Facial Emotion Detection By Machine Learning

Final report

Cheuk Hong lp

Jing Li

Shufang Wen

Wenbin Wang

Department of Applied Data Science, San Jose State University

DATA 298A: MSDA Project I

Instructor: Dr. Jerry Gao

Date: 02/15/2023

Abstract

Rapid machine-learning technology has made it feasible to accomplish human emotion recognition by computer. Detecting human emotion is essential for modern artificial intelligent systems, among many other fields, since it's informative in making informed decisions. The focus of the study is to establish an accurate and practical model to process feature extraction of human face images and eventually detect facial emotions with high accuracy. Existing research used deep learning to detect facial emotion without data augmentation, and they have used the entire image to implement the models. The Fusion is a famous deep learning model that can achieve the highest accuracy rate with 91% in 2017. Another favorite deep learning model is the Deep Boltzmann Machine, which accomplished an 89.3% of accuracy rate. However, this project used machine learning algorithms (i.e., Logistic Regression, Naive Bayes, K Nearest Neighbors, Support Vector Machine, and Random Forest) to solve the problem. Also, the HOG face detector is applied to crop the human facial partition to reduce background noise. Data augmentation, resize, grayscale, normalization, and dimension reduction are applied to enhance performance. The accuracies of our models are 75%. SVM performs the best because it handles high-dimension data. However, the distinction between the human emotion of anger and sadness is minor, which gives a challenge to the classification model created in this project. Hence in the future, more advanced algorithms for model training will be applied and more features such as electric signal (EEG) will be considered.

Content

Chapter 1 Introduction

Project Background and Executive Summary

Project Background, Motivations, Target Problem, Goals, Needs and Importance

Over the past decades, machine learning and artificial intelligence (AI) have been developing rapidly. They are applied in many domains, including emotion expression recognition. People express feelings such as joy, hate, fear, and grief on their faces, which can be glimpsed by either images or videos. Classifying the emotions from the original materials is needed so that computers can simulate reactions to humans. Moreover, it is valuable for business decision-making, commercial promotion, security detection systems, and so on. The accuracy of recognizing human faces once was constrained by lighting, face position, and background noise. However, many researchers have explored the field of emotion intelligence detection, accomplishing prediction not only from static images but also from real-time video.

Psychologists often find it hard to efficiently help their patients through talking. Without efficient interpretation of patients' facial reactions, it would be difficult for psychologists to build an accurate patient profile or apply the correct mental approach to help patients restore mental health and ease their pains. Another case of patients is people who've been affected by autism, a high-accurate emotion detection system would help identify people with autism, and they could receive help from specialists at an earlier time. On the law enforcement side, many times, it's challenging for police officers to obtain from potential suspects beneficial or crucial information that could help stop further crimes or help victim families. In another law enforcement case, detectives managed to identify the criminal picture at a much younger age. Still, they could not use it to release an all-points bulletin (APB) of arrest, and the facial emotion recognition (FER) system could help in such a case with the model that could recognize the same facial features for

the same people at different ages. Another area for FER is autonomous driving or commercial fleet management, and many accidents happen due to drivers falling asleep or getting distracted. Accidents could greatly be avoided with the aid of automatic facial recognition for raising warnings in time.

The problem is that human emotions are very informational in interpersonal relations, but so far, they lack efficient methods to recognize and utilize this information effectively. This research is trying to solve the problem of utilizing machine learning technology to identify human emotions.

The goal is to establish an accurate and practical model to do feature extraction of human face images and eventually detect facial emotions with accuracy. It bears great practical importance in helping psychologists better evaluate their patients and, therefore, better help them to recover, and also in helping police officers fight crime by helping obtain more useful information via face emotion analysis during crime interrogation.

Project Approaches and Methods

The data required for this project are facial images with emotions. Web scrapping from the website was used to collect the image data. Pexels.com can provide royalty-free images and offer free APIs. The dataset contains three kinds of facial emotion pictures, happy, sad, and angry. Then, OpenCV was applied to transform images, and PCA was used to reduce the size of the features to prevent overfitting.

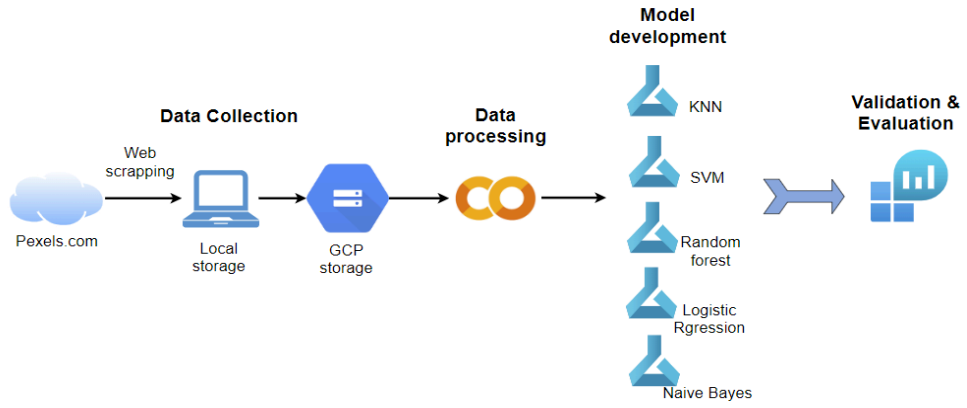
Lahaw et al. (2018) have presented an approach by comparing several models, including Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Wavelet Transform (DWT), Principal Component Analysis (PCA), and Support Vector Machine (SVM). This experimentation is based on the AT&T database. The research result achieved 96% of

accuracy by executing a hybrid method, which is LDA or PCA for dimension reduction, and the SVM method for classification. Sabri et al. (2018) showed their research results by resembling three kinds of machine learning models, which are Multi Linear Perceptron (MLP), Naive Bayes (NB), and SVM classifiers to classify the human face. NB attained high accuracy of 93.16%. The outcome indicated a comprehensive process of SVM and MLP got better precision (pp. 116-120).

After investigating other researchers' work, several widely used and contain good-performance machine learning models were selected as this project's approach. Five machine learning models including SVM, KNN, RF, NB, and Logistic Regression were implemented to detect facial emotions. Then, the F1 score, accuracy, precision, recall, etc., were applied to measure the models' performances. Figure 1 shows the overview of the project workflow.

Figure 1

High-level workflow



Note. This is an overview of high-level project workflow.

Expected Project Contributions and Applications

Patients might refuse to present their feelings due to some personal reason. In these circumstances, this project is beneficial for psychological counseling since psychologists can apply these models to recognize patients' emotions. Recognition of a human's true feelings also is helpful for interrogation since police can use the predicted emotion to figure out whether the criminal suspect is lying or not. Emotion detection applications can also be introduced to AI interaction, like smart home facilities or even smart cities.

Project Requirements

Functional Requirements

San Jose State University provides authorization to students for one year of Google Cloud access. Hence, this is the primary cloud platform for the project since Google Cloud is the central hub for storing images while providing high-performance GPUs to accelerate specific tasks such as running machine learning models and analyzing and evaluating data. To run Jupyter Notebook in Google Colab and to access Google Cloud storage, the authentications required to enable API such as Compute Engine API, Cloud Deployment Manager V2 API, Cloud Runtime Configuration API, and Google Cloud Storage API ("Google APIs Explorer," n.d.), and to purchase both GCE and VM applications from GCP marketplace. In addition, Pexels.com provide royalty-free images and offer free APIs for the dataset. Therefore, it does not have any legal or regulatory requirements for the project.

Pexels.com API would require authorization, and access is provided instantly after requesting an API key. Credit is needed to give to the photographers when possible or to show a prominent Pexels link or logo. There is a rate limit of 200 requests per hour and 20,000 requests per month. After collecting images, testing and training data are labeled for all the supervised

machine learning models to let all models work properly and classify facial emotion images accurately.

AI Requirements

Five machine learning models were implemented in the paper: SVM, KNN, RF, NB, and Logistic Regression, while using established deep learning models from artificial intelligence-powered libraries such as the Deepface library to label image categories and to test the accuracy against the machine learning models. On the other side, training and testing data were needed to test the functionalities of the data. Then, after using the models' hyperparameters to set the best parameter, such as finding the best K for the KNN model, data can be compared with the new dataset to measure the performance.

Random Forest is a supervised learning model and is an ensemble of a decision tree that contains many single decision trees. The reason to use this model is that it can handle high-dimension features very well since it only uses subsets of data from the original data to test the model and because the images contain extensive features in the dataset (Rigatti et al., 2017, pp. 31-39). The implementation of the model:

1. Images were labeled, resized, recolored to grayscale, and categorized into two sets: training sets and testing sets.
2. The data were randomly selected into subsets.
3. Each parameter, in this case, the number of trees, was tested to find the best parameter for the model. After that, the model was trained.
4. The class prediction was determined by a majority of votes from each tree.

K Nearest Neighbor (KNN) is one of the simplest and most fundamental supervised learning models for solving classification and regression problems. With the given data, the

KNN model helps find the K closet point by calculating the distance between two data points to get the predicted value (Guo et al., 2003, pp. 986–996). Unfortunately, KNN does not have a training process, so it cannot optimize effectiveness measures. In addition, it does not work nicely with high-dimensional and large datasets. (Hall et al., 2008, pp. 2135-2152). Despite lacking high accuracy of classifier image/multiple features, KNN was chosen as a comparison object. The implementation of the model:

1. Images were labeled, resized, recolored to grayscale, and split into training and testing sets.
2. Build a KNN model with training data.
3. For each parameter, in this case, the number of K was adjusted and tested to find the best parameter for the model.
4. Testing data was used to predict the accuracy of the class prediction.

SVM is one of the effective supervised learning models and is widely used for classification and regression analysis. For the purpose of maximizing the distance between two classes, SVM maps 2D training data into high-dimensional spaces. The process is finding a hyperplane in space to divide training data more accurately. The kernel function is implemented for non-linear classification. The parameters chosen can affect the precision in either binary or multi-class classification. Swinkels et al. (2017) have used a multi-class SVM instead of binary to detect various human emotions, and the research result showed the efficiency of non-linear classification (pp. 86-92). SVMs are helpful for image classification. Many practical developments showed that it can gain significantly better accuracy than traditional classifier methods after several appropriate feedback. It also has been devoted to image segmentation, including a modified privileged approach version of SVM (Barghout, 2015, pp. 285-318). To

summarize, SVM was chosen for the project because it has been widely applied in emotion detection. This model was chosen to build an adaptive system based on such massive outstanding work of other researchers. The implementation of the model:

1. Images were labeled, resized, recolored to grayscale, and split into training and testing sets.
2. Build an SVM model with training data.
3. Adjust the kernel type (e.g., linear, poly, or RBF).
4. Change margin (e.g., soft or hard margin).
5. Testing data was used to predict the accuracy of the class prediction.

Logistic Regression is an error-based learning algorithm. It is widely used for classification problems (Kelleher et al., 2015, p. 383). Also, due to its ability to identify the most reliable linear combination of variables with the highest likelihood, it is effective and strong when dealing with categorizing target variables (Stoltzfus, 2011, pp. 1099-1104). Since the project goal is to detect facial emotion from images, and the target variable is a categorical variable, Logistic Regression is a good fit. Regularization impacts the performance of the model, so different regularization strengths were used to optimize the model (Salehi et al., 2019). The implementation of the model:

1. Images were labeled, resized, recolored to grayscale, and split into training and testing sets.
2. Build the Logistic Regression model with the training data.
3. Adjust the regularization strengths to optimize the model.
4. Testing data was used to predict the accuracy of the class prediction.

Naive Bayes is a probability-based algorithm, and it is widely used as the baseline model for classifying categorical target variables. Also, it is simple to train and resistant to the dimensionality curse (Kelleher et al., 2015, p. 301). Compared to other models, it is less computationally complex and the accuracy for it to detect facial emotion from the images is not low (Ayache & Ali, 2020, pp. 267-275). It has a hyperparameter (var_smoothing). It is a portion of the largest variance of all characteristics that contributed to the variance to ensure computation stability. The default value is 1^{-9}. The best accuracy usually did not occur when it was at the default value (Kuruvila et al., 2020, pp. 452-457). This model was chosen to serve as a baseline model to obtain baseline accuracy. The implementation of the model:

1. Images were labeled, resized, recolored to grayscale, and split into training and testing sets.
2. Build the Naive Bayes model with the training data.
3. Adjust the var_smoothing to optimize the model.
4. Testing data was used to predict the accuracy of the class prediction.

Data Requirement

As per the data requirements, non-corrupted, high-quality, and royalty-free images in JPG format would be required, and need to make no duplicate images have been uploaded to the cloud. The goal is to collect 500 to 2,000 images from each category from Pexels.com. Pexels.com would require authorization, and Pexels.com API would be 200 per hour and 20,000 per month. The dataset has a total of 2,000 to 8,000 images that contain three different emotions: happy, angry, and sad, with balanced data to ensure classification performance is accurate. For this stage, the dataset contains 160 images in jpg format. Examples of the dataset and the sample images are shown in Figures 2, 3, and 4.

Figure 2

Example of the Current Dataset

1happy266004.jpg	Today at 5:24 PM	4 MB JPEG image
1happy459957.jpg	Today at 5:24 PM	547 KB JPEG image
1happy755280.jpg	Today at 5:25 PM	1.2 MB JPEG image
1happy773371.jpg	Today at 5:24 PM	1.6 MB JPEG image
1happy774866.jpg	Today at 5:23 PM	6.7 MB JPEG image
1happy775358.jpg	Today at 5:24 PM	383 KB JPEG image
1happy799420.jpg	Today at 5:23 PM	14.1 MB JPEG image
1happy839633.jpg	Today at 5:24 PM	1.2 MB JPEG image
1happy977539.jpg	Today at 5:25 PM	12 MB JPEG image
1happy1024311.jpg	Today at 5:23 PM	19 MB JPEG image
1happy1068205.jpg	Today at 5:24 PM	306 KB JPEG image
1happy1130624.jpg	Today at 5:24 PM	16.1 MB JPEG image
1happy1133721.jpg	Today at 5:24 PM	5.1 MB JPEG image
1happy1149022.jpg	Today at 5:25 PM	14.9 MB JPEG image
1happy1181424.jpg	Today at 5:24 PM	12.3 MB JPEG image
1happy1181686.jpg	Today at 5:24 PM	12.6 MB JPEG image
1happy1181742.jpg	Today at 5:24 PM	4.9 MB JPEG image
1happy1222271.jpg	Today at 5:25 PM	9.9 MB JPEG image
1happy1239288.jpg	Today at 5:24 PM	16.4 MB JPEG image
1happy1239291.jpg	Today at 5:23 PM	15.4 MB JPEG image
1happy1266193.jpg	Today at 5:26 PM	25 MB JPEG image
1happy1320701.jpg	Today at 5:24 PM	5.6 MB JPEG image
1happy1524105.jpg	Today at 5:24 PM	6.2 MB JPEG image
1happy1679777.jpg	Today at 5:40 PM	14.2 MB JPEG image
1happy1786258.jpg	Today at 5:26 PM	16.4 MB JPEG image
1happy1805843.jpg	Today at 5:25 PM	5.5 MB JPEG image
1happy1882309.jpg	Today at 5:23 PM	6.1 MB JPEG image
1happy1987301.jpg	Today at 5:23 PM	7.9 MB JPEG image
1happy2050999.jpg	Today at 5:24 PM	9.4 MB JPEG image
1happy2092872.jpg	Today at 5:39 PM	7 MB JPEG image
1happy2156656.jpg	Today at 5:26 PM	4.9 MB JPEG image
1happy2167673.jpg	Today at 5:25 PM	3.4 MB JPEG image
1happy2379004.jpg	Today at 5:23 PM	5.8 MB JPEG image
1happy2535859.jpg	Today at 5:23 PM	15 MB JPEG image
1happy2536579.jpg	Today at 5:25 PM	1.3 MB JPEG image
1happy2552127.jpg	Today at 5:23 PM	10.6 MB JPEG image
1happy2568412.jpg	Today at 5:25 PM	74.9 MB JPEG image
1happy2669601.jpg	Today at 5:25 PM	7.5 MB JPEG image
1happy2701660.jpg	Today at 5:24 PM	13 MB JPEG image

Note. The screenshot contains an example of the current dataset.

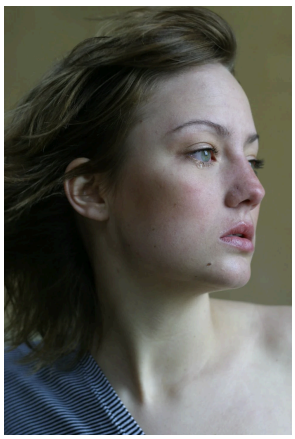
Figure 3

Smiling Woman with Red Hair

Note. Tomaz Barcellos. (2019). *Smiling Woman with Red Hair* [Image]. https://www.pexels.com/photo/smiling-woman-with-red-hair-1987301/

Figure 4

Woman Looking to Her Left



Note. Rene Asmussen. (2019). *Woman Looking to Her Left* [Image]. https://www.pexels.com/photo/woman-looking-to-her-left-2515420/

Project Deliverables

Deliverables included everything from project proposals to project reports (group and individual). This included all the tasks starting with the proposal. It could also include any prototype or proof of concept specific to this project.

The project proposal provided insight into the research problem, which was to find the most effective and accurate models for facial detection, gather different literature surveys to support the machine learning models that were used in the paper and provide examples of data sources that are from and Pexels.com as well as their limitations.

The data collection plan was explained using the Requests Python library and web scraping technique to collect images from Pexels.com API. The plan included finding the images from Pexels.com, the authentication of requesting the API from Pexels.com, the limitation of gathering the images, the cloud system that stored the images, and the people responsible.

For Data Engineering, raw data were collected with cleaning and validation tools, then datasets were preprocessed to meet the desired format. The results of the progressive report with a diverse big data visualization format were summarized. Data augmentation was applied to increase the size of the training dataset such as adding noise and rotating images to avoid overfitting.

As the model prototypes, the Extract Transform Loading (ETL) technique, model training, and model evaluation were continually implemented to build the best-performing model. Using the principal component analysis (PCA) algorithm can reduce the feature set in this research. OpenCV was the first choice for image transformation to extract the image features.

The model proposal report included a comparison among SVM, KNN, Random Forest, Naive Bayes, and Logistic Regression to determine the best model for classifying facial emotional expressions. The confusion matrix, F-1 score, five folds cross-validation were used to measure the performance of different models. Table 1 shows the project deliverables and the schedule.

Table 1

Table Showing the Project Deliverables and the Due Date.

Deliverable	Description	Due Date
Project Proposal	Project background	Sept 24th, 2022
	Data and Project requirements	Sept 24th, 2022
	Find literature, technology, and literature survey of existing research	Sept 26th, 2022
	Final Project Proposal	Sept 28th, 2022
Data & Project Management Plan	Data collection and management method	Oct 5th, 2022
	Data analytics with AI development	Oct 10th, 2022

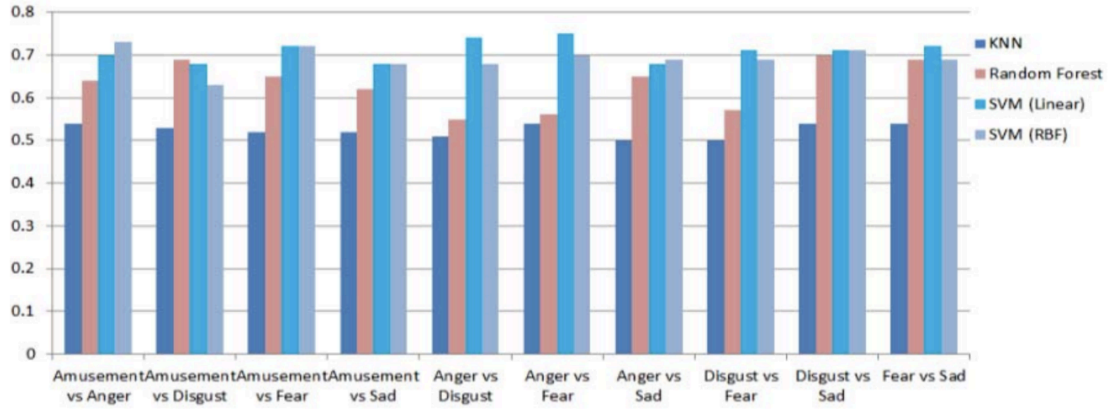
Deliverable	Description	Due Date
	Work breakdown structure *Adjust the phase, and deliverable as needed	Oct 13th, 2022
	Resource Requirements Plan	Oct 13th, 2022
	Gantt Chart & PERT Chart	Oct 17th, 2022
Data Engineering	Decide approaches and steps of deriving raw, training, validation, and test datasets	Oct 21st, 2022
	Define the parameters and source of the raw dataset; Collect Necessary new dataset	Oct 28th, 2022
	Pre-process collected raw data with cleaning and validating tool	Nov 4th, 2022
	Transform pre-processed datasets to desired formal	Nov 9th, 2022
	Summarize and present diverse data analytics result	Nov 14th, 2022
Model Development	Applying machine learning models to each of the datasets (in terms of concepts, inputs/outputs, features, model architectures, algorithms)	Nov 18th, 2022
	Describe the model, framework, environment, and technology support; provide diagrams, components, and data flow	Nov 22nd, 2022
	Compare the final selected models regarding the intelligent solution	Nov 15th, 2022
	Present evaluation methods and metrics for each model	Nov 30th, 2022
	Present and compare the machine learning models result based on the model evaluation methods	Dec 9th, 2022

Technology and Solution Survey

There are lots of models regarding face emotion recognition, and they can also work with real-time detection. Mostafa et al. (2018) used the BioVid Emo DB dataset to do facial emotion recognition. The dataset contains 430 videos from 86 participants. Each participant recorded a two minutes video for each emotion (i.e., amusement, fear, disgust, anger, and sadness). Because these are videos, the first stage of their research was to locate and detect the face from the videos. Then, they extracted two kinds of features (i.e., Geometric-based Features and Appearance Based Features) from these images. Geometric-based features contain 68 2D landmarks (i.e., the facial points for eyes, mouth, nose, etc.) and 18 Euclidean distances between specific points. They used these geometric features and appearance-based features to train machine learning and deep learning models. For machine learning models, they used SVM, random forest, and KNN. For the deep learning model, they only used LSTM-RNN. They split the dataset into training and test set and applied five-fold cross-validation to test different models' performances. At last, the authors have compared the performances of geometric-based features and appearance-based features by comparing the classification accuracies of emotion in pairs. Figure 5 and Figure 6 show the classification accuracies of emotion in pairs for geometric-based features and appearance-based features. In general, geometric-based features performed better than appearance-based features. Also, SVM (LInear) outperformed other machine learning models. However, LSTM-RNN is better than SVM (Linear) based on the classification accuracy of emotion in pairs based on Figure 7 (pp. 417-422).

Figure 5

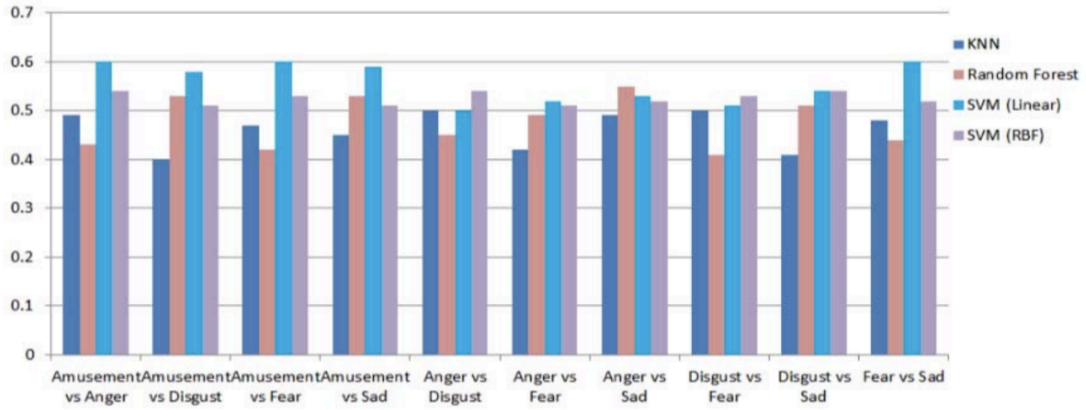
Geometric-based features results (Classification accuracy of emotion in pairs)



Note. Mostafa et al. (2018, p.420). *Geometric-based features results (Classification accuracy of emotion in pairs)*. https://ieeexplore.ieee.org/document/8639182

Figure 6

Appearance-based features results (Classification accuracy of emotion in pairs)



Note. Mostafa et al. (2018, p. 420). *Appearance-based features results (Classification accuracy of emotion in pairs)*. https://ieeexplore.ieee.org/document/8639182

Figure 7

Classification Accuracy of Emotion in Pairs Using LSTM

	Amuse.	Anger	Disgust	Fear	Sad
Amuse.	-	0.70	0.61	0.64	0.77
Anger	0.70	-	0.82	0.65	0.73
Disgust	0.61	0.82	-	0.68	0.65
Fear	0.64	0.65	0.68	-	0.61
Sad	0.77	0.73	0.65	0.61	-

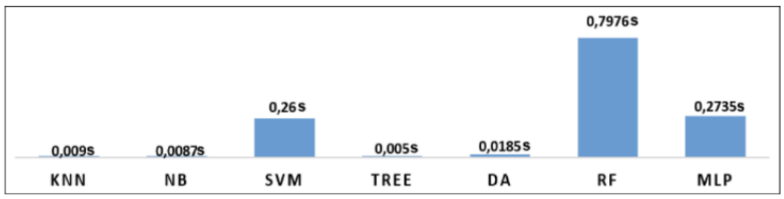
Note. Mostafa et al. (2018, p. 420). *(Classification accuracy of emotion in pairs)*. https://ieeexplore.ieee.org/document/8639182

Ayache and Adel (2020) evaluated machine learning models' performance for detecting human facial emotions. They used the CK+ dataset, which contains 326 face images, and these images are classified into seven categories (i.e., happy, sad, surprise, anger, disgust, fear, and contempt). They first used the Viola-Jones algorithm to locate the faces from the images because some images have complex backgrounds. Then, they used Active Shape Model (ASM) to extract the features and 68 landmark points from the pre-processing images. ASM used statistical methods to figure out the mean shape and the variation of the face images. Since using the ASM to extract faces, it would introduce some noise terms such as scaling, rotation, and translation issues for the face shapes. Hence, they used Generalized Procrust Analysis to normalize these landmarks vectors in order to eliminate the noises. After the normalizing stage, they stored the normalized vectors in the database and used the dataset to perform the classification tasks. They used two different methods to split the dataset, which was test vs training and one vs others. They also trained seven different kinds of classification models, which are Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree (TREE), the Quadratic Classifier (DA), Naive Bayes (NB), and Multi-Layer Perceptron (MLP). The best model should have high accuracy and a short execution time. To figure out the execution time for

each model, they used the most common algorithm in FER, and they found out that the Quadratic Classifier, Decision Tree, KNN, and Naive Bayes had the least execution time. Figure 8 shows the execution time for Random Forest (RF), K-nearest Neighbors (KNN), Decision Tree (TREE), the Quadratic Classifier (DA), Naive Bayes (NB), and Multi-Layer Perceptron (MLP) (pp. 267-275).

Figure 8

Execution Time Comparison Between Seven Classifiers



Note. Ayache and Adel (2020, p. 271). *Execution time comparison between seven classifiers*. https://doi.org/10.18280/ria.340304

In order to compare how accurate each model is, they used accuracy, precision, recall, and F1 score. Quadratic Classifier obtained the highest accuracy for test vs training. Figure 9 shows the accuracy, precision, recall, and F1 score for all classifiers using the test vs training approach.

Figure 9

Performance Comparison Between Seven Classifiers Using Test vs Training Approach

Size		7	14	21	28	35	42	49	56	63	70
KNN	Acc	100	92.85	95.23	89.28	88.57	78.57	77.55	69.64	65.07	62.85
	Pre	100	92.85	95.23	89.28	88.57	78.57	77.55	69.64	65.07	62.85
	Rec	100	95.23	96.42	92.38	91.83	74.14	74.28	64.76	55.71	53.84
	F1s	100	94,03	95.82	90.80	90.17	76.29	75.88	67.12	60.03	58.00
NB	Acc	100	100	90.47	89.28	82.85	76.19	75.51	66.07	61.90	60.00
	Pre	100	100	90.47	89.28	82.85	76.19	75..51	66.07	61.90	60.00
	Rec	100	100	92.85	90.71	86.59	74.48	74.28	64.76	57.56	55.13
	F1s	100	100	91.65	89.99	84.68	75.33	74.89	65.41	59.65	57.46
SVM	Acc	100	100	95.23	82.14	85.71	83.33	75.51	64.28	61.90	58.57
	Pre	100	100	95.23	82.14	85.71	83.33	75.51	64.28	61.90	58.57
	Rec	100	100	96.42	86.42	87.44	78.57	71.59	70.30	55.84	53.23
	F1s	100	100	95.82	84.23	86.57	80.88	73.49	67.15	58.71	55.77
TREE	Acc	71.42	92.85	71.42	67.85	71.42	66.66	59.18	50.00	53.96	54.28
	Pre	71.42	92.85	71.42	67.85	71.42	66.66	59.18	50.00	53.96	54.28
	Rec	57.14	95.23	78.57	74.04	75.79	65.60	69.82	56.25	51.12	48.14
	F1s	63.49	94.03	74.82	70.81	73.54	66.13	64.06	52.94	52.50	51.03
DA	Acc	**100**	**100**	**100**	**96.42**	**94.28**	**85.71**	**77.55**	**73.21**	**71.42**	**68.57**
	Pre	**100**	**100**	**100**	**96.42**	**94.28**	**85.71**	**77.55**	**73.21**	**71.42**	**68.57**
	Rec	**100**	**100**	**100**	**97.14**	**95.91**	**78.57**	**74.28**	**71.90**	**57.14**	**54.76**
	F1s	**100**	**100**	**100**	**96.78**	**95.09**	**81.98**	**75.88**	**72.55**	**63.49**	**60.89**
RF	Acc	100	100	90,47	85,71	88.57	71.42	69.38	67.85	58.73	60.00
	Pre	100	100	90,47	85,71	88.57	71.42	69.38	67.85	58.73	60.00
	Rec	100	100	92,85	87.14	91.15	73.94	72.02	64.76	56.72	54,76
	F1s	100	100	91.65	86.42	89.84	72.66	70.68	66.27	57.70	57,26
MLP	Acc	71.42	50.00	57.14	53.57	60.00	69.04	53.06	58.92	52,38	54,28
	Pre	71.42	50.00	57.14	53.57	60.00	69.04	53.06	58.92	52,38	54,28
	Rec	57.14	40.47	47.85	47.78	52.55	56.29	46.25	53.90	51,36	42,92
	F1s	63.49	44.73	52.08	50.50	56.02	62.02	49.42	56.30	51,86	47,94

Note. Ayache and Adel (2020, p. 273). *Performance comparison between seven classifiers using Test vs training Approach*. https://doi.org/10.18280/ria.340304

Since the quadratic classifier achieved the highest accuracy and obtained the short execution time, it is the best classification model among these seven models in this research (Ayache & Adel, 2020, pp. 267-275).

According to Zhang et al. (2020), the advances in machine learning technology, it opens the possibility to endow machines with the ability to emotional understanding, recognition, and analysis. First, this paper reviewed the emotion recognition methods based on multi-channel EEG signals and physiological signals because the use of physiological signals can be misleading and unreliable sometimes. Also, this paper analyzed the correlation between emotions and EEG rhythms as well as the brain areas, then compared the difference between machine learning and deep learning algorithms when recognizing emotion. The methodology of this paper was to survey 220 papers, and throughout these 220 papers they surveyed, the machine

learning models such as KNN, SVM, and CNN can accurately predict human facial and emotional expressions based on Figure 10 (pp. 103-126).

Figure 10

Performance Comparison of DL Methods for Emotion Recognition Based on Audio-visual Data

Dataset	DL method	Accuracy [%]	Ref.
Enterface	mel-spectrogram, face images, cnn for audio, 3 d cnn for video	85.97	Zhang et al. (2017) [160]
Audio-visual big data of emotion	2 d cnn for speech, 3 d cnn for video, elm-based fusion	99.9	Hossain et al. (2019) [166]
eNTERFACE	Audio features, facial features, triple stream DBN	66.54	Jiang et al. (2011) [175]
IEMOCAP; facial markers	Feature selection and DBN	70.46–73.78	Kim et al. (2013) [176]
EmotiW 2014	CNN for video, DBN for audio, 'bag of mouth' model, and auto-encoder	47.67	Kahou et al. (2016) [177]
eNTERFACE	Multidirectional regression, SVM	84	Hossain et al. (2016) [178]
enterface	mdr, ridgelet transform, elm	83.06	Hossain et al. (2016) [179]
Audio-visual big data of emotion	lbp features for speech, idp features for face images, svm classifier	99.8	Hossain et al. (2018) [180]
MAHNOB-HCI	CDBN	58.5	Ranganathan et al. (2016) [181]
EmotiW 2015; CK+	Audio features, dense features, CNN extracted features	54.55; 98.47	Kaya et al. (2017) [182]

Legenda: DBN = Deep Belief Network; CDBN = Convolutional DBN; CNN = Convolutional Neural Network; SVM = Support Vector Machine; ELM = Extreme Learning Machine; MDR = Multi-Directional Regression; LBP = Local Binary Pattern; IDP = Interlaced Derivative Pattern.

Note. Zhang et al. (2020, p. 121). *Performance comparison of DL methods for emotion recognition based on audio-visual data*. https://doi.org/10.1016/j.inffus.2020.01.011

While some of the significant findings included (1) the baseline EEG data was often ignored by the researcher; (2) Gama sub-band features have the highest classification accuracy and were the most sensitive to emotional changes; (3) Placing the EEG electrodes on the frontal lobe achieved a classification accuracy of over 90% (Zhang et al., 2020, pp. 103-126).

The limitation of using the EGG head cap for signal measurement is inconvenient. Therefore, the research suggested using wearable and wireless devices to measure in future experiments. In addition, a challenge in the future would be the studies of multi-modal emotion recognition due to the fact that there are many ways to interpret human emotional states, which include audio-visual, physiological and contextual. Therefore, it is essential to combine multiple modalities and properly aggregate them to find the innate priority among each emotional state (Zhang et al., 2020, pp. 103-126).

In “Facial Emotion Recognition”, Ma et al. (2017) aimed to find a suitable FER model by implementing and comparing different methods. The authors also made an attempt at feature fusion and model merging. Firstly, FERA 2015 (Second Facial Expression Recognition and Analysis Challenge) was chosen as training/testing data for building a machine learning model. Secondly, the decision-level fusion method was utilized to combine two kinds of features: geometric and appearance features. Then in the learning stage, the authors implemented a support vector machine (SVM) and a deep Boltzmann Machine (DBM). After building the model, implemented data was collected from SEMAINE, a large audiovisual database, which was used for classifying the occurrence of the Action Unit (AU). Based on Figure 11, SVC worked well with unbalanced data, which means SVM works better with unbalanced data as well. However, fusion outperformed SVM and DBM in general (pp. 77-81).

Figure 11

Classification Results of Occurance Detection of AUS with SVC And DBM

	Precision	Recall	F1	UAR	Accuracy
Average of SVC	0.456	0.191	0.237	0.573	0.857
Average of DBM	0.103	0.025	0.030	0.504	0.893
Fusion	0.481	0.236	0.359	0.602	0.910
Var of SVC	0.044	0.026	0.035	0.005	0.012
Var of DBM	0.007	0.002	0.002	0.0000008	0.008
Var of Fusion	0.052	0.032	0.040	0.008	0.016

Note. Ma et al. (2017, p. 80). *CLASSIFICATION RESULTS OF OCCURRENCE DETECTION OF AUS WITH SVC AND DBM*. doi: 10.1109/SIPROCESS.2017.8124509

Swinkels et al. (2020) developed a novel algorithm to detect emotions based on real time. According to the paper, they implemented an ensemble of regression trees to fetch facial feature points which are located on human landmarks. The extended Cohn-Kanade dataset (CK+) was

chosen to train the learning model and validate the developed algorithm. Then displacement ratios as algorithm inputs were calculated by these feature points. Finally, a combination of multi-class and binary SVM is constructed, and each emotion sequence was separated into a training set and a testing set. The results in Figure 12 indicated this model reaches an average accuracy of 89.78%, which was a satisfying accomplishment compared to a state-of-the-art emotion recognition system (pp. 86-92).

Figure 12

Performance Comparison of Developed Algorithm

	Developed algorithm	[22]	[23]
Anger	81,82%	71,39%	**92,31%**
Contempt	**100%**	/	/
Disgust	**96,55%**	95,33%	93,62%
Fear	**91,67%**	81,11%	90,57%
Happy	94,12%	**95,42%**	84,62%
Sadness	64,29%	**88,01%**	86,05%
Surprise	**100%**	98,27%	90,24%
Neutral	/	/	**90,74%**
Average	**89,78%**	88,26 %	89,74 %

Note. Swinkels et al. (2017, p. 91). *Performance comparison of the developed algorithm*. https://doi:10.1109/DESEC.2017.8073838.

Another achievement was the detection speed which was less than 30 ms, hence making it able to apply to real-time emotion detection implementation. A testing application has been built on the local computer to monitor the detection speed. For future applications, this detection model needs improvement to fix the misclassification problem, which was caused by feature points drifting. That meant in the current real-time applications, users needed to maintain the same distance from the camera. Otherwise, the unstable distance would cause inaccuracy of

detection. The authors mentioned that re-scaling neutral distances based on the users' movement would be a promising way (Swinkels et al., 2017, pp. 86-92).

According to the article by Dahmane and Meunier (2011), several factors like changes in environment and appearance, and orientations of the face, increased the difficulty of tracking face performance. Thus AFEA was highly susceptible. A baseline method was utilized as a dynamic dense appearance description to address the automated facial expression analysis (AFEA). The authors accomplished a framework by collaborating dynamic dense grid-based Histograms of oriented gradients (HOG), along with the algorithm of accumulating the gradient magnitudes. This paper studied the influence of variance of descriptor parameters under the according HOG implementations. Fine-scale gradients, binning, and normalization in overlapping descriptor blocks were all considered important parameters. They used Open CV to extract facial location features, then applied contrast amelioration as well as brightness normalization. Multi-class SVMD was applied as the base learners of the framework. Due to the sensitivity of recognition, they proposed an advanced epoch-based approach to determine the critical parameters for Radial Basis Function. Testings proceeded to compare their feature set with the static Local Binary Pattern (LBP) method. The final detection accuracy with the HOG feature set has reached 70%, compared to 56% with the LBP feature set as shown in Figure 13 (pp. 884-888).

Figure 13

Independent/Specific Partition Classification Rate for Every Emotion

Person Independent	Person specific	Overall	
anger	0.93	0.92	0.93
fear	0.07	0.90	0.40
joy	0.95	0.73	0.87
relief	0.75	0.80	0.77
sadness	0.20	1.00	0.52
avg.	0.58	0.87	**0.70**

Note. Dahmane & Meunier (2011, p. 887). *Independent/specific partition classification rate for every emotion*. https://doi:10.1109/FG.2011.5771368.

This satisfying result was on account of orientation binning, gradient magnitude corresponding, grid dynamic scaling, and relative alignment. Thus, it proved that HOG features could be implemented in real-time. The drawback of this work was no manual verification due to the absence of labeled video (Dahmane & Meunier, 2011, pp. 884-888).

Gupta (2018) used a machine learning approach to detect the face, extract face landmarks, and analyze their emotions from both images and real-time video sources. The program he purposed included three stages, which were face detection, feature extraction, and emotion classification. He classified emotions and used numeral numbers one to seven to encode seven different facial emotions. He used Cohn-Kanade Database and Extended Cohn-Kanade (CK+) database as the data sources for the static face images. The static images are 640X400 pixels and have target expressions that are all coded by FACS (Facial Action Coding System) for validating purposes. He used the HAAR filter from OpenCV to detect the faces in the static images. The dataset was split into two sections, 80% training set, and 20% testing set. Then they used the Fisher Classifier and SVM to do the classification and modeling. In the end, they

achieved an average of 92.1% accuracy, with 95% accuracy on ‘happy’ emotion as shown in Figure 14 (pp. 553-560).

Figure 14

Accuracy for Different Emotions

In static images

Emotion	Happy	Sad	Fear	Angry	Contempt	Disgust	Surprise	Neutral	Accuracy
Happy	**95**	**0**	**0**	**3**	**1**	**0**	**0**	**1**	**95%**
Sad	**1**	**85**	**2**	**0**	**3**	**0**	**1**	**0**	**92.3%**
Fear	**2**	**2**	**82**	**0**	**0**	**4**	**0**	**0**	**91.1%**
Angry	**2**	**1**	**3**	**78**	**0**	**3**	**0**	**0**	**89.6%**
Contempt	**0**	**2**	**1**	**3**	**64**	**0**	**3**	**0**	**87.7%**
Disgust	**1**	**3**	**2**	**0**	**0**	**84**	**0**	**1**	**92.3**
Surprise	**2**	**0**	**2**	**0**	**2**	**1**	**91**	**0**	**92.9**
Neutral	**0**	**2**	**1**	**1**	**0**	**0**	**1**	**81**	**91.2%**
Overall Accuracy = (660/717*100%) = 92.1%									

Note. Gupta. (2018, p. 559). *Accuracy for different emotions*. https://doi.org/10.1109/ICISC.2018.8398861

For the real-time video, he used his webcam first. The Facial Landmarks approach was used to detect emotions. In the extracting phrase, they used a corner point detection algorithm to find all the points around the feature area on the face. All those feature points have coordinates, but before using them, they normalized them using the Min-Max method. Then they used linear SVM, Polynomial SVM, K-Means Clustering, and Random Forest Classifier to train the model and perform the testing. After their experiments, they achieved their best results by using the Linear SVM and got 94.1% accuracy with all features, 91.2% accuracy with Polynomial SVM, 88.7% with K-Means Clustering, and 88.1% with Random forest classifier as shown in Figure 15 (pp. 553-560).

Figure 15

Accuracy for Different Classifiers and Features.

In Real Time

	ACCURACY		
Classification algorithm	With all features	With only vector length and angles	With just raw coordinates
Linear SVM	**94.1%**	**92.3%**	**91.5%**
Polynomial SVM	**91.2%**	**89.8%**	**89.9%**
K-Means Clustering	**88.7%**	**87.4%**	**86.6%**
Random forest classifier	**88.1%**	**81.5%**	**78.9%**

Note. Gupta. (2018, p. 559). *Accuracy for different classifiers and features.* https://doi.org/10.1109/ICISC.2018.8398861

Sharma and Bhatt (2020) stated in their paper that face recognition technology was a popular topic in the artificial intelligence area. It already became part of our life. After extracting the face features by using the Principal Component Analysis (PCA), they analyzed the data by using multiple machine learning techniques, including Linear Discriminant Analysis with Class dependent and independent transformation, Multinomial and Gaussian Naive Bayes Classifier, Support Vector Machine Classifier, Multilayer Perceptron Classifier. Finally, by using the Olivetti face dataset, they performed the modeling and testing. They used PCA+SVM and PCA+LDA with different ratios of the training dataset and testing dataset. And they compared these results with the previous results tested by other teams. Other teams used DWT+LDA+SVM, Naive Bayes, and HOG+SVM. The best among them was DWT+LDA+SVM with 96%. And the authors achieved 100% accuracy with PCA+SVM (90:10) and PCA+LDA (90:10) as shown in Figure 16 (pp. 1162-1168).

Figure 16

Comparative Study

Approaches	Method Used	Accuracy (%)
[4]	DWT+LDA+SVM	96
[5]	Naive Bayes	93.16
[6]	HOG+SVM	92.68
Proposed Approach	PCA+SVM (80:20)	94.00
	PCA+SVM (90:10)	100.00
	PCA+LDA (80:20)	97.00
	PCA+LDA (90:10)	100.00

Note. Sharma et al. (2020, p. 1167). *Comparative Study.* https://doi.org/10.1109/ICCES48766.2020.09137850

Table 2 shows the differences and similarities among all papers mentioned above except the paper written by Zhang et al. (2020) because it is a summary of 220 research papers, and they did not use any datasets and implement any models (pp. 103-126).

Table 2

Table Showing the Differences and Similarities among Seven Research Papers

Reference	Dataset	Feature Extraction	Feature Normalization?	Models	Consider Execution Time?	Final Model & Accuracy (%)
Mostafa et al. (2018, pp. 417-422)	BioVid Emo DB	Local Binary Pattern (LBP)	No	SVM, RF, LSTM-RNN	No	LSTM-RNN
Ayache & Adel (2020, pp. 267-275)	CK+	Active Shape Model (ASM)	Yes	DA, RF, MLP, Decision Tree, SVM, NB, KNN	Yes	DA: 68.57% for test size = 70
Ma et al. (2017, pp. 77-81))	FERA 2015	LBP	No	SVM, Fusion, Deep Boltzmann Machine (DBM)	Yes	Fusion: 91%

Reference	Dataset	Feature Extraction	Feature Normalization?	Models	Consider Execution Time?	Final Model & Accuracy (%)
Swinkels et al.(2017, pp. 86-92)	CK+	Ensemble of Regression Trees	No	Multi-class and Binary SVM	No	Multi-class & Binary SVM: 89.78%
Dahmane & Meunier (2011, pp. 884-888)	GEMEP-FERA	Open CV, HOG, LBP	No	Multi-class SVMD	No	HOG feature: 70%
Gupta (2018, pp. 553-560)	CK+, Webcam	HAAR, corner point detection algorithm, normalized formula	Yes	SVM, Linear SVM, Polynomial SVM, K-Means Clustering, and Random Forest Classifier, Fisher Classifier	No	Fisher Classifier and SVM: 92.1% on static images. Linear SVM: 94.1% on real-time video
Sharma et al. (2020, pp. 1162-1168)	Olivetti Face Dataset	PCA, DWT	No	LDA, Multinomial and Gaussian Naive Bayes Classifier, SVM, Multilayer Perceptron Classifier	No	PCA+SVM (90:10): 100% PCA+LDA (90:10): 100%

Literature Survey of Existing Research

In the paper by B. K. Bhavitha et al. (2017), they introduced multiple sentimental analysis machine learning techniques, including, Decision Tree, Neural Network, SVM, Naive Bayes, Bayesian Network, Maximum Entropy, and KNN. They compared various approaches with different datasets, and the results are shown in Figure 17 (pp. 216-221).

Figure 17

Performance Comparison of Sentiment Classification Technique

	Method	Data set	Accuracy
Machine learning	SVM	Movie reviews	86.40%
	CoTraining SVM	Twitter	82.52%
	Deep learning	Standard benchmark	80.70%
Lexicon based	Corpus	Product Reviews	74.00%
	Dictionary	Amazon	---
Cross-lingual	Ensemble	Amazon	81.00%
	Co-Train	Amazon, IT168	81.30%
	EWGA	IMDb moview review	>90%
	CLMM	MPQA, NTCIR,ISI	83.02%
Cross-domain	Active learning	Book, DVD, Electronics, Kitchen	80% of average
	Thesaurus		
	SFA		

Note. Bhavitha et al. (2017, p. 220). *Performance comparison of sentiment classification technique.* https://doi:10.1109/ICICCT.2017.7975191

They collected movie reviews online to analyze the sentiments. They compared SVM, Naive Bayes, and KNN techniques with ten experiential learning, the results are shown in Figure 18 (pp. 216-221).

Figure 18

Accuracy Obtained After Testing Data Set

# experiment	# reviews	SVM (%)	Naïve Bayes (%)	kNN (%)
1	50	60.07	56.03	64.02
2	100	61.53	55.01	53.97
3	150	67.00	56.00	58.00
4	200	70.50	61.27	57.77
5	400	77.50	65.63	62.12
6	550	77.73	67.82	62.36
7	650	79.93	64.86	65.46
8	800	81.71	68.80	65.44
9	900	81.61	71.33	67.44
10	1000	81.45	75.55	68.70

Note. Bhavitha et al. (2017, p. 220). *Accuracy obtained after testing data set.* https://doi:10.1109/ICICCT.2017.7975191

For a dataset with a small feature set, Naive Bayes works well, but with a large feature set, the SVM is the better choice. The max entropy also works well, but it could face the overfitting problem.

Putranto et al. (2016) extracted the face into the eigenface, and then they used the Naive Bayes approach as their predicting model. And also used normalization z-score and cross-validation techniques. The final results were, without a z-score, they achieved 70%, and with a normalization z-score, they achieved 89.5%.

Mahmud et al. (2015) used PCA, and LDA approaches with a linear projection method to solve the face recognition problem, then they tested these two models on two datasets. The UMIST dataset contains side views, ORL dataset contains frontal views. For the UMIST, they got an average of 88.33% for PCA, and 91.11% for LDA. For ORL, they got an average of 96.11% for PCA, and 98.22% for LDA.

Lahaw et al. (2018) approached the face recognition problem with 2D-DWT, ICA, PCA, LDA, and SVM techniques. They achieved 96% accuracy with both DWT+PCA+SVM and DWT+LDA+SVM approaches and 94.5% with the DWT+ICA+SVM approach. Kotsia et al. (2008) used Gabor wavelets, the DNMF algorithm, and shape-based SVMs approaches. They found that different area of face occlusion has a different level of effect on the prediction. The face of the human's left and right sides have low impact, whereas the mouth occlusion will affect the model’s performance significantly (pp. 1052-1067).

Also, it is essential to extract good features to help identify the target correctly. Therefore, many feature detector algorithms were surveyed. Kazemi & Sullivan (2014) transform the image based on the current shape estimate and standard coordinate system. After extraction, features are exploited to recalculate the shape parameters until they get a good convergence. To detect

subtle emotions, Ngo et al. (2016) used motion properties for magnification, which can increase detection accuracy. Both amplitude-based and phase-based motion magnification are implemented based on these properties (pp.1243-1247). Facial landmark is a widely used term in facial detection. Facial landmarks related to emotions have presented a set of 19 features (Kaya, 2013). The set of 19 features was a subset of the marker-less identification and localization landmark system, which consists of 66 2D features. Eccentricity features utilized the concept of ellipses which is the ellipse's deviation from a circle. For instance, an eccentricity greater than 0 presented smiling, but when it was closer to 0, it indicated surprise (Saragih et al., 2011, pp. 213-220).

The viola-Jones face detector is another face detector that uses the Ada-boost algorithm. This method integrates several weak classifiers to form a robust classifier by iterations. Meanwhile, weak classifiers decrease the weighted error rate (Dahmane & Meunier, 2011, pp. 884-888). The authors also introduced the Local Binary Pattern (LBP) in the same paper. LBP is a straightforward and robust feature extraction technique independent of the images' illumination differences. The pixel values of the matrix are transformed to decimal values which are used as image features for classification. After reevaluating existing edge and gradient-based descriptors, many research papers adopted an object detection approach called Histogram of Oriented Gradient (HOG). This technology crops images into smaller regions also called cells. Dalal and Triggs (2005) showed experimentally that HOG descriptors significantly exceed other feature sets for human detection.

Lyons et al. (2000) used three different facial image datasets. They proposed two approaches to process face images, which are Gabor-wavelet-labelled elastic graph matching and Fisherface algorithms. Then, they found out the LDA outperformed non-linear perceptron with

92% accuracy (pp. 202-207). However, according to Bartlett et al. (2003), when they applied Garbor representation to classify facial emotion detection, they found out SVM was more effective than LDA in classifying emotion when the classes did not follow Gaussian distribution, and LDA was optimal for Gaussian distributed classes (pp. 53–53).

While Lyons et al. (2000) and Bartlett et al. (2003) worked on detecting emotion through images, Michel and Kaliouby (2003) worked on real-time videos. They implemented SVM to predict emotion from live videos, and it performed well on recognition accuracy and execution time. Lyons et al (2000)., Bartlett et al. (2003), and Michel and Kaliouby (2003) only used one dataset to implement multiple models and then figured out the best model based on the model performances (e.g., accuracy, execution time).

However, Barman and Dutta (2021) only implemented one model (i.e., MLP) and fed different features (i.e., Distance, Shape, and the combination of Distance and Shape) into MLP to compare which feature performed the best in distinguishing different emotions. Also, they investigated these on CK+, JAFFE, MMI, and MUG datasets. Based on the experiment, they found out that the combination of Distance and Shape outperformed other features with 100% accuracy for CK+, 96.4% accuracy for JAFFE, 81.9% accuracy for MMI, and 97.7% accuracy for MUG (pp.254-261).

Chuang and Shih (2006) introduced a different method for face extraction, which was independent component analysis, and most of the other researchers used Active Shape Model. They also adopted SVM for the classification task and compared their model with three existing FER systems (i.e., Tian, Donato, and Bazzo). Their model had a higher recognition rate (i.e., 100% correct rate) and shorter execution time (i.e., 1.8 ms).

Chapter 2 Data & Project Management Plan

Data Management Plan

Data Collection Approaches

For the project on facial emotion recognition, many high-quality unedited, firsthand, royalty-free images were needed. Hence, Pexels.com was the priority data source for the project. It offers free API and millions of free stock photos; however, authorization was required from Pexels.com to request an API key. The rate limit on the API is 200 requests per hour and 20,000 requests per month, or requesting a higher allowance if needed. Based on the rate limit on the Pexels.com API, approximately a two-week duration was required to collect a total of 2,000 – 3,000 images for each category which included sad, happy, and angry, and the size of the dataset was 30GB in total. The reason to choose Pexels.com as the priority data source was that it offered many frontal face images with the emotion of the person in the picture that was suitable for the need of the feature extraction.

Pexels's photos and videos are used for free, and attribution is not required but appreciated when giving credit to Pexels or the photographer. In addition, Pexels allows users to modify the photos but not redistribute or sell unaltered copies of the photos or videos.

The image was extracted using the web scraping technique with the Python library while accessing the Pexels content library using their API. Images needed to be in JPG format, royalty-free, and high resolution. Non-individual portrait images were eliminated throughout the data collection process. The steps of web scraping in Pexels.com through Python:

1. From pexels_api import API.

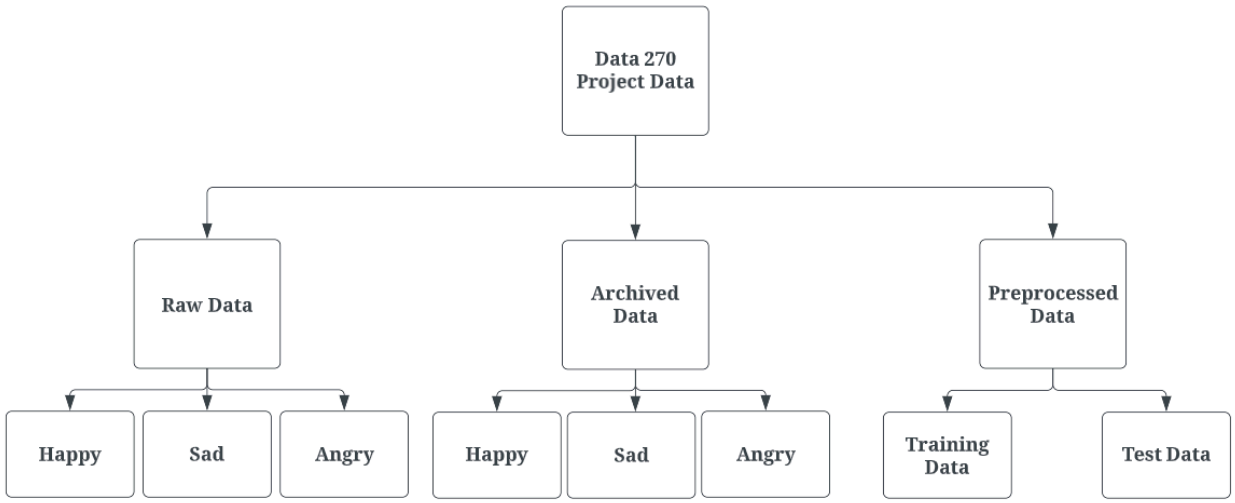
2. Pexels API library offers a search engine to look for images and be able to extract the content URL, and get information about the image, such as the id, owner, and various sizes of the image URL.
3. Based on the function of api.search, the category of image, page number, and the number of images per page can be searched with the api.search function.
4. Based on the function of api.get_entries, the information of images can be found, such as image ID, photographer, and different sizes of image URL.
5. Import Requests library.
6. Once the URL of the image is found from the function of requests.get, the images can be stored with the function of image.content into the local folder.

Data Storage Methods

All the images were stored in a google cloud shared folder and had sufficient storage to store all the images while only allowing the people who had access to view or make any edits. Within the shared folder (i.e., Data 270 Project Data), there were several sub-folders (i.e., Raw Data, Preprocessed Data, Archived Data, Preprocessed Data, and Raw Data Backup), as shown in Figure 19.

Figure 19

Folder Structure



Note. This diagram shows the folder structure and the folders' names.

Three subfolders, including happy, sad, and angry, were assigned under the raw data folder. The happy folder stored the happy emotion images. The sad emotion images were stored in the sad folder, and the angry emotion images were in the angry folder. Each folder contained around 1000 images from Pexels.com. The images name conversions are followed by page number, category, and photo id. All the images were in jpg format. For example, if the image was on page 1 in Pexels.com and it belonged to the category of happy with a unique photo id of 12345, its name was 1happy12345.jpg. There was only one version of the raw data because it was a growing dataset.

Since the HoG face detector in the Dilb library was hard to process images when they contained more than one person or were not frontal view, these unfitted images were stored in the archived data folder. However, this challenge was unavoidable because there was no guarantee of the type of emotional images collected by the web-scrapping. The archived data

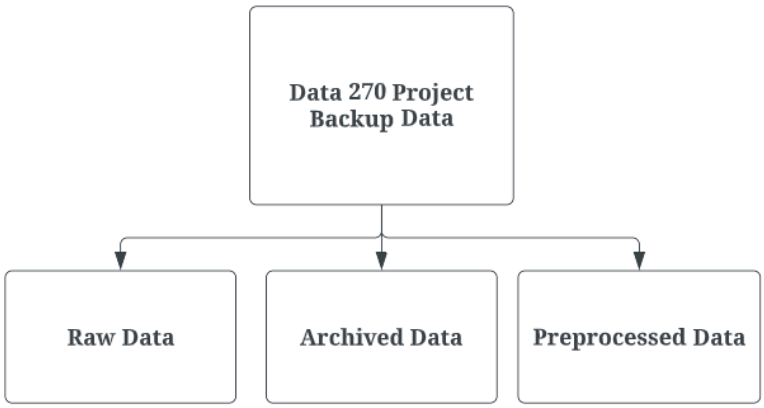
folder also had three folders (i.e., happy, sad, and angry). These unfitted images were assigned to these three folders according to their categories. The reason to archive these images was that there would be a possibility of reusing these images in the future if the models could improve to the extent that they could classify side-view face images and multiple persons' images in the future. The archived data only had one version since it is a growing dataset.

All the images in the preprocessed data folder were in greyscale and only contained the face. The file name convention was the original file name followed by "_preprocessed" (e.g., 1happy12345_preprocessed.jpg). Images were not remodified or deleted after preprocessing. Therefore, there was only one version for the preprocessed images. Preprocessed images were split into training data and test data and stored accordingly.

The backup data was stored in another google drive account, so the data could quickly be recovered if the data was corrupted or lost in the main google drive account. The folder contained three subfolders (i.e., raw, archived, and preprocessed data) as shown in Figure 20. Everything in the raw data, archived data, and preprocessed data folder in the main google drive account were in the backup data folder. The backup data folder only contained one backup copy, and it was a growing dataset, so all the images in it had one version. Once a new image was stored in either folder, it would immediately back up manually by team member who is Wen, and she was also responsible for any data recovery if needed.

Figure 20

Backup Data Folder Structure



Note. This diagram shows the backup data folder structure and the folders' names.

Data Management Methods

The collected facial image datasets were stored on Google Cloud Drive, therefore, also protected by Google Drive permission. All team members had permission to access the data, while the datasets were also accessible from Colab since Colab could directly connect to Google Drive.

The facial datasets were collected from royalty-free websites, so citations were included from the website. Therefore, the public could also find the original image repository on the website. If other users wanted to access the particular dataset that was collected from the website within the project, they could request permission to access the google drive folder. However, according to the project plan, the dataset would only be preserved for two months.

Data Usage Mechanisms

The image data were stored in Google Cloud. According to the project expenditure plan, the dataset was preserved for two months. During this time, other users who wanted to get access

to the Google Cloud could request permission from any team member. The dataset documentation and metadata logs were stored in the virtual machine that was created inside the Google cloud platform. The dataset documentation is shown in Table 3.

Table 3

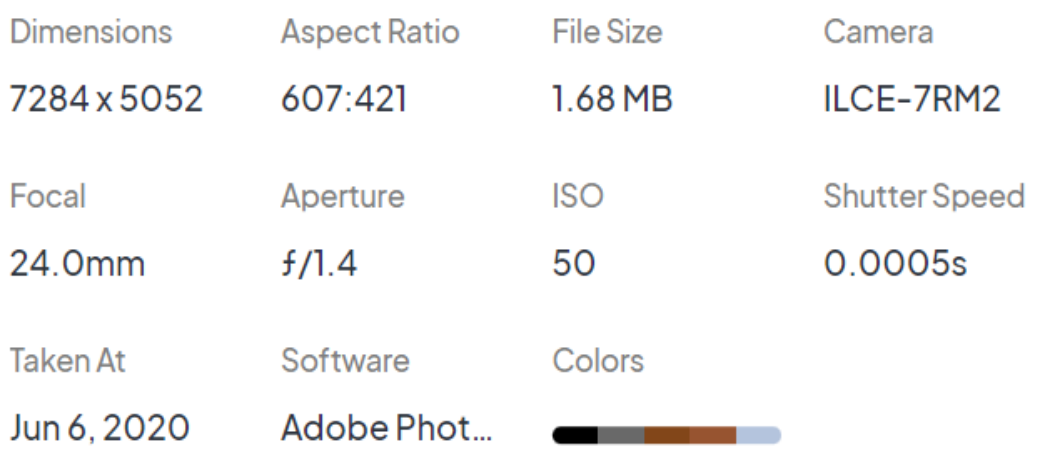
The dataset documentation.

Attribute	**Information**
Principal Creator	Cheuk Hong lp
Secdonary Creators	Shufang Wen, Wenbin Wang, Jing Li
Title of Dataset	Data 270 Project Data
Subtitle of Dataset Files	Raw Data, Preprocessed Data, Archived Data, Preprocessed Data, and Raw Data Backup.
Image Name Convention	Website Page Number, Emotion Category, and Photo ID
Date that Dataset File Was Created	10/14/2022
Date(s) of Data Collection	10/20/2022 - 10/24/2022
Data format	JPG
Data Source	Pexels.com
Licensing	Royalty-free Images But Need Team Members' Permission to Access the Dataset
Methods of Data Collection	Web Scraping Technique Using Python Library
Methods Used for Data Processing	Label Images with emotion Manually and Stored in Separate Folder. Resize, Recolor, and Remove Background for All the Images

A log file was created for recording each image's URL information, therefore, future users could quickly locate the image's original source. The log also recorded the metadata of each picture, including picture dimensions, aspect ratio, file size, Camera, Focal, Aperture, ISO, Shutter Speed, and taken date as shown in Figure 21, and Table 4 shows each team members' responsibilities.

Figure 21

MetaData Example



Note. This is an example of image metadata.

Table 4

The team members' responsibility.

Phases of Data Management	**Assigned Person**
Data collection	Cheuk Hong lp, Shufang Wen, Wenbin Wang, Jing Li
Documentation and Metadata	Jing Li
Data Storage and Backup	Shufang Wen
Data Sharing and Preservation	Wenbin Wang

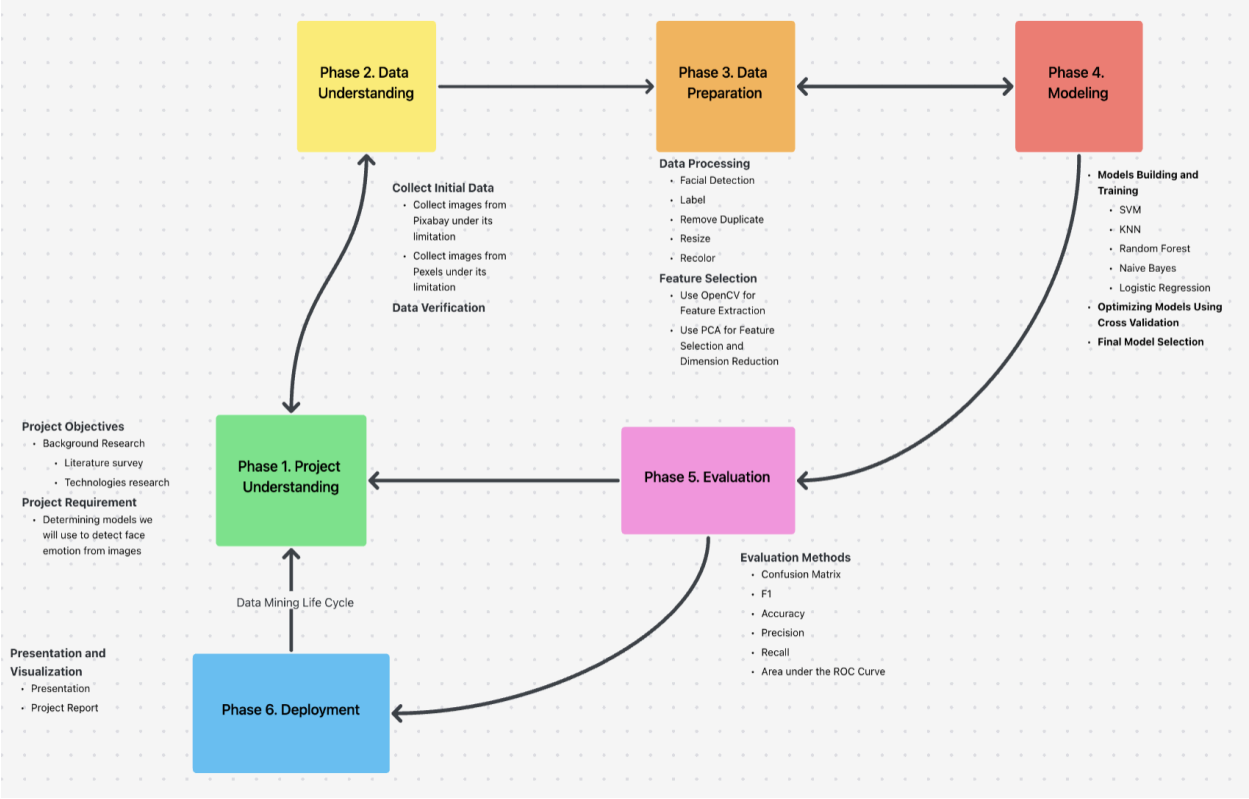
Project Development Methodology

System Development Life Cycle

CRISP-DM is a well-known model. Most data scientists use it to demonstrate the framework of the data science project. CRISP-DM was applied to display the data mining approach throughout the project development phase, where the team used it to schedule and manage the project effectively. There were six phases in the CRISP-DM model: project/business understanding, data understanding, data preparation, modeling, evaluation, and development. Figure 22 is shown the team's CRISP-DM model.

Figure 22

CRISP-DM



Note. Six phases in *CRISP*-DM.

Planned Project Development Processes and Activities

In the project/business understanding phase, literature surveys and technical research were analyzed to understand what others accomplished to classify facial emotion using images and how they processed the images, extracted features from the images, etc. Also, this research provided insight into the project requirements (e.g., where to collect the royalty-free images, what machine learning models to use, etc.).

In the data understanding phase, images were gathered from Pexels.com. After collecting the data, all images must be re-evaluated to ensure image quality. In the data preparation phase, there was additional processing of the images (i.e., removing the background, duplicates, resizing, recoloring, and labeling all the images). OpenCV was utilized to perform feature extraction, and Roboflow was used for data augmentation such as adding noise, rotating images, and splitting the data into 80% of the training and 20% of the testing sets. Then, Min-Max normalization and PCA were implemented for the feature transformation and dimensionality reduction.

In the modeling phase, machine learning models were implemented: SVM, KNN, Random Forest, Naive Bayes, and Logistic Regression. Furthermore, cross-validation was applied to optimize the models' performances.

On top of this, the final model was selected based on the performance. Throughout the evaluation phase, the final model was used to make predictions on the test images and then used the evaluation metrics (i.e., confusion matrix, F1 score, precision, recall, accuracy, and area under the ROC curve) to evaluate the final model performance. The last phase was the deployment phase which required some visualizations to demonstrate the model prediction results.

Project Organization Plan

The work breakdown structure of this project as shown in Figure 23 contains four phases which are project preparation, data and project management plan, data engineering, and model development. The preparation phase had to be finished before the data and project management phase. Then, the data engineering phase and model development phase accordingly were processed.

In the project preparation phase, there were three main tasks. Firstly, the literature reviews how others handle classifying facial emotions using images, to help figure out what limitations they faced, what models they used, and how accurate their models are. Because the requirement of data is first-hand, the method of web scraping images from the internet was adopted. Where to collect high-quality images became an important issue of this project. Therefore, the second task of project preparation was to find out where to collect royalty face emotion images. Also, coding tutorials about how to process images were prepared.

In the data and project management plan phase, there were two main tasks (i.e., data management and project management). For data management, APIs and licenses to web-scrape images from the websites were requested, then Google cloud was set up for storing these images. For project management, a data mining life cycle was created followed by CRISP-DM methodology, WBS showed the high level of project tasks, a Gantt chart showed all of the tasks in detail, and a PERT chart showed the tasks' dependencies.

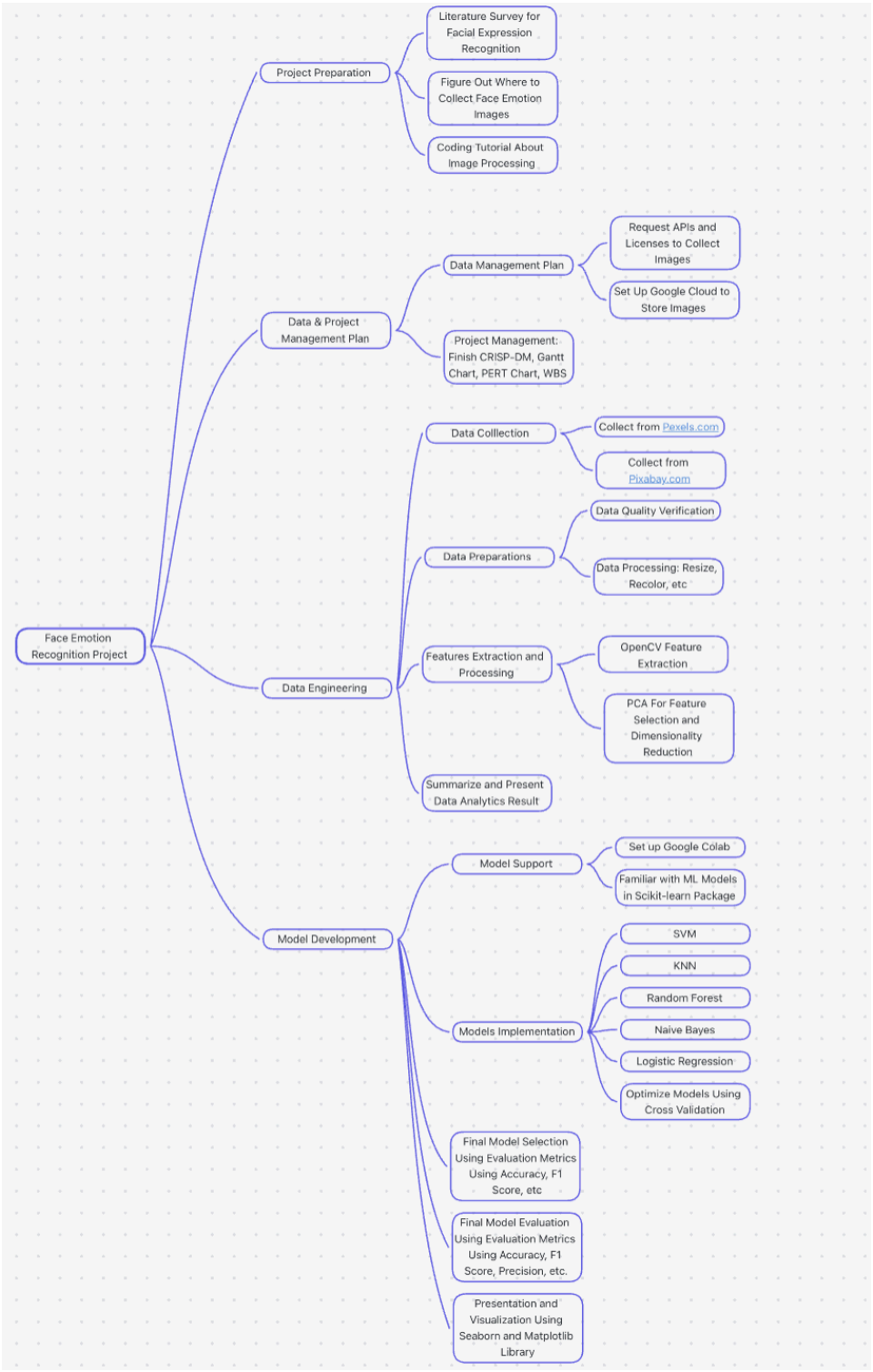
In the data engineering phase, there were four tasks which are data collection, data preparations, feature extraction and processing, and data analytics results summarizing and presenting. All the images were collected from Pexels.com. After finishing collecting the data, it was the phase of data preparation (i.e., verifying the quality of the images and resizing,

recoloring, and removing the background for the images). Then, the feature extraction and processing tasks proceeded. Features were extracted from the images by OpenCV. PCA analysis was performed along with the cross-validation, and the scree plot helped to find out the best number of PCs for model implementation. The dimensionality reduction of the features prevents overfitting. Lastly, the analysis results were summarized and put in the report.

In the last phase of model development, there were five main tasks. For the model support task, Google Colab was set up to write the code and get familiar with how to use the machine learning models in the sci-kit-learn package. Then, five models were implemented(i.e., SVM, KNN, Random Forest, Logistic Regression, and Naive Bayes. There are some hyperparameters for these models. Hence, cross-validation was used to figure out the best value for these hyperparameters to optimize the model and improve model performance. After finishing developing five different machine learning models, the evaluation metrics (i.e., accuracy, precision, recall) were applied for the final model selection. Then, the final model was used for predicting and performing evaluation for the test images. Lastly, the report and the presentation were finalized. Also, some visualization was created using seaborn and Matplotlib library.

Figure 23

WBS



Note. This is the deliverable-oriented WBS.

Project Resource Requirements, and Plan

Hardware Requirements

Depending on the choice of the hardware, there are two plans: (1) Priority Plan and (2) Back-up Plan. For the priority plan, one of the hardware stores is the google cloud standard (Region: Oregon/us-west1) for $0.020 per GB-month. The purpose of google cloud storage is to store raw data. Another hardware is the free version of the Google Colab with 12 GB RAM to develop EDA and ML Models. Google Colab GPU is Nvidia K80/T4 with 12 GB memory to process high-resolution images. At the same time, the CPU is Intel(R) Xeon(R) with the CPU Freq 2.30GHz to run a range of tasks more efficiently. For the Backup plan, the hardware is the Mac Mini M1 which has an 8-core CPU with four performance cores and four efficiency cores to run a range of tasks efficiently. In addition, Mac Mini M1 has an 8-core GPU to process high-resolution images and 1 TB of storage to store raw data. Table 5 shows the priority plan and Table 6 shows the back-up plan.

Table 5

Table Showing the Hardware, Configuration, and Purpose of Use in the Priority Plan.

Hardware	Configuration	Purpose of Use
Google Cloud Storage – Standard	Region: Oregon (us-west1) $0.020 per GB-month	To Store Raw Data
Google Colab (Free Version)	12GB RAM	To Develop EDA and ML Models
	GPU: Nvidia K80 / T4, 12GB Memory	Be Able To Process High-Resolution Images
	CPU: Intel(R) Xeon(R), CPU Freq 2.30GHz	Be Able to Run a Range of Tasks Faster

Table 6

Table Showing the Hardware, Configure, and Purpose of Use in the Back-up Plan.

Hardware	Configure	Purpose of Use
Mac Mini M1	8-core CPU with four performance cores and four efficiency cores	Be Able to Run a Range of Tasks Faster
	8-core GPU	Be Able To Process High-Resolution Images
	1TB Storage	To Store Raw Data

Software Requirement

After scaping web images using Python software Version 3.9, coding is implemented to the Requests Library Version 4.6.0 to read the JSON file, extract the content from the URL, and implement to the Pexels_api Library to get access and search for an image from Pexels.com. Jupyter Notebook software Verison 6.4.12 is utilized to develop EDA and ML models. Pandas Library Version 1.5.0 is used to build DataFrame; Numpy Library Version 1.21.5 is utilized for working with arrays; Scikit-learn Library Version 0.21 is operated to implement ML models; Open CV Library Version 4.6.0 is used to process the image. Table 7 shows the software, version, and purpose of use.

Table 7

Table Shows the Software, Version, and Purpose of Use.

Software	Version	Purpose of Use
Python	Version 3.9	To Web Scaping Image
Jupyter Notebook	Version 6.4.12	To Develop EDA and ML Model
Pandas	Version 1.5.0	To Build DataFrame
Numpy	Version 1.21.5	To Use for Working with Arrays
Scikit-learn	Version 0.21	To Implement ML Models
Open CV	Version 4.6.0	To Process the Images
Requests	Version 2.18.4	To Read JSON File and Get the Content in URL
Pexels_api	Not Applied	To Get Access and Search for Images in Pexels.com

Tools and Licenses

Clickup is a project management tool to help the team schedule, assign and monitor task completion. This project management tool costs $9/month for each member. Zoom is a secure and reliable video platform to help communicate effectively for free. Pexels API is an open-source license to collect high-resolution and royalty-free images. Table 8 shows the tool name, license, and purpose of use.

Table 8

Table Showing the Tool Name, License, and Purpose of Use.

Tool Name	License	Purpose of Use
Clickup	$9/month for each member	To Manage Project with Schedule and Task
Zoom	Free	Communicate with each member
Pexels API	Open Source (Give Credit to Pexels Team in the Reference)	To Collect High-Resolution Royalty-free Image

Project Cost and Justification

For the priority plan, the total cost is $73.2, which include the cost of the Google Cloud Storage Standard for $1.2 and Clickup for $72. The total cost for the backup plan is $1,171, including the Mac Mini M1, which is $699, 1 TB Storage for $400, and Click up for $72. Google Colab, Jupyter Notebook and Python software, Pandas library, Numpy library, Scikit-learn library, Open CV library, Requests library, and Pexels_api are free of charge. Table 9 shows the hardware and software tools.

Table 9

Table Showing the Hardware/Software/Tool, Configuration/Version, Duration, and Cost.

Hardware/Software/ Tools	**Configuration/Version**	**Duration**	**Cost**
Google Cloud Storage – Standard	Region: Oregon (us-west1) $0.020 per GB-month 30GB of Data	Two Months	$1.2
Clickup	$9/month for each member	Two Months	$72
Mac Mini M1 (Backup Plan)	8-core CPU with four performance cores and four efficiency cores & 8-core GPU	Two Months	$699
	1TB Storage		$400
Python	Version 3.9	Two Months	Free
Google Colab (Free Version)	GPU: Nvidia K80 / T4, 12GB Memory	Two Months	Free
	CPU: Intel(R) Xeon(R), CPU Freq 2.30GHz		
	12GB RAM		
Jupyter Notebook	Version 6.4.12	Two Months	Free
Pandas	Version 1.5.0	Two Months	Free

Hardware/Software/ Tools	Configuration/Version	Duration	Cost
Numpy	Version 1.21.5	Two Months	Free
Scikit-learn	Version 0.21	Two Months	Free
Open CV	Version 4.6.0	Two Months	Free
Requests	Version 2.18.4	Two Months	Free
Pexels_api	Open Source	Two Months	Free
Plans			Total Cost
Priority Plan			$ 73.2
Backup Plan			$ 1,171

Project Schedule

Gantt Chart

There were four milestones in Gantt chart (see Appendix A), including Project Preparation, Project Management Plan, Data Engineering, and Model Development.

The Project Preparation milestone lasted 20 days. In this phase, the project topics were discussed, and determined that ‘Facial Expression Recognition’ would be the topic. Then, the project goal was discussed on Sep. 7th, which was to find the best model to classify emotion in a picture, and this task was dependent on the project topic task. Next was the project requirements task, depending on the project goal, which contained three sub-tasks, and lasted from Sep. 8th to Sep. 13th. Team member Hong figured out where to collect royalty-free images in two days, Hong and Wen got familiar with google cloud and Google Colab over the weekend, and Li and Wang determined the machine learning models in two days. And Wen and Hong have searched for coding tutorials according to the machine learning model determination during Sep. 14th to Sep. 21st, and in the meantime, Wang and Li worked on Literature Review from Sep. 18th to Sep. 26th.

The second phase was the project management plan milestone, from Sep. 27th to Oct. 17th. The first three days were used to decide the CRISP-DM by Li and Wang, then Wen and Hong determined the WBS in the next three days, after the completion of CRISP-DM, then Hong and Wang finished the PERT Chart after the completion of WBS. Then Wen and Li finalized the Gantt Chart based on the WBS. And Hong decided on the Data Collection Plan. The next task was the resource requirements plan, which contained three tasks, from Oct. 12th to 17th, in the first four days, two subtasks happened simultaneously, which were familiar with PM tools by Li and Wen, and Google Cloud for data storage by Hong, then Wang worked on the Google Colab to use Jupyter Notebook.

The third milestone was Data Engineering from Oct. 18th to Nov. 11th. In the first two days, Hong and Wang set up the Google Colab environment. Then the data collection task was dependent on the data collection plan, the subtask including collecting from Pexels, from Oct. 20th to 24th. And for the next seven days, Wang finished the data quality verification, after that, Hong and Li started doing the Data Cleaning: resize, recolor, etc. And Wang and Wen worked on the ETL after the data cleaning for three days and four days for Utilizing OpenCV for Feature Extraction. And then Hong and Li used PCA to do the feature transformation and dimension reduction for four days after the feature extraction. Then the data statistics was done by all team member together for three days.

The last phase was model development lasting 22 days, from Nov. 12th to Dec. 3th. After completing the Feature Selection, the model implementation lasted four days; Hong implemented SVM, Wang developed RF, Li developed the KNN, Wen implemented NB, and the team implemented Logistic Regression together. The next task was the Final Model Selection, which depended on Models Implementation; Wen, Li, and Hong finished this task within four days.

And Li and Hong made the test set prediction using the final evaluation. Then Wen and Wang evaluated the final model from Nov. 23rd to 26th. Lastly, the report and presentation both depended on the final model evaluation task, and the report also depended on data statistics, and both tasks were performed by all team members, which took four days.

PERT Chart

Milestone-Oriented PERT Chart was used to present the overview of the project management plan. Four major milestones were project preparation, project management plan, data engineering, and model development; multiple sub-milestones were divided in PERT Chart.

As showing in figure 24, the red line in the chart was the project's critical activities, and the total estimated duration days of the whole project was 72 days. In the first beginning, an agreement was made to explore the field of image classification by machine learning. Then, dozens of literature and technology from related papers were surveyed by team members. The project preparation lasted 20 days in total, including research of the background and requirements.

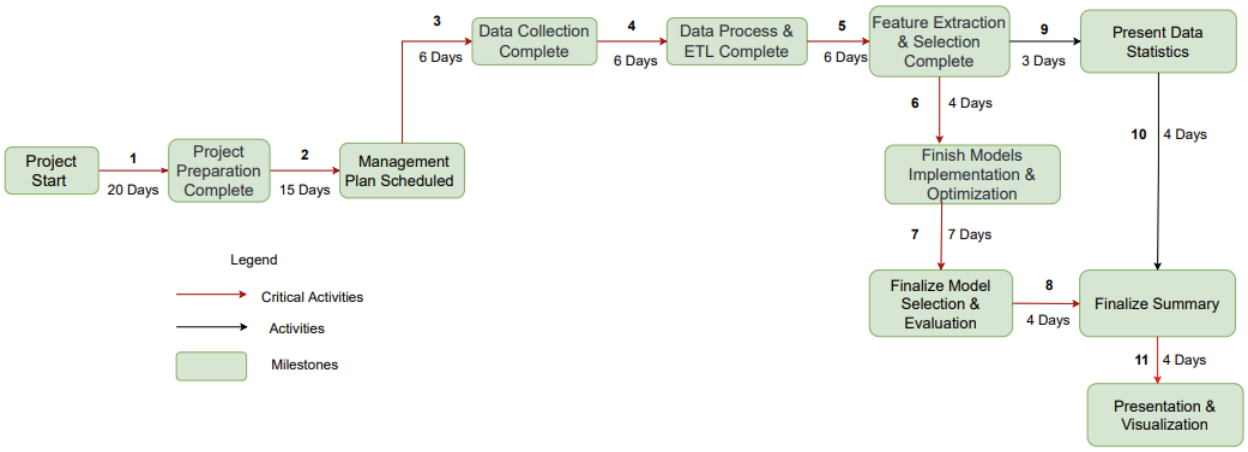
The next step was to make the management plan, which lasted 15 days. WBS, Gantt chart, and PERT chart were created by PM tool Clickup during this stage. Then the phase of data engineering was initiated, and image data were collected from Pexels.com and stored in Google Cloud. The data processing cost six days, including verifying, cleaning, resizing, and recoloring images. After completing the data collection, OpenCV was used to extract features, and PCA was used to transform features and reduce dimensions, which also costed six days. At the end of this stage, the data analysis result was sorted and visualized.

Lastly, in the stage of model development, building and implementing five kinds of models instantaneously cost four days. The five models are SVM, KNN, RF, NB, and Logistic

Regression, and they were optimized by cross-validation. The final model was selected after comparing and evaluating the results. After having the analysis result both from data engineering and model development, the final report was completed in four days. Finally, another four days were spent on presentation preparation.

Figure 24

PERT Chart



Note. This is the PERT chart.

Chapter 3 Data Engineering

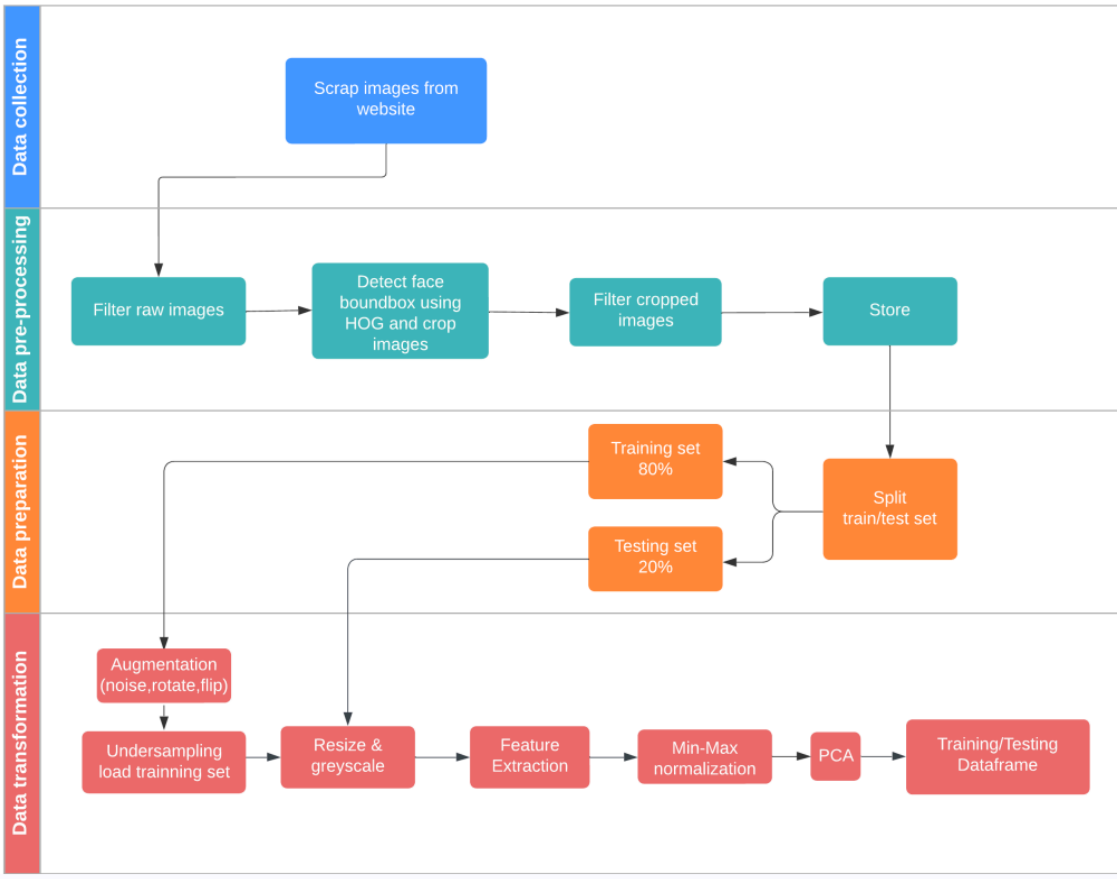
Data Process

As shown in Figure 25, images were web-scraped from Pexels.com. These first-hand images have several issues, such as missing emotion labels, various backgrounds, and some photos being cartoon emoji instead of the human face. First, the images were manually filtered to ensure they were well and correctly represented. Then, the faces were cropped out using SVM and HOG. At the end of the data pre-processing, the cropped face images were manually filtered again to ensure the data quality. Finally, they were stored in the Preprocessed Data folder.

Data Preparation was before data transformation because augmentation needed to be performed to enhance the models' performance. Also, data augmentation was only applied to the training sets. During the data preparation stage, the data was split into training and test sets with a ratio of 80/20. Then, during the data transformation stage, all the training images were rotated, flipped, and added noises. They were loaded into the Jupyter notebook, and undersample was applied because the dataset is unbalanced. To make the data consistent, all the images were resized and grayscale. Lastly, PCA was used to reduce dimensionality.

Figure 25

Data Process workflow



Note. This is the workflow of how to process data.

Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are two methods to reduce high-dimensional data into lower dimensions while the critical information is retained. According to Shamir (2015, pp. 144-152), both ways were eigenvalue approaches. Still, PCA's calculation used the SVD on the covariance matrix and ranked the SVD features so that PCA could achieve more analysis work. If the data was strongly non-linear, SVD might not work satisfactorily, and the results after using SVD were not the best for visualization. SVD intensely focused on variance. It would discard useful information when sometimes variance did not have a direct relationship with predictive power. On the other hand, PCA was efficient and could be applied to big matrics, and it tended to perform sufficiently for most datasets (Lipovetsky, 2009, p.68-76). Therefore, PCA was chosen as the project dimension reduction method over SVD.

Data Collection

Data Collection Plan

The Data Collection Plan in detail is shown in Table 10 below.

Table 10

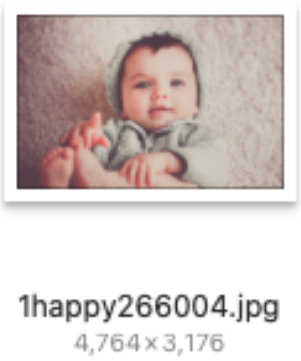
Data Collection Plan for Web Scraping

Description of the data collection					
1	Image is collected from Pexels.com				
2	Searching for images with facial emotions, including happy, sad, and angry.				
What will be done with the data once it has been collected?					
1	Filter out the images that don't meet the requirement, such as not the frontal face image, having odd angles that only show half of the face, or don't show any emotions.				
2	Manually label the photos according to the facial emotion.				
Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)					
		1	**2**	**3**	**4**
What?	Variable title	Happy facial image	Angry facial image	Sad facial image	Emotion
	Input (X) or output (Y) variable?	X1	X2	X3	Y
	Unit of measurement	pixel	pixel	pixel	character
	Data type	JPG	JPG	JPG	String
	Collection method	Web scraping from Pexels.com with free API	Web scraping from Pexels.com with free API	Web scraping from Pexels.com with free API	Manual label
	If manual	Yes	Yes	Yes	Yes
MSA	Gauge/instrument	Digital Camera	Digital Camera	Digital Camera	Cloud storage
	Gauge calibrated?	Yes	Yes	Yes	Yes
	Measurement system checked?	Yes	Yes	Yes	Yes
	Precision (R&R) adequate?	Yes	Yes	Yes	Yes
	Accuracy adequate?	Yes	Yes	Yes	Y
Historical data	Historical data exist?	Yes	Yes	Yes	N/A
	Source of historical data	website	website	website	N/A
	Historical data representative/reliable?	Yes	Yes	Yes	N/A
Who?	Data collector	Cheuk Hong lp	Jing Li	Wenbin Wang	Shufang Wen
	Operational definition exist?	Photograph	Photograph	Photograph	No
	Data collector trained?	No	No	No	Yes
	Resources available for data collector?	Yes	Yes	Yes	Yes
When?	Start date	10/25/2022	10/25/2022	10/25/2022	11/5/2022
	Due date	11/5/2022	11/5/2022	11/5/2022	11/8/2022
	Duration (in days)	10	10	10	3

In the present data collection plan, images were collected from Pexels.com that offered high-quality unedited and firsthand royalty-free images. Three categories, including happy, sad, and angry, were collected via the free API from Pexels.com within a two-week timeframe. 500 - 800 images were collected in each category in JPG format. Each image was named based on the page number, category, and ID number from Pexels.com. For example, in Figure 26, 1happy266004.jpg means this image was collected from page number 1, and the ID number from Pexels.com was 266004. Also, it is a happy face image.

Figure 26

Sample Image with File Name



Note. Sample image showing file name and file format.

Raw Dataset Samples

Images were web-scraped from the internet, and the data collection process was finished in two weeks. There are 640 raw happy images, which take 6.49 GB. There are 888 raw angry images, and they take 6.9135 GB. Also, there are 800 raw sad images, and they take 7.92 GB. In total, there are 2328 raw images, and they use 21.3235 GB of memory. Figures 27, 28, and 29 show the sample face emotion image.

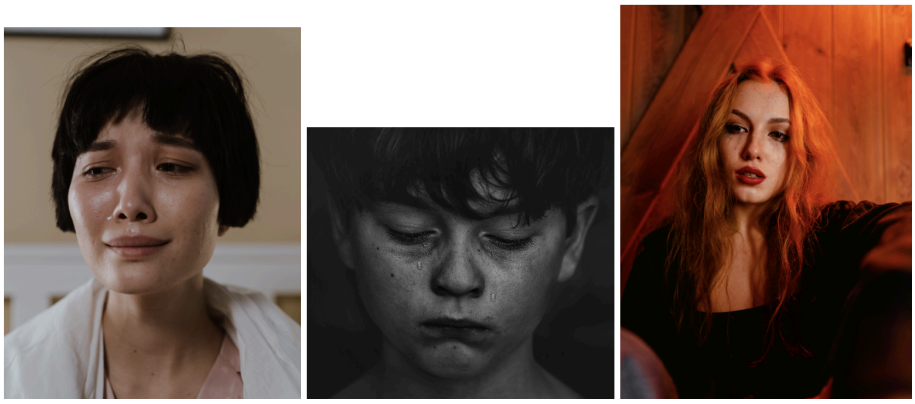
Figure 27

Happy Images

Note. This figure shows the three happy sample images.

Figure 28

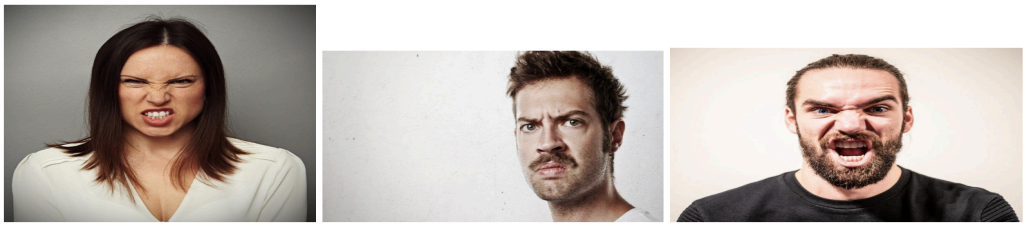
Sad Images



Note. This figure shows the three sad sample images.

Figure 29

Angry Images

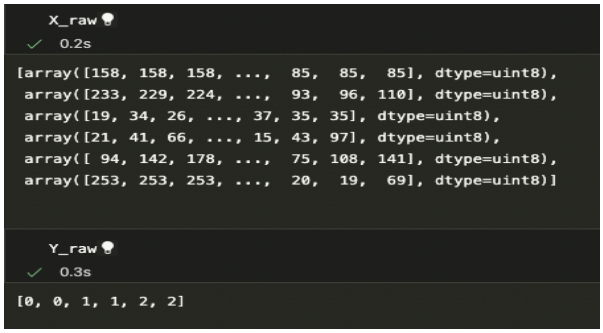


Note. This figure shows the three angry sample images.

To load all the images into the data frame, they had to be converted into pixels first. However, the data frame, CSV, and excel sheet cannot store them because every image has too many pixels. There are 18874368 pixels for each image. If they can be loaded into the data frame, there are 18874369 columns. The last column is the class (i.e., happy, sad, or angry). The first 18874368 columns is the pixels of the image. Also, there are 2328 rows since there are 2328 raw images. The dataset does not contain any null values. The column names are ordered numbers because they are the pixels for each image, and there are no specific meanings for the column names. Figure 30 shows the pixels of six sample images.

Figure 30

Six Sample Images Pixels and Class Data



Note. X_raw contains the pixels for each sample image. Y_raw is the class of the image. 0 means happy. 1 means sad, and 2 means angry. The data type for the pixels for each image is int.

Data Pre-Processing

Exploratory Data Analysis

In Figure 31, this descriptive statistic table stored the height, width, resolution, and dimension of the images and labeled those images into the category as happy, sad, and angry. Accordingly. Looking at the minimum and maximum of the images' height and width helped set the parameters' values when resizing the images. If the image dimension equals 3, it means the image is in color. If the image dimension equals 2, it means the image is in greyscale.

Figure 31

Descriptive Statistic of Raw Dataset

	length	widht	resolution	dimension
count	2327.000000	2327.000000	2.327000e+03	2327.0
mean	4444.296519	4313.709067	2.012743e+07	3.0
std	1594.592657	1613.476416	9.449735e+06	0.0
min	146.000000	180.000000	3.314200e+04	3.0
25%	3645.000000	3456.000000	1.428886e+07	3.0
50%	4419.000000	4000.000000	2.201923e+07	3.0
75%	5760.000000	5628.500000	2.416026e+07	3.0
max	10846.000000	11200.000000	8.374240e+07	3.0

Note. Descriptive statistic of raw dataset.

Outlier Detection

In Figure 32, the box plot was used to recognize any outlier and extremely low-resolution images because the pictures were blurry or indistinct when they were resized in the data prepossessing stage.

Figure 32

Box Plot Showing Outlier

Note. Box plot showing outlier from resolution.

Data Balance

In Figure 33, this bar chart could show the total of images in each category and offered insight into which images should collect more or reduce to balance out the number of images in each category.

Figure 33

Bar Chart of Number of Images in Each Class

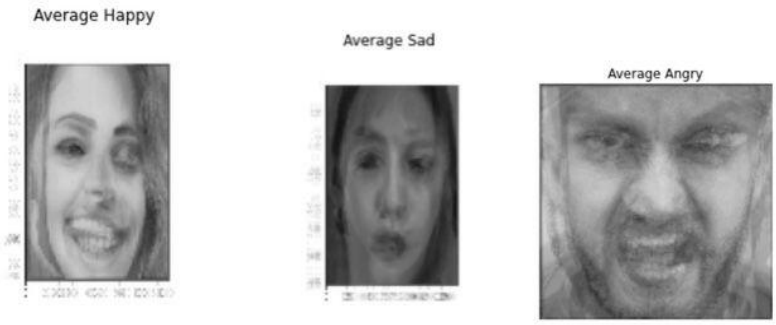
Note. Showing a number of images in each category in a bar chart to check for data balance.

Important Features

Using the average faces in Figure 34 can help figure out the essential features to distinguish facial emotions. The average faces from the Happy, Sad, and Angry categories. Figure 34 shows the critical feature of determining facial emotion. For example, the average happy face has cheeks raised and lip corners raised diagonally; the average sad face has drooping eyelids and downcast eyes; the average angry face has eyebrows pulled down.

Figure 34

Average Face of Three Emotion Category



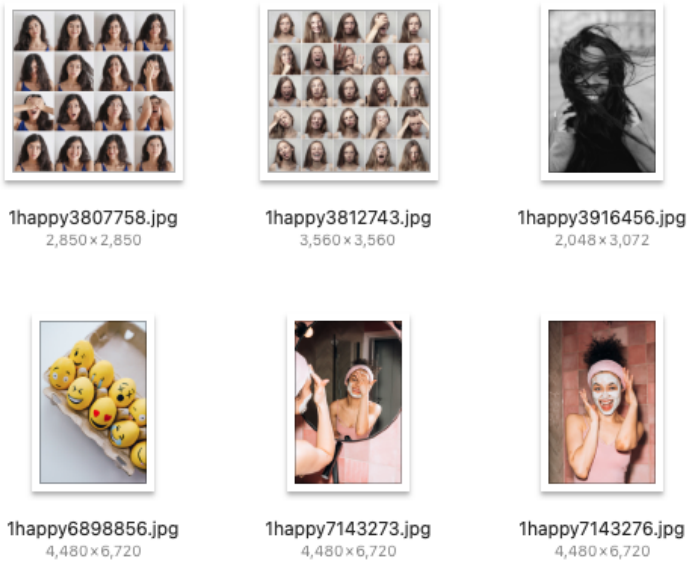
Note. Showing important features of each emotion category.

Before Data Cleaning

Since the images were being web-scraped and bulk loaded from the internet, some of the images do not contain human faces or emotions do not fit the requirement. In this case, the images were manually filtered. Also, those that do not meet the requirement were deleted to handle the data inconsistency issue. Figure 35 shows the image that doesn't meet the requirements. Figure 36 would be the ideal image and meet the requirement.

Figure 35

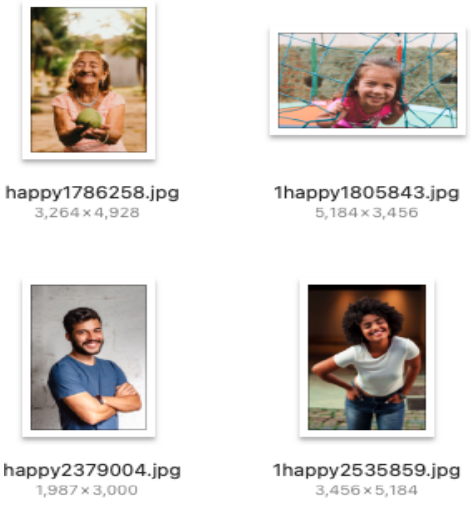
Images that Do Not Meet the Criteria



Note. Showing images that do not meet the project requirement.

Figure 36

Images that Meet the Criteria



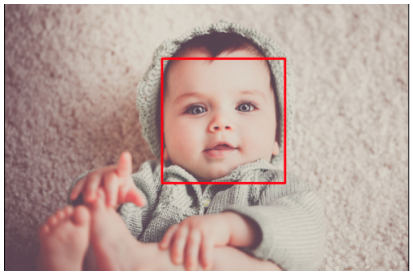
Note. Showing images that meet the project requirement.

Data Cleaning

After filtering all the images that did not meet the project requirement, images were processed using the OpenCV library. One of the biggest challenges was the background noise. In order to solve this problem, the HOG and SVM detection was used to detect faces by using a pre-trained method function in the Dlib library, and the face was captured by using the rectangular function in OpenCV. See Appendix B Figure B1 for feature extraction code implementation. In Figures 37 and 38, the baby's face was detected by the rectangular box and was cropped out to reduce the background noise. Once the faces were cropped, it was saved in a local folder, and they were manually filtered to ensure the quality of the images.

Figure 37

Sample Image of Using the HOG Detector on Baby Photo



Note. Showing images that capture the face in the red box.

Figure 38

Sample Image of Cropped Face

Note. Showing an image that is cropped from the original image.

Data Preparation

Because data augmentation has to be applied for the training set, data preparation proceeded before data transformation. Also, since the number of images for each category label was not balanced, it was desirable to split the dataset into train and test sets in a way that maintained the same proportions in each category as observed in the original dataset. Splitting the data into training and testing datasets with a ratio of 80/20 is a common choice in machine learning, and this ratio is also referred to as the Pareto principle. With the dataset containing sufficient data, 80/20 is a good starting. However, the test set should represent most of the variance of the dataset (Pawluszek-Filipiak et al., 2020). The filtered and cropped image dataset was split with a ratio of 80/20. There are 661 training images and 166 test images. There are lots of pixels, so it is impossible to load them into the data frame. Hence, they were stored with the file name and the corresponding class in the data frame. Figure 39 and Figure 40 are the lists of training/testing images.

Figure 39

Training images Data Frame

	file name	class
13	1sad551590.jpg	sad
5	1happy1181686.jpg	happy
1	1happy1130624.jpg	happy
16	1angry3799830.jpg	angry
18	1angry783941.jpg	angry

Note. This is the first five rows of the training images list. In total 661 images.

Figure 40

Testing images Data Frame

	file name	class
19	1angry53549.jpg	angry
14	1sad2847590.jpg	sad
11	1sad1378020.jpg	sad
22	1angry206481.jpg	angry
6	1happy266004.jpg	happy

Note. This is the first five rows of the testing images list. In total 166 images.

Considering the size of the data, cross-validation is a better choice than the holdout method. The dataset is not big enough to proceed properly hold out the validation set. Also, if the data set was split into training, validation, and test sets, there was a chance that there was enough data in the test set to evaluate the model's performance effectively. On the other hand, since cross-validation is computationally expensive, only five-fold cross-validation was applied.

Data Transformation

In order to increase the size of our training set to improve the models' performances, the Roboflow website was used to perform data augmentation. Every image in the training dataset was rotated in seven ways and was added noise. Hence, there were eight different images created for each training image. Figure 41 shows the output for data augmentation using one sample image.

Figure 41

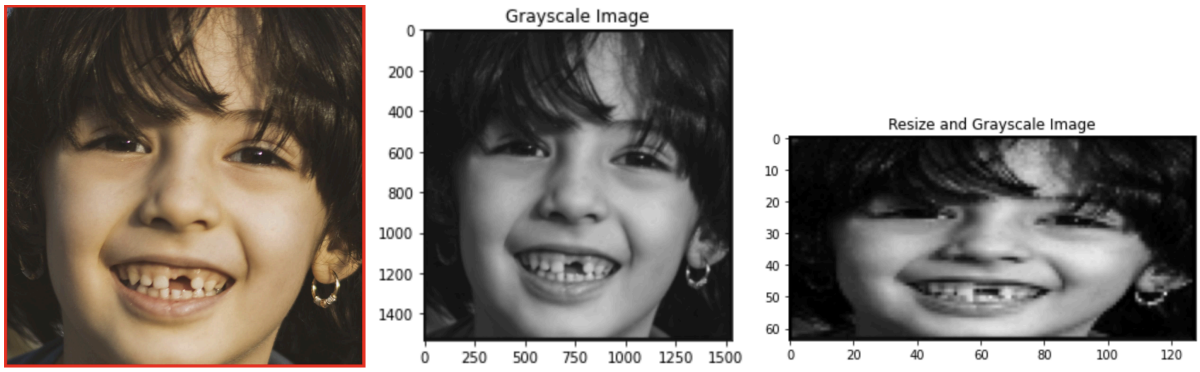
Sample Data Augmentation

Note. The first image is the original image, and the second image is the image with noise. The last seven images are rotated images.

After data augmentation, undersampling was applied to handle the unbalance data issue. Then, the images were loaded into the Juypter notebook for data transformation. The cropped face images were inconsistent because they were colorful and their sizes were different. Therefore, all the training and test set images were resized and grayscaled. This step ensured that all the images were consistent. Figure 42 below shows the outputs of resizing and grayscaling.

Figure 42

Resize and Grayscale Images



Note. The first image is the original image and the second image is the grayscale image. The last image is the image with grayscale and resizing.

Then, the images were converted into pixels and the pixels were stored in the data frame. The columns except for the class column in the data frame were the features. Figure 43 below shows the training set pixels data frame after the images were resized and grayscaled.

Figure 43

Training Set Pixels Data Frame After Resized and Grayscaled the Images

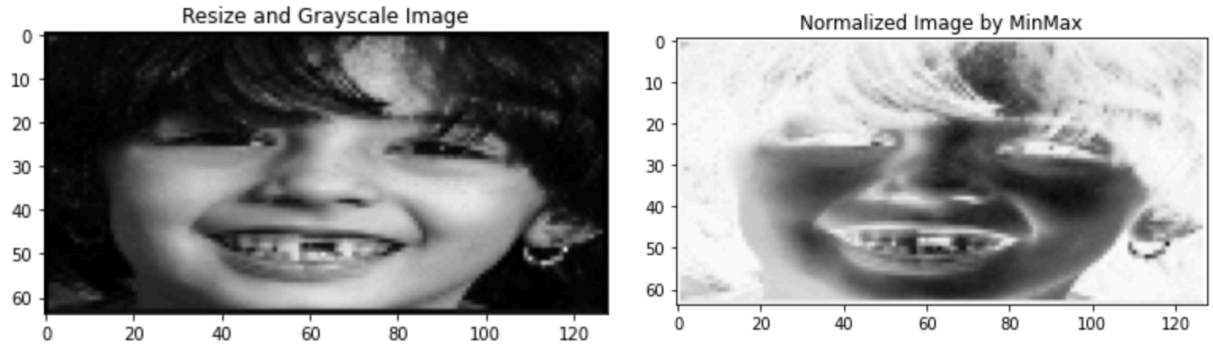
	0	1	2	3	4	5	6	7	8	9	...	8183	8184	8185	8186	8187	8188	8189	8190	8191	class
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	29	35	38	37	39	37	37	27	39	32	...	26	29	29	27	29	38	28	33	31	0
2	30	159	156	155	157	157	157	158	159	160	...	128	122	122	124	122	123	123	125	30	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	30	29	29	30	31	30	29	30	29	29	...	30	30	30	29	29	29	29	29	29	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Note. The figure shows the top five rows of the training data frame. Each row represents one image. The first 8191 columns are the pixels. The last column class is the categories of the images. 0 represents happiness. 1 represents sadness, and 2 represents anger.

After the images were resized and grayscaled, the Min-Max scaler with a range between 0 and 1 was applied. Figures 44 and 45 show the resulting images and the sample training set data frame after applying a min-max scaler.

Figure 44

Sample Image After Min-Max Scaler



Note. The first image is the resized and grayscaled image, and the second image is the first image after applying the min-max scaler.

Figure 45

Training Set Data Frame After Min-Max Scaler

	0	1	2	3	4	5	6	...	8185	8186	8187	8188	8189	8190	8191
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
1	0.117886	0.137255	0.149020	0.145098	0.153543	0.145669	0.145098	...	0.114173	0.106299	0.113725	0.149606	0.110236	0.129921	0.124
2	0.121951	0.623529	0.611765	0.607843	0.618110	0.618110	0.615686	...	0.480315	0.488189	0.478431	0.484252	0.484252	0.492126	0.120
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
4	0.121951	0.113725	0.113725	0.117647	0.122047	0.118110	0.113725	...	0.118110	0.114173	0.113725	0.114173	0.114173	0.114173	0.116

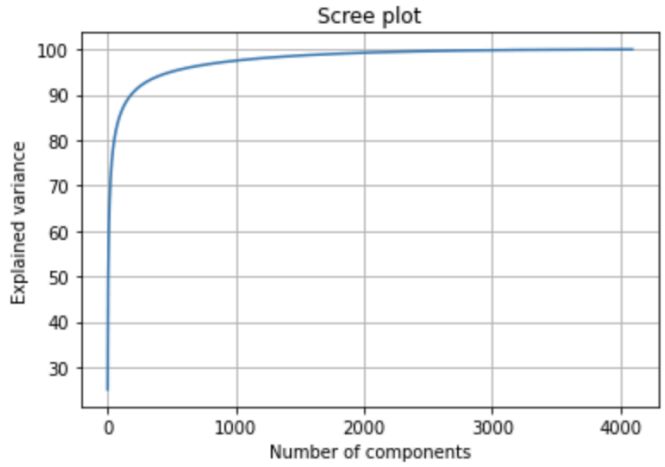
Note. It shows the top five rows and all the columns of the data frame after applying the min-max scaler.

Because there were 8191 features in the data frame, PCA was performed to reduce the feature dimension to avoid overfitting. In order to choose the best number of PCs, the scree plot was used. From Figure 46, 796 PCs were selected because they captured 97% of the variance in

the original data and the total explained variance did not increase a lot afterward. See Appendix B Figure B2 for Min-Max scaler and PCA code implementation.

Figure 46

Scree Plot For Min-Max Scaler Normalized Data Frame

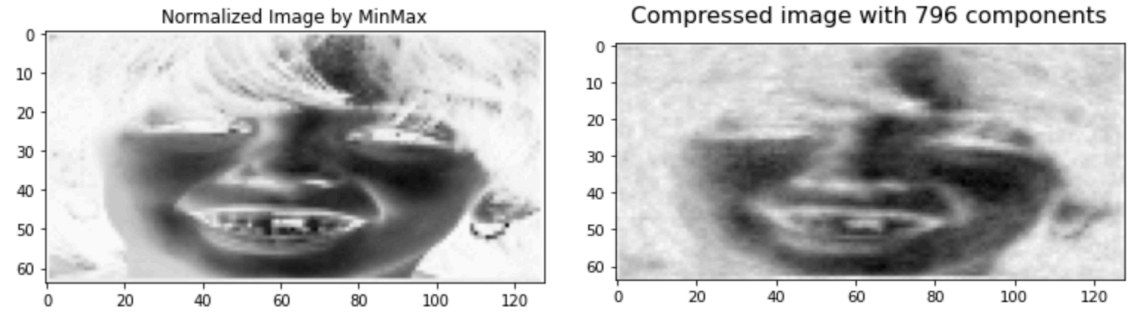


Note. This is the scree plot for the normalized dataset after applying the min-max scaler.

Figure 47 shows the sample image was only applied min-max normalization, and the sample image was applied min-max normalization and compressed with 796 PCs.

Figure 47

Min-Max Normalized and compressed image



Note. The figure on the left-hand side shows the picture that applied the min-max scaler, and the right-hand side figure shows the pictures with 796 PCs after the min-max scaler.

After performing the Principle Component Analysis and figuring out the best number of principle components using the scree plot, the data frame using the Min-Max scaler was reduced from 8192 columns to 796 columns as shown in Figure 48.

Figure 48

Training Set Data Frame After Min-Max Normalization and PCA

	0	1	2	3	4	5	6	...	790	791	792	793	794	795	class
0	-14.922464	5.033443	-4.291428	-5.377909	-2.673974	-0.442076	-3.178778	...	0.080153	0.143525	0.184311	0.142441	-0.486882	0.500702	0
1	1.227210	6.855914	3.284829	2.134742	-4.173556	-0.131871	-0.176243	...	0.092863	-0.090462	-0.173765	0.010208	0.048242	-0.009578	0
2	-10.466131	3.801826	-0.076059	-0.142212	-6.795634	0.693070	0.587115	...	0.153429	-0.039235	-0.076902	-0.103526	-0.093604	0.104499	0
3	-0.291550	0.134163	2.101741	-0.276473	-4.555051	-2.997156	0.187179	...	-0.098393	-0.046519	0.212531	0.522568	0.106917	0.328793	0
4	-7.083975	-3.525638	4.054232	0.288740	4.363773	-6.333950	0.444656	...	-0.056845	0.165073	-0.036822	-0.152718	0.023589	-0.071052	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3484	16.243427	1.164604	6.203519	1.043491	-4.654441	1.694157	-1.428514	...	-0.017741	-0.079990	0.119614	0.035200	-0.058562	0.016073	2
3485	-11.231110	3.368416	1.495940	9.340180	6.275326	1.532026	-3.587100	...	-0.038051	-0.022786	-0.068932	-0.083994	-0.124310	0.116441	2
3486	-3.506366	3.136393	1.696787	8.290785	6.275972	-2.576262	-1.952776	...	0.159753	-0.037254	0.034715	0.030747	-0.055056	-0.232813	2
3487	9.544817	-0.677706	3.601819	1.810476	2.907359	-1.686533	-2.439230	...	-0.123835	0.103740	-0.189402	-0.178245	0.371590	-0.312255	2
3488	0.910446	2.053325	-10.187512	-0.638319	-1.829592	-2.937545	-3.997237	...	0.076957	-0.044277	-0.112178	-0.036241	0.023788	0.006683	2

3489 rows × 797 columns

Note. It shows the rows and columns of the training set data frame after applying min-max normalization and PCA.

Data Statistics

The raw dataset contains three parts, happy, angry, and sad. The happy dataset has 640 images and a size of 6.49GB. The angry dataset, which includes images from Google, contains 888 images and 6.9135GB, and the sad dataset contains 800 images and size of 7.92GB. In total, 2328 images, 21.3235GB.

After the completion of face detection using HOG and SVM, the recognized faces were chopped from the original image. The images that could not be detected were dropped. Then, the happy dataset has 590 images, 263MB, the angry dataset has 566 images and 369.6MB, and the sad dataset has 477 images, 180.5MB. In total, 1633 images and 813.1MB. Compared to the last step, the image size is reduced significantly.

Then each dataset was handed to the team members for manual filtering. The images that are blurred or don't contain the face was dropped. The filtered happy dataset has 290 images, and 221MB, the angry dataset has 332 images and 186MB, and the sad dataset has 205 images, and 147MB. In total, 827 images and 554MB.

The filtered image dataset was split into the training and testing set by the Roboflow website by the ratio of 80:20. Then for the purpose of higher predicting accuracy, the image augmentation was done to the data to increase the volume of the training set.

Then, the happy set is increased to 1963 images, 449MB for the training set, and 58 images, 12.2MB for the testing set. The training set for angry has 2226 images and 373MB, and the testing set has 67 images and 9.49MB. And for the training set of the sad dataset has 1368 images, 308MB, and the test set has 41 images and 6.69MB. In total, 5723 images and 1158.38MB.

Next, the under-sampling was performed to balance the datasets, and then all the images were grayscaled, resized, and loaded into the data frame. Then, the training set contains 4104 rows and 8193 columns, and 282MB. The testing set has 167 rows and 8193 columns, and 11.4MB.

After the images were loaded into the data frame in pixel format. The min-max normalization technique and PCA were performed. The data frame was even reduced to a smaller size. The training set is 4104 x 797, 56.2MB, and the testing set is 167 x 797, 1.81MB. Figures 49, 50, and 51 show the changes in data after each processing step. Table 11 shows the summary of data statistics after each processing step.

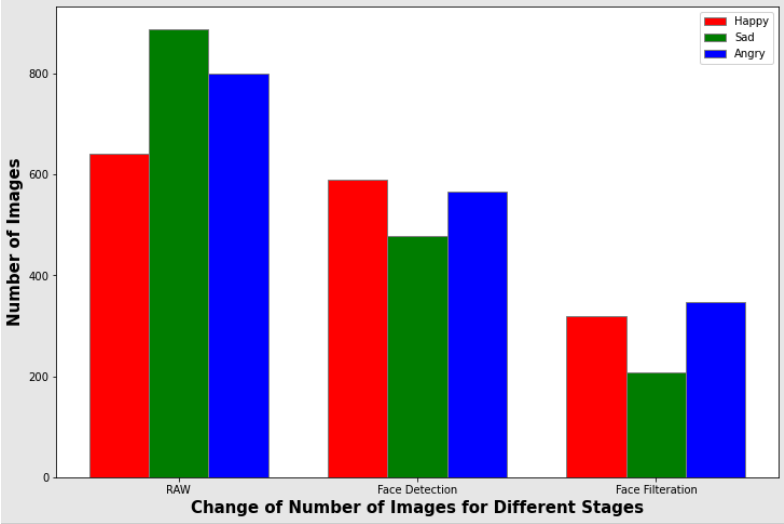
Table 11

Statistics of every dataset during each stage of the data process

Stage	Dataset	Statistics (Images/Rows x Columns)
Raw	Happy	640
	Sad	888
	Angry	800
Pre-processing	Face detection	
	Happy	590
	Sad	477
	Angry	566
	Face filter	
	Happy	290
	Sad	205
	Angry	332
Preparation	Training	
	Happy	232
	Sad	164
	Angry	265
	Testing	
	Happy	58
	Sad	41
	Angry	67
	Validation	five-fold cross-validation
Transformation (the second column might change depending on your data transformation approaches)	Augmentation	
	Happy	train: 1963, test:58
	Sad	train: 1368, test:41
	Angry	train: 2226, test:67
	Resize & Greyscale	train: 4104 x 8193, test: 167 x 8193
	Min Max + PCA	train: 4104 x 797, test: 167 x 797

Figure 49

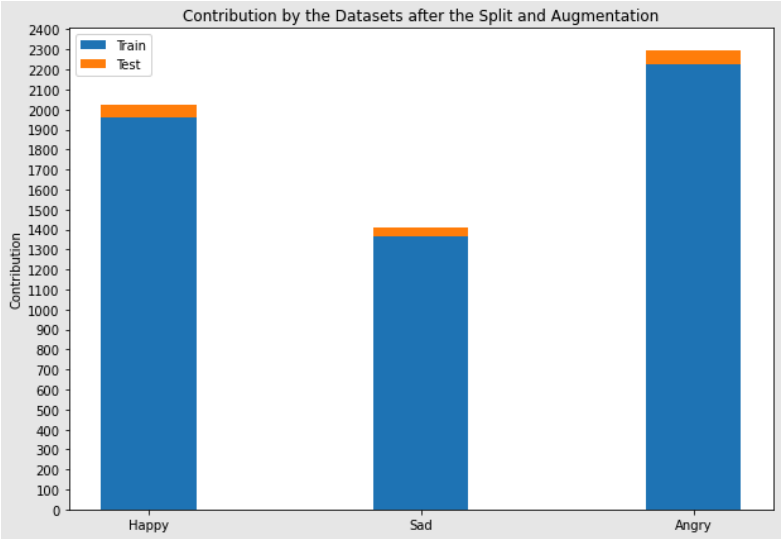
Change of Number of Images for Different Stages



Note. It shows the number of images from raw to filtered.

Figure 50

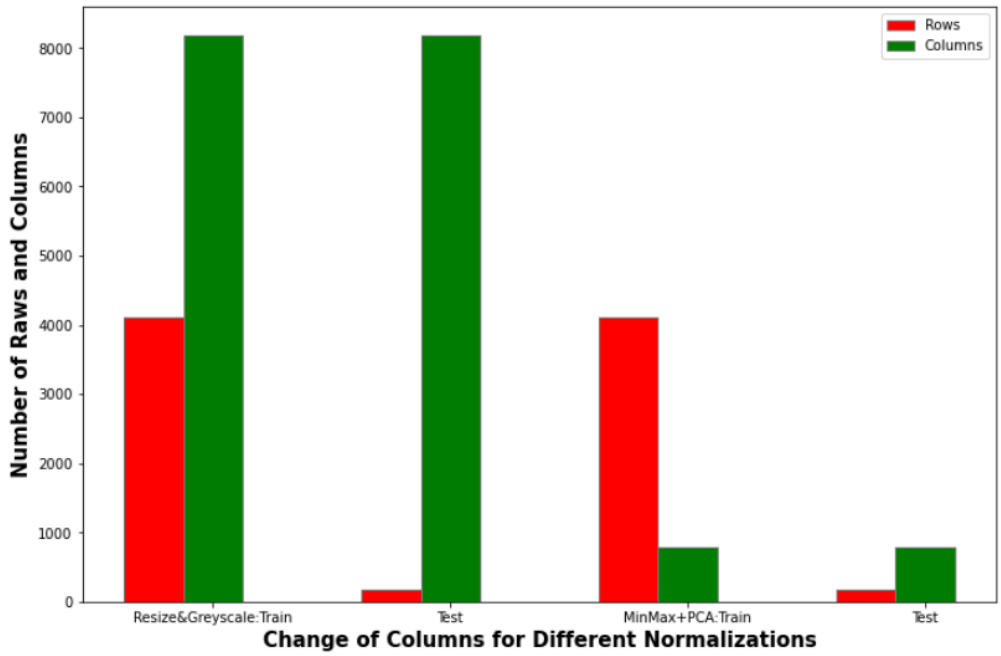
Contribution by the Datasets after the Split and Augmentation



Note. It shows the train and test distribution of the three datasets.

Figure 51

Change of Columns for MinMax Normalization and PCA



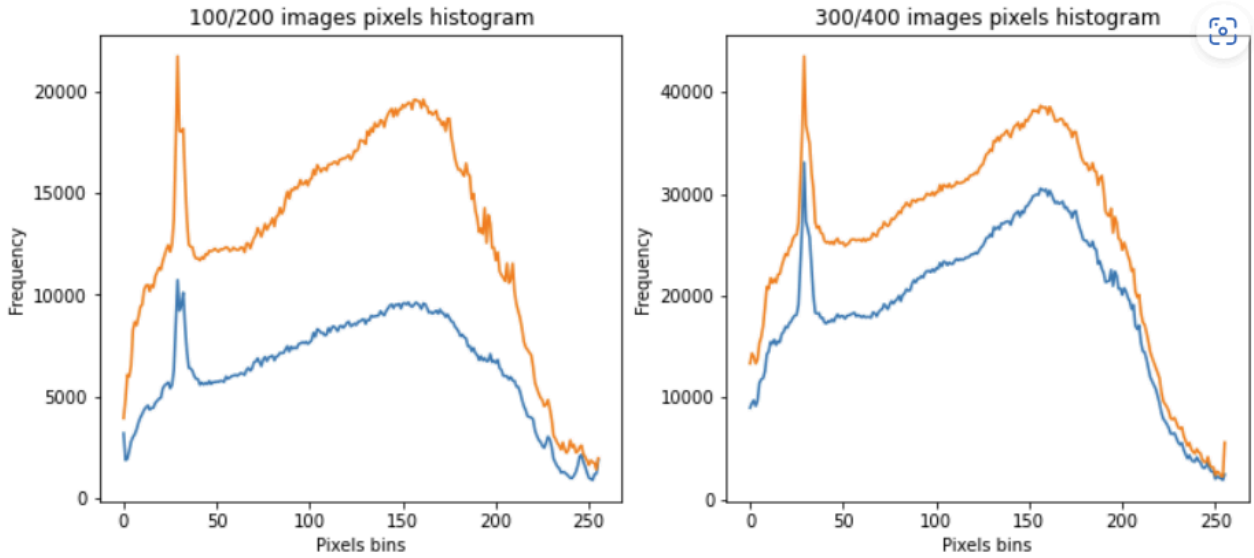
Note. It shows the reduction in columns for both the train and test set after the MinMax and PCA.

Data Analytics Results

This project was using the image dataset. There are some interesting functions in the Opencv library, which can use analytics graphs to visualize the image's pixels. The frequency of pixels in the range of 0-255 was analyzed. 100/200/300/400 images were randomly chosen in the dataset to accumulate the pixels. Then, the histograms were created to visualize the distribution of the total pixels. Figure 52 shows the pixel value histogram of different images.

Figure 52

Pixel Histogram of multiple images



Note. These are pixels frequency histograms for 100/200/300/400 images. In the left graph, the orange line is for 200 image pixels and the blue line is for 100 image pixels. In the right graph, the orange line is for 400 image pixels and the blue line is for 300 image pixels.

An average face image was created to explore the important features, and the BGR combination of the average image was analyzed. Figure 53 is the average image of 100 images in the 'angry' class of the training dataset. Figure 54 is the graph that plots each 2D color histogram one for each BGR combination pair, separating the image into different value channels in order to work with the channels exclusively.

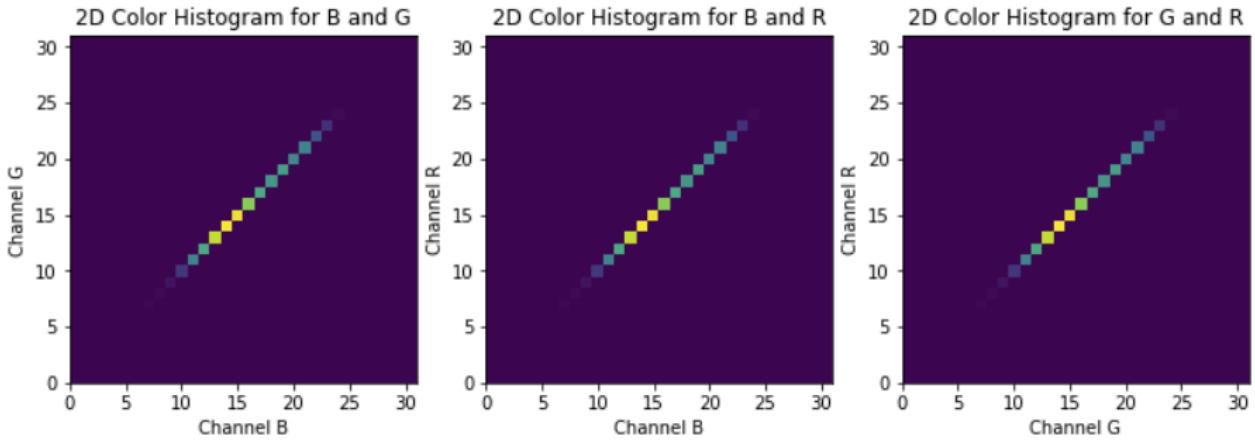
Figure 53

Average of 100 images

Note. This is the image of 100 images on average.

Figure 54

2D color Histogram

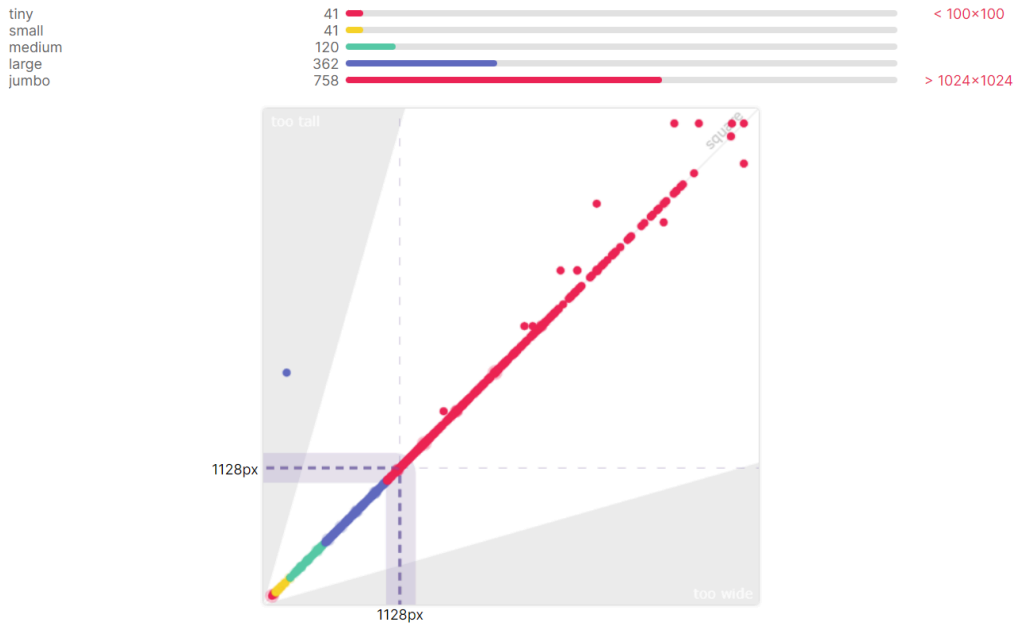


Note. These are 2D color histograms for 100 images on average.

After the data pre-processing, the dimension insights of the whole dataset was analyzed. Figure 55 shows the size distribution of the whole dataset, and the purple box indicates the median width by median height image (1128 x1128). The aspect ratio distribution of the dataset shows most of the images are square.

Figure 55

Dimension distribution of image dataset



Note. Dimension (size & ratio) distribution of the whole dataset.

Chapter 4 Modeling

Model Proposals

Random Forest

Random Forests, more frequently referred to as RF, is a technique for ensemble learning that has lately gained a lot of popularity. The approach entails constructing numerous random Decision Trees, then they are combined to generate a Random Forest (RF), which may be put to use for either classification or regression. Training such RFs is simple and does not take up much time. As a consequence of the significant capabilities that voting averaging approaches provide to RF, they have been exceedingly effective in a wide range of cross-field applications including cheminformatics, bioinformatics, ecology, computer vision, and data mining (Wang et al., 2018, pp. 3510-3523).

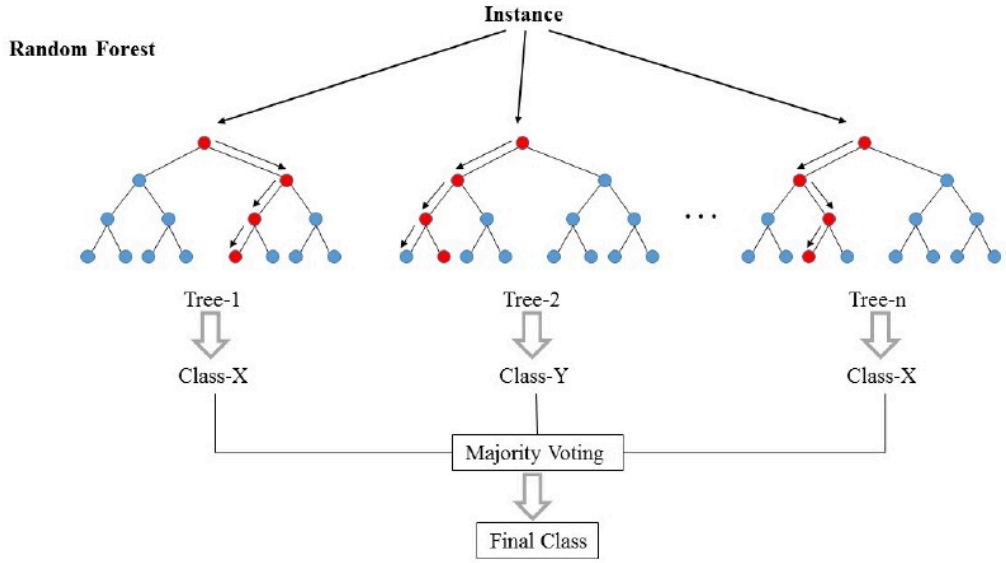
Kremic and Subasi (2016) demonstrated the use of the Random Forest machine model to solve the face recognition problem aim for mobile applications. They applied skin color detection, and greyscale for the images for the preprocessing, and then the Random Forest achieved 0.972 F-measure with 0.84 seconds of model build time (pp. 287-293).

Wang et al. (2018) stated in their article that supposes the dataset Dn has n cases like (X, Y) in which $X \in R^D$. The Breiman approach creates a forest by combining a number of decision trees that have been trained independently of one another. The process of constructing each tree can be treated as a different division of the available data space. So that if the whole data space is represented by the notation R^D, then a leaf corresponds to a partition of R^D, and each node is related to a hyperrectangular cell. The Random Forest algorithm's specifics are described in more depth down below (pp. 3510-3523).

In the first step of the tree-building process, the RF model makes a random selection of n data points from the dataset Dn that has been provided to the model with replacement. The construction of the present tree only uses these particular bootstrap samples. Use of classification and regression trees. For each node in the tree, the mtry features are randomly drawn from the initial D features (mtry D), and these features are then utilized to choose the split features and the split points. Depending on whether one is attempting classification or regression, the criteria is the largest decrease in Gini impureness or the maximum reduction in mean square error (MSE). Following the aforementioned steps, nodes are added to the tree sequentially until a halting condition is met, such as when the size of an instance reaches a certain threshold in a leaf node. To make its predictions, the RF algorithm takes the average of all of the trees' votes on Y, also known as the majority vote of Y for classification (Wang et al., 2018, pp. 3510-3523). The RF algorithm visualization is shown in Figure 56.

Figure 56

Random forest algorithm



Note. Dimitriadis and Liparas (2018, pp. 962-970). *Random forest algorithm.* https://doi.org/10.4103/1673-5374.233433

The Random Forest machine learning model has many parameters to tune, and there are a few parameters that are more significant than the others, which are the number of features, subsample size, number of trees, and tree depth. There is no theory that can be utilized to determine the number of iterations that should be conducted on each cell's preselected attributes before separating them. If mtry is small and the direction of the split is uniformly random in all directions, the CART method with mtry equal to d will perform computationally better than the original tree. If mtry is too big, the computation time is the same as what it would be for the tree(cart, mtry = d), and the splitting direction is near the optimal splitting direction. But where exactly to find a good compromise between those two options is not quite evident. It appears that the default number is too low based on past experience. For the number of trees, the number of repetitions in Monte Carlo simulations corresponds to the number of trees that have an effect on

forest performance. For optimal performance, the forest must contain enough number of trees, but this increases the computing time linearly. As a consequence of this, the optimal number of trees is reached when the inaccuracy in the forest is at its highest. After making adjustments to the remaining forest parameters, the error in the forest may be seen as a function of M and is almost brought up to its maximum value. Both the subsample size and the tree depth of the random forest will remain the same after the modifications to the parameter values. The nodesize parameter determines the maximum number of observations that may be stored in a single cell, the maxnodes parameter determines the maximum number of terminal nodes, and the tree levels kn parameter guarantees that no more than kn levels of the subdivision is used for any given cell (Scornet, 2017, pp. 144-162).

Support Vector Machine

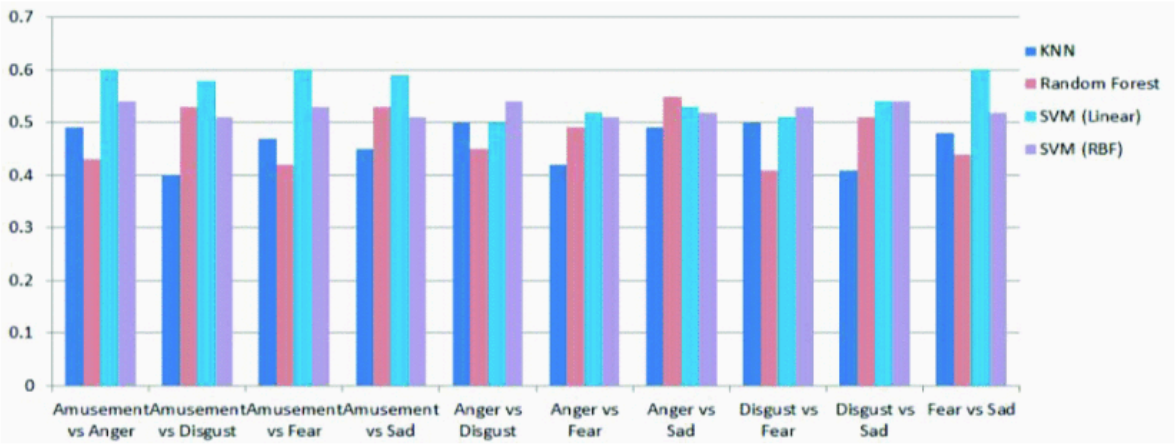
Support Vector Machine (SVM) is a robust supervised machine learning algorithm. There have been various breakthroughs in the past few years regarding facial feature extraction mechanisms that use SVM for expression classification. SVM has been broadly applied in machine learning and pattern recognition since it can effectively determine "non-linear and high-dimensional problems for an optimal global solution and have powerful generalization ability without over-learning" (Yao et al., 2021, p. 24292). Because of SVM's efficiency and non-linear classification ability could effectively lower the probability of classification errors (Hsieh et al., 2016). The SVM approach is also robust to noisy data and accommodating to some misclassifications through training a classification model that can identify any outliers (Tsai & Chang, 2018). The fundamentals of SVM classification can be summarized as splitting hyperplanes in n-dimension space that can distinguish different levels of classes from the target features. Then, find the proper margin, the distance from the decision boundary close to the

training data, defined as the support vector. (Kelleher et al., 2015, p. 436). "The data in the low-dimensional input space is mapped to the high-dimensional space through the kernel function in linear non-separable problem and in the original low-dimensional space can be transformed into a linearly separable problem in the high-dimensional space" (Yao et al., 2021, p. 24291).

Mostafa et al. (2019) used the BioVid Emo DB dataset to perform face emotion identification. The collection includes 430 videos contributed by 86 different people. They began by separating the dataset into a training set and a test set before using five-fold cross-validation to evaluate the effectiveness of the various models (pp. 417-422). It can be seen in Figure 57 below that the SVM performs the best when it comes to identifying facial expressions.

Figure 57

Appearance-based features results



Note. Mostafa et al. (2019, pp. 417-422). *Appearance-based features results. https://doi.org/10.1109/ICCES.2018.8639182*

According to the paper written by Olatomiwa et al. (2015), they concluded the algorithm for the Support Vector Machine. Given the SVM model, the dataset contains data in form of

$\{x_i, d_i\}_i^n$. The x_i is the vector from the input data, and the d_i is the target record, and the n is the number of rows in the dataset (pp. 632–644). The formula is shown in Figure 58.

Figure 58

The formula of SVM

$$f(x) = w\varphi(x) + b$$

$$R_{SVM}(C) = \frac{1}{2}||w||^2 + C\frac{1}{n}\sum_{i=1}^{n} L(x_i, d_i)$$

Note. Olatomiwa et al. (2015, pp. 632–644). *The formula of RF.* https://doi.org/10.1016/j.solener.2015.03.015

In the above formulas, $\varphi(x)$ maps the input vector x_i, representing it as high dimensional-space features. The w is just a vector, the $C\frac{1}{n}\sum_{i=1}^{n} L(x_i, d_i)$ part is the empirical risk, and the value b is a scalar. Estimating the values of the parameters w and b requires first adding a positive slack variable which contains the upper and lower deviations, and then minimizing the regularized risk function (Olatomiwa et al., 2015, pp. 632–644). And the kernel functions of the SVM are usually seen in Figures 59, 60, and 61.

Figure 59

Polynomial kernel function

$$kernel(\mathbf{d}, \mathbf{q}) = (\mathbf{d} \cdot \mathbf{q} + 1)^p$$

Note. Kelleher et al. (2015, p. 440). This is the Polynomial kernel function.

Figure 60

Gaussian Radial Basis Function

$$kernel(\mathbf{d}, \mathbf{q}) = \exp(-\gamma \|\mathbf{d} - \mathbf{q}\|^2)$$

Note. Kelleher et al. (2015, p. 440). This is the Gaussian Radial Basis Function.

Figure 61

Linear Kernel Function

$$kernel(\mathbf{d}, \mathbf{q}) = \mathbf{d} \cdot \mathbf{q} + c$$

Note. Kelleher et al. (2015, p. 440). This is the Linear Kernel Function.

There are numerous methods to improve the predictive power of the SVM model, such as TSMO, PSO, and tDPSO. The proposed TSMO first locates candidates for support vectors, and then focuses its efforts only on these candidates in order to accelerate the training process. The TSMO lowers the amount of money spent on computations without compromising accuracy. PSO is a fantastic option since it has the possibility of doing quick searches. However, the issue of early convergence that plagues PSO can not be ignored; this must be addressed. A two-stage differential learning system is the suggested theoretical contribution of the PSO that was just described. The curriculum offers three different levels of study to choose from. By operating in this manner, PSO is able to strike a healthy balance between its exploration and exploitation activities. The results of the experiments show that the SVM combined with the tDPSO performs better than the alternatives when used to undersampled datasets. The innovative tDPSO approach is utilized for estimating the parameters of the SVM (Grandini et al., 2020).

Gaussian Naive Bayes

Kelleher et al. (2015) mentioned that Naive Bayes is a probability-based algorithm. Compared to other models, it is not robust. However, it can usually provide reasonably accurate predictions for classifying categorical target features. Also, it is easy to train and robust to the curse of dimensionality. Therefore, it is widely used to set a baseline accuracy score or when the data is limited (p. 301). Ayache and Alti (2020) used Naive Bayes to classify facial emotions using images. Compared to other classifiers that they used, Naive Bayes obtains the second least computation time. It used 0.0087 seconds, while Random Forest used 0.7976 seconds. The computation time differs a lot. Also, it achieves 100% accuracy when the test size is seven and 60% accuracy when the test size is 70. Therefore, Naive Bayes can be trained relatively fast (pp. 267-275). Sharma et al. (2020) implemented the Gaussian Naive Bayes model to detect facial emotion using the CK+ dataset. The accuracy for the Naive Bayes model is 93.16%, which shows it is useful and accurate for facial emotion recognition (pp. 1162-1168). Therefore, the Gaussian Naive Bayes model was chosen as one baseline model to detect facial emotions (pp. 1162-1168).

According to Kelleher et al. (2015), the Naive Bayes model returns the conditional probabilities of the target feature and assumes that the descriptive features are conditionally independent (pp. 340-341). More specifically, the Naive Bayes model is defined as shown in Figure 62.

Figure 62

Naive Bayes Model

$$\mathbb{M}(\mathbf{q}) = \underset{l \in levels(t)}{\arg\max} P(t = l \mid \mathbf{q}[1], \ldots, \mathbf{q}[m])$$
$$= \underset{l \in levels(t)}{\arg\max} \frac{P(\mathbf{q}[1], \ldots, \mathbf{q}[m] \mid t = l) \times P(t = l)}{P(\mathbf{q}[1], \ldots, \mathbf{q}[m])}$$

Note. Kelleher et al. (2015, p. 292). *Naive Bayes Model.* t is the target feature. q is a query instance of the descriptive features. $q[i]$ is the i^{th} descriptive feature.

The Naive Bayes model has a hyperparameter that is var_smoothing. It is a portion of the largest variance of all characteristics that contributed to the variances to ensure computation stability. The default value is 1^{-9}. The default value works well for a specific dataset. However, most of the time, the best accuracy will occur when it is not at the default value (Kuruvila et al., 2020, pp. 452-457). Hence, different values for var_smoothing can be used to optimize the model.

Logistic Regression

Stoltzfus (2011) mentions that regression analysis is compelling in multiple fields since it can predict numerical and categorical variables, discover relationships between variables, etc. Logistic Regression is one of the regression analysis techniques. It is an extension of linear regression because the output of the linear regression simply passes through the sigmoid function shown in Figure 63 (pp. 1099-1104).

Figure 63

Sigmoid/Logistic Function

$$S(x) = \frac{1}{1 + e^{-x}}$$

Note. Stoltzfus (2011, p. 1100). *Sigmoid/Logistic Function.* https://doi.org/10.1111/j.1553-2712.2011.01185.x

However, it is efficient and powerful for classifying categorical target features because it can determine the most robust linear combination of variables with the highest likelihood. Also, the variables passed into the logistic regression must be independent. The output of the logistic regression is the probability, and it is shown in Figure 64 (pp. 1099-1104).

Figure 64

Output of Logistic Regression

$$\text{Probability of outcome}(\hat{Y}_i) = \frac{e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_i X_i}}{1 + e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_i X_i}}$$

Note. Stoltzfus (2011, p. 1100). *The output of Logistic Regression.* https://doi.org/10.1111/j.1553-2712.2011.01185.x

According to Kelleher et al. (2015), it is widely used for classification problems because it works well for predicting categorical target features (p. 383). Since the project aims to detect facial emotions from images, this is a classification problem. Therefore, logistic regression is a good fit.

Regularization impacts logistic regression with high dimensional data (Salehi et al., 2019). The first training and test sets have 796 dimensions. The second training and test sets

have 822 dimensions. Therefore, the logistic regression model can be optimized using L2 regularization and the regularization strength.

K Nearest Neighbor

KNN algorithm has wide usage in many analysis and commercial fields like 3D entity rendering, content-based image recovery, customer behavior analysis, medic (symptom classification), etc. For this project, K Nearest Neighbor (KNN) was chosen as the first model which was also considered the baseline model, since KNN is one of the most elemental and simple supervised learning models. The fundamental idea of KNN is the similarity-based approach, or in other words, feature similarity measurement (Kelleher et al., 2003, pp. 225-228).

KNN is effective for classification in many cases. Still, the main weakness is being a 'lazy' machine learning model restricts it in certain fields, such as active web mining for large data volume. And the prediction performance is significantly dependent on the selection of value K (Guo et al., 2003, pp. 986–996).

According to Hall et al. (2008), the absence of techniques for the practical choice of k hampers the application of KNN. This paper details how the value of k resolves the misclassification error. They considered building two training models, Poisson and Binomial, to find new methods for choosing the value K. The first model arranges data in a Poisson stream, and training samples follow a Poisson distribution. The second model fixes the total number of training data, and the value is assigned randomly. These two models have different value risks but are almost identical to the first order. In addition, the risks correlated with kernel-based algorithms customized to two derivatives (pp. 2135-2152).

Feature space of instances and similarity measurement are two essential components of the nearest neighbor algorithm (Kelleher et al., 2003, pp. 225-228). The training stage is to build

a model that involves storing training data in memory as a simple list. During the prediction, calculate every distance between the query and every sample in the memory, and the model returns the outcome of the instance's target features which are nearest to the query. The simplest way to validate the similarity is computing the distance between the query and instance, which is also called distance metric. "Mathematically, a metric must conform to the following four criteria: Non-negativity, Identity, Symmetry, Triangular Inequality" (Kelleher et al., 2003, p. 252). The default distance metric used in KNN is the Euclidean distance, which simply calculates the straight length of two points in 2D space. As far as in m-dimensional space, the Euclidean distance function is shown in Figure 65.

Figure 65

Euclidean Distance

$$Euclidean(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{m} (\mathbf{a}[i] - \mathbf{b}[i])^2}$$

Note. Kelleher et al. (2015, p. 221). This is the Euclidean Distance.

Another distance function is the Manhattan distance or called taxi-cab distance, that convention comes from this method is very similar to the route of a taxi driver driving from one location to another in the city block road system (Kelleher et al., 2003, p. 221). In m-dimensional space, the Manhattan distance is defined as shown in Figure 66.

Figure 66

Manhattan Distance

$$Manhattan(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i])$$

Note. Kelleher et al. (2015, p. 221).This is the Manhattan Distance.

Euclidean and Manhattan distance are both distance metrics to measure the differences; actually, they're sub-class of Minkowski distance, which is the fountainhead of distance metrics (Kelleher et al., 2003, p. 222). The definition of this distance is displayed in Figure 67.

Figure 67

Minkowski Distance

$$Minkowski(\mathbf{a},\mathbf{b}) = \left(\sum_{i=1}^{m} abs(\mathbf{a}[i] - \mathbf{b}[i])^{p}\right)^{\frac{1}{p}}$$

Note. Kelleher et al. (2015, p. 222).This is the Minkowski Distance.

When a different positive value of p is set, it can form other distance metrics. For example, it's Manhattan distance when p equals one. Hence, there is an endless amount of distance metrics. However, Euclidean distance and Manhattan distance are the most commonly chosen for the KNN algorithm. Furthermore, the second one has a slight advantage over the first one since the squaring and square root is omitted in Manhattan distance, which computational cost should be considered when dealing with a vast dataset. Beyond computational consideration, Euclidean distance is set as default.

The pseudocode description of the sequential nearest neighbor algorithm is described as follows. Firstly, three inputs to train the model: a series of training instances d which should contain features space and target feature, a query instance q, and the neighbor's number k. During the prediction, loop through every single instance in the set of training instances and calculate each distance between query q and training instance d[i]. The calculation result is stored in a list D. Then, sort the list D in ascending order and the first k neighbors is obtained. At last, the first k neighbor’s target feature is acquired, the majority vote of which would be the prediction.

KNN algorithm is able to work well with clean and regular-size datasets. However, the data is not clean enough, or the size is huge, even may include various data types. Thus lots of extensions of the algorithm are developed to handle these cases. Consequently, KNN is susceptible to noise because any false in the description and target features would cause inaccurate models and eventually inaccurate predictions. Therefore, the most straightforward way to reduce the influence of noise data is by weakening the dependency on potentially noisy instances. The simplest solution is to set the model to return the majority target feature within the k nearest neighbors.

However, the trade-off of setting value k should be considered. The model would be acute to noise data and overfitting if the k value is low. On the contrary, if the k is overly high, the risk of losing the characteristic of training data and causing under-fitting occurs. Especially model associated with high-value k is more sensitive when handling an imbalanced dataset. The first common approach to address this problem is to apply the evaluation experiment, scikit-learn library GridSearchCV is applied. Another way to fix this issue is to use a weighted k nearest neighbor. The contribution of each neighbor to the distance metrics is in an inverse way when using the weighted k. Thus when computing the majority vote, the neighbors closer to the query

get more weight. This approach returns a prediction in terms of the highest target score when totaling the weights of all votes for each target feature. In other words, the value of the training size was set to the k value, so it has the drawback of being computationally costly when the dataset is large.

Model Supports

Environment, Platform, and Tools

The model implementation project ensures that the hardware can run the model correctly. In addition, the tools and software are updated to the version capable of using all the functions properly in the Python library. These are one of the project's priorities and one of the essential requirements to complete the project. In this project, two sets of setup and configuration were used.

The first setup included the following, the CPU is Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz with eight Cores and 16 Logical processors. The memory used is 32GB of RAM with 2133MHz DDR4 speed. And then the GPU used is NVIDIA GeForce RTX 2070 SUPER with the updated driver, DirectX12, and eight GB of GPU RAM. The Windows 11 system OS was used for the software configuration setup, and the model implementation, training, and testing used Windows-compatible Anaconda Jupyter Notebook, and Python3.8.

The second setup used the Mac Mini M1, including the configuration with the eighth core of the CPU with four performance and efficiency cores, eight cores of GPU, eight GB of memory, and one TB of SSD storage. Second, three of the software employed in the individual model implementation are Visual Studio Code (VSCode), Jupyter Notebook, and Python version 3.10. Visual Studio Code is an all-in-one open-source integrated development environment (IDE) that includes all the popular programming languages, such as Python, C++, Jupyter Notebook,

JAVA, JavaScript, and the rest, without switching the editor for different programming languages. Lastly, Scikit-learn, Pandas, Numpy, OpenCV, Mathplotlib, Seaborn, OS, and Glob are the Python libraries applied in the individual model implementation.

For example, in the Scikit-learn library, the function of train_test_split is to split the training and testing dataset, the SVC is the SVM classifier to train the model, GridSerachCV is used to find the best parameter for the training model, MinMaxScaler is used to normalize the data from zero and one, BaggingClassifier is used to ensemble the model, and classification_report, confusion_matrix, and accuracy_scores is used to evaluate the model and show evaluation matrix. On the other hand, Pandas is used to create a data frame and read and save CSV files, Numpy is used to calculate the statistics such as mean and standard deviation, Mathplotlib and Seaborn are used to make a visualization, and OS and Glob are used to get file path and name, and load the file in the folder. The detailed libraries, methods, tools, and purpose of use are shown in Table 12.

Table 12

Tools and Libraries Used

Library	Method		Usage
Scikit-Learn	sklearn.model_selection	train_test_split	Split dataset into train and test set
	sklearn.svm	svc	Implement Support Vector Machine model
	sklearn.ensemble	RandomForestClassifier	Implementation of Random Forest model
	sklearn.neighbors	KNeighborsClassifier	Implementation of KNN model

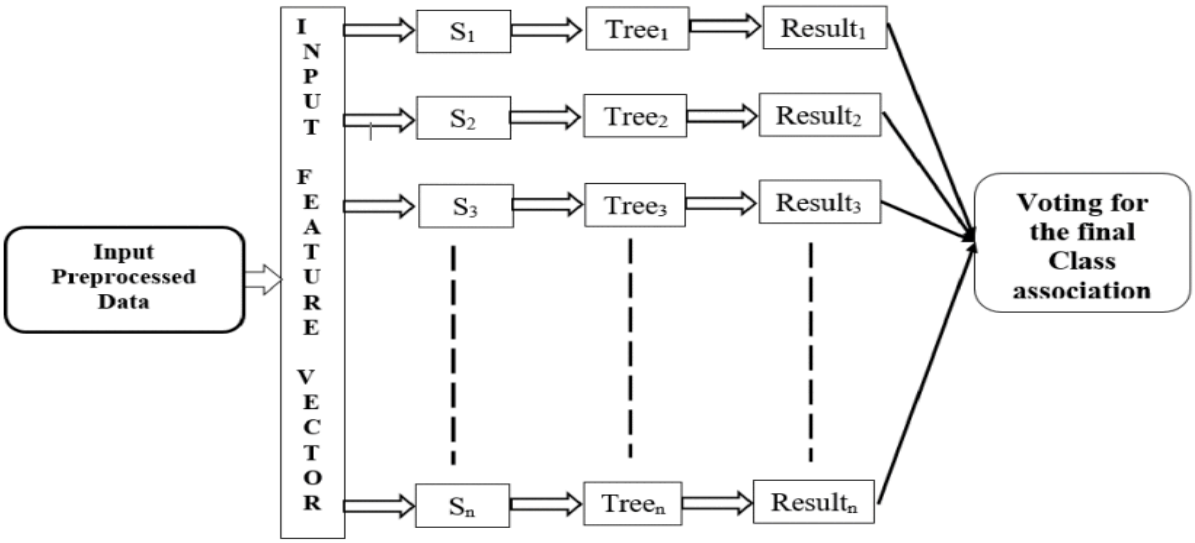
Library	Method		Usage
	sklearn.naive_bayes	GaussianNB	Implementation of Naive Bayes model
	sklearn.linear_model	LogisticRegression	Implementation of Logistic Regression model
	sklearn.model_selection	RepeatedStratifiedKFold, cross_val_score, KFold	Cross-validation for models
	sklearn.model_selection	GridSearchCV	Parameter tuning
	sklearn.ensemble	BaggingClassifier	Reduce variance
	sklearn.metrics	classification_report, confusion_matrix, accuracy_score, precision_score, recall_score	Evaluate the models' performance
Pandas	DataFrame	Read_csv, drop, head	Put dataset into data frame
Numpy	np	arange, array, concatenate, mean	Process numerical data
matplotlib	matplotlib.pyplt	grid, plot, title, xlabel, ylabel, subplots, legend, show	Plot the graph to visualize the data
Scikit-Image	Skimage.feature	hog	Detect face
OpenCV	cv2	imread, cvtColor, CascadeClassifier, rectangle, resize	Process the face images
Keras	keras.preprocessing import image	load_img, img_to_array	Convert images to array
Time	time	process_time	Calculate the Model Training Time, and CPU Execution Time

Model Architecture and Data Flow

Random Forest. The data in the proposed system have been categorized in accordance with the random forest classification model, which is shown in the following graphic. A random forest classifier is given a sample set consisting of n preprocessed samples to analyze. To build n unique trees for usage in RF, several subsets of the attributes are employed in the generation process. A classification result is produced by each tree, and the conclusion of the classification model is decided by taking the vote of the majority of the trees. The category that received the greatest number of votes overall is the one that receives the most samples (Ahmad et al., 2018, pp. 33789-33795). And the detailed architecture of the RF is shown in Figure 68.

Figure 68

The architecture and data flow example of RF

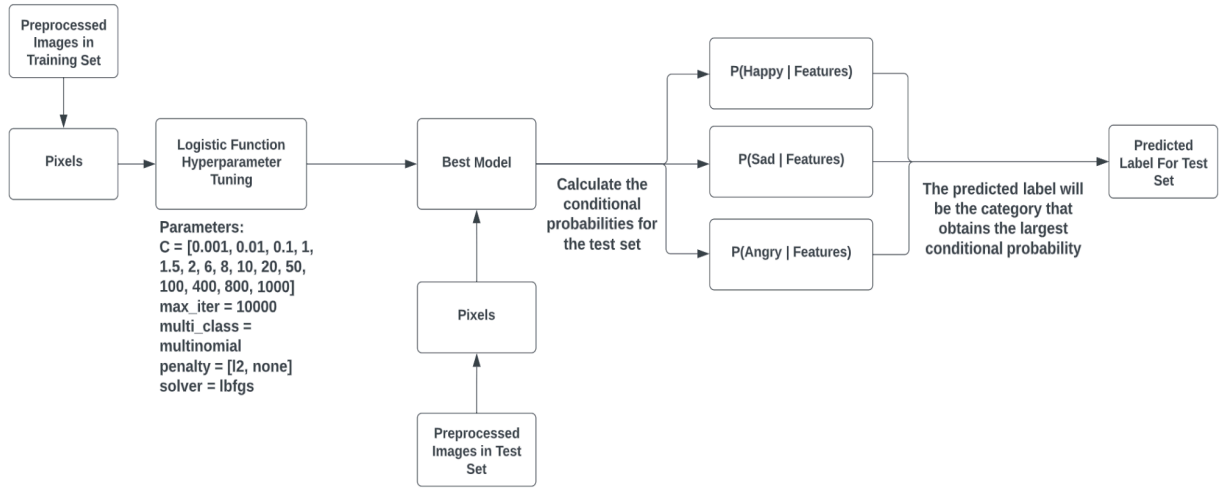


Note. Ahmad et al. (2018, pp. 33789-33795). *The architecture of the RF.* https://doi.org/10.1109/ACCESS.2018.2841987

Logistic Regression. As shown in Figure 69, there are six components (i.e., the inverse of regularization strength, solver, multiclass, penalty, max iterations, and class weight) to achieve the results. C represents the inverse of regularization strength, which is the measure of regularization. A larger C means a smaller penalty. The range of penalties was set to be 0.001, 0.01, 0.1, 1, 1.5, 2, 6, 8, 10, 20, 50, 100, 400, 800, and 1000. They represent large penalties, moderate penalties, and small penalties. For the solver, LBFGS was used. It can handle multinormal loss, which is suitable for multiclass classification. Solver LBFGS supports L2 penalty and no penalty. The value for the multiclass is multinomial because the model has to classify three kinds of emotions. Max iteration is set to 10000 because there is a higher chance of reaching the converge point to obtain a solution.

Figure 69

Model Architecture and Data Flow of Logistic Regression



Note. This is the model architecture and data flow of the Logistic Rgression algorithm in detail.

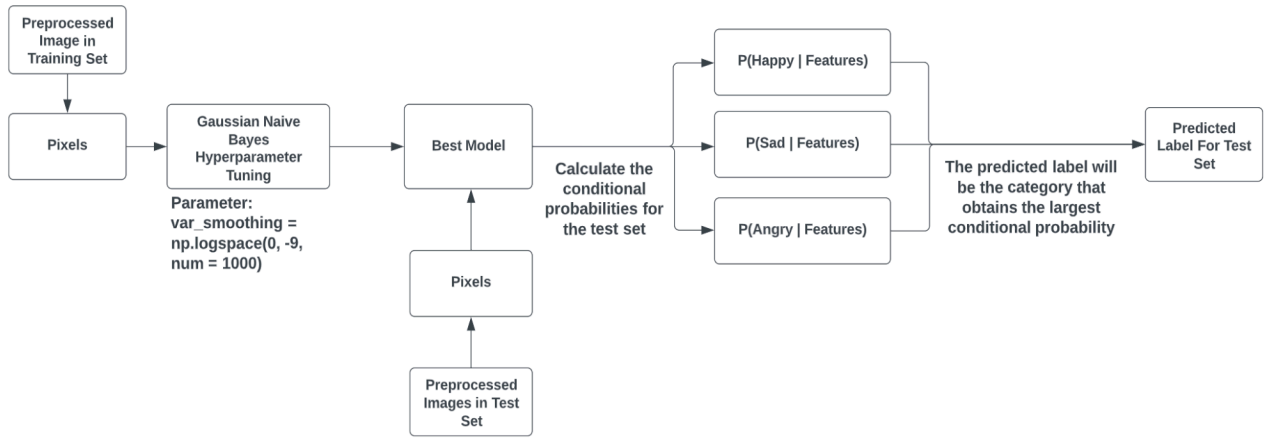
Figure 69 shows that the preprocessed images become pixels that pass in to train the logistic regression. There are two hyperparameters to tune: the penalty and inverse of

regularization strength. GridsearchCV along with Cross Validation is applied to tune the model to optimize the model performance. After figuring out the best model using the evaluation metric, it calculated the conditional probabilities for each category using the test set pixels. The label with the most considerable conditional probability is the predicted label for this corresponding test set image.

Gaussian Naive Bayes. As shown in Figure 70, there is only one hyperparameter to tune to optimize the model performance. The var_smoothing is a portion of the most significant variance of all characteristics contributing to the variances to ensure computation stability. The range of it was set to be between 1, and 1^{-9}. There are 1000 values in this range. The overall flow is that the preprocessed images will be converted into pixels. Then, Gridsearch and Cross-Validation are applied to tune the model. An evaluation metric is used to find the best model. After the best model is implemented, it calculates the conditional probabilities for each label for each test image. The final prediction is the label with the highest probability.

Figure 70

Model Architecture and Data Flow of Gaussian Naive Bayes

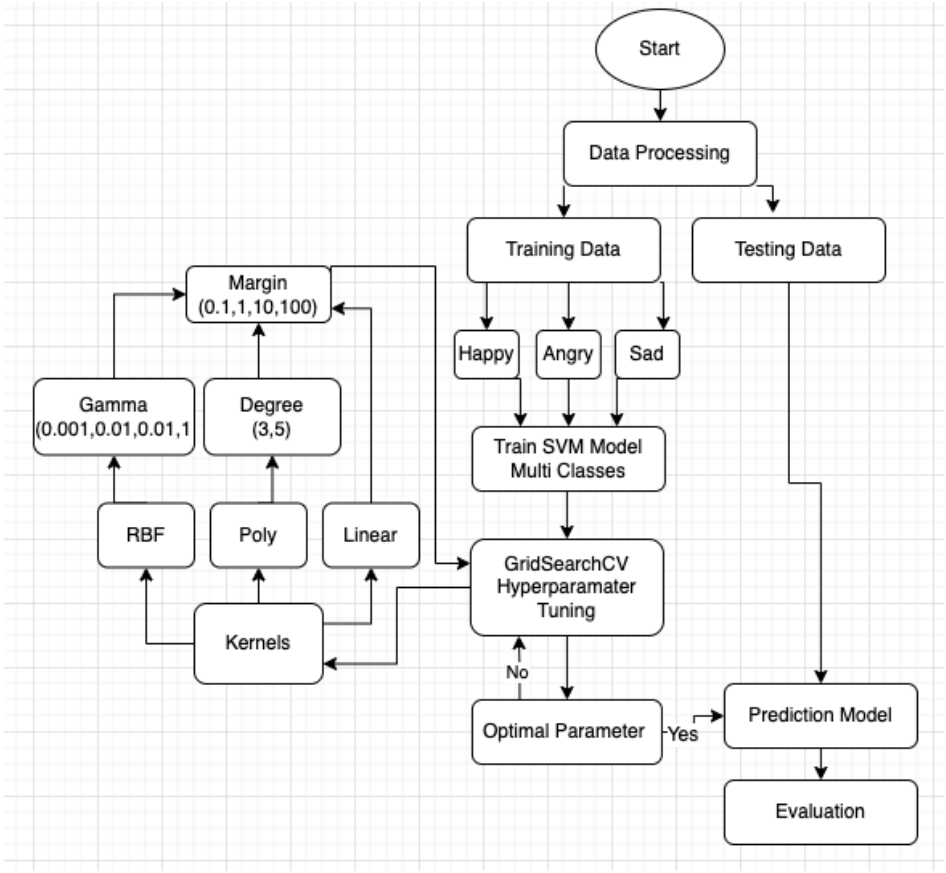


Note. This is the model architecture and data flow of Gaussian Naive Bayes algorithm in detail.

Support Vector Machine. Model architecture and data flow, shown in Figure 71, starts with data processing, including feature extraction, normalization, such as min-max normalization, and dimension reduction, such as PCA. After the data processing, the dataset is separated into two datasets which are 70 percent of the training set and 30 percent of the testing set. Next, train the SVM model with training data for three classes of data which are happy, angry, and sad images. GridSearchCV is the method of executing hyperparameter tuning in order to determine the optimal values for a given model. Additionally, GridSearchCV also executes the five flows cross-validation process. If the optimal parameter is encountered, testing data will be utilized to test the training data's optimal parameter and to evaluate the result based on the evaluation matrix with different evaluation methods such as accuracy, F1 score, Precision, Recall, ROC Curve, and AUC. If an optimal parameter cannot be located, GridSearchCV repeatedly processes different parameter combinations to fine-tune the model. The first parameter combination goes from the RBF kernel to four gamma values to four margin values, which is a total of sixteen combinations for the RBF kernel. The second parameter combination goes from the Polynomial kernel to two-degree values and four margin values, which is a total of eight combinations for the Polynomial kernel. The third combination of the parameter goes from the Linear kernel to four margin values, which is a total of four combinations for the Linear kernel. After running all the possible parameter combinations, the optimal parameter will be adjusted. Then the testing data will be used to test the training data's optimal parameter and to evaluate the result with different evaluation methods such as accuracy, F1 score, Precision, Recall, ROC Curve, and AUC.

Figure 71

Model Architecture and data Flow of SVM Model



Note. This is the model architecture and data flow of SVM in detail.

K-Nearest Neighbor. In machine learning applications, KNN is one of the practical and efficient algorithms for classification. As the instance-based estimator, KNN performs on the presumption that classifying unknown samples can be accomplished by associating the similarity of the novel to the known by the distance calculating. The closer the distance, the more proximal similarity of the query. The KNN architecture requires inputs of training samples and query samples. The KNN classifier calculates the distance between the input sample and all the instances in the training set. Many well-known distance metrics can be chosen: Minkowski distance, Chebyshev distance, Euclidean distance, Manhattan distance, Cosine distance, etc. (Chomboon et al., 2015, pp. 280-285). Then sort the distances in ascending order, and the sorting algorithms could be a bubble, select, quick, etc. (Yang et al., 2011, pp. 1314-1317). Finally, an output target is selected according to the class of the majority vote of minimum distance toward K neighbors. The number of the nearest neighbors should consider the trade-off between accuracy and sophistication. Thus, the 1-NN classifier is naive while assuming that the result is over-fitting.

Model Comparison and Justification

According to the current research, five models were compared and analyzed for justification including the RF (Random Forest), SVM (Support Vector Machine), Logistic Regression, Gaussian Naive Bayes, and K-Nearest Neighbor. Table 13 shows the similarity and differences in various categories.

Table 13

RF, SVM, LR, NB, and KNN Comparison

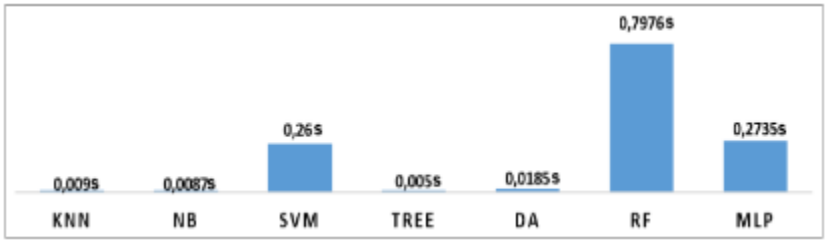
Characteristic	Random Forest	SVM	Logistic Regression	Gaussian Naive Bayes	KNN
Architecture	Ensemble	Linear	Linear	Probabilistic ML Model	Parallel
Data Type	Tabular	Tabular	Tabular	Tabular	Tabular
Do the models work better with small-size datasets or sparse data?	Bad with sparse data	Works well with sparse data and small datasets.	Large Data Size	Small/Large Data Size	Work well with small dataset
Known issues	Slow prediction makes it hard to do real-time processing	Perform poorly in large datasets and imbalanced datasets	Have Chances to Overfit or Underfit	Have Chances to Overfit or Underfit	Slow when data size is huge.
Preprocessing required	Requires data in the tabular format	Requires data in the vector format	Has to Transform the Images to Pixels	Has to Transform the Images to Pixels	Requires data in the tabular format
Training time	Fast	Slow	Moderate	Fast	Relatively short
Space complexity	Moderate	Large	Small	Small	Small
Computational complexity	Low	High	Moderate	Fast	Relatively low
Strengths	Able to resist noise in the dataset. Able to reduce the effect of overfitting. Handles large datasets.	Effective in high-dimensional spaces. Relatively memory efficient. Works well with sparse data. Effective when the number of dimensions is greater than the number of samples.	Works Well with Binary and Categorical Target Features. Easier to Implement and Interpret. Efficient to Train.	Runs Very Fast Compared to Other Complicated Models. Get the Baseline Accuracy Score.	Simplicity, effectiveness, and intuitiveness.

Characteristic	Random Forest	SVM	Logistic Regression	Gaussian Naive Bayes	KNN
Limitations	Big data updates quickly. The RF model might become outdated and less accurate in classifying data over time. There might be efficiency and accuracy problems because of the redundancy caused by oversized scaling.	Requires more time doing the GridSearchCV to tune the model. It suffers from imbalanced datasets. It suffers from large datasets.	Descriptive Features Have to Be Linearly Independent of Each Other. Not Accurate With Small Data Size.	Descriptive Features Have to Be Conditional Independent. May Encounter Zero Probability Problem.	Does not work well with high-dimensional and massive datasets.

Figure 72 shows a computational time comparison between seven models. The Random Forest consumed 0.7976 seconds, which is three times as SVM consumed. And KNN took 0.009 seconds, and the NB only consumed 0.0087 seconds. Therefore, the RF's computation time is much higher than the rest of the models, and NB is the fastest in Figure 72.

Figure 72

Execution time comparison between seven classifiers



Note. Ayache and Alti (2020, pp. 267-275). *Execution time comparison between seven classifiers.*

Model Evaluation Methods

Precision

The precision score is determined by combining the TP (true positives) with the FP (false positives). It refers to the proportion of emotions that were properly predicted out of the total number of emotions identified (Ayache & Alti, 2020, pp. 267-275). The formula may be written as in Figure 73.

Figure 73

Precision formula

$$Pre = \frac{TP}{TP + FP}$$

Note. Ayache and Alti (2020, pp. 267-275). *Precision formula.*

Recall

The recall score is derived by combining the percentage of True Positives (TP) and False Negatives (FN). It refers to the proportion of the total anticipated emotions that have been successfully predicted (Ayache & Alti, 2020, pp. 267-275). The formula may be expressed as follows in Figure 74.

Figure 74

Recall formula

$$Rec = \frac{TP}{TP + FN}$$

Note. Ayache and Alti (2020, pp. 267-275). *Recall formula.*

F1-Score

The F1-score takes into account both the accuracy and recall scores, making it a well-liked assessment of performance. The F1-score was given the highest significance in this research (Ayache & Alti, 2020, pp. 267-275). And the formula is shown in Figure 75.

Figure 75

F1-Score formula

$$F1s = 2 * \frac{Pre * Rec}{Pre + Rec}$$

Note. Ayache and Alti (2020, pp. 267-275). *F1-Score formula.*

Macro F1-Score

Since this project's target column has multi-class categorical variables. The metrics for multi-class classification are essential. For F1-Score, the macro F1-score is the appropriate approach in this scenario. As the F1-Score takes both precision and recall into consideration, the macro F1-Score takes both macro average precision and recall into consideration, and it also means the average of all the F1-Score from different classes (Grandini et al., 2020). Figure 76 below shows the calculation process for the macro average precision and recall, and Figure 77 shows the formula for macro F1-Score.

Figure 76

Macro average precision and recall formula

$$MacroAveragePrecision = \frac{\sum_{k=1}^{K} Precision_k}{K}$$

$$MacroAverageRecall = \frac{\sum_{k=1}^{K} Recall_k}{K}$$

Note. Grandini et al. (2020). *Macro average precision and recall formula.* https://doi.org/10.48550/arxiv.2008.05756

Figure 77

Macro F1-Score formula

$$Macro\ \text{F1-Score} = 2 * (\frac{MacroAveragePrecision * MacroAverageRecall}{MacroAveragePrecision^{-1} + MacroAverageRecall^{-1}})$$

Note. Grandini et al. (2020). *Macro F1-Score formula.* https://doi.org/10.48550/arxiv.2008.05756

ROC and AUC

To understand the Receiver Operating Characteristic curve, the background information about the TPR, FPR, TNR, and FNR is essential, the calculation formula is shown in Figure 78.

Figure 78

Formula for TPR, TNR, FPR, FNR

$$TPR = \frac{TP}{(TP + FN)}$$
$$TNR = \frac{TN}{(TN + FP)}$$
$$FPR = \frac{FP}{(TN + FP)}$$
$$FNR = \frac{FN}{(TP + FN)}$$

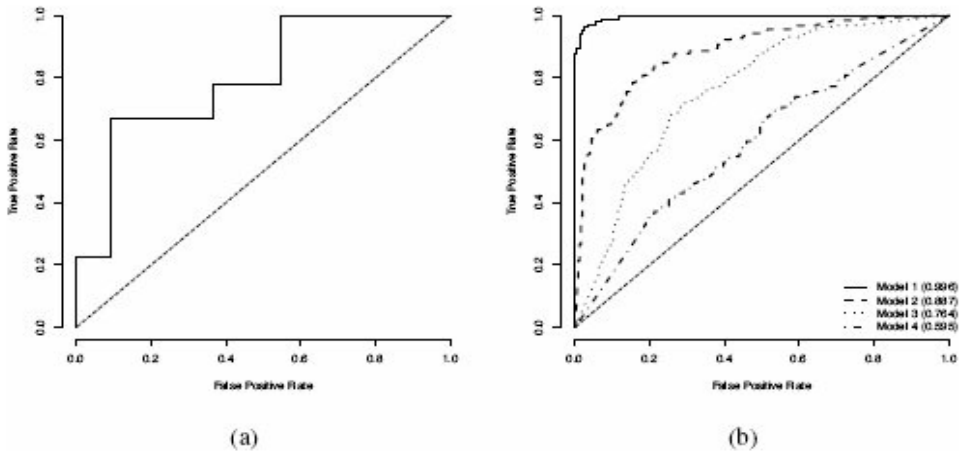
Note. Kelleher et al. (2015, p. 442). *Formula for TPR, TNR, FPR, FNR.*

The TP, FP, and FN were already covered in the previous sections, and the TN means the true negative. And with these measures, there are relations such that FNR = 1 − TPR, and FPR = 1 − TNR.

On a ROC plot, the TPR is shown on the y-axis and the FPR is displayed on the x-axis. Both TP and FP are variable based on how much the output threshold varies within its range (0 and 1) (Huang & Ling, 2005, pp. 299-310). An example of a ROC curve plot is shown in Figure 79.

Figure 79

ROC Curve Plot



Note. Kelleher et al. (2015, p. 455). *ROC Curve Plot.*

Graph(a) in Figure 79 is showing one ROC curve, and graph(b) in Figure 79 shows multiple ROC curves for multiple models comparison.

The dotted line from (0, 0) to (1, 0) is a reference line, which means the model does not help with predicting the results. Therefore the ROC curves for trained models should always be located above this reference line. As the predictive power of a model increases, the ROC curve grows to the top left corner point where TPR=1 and FPR=0. Thus, the ROC curve graphically represents the prediction ability of a model (Kelleher et al., 2015, p. 455).

Visually examining a model's performance using ROC curves may be helpful, but numerical performance measurements almost always provide more accurate results. The calculation of the ROC curve index or area under the curve (AUC) is useful for this purpose. And it is possible to determine the area of a ROC curve by integrating it. Due to the discrete and stepped structure of the trapezoidal methodology, integrals of ROC curves may be quickly calculated using the formula below in Figure 80 (Kelleher et al., 2015, p. 456).

Figure 80

The formula of the ROC index

$$ROC\ index = \sum_{i=2}^{|\mathbf{T}|} \frac{(FPR(\mathbf{T}[i]) - FPR(\mathbf{T}[i-1])) \times (TPR(\mathbf{T}[i]) + TPR(\mathbf{T}[i-1]))}{2}$$

Note. Kelleher et al. (2015, p. 456). *The formula of the ROC index.*

The T is the collection of thresholds, |T| is the number of thresholds evaluated, and TPR(T[i]) and FPR(T[i]) are based on the threshold i value. The ROC index's value range is [0, 1]. If it is less than 0.5, the model is misclassified in the other direction. If it equals 0.5, the model is not useful for prediction, which is essentially a random categorization outcome. When the value is more than 0.5, model performance is better when the value is bigger (Kelleher et al., 2015, p. 456).

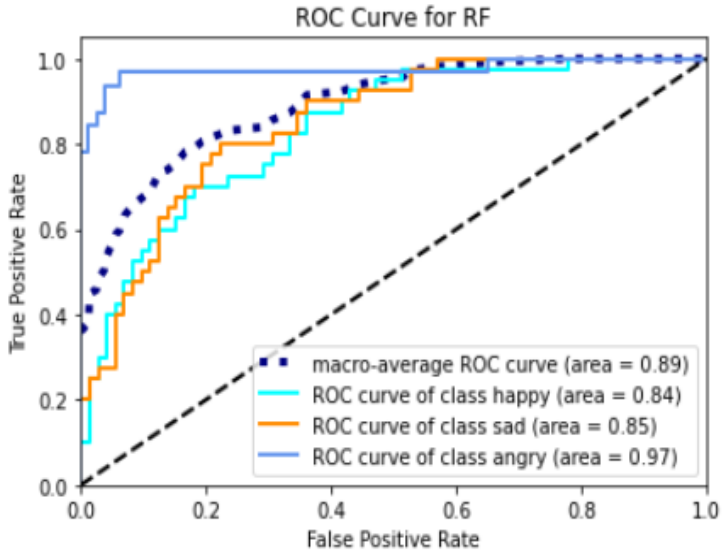
Model Validation and Evaluation

Random Forest

The random forest predictive model achieved 0.77 macro average precision, 0.76 macro average recall, and 0.76 macro average f1-score. See Appendix B Figure B10, Figure B11 and Figure B12 for detailed coding to implement and evaluate the Random Forest model. And the ROC curve is shown below in Figure 81.

Figure 81

ROC curve for Random Forest



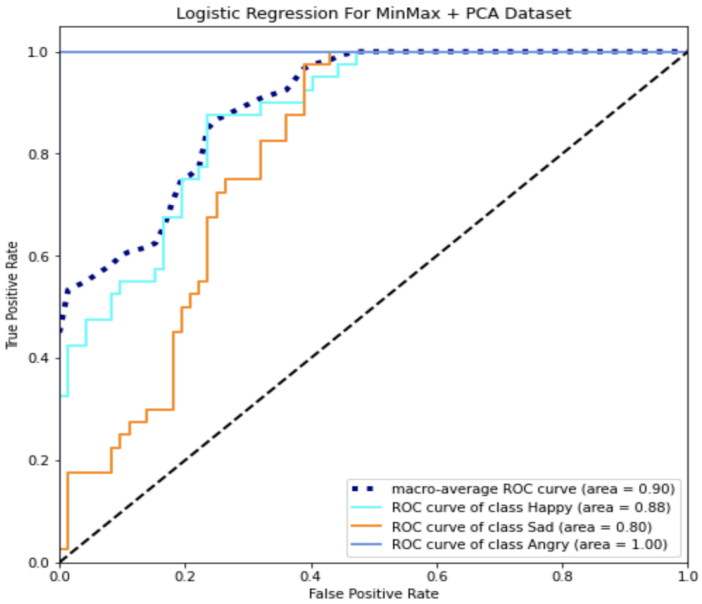
The light blue line is the ROC curve for happy, and the AUC is 0.84. The orange line is the ROC curve for sad, and the AUC is 0.85. The blue line is the ROC curve for angry, and the AUC is 0.97. And the dark blue dotted line is the macro average ROC curve for the RF model, and the macro average AUC is 0.89.

Logistic Regression

The accuracy and Macro-F1 score are both 0.79. The Macro-Precision and Macro-Recall are both 0.8. These four metrics are relatively high, showing that the model can distinguish different emotions from most images. As shown in Figure 82, the AUC for class happy is 0.88. The AUC for sad images is 0.80, and the AUC for angry is 1. Also, the Macro-AUC is 0.9. It means that the availability that the model predicts angry images is more robust than happy and sad images. Since the AUC for the angry class is 1, the Logistic Regression can detect all the angry images. See Appendix B Figure B3, and Figure B4 for detailed coding to implement and evaluate the Logistic Regression model.

Figure 82

ROC Curve and AUC of Logistic Regression for MinMax & PCA Dataset:

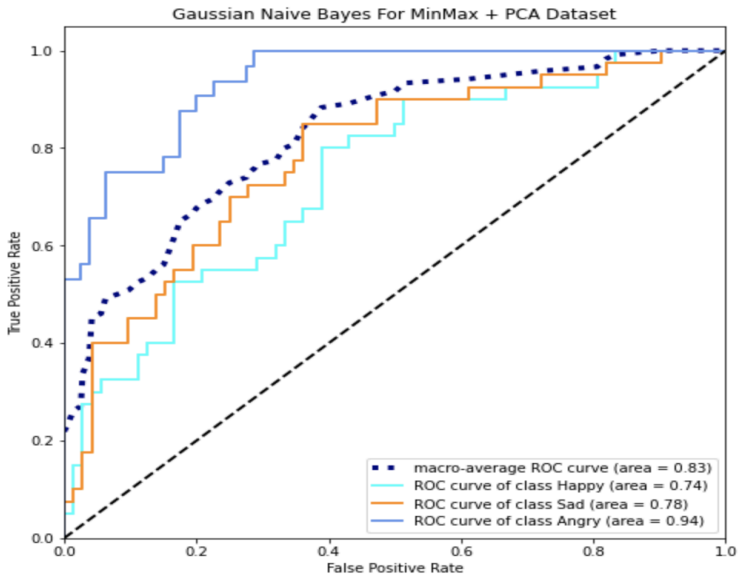


Gaussian Naive Bayes

The Gaussian Naive Bayesian was also implemented using two different datasets. The accuracy, Macro-Precision, Macro-Recall, and Macro-F1 are 0.59, 0.65, 0.59, and 0.6. From Figure 83, the AUCs for the happy, sad, and angry classes are 0.74, 0.78, and 0.94. The Macro-AUC is 0.83. Compared with the accuracy, Macro-Precision, Macro-Recall, and Macro-F1, the AUCs are relatively high. See Appendix B Figure B5, and Figure B6 for detailed coding to implement and evaluate the logistic regression model.

Figure 83

ROC Curve and AUC of Gaussian Naive Bayes for MinMax & PCA Dataset

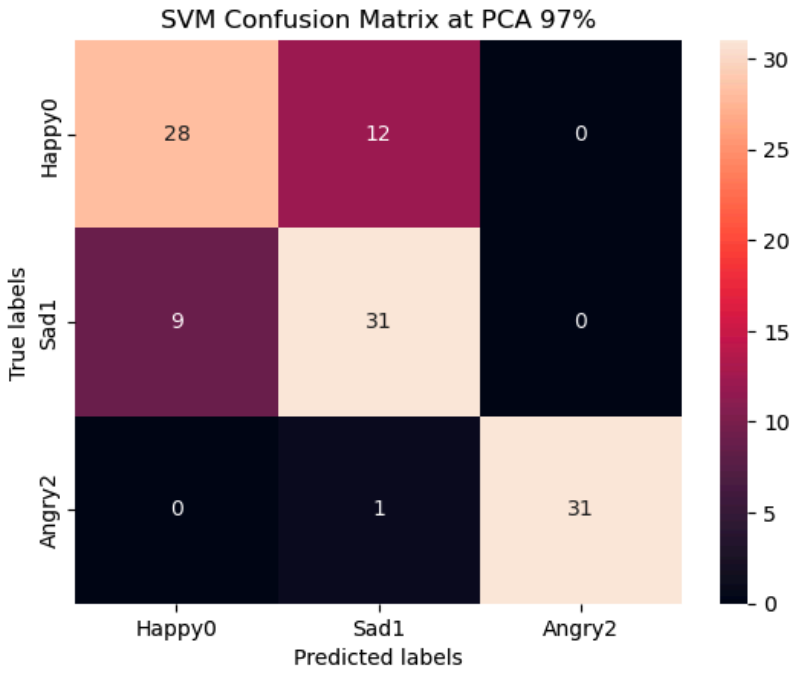


Support Vector Machine

The confusion matrix is used for evaluating a classification model's performance. The SVM Confusion Matrix at PCA 97% in Figure 84 visualizes and summarizes the performance of the SVM model. There are 28 records of the Happy class identified correctly in the model, which is the true positive outcome from the total of 40 records of the Happy class. There are 31 records of the Sad class identified correctly in the model, which is the true positive outcome from the total of 40 records of the Sad class. There are 31 records of the Angry class identified correctly in the model, which is the true positive outcome from the total of 40 records of the Angry class. See Appendix B, Figure B7 for detailed coding to implement SVM and the confusion matrix.

Figure 84

SVM Confusion Matrix



Note. Confusion matrix for SVM model.

One of the performance evaluation metrics of a classification mode in machine learning is the classification report; see the following SVM classification report in Figure 85. The report shows the model's Precision, Recall, F1 score, and support. The Marco-Average-Precision is 82%, the Marco-Average-Recall is 81%, and the Macro-Average-F1-Score is 82%. The accuracy of the F1 score is 80%. See Appendix B, Figure B8 for detailed coding to implement the classification report for SVM.

Figure 85

SVM Classification Report

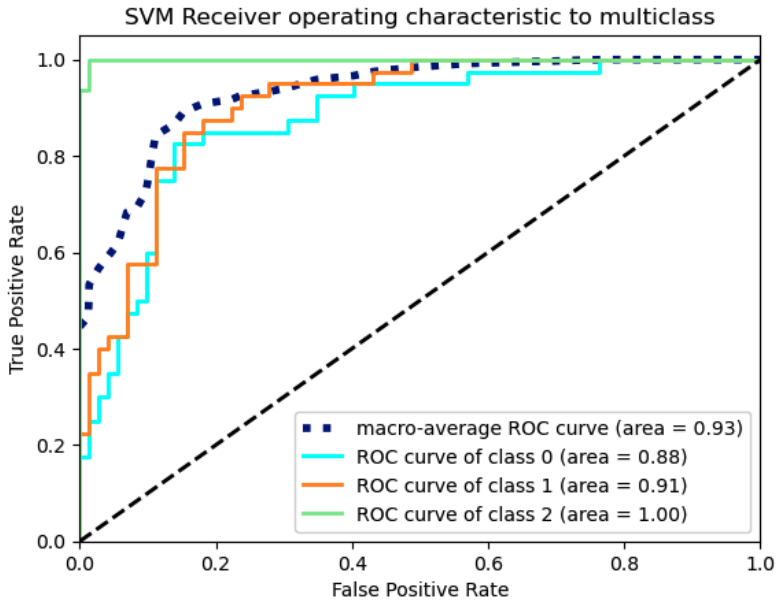
	precision	recall	f1-score	support
happy	0.76	0.70	0.73	40
sad	0.70	0.78	0.74	40
angry	1.00	0.97	0.98	32
accuracy			0.80	112
macro avg	0.82	0.81	0.82	112
weighted avg	0.81	0.80	0.80	112

Note. Marco average and accuracy are used for model evaluation.

The ROC Curve and the AUC score are crucial tools to evaluate the classification model and show the class separability by all possible thresholds and how well the model is classifying each class. In Figure 86, the AUC for class 2, which is Angry, is higher than the class 1 curve, which is Sad, and the class 0 curve, which is Happy. The AUC for class Happy is 0.88, the AUC for class Sad is 0.91, and the AUC for class Angry is 1. In addition, the Marco-Average AUC is 0.93, which means the AUC for the Angry class performs better on the job of classifying the positive class in the dataset. See Appendix B, Figure B9 for detailed coding to implement ROC-AUC for SVM.

Figure 86

SVM ROC-AUC Graph



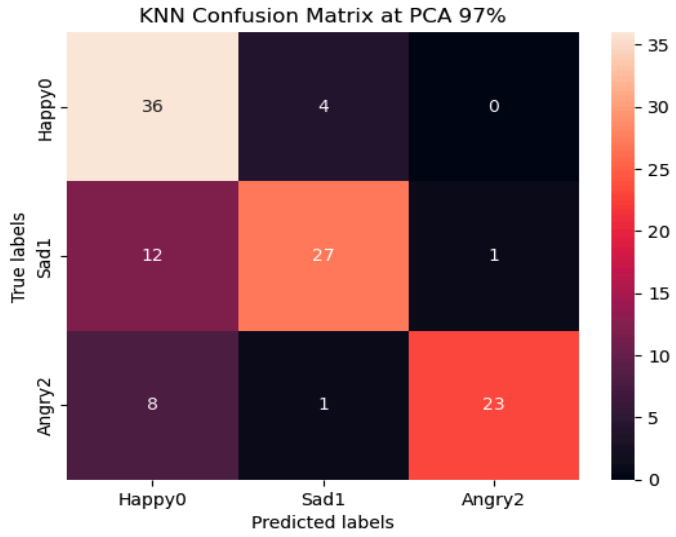
Note. Class 0 is happy, class 1 is sad, and class 2 is angry.

K Nearest Neighbors

K Nearest Neighbors (KNN) Confusion Matrix at PCA 97% in Figure 87 visualizes and summarizes the KNN model's performance. There are 36 records of the Happy class identified correctly in the model, which is the true positive outcome from the total of 40 records of the Happy class. There are 27 records of the Sad class identified correctly in the model, which is the true positive outcome from the total of 40 records of the Sad class. Finally, 23 records of the Angry class were identified correctly in the model, which is the true positive outcome from the total of 40 records of the Angry class. The confusion matrix is shown in Figure 87. See Appendix B Figure B13, and Figure B14 for detailed coding to implement and evaluate the KNN model.

Figure 87

KNN Confusion Matrix



Note. Confusion matrix for KNN model.

The KNN classification report in Figure 88 shows the Marco-Average-Precision is 81%, the Marco-Average-Recall is 76%, and the Macro-Average-F1-Score is 77%. The accuracy of the F1-score is 77%.

Figure 88

KNN Classification Report

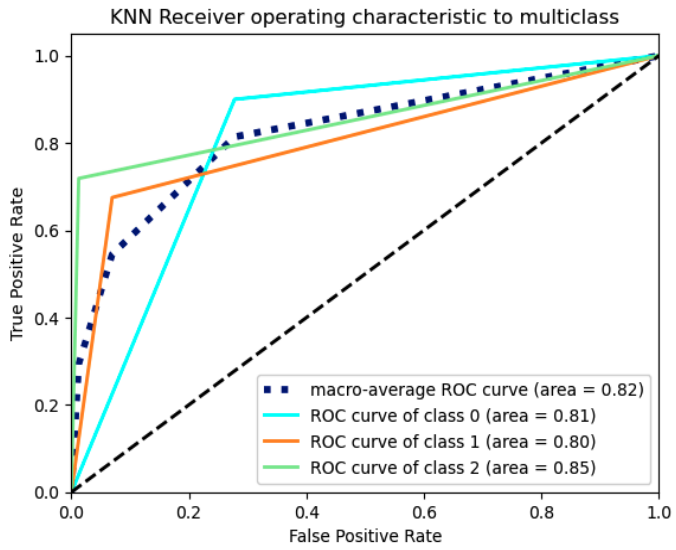
	precision	recall	f1-score	support
happy	0.64	0.90	0.75	40
sad	0.84	0.68	0.75	40
angry	0.96	0.72	0.82	32
accuracy			0.77	112
macro avg	0.81	0.76	0.77	112
weighted avg	0.80	0.77	0.77	112

Note. Marco average and accuracy are used for model evaluation.

In Figure 89, the AUC for class 2, Angry, is higher than the class 1 curve, Sad, and the class 0 curve, Happy. The AUC for class Happy is 0.81, the AUC for class Sad is 0.80, and the AUC for class Angry is 0.85. In addition, the Marco-Average AUC is 0.82, which means the AUC for the Angry class performs better on the job of classifying the positive class in the dataset. See Appendix B, Figure B15 for detailed coding to implement ROC-AUC for KNN.

Figure 89

KNN ROC-AUC Graph



Note. Class 0 is happy, class 1 is sad, and class 2 is angry.

The comparison of the performance of five models using the mentioned evaluation metrics is shown in Table 14.

Table 14

Evaluation Methods and Scores

Model	Accuracy	Marco F1-Score	Marco Precision	Marco Recall	Marco-AUC
SVM	80%	82%	82%	81%	93%
Logistic Regression	79%	79%	80%	80%	90%
Gaussian Naive Bayes	59%	60%	65%	59%	83%
Random Forest	75%	76%	77%	76%	89%
SVM	82%	83%	83%	83%	93%

Note. An overview of evaluation methods and scores and SVM perform the best.

Conclusion

This paper pursues to establish a precise and functional model to process facial feature extraction from human face photos and ultimately accurately detect facial emotions. The performance of SVM, Logistic, Gaussian Naive Bayes, SVM, and KNN is assessed with different evaluation methods such as accuracy, F1 score, Precision, Recall, ROC Curve, and AUC. With the trained SVM classifier, the outcome showed that the classification performance of the SVM in predicting facial expressions is better than other models for the time being.[1]

Generally, the final model's performance is acceptable but may be better. Although it can locate every angry image, it sometimes mispredicts happy and sad images.

Based on the previous research and the implementation of this project, in the future, the direction of the study will be applying more advanced feature selection methods and

[1] GitHub Link for the project code. https://github.com/cheukhongip/Group5EmotionDetection

implementing deep learning models to solve the problem. During the data collection and cleaning stage, the data is manually filtered by the team members. Therefore, an automatic filtration method is essential for processing large datasets in the future.

Because only machine learning models were applied to identify facial expressions, future objectives include the development of more deep learning models (such as Deep Neural Networks, Convolutional Neural Networks, Artificial Neural Networks, etc.) to assess their performance.

Additionally, it is unable to handle real-time and video analysis. The existing machine learning models will be improved, and new deep learning models will be implemented to conduct real-time emotion recognition.

Lastly, the dataset with only the facial images is insufficient to accurately predict the true emotion in the future, and images will be integrated with additional features (such as the electrocardiogram, body temperature variance, pulse rate, and so on) to enhance the models' performance.

References

Ahmad, Basheri, M., Iqbal, M. J., & Rahim, A. (2018). Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection. *IEEE Access, 6*, 33789–33795. https://doi.org/10.1109/ACCESS.2018.2841987

Ayache, F., & Alti, A. (2020, May 7). Performance Evaluation of Machine Learning for Recognizing Human Facial Emotions. *Rev. d'Intelligence Artif.*, *34*(3), 267-275. https://doi.org/10.18280/ria.340304

Barghout, L. (2015). Spatial-Taxon Information Granules as Used in Iterative Fuzzy-Decision-Making for Image Segmentation. In *Granular Computing and Decision-Making*. *Studies in Big Data*. Vol. 10. pp. 285–318. doi:10.1007/978-3-319-16829-6_12. ISBN 978-3-319-16828-9. S2CID 4154772.

Barman, & Dutta, P. (2021). Facial expression recognition using distance and shape signature features. *Pattern Recognition Letters*, 145, 254–261. https://doi.org/10.1016/j.patrec.2017.06.018.

Bartlett, Littlewort, G., Fasel, I., & Movellan, J. R. (2003). Real Time Face Detection and Facial Expression Recognition: Development and Applications to Human Computer Interaction. *2003 Conference on Computer Vision and Pattern Recognition Workshop*, *5*, 53–53. https://doi.org/10.1109/CVPRW.2003.10057

Bhavitha, B. K., Rodrigues, A. P., & Chiplunkar, N. N. (2017). Comparative study of machine learning techniques in sentimental analysis. Paper presented at the - *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT),* 216-221. doi:10.1109/ICICCT.2017.7975191

Chuang,C.F., & Shih, F. Y. (2006). Recognizing facial action units using independent component analysis and support vector machine. *Pattern Recognition,* 39(9), 1795–1798. https://doi.org/10.1016/j.patcog.2006.03.017

Dahmane, M., & Meunier, J. (2011, March). Emotion recognition using dynamic grid-based HoG features. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)* (pp. 884-888). IEEE.https://doi.org/10.1109/FG.2011.5771368

Dalal, & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, VOL 1, PROCEEDINGS*, *I*, 886–893. https://doi.org/10.1109/CVPR.2005.177

Dimitriadis, & Liparas, D. (2018). How random is the random forest? Random forest algorithm on the service of structural imaging biomarkers for Alzheimer's disease: from Alzheimer's disease neuroimaging initiative (ADNI) database. *Neural Regeneration Research, 13(6)*, 962–970. https://doi.org/10.4103/1673-5374.233433 doi:10.1109/KICSS.2016.7951418

Google APIs Explorer. (n.d.). Retrieved September 28, 2022, from https://developers.google.com/apis-explorer/?hl=SR&skip_cache=true

Grandini, Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: an Overview*. https://doi.org/10.48550/arxiv.2008.05756

Guo, Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2888*, 986–996. https://doi.org/10.1007/978-3-540-39964-3_62

Gupta. (2018). Facial emotion recognition in real-time and static images. *Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018*, 553–560. https://doi.org/10.1109/ICISC.2018.8398861

Hall, Park, B. U., & Samworth, R. J. (2008). Choice of Neighbor Order in Nearest-Neighbor Classification. *The Annals of Statistics*, *36*(5), 2135–2152. https://doi.org/10.1214/07-AOS537

Hsieh, C. C., Hsih, M. H., Jiang, M. K., Cheng, Y. M., & Liang, E. H. (2016). Effective semantic features for facial expressions recognition using SVM. *Multimedia tools and applications. 75*(11), 6663-6682. https://doi.org/10.1007/s11042-015-2598-1

Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering,* 17(3), 299-310.

Kaya. (2013). A hybrid model for classification of remote sensing images with linear SVM and support vector selection and adaptation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *6*(4), 1988–1997. https://doi.org/10.1109/JSTARS.2012.2233463

Kazemi, & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. *2014 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)*, 1867–1874. https://doi.org/10.1109/CVPR.2014.241

Kelleher, John D., Mac Namee, Brian, D'Arcy, Aoife (2015). *Fundamentals of Machine Learning For Predictive Data Analytics, Algorithms, Worked Examples, and Case Studies*. Cambridge: The MIT Press

Kotsia, Buciu, I., & Pitas, I. (2008). An analysis of facial expression recognition under partial facial image occlusion. *Image and Vision Computing*, *26*(7), 1052–1067. https://doi.org/10.1016/j.imavis.2007.11.004

Kremic, E., & Subasi, A. (2016). Performance of random forest and SVM in face recognition. *Int. Arab J. Inf. Technol.*, *13*(2), 287-293.

Kuruvila, A. P., Kundu, S., & Basu, K. (2020, July). Analyzing the efficiency of machine learning classifiers in hardware-based malware detectors. In *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (pp. 452-457). IEEE.

Lahaw, Essaidani, D., & Seddik, H. (2018). Robust Face Recognition Approaches Using PCA, ICA, LDA Based on DWT, and SVM Algorithms. *2018 41st International Conference on Telecommunications and Signal Processing, TSP 2018*. https://doi.org/10.1109/TSP.2018.8441452

Le Ngo, Yee-Hui Oh, Phan, R. C.-W., & See, J. (2016). Eulerian emotion magnification for subtle expression recognition. *2016 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING PROCEEDINGS*, *2016-*, 1243–1247. https://doi.org/10.1109/ICASSP.2016.7471875

Lipovetsky, S. (2009). PCA and SVD with nonnegative loadings. Pattern Recognition, 42(1), 68-76.https://doi.org/10.1016/j.patcog.2008.06.025.

Lyons, Budynek, J., Plante, A., & Akamatsu, S. (2000). Classifying facial attributes using a 2-D Gabor wavelet representation and discriminant analysis. *Proceedings - 4th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2000*, 202–207. https://doi.org/10.1109/AFGR.2000.840635

Ma, Xiaoxi, Weisi, L., Dongyan, H., Minghui, D., & Li, H. (2017). Facial emotion recognition. *2017 IEEE 2nd International Conference on Signal and Image Processing, ICSIP 2017*, *2017-*, 77–81. https://doi.org/10.1109/SIPROCESS.2017.8124509

Mahmud, Khatun, M. T., Zuhori, S. T., Afroge, S., Aktar, M., & Pal, B. (2015). Face recognition using Principle Component Analysis and Linear Discriminant Analysis. *2nd International Conference on Electrical Engineering and Information and Communication Technology, iCEEiCT 2015*. https://doi.org/10.1109/ICEEICT.2015.7307518

Michel, & El Kaliouby, R. (2003). Real time facial expression recognition in video using support vector machines. *International Conference on Multimodal Interfaces: Proceedings of the 5th International Conference on Multimodal Interfaces; 05-07 Nov. 2003*, 258–264. https://doi.org/10.1145/958432.958479

Mostafa, Khalil, M. I., & Abbas, H. (2019). Emotion Recognition by Facial Features using Recurrent Neural Networks. *Proceedings - 2018 13th International Conference on Computer Engineering and Systems, ICCES 2018*, 417–422. https://doi.org/10.1109/ICCES.2018.8639182

Olatomiwa, Mekhilef, S., Shamshirband, S., Mohammadi, K., Petković, D., & Sudheer, C. (2015). A support vector machine–firefly algorithm-based model for global solar radiation prediction. *Solar Energy, 115,* 632–644. https://doi.org/10.1016/j.solener.2015.03.015

Putranto, E. B., Situmorang, P. A., & Girsang, A. S. (2016). Face recognition using eigenface with naive bayes. In *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS),* 1-4.

Rigatti, S. J. (2017). Random forest. *Journal of Insurance Medicine, 47*(1), 31-39.

Sabri, Henry, J., Ibrahim, Z., Ghazali, N., Abu Mangshor, N. N., Mohd Johari, N. F., & Ibrahim, S. (2019). A comparison of face detection classifier using facial geometry distance measure. *2018 9th IEEE Control and System Graduate Research Colloquium, ICSGRC 2018 - Proceeding*, 116–120. https://doi.org/10.1109/ICSGRC.2018.8657592

Salehi, Abbasi, E., & Hassibi, B. (2019). The Impact of Regularization on High-dimensional Logistic Regression. arXiv.org.

Saragih, Lucey, S., & Cohn, J. F. (2011). Real-time avatar animation from a single image. *2011 IEEE International Conference on Automatic Face and Gesture Recognition and Workshops, FG 2011*, 213–220. https://doi.org/10.1109/FG.2011.5771400

Scornet. (2017). Tuning parameters in random forests. *ESAIM. Proceedings and Surveys, 60*, 144–162. https://doi.org/10.1051/proc/201760144

Shamir, O. (2015, June). A stochastic PCA and SVD algorithm with an exponential convergence rate. In International conference on machine learning (pp. 144-152). PMLR.

Sharma, Bhatt, M., & Sharma, P. (2020). Face recognition system using machine learning algorithm. *Proceedings of the 5th International Conference on Communication and Electronics Systems, ICCES 2020*, 1162–1168. https://doi.org/10.1109/ICCES48766.2020.09137850

Stoltzfus, J. C. (2011). Logistic regression: a brief primer. *Academic emergency medicine*, *18*(10), 1099-1104.

Swinkels, Claesen, L., Xiao, F., & Shen, H. (2017). SVM point-based real-time emotion detection. *2017 IEEE Conference on Dependable and Secure Computing*, 86–92. https://doi.org/10.1109/DESEC.2017.8073838

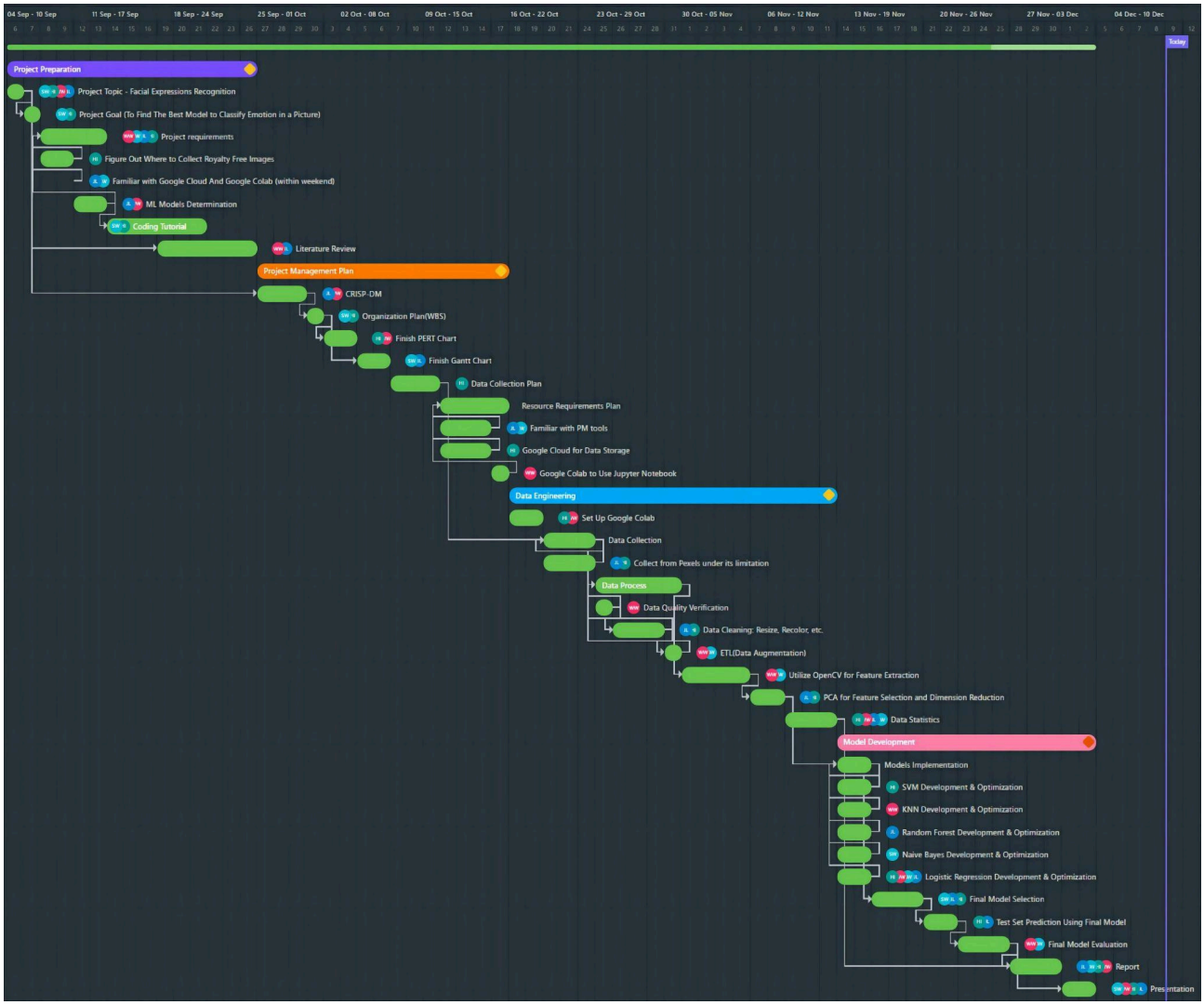
Wang, Xia, S.-T., Tang, Q., Wu, J., & Zhu, X. (2018). A Novel Consistent Random Forest Framework: Bernoulli Random Forests. *IEEE Transaction on Neural Networks and Learning Systems*, 29(8), 3510–3523. https://doi.org/10.1109/TNNLS.2017.2729778

Yao, L., Wan, Y., Ni, H., & Xu, B. (2021). Action unit classification for facial expression recognition using active learning and SVM. *Multimed Tools Appl 80*, 24287–24301. https://doi.org/10.1007/s11042-021-10836-w

Zhang, Yin, Z., Chen, P., & Nichele, S. (2020). Emotion recognition using multi-modal data and machine learning techniques: A tutorial and review. *Information Fusion*, *59*, 103–126. https://doi.org/10.1016/j.inffus.2020.01.011

Appendix A

Gantt Chart



Appendix A. This is the overview of the Gantt chart.

Appendix B

Sample Coding

Testing Cascade Face Detector And Save image in testimage Folder

```
a=[]
b=[]
#266004
#774866
#1happy2167673
imgs = np.array(cv.imread('imgs/Test/happy/1happy266004.jpg'))
# converting BGR to RGB
img_rgb = cv.cvtColor(imgs, cv.COLOR_BGR2RGB)
mountCascade = cv.CascadeClassifier('haarcascade_mcs_mouth.xml')
faceCascade = cv.CascadeClassifier('haarcascade_frontalface_alt.xml')
gray = cv.cvtColor(img_rgb, cv.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(img_rgb,scaleFactor = 1.05, minNeighbors = 5, minSize = (50,50))
for (x, y, w, h) in faces:
    cv.rectangle(img_rgb, (x, y), (x+w, y+h), (255, 0 , 0), 25)
    faces = img_rgb[y:y + h, x:x + w]
faces_gray  = cv.cvtColor(faces, cv.COLOR_BGR2GRAY)
resize_img = cv.resize(faces_gray,(64,64))
pixel = resize_img.flatten()
a.append(resize_img)
print(resize_img.shape)
print(a[0])
plt.imshow(img_rgb)
plt.show()
plt.imshow(faces_gray,cmap="gray")
plt.show()
plt.imshow(faces)
plt.show()
plt.imshow(resize_img,cmap="gray")
plt.show()

cv.imwrite('./testimage/' + 'baby.jpg',cv.cvtColor(faces, cv.COLOR_RGB2BGR))
```

Figure B1. This shows how to extract features from images.

```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
# MinMaxScaler for GrayScale X tain test
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

pca = PCA(n_components=0.97)
x_train_minmax_pca_97 = pca.fit_transform(X_train_scaled)
x_test_minmax_pca_97 = pca.transform(X_test_scaled)

x_train_minmax_pca_97 = pd.DataFrame(x_train_minmax_pca_97)
x_train_minmax_pca_97['class'] = y_train
x_test_minmax_pca_97 = pd.DataFrame(x_test_minmax_pca_97)
x_test_minmax_pca_97['class'] = y_test

# StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train_sc = sc.transform(X_train)
X_test_sc= sc.transform(X_test)

pca = PCA(n_components=0.97)
x_train_sc_pca_97 = pca.fit_transform(X_train_sc)
x_test_sc_pca_97 = pca.transform(X_test_sc)

x_train_sc_pca_97 = pd.DataFrame(x_train_sc_pca_97)
x_train_sc_pca_97['class'] = y_train
x_test_sc_pca_97 = pd.DataFrame(x_test_sc_pca_97)
x_test_sc_pca_97['class'] = y_test
```

Figure B2. This shows how to normalize the pixels and reduce dimensions.

Logistic Regression

```python
# class_weight='balanced', The "balanced" mode uses the values of y to automatically adjust weights inversely
# proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)).
# multi_class='multinomial'
# For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss
# C: Inverse of regularization strength; must be a positive float.
# larger c, smaller penality

grid_values = {'C': [0.001, 0.01, 0.1, 1, 1.5, 2, 6, 8, 10, 20, 50, 100, 400, 800, 1000],
               'solver': ['lbfgs'],
              'multi_class': ['multinomial'],
              'penalty': ['l2'],
               'max_iter': [10000],
              'class_weight': ['balanced']}
log = LogisticRegression()
log_minmax = GridSearchCV(log, param_grid = grid_values, scoring = 'f1_macro', cv=5, verbose=1)
log_minmax.fit(X_train, y_train)
```

Figure B3. This shows how to implement the Logistics Regression model.

```
log_minmax_pred = log_minmax.best_estimator_.predict(X_test)
cm = confusion_matrix(y_test, log_minmax_pred)

print ("Confusion Matrix For MinMax + Logistic Regression: \n", cm)
print(classification_report(y_test, log_minmax_pred, target_names=['happy', 'sad', 'angry']))
```

```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from itertools import cycle
```

```
# Binarize the output
y_train1 = label_binarize(y_train, classes=[0, 1, 2])
y_test = label_binarize(y_test, classes=[0, 1, 2])
n_classes = 3
y_score = log_minmax.best_estimator_.decision_function(X_test)
# Compute ROC curve and ROC area for each class
lw = 2
n_class = 3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_class):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure(figsize=(8, 8))

plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (area = {0:0.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)

colors = cycle(["aqua", "darkorange", "cornflowerblue"])
classes = ['Happy', 'Sad', 'Angry']
for i, color in zip(range(n_classes), colors):
    plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=lw,
        label="ROC curve of class {0} (area = {1:0.2f})".format(classes[i], roc_auc[i]),
    )

plt.plot([0, 1], [0, 1], "k--", lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistic Regression For MinMax + PCA Dataset")
plt.legend(loc="lower right")
plt.show()
```

Figure B4. This shows how to evaluate Logistics Regression performance.

```
# var_smoothing is a stability calculation to widen (or smooth) the curve and therefore account
# for more samples that are further away from the distribution mean. In this case, np.logspace returns numbers
# spaced evenly on a log scale, starts from 0, ends at -9, and generates 100 samples.
param_grid_nb = {
    'var_smoothing': np.logspace(0,-9, num =1000)
}

nb_minmax = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1, cv=5, scoring = 'f1_macro')
nb_minmax.fit(X_train, y_train)

result_nb_minmax = pd.DataFrame(nb_minmax.cv_results_)
result_nb_minmax.sort_values(by=['mean_test_score'],ascending=False)
result_nb_minmax[['params', 'mean_test_score']].head(5)
```

Figure B5. This shows how to implement the Naive Bayes model.

```
nb_minmax_pred = nb_minmax.best_estimator_.predict(X_test)
cm = confusion_matrix(y_test, nb_minmax_pred)

print ("Confusion Matrix For MinMax + Gaussian NB: \n", cm)
print(classification_report(y_test, nb_minmax_pred, target_names=['happy', 'sad', 'angry']))
```

```
# Binarize the output
n_classes = 3
y_score = nb_minmax.best_estimator_.predict_proba(X_test)
# Compute ROC curve and ROC area for each class
lw = 2
n_class = 3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_class):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure(figsize=(8, 8))

plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (area = {0:0.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)

colors = cycle(["aqua", "darkorange", "cornflowerblue"])
classes = ['Happy', 'Sad', 'Angry']
for i, color in zip(range(n_classes), colors):
    plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=lw,
        label="ROC curve of class {0} (area = {1:0.2f})".format(classes[i], roc_auc[i]),
    )

plt.plot([0, 1], [0, 1], "k--", lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Gaussian Naive Bayes For MinMax + PCA Dataset")
plt.legend(loc="lower right")
plt.show()
```

Figure B6. This shows how to evaluate the Naive Bayes model.

```
result = pd.DataFrame(grid.cv_results_)
result = result.sort_values(by=['mean_test_score'],ascending=False)
result.to_csv('SVM97.csv')
```

```
model = SVC(C= 10, gamma= 0.001, kernel= 'rbf')
model.fit(X_train_97, y_train_97.values.ravel())
y_pred = model.predict(X_test_97)
score = accuracy_score(y_pred,y_test_97)
print("Accuracy:", score)
cm = confusion_matrix(y_test_97.values.ravel(), y_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('SVM Confusion Matrix at PCA 97%')
ax.xaxis.set_ticklabels(['Happy0', 'Sad1' , 'Angry2']); ax.yaxis.set_ticklabels(['Happy0', 'Sad1' , 'Angry2'])
```

```
Accuracy: 0.8035714285714286

[Text(0, 0.5, 'Happy0'), Text(0, 1.5, 'Sad1'), Text(0, 2.5, 'Angry2')]
```

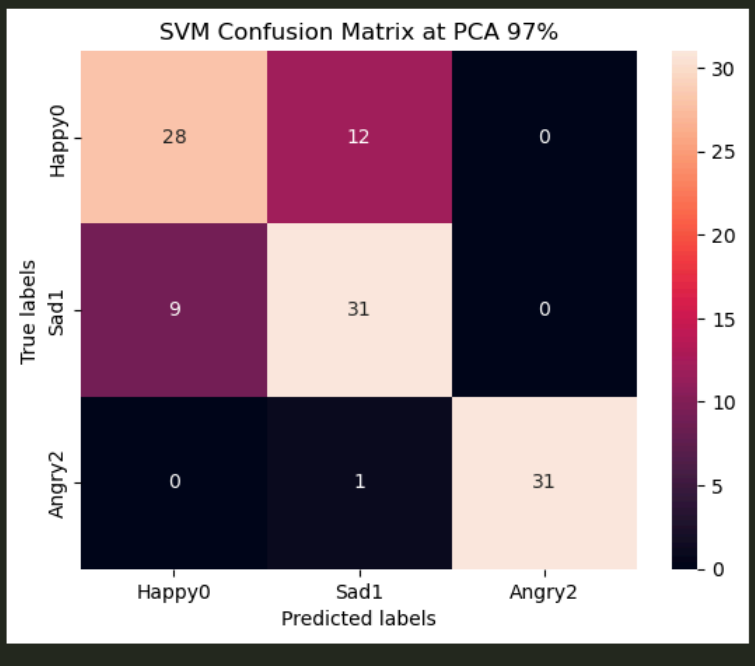


Figure B7. This shows how to implement SVM and evaluate the model performance with confusion matrix for SVM.

SVM

SVM PCA 97%

```
svm_clf = SVC()
param_grid = { 'kernel': ['rbf','poly'], 'gamma': [1,0.1,0.01,0.001], 'degree': [3,5],'C':[0.1,1,10,100]}
grid = GridSearchCV(svm_clf, param_grid, cv =5, scoring = 'accuracy', refit=True, verbose = 1, return_train_score = True)
grid.fit(X_train_97, y_train_97.values.ravel())
print(grid.best_score_)
```

```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
0.8598487993981575
```

```
print(grid.best_params_)
```

```
{'C': 10, 'degree': 3, 'gamma': 0.001, 'kernel': 'rbf'}
```

```
grid_pred = grid.predict(X_test_97)
cm = confusion_matrix(y_test_97.values.ravel(), grid_pred)
print ("Confusion Matrix : \n", cm)
print(classification_report(y_test_97.values.ravel(), grid_pred, target_names=['happy', 'sad', 'angry']))
```

```
Confusion Matrix :
 [[28 12  0]
 [ 9 31  0]
 [ 0  1 31]]
              precision    recall  f1-score   support

       happy       0.76      0.70      0.73        40
         sad       0.70      0.78      0.74        40
       angry       1.00      0.97      0.98        32

    accuracy                           0.80       112
   macro avg       0.82      0.81      0.82       112
weighted avg       0.81      0.80      0.80       112
```

Figure B8. This shows how to implement SVM and evaluate the model performance with a classification report for SVM.

```
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
lw = 2
# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])
# Finally average it and compute AUC
mean_tpr /= n_classes
fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (area = {0:0.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)
colors = cycle(["aqua", "darkorange", "lightgreen"])
for i, color in zip(range(n_classes), colors):
    plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=lw,
        label="ROC curve of class {0} (area = {1:0.2f})".format(i, roc_auc[i]),
    )
plt.plot([0, 1], [0, 1], "k--", lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("SVM Receiver operating characteristic to multiclass")
plt.legend(loc="lower right")
plt.show()
```

Figure B9. This shows how to implement SVM and evaluate the model performance with ROC-AUC for SVM.

RF Grid Search

```
# defining parameter range

#number of trees
n_estimators=np.arange(100,302,20)
#number of features
max_features=['auto','sqrt']
#max levels
#max_depth=None
#min number of samples to split a node
min_samples_split=np.arange(2,11,1)
#min number of samples each leaf node
min_samples_leaf=np.arange(1,11,1)
#method selecting sample for training each tree
bootstrap=[True,False]

param_grid = {'n_estimators': n_estimators,
              'max_features':max_features,
              'min_samples_split':min_samples_split,
              'min_samples_leaf':min_samples_leaf,
              'bootstrap':bootstrap
             }

rf_Grid = GridSearchCV(rf_model, param_grid, cv=5, refit = True, verbose = 3, n_jobs=-1)

# fitting the model for grid search
rf_Grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 3960 candidates, totalling 19800 fits
```

Figure B10. This shows how to implement Random Forest using GridSearchCV.

```
In [4]: rf_model1=RandomForestClassifier(bootstrap=False, max_features='sqrt', min_samples_leaf=5, min_samples_split=6, n_estimators=241)
        rf_model1.fit(X_train, y_train)
        # print prediction results
        print(classification_report(y_test, rf_model1.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.71      0.68      0.69        40
           1       0.69      0.78      0.73        40
           2       0.93      0.84      0.89        32

    accuracy                           0.76       112
   macro avg       0.78      0.76      0.77       112
weighted avg       0.77      0.76      0.76       112
```

Figure B11. This shows how to implement RF using the result from GridSearchCV and evaluate the model.

In [98]:
```
# Binarize the output
y_train1=label_binarize(y_train, classes=[0,1,2])
y_test1=label_binarize(y_test, classes=[0,1,2])
y_score=rf_model1.predict_proba(X_test)
n_classes = 3
lw=2
# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test1[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes
fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (area = {0:0.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)
colors = cycle(["aqua", "darkorange", "cornflowerblue"])
for i, color in zip(range(n_classes), colors):
    plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=lw,
        label="ROC curve of class {0} (area = {1:0.2f})".format(emo_class[i], roc_auc[i]),
    )
plt.plot([0, 1], [0, 1], "k--", lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve for RF")
plt.legend(loc="lower right")
plt.show()
```

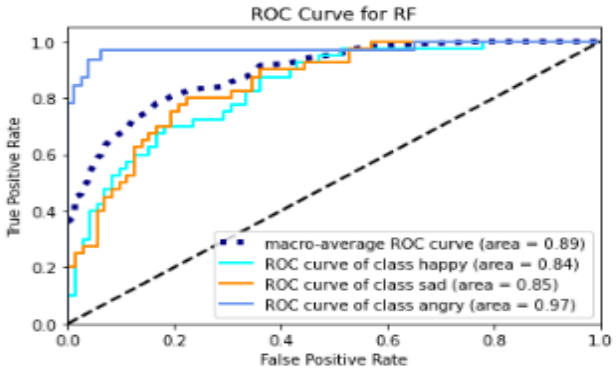


Figure B12. This shows how to draw the ROC Curve for RF.

```
model = KNeighborsClassifier(n_neighbors = 1, p= 2, weights= 'uniform')
model.fit(X_train_97, y_train_97.values.ravel())
y_pred = model.predict(X_test_97)
score = accuracy_score(y_pred,y_test_97)
print("Accuracy:", score)
cm = confusion_matrix(y_test_97.values.ravel(), y_pred)
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax)
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('KNN Confusion Matrix at PCA 97%')
ax.xaxis.set_ticklabels(['Happy0', 'Sad1' , 'Angry2']); ax.yaxis.set_ticklabels(['Happy0', 'Sad1' , 'Angry2'])
```

```
Accuracy: 0.7678571428571429

[Text(0, 0.5, 'Happy0'), Text(0, 1.5, 'Sad1'), Text(0, 2.5, 'Angry2')]
```

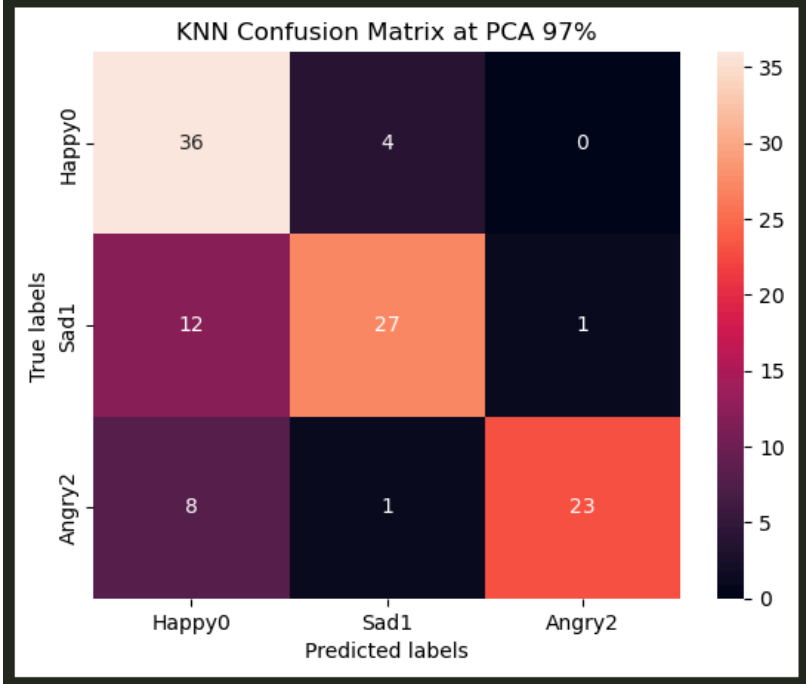


Figure B13. This shows how to implement KNN and evaluate the model performance with a confusion matrix.

```
    from sklearn.model_selection import GridSearchCV
    model = KNeighborsClassifier()
    # prepare a range of alpha values to test
    param_grid = {'n_neighbors':list(range(1,30)) , 'weights': ['uniform'] , 'p':[1,2]}
    grid = GridSearchCV(model, param_grid, cv =5, scoring = 'accuracy', refit=True, verbose = 1, return_train_score = True)
    grid.fit(X_train_97, y_train_97.values.ravel())
    print(grid.best_score_)
    print(grid.best_estimator_)
```

```
Fitting 5 folds for each of 58 candidates, totalling 290 fits
0.7844639120586384
KNeighborsClassifier(n_neighbors=1)
```

```
    print(grid.best_params_)
```

```
{'n_neighbors': 1, 'p': 2, 'weights': 'uniform'}
```

```
    grid_pred = grid.predict(X_test_97)
    cm = confusion_matrix(y_test_97.values.ravel(), grid_pred)
    print ("Confusion Matrix : \n", cm)
    print(classification_report(y_test_97.values.ravel(), grid_pred, target_names=['happy', 'sad', 'angry']))
```

```
Confusion Matrix :
 [[36  4  0]
 [12 27  1]
 [ 8  1 23]]
              precision    recall  f1-score   support

       happy       0.64      0.90      0.75        40
         sad       0.84      0.68      0.75        40
       angry       0.96      0.72      0.82        32

    accuracy                           0.77       112
   macro avg       0.81      0.76      0.77       112
weighted avg       0.80      0.77      0.77       112
```

Figure B14. This shows how to implement KNN and evaluate the model performance with a classification report.

```
from itertools import cycle
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
lw = 2
# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (area = {0:0.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)
colors = cycle(["aqua", "darkorange", "lightgreen"])
for i, color in zip(range(n_classes), colors):
    plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=lw,
        label="ROC curve of class {0} (area = {1:0.2f})".format(i, roc_auc[i]),
    )
plt.plot([0, 1], [0, 1], "k--", lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("KNN Receiver operating characteristic to multiclass")
plt.legend(loc="lower right")
plt.show()
```

Figure B15. This shows how to implement KNN and evaluate the model performance with ROC-AUC.