

## 1. INTRODUCTION

### 1.1 OVERVIEW

Today, the computer vision technique from face detection, ...etc. and machine learning is no longer strange to everyone, especially when people are in the era of 4.0 technology. Accordingly, there are the numerous needs for applying these technologies for many innovations that can improve human life quality or help people to make the more powerful decisions. From the inspiration of customer buying process in which when customers in the 'Consideration' stage, they must go through many reads, comparisons from many people from the online websites or customer's friends before making the buying decision. That 'Consideration' stage can cost customers the huge amount of time and energy. Accordingly, we have decided to build one 'Suggestion System' and choose beverage which are Soft Drinks (such as Coca Cola, Pepsi), Energy Drinks (including Red Bull, Monster Energy, and coffee), and Alcoholic Beverages (encompassing cocktails) as the product that the customers can buy. Instead of asking for the reviews or comparisons, in our 'Beverage Suggestion System', the customer can take the beverage suggestion from many people that have the same characteristics such as age, gender, income, temperature, facial expressions, and mood.

Our machine learning operations applied in system will learn the decision from many customers and people who share nearly the same characteristics and circumstances. Thus, the system provides highly personalized suggestions, ensuring that each customer receives beverage recommendations tailored to their unique profile and circumstances. Moreover, a positive and personalized experience during the "Consideration" stage contributes to overall customer satisfaction, potentially fostering brand loyalty and positive word-of-mouth. By integrating various data points and cutting-edge technologies, this system strives to make beverage recommendations that go beyond mere preferences, catering to the holistic needs and emotions of each individual customer.

## 1.2 MISSION

Below are the set tasks and the approach for addressing the thesis.

- ❖ Examine the suitability of the characteristic and circumstances which are appropriated for our beverage suggestion system. And research the data collection way then takes the appropriated survey for obtaining the customer data for beverages choices for our machine learning.
- ❖ Conduct in-depth research and cultivate a comprehensive understanding of convolutional neural networks (CNNs), exploring their intricacies, advancements, and applications in the realm of artificial intelligence.
- ❖ Incorporate relevant research papers that align with our system's objectives, thereby ensuring the continuous enhancement and optimization of our technology for delivering unparalleled personalized beverage recommendations.
- ❖ Undertake a comprehensive exploration and understanding of machine learning operations and a diverse array of data science methodologies, delving into their intricacies, applications, and advancements within the context of dataset science.
- ❖ Meticulously selecting and implementing the most fitting models based on robust data science methods, we aspire to enhance the intelligence, accuracy, and personalization of our system, ensuring it remains at the forefront of innovation in delivering tailored and exceptional beverage recommendations to our users.
- ❖ Build the work or process flow for our application and deploy beverage suggestion system as software.

## **2. COMPUTER VISION**

### **2.1 AGE AND GENDER**

#### **2.1.1 Introduction**

In the realm of data science and machine learning, capturing new data for applications is crucial. With this in mind, our approach involves leveraging computer vision techniques alongside Convolutional Neural Network (CNN) architectures. This methodology is specifically tailored to extract age and gender information from users within our Suggestion System.

However, we encountered constraints in terms of training resources and hardware capabilities. To navigate these limitations, we opted to bypass the initial AI research phase. Instead, we focused on adapting and employing a pretrained model. This model encompasses both the weights and CNN architectures detailed in the influential AI paper "Age and Gender Classification using Convolutional Neural Networks" by Gil Levi and Tal Hassner.

Levi and Hassner's research demonstrate that utilizing CNNs to learn representations markedly enhances performance in age and gender classification tasks. They advocate for a streamlined convolutional net architecture, which remains effective even with limited learning data. This aspect is particularly beneficial for our project, given our resource constraints.

Furthermore, the authors strive to bridge the disparity between automated face recognition and age/gender estimation methodologies. Their elegantly simple network architecture takes into account the scarcity of precise age and gender labels in facial datasets. This consideration is pivotal for applications like ours, where accuracy in age and gender prediction is essential, yet must be balanced with the practicalities of dataset limitations and computational resources.

### 2.1.2 Overview about the CNN architecture in Age and Gender classification

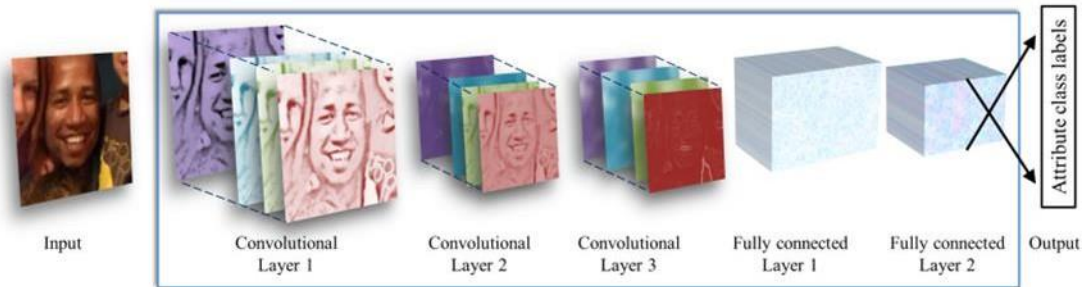


Figure 2-1. Illustration of our CNN architecture

Building upon the overview of our CNN architecture, let's delve into how each convolutional layer and fully connected layer contributes to feature extraction in the age and gender classification problem:

- **First Convolutional Layer (96 filters of  $7 \times 7$  pixels):** This layer is the first step in feature extraction. The larger filter size ( $7 \times 7$ ) allows the network to capture more complex and larger-scale features in the input images, such as shapes and edges, which are fundamental in recognizing facial features that contribute to age and gender prediction. Local response normalization following this layer enhances the model's generalization by reducing the likelihood of overfitting.
- **Second Convolutional Layer (256 filters of  $5 \times 5$  pixels):** With a slightly smaller filter size, this layer can detect more refined features, building upon the initial patterns recognized by the first layer. This layer continues to focus on important facial attributes, such as the contours of the eyes, nose, and mouth, which are crucial indicators of age and gender.
- **Third Convolutional Layer (384 filters of  $3 \times 3$  pixels):** The smallest filters in this layer capture the most intricate and subtle features, further refining the model's understanding of the facial characteristics. This level of detail aids in

distinguishing finer age-related traits and subtle gender-specific features, which might be overlooked by larger filters.

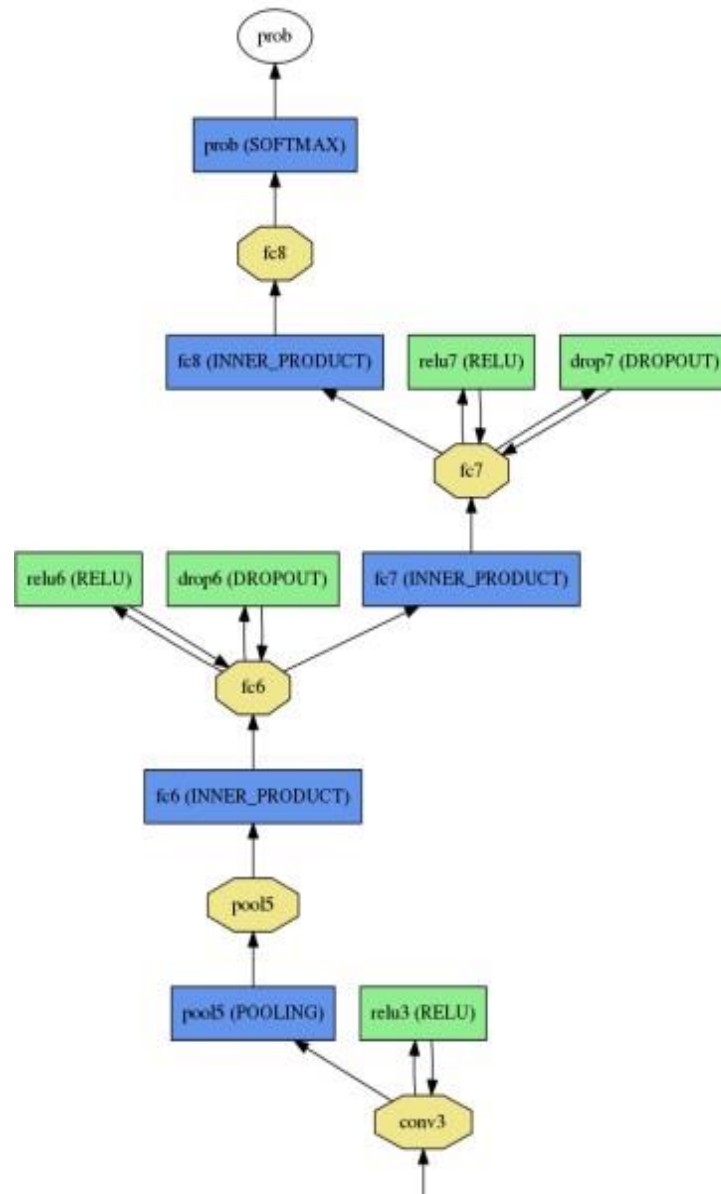
- **Rectified Linear Operation and Pooling Layer:** Following each convolutional layer, these operations aid in reducing the dimensionality of the feature maps while retaining the most significant features. This not only improves computational efficiency but also helps in extracting robust features that are invariant to minor variations and noise in the input images.
- **Fully Connected Layers (each with 512 neurons):** These layers integrate the features extracted by the convolutional layers, allowing the network to learn complex relationships between them. The high number of neurons in each layer provides the network with the capacity to learn a wide array of features representing various age groups and gender characteristics. These layers are crucial in making final predictions, as they combine all learned features into a comprehensive understanding of the input data.

The field of age and gender classification from facial images has seen significant advancements in recent years. Various methods have been explored, ranging from early techniques that calculate ratios between facial features to more contemporary methods utilizing subspace or manifold representations. Accordingly, each layer of the CNN architecture plays a pivotal role in recognizing and interpreting the diverse and nuanced features relevant to age and gender classification. The convolutional layers focus on hierarchical feature extraction, from basic to complex, while the fully connected layers synthesize these features to make accurate predictions. This structure ensures that the network is both powerful and efficient, capable of handling the intricacies involved in accurately determining age and gender from facial characteristics.

### **2.1.3 Full schematic and detailed CNN for age and gender estimation**

The two authors had their choice of smaller network design is motivated both

from their desire to reduce risk of overfitting as well as the nature of problems they are attempting to solve: age classification on the Adience set requires distinguishing between eight classes; gender only two.



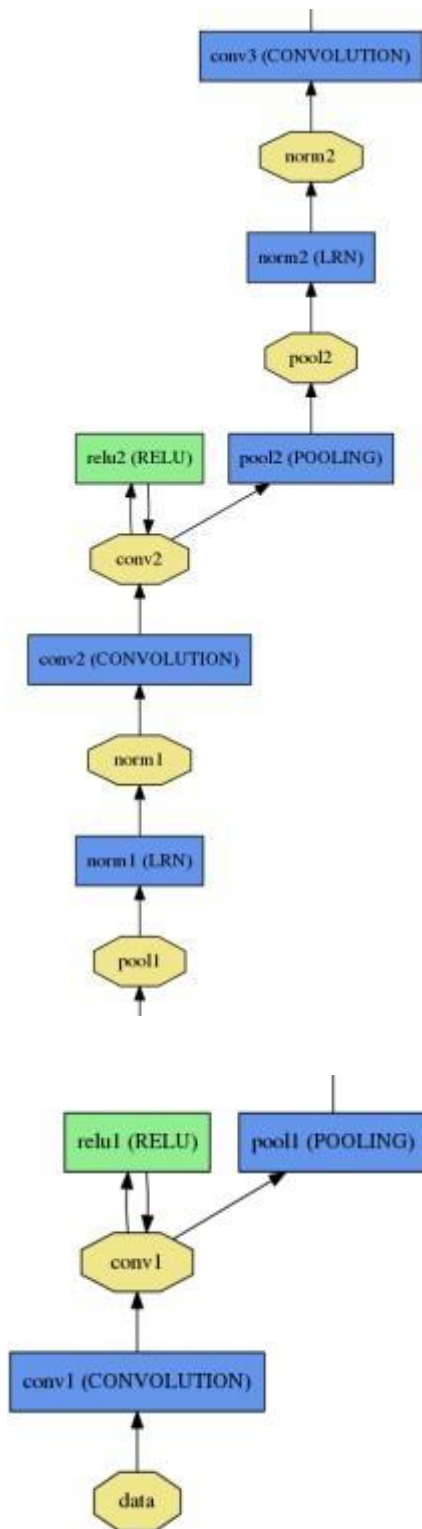


Figure 2-2: Full schematic diagram of our network architecture.

The CNN architecture, tailored for age and gender classification, efficiently processes images through a series of layers, each with its specific function and structure:

- **Initial Image Processing:** Images are first resized to 256x256 pixels, from which a 227x227 pixel crop is extracted. This standardized input size ensures consistency across all images fed into the network.
- **First Convolutional Layer:** Here, 96 filters of 3x7x7 pixels are applied to the input. This is followed by a Rectified Linear Unit (ReLU) for non-linear processing, a max pooling layer that extracts the maximum value from 3x3 pixel regions with strides of two pixels, and a local response normalization layer. This layer focuses on extracting primary features from the raw pixel data.
- **Second Convolutional Layer:** The output from the first layer (96x28x28) undergoes further processing with 256 filters of 96x5x5 pixels. This layer, also followed by ReLU, max pooling, and local response normalization, refines the feature extraction, focusing on more complex patterns and textures.
- **First Fully Connected Layer:** This layer contains 512 neurons and receives input from the last convolutional layer. It's followed by ReLU and a dropout layer, which helps prevent overfitting by randomly deactivating neurons during training.
- **Second Fully Connected Layer:** Similar to the first, this layer also contains 512 neurons and processes the output from the preceding layer. It is followed by ReLU and another dropout layer, further refining the feature integration.
- **Third Fully Connected Layer:** This layer is crucial as it maps the extracted and processed features to final classes for age and gender. It essentially makes the final decision based on the comprehensive analysis performed by previous layers.
- **Output with Softmax Layer:** The output from the last fully connected layer is then fed into a softmax layer. This layer assigns a probability to each class,



and the classification is determined by selecting the class with the highest probability.

Overall, this architecture not only captures and processes the raw data from images but also systematically refines and interprets these features to make accurate age and gender predictions. The layer-by-layer progression is designed to gradually move from basic feature detection to complex pattern recognition, culminating in a sophisticated classification decision.

#### **2.1.4 Network training**

In the study paper, we employed a streamlined network architecture and incorporated two key strategies to minimize overfitting. Firstly, they integrated dropout learning, which involves randomly nullifying the output of network neurons. This was achieved through two dropout layers, each with a 50% probability of deactivating a neuron's output. Secondly, they enhanced our data processing by randomly cropping a 227x227 pixel section from the original 256x256 pixel input image and randomly mirroring it during each training cycle. This approach is akin to the varied cropping and mirroring techniques utilized in other studies.

The training process relied on stochastic gradient descent with batches of fifty images. We began with an initial learning rate of  $e^{-3}$ , which was later decreased to  $e^{-4}$  after 10,000 iterations.

For age and gender prediction using novel faces, we tested two different methods:

- **Center Crop:** Here, we fed the network with a face image cropped to 227x227 pixels, focusing on the face's center.
- **Over-sampling:** This involved extracting five 227x227 pixel sections from the face image, including four from the corners and one from the center. We presented all these images, along with their horizontal reflections, to the network. The final prediction was the average of the outcomes from these variations.

They observed that slight misalignments in the Adience image dataset, often due to challenges like occlusions and motion blur, significantly affected our results.

The over-sampling method was specifically designed to counter these misalignments. Instead of improving alignment quality, they provided the network with variously translated versions of the same face, allowing it to adjust for these minor discrepancies.

### **2.1.5 The Adience dataset/ benchmark**

In the research, we evaluated the effectiveness of our Convolutional Neural Network (CNN) design using the Adience benchmark, a resource specifically developed for age and gender classification. The uniqueness of the Adience dataset lies in its composition; it contains images that were automatically uploaded to Flickr from smartphones. Unlike other image sets often used in media webpages or social websites, the Adience collection was not manually curated before upload. This lack of pre-selection means that the Adience images represent a wide array of real-world conditions, including variations in head pose, lighting quality, and other environmental factors.

The Adience dataset comprises approximately 26,000 images representing 2,284 individuals, categorized into various age groups. To assess age and gender classification, they employed a standard five-fold, subject-exclusive cross-validation protocol. This method ensures that each subject's images are only in one of the test or training sets, avoiding any bias from repeated appearances. We used the in-plane aligned version of the faces from the Adience dataset, following the original alignment used in previous studies. This choice allows two authors to focus on the gains achieved through our network architecture, rather than improvements from more advanced preprocessing techniques.

A key aspect of their study was the consistent use of the same network architecture across all test folds and for both gender and age classification tasks. This consistency was crucial for validating our results across different scenarios and for demonstrating the versatility of our proposed network design. They aimed to show that our architecture is capable of delivering robust performance across varied and

related challenges.

### **2.1.6 Result**

Their method is implemented by using and modifying the structure of CNN layers with Caffe open-source frameworks. For short explanation about Caffe model files, these are binary files that store the weights and biases of a trained neural network. After training a network defined by a prototxt file, Caffe saves the learned parameters in a model file (typically with a .caffemodel extension). This file can then be loaded to deploy the trained model for tasks like classification, detection, etc.

In their study on age classification, we evaluated the performance of our algorithm in two ways. First, they assessed its accuracy in correctly identifying the precise age group of a subject. Second, they considered cases where the algorithm's prediction was close but not exact, specifically when it identified an age group that was immediately older or younger than the actual age group of the subject. This approach is consistent with previous studies in the field and acknowledges the inherent challenges in age classification. The rationale behind this method is that facial features can be quite similar between the oldest individuals in one age group and the youngest in the next, making precise classification difficult.

The two authors of paper has compared their accuracy with two papers which are 'Age and gender estimation of unfiltered faces' from E. Eidinger, R. Enbar, and T. Hassner and 'Effective face frontalization in unconstrained images' from Proc. Conf. Comput. Vision Pattern Recognition, 2015 from T. Hassner, S. Harel, E. Paz, and R. Enbar. These two other methods have the accuracy of nearly 80. However, for Gil Levi and Tal Hassner's method which are proposed using single crop and proposed using over-sampling, the mean accuracy  $\pm$  standard error over all age categories has risen above 80.

Proposed using single crop	$85.9 \pm 1.4$
Proposed using over-sample	$86.8 \pm 1.4$

Table 2-1: Gender estimation results on the Adience benchmark.

For the age estimation results, there are the ‘Exact’ and ‘1-off ‘ for evaluation

Proposed using single crop	$49.5 \pm 4.4$	$84.6 \pm 1.7$
Proposed using over-sample	$50.7 \pm 5.1$	$84.7 \pm 2.2$

Table 2-2: Age estimation results on the Adience benchmark

In Figures 3, we present several instances where the system incorrectly classified gender and age. These examples highlight that a significant number of errors are attributable to the demanding visual conditions found in some images of the Adience benchmark. Particularly, errors often arise from issues like blurriness, low resolution, and obstructions, such as those caused by heavy makeup. Additionally, they observed frequent misjudgments in gender determination for images of infants and very young children, where gender-specific characteristics are not distinctly apparent.



Figure 2-3: Gender misclassifications. Top row: Female subjects mistakenly classified as males.  
Bottom row: Male subjects mistakenly classified as females

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60-
0-2	<b>0.699</b>	0.147	0.028	0.006	0.005	0.008	0.007	0.009
4-6	0.256	<b>0.573</b>	0.166	0.023	0.010	0.011	0.010	0.005
8-13	0.027	0.223	<b>0.552</b>	0.150	0.091	0.068	0.055	0.061
15-20	0.003	0.019	0.081	<b>0.239</b>	0.106	0.055	0.049	0.028
25-32	0.006	0.029	0.138	0.510	<b>0.613</b>	0.461	0.260	0.108
38-43	0.004	0.007	0.023	0.058	0.149	<b>0.293</b>	0.339	0.268
48-53	0.002	0.001	0.004	0.007	0.017	0.055	<b>0.146</b>	0.165
60-	0.001	0.001	0.008	0.007	0.009	0.050	0.134	<b>0.357</b>

Table 2-3: Age estimation confusion matrix on the Adience benchmark.

Analyzing the confusion matrix in table 3 for age classification, we can draw several insights:

- The True Positive rate: the highest values are along the diagonal, indicating correct classifications. For instance, the age group '25-32' shows a high accuracy rate of 0.613, suggesting the model performs best in this category.
- Age Group '0-2': There's a reasonably high accuracy (0.699) in classifying the youngest age group ('0-2'). However, there's a notable confusion with the '4-6' age group, where 25.6% of the '0-2' age group is misclassified.
- Adjacent Age Groups: The confusion primarily occurs with adjacent age groups. For example, '8-13' is often confused with '4-6' and '15-20'. This is common in age classification due to gradual changes in facial features.
- Older Age Groups: For older age groups like '48-53' and '60-', the accuracy decreases (0.146 and 0.357, respectively). There's significant confusion in these groups with adjacent categories, indicating challenges in distinguishing features in older age ranges.
- Misclassification Trends:

- Younger age groups ('0-2', '4-6', '8-13') are less likely to be confused with much older groups.
- Middle age groups, especially '25-32', show confusion with a broader range of ages, indicating more subtle differences in facial features across these ranges.
- The model struggles more with the oldest groups ('48-53', '60-'), possibly due to more pronounced variations in aging signs among individuals.

Overall, this matrix in table 3 reveals the model's strengths and weaknesses in age classification, particularly its ability to distinguish between closely related age groups and its challenges with older age categories.

## **2.2 FACE EXPRESSION**

### **2.2.1 Introduction**

As we progress with enhancing data acquisition for our recommended applications, we will implement computer vision techniques to empower our applications in capturing facial expressions of both 'Happy' and 'Normal' emotions. Rather than adjusting pre-existing weights, our approach involves training the model using the Keras framework to construct the CNN layers, ensuring the creation of weight configurations adept at accurately discerning emotions displayed on human faces.

The intricate dynamics of human interactions are profoundly shaped by emotions, exerting a substantial impact on our actions and choices. Grasping and deciphering these emotions poses a formidable challenge, yet proves indispensable across diverse domains like psychology, market research, and human-computer interaction. Notably, the realm of machine learning has witnessed a burgeoning fascination with the automated detection and classification of emotions. Among the

formidable array of techniques, the Convolutional Neural Network (CNN) stands out as a potent force in this evolving landscape.

Emotion detection involves the identification and classification of human emotions derived from diverse sources, including facial expressions, voice intonation, and textual content. Its primary objective is to offer valuable insights into individuals' emotional states and facilitate responsive actions by systems. The rapid progress in artificial intelligence and deep learning has significantly propelled the field of emotion detection, fostering its widespread adoption across various domains. This surge in interest is attributed to the promising applications it holds in leveraging nuanced emotional understanding for enhanced human-computer interactions, personalized services, and more.

A Convolutional Neural Network (CNN) stands as a specialized neural network particularly applied in tasks related to computer vision. Its effectiveness lies in its adeptness at analyzing and discerning patterns from visual data, making it a valuable tool for endeavors like image classification and object recognition. Comprising various layers, including convolutional, pooling, and fully connected layers, CNNs collaboratively engage in the hierarchical extraction and processing of features.

For the specific task of emotion detection, the CNN undergoes training using an extensive dataset populated with labeled emotional expressions. Typically, this dataset undergoes preprocessing to ensure data uniformity and eliminate extraneous noise. The CNN model is then constructed and trained on this refined dataset, with the overarching goal of acquiring the capability to recognize patterns and features associated with distinct emotions. The performance of the trained model is subsequently assessed using metrics such as accuracy, precision, and recall, providing a quantitative measure of its effectiveness in emotion detection tasks. This robust methodology showcases the CNN's prowess in discerning nuanced visual cues indicative of various emotional states.

Despite its efficacy, the utilization of CNNs for emotion detection encounters various challenges and constraints. One such challenge arises from the considerable variability in emotional expressions exhibited by individuals. The diverse ways in which people express the same emotion present a hurdle, complicating the creation of a comprehensive model capable of generalizing across heterogeneous populations.

Cross-cultural disparities in expressions further compound these challenges. Emotions are susceptible to influence from cultural norms and societal factors, necessitating the training of models on diverse datasets to enhance accuracy and inclusivity.

Additionally, privacy and ethical considerations emerge as crucial facets. The deployment of emotion detection technology prompts concerns regarding data privacy, consent, and the potential for the improper use of personal information. Implementation of safeguards becomes imperative to protect user privacy and ensure the responsible application of this technology.

### **2.2.2 Technology stacks**

Emotion recognition has gained substantial attention in various domains, including human-computer interaction and mental health analysis. Convolutional Neural Networks (CNNs) have proven to be powerful tools for image-based tasks, and the Python programming language, coupled with the Keras framework, provides an efficient and accessible platform for implementing and training CNN models. This report explores the efficiency of Python coding and the Keras framework in building CNN layers for detecting facial emotions.

Emotion detection from facial expressions plays a pivotal role in understanding human behavior. The utilization of deep learning techniques, specifically CNNs, has significantly improved the accuracy and efficiency of emotion recognition systems. Python, as a versatile and widely-used programming language, combined with the Keras deep learning framework, offers a streamlined approach to developing and training CNN models.



Python's simplicity and readability make it an ideal language for rapid prototyping and development. Leveraging Python libraries such as NumPy, OpenCV, and Matplotlib enhances data preprocessing, image manipulation, and result visualization. Keras, a high-level neural networks API, provides a user-friendly interface for building and training deep learning models.

The architecture of a CNN is crucial for its performance in facial emotion recognition. Python's syntax facilitates the construction of intricate CNN layers using libraries like TensorFlow and PyTorch. Keras further simplifies this process by offering pre-defined layers and models, allowing developers to focus on the model's architecture and hyperparameter tuning.

Keras simplifies the training of CNN models with its high-level API. Developers can easily define loss functions, optimizers, and metrics, streamlining the training process. Python's compatibility with GPU acceleration through libraries like TensorFlow GPU or PyTorch accelerates model training, significantly reducing computation time.

Python's extensive ecosystem of machine learning libraries, coupled with Keras' simplicity, facilitates the evaluation of model performance. Matplotlib and Seaborn aid in visualizing training and validation metrics, guiding the hyperparameter tuning process to enhance the model's accuracy and generalization.

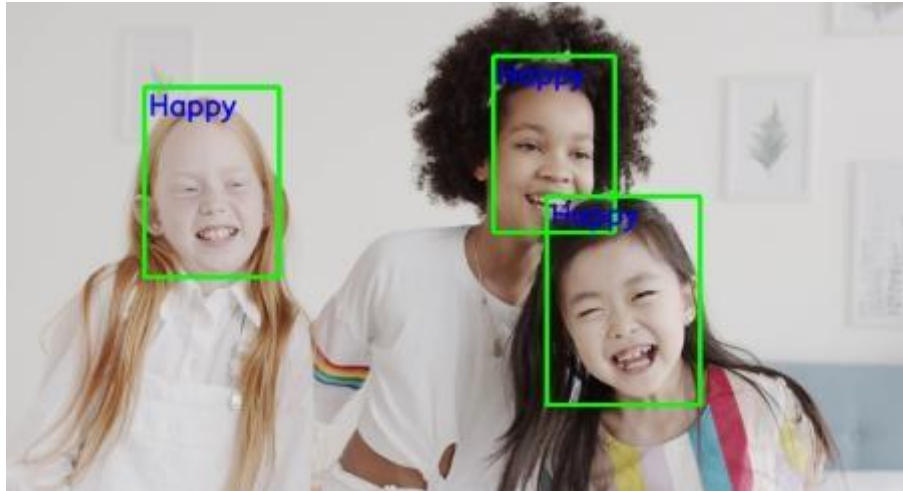


Figure 2-4: Example of face expression

### 2.2.3 Facial expression dataset

The FER2013 dataset, available on Kaggle, has emerged as a valuable resource for training Convolutional Neural Networks (CNNs) dedicated to facial expression recognition. This dataset encompasses a diverse range of facial expressions, providing a rich source for developing robust models.

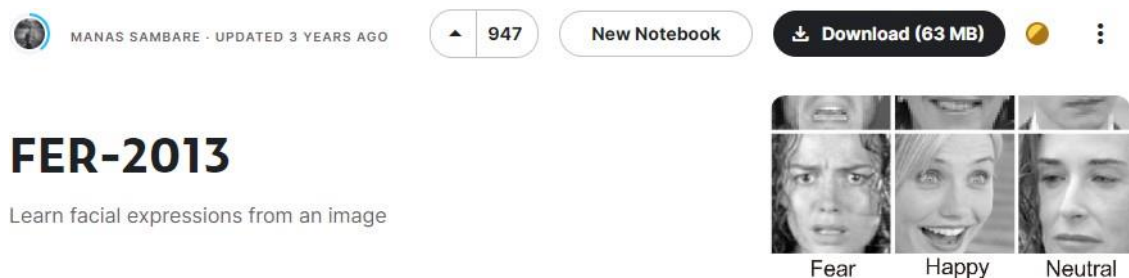


Figure 2-5: FER-2013 dataset in kaggle

FER2013 comprises over 35,000 labeled images, categorized into seven emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral. But in our applications or in our training process, we need only two expressions which are the 'Happy' and 'Neutral'. Each image is 48x48 pixels, making it a manageable yet comprehensive dataset for emotion recognition model training. The dataset's diversity captures a wide spectrum of facial expressions, enhancing the model's ability to generalize to various real-world scenarios.

The FER2013 dataset facilitates the development of CNN models that generalize well to unseen data. The variety of subjects, lighting conditions, and facial orientations in the dataset challenges the model to learn robust features, reducing the risk of overfitting to specific instances. This generalization capability is crucial for the model's real-world applicability.

Imbalance in class distribution is a common challenge in emotion recognition datasets. The FER2013 dataset, while not perfectly balanced, offers a reasonable distribution across emotion classes. Python's data manipulation libraries, such as NumPy and pandas, make it easy to implement strategies like oversampling or data augmentation to address class imbalances during model training.

Our training process have two folders: test and train. The test folder will contain 1774 images for 'Happy' and 1233 images for 'Neutral'. Moreover, the training folder will contain the 4965 images for 'Neutral' and 7215 images for 'Happy'. This imbalance in train folder might influence the model's learning process, emphasizing the need for careful handling through techniques like oversampling, undersampling, or the use of class weights during training. Furthermore, the test folder is comparatively smaller, with 1774 images for 'Happy' and 1233 images for 'Neutral.' It's crucial to ensure that the test set is representative of real-world scenarios and contains a diverse set of images to assess the model's generalization performance accurately.

The training set, especially for the 'Neutral' class, could benefit from data augmentation to artificially increase the diversity of images. Techniques like rotation, flipping, and zooming can help the model generalize better to unseen instances and improve its robustness.

Given the class imbalance in the training set, the evaluation metrics chosen during model assessment should consider this skewness. Metrics like precision, recall, and F1-score for both classes can provide a more comprehensive understanding of how well the model performs on each emotion category.

Python, with its extensive set of image processing libraries, facilitates efficient preprocessing of FER2013 images. OpenCV can be employed for tasks such as

resizing, normalization, and histogram equalization, ensuring that the input data is conducive to effective CNN model training. The simplicity of Python code streamlines the integration of these preprocessing steps with the model development pipeline.

### 2.2.4 Training configuration

Firstly, in our training process, we have the Image data generators ('train\_data\_gen' and 'validation\_data\_gen') are initialized with rescaling, which scales pixel values by dividing them by 255. Both generators resize images to (48, 48) pixels, use grayscale color mode, and organize data into batches of 64.

```
# Initialize image data generator with rescaling
train_data_gen = ImageDataGenerator(rescale=1./255)
validation_data_gen = ImageDataGenerator(rescale=1./255)

# Preprocess all test images
train_generator = train_data_gen.flow_from_directory(
    '/space/hotel/hienng/Face_Emotion/Face_Emotion/train',
    target_size=(48, 48), #each image will be resized to this size
    batch_size=64, #Instead of processing all the data at once, it is divided into smaller groups or batches, where each batch contains a fixed number of samples.
    color_mode="grayscale",
    class_mode='categorical')

# Preprocess all train images
validation_generator = validation_data_gen.flow_from_directory(
    '/space/hotel/hienng/Face_Emotion/Face_Emotion/test',
    target_size=(48, 48),
    batch_size=64, #Instead of processing all the data at once, it is divided into smaller groups or batches, where each batch contains a fixed number of samples.
    color_mode="grayscale",
    class_mode='categorical')
```

Figure 2-6: Train and Validation generator

Secondly for the model compilation, the model is compiled using the Adam optimizer with a learning rate of 0.0001 and a decay factor of 1e-6. Categorical cross-entropy is chosen as the loss function, and accuracy is used as the evaluation metric.

Thirdly for the training parameter, The model is trained using the 'fit\_generator' method and 'steps\_per\_epoch' is set to 12180 // 64, which is the number of training samples divided by the batch size (this numbers of training samples can be changed when we have the data augmentation technique). Training occurs over 40 epochs (epochs=40). Validation data is provided, and 'validation\_steps' is set to 3007 // 64, the number of validation samples divided by the batch size.

```
# Train the neural network/model
emotion_model_info = emotion_model.fit_generator(
    train_generator,
    steps_per_epoch=12180 // 64, #the number of samples and batch size
    epochs=40,
    validation_data=validation_generator,
    validation_steps=3007 // 64)
```

Figure 2-7: Config for training

Lastly for model saving, The model architecture is saved in a JSON file named ‘emotion\_model.json’. The trained model weights are saved in an HDF5 file named ‘emotion\_model.h5’.

```
# save model structure in json file
model_json = emotion_model.to_json()
with open("emotion_model.json", "w") as json_file:
    json_file.write(model_json)

# save trained model weight in .h5 file
emotion_model.save_weights('emotion_model.h5')
```

Figure 2-8: Code for saving weights

These configurations collectively set up and train a Convolutional Neural Network for emotion recognition using image data. The model is saved for future use or deployment, and the training process is configured with parameters such as batch size, learning rate, and number of epochs.

### 2.2.5 Model architecture

The Convolutional Neural Network (CNN) model for emotion recognition is structured to effectively capture hierarchical features from input facial images.

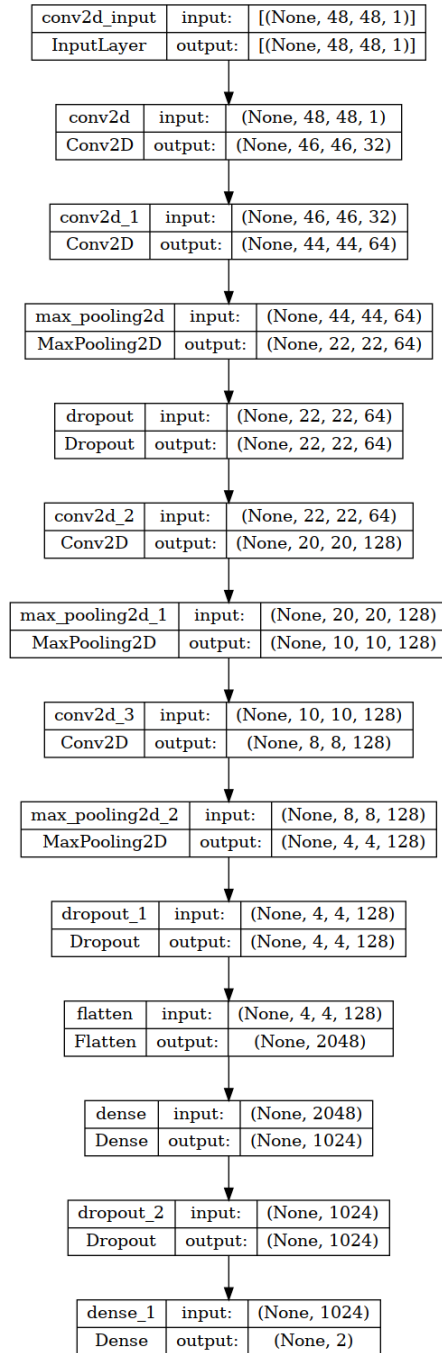


Figure 2-9: Model architecture

The first layer, '**Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48, 48, 1))**', applies 32 filters of size 3x3 to the input images with a Rectified Linear Unit (ReLU) activation function. This layer extracts low-level features such as edges and basic patterns. The total number of parameters in this layer is calculated as  $(3 \times 3 \times 1 + 1) \times 32 = 320$ , where 1 represents the bias term.

The subsequent convolutional layers (**'Conv2D(64, kernel\_size=(3, 3), activation='relu')'**, **'Conv2D(128, kernel\_size=(3, 3), activation='relu')'**, **'Conv2D(128, kernel\_size=(3, 3), activation='relu')'**) progressively increase the number of filters, enabling the extraction of more complex and abstract features from the input images.

After each convolutional layer, a **'MaxPooling2D(pool\_size=(2, 2))'** layer is applied. Max pooling reduces the spatial dimensions of the feature maps, helping to retain important information while reducing computation. It uses a 2x2 window to retain the maximum value within each region.

Dropout layers (**'Dropout(0.25) and Dropout(0.5)'**) are strategically inserted to mitigate overfitting. They randomly deactivate a fraction of neurons during training, promoting better generalization to unseen data.

The **'Flatten()'** layer transforms the 3D feature maps into a 1D vector, preparing the data for input into fully connected layers.

The first dense layer, **'Dense(1024, activation='relu')'**, consists of 1024 neurons with a ReLU activation function. This layer serves as a feature extractor, learning complex patterns from the flattened input. Another dropout layer (**'Dropout(0.5)'**) follows to further prevent overfitting. The final dense layer, **'Dense(2, activation='softmax')'**, contains only 2 nodes representing the target classes (emotion categories). It employs the softmax activation function to produce probability scores for each class, facilitating multi-class classification.

In summary, this CNN architecture systematically extracts features from facial images through convolutional and pooling layers, reduces overfitting with dropout layers, and finally classifies emotions using fully connected layers.

### 2.2.6 Training process and model evaluation

As we have mentioned early in the dataset part, imbalance in train folder might influence the model's learning process, emphasizing the need for careful handling

through techniques like data augmentation. Accordingly, we will training our model with two methods: one is still keep the same numbers of images from train folder which are the 4965 images for ‘Neutral’ and 7215 images for ‘Happy’ and one is applying the data augmentation for obtaining more images for ‘Neutral’ label.

For the data augmentation we will randomly choose 2000 images from the original images in ‘Neutral’ folder. Then images be rotated within a range of -20 to +20 degrees, shifted horizontally and vertically by up to 20% of their dimensions, sheared with a maximum intensity of 20%, zoomed by a maximum of 20%, horizontally flipped with a 50% probability, and newly created pixels during transformations are filled using the nearest pixel value. These augmentation techniques enhance the diversity of the training data, fostering robustness in the model by exposing it to various orientations, positions, and scales of objects in the images. This diversification aids in improving the model's ability to generalize to unseen data.

```
# Initialize ImageDataGenerator for augmentation
data_generator = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
```

Figure 2-10: Data augmentation for training

At first, we have trained the CNN model, using the original ‘Happy’ image folder and the augmented image data ‘Neutral’. Both conclude 16111 images. The epoch we set to 30 during the training. The valuation accuracy has increased gradually and reach the above 85%.



```

Epoch 21/30
251/251 [=====] - 9s 37ms/step - loss: 0.1993 - accuracy: 0.9197 - val_loss: 0.2989 - val_accuracy: 0.8787
Epoch 22/30
251/251 [=====] - 9s 38ms/step - loss: 0.1913 - accuracy: 0.9248 - val_loss: 0.3104 - val_accuracy: 0.8750
Epoch 23/30
251/251 [=====] - 10s 39ms/step - loss: 0.1853 - accuracy: 0.9258 - val_loss: 0.2990 - val_accuracy: 0.8767
Epoch 24/30
251/251 [=====] - 10s 39ms/step - loss: 0.1793 - accuracy: 0.9303 - val_loss: 0.2886 - val_accuracy: 0.8828
Epoch 25/30
251/251 [=====] - 9s 37ms/step - loss: 0.1730 - accuracy: 0.9328 - val_loss: 0.3129 - val_accuracy: 0.8798
Epoch 26/30
251/251 [=====] - 9s 37ms/step - loss: 0.1711 - accuracy: 0.9329 - val_loss: 0.3210 - val_accuracy: 0.8815
Epoch 27/30
251/251 [=====] - 9s 35ms/step - loss: 0.1636 - accuracy: 0.9344 - val_loss: 0.3206 - val_accuracy: 0.8767
Epoch 28/30
251/251 [=====] - 10s 39ms/step - loss: 0.1534 - accuracy: 0.9410 - val_loss: 0.2974 - val_accuracy: 0.8865
Epoch 29/30
251/251 [=====] - 10s 39ms/step - loss: 0.1517 - accuracy: 0.9401 - val_loss: 0.3159 - val_accuracy: 0.8818
Epoch 30/30
251/251 [=====] - 10s 38ms/step - loss: 0.1462 - accuracy: 0.9435 - val_loss: 0.3162 - val_accuracy: 0.8794

```

Figure 2-11: Training process

We saved weights to the h5 file and structure of model to json file. Then we modify it to examining the model evaluation with confusion matrix. Shortly, the confusion matrix is a crucial tool in model evaluation as it provides a clear and concise summary of a classification model's performance. It breaks down the predictions into four fundamental categories: true positives (correctly predicted positive instances), true negatives (correctly predicted negative instances), false positives (incorrectly predicted as positive), and false negatives (incorrectly predicted as negative). This matrix enables a deeper understanding of a model's strengths and weaknesses, allowing practitioners to assess its accuracy, precision, recall, and other performance metrics. By analyzing the confusion matrix, one can identify patterns of misclassifications and make informed decisions to enhance the model's overall effectiveness and reliability in real-world applications.

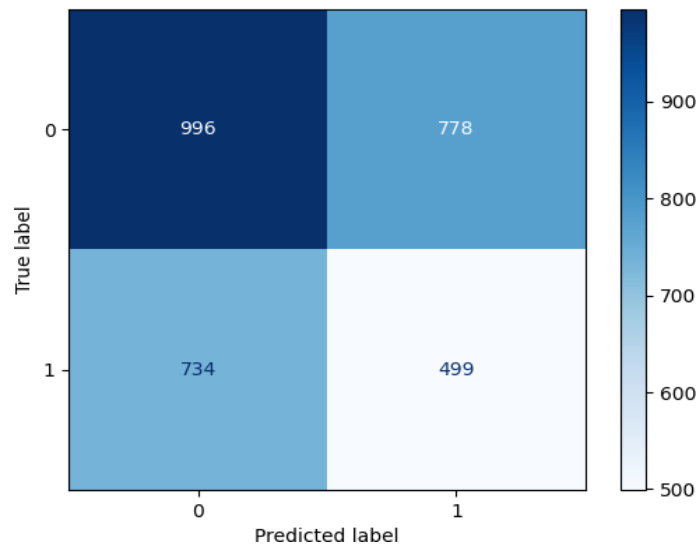


Figure 2-12: Confusion matrix

The confusion matrix indicates a balanced prediction with 996 true negatives, 499 true positives, 778 false positives, and 734 false negatives, suggesting potential areas for improving sensitivity and specificity in the model.

- Sensitivity (Recall) =  $TP / (TP + FN) = 499 / (499 + 734) \approx 0.405$
- Specificity =  $TN / (TN + FP) = 996 / (996 + 778) \approx 0.562$

The model shows moderate sensitivity but lower specificity.

For our training with the original ‘Happy’ and ‘Neutral’ images data which conclude about 1280 images, we applied 40 epochs and observe some slight improvements of the valuation accuracy:

```

Epoch 31/40
190/190 [=====] - 8s 42ms/step - loss: 0.1637 - accuracy: 0.9372 - val_loss: 0.2845 - val_accuracy: 0.8893
Epoch 32/40
190/190 [=====] - 8s 44ms/step - loss: 0.1606 - accuracy: 0.9383 - val_loss: 0.2801 - val_accuracy: 0.8906
Epoch 33/40
190/190 [=====] - 8s 43ms/step - loss: 0.1506 - accuracy: 0.9442 - val_loss: 0.2849 - val_accuracy: 0.8886
Epoch 34/40
190/190 [=====] - 8s 44ms/step - loss: 0.1474 - accuracy: 0.9431 - val_loss: 0.2953 - val_accuracy: 0.8896
Epoch 35/40
190/190 [=====] - 8s 43ms/step - loss: 0.1412 - accuracy: 0.9462 - val_loss: 0.3007 - val_accuracy: 0.8906
Epoch 36/40
190/190 [=====] - 8s 42ms/step - loss: 0.1374 - accuracy: 0.9468 - val_loss: 0.2905 - val_accuracy: 0.8910
Epoch 37/40
190/190 [=====] - 8s 43ms/step - loss: 0.1313 - accuracy: 0.9522 - val_loss: 0.2986 - val_accuracy: 0.8940
Epoch 38/40
190/190 [=====] - 8s 42ms/step - loss: 0.1245 - accuracy: 0.9535 - val_loss: 0.2983 - val_accuracy: 0.8906
Epoch 39/40
190/190 [=====] - 8s 43ms/step - loss: 0.1192 - accuracy: 0.9566 - val_loss: 0.3014 - val_accuracy: 0.8889
Epoch 40/40
190/190 [=====] - 8s 43ms/step - loss: 0.1151 - accuracy: 0.9557 - val_loss: 0.3099 - val_accuracy: 0.8872

```

Figure 2-13: Training process

The confusion matrix indicates that the model achieved 1086 true negatives and 492 true positives, while misclassifying 741 instances as false negatives and 688 instances as false positives. The balanced distribution of true positives and true negatives suggests a moderate overall performance, but the elevated numbers of false positives and false negatives highlight potential areas for improving precision and recall.

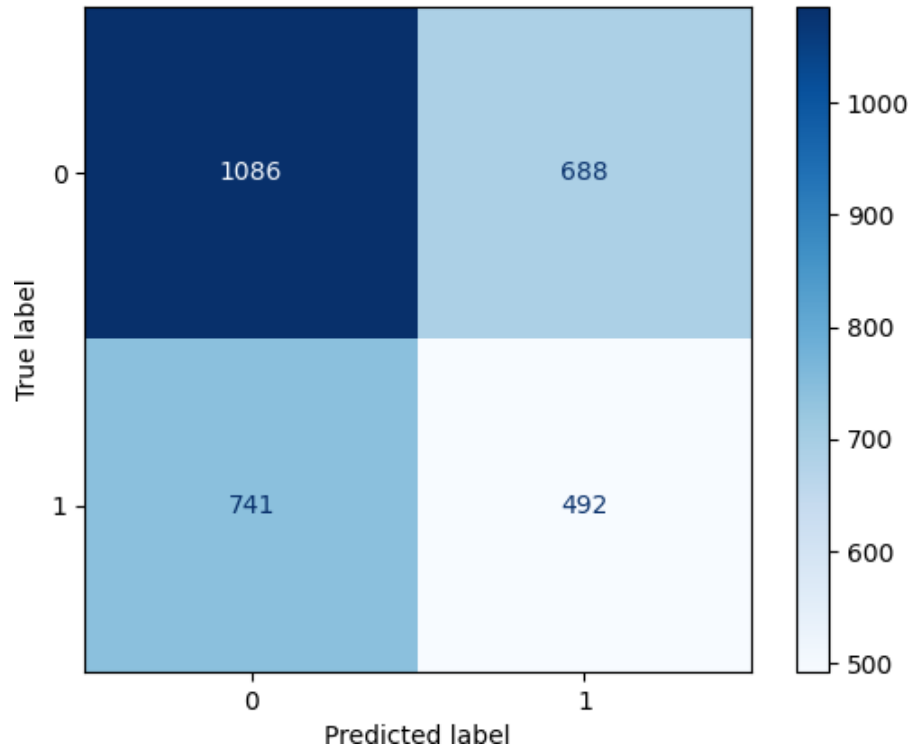


Figure 2-14: Confusion matrix

The analysis reveals that in two training scenarios – one involving augmented data and the other retaining all original data with imbalances in image quantities – both models exhibit effective predictions for 'Happy' faces. However, when classifying 'Neutral' or 'Normal' faces, suboptimal performance is observed, likely attributed to the resemblance between expressions of 'Neutral' and 'Happy' individuals. Despite these challenges, the weights trained on the original data will be retained and applied to our suggested applications, emphasizing their utility and generalization potential.

### 3. MACHINE LEARNING OPERATION

The most important element for our ‘Suggestion system’ is choosing the best and the most appropriated machine learning model for our problem. Accordingly, our machine learning operation will include these steps: data collection, data cleaning, data preprocessing, learning and evaluation, make prediction from new data captured in application

In a typical machine learning operation, the process begins with the collection of relevant data from various sources, followed by the crucial step of cleaning the data to address errors and ensure reliability. Subsequently, the data undergoes preprocessing, including normalization and encoding, to ready it for the learning phase, during which the model is trained on a prepared dataset using a specific algorithm. The model's performance is then evaluated using a separate testing set, employing metrics like accuracy and precision. Once successfully trained and evaluated, the model is deployed to make predictions on new, unseen data captured in applications or other sources, leveraging the patterns it learned during training. This iterative process may involve adjusting hyperparameters or acquiring more diverse data to enhance the model's accuracy and reliability over time.

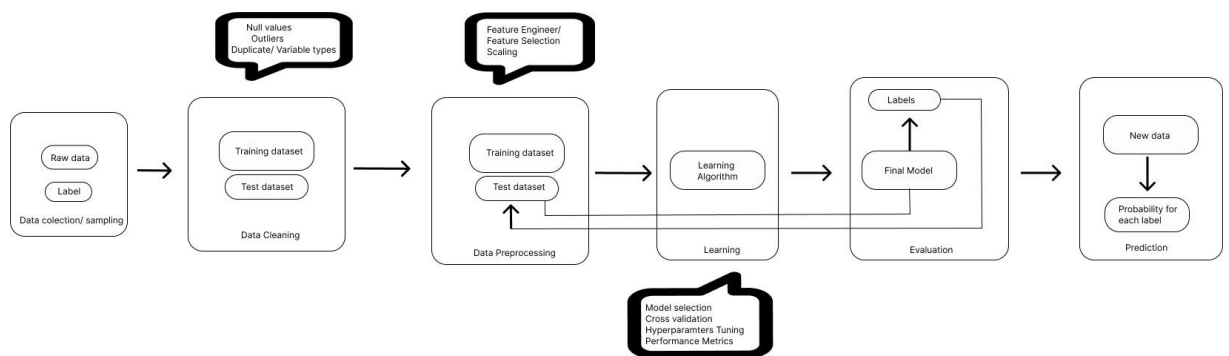


Figure 3-1: Machine learning operation

#### 3.1 DATA COLLECTION/ DATA SAMPLING

For building the ‘Suggestion system’, we need to collect data information about: age and gender, current expression or emotion, overall mood or emotional state over the past few days, temperature, income, beverage choices.

Utilizing both quantitative and qualitative surveys in data collection provides a comprehensive understanding of a subject by allowing for the statistical analysis of numerical data alongside the rich insights and contextual depth gained from open-ended responses, enhancing the overall robustness and depth of the collected information. In short, Quantitative data refers to numerical information that can be measured and expressed in terms of quantities, allowing for statistical analysis and objective comparisons, while qualitative data comprises non-numerical information, providing insights into the qualities, characteristics, and nuances of a subject, often obtained through open-ended responses or observations.

The type and choices for each element will be:



- Age: quantitative
- Gender: quantitative
  - Female
  - Male
- Expression: quantitative
  - Happy
  - Normal
- Overall mood: quantitative
  - Positive
  - Negative
- Current temperature: quantitative
- Income: quantitative
- Beverage choices; quantitative
  - Soft drink (Coca Cola, Pepsi,.....)
  - Energy drink (Red Bull, Monster Energy, Coffee,.... )
  - Alcoholic (Cocktail, .... )

We had take advantage of ‘**Google survey**’ to obtain information from the customers/ people.

## Beverage Choices Survey

Welcome to our beverage choices survey! We're excited to learn more about your preferences. Please take a moment to provide us with some quick information regarding your favorite beverage options. Your input will help us better understand and cater to your taste. Thank you for participating!

minhhien2771@gmail.com [Chuyển đổi tài khoản](#)

 Không được chia sẻ  Đã lưu bản nháp

\* Biểu thị câu hỏi bắt buộc

Are you male or female? \*

☐ Male

☐ Female

How old are you? \*

Câu trả lời của bạn

Are you happy or normal right now? \*

☐ Happy

☐ Normal

How would you describe your overall mood or emotional state over the past few days? \*

☐ Positive

☐ Negative

Can you share any specific positive or negative experiences or moments that stood out to you recently?

Câu trả lời của bạn

What is the current temperature like where you are? \*

Câu trả lời của bạn

Can you share the income you earn per month or the amount of money you have per month? (in VNĐ) \*

Câu trả lời của bạn

Which one will you choose? \*

☐ Soft drink (Coca Cola, Pepsi, ....)

☐ Energy drink (Red Bull, Monster Energy, Coffee, ...)

☒ Alcoholic (Cocktail, ....)

Figure 3-2: Survey form

Moreover, the most important thing to notice in data collection step is



choosing the appropriated data sampling technique. Accordingly, we have chosen the **‘Stratified sampling technique’** for our **‘Suggestion system’**.

In definition, **‘Stratified sampling’** involves dividing the population into distinct subgroups, or strata, based on certain characteristics, and then randomly selecting samples from each stratum in proportion to its representation in the overall population. This technique ensures a more accurate representation of diverse subgroups within the population, enhancing the precision and reliability of the study's results by capturing the variability present in each stratum, ultimately providing a more comprehensive and insightful understanding of the entire population.

In our problem, we have divided the whole populations (people) that we want to take survey on, into many subgroups. For example, one subgroup can be: Male\_Young age\_Positive mood\_Hot temperature\_Low income. Futhermore, stratifying users into subgroups based on their preferences, behaviors, or demographics allows the recommendation system to provide more accurate and personalized suggestions. By tailoring suggestions to specific customer segments, the system can better understand and cater to diverse preferences. Tailoring suggestions to specific subgroups increase the likelihood of users finding relevant content, leading to higher satisfaction and engagement. Users are more likely to interact with a system that understands their preferences and provides content aligned with their interests. Moreover, customers preferences can change over time or under different contexts. Stratification allows the recommendation system to adapt to these dynamics by continuously updating customer segments and refining suggestions based on evolving preferences.

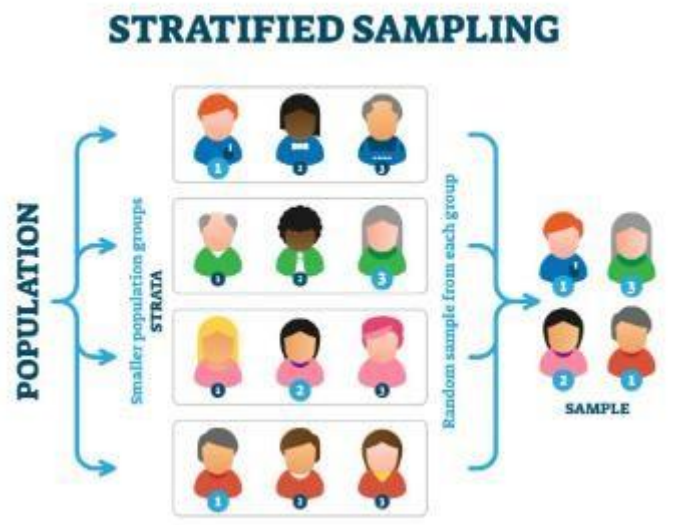


Figure 3-3: Stratified sampling

During the stage of data collectioning, many problems and obstacles has been arising. Firstly, respondents may be hesitant to disclose their age or gender due to privacy concerns or discomfort with providing personal information. Additionally, some individuals may fall outside the conventional binary gender categories, making the options limiting. Secondly, collecting temperature data through self-reporting relies on respondents' perceptions and might not reflect accurate measurements. Weather conditions or environmental factors may also influence respondents' perceptions of temperature, leading to variability in the reported data. Moreover, identifying individuals who fit into specific stratified groups can be challenging. The survey distribution may not reach a representative sample for each subgroup, leading to an imbalance in the data. Certain groups may be underrepresented, affecting the generalizability of findings. To mitigate the last problem, we must modify the 'Oversampling' technique that will appear later in our 'Data cleaning' stage.

### 3.2 DATA CLEANING

Our data format has been saved in 'csv' format and the first 5 rows of our data:

	Age	Gender	Emotion	Sentiment	Temperature	Income	Target
0	41	Male	Normal	Positive	34	22000000	Soft drink (Coca Cola, Pepsi, ...)
1	20	Female	Happy	Positive	32	3000000	Soft drink (Coca Cola, Pepsi, ...)
2	49	Female	Normal	Negative	29	34000000	Alcoholic (Cocktail, ...)
3	51	Female	Happy	Positive	27	36000000	Energy drink (Red Bull, Monster Energy, Coffee...)
4	15	Male	Happy	Positive	36	17000000	Energy drink (Red Bull, Monster Energy, Coffee...)

Figure 3-4: First 5 rows of our data

Firstly, the presence of null values, representing missing or undefined data points within a dataset, can have significant implications for the robustness and accuracy of machine learning models, as the absence of information in specific features may distort the learning process and compromise the model's ability to discern patterns, leading to biased predictions or reduced generalization performance; addressing null values through appropriate imputation or handling techniques is crucial to ensure the completeness and reliability of the dataset, thereby contributing to the overall effectiveness and integrity of the machine learning analysis. Luckily for using 'Google form' for our survey/ data collection, we force people to input all information before their submits, so that we won't have any null values in our data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6899 entries, 0 to 6898
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Age             6899 non-null   int64
1   Gender          6899 non-null   object
2   Emotion         6899 non-null   object
3   Sentiment       6899 non-null   object
4   Temperature     6899 non-null   int64
5   Income          6899 non-null   int64
6   Target         6899 non-null   object
dtypes: int64(3), object(4)
memory usage: 377.4+ KB
```

Figure 3-5: Information of features

Moreover, we also focus on type of our variables. Successfully, all these

variables's type has suited with the quantitative and qualitative format from our data collection stage.

Secondly, when engaging in the data cleaning process, a meticulous and comprehensive examination of unique values within each column is essential, as it enables the identification of outliers, discrepancies, or unexpected entries, thereby facilitating the development of informed strategies for data imputation, transformation, or removal, ultimately ensuring the integrity, consistency, and reliability of the dataset for subsequent analytical procedures and contributing to the overall efficacy of the data cleaning and preprocessing pipeline in preparation for machine learning or statistical analyses.

Before checking the outliers in our numeric variables, we had some analysis about the data distribution. Examining the data distribution for numeric columns prior to identifying outliers holds considerable advantages as it provides crucial insights into the inherent patterns, central tendencies, and variability within the dataset, enabling a more informed and nuanced approach to outlier detection; a comprehensive understanding of the distribution assists in setting appropriate thresholds or criteria for identifying anomalous values, distinguishing between natural variations and potentially influential outliers

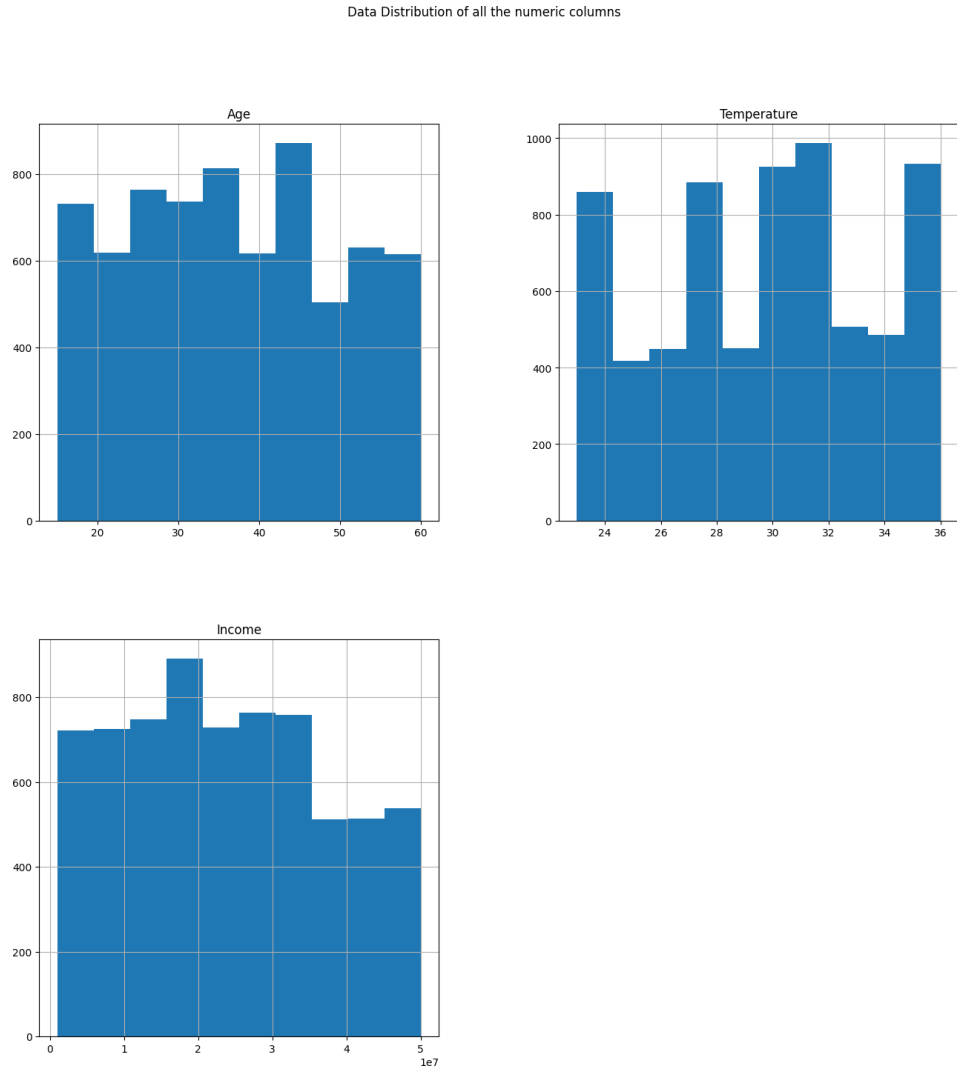


Figure 3-6: Data Distribution of all the numeric columns

Thirdly, detecting and removing outliers in our data is crucial as outliers, being extreme values, can disproportionately influence statistical analyses and machine learning models, leading to biased results, reduced model performance, or inaccurate insights; we have applied the Winsorization method, which, given a set of columns and a range defined by percentiles 'a' and 'b', identifies outliers and removes them from the training data, thereby enhancing the robustness of subsequent analyses by mitigating the impact of extreme values.

Lastly, our data target for suggestion is three type of beverage. Accordingly, it is essential for us to have the balance in the numbers of bevergae choices from

customers. Maintaining balance in the target variable, also known as class balance, is crucial for machine learning predictions as it ensures that the model is exposed to an equitable representation of each outcome or class within the dataset, thereby enhancing the model's ability to generalize and make accurate predictions across all classes; an imbalanced target variable, where one class significantly outnumbers others, can lead the model to exhibit biased behavior by favoring the majority class and potentially neglecting the minority classes, resulting in suboptimal performance, reduced predictive accuracy, and skewed evaluations, emphasizing the necessity of addressing class imbalance through techniques like oversampling, undersampling.

The last step for data cleaning is to apply one-hot encoding for our 'Target' columns for converting the string/ object format into numeric format (1,2,3). For the reason, numerical encoding can enhance the performance of certain machine learning models, especially those that rely on mathematical computations. It helps the algorithm understand the ordinal relationship between different classes, potentially leading to improved predictive accuracy. Moreover, some machine learning algorithms are more efficient and converge faster when dealing with numerical labels. Encoding the target column simplifies the training process, making it computationally more efficient and reducing the complexity of the model.

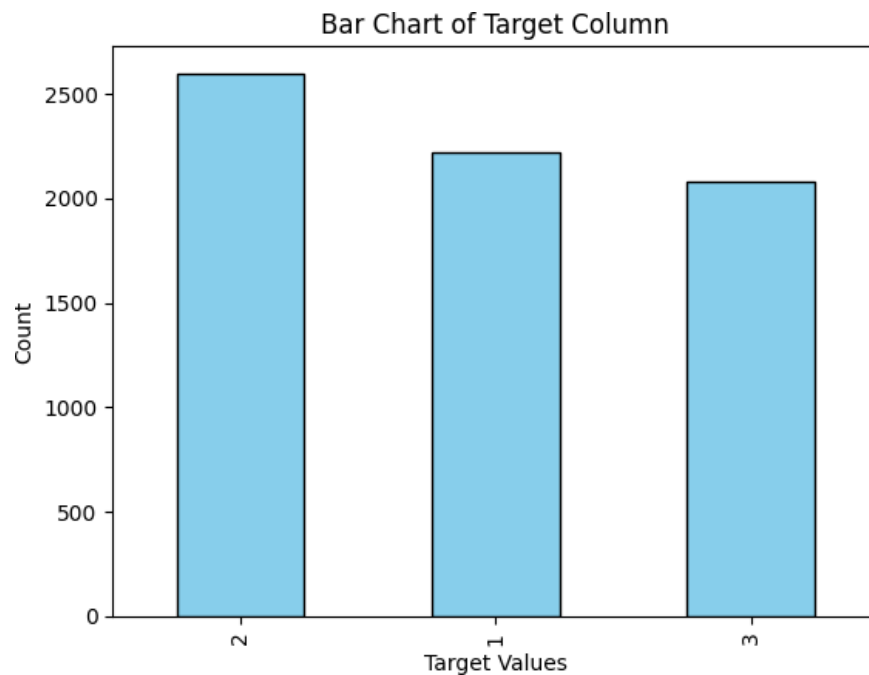


Figure 3-7: Bar Chart of Target Column

As the chart above shows that after applying the oversampling and undersampling for our data target column, the numbers of each element in column has been in balance form.

### 3.3 DATA PREPROCESSING

Data preprocessing, encompassing feature engineering and feature selection, holds paramount importance in the realm of machine learning and data analysis as it serves to enhance the quality, relevance, and efficiency of the input data, allowing for the extraction of meaningful patterns and insights; feature engineering enables the creation of new informative features or the transformation of existing ones, providing the model with richer information, while feature selection, by identifying and retaining the most impactful features, streamlines the model's learning process, mitigates the curse of dimensionality, and ultimately contributes to the development of more accurate, interpretable, and computationally efficient machine learning models.

We primarily modify and apply two libraries: Pandas and Sklearn for data preprocessing and further is for learning, choosing suitable machine learning model. The efficiency of leveraging the scikit-learn (sklearn) package and pandas in data preprocessing lies in their combined capabilities, where pandas facilitates seamless data manipulation, cleaning, and feature engineering through its versatile DataFrame functionalities, and scikit-learn streamlines preprocessing workflows by providing a comprehensive set of tools for tasks such as scaling, encoding, and feature selection.

Firstly, this is our first ten rows of our current data:

	Age	Gender	Emotion	Sentiment	Temperature	Income	Target
0	42	Female	Normal	Negative	26	32000000	2
1	38	Female	Happy	Positive	23	31000000	2
2	24	Male	Normal	Negative	29	14000000	3
3	43	Female	Happy	Positive	29	23000000	2
4	48	Female	Normal	Negative	26	44000000	2
5	19	Female	Normal	Negative	32	11000000	3
6	43	Male	Normal	Positive	32	24000000	1
7	37	Male	Happy	Positive	25	22000000	2
8	26	Female	Normal	Positive	27	7000000	2
9	31	Female	Normal	Negative	34	6000000	1
10	41	Male	Normal	Positive	30	30000000	2


Figure 3-8: First 10 rows of our data

The foremost technique that we have focused on in this stage is creating dummy variables techniques. The creation of dummy variables, a prevalent technique in data preprocessing, involves converting categorical variables into binary representations, providing an effective means to incorporate categorical information into machine learning models; this technique's benefits extend to enhancing model interpretability and performance, as it enables algorithms to comprehend and utilize categorical features by assigning binary values corresponding to distinct categories, thereby avoiding the imposition of erroneous ordinal relationships, preventing biased



interpretations, and fostering improved predictive accuracy and generalization, ultimately contributing to the efficacy and reliability of machine learning predictions.

Gender	
Female	
Male	
Male	
Female	



Gender	
1	
0	
0	
1	

Figure 3-9: Example of dummy variables

Essentially, we combine the dummy method with binning or discretization technique. To take note, binning or discretization is a technique in data preprocessing that involves categorizing continuous numerical data into discrete intervals or bins, assigning each data point to a specific range and facilitating the transformation of numeric features into categorical ones; the benefits of this technique for machine learning predictions include simplifying complex data distributions, reducing the impact of outliers, and enhancing the interpretability of models, ultimately aiding in the extraction of meaningful patterns and improving the performance and generalization of machine learning algorithms across various applications.

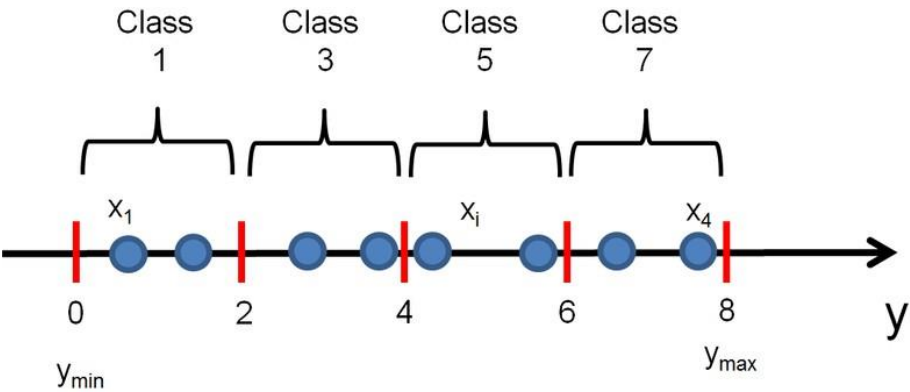


Figure 3-10: Example of binning

In our binning method for suggestion system, there are some threshold for our data:

- Age:
  - From 15 to 31: Young Adulthood
  - From 31 to 46: Early Adulthood
  - From 46 to 60: Middle Adulthood
- Temperature:
  - From 23 to 30: Cold
  - From 30 to 36: Hot
- Income:
  - From 1000000 to 17000000: Normal income
  - From 17000000 to 33000000: Middle\_Up Income
  - From 33000000 to 50000000: High\_Income

Go after the binning method is the create dummy method to convert categorical variables into binary representations. For example, our data will be like below after dummy and binning method:

Age_categories_Young Adulthood	Age_categories_Early Adulthood	Age_categories_Middle Adulthood
0	1	0
0	1	0
1	0	0
0	1	0
0	0	1

Figure 3-11: Example of variables after binning

Next step is our feature selection stage. The importance of feature selection in machine learning operations lies in its ability to enhance model efficiency, reduce computational complexity, and mitigate the risk of overfitting, by systematically

identifying and retaining the most informative features, thereby streamlining the learning process, improving model interpretability, and contributing to the development of more robust and accurate predictive models across diverse applications.

The first method we used in feature selection is ANOVA F-score. In short, ANOVA (Analysis of Variance) F-score, in conjunction with its associated p-value, serves as a powerful statistical tool for discerning the significance of feature columns concerning a target variable within a machine learning context; the F-score quantifies the variance between group means relative to within-group variance, generating a metric that gauges the impact of individual features on the target, and the accompanying p-value offers a statistical measure of the probability that these observed effects are due to random chance; by evaluating the F-score and comparing the p-value against a predetermined significance level, often denoted as alpha ( $\alpha$ ), feature columns exhibiting substantial variance with respect to the target can be identified. The more F -score one feature get, the less important that feature is. But the less p value one feature has, the more important that feature is meaning for predictions.

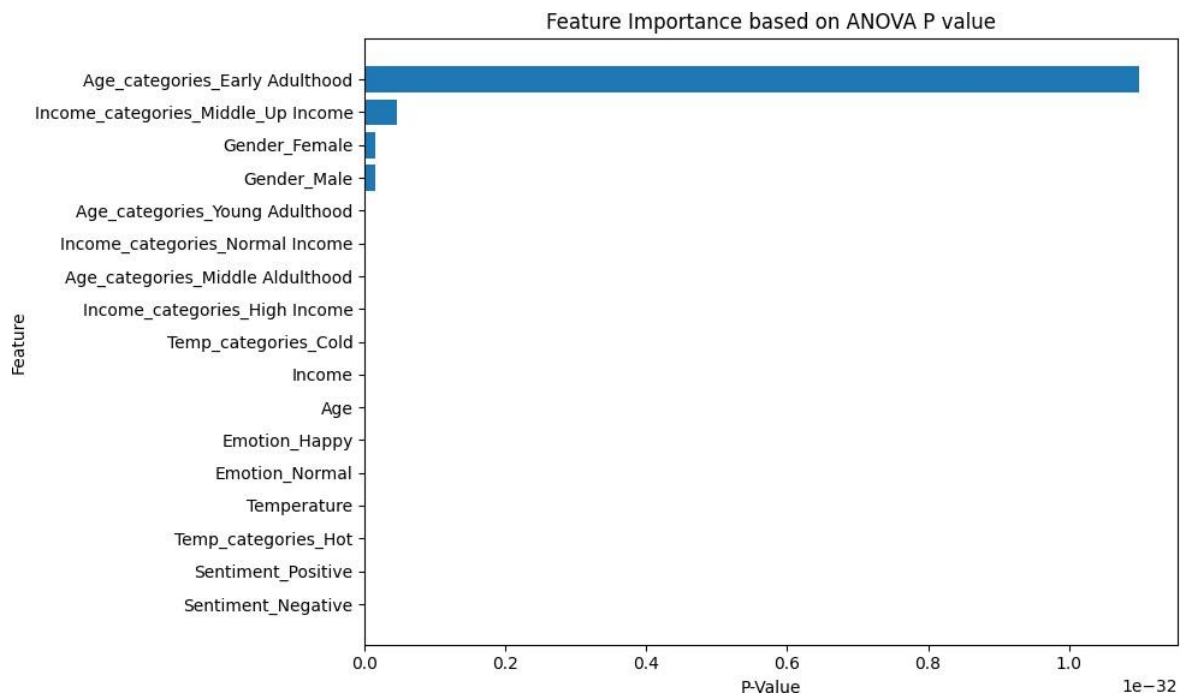


Figure 3-12: Feature Importance based on ANOVA P value

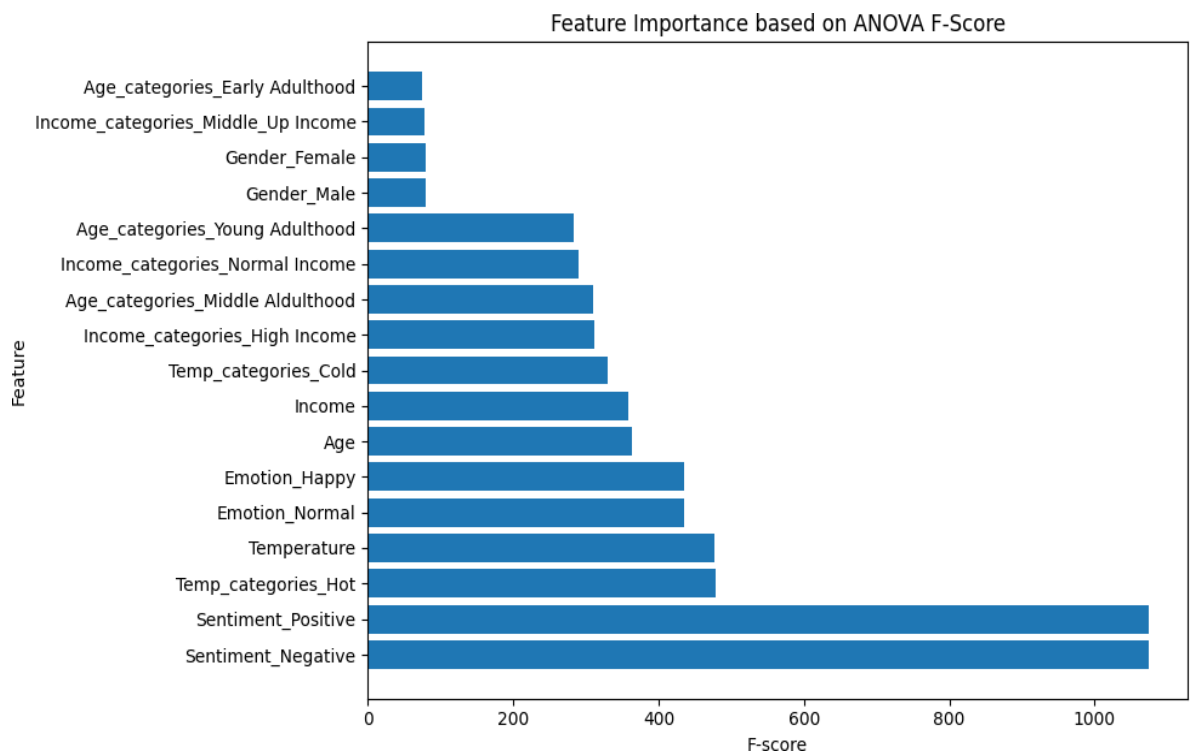


Figure 3-13: Feature Importance based on ANOVA F-Score

As for two chart above, it can be witnessed that the ‘Age\_categories\_EarlyAdulthood’ and both ‘Sentiment\_Positive’, ‘Sentiment-

Negative' are not extremely contribute for the prediction or suggestion for our beverage choices ( Target column). However, we didn't remove these columns but keep going on examining another selections method.

The second metric we used is the correlation between the features and target column. The effective use of correlation metrics in feature selection provides valuable insights into the relationships between different features, allowing data scientists to identify and retain the most relevant variables for predictive modeling; by measuring the strength and direction of linear associations between features and the target variable or between features themselves, correlation aids in selecting features that contribute unique information, avoiding multicollinearity issues, and ultimately enhancing the efficiency and interpretability of machine learning models through the inclusion of the most influential and informative features.

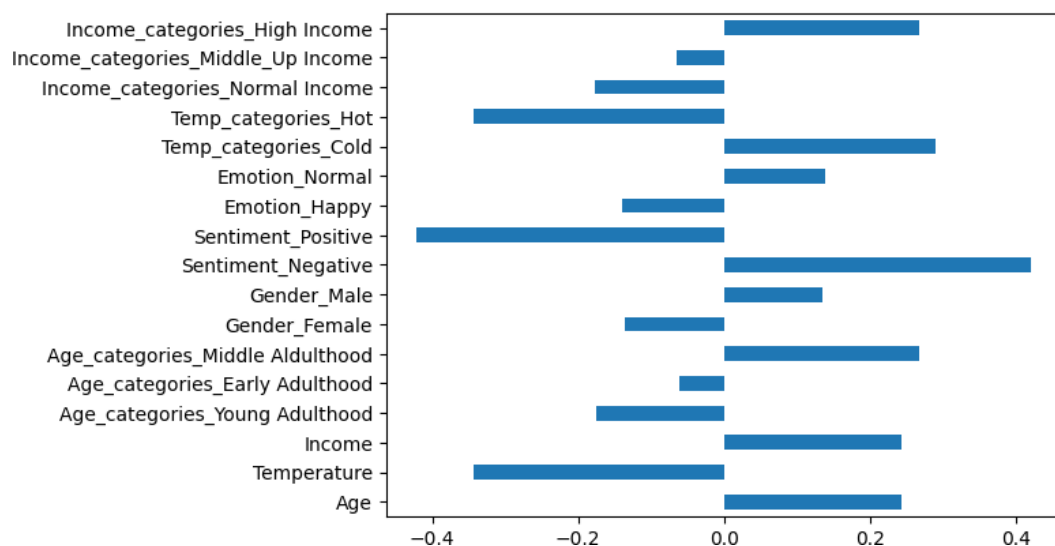


Figure 3-14: Correlation between features and target

As the correlation graph above, it can be observed that there is not any features that have very low correlation toward the target column.

Although we have small set of features, we must have to examine the multicollinearity between the features to understand the concept of “dummy variable trap”. In short, multicollinearity in a machine learning dataset refers to the high correlation

between two or more predictor variables, indicating a linear relationship among them; this condition poses challenges for predictive models as it can lead to inflated standard errors, making it difficult to discern the individual impact of correlated variables, and it may result in unstable coefficient estimates, diminishing the interpretability and reliability of the model. We have taken advantage of heatmap to determine the multicollinearity between features.

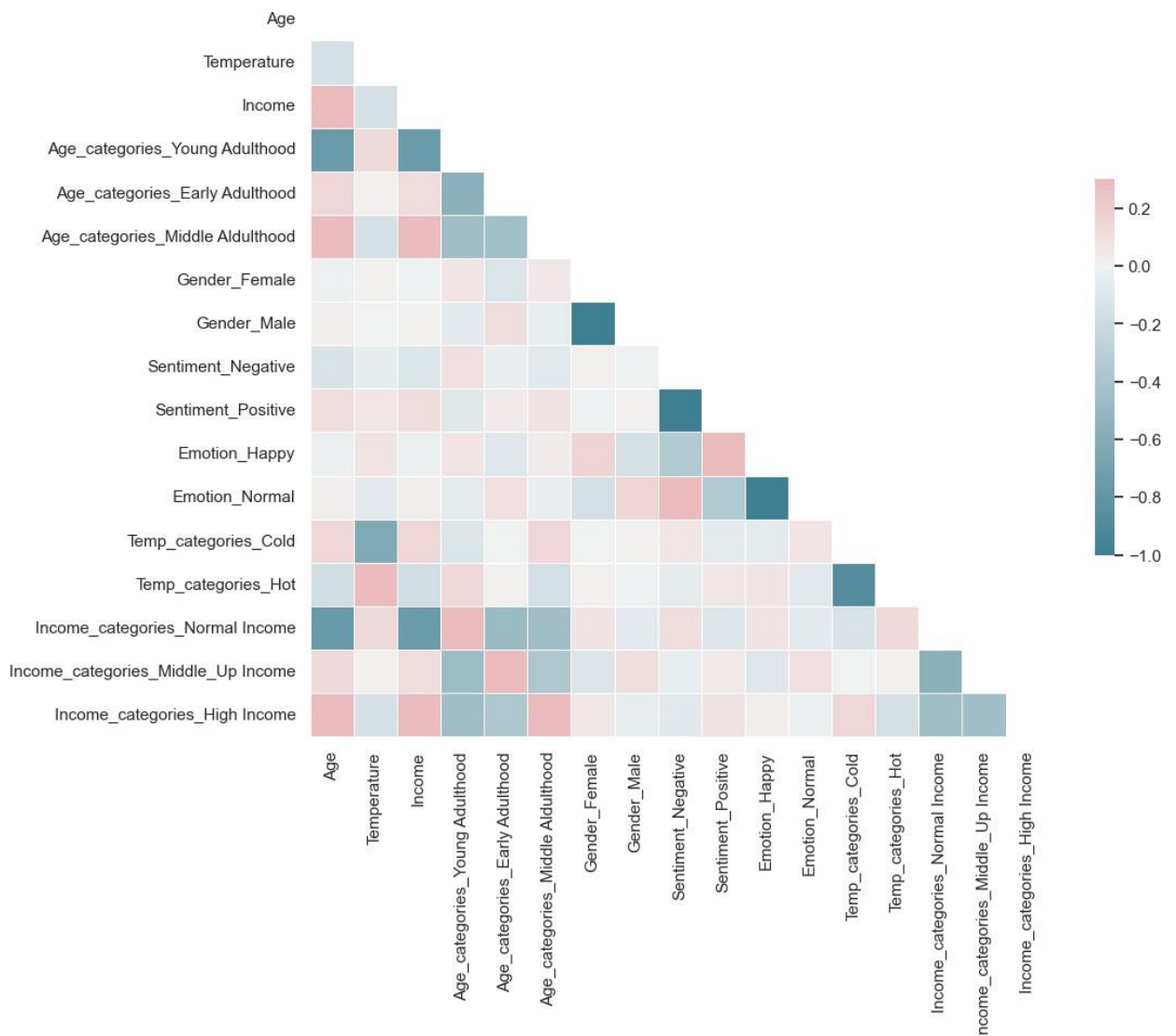


Figure 3-15: Heatmap for correlation

As the heatmap above, there are many correlations between the original columns and the dummy variables and among the dummy variables. For example, the 'Gender\_Female' always be in high correlation with 'Gender' and the 'Sentiment

Negative' have the high correlation with 'Sentiment Positive'. That explains why there are some columns that are not sufficient for our data which have been determined in 'ANOVA F-score'. Our solution here is to drop list of columns: 'Age', 'Temperature', 'Income', 'Target', 'Emotion\_Normal', 'Age\_categories\_Young Adulthood', 'Gender\_Male', 'Sentiment\_Negative', 'Temp\_categories\_Hot', 'Income\_categories\_High Income'. For dropping these columns, we have successfully eluded the dummy trap.

### **3.4 LEARNING AND MODEL EVALUATION**

#### **3.4.1 Model evaluation**

For choosing the most suitable machine learning model for our problem and examine the evaluation metric, we will firstly split the data into train and test dataset which we get from randomly split 30% size of train dataset. This method provides a rapid assessment of a model's performance by dividing the dataset into training and testing sets, allowing for a quick evaluation of how well the model generalizes to unseen data. However, if a model undergoes multiple adjustments based on its performance on the test set, there is a risk of overfitting to that specific test set, compromising its ability to generalize to new, unrelated data.

Our function will conduct a comparative evaluation of multiple classification models, including Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest; it calculates and stores precision, recall, and F1 score metrics for each model on the test data, subsequently organizing the results into a pandas DataFrame sorted by recall in descending order, offering a concise summary of the classification performance for the considered models.

The inclusion of Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest models in the `train_test_split` method serves the purpose of comprehensively evaluating the performance of diverse classification algorithms on the dataset; this approach allows for a comparative analysis of how different models handle the data, offering insights into their strengths, weaknesses, and suitability for

the specific task, thereby aiding in the informed selection of the most effective algorithm for the given problem during the model evaluation stage.

The result of this method will be:



	Recall	Precision	F1_Score
RandomForestClassifier	0.754106	0.754106	0.754106
DecisionTreeClassifier	0.751691	0.751691	0.751691
KNeighborsClassifier	0.745411	0.745411	0.745411
LogisticRegression	0.628502	0.628502	0.628502

Figure 3-16: Evaluation metrics for train\_test\_split method

From the result above, we decided to choose the Random Forest Classifier with the parameters: **n\_estimators=200** and **criterion="gini"** as our machine learning model for our problem.

However, the train\_test\_split method has some drawbacks:

- The split performed by train\_test\_split is random, meaning that different runs may yield different training and testing sets. This randomness can lead to variability in model performance evaluation.
- Depending on how the data is split, you may end up with a training set that doesn't represent the overall distribution of the data well. This can lead to overfitting or underfitting, especially if the dataset is not large.

Our grid search along with cross validation can help to solve the problems that come from the train\_test\_split method. Shortly, grid search is a technique used for hyperparameter tuning, which involves searching through a predefined set of hyperparameter values for a given machine learning model. The goal is to find the combination of hyperparameters that optimizes a specified performance metric.

Furthermore, Grid search allows you to search over a specified parameter grid, trying different combinations of hyperparameters. Moreover, cross-validation



ensures that the performance is evaluated across multiple folds, helping to find the optimal hyperparameters that generalize well to different subsets of the data. Moreover, cross-validation divides the data into multiple folds, and the model is trained and validated on different subsets in each iteration. This leads to better utilization of the available data for training and testing, reducing the risk of overfitting or underfitting due to a single split.

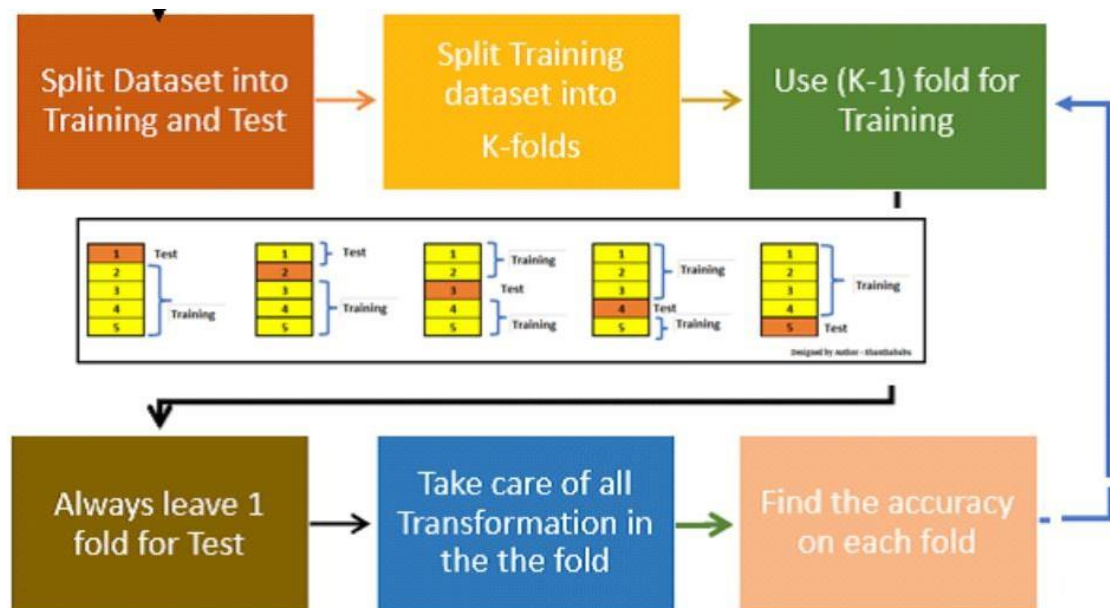


Figure 3-17: K - Fold Cross Validation

Accordingly, we will apply the grid search method with cross validation of 10 folds to indicate the best parameters for Random Forest Classifier model.

```

RandomForestClassifier
-----
Best Score: 0.7463429460886392
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 8, 'min_samples_split': 2, 'n_estimators': 6}
  
```

Figure 3-18: The result for best parameters for Random Forest Classifiers

The **RandomForestClassifier** achieved a best cross-validated accuracy score of nearly 75%, indicating its ability to generalize well to unseen data based on the chosen hyperparameters.

- Best Parameters:
  - **'criterion': 'entropy'**: The criterion for measuring the quality of a split in the decision tree is based on entropy, a measure of impurity.
  - **'max\_depth': 10'**: This parameter limits the maximum depth of each decision tree in the ensemble to 10 levels. Constraining the depth helps prevent overfitting by restricting the trees from becoming too complex and capturing noise in the training data. It promotes better generalization to unseen data.
  - **'max\_features': 'sqrt'**: The maximum number of features considered for splitting at each node is set to the square root of the total number of features. This choice promotes diversity among the decision trees in the ensemble, as each tree is trained on a different subset of features. It helps reduce correlation among the trees, which is beneficial for the overall model's performance.
  - **'min\_samples\_leaf': 8'**: This parameter sets the minimum number of samples required to be in a leaf node to 8. A higher value for this parameter contributes to the model's robustness by preventing the creation of small, noisy leaf nodes. It helps in producing more stable and reliable predictions.
  - **'min\_samples\_split': 2'**: The minimum number of samples required to split an internal node is set to 2, allowing for fine-grained decision-making.
  - **'n\_estimators': 6'**: the ensemble comprises 6 trees. Having more trees generally improves the model's performance, but it's crucial to strike a balance between complexity and computational efficiency. A smaller number of trees (6 in this case) is chosen to maintain a reasonable level of complexity while achieving good predictive performance.

This information collectively suggests that, based on the specified hyperparameters, the RandomForestClassifier is well-tailored to the characteristics of the dataset, demonstrating robust performance and achieving a commendable accuracy level of nearly 75%. If these hyper parameters are set to reasonable values,

it helps prevent the model from fitting the noise in the training data and encourages better generalization.

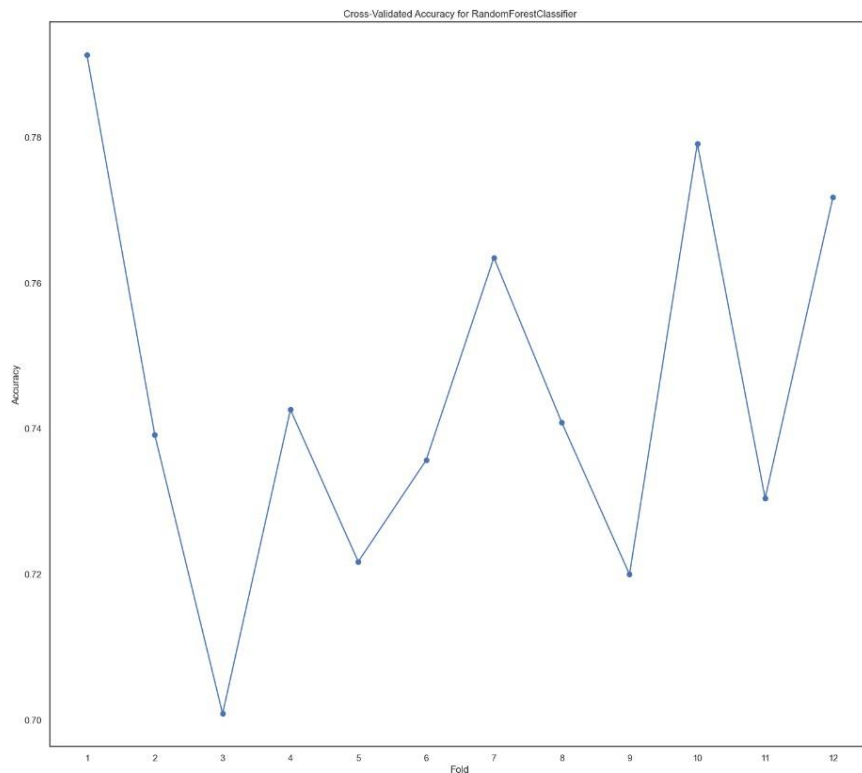


Figure 3-19: Test our best model accuracy in cross validation with 12 folds

Then, for making clear about the generalization in our model, we will apply the model with hyper-parameters above to test on the validation datasets with size of 600 that we have prepared separately.

	precision	recall	f1-score	support
1	0.81	0.77	0.79	184
2	0.76	0.67	0.71	226
3	0.72	0.85	0.78	190
accuracy			0.76	600
macro avg	0.76	0.77	0.76	600
weighted avg	0.76	0.76	0.76	600

Figure 3-20: The classification report

We have some observations from the precision, recall and f score:

- The model performs relatively well, with precision, recall, and F1-scores in the range of 0.71 to 0.81 for each class.
- Class 2 has the lowest recall (0.67), indicating that the model may not be as effective at capturing instances of this class.
- Class 3 has the highest recall (0.85), suggesting the model is better at identifying instances of this class.
- The macro and weighted averages indicate a balanced performance across classes.

If the precision, recall, and F1-scores are consistent across different classes, it suggests that the model is not overly tailored to specific patterns in the training data. Balanced performance often indicates improved generalization. Moreover, Recall measures the ability of the model to capture positive instances. Higher recall values across multiple classes suggest that the model is effective at identifying relevant patterns in the data, contributing to better generalization. Lastly, when the macro and weighted averages of precision, recall, and F1-score are close, it indicates that the model's performance is consistent across different classes and is not skewed by imbalances in the dataset. This can contribute to increased generalization.

Notably, the lowest recall in Class 2 (0.67) doesn't imply overfitting to specific patterns, but rather indicates a nuanced understanding of different class characteristics. Moreover, the model's exceptional recall of 0.85 in Class 3 showcases its proficiency in identifying instances of this class, substantiating its high generalization capabilities. The balanced performance, reflected in macro and weighted averages, reinforces the notion that the model doesn't favor specific patterns, emphasizing its resistance to overfitting. Furthermore, the higher recall values across multiple classes further confirm the model's adeptness at capturing relevant patterns, indicative of its superior generalization prowess. In conclusion, Consistency in precision, recall, and F1-scores across classes suggests a robust model

that generalizes effectively, minimizing the risk of being overly tailored to idiosyncrasies in the training data.

### **3.4.2 Trade-offs and Considerations:**

We've applied two different approaches to train a Random Forest model on your dataset: one with a basic train-test split and another with grid search and cross-validation.

About the model performance, the grid search and cross-validation approach yielded a model with better precision, recall, and F1-scores compared to the basic train-test split. This indicates that the hyperparameter tuning process helped the model generalize well to unseen data. Precision is important when false positives are costly, recall is crucial when false negatives are a concern, and F1-score balances both metrics. The improvement in these metrics suggests that the tuned model performs better across various aspects. Moreover, examining class-specific metrics is crucial. The second approach appears to provide a more balanced performance across different classes. It improved recall for class 2 and precision for class 3, indicating a better ability to correctly identify instances of these classes.

About the trade-off of training time, the basic train-test split is more time-efficient, suitable for quick model prototyping and testing. However, it may not explore the hyperparameter space thoroughly. While Grid search with cross-validation, while time-consuming, provides a more exhaustive search for optimal hyperparameters. This is beneficial for obtaining a well-tuned model. On the one hand, the basic train-test split approach is faster in terms of training time but may not fully explore the hyperparameter space, potentially leading to suboptimal results. On the other hand, Grid search with cross-validation takes more time but provides a more thorough exploration of hyperparameter combinations, potentially leading to better-tuned models.

The basic approach might be more prone to overfitting since it lacks the comprehensive search for hyperparameters. The second approach, with cross-

validation, helps mitigate overfitting by assessing the model's performance on multiple subsets of the data.

In conclusion, the choice between the two approaches can be iterative. Initially, a quick train-test split can be used for preliminary exploration. As the project progresses and more resources become available, a more thorough hyperparameter search can be conducted. The decision between the two approaches depends on the project's specific needs, time constraints, and available resources. It's a balance between efficiency and model performance.

## **4. DATA CAPTURE AND DEMO APPLICATIONS**

### **4.1 AGE AND GENDER DATA**

In the implemented face detection module, the code employs a Convolutional Neural Network (CNN) with the MobileNet architecture, configured to output confidence scores and bounding box coordinates for faces in the input video frames. The confidence threshold of 0.7 ensures that only highly confident detections are considered, reducing false positives.

Following face detection, the code proceeds to age and gender classification. The age classification utilizes a deep neural network with a prototxt file and a caffemodel file, while the gender classification utilizes a separate deep neural network with its corresponding prototxt and caffemodel files. The chosen architecture and model files contribute to accurate predictions.

The age and gender inference is performed on the facial regions defined by the bounding boxes. These regions undergo pre-processing in the form of blob conversion to match the input format expected by the age and gender networks. The resulting predictions are then mapped to predefined age groups and gender categories, creating a contextual understanding of the demographic attributes of the detected faces.

To enhance the user experience and facilitate interpretation, the code incorporates visual cues in the form of rectangular bounding boxes drawn around the

detected faces, along with dynamically updated text labels indicating the predicted gender and age group.

## 4.2 FACE EXPRESSION

We implement a real-time face expression detection system using computer vision and a pre-trained deep learning model. The system leverages the OpenCV library for video capture and facial recognition, and the Keras library to load a pre-trained convolutional neural network (CNN) model for emotion prediction.

Upon initializing the video capture, the code continuously processes frames from the camera feed. It resizes each frame to enhance processing efficiency. The Haar Cascade classifier is employed to detect faces in the video frames. Detected faces are enclosed by bounding boxes, providing a visual indication of the recognized facial regions.

For each identified face, a region of interest (ROI) is extracted from the grayscale version of the frame. The ROI is preprocessed by resizing it to the input dimensions expected by the emotion prediction model (48x48 pixels). The preprocessed image is then fed into the loaded CNN model for emotion prediction.

The emotion prediction yields a probability distribution over different emotion classes. The code extracts the emotion with the highest probability and associates it with one of the predefined emotion labels, such as 'Happy' or 'Normal' in this case. The detected emotion label is then displayed on the video frame along with its corresponding bounding box.

The implementation showcases a dynamic visualization of real-time emotion detection, enabling the user to observe and interpret the emotional expressions captured by the camera.

## 4.3 TEMPERATURE

We establish a connection with the OpenWeatherMap API to retrieve real-time temperature data for Ho Chi Minh City. Utilizing the 'requests' library, the code constructs a URL with the necessary parameters, including the API key and the city

name. The API call returns a JSON response, which is then parsed to extract the current temperature in Kelvin from the 'main' section.

To enhance user understanding and usability, a helper function 'kelvin\_to\_celsius\_fahrenheit' is employed to convert the temperature from Kelvin to Celsius and Fahrenheit. The calculated temperature values are then stored in the 'temp' and 'temp\_fahrenheit' variables.

This implementation provides a practical example of accessing external weather data through an API, enabling applications to dynamically capture and utilize temperature information

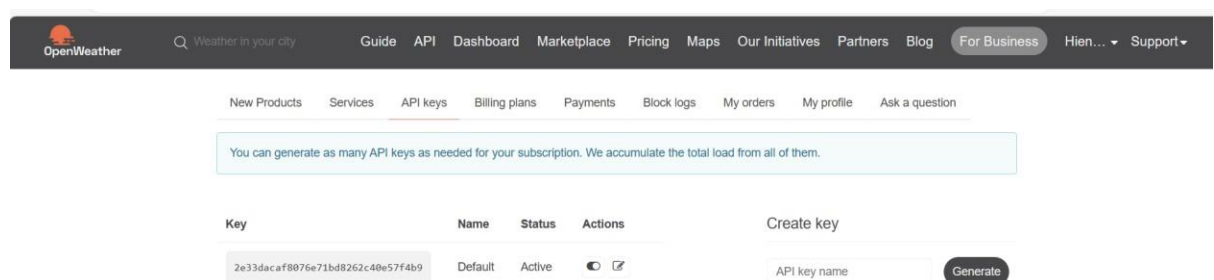


Figure 4-1: OpenWeather API web

## 4.4 OVERALL MOOD

We encapsulate a sentiment analysis workflow, incorporating audio recording, transcription, and sentiment classification. Initially, the code prompts the user to share thoughts about their day through an audio recording that spans a duration of 30 seconds. The 'sounddevice' library is employed to record the audio at a sampling frequency of 44,100 Hz.

Subsequently, the recorded audio is saved as a WAV file named 'Train.wav' using the 'wave' module. The 'upload' function is then called to obtain a URL for the uploaded audio file, and the 'save\_transcript' function is invoked to transcribe the audio, incorporating sentiment analysis.

The sentiment analysis results are stored in a JSON file named 'test\_sentiments.json'. The code proceeds to parse this file, categorizing the



transcribed text into positive, negative, and neutral sentiments based on the 'sentiment' key in the JSON data.

The sentiment ratios are calculated by counting the number of positive and negative sentiments. The overall sentiment ratio is determined by dividing the number of positive sentiments by the sum of positive and negative sentiments. If the resulting ratio is less than 50%, the overall sentiment is classified as 'Negative'; otherwise, it is labeled as 'Positive'.

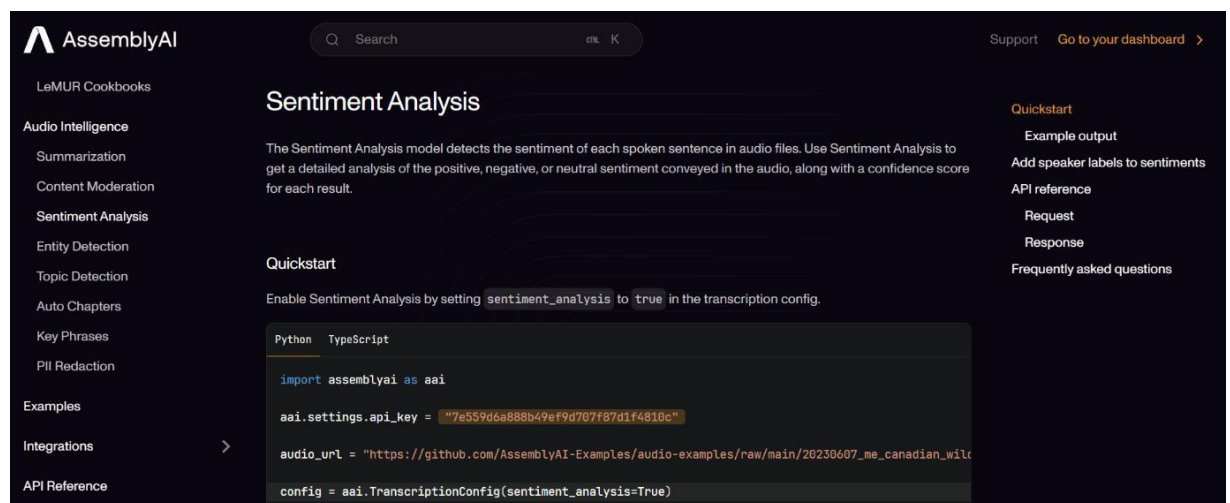


Figure 4-2: Assembly AI

## 4.5 MONEY

We allow users to manually input a numerical value representing a monetary amount. Upon execution, the user is prompted to enter a number, which is then converted to an integer using the 'input' function and the 'int' conversion. The code subsequently performs conditional checks on the entered amount.

If the entered amount is less than 1,000,000, the code sets it to a minimum value of 1,000,000. Similarly, if the entered amount exceeds 50,000,000, it is capped at a maximum value of 50,000,000. These conditional checks ensure that the entered monetary value falls within a specified range, preventing amounts below the minimum threshold or above the maximum limit.

## 4.6 APPLICATION

Our application contains 4 blocks: data capture, generate data frame, data preprocessing, generate suggestions probability.

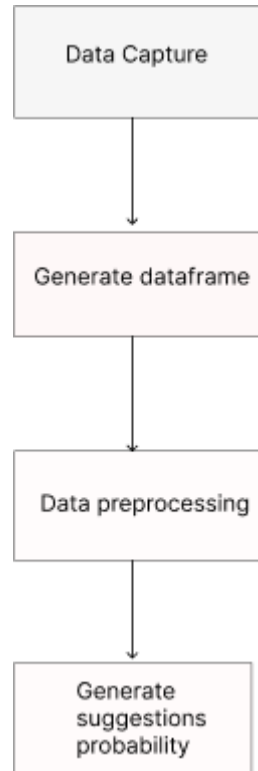


Figure 4-3:Application function block

The data capture block of the Beverage Suggestion System plays a pivotal role in gathering essential user information to personalize beverage recommendations. This block collects a diverse set of data points, including age, gender, facial expressions, sentiment analysis, and body temperature, creating a comprehensive profile for each user. The convergence of age, gender, facial expressions, sentiment, and temperature data forms a holistic user profile. This rich set of information empowers the Beverage Suggestion System to offer personalized and contextually relevant beverage recommendations, ensuring an enhanced user experience.

The data capture block is designed for continuous learning, allowing the system to adapt and refine its suggestions over time. As users interact with the system, the captured data contributes to an evolving understanding of individual preferences, leading to increasingly accurate and personalized beverage suggestions.

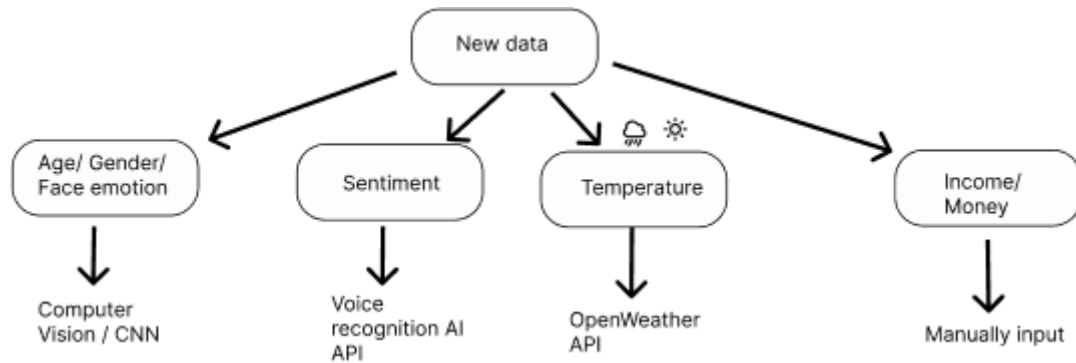


Figure 4-4: New data and its sources

The generation of a DataFrame through the "generate dataframe" block is instrumental in preparing the user data for seamless integration into the subsequent machine learning processes and models within the Beverage Suggestion System. By transforming disparate user attributes, including age, gender, temperature, emotional states, sentiment, and income, into a structured DataFrame format, the data becomes amenable to the intricacies of machine learning algorithms.

Furthermore, the use of a DataFrame enhances the interpretability and accessibility of the data, making it more conducive to feature engineering, a critical step in refining the dataset to enhance the predictive capabilities of machine learning models. The structured format also supports data preprocessing steps, such as create dummy or encoding categorical variables, which are essential for ensuring the robustness and effectiveness of the machine learning algorithms.

Next, the data preprocessing block plays a pivotal role in refining and optimizing the Data Frame created during the data capture stage, aligning it with the specific requirements of the machine learning models within the Beverage Suggestion System. This block encompasses a series of strategic transformations

aimed at enhancing the quality and relevance of the input data, thereby contributing to the overall effectiveness of the subsequent modeling processes. The functions within this block exemplify the creation of meaningful categorical variables from continuous features like age and temperature, income. This step, known as binning or discretization, not only simplifies the representation of these attributes but also captures underlying patterns more effectively, enabling the machine learning models to discern non-linear relationships and improve predictive accuracy.

Moreover, the creation of dummy variables through functions like `get_dummies` allows for the effective encoding of categorical features, such as 'Gender', 'Temp\_categories', 'Age\_categories', and 'Sentiment', into numerical representations. This transformation is crucial as many machine learning algorithms operate more efficiently with numerical input, ensuring that the system can leverage the full power of these models in generating beverage suggestions.

In essence, the data preprocessing block serves as a critical bridge between the raw user data captured initially and its incorporation into the machine learning models. By transforming, encoding, and refining the DataFrame, this block optimizes the input data to enhance the learning capabilities of the models.

Lastly, the "Generate Suggestions Probability" block represents a crucial stage in the Beverage Suggestion System, where machine learning models are utilized to predict probabilities associated with different beverage origins. This block employs a Random Forest classifier, a powerful ensemble learning algorithm, to provide nuanced and accurate predictions based on the features within the preprocessed DataFrame. The resulting DataFrame captures the probability scores associated with each beverage origin, serving as a valuable output for the system. These probabilities encapsulate the system's understanding of the likelihood of a user's preference for each beverage origin based on the input features.

In summary, the "Generate Suggestions Probability" block integrates sophisticated machine learning techniques, specifically Random Forest classifiers, to generate nuanced predictions of beverage origin probabilities.

Our homepage for application will have some introductions about the suggestion systems:

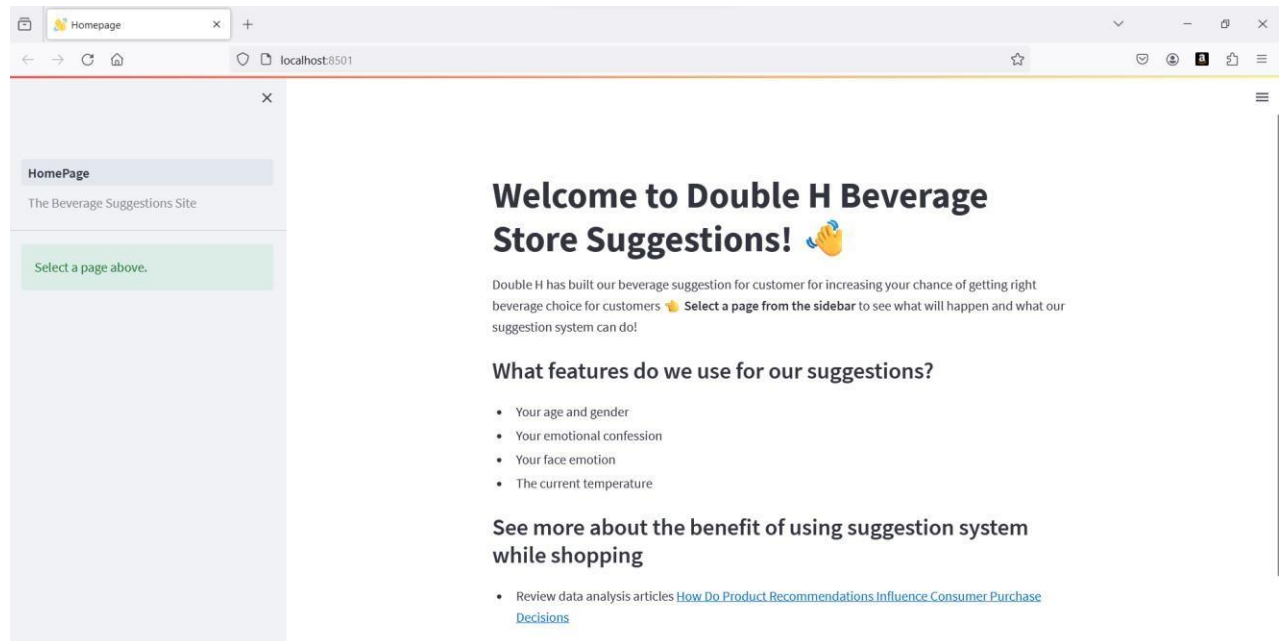


Figure 4-5: Home page

Then we go for “The Beverage Recommendation Site” in which the most suitable beverage products will be suggested to the users.

At first, the customer must input their income/ money:

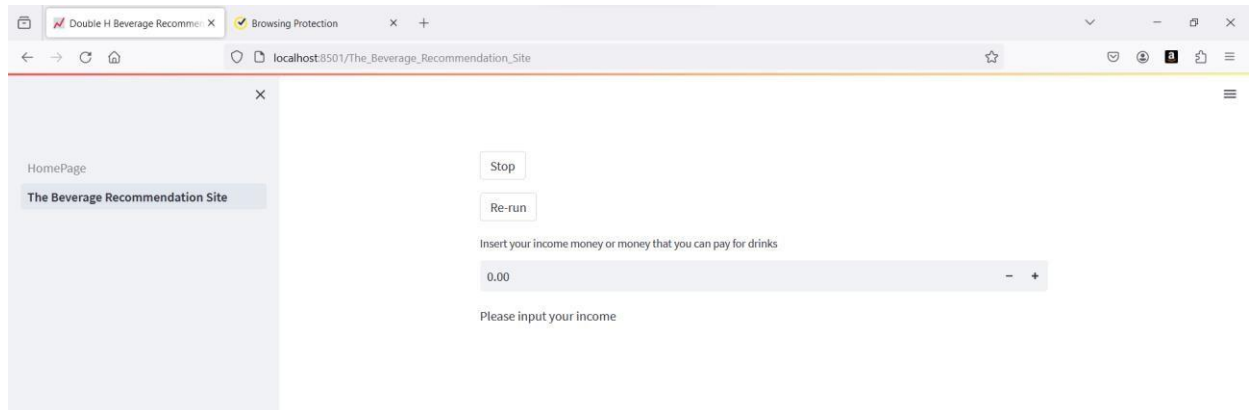


Figure 4-6: Place for money input

Then the customer can have 15 seconds to 30 seconds to address about their mood in past few days. The recording voice will be saved and go through sentiment analysis API to get the positive and negative in our mood.

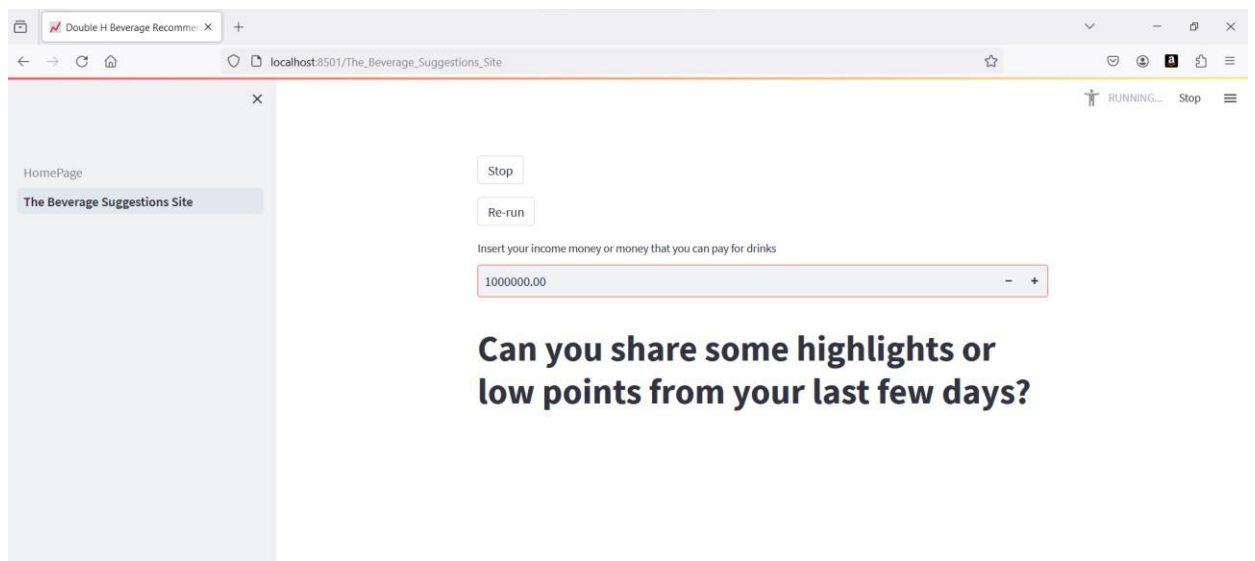


Figure 4-7: Capture overall mood from customer's sharing

After waiting for 30 seconds for our system to process the sentiment in customer's expression sharing, the system then captures user's gender and age.

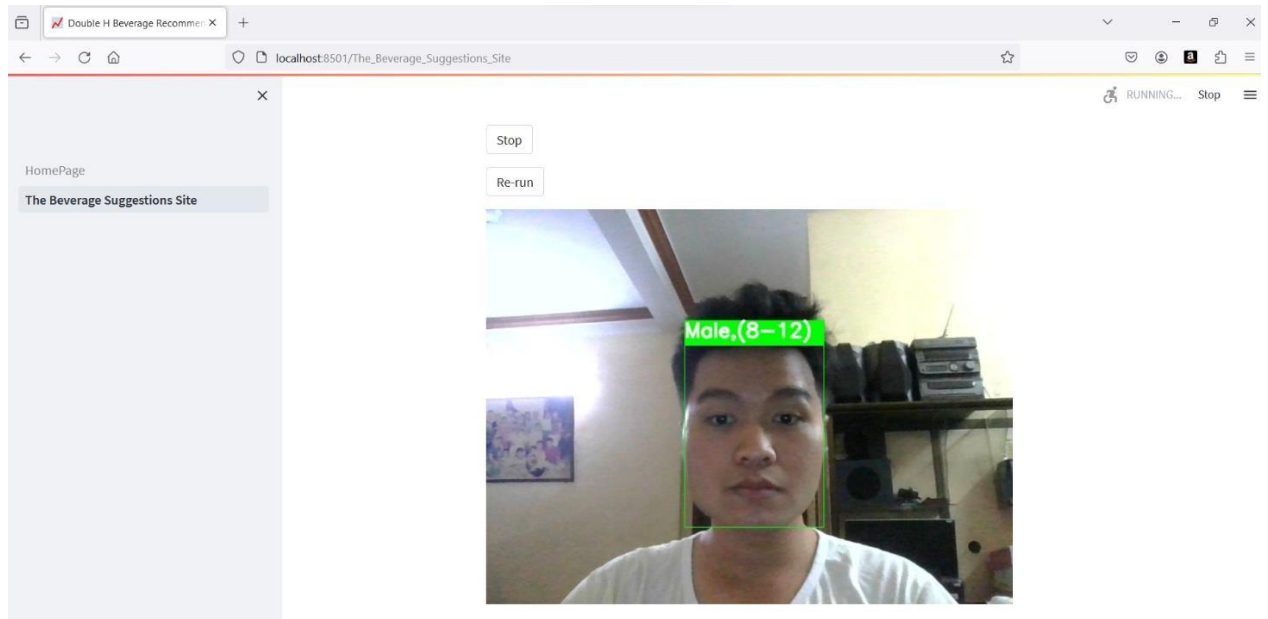


Figure 4-8: Capture Age and Gender

Next, system will capture customer's face expression.

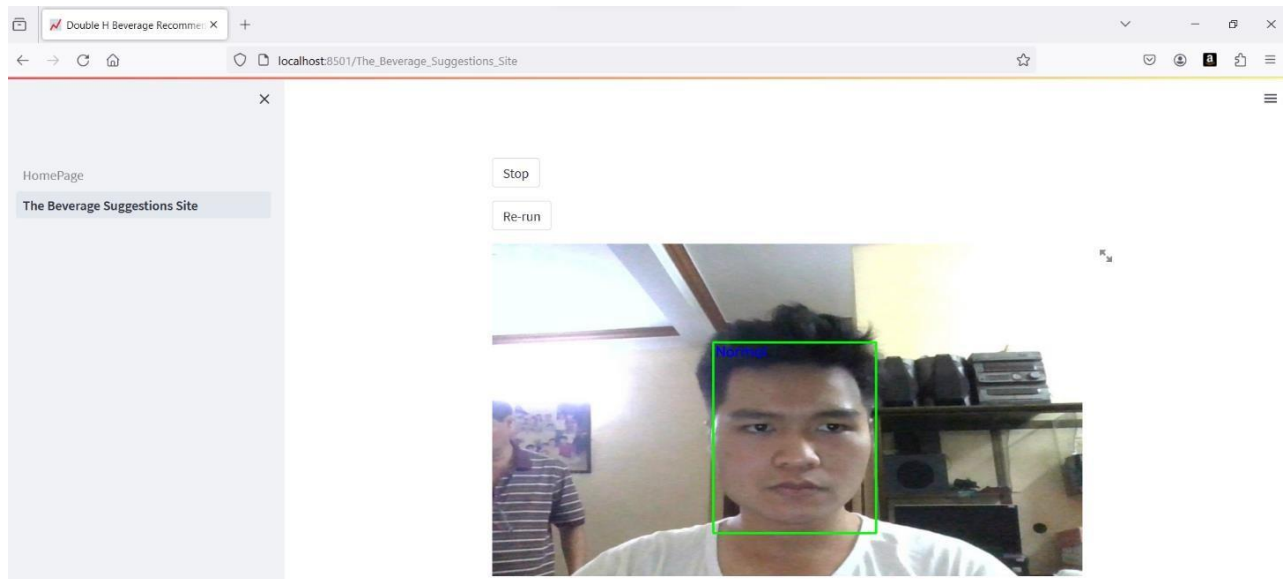


Figure 4-9: Face expression data capture

The temperature will be captured but not being showed on the frontend of our application.

Overall, after having all data to be processed by the machine learning methods and model, our system will output the suitable probability for customer in each beverage products.



Figure 4-10: Final result for beverage suggestions

We have obtained more result from the customers/ users using our system, here will be our 2 another example results with different features from customers.

There are 0% that Energy drink fit you, There are 29% that Soft drink fit you, There are 69% that Alcoholic fit you

Figure 4-11: Another customer result

There are 31% that Energy drink fit you, There are 64% that Soft drink fit you, There are 0% that Alcoholic fit you

Figure 4-12: Another result for customer



The analysis of the three pictures depicting various results strongly suggests that substantial variations or subtle discrepancies exist among the outcomes for customers with distinct features. This observation underscores the importance of implementing robust machine learning operations to discern and respond effectively to these differences. Fortunately, our machine learning endeavors have proven successful in adopting an appropriate data sampling method. This strategic approach aids our model in identifying and understanding intricate patterns inherent in customers' features.

The careful selection of a suitable model plays a pivotal role in mitigating bias in the final results generated by our application. By utilizing an unbiased model, we ensure that our application delivers personalized beverage choices for customers without unfair inclinations. This not only enhances the overall user experience but also fosters a sense of trust and satisfaction among our clientele.

Furthermore, the observed variance in results based on different customer features indicates a high level of generalization in our model. Generalization is a critical aspect of machine learning, demonstrating the model's ability to make accurate predictions and decisions on new, unseen data. In the context of beverage choices, this implies that our model is adept at accommodating diverse customer preferences and characteristics, thereby providing a comprehensive and tailored experience for each individual.

## **5. CONCLUSION AND OUTLOOK**

### **5.1 CONCLUSION**

This thesis introduces an innovative "Beverage Suggestion System" driven by the application of various machine learning operations. The core of the system involves the careful selection of a machine learning model tailored to the specific requirements and applications of the beverage recommendation problem. A comprehensive approach is taken, incorporating data sampling methods and

leveraging insights obtained through a survey conducted on the Google platform. These efforts contribute to enhancing the accuracy of predicting the most suitable beverage products for customers while uncovering valuable patterns within the dataset.

We successfully implementing the proposed beverage suggestion system involved the integration of cutting-edge technologies. The inclusion of computer vision with convolutional layers enabled the system to process visual data efficiently. Sentiment analysis, coupled with voice recognition APIs, added a nuanced layer by understanding user preferences based on their expressed sentiments. Additionally, the integration of an API for capturing real-time temperature data provided contextual information to refine beverage recommendations.

In conclusion, the Beverage Suggestion System presented in this thesis showcases the successful integration of machine learning operations, data sampling methodologies, and cutting-edge technologies such as computer vision, sentiment analysis, and real-time temperature capture. By meticulously selecting and optimizing machine learning models, the system demonstrates improved accuracy in predicting customer preferences, contributing to the advancement of data science approaches in the domain of beverage recommendation systems.

## 5.2 OUTLOOK

With the continuous development of artificial intelligence and data science technology from computer vision field to natural language processing, the suggestion system has demonstrated significant flexibility and potential. In the future, there are several directions for further development that we can consider enhancing and expand the system:

- Increase the numbers of features: Currently, the system leverages data related to age, gender, and basic preferences. To enhance accuracy, expanding the feature set to include more nuanced aspects such as real-time information on frequently purchased products, preferred colors, and other relevant details

could contribute to a more fine-tuned and personalized recommendation process. By incorporating these additional features, the system can better adapt to the evolving and diverse preferences of individual customers.

- Elevate the system's predictive capabilities: rather than relying solely on explicit user input, the system could benefit from a real-time behavioral analysis framework. This entails capturing data seamlessly and unobtrusively, potentially using cameras in beverage shops or discreet listening devices. By analyzing customer behavior in a non-intrusive manner, the system gains a deeper understanding of preferences, potentially uncovering patterns that users may not explicitly express. This shift towards covert data capture aligns with the need for a more natural and integrated user experience, ultimately leading to increased reliability in the system's output.
- Further enhance the richness of data: this involves combining information from various sensors and devices, such as cameras, microphones, and perhaps even wearable devices. By amalgamating data from different modalities, the system gains a holistic view of the user's environment, preferences, and interactions. This approach goes beyond traditional data capture methods, providing a more comprehensive and context-aware understanding of the user's preferences.

## 6. REFERENCES

- [1] *Age and Gender Classification Using Convolutional Neural Networks*. (2015, June 1). Tal Hassner. [https://talhassner.github.io/home/publication/2015\\_CVPR](https://talhassner.github.io/home/publication/2015_CVPR)
- [2] Mai, D. S., & Khue, D. B. (2016, May 19). *The Trend to Use Beverages Based on Age, Gender, Job, Income and Location of Consumers*. ResearchGate. [https://www.researchgate.net/publication/341480318\\_The\\_Trend\\_to\\_Use\\_Beverages\\_Based\\_on\\_Age\\_Gender\\_Job\\_Income\\_and\\_Location\\_of\\_Consumers](https://www.researchgate.net/publication/341480318_The_Trend_to_Use_Beverages_Based_on_Age_Gender_Job_Income_and_Location_of_Consumers)
- [3] Mallick, S., & Mallick, S. (2023, September 12). *Blob Detection Using OpenCV (Python, C++)* /. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Examples and Tutorials. <https://learnopencv.com/blob-detection-using-opencv-python-c/>

- [4] *FER-2013*. (2020, July 19). Kaggle. <https://www.kaggle.com/datasets/msambare/fer2013>
- [5] Rosebrock, A. (2023, June 8). *Convolutional Neural Networks (CNNs) and Layer Types* - PyImageSearch. PyImageSearch. <https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/>
- [6] *Machine Learning Operations (MLOps): Overview, Definition, and Architecture*. (2023). IEEE Journals & Magazine | IEEE Xplore. <https://ieeexplore.ieee.org/document/10081336>
- [7] *Papers with Code - FER2013 Benchmark (Facial Expression Recognition (FER))*. (n.d.). <https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013>
- [8] Bevans, R. (2023, June 22). *One-way ANOVA | When and How to Use It (With Examples)*. Scribbr. <https://www.scribbr.com/statistics/one-way-anova/>
- [9] Gupta, A. (2023, April 26). *Feature Selection Techniques in Machine Learning (Updated 2023)*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/>
- [10] Hayes, A. (2023, November 6). *How Stratified Random Sampling Works, with Examples*. Investopedia. [https://www.investopedia.com/terms/stratified\\_random\\_sampling.asp](https://www.investopedia.com/terms/stratified_random_sampling.asp)
- [11] Shah, R. (2023, October 6). *Tune Hyperparameters with GridSearchCV*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/>
- [12] S. (n.d.). *GitHub - streamlit/streamlit: Streamlit — A faster way to build and share data apps*. GitHub. <https://github.com/streamlit/streamlit>
- [13] F. (2022, October 25). *Updates, Insights, and News from FutureLearn | Online Learning for You*. FutureLearn. <https://www.futurelearn.com/info/courses/applied-data-science/0/steps/169276#:~:text=Data%20cleaning%20>
- [14] David, D. (2020, August 13). *Random Forest Classifier Tutorial: How to Use Tree-Based Algorithms for Machine Learning*. freeCodeCamp.org. <https://www.freecodecamp.org/news/how-to-use-the-tree-based-algorithm-for-machine-learning/>
- [15] Agrawal, S. (2023, June 4). *Hyperparameter Tuning of KNN Classifier* - Saurav Agrawal - Medium. Medium. <https://medium.com/@agrawalsam1997/hyperparameter-tuning-of-knn-classifier-a32f31af25c7>

- [16] *Confusion Matrix in Machine Learning*. (2023, March 21). GeeksforGeeks. <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>
- [17] *Logistic Regression: Equation, Assumptions, Types, and Best Practices*. (2022, April 18). Spiceworks. <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/>
- [18] Yasar, K., & Biscobing, J. (2023, May 2). *data sampling*. Business Analytics. <https://www.techtarget.com/searchbusinessanalytics/definition/data-sampling>
- [19] *ML Overview of Data Cleaning*. (2023, June 10). GeeksforGeeks. <https://www.geeksforgeeks.org/data-cleansing-introduction/>
- [20] Ghosh, S. (2023, August 22). *A Comprehensive Guide to Data Preprocessing*. neptune.ai. <https://neptune.ai/blog/data-preprocessing-guide>