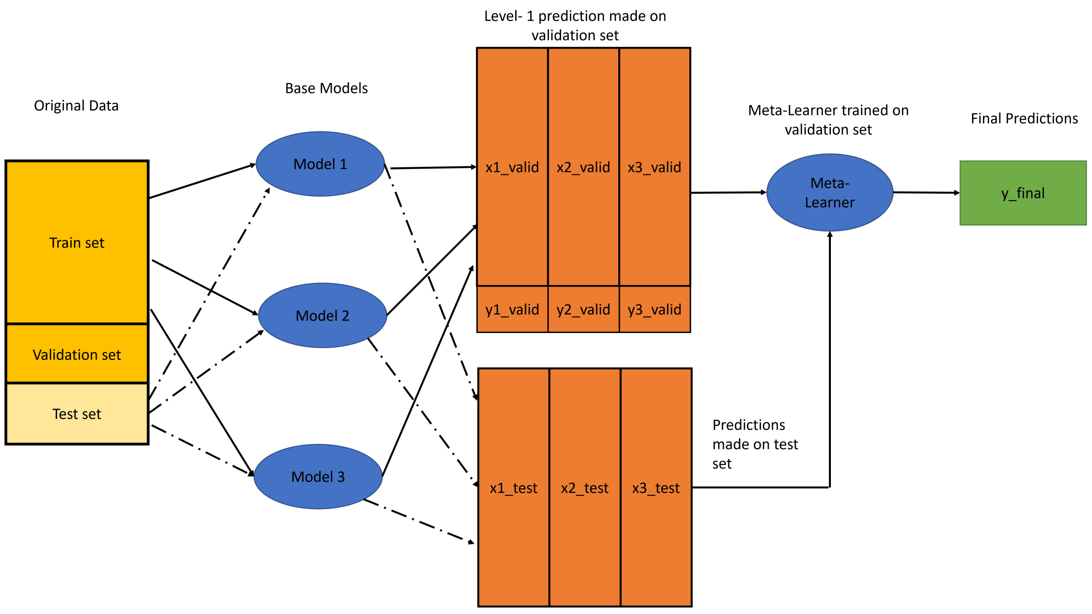

Figure 15. This figure illustrates the stacking structure implemented in this study. As can be seen, that validation set was used to combine predictions for the meta-level training. The test set was used to combine predictions from base models and serve as input to the meta-learner for a final set of predictions.

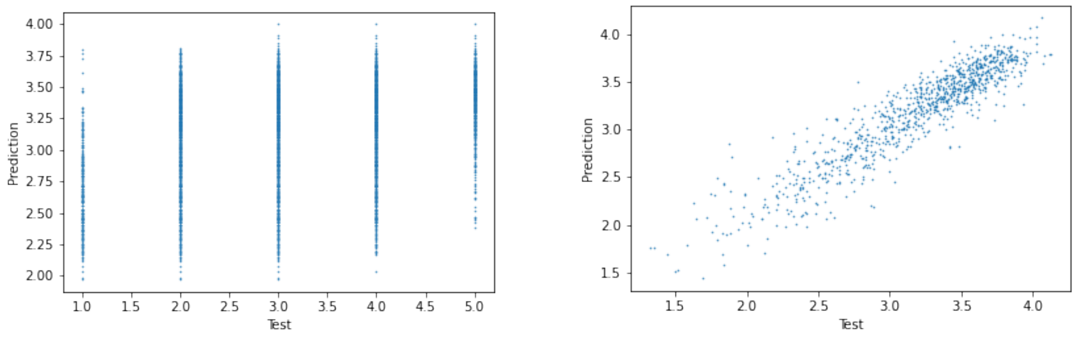

Figure 16. Stacking baseline for both discrete ratings (left) and MOS from the test set (right).

	SRCC	PLCC	MAE	RMSE
Individual Ratings	0.315	0.334	0.947	1.102
MOS	0.916	0.926	0.150	0.20

Table 5. This table shows the baseline scores from the stacking baseline model.

The structure set in the baseline models was used for both clusters of users and single-user ensembles. These parameters were selected based on trial and error and are not optimized due to resource constraints. The parameters selected for LGBM were those which were specific to the parameters required by underlying decision trees. Similarly, for MLP the parameter for hidden layer size was set to be equal to 2/3rd of the total number of workers considered for study. This choice was made by the standard rule of thumb when no benchmark can be established for selecting the best size for the hidden layer unless hyper-parameter optimization is implemented. For the k-nn model, the total number of neighbors specified was set to 5, since the distinct number of ratings from the dataset was 5. Again, the optimal set of parameters can be established through an exhaustive search of trial and error of models from a pre-defined hyperparameter space.

5.3 Group of Users

As mentioned in chapter 4, 3 groups of users were identified. Each group is trained, with the same set of models as described in the previous section. The models for each cluster are analyzed by evaluating scores on the images from the test set. The predictions made by each "cluster" model are evaluated with individual ratings from the test set belonging to the corresponding cluster. This step was carried out to understand the variation in performance from each cluster model. The details of the evaluation scores are as follows:

- **Cluster 1** The table 6 shows the variation of scores by implementing each model to ratings of workers from cluster 1.

Model	SRCC	PLCC	MAE	RMSE
MLP	0.310	0.334	0.853	1.008
LGBM	0.313	0.339	0.854	1.007
Stacking	0.315	0.340	0.854	1.006

Table 6. This table shows the scores of each model from Cluster 1. These scores are obtained by evaluating against individual ratings from the test set of cluster 1.

- **Cluster 2:** The table 7 shows the variation of scores by implementing each model to ratings of workers from cluster 2. During the evaluation, it was found that 48 images belonging to the train set did not have any ratings from workers from cluster 2. In the corresponding test set 5 images also had no ratings from workers belonging to cluster 2.

Model	SRCC	PLCC	MAE	RMSE
MLP	0.272	0.300	0.676	0.810
LGBM	0.249	0.276	0.657	0.843
Stacking	0.275	0.303	0.675	0.809

Table 7. This table shows the scores of each model from Cluster 2. These scores are obtained by evaluating predictions against individual ratings from the test set of cluster 2.

- **Cluster 3:** The table 8 shows the variation of scores by implementing each model to ratings of workers from cluster 3. It can be seen from table 1 that cluster 3 is the largest group of raters having a variety of ratings for all the images from each set. Therefore cluster 3 models has shown optimal results as compared to the rest.

Model	SRCC	PLCC	MAE	RMSE
MLP	0.436	0.456	0.845	1.000
LGBM	0.436	0.460	0.844	0.994
Stacking	0.441	0.464	0.841	0.992

Table 8. This table shows the scores of each model from Cluster 3. These scores are obtained by evaluating predictions against individual ratings from the test set of cluster 3.

The results above show the variation in scores of models corresponding to each cluster. It can be seen that cluster 2 had relatively weak SRCC and PLCC scores as compared to cluster 1 and cluster 3. This can be due to the fact that cluster 2 had the fewest number of raters. As compared to cluster 2, cluster 1 had relatively better scores and Cluster 3 had the highest scores. It can be deduced that the clusters that had the most number of raters, the corresponding model in the ensemble produced optimal results as compared to the rest.

To assess the overall scores of MOS of predictions against MOS of the test set, the bagging, boosting, and stacking method produced the results as shown in figure 17 and table 9.

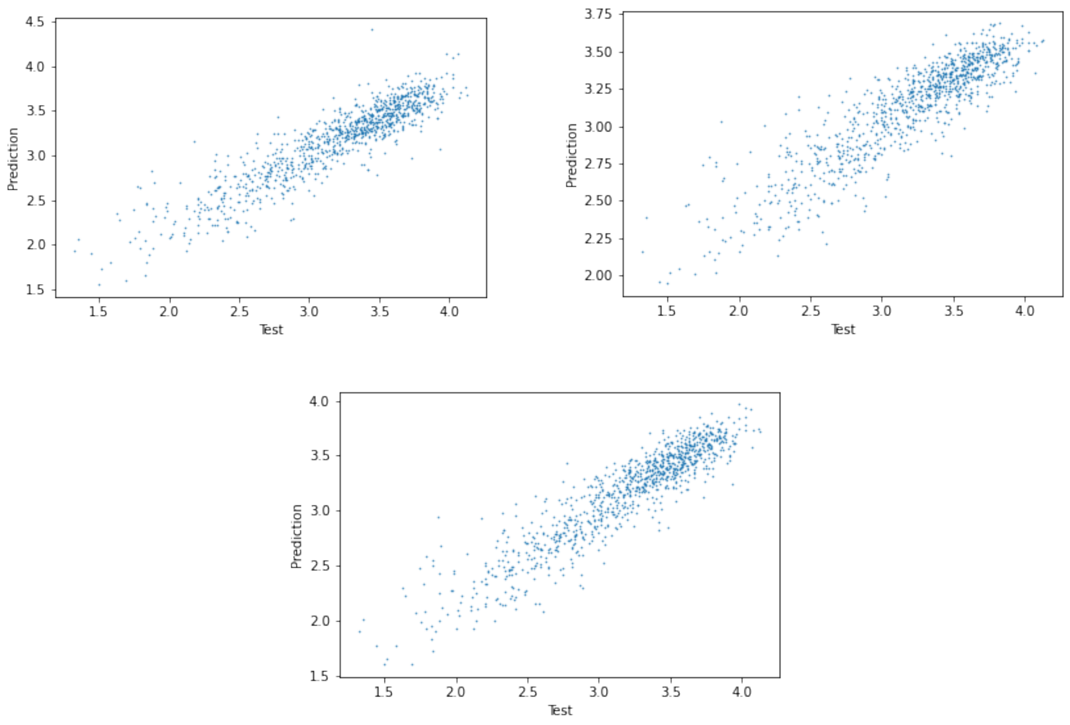

Figure 17. Evaluation of scores using ensemble methods from clusters against MOS from the test set. The figure on the top-left is the result obtained from the Bagging method, on the top-right is the figure which visualizes the performance of the Boosting method, and on the bottom is the figure which visualizes the predictions from the Stacking method.

Model	SRCC	PLCC	MAE	RMSE
Bagging	0.903	0.911	0.166	0.221
Boosting	0.887	0.896	0.218	0.275
Stacking	**0.916**	**0.924**	**0.157**	**0.207**

Table 9. This table shows the scores of each ensemble method implemented, combining predictions of base models belonging to each cluster. These scores are obtained by evaluating predictions against the MOS of images from the test set.

5.4 Single Users

In this section, I will discuss the results obtained by implementing ensemble methods on ratings from a single user. Each model in the ensemble was trained on the ratings produced by one single user as described earlier. To avoid the over-fitting of the models (both MLP and LGBM) the regularization was enabled by means of "early stopping". Early stopping enables the model to run for several number of iterations before it begins to over-fit. This technique uses the validation set to check for over-fitting by evaluating loss on each iteration. In this case, the images and their ratings given by the corresponding user from the validation set are used for the early stopping procedure. For the Stacking method, the stacking structure described in figure 16 was implemented for each user. There were 756 users in total and this strategy for implementing ensemble methods took the most time - approximately 21 hours of training time. The MOS of images from the test set was used to evaluate the aggregated predictions from each ensemble method. The results are visualized in figure 18 and mentioned in table 10.

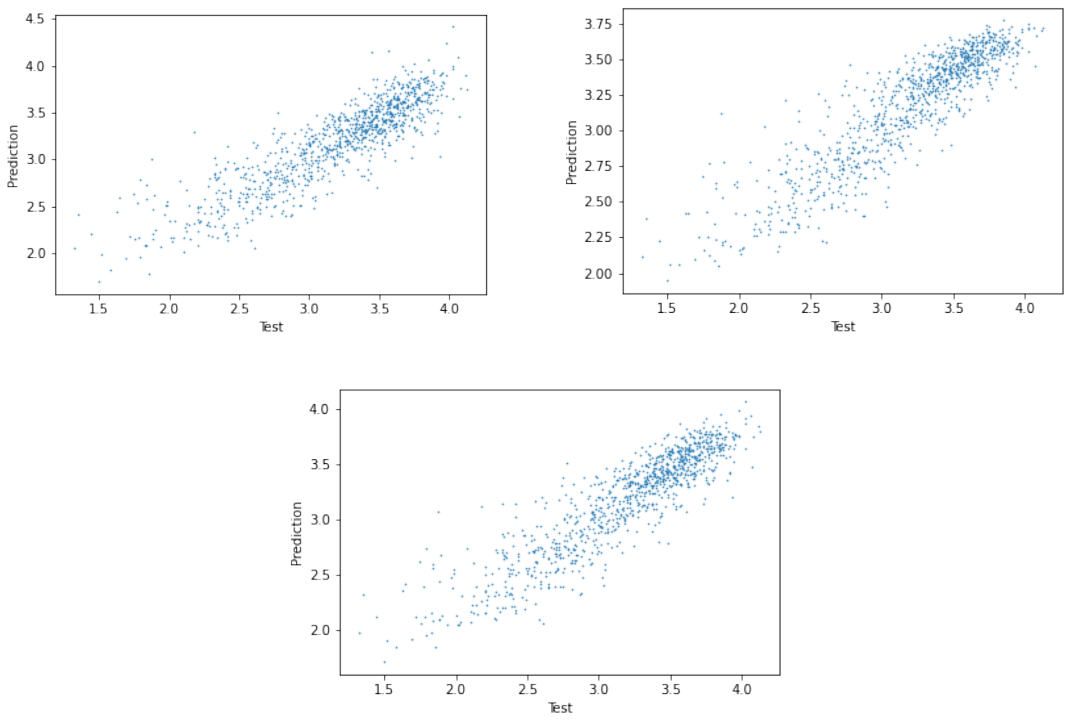

Figure 18. Evaluation of scores using ensemble methods from single users against MOS from the test set. The figure on the top-left is the result obtained from the Bagging method, on the top-right is the figure which visualizes the performance of the Boosting method, and on the bottom is the figure which visualizes the predictions from the Stacking method.

Model	SRCC	PLCC	MAE	RMSE
Bagging	0.880	0.882	0.185	0.248
Boosting	0.895	0.900	0.180	0.242
Stacking	**0.896**	**0.901**	**0.169**	**0.23**

Table 10. This table shows the scores of each ensemble method implemented, combining predictions of base models belonging to each worker. These scores are obtained by evaluating predictions against the MOS of images from the test set.

To conclude, it can be seen that an ensemble of models belonging to each cluster produced better SRCC and PLCC scores than the ensemble of models belonging to ratings from each user.

CHAPTER 6

Conclusion

In this chapter, I will conclude the thesis by discussing the limitations and future works building up on the possible improvements in critical areas of the thesis. In the first section, I will discuss the hurdles faced in the implementation, followed by possible suggestions and improvements in the following section. The final concluding remarks are written toward the end of this chapter.

6.1 Limitations

Limitations were faced in several critical areas of the experiment; mainly, it was on the grounds of resource constraints. The crucial limitation was the availability of the (Graphics Processing Unit) GPU and dedicated GPU RAM to train on the data size spanning several hundreds of megabytes. GPUs were required to speed up the training process of MLP and LGBM. LGBM further speeds up training with the availability of GPU. LGBM is efficient in training by implementing the leaf-growth strategy on GPUs.

An alternative approach was tested for implementing a custom-tailored approach of stochastic gradient boosting (Appendix 2). In this approach, the base learners were set to be the MLP such that each boosting ensemble would have at most 200 base learners. This approach enabled regularization by implementing a basic early stopping procedure after each iteration by evaluating loss on the validation set. However, it was computationally expensive since each model corresponding to the cluster or a single user would require at most 200 models for that specific implementation of boosting procedure in the ensemble. Therefore, an LGBM model was implemented as the ensemble for the boosting strategy.

6.2 Improvements & Future Work

This empirical study returned optimal results from the stacking ensemble method from the groups of users. Its evaluation on other IQA datasets can be another study area while exploring its limitations and possibilities in optimizing the results. This optimization can be done by implementing different state-of-the-art feature engineering approaches. However, further improvements and options for this experiment can be studied as follows:

- The clustering method discussed is based on the correlation coefficient matrix of users. Since each user rated different sets of images and different numbers of images, finding accurate correlation coefficients is hard. This is because there can be multiple confounding factors that can influence the relationship between similar users. Such as the type of images being rated, time spent on rating the images, and many more. Since the range of image ratings varies among users, this can also restrict the scope of correlation coefficients computed. Techniques like matrix factorization or graph-based methods might yield different results.
- The hyper-parameter optimization on models was not carried out. Optimizing each base model can further add the prediction variations, improving the results. With enough time and computational resources, this is another angle that can be explored.
- Bagging methodology was implemented by selecting one set of models. Different models can produce different sets of scores and would be a good baseline to measure the quality of this specific ensemble method. There is always a trade-off with time and computational complexity in experiments, and therefore may not be the optimal approach. However, bagging with different sets of models ranging from weak to high complexity can be another study angle for comparative study.
- For boosting, LGBM was used for the ensemble. Other algorithms like Extreme Gradient Boosting (XGB) and Category Boosting (CatBoost) can also be used to explore further the variation in SRCC and PLCC scores on multiple datasets.
- MLP, LGBM, and k-NN are base models in the stacking ensemble method. Models like Support Vector Machines (SVM), Random Forests (RF), and other conventional models can produce varying results. Stacking is a complex ensemble method with numerous variations in its structure; a comparative study using only the stacking method with variations in the underlying structure may return a different set of results.

While the possibilities to further this experiment are endless, the current implementation has returned promising SRCC and PLCC scores on the KonIQ-10k dataset. The baseline stacking ensemble obtained the highest SRCC score of **0.916** and PLCC score of **0.926**. This is better than the ensemble methods implemented for the group of workers and the single workers. The second-best scores were returned by the stacking ensemble of groups of users with SRCC score of **0.916** and PLCC score of **0.924**. All in all, the stacking ensemble method has returned optimal results among bagging, boosting, and stacking. This is because the stacking method can better generalize by reducing bias and variance in learning patterns from the data set [Dor20].

Bibliography

[AA19] Nisar Ahmed and Hafiz Muhammad Shahzad Asif. Ensembling convolutional neural networks for perceptual image quality assessment. In *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, pages 1–5. IEEE, 2019.

[BT] RECOMMENDATION ITU-R BT. Specifications and alignment procedures for setting of brightness and contrast of displays.

[CAY$^+$20] Andrei Chubarau, Tara Akhavan, Hyunjin Yoo, Rafał K. Mantiuk, and James H. Clark. Perceptual image quality assessment for various viewing conditions and display systems. *Electronic Imaging*, 2020.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 08 2016.

[Cha95] Sugato Chakravarty. Methodology for the subjective assessment of the quality of television pictures. 1995.

[CJC$^+$07] Ji-Hwan Choe, Tae-Uk Jeong, Hyunsoo Choi, Eun-Jae Lee, Sang-Wook Lee, and Chul-Hee Lee. Subjective video quality assessment methods for multimedia applications. *Journal of Broadcast Engineering*, 12, 03 2007.

[CYKL21] Manri Cheon, Sung-Jun Yoon, Byungyeon Kang, and Junwoo Lee. Perceptual image quality assessment with transformers. *CoRR*, abs/2104.14730, 2021.

[D$^+$02] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.

[DMWS21] Keyan Ding, Kede Ma, Shiqi Wang, and Eero Simoncelli. Comparison of full-reference image quality models for optimization of image processing systems. *International Journal of Computer Vision*, 129, 01 2021.

[Dor20] Shayan Doroudi. The bias-variance tradeoff: How data science can inform educational debates. *AERA open*, 6(4):2332858420977208, 2020.

[DSS$^+$22] Shahi Dost, Faryal Saud, Maham Shabbir, Muhammad Gufran Khan, Muhammad Shahid, and Benny Lövström. Reduced reference image and video quality assessments: review of methods. *EURASIP Journal on Image and Video Processing*, 2022, 01 2022.

[Fri02] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 2002.

[GB16] Deepti Ghadiyaram and Alan C. Bovik. Massive online crowdsourced study of subjective and objective picture quality. *IEEE Transactions on Image Processing*, 25(1):372–387, 2016.

[Gni13] Zenon Gniazdowski. Geometric interpretation of a correlation. 7:27–35, 09 2013.

[GZY$^+$13] Ke Gu, Guangtao Zhai, Xiaokang Yang, Wenjun Zhang, and Min Liu. Subjective and objective quality assessment for images with contrast change. In *2013 IEEE International Conference on Image Processing*, pages 383–387, 2013.

[GZYZ14] Ke Gu, Guangtao Zhai, Xiaokang Yang, and Wenjun Zhang. Hybrid no-reference quality metric for singly and multiply distorted images. *IEEE Transactions on Broadcasting*, 60(3):555–567, 2014.

[HFH21] Dounia Hammou, Sid Ahmed Fezza, and Wassim Hamidouche. Egb: Image quality assessment based on ensemble of gradient boosting. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 541–549, 2021.

[HGS19] Vlad Hosu, Bastian Goldlucke, and Dietmar Saupe. Effective aesthetics prediction with multi-level spatially pooled features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9375–9383, 2019.

[HHK12] Xiaodi Hou, Jonathan Harel, and Christof Koch. Image signature: Highlighting sparse salient regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):194–201, 2012.

[HLSS20] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe. Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment. *IEEE Transactions on Image Processing*, 29:4041–4056, 2020.

[Hub92] Peter J Huber. Robust estimation of a location parameter. *Breakthroughs in statistics: Methodology and distribution*, pages 492–518, 1992.

[JG20] Haoyu Chen Xiaoxing Ye Jimmy Ren Chao Dong Jinjin Gu, Haoming Cai. Pipal: a large-scale image quality assessment dataset for perceptual image restoration. In *European Conference on Computer Vision (ECCV) 2020*, pages 633–651, Cham, 2020. Springer International Publishing.

[JMMB12] Dinesh Jayaraman, Anish Mittal, Anush Moorthy, and Alan Bovik. Objective quality assessment of multiply distorted images. pages 1693–1697, 11 2012.

[KMF$^+$17a] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[KMF$^+$17b] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[KS05] Ivan Kopilovic and Tamas Sziranyi. Artifact reduction with diffusion preprocessing for image compression. *Optical Engineering*, 44(2):1 – 14, 2005.

[KYLD14] Le Kang, Peng Ye, Yi Li, and David Doermann. Convolutional neural networks for no-reference image quality assessment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1733–1740, 2014.

[KZG$^+$17] Jongyoo Kim, Hui Zeng, Deepti Ghadiyaram, Sanghoon Lee, Lei Zhang, and Alan Bovik. Deep convolutional neural models for picture-quality prediction: Challenges and solutions to data-driven image quality assessment. *IEEE Signal Processing Magazine*, 34:130–141, 11 2017.

[LPFY16] Yuming Li, Lai-Man Po, Litong Feng, and Fang Yuan. No-reference image quality assessment with deep convolutional neural networks. In *2016 IEEE International Conference on Digital Signal Processing (DSP)*, pages 685–689. IEEE, 2016.

[LWK$^+$15] Joe Yuchieh Lin, Chi-Hao Wu, Ioannis Katsavounidis, Zhi Li, Anne Aaron, and C.-C. Jay Kuo. Evqa: An ensemble-learning-based video quality assessment index. In *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, 2015.

[LZW+15] Chaofeng Li, Yu Zhang, Xiaojun Wu, Wei Fang, and Li Mao. Blind multiply distorted image quality assessment using relevant perceptual features. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4883–4886, 2015.

[MAET15] Sadaaki Miyamoto, Ryosuke Abe, Yasunori Endo, and Jun-ichi Takeshita. Ward method of hierarchical clustering for non-euclidean similarity measures. In *2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pages 60–63, 2015.

[MB11] Anush Krishna Moorthy and Alan Conrad Bovik. Blind image quality assessment: From natural scene statistics to perceptual quality. *IEEE Transactions on Image Processing*, 20(12):3350–3364, 2011.

[ML11] Fionn Murtagh and Pierre Legendre. Ward's hierarchical clustering method: Clustering criterion and agglomerative algorithm. 11 2011.

[ML17] Mengzhu Ma and Chaofeng Li. Blind multiply distorted image quality assessment using an ensemble random forest. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5. IEEE, 2017.

[MMB12] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012.

[MMP12] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2408–2415, 2012.

[MSB13] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013.

[PGV+18] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: Unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6639–6649, Red Hook, NY, USA, 2018. Curran Associates Inc.

[PJ14] Margaret H Pinson and Lucjan Janowski. A new subjective audiovisual & video quality testing recommendation. *An Era of Change*, page 51, 2014.

[PJI$^+$15] Nikolay Ponomarenko, Lina Jin, O. Ieremeiev, Vladimir Lukin, Karen Egiazarian, J. Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, and C.-C. Jay Kuo. Image database tid2013: Peculiarities, results and perspectives. *Signal Processing: Image Communication*, 30:57–77, 01 2015.

[PSZ$^+$17] Da Pan, Ping Shi, Dixiu Zhong, Ming Hou, Zefeng Ying, and Sizhe Fu. Weighted ensemble learning prediction for blind symmetrically distorted stereoscopic images. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1297–1301. IEEE, 2017.

[RP07] Niall Rooney and David Patterson. A fusion of stacking with dynamic integration. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 2844–2849, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99, 2000.

[SBC11] Michele A. Saad, Alan C. Bovik, and Christophe Charrier. Dct statistics model-based blind image quality assessment. In *2011 18th IEEE International Conference on Image Processing*, pages 3093–3096, 2011.

[SHH$^+$16] Dietmar Saupe, Franz Hahn, Vlad Hosu, Igor Zingman, Masud Rana, and Shujun Li. Crowd workers proven useful : a comparative study of subjective video quality assessment. In *QoMEX 2016 : 8th International Conference on Quality of Multimedia Experience*, 2016.

[SR18] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[SSB06a] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

[SSB06b] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

[TSF$^+$16] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: the new data in multimedia research. *Commun. ACM*, 59(2):64–73, 2016.

[Var22] Domonkos Varga. No-reference image quality assessment with convolutional neural networks and decision fusion. *Applied Sciences*, 12(1), 2022.

[VQE] Official Web Site VQEG. Subjective assessment of standard definition digital television (sdtv) systems. *Rec. ITU-R BT. 1129-2*.

[WB11] Zhou Wang and Alan C. Bovik. Reduced- and no-reference image quality assessment. *IEEE Signal Processing Magazine*, 28(6):29–40, 2011.

[WBL02] Zhou Wang, Alan C. Bovik, and Ligang Lu. Why is image quality assessment so difficult? In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages IV–3313–IV–3316, 2002.

[WBSS04] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[WZ04] G.I. Webb and Z. Zheng. Multistrategy ensemble learning: reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991, 2004.

[YLKT14] J. Yan, S. Lin, S. Kang, and X. Tang. A learning-to-rank approach for image color enhancement. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2987–2994, Los Alamitos, CA, USA, jun 2014. IEEE Computer Society.

CHAPTER A

Appendix

A.1 Dot product expressed as Pearson coefficient

The similarity between user ratings was calculated to compute the distinct number of clusters of workers who gave similar ratings. To compute dissimilarity from similarity, the cosine theorem was used. Therefore, it is imperative to understand with proof why it is used in the grouping strategy.

For this section, I will mention Zenon et al [Gni13] literature in which he shows how Pearson coefficient correlation is equal to the cosine of the angle between random variables. In multi-dimensional space the vector $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_n})$, has the euclidean norm $||\boldsymbol{x}||$ which is also the magnitude of the vector $\boldsymbol{x}$. This Euclidean norm can be represented by the equation (10):

$$||\boldsymbol{x}|| = \sqrt{\sum_{i=1}^{n} \boldsymbol{x_i^2}}. \tag{10}$$

Hence, in a plane, the dot product of two vectors $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_n})$ and $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots \boldsymbol{y_n})$ can be represented as (11). This dot product is also called the scalar product of two vectors.

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} \boldsymbol{x_i y_i}. \tag{11}$$

This dot product can simultaneously be expressed as a cosine function (12) where $\mathbf{cos}(\boldsymbol{x}, \boldsymbol{y})$ is the cosine value of the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

$$\boldsymbol{x} = ||\boldsymbol{x}|| \cdot ||\boldsymbol{y}|| \cdot \mathbf{cos}(\boldsymbol{x}, \boldsymbol{y}) \tag{12}$$

Thereby with simplification, the $\mathbf{cos}(\boldsymbol{x}, \boldsymbol{y})$ can be expressed as:

$$\mathbf{cos}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \tag{13}$$

To understand further how Pearson Correlation is related to (13) we take a look at the derivation of the Pearson Correlation coefficient. Pearson Correlation $\boldsymbol{\rho}$ can be defined as:

$$\boldsymbol{\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}} \tag{14}$$

where $\boldsymbol{Cov(X,Y)}$ is the covariance between two variables X and Y, and $\boldsymbol{\sigma_X}$ and $\boldsymbol{\sigma_Y}$ are the standard deviation between $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively. Covariance measures the relationship between any two variables.

Furthermore, covariance can be expressed as:

$$\boldsymbol{\sigma_{X,Y} = E\left[(X - \mu_X)(Y - \mu_Y)\right].} \tag{15}$$

where $\boldsymbol{\mu}$ is the mean of $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively.

Henceforth, from the (14), Pearson Correlation Coefficient can be expressed in the form (16):

$$\boldsymbol{\rho_{X,Y} = \frac{\sigma_{X,Y}}{\sigma_X \sigma_Y} = \frac{E\left[(X - \mu_X)(Y - \mu_Y)\right]}{\sqrt{E\left[(X - \mu_X)^2\right] E\left[(Y - \mu_Y)^2\right]}}.} \tag{16}$$

which can be further simplified into the form:

$$\boldsymbol{\rho_{X,Y} = \frac{\sum_{i=1}^{n}\left[\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)\right]}{\sqrt{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2}}.} \tag{17}$$

The correlation coefficient of variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ can also be considered as the covariance of variables being standardized. Therefore from standardized form

$$\boldsymbol{x_i = X_i - \bar{x}}$$

(17) can be expressed as:

$$\boldsymbol{\rho_{X,Y} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}} \tag{18}$$

From (18) in comparison with (13) it can be seen that numerator can be considered as the scalar product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and the denominator is the product of the magnitude of each. Hence it can be proved that

$$\boldsymbol{\rho_{X,Y} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} = \frac{x \cdot y}{\|x\| \cdot \|x\|} = \cos(x,y)} \tag{19}$$

correlation coefficient and the dot product of two vectors are identical to each other.

A.2 Algorithms

Algorithm 1 Stochastic Gradient Boosting

```
$F_0(x) = \arg\min_\gamma \sum_{i=1}^{N} \Psi(y_i, \gamma)$
for $m = 1$ to $M$ do
    $\pi(i)\}_1^N = \text{rand\_perm}\{i\}_1^N$
    $\tilde{y}_{\pi(i)m} = -\left[\frac{\partial\Psi(y_{\pi(i)}, F(x_{\pi(i)}))}{\partial F(x_{\pi(i)})}\right]_{F(x)=F_{m-1}(x)}, i = 1, \tilde{N}$
    $\{R_{lm}\}_1^L = L-$ terminal node tree $\left(\{\tilde{y}_{\pi(i)m}, x_{\pi(i)}\}_1^{\tilde{N}}\right)$
    $\gamma_{lm} = \arg\min_\gamma \sum_{x_{\pi(i)} \in R_{lm}} \Psi\left(y_{\pi(i)}, F_{m-1}(x_{\pi(i)}) + \gamma\right)$
    $F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_{lm} 1(x \in R_{lm})$
end
```

Algorithm 2 Stochastic Gradient Boosting Customized

```
$F_0(x) = \arg\min_\gamma \sum_{i=1}^{N} \Psi(y_i, \gamma)$
for $m = 1$ to $M$ do
    $\pi(i)\}_1^N = \{x_i \mid i \in D\}_1^N$
    $\tilde{y}_{\pi(i)m} = -\left[\frac{\partial\Psi(y_{\pi(i)}, F(x_{\pi(i)}))}{\partial F(x_{\pi(i)})}\right]_{F(x)=F_{m-1}(x)}, i = 1, \tilde{N}$
    $F_m(x) = F_{m-1}(x) + 1(x \in R_{lm})$
end
```

A.3 Model Parameters

Parameter	Value	Description
`Size of hidden layer`	523	This is the size of the hidden layer which was set to be two-thirds of the total number of users in this study.
`Number of Hidden Layers`	1	For a simple model structure, only one hidden layer was used for the experiment.
`Optimizer`	Adaptive Momentum	Gradient descent optimizer to minimize the loss.
`Output Units`	1	Expected output of the model is a single continuous value.
`Patience`	10	Stop training if there is no improvement after 10 iterations. Useful in avoiding over-fitting
`Loss Function`	0.05	Error metric for evaluating learning of the model

Table 11. This table shows the selection of hyper-parameters for the MLP model.

Parameter	Value	Description
`boosting_type`	`gbdt`	Use gradient boosting trees for generating weak models
`Objective`	`regression`	Use l2 regularizer for regression task.
`learning_rate`	0.05	Learning rate or also known as "shrinkage" is the tuning parameter for minimizing the loss.
`verbose`	0	Control console output during model training.
`max_depth`	0.05	Limit the maximum depth of the tree
`num_leaves`	128	Limit maximum number of leaf nodes in one tree.
`max_bin`	128	Maximum number of bins to encapsulate features from dataset.
`num_iteration`	1000	Boosting iterations in the model or total number of weak learners in the model.
`device`	gpu	Parameter to speed up training process by utilising GPU.

Table 12. This table shows the selection of hyper-parameters for the LGB model.