Introduction

Pattern classification and recommender systems are important applications in data science in both small data and Big Data context. And they could be applied to many real-world problems.

The goal of pattern classification is to classify unseen samples into one or more predefined classes. It is a very important and challenging problem to build a classification model on raw data. The raw data can be unstructured, dirty, and redundant, posing a huge challenge on the classification performance.

Recommender systems, on the other hand, aim to recommend items or content to users based on their past behaviour and preferences. Recommender systems are widely used in e-commerce, entertainment, social media, and many other domains. The goal of a recommender system is to provide users with personalized recommendations that are relevant and useful to them.

Both pattern classification and recommender systems rely on machine learning algorithms to extract patterns and insights from data. To build effective models, it is important to carefully select and pre-process the data, choose appropriate feature representations, and select the right machine learning algorithms and hyperparameters.

Despite these challenges, pattern classification and recommender systems have proven to be powerful tools for solving a wide range of real-world problems. From predicting customer churn to recommending movies, these techniques have the potential to transform industries and improve people's lives.

# Understanding of the given problems:

This report addresses two distinct tasks: the first task is to apply pattern classification to develop a Face Mask Detection system using two datasets, namely the WIDER Face dataset and the MAFA dataset. The WIDER Face dataset was used to detect faces in an input image, while the MAFA dataset was used to classify whether a face is wearing a mask or not. The second task involves developing a content-based movie recommender system using the TMDB5000 Movies Dataset.

In the first task, the goal is to develop a machine/deep learning model that can detect whether an individual in an image is wearing a mask or not. The WIDER Face dataset contains images with annotated bounding boxes, which were used to train a face detection model. This model was integrated with the MAFA dataset, which contains information about the degree and type of occlusion in the images. The degree of occlusion, ranging from 1 to 3, indicates the extent to which the face is occluded in the image. The occlusion type, on the other hand, provides more detailed information about the occlusion, including whether it is a simple mask, a complex mask (such as a chemist's mask or a dark mask), or whether the face is occluded by hand., The Face Mask Detection system then combines these two models to determine whether an individual is wearing a mask or not based on an input image.

The second task involves developing a content-based movie recommender system using the TMDB5000 Movies Dataset. The goal is to recommend movies based on the similarity between their content. This dataset contains information on movies such as genre, cast, crew, and movie description. The recommender system uses this information to identify movies with similar content and recommends them to the user.

Problem decomposition into subtasks:Top of Form

The first task can be decomposed into the following subtasks:

1. **Data acquisition**: Collecting and preparing the required datasets.
2. **Data pre-processing:** Pre-processing the datasets to prepare them for training.
3. **Face detection**: Using the WIDER Face dataset to train a model that can detect faces in an image.
4. **Face cropping**: Once a face is detected, it needs to be cropped from the frame.
5. **Mask detection**: Using the MAFA dataset to train a model that can determine if a person is wearing a mask or not.
6. **Integration:** Integrating the face detection and mask detection models to create a Face Mask Detection system.
7. **Building Real-Time application:** Building an application that utilizes a camera to detect faces in front of it and classify whether they are wearing a mask or not.

The second task, developing a content-based movie recommender system, can be decomposed into the following subtasks:

1. **Data acquisition:** Collecting and preparing the required dataset, in this case, the TMDB5000 Movies Dataset.
2. **Data pre-processing:** Pre-processing the dataset to prepare it for training.
3. **Feature extraction:** Extracting features from the dataset, such as movie genre, cast, crew, budget, and revenue, to create a representation of the movie.
4. **Similarity calculation:** Calculating the similarity between different movies based on their extracted features.
5. **Recommendation generation:** Generating a list of recommended movies based on the user's movie preferences and the similarity between movies.
6. **Building real-time application:** Building an application that allows the user to input their movie and receive real-time movie recommendations based on their input.

# Literature review:

**Face Mask Detection**:  
Face mask Detection has gained significant attention during the COVID-19 pandemic due to the widespread use of face masks as a preventive measure. Various methods have been proposed for Face Mask Detection using machine/deep learning. This report discusses two papers related to face detection and recognition in unconstrained scenarios.

The first paper titled **“Detecting Masked Faces in the Wild with LLE-CNNs”** (Shiming Ge J. L., n.d.)proposes a deep learning approach to detecting masked faces using a combination of locally linear embedding (LLE) and convolutional neural networks (CNNs). The authors of this paper also introduce a new dataset called the MAFA dataset, which contains labeled images of faces wearing and not wearing masks. The proposed approach achieves high accuracy in detecting masked faces in unconstrained scenarios.

The second paper titled **“Face Attention Network: An Effective Face Detector for the Occluded Faces”** (Jianfeng Wang, n.d.)presents a novel face detector called the Face Attention Network (FAN), which significantly improves the recall of the face detection problem in the occluded case without compromising speed. The paper also uses both the WiderFace and MAFA datasets to evaluate the proposed approach. The results show that the FAN approach achieves high accuracy in detecting occluded faces, including those with masks.

The third paper is titled **“FaceBoxes: A CPU Real-time Face Detector with High Accuracy”** (Shifeng Zhang, n.d.)This paper presents a model trained on WiderFace dataset to detect faces, FaceBoxes is a deep learning model developed for real-time face detection with high accuracy. It is designed to run on a CPU, which makes it a highly efficient and accessible solution for many applications.

Overall, these three papers propose innovative approaches to detect and recognize faces in unconstrained scenarios, including occluded faces with masks. The MAFA dataset introduced by the authors of the first paper has become a valuable resource for researchers in the field of face detection and recognition. The proposed approaches in these papers show promising results and have potential applications in various fields, including security, public health, and entertainment.

Innovative thinking after the review:

**Face Mask Detection:**After conducting a literature review and analysing the available datasets, some innovative ideas were identified to improve the Face Mask Detection system.

1. **Using FaceBoxes pretrained model**: which is a highly accurate and efficient face detection model trained on the WiderFace dataset. This model was implemented to detect faces in the input images.
2. **Fine-tuning a model to classify images using the TensorFlow framework:** using the MAFA dataset to train a model for detecting face masks. This allowed for the development of a more robust and accurate Face Mask Detection system.

By incorporating these innovative ideas into the project, the accuracy and efficiency of the Face Mask Detection system were significantly improved. The use of pre-trained models and fine-tuning techniques helped to reduce the amount of training data required and resulted in faster model convergence.

**Recommender System:**Based on the literature review, the following innovative ideas were generated for building a content-based movie recommender system using the TMDB5000 Movies Dataset:

1. **Utilizing TfidfVectorizer:** One potential approach to extract important keywords from the movie summaries is to use TfidfVectorizer. By converting each summary into a vector representation of its most important terms, this method can identify similarities between different movie summaries and aid in recommending similar movies to the user.
2. **Recommending based on movie summary:** A system could be created that recommends movies based on the similarity between their summaries. This system would allow users to input a summary or keywords related to a movie they enjoyed and receive recommendations of similar movies.
3. **Recommending based on genres, keywords, and cast:** Another system could be developed that recommends movies based on specific features such as genre, keywords, and cast. This system could take into account the user's past viewing habits and preferences to provide personalized recommendations.
4. **Combining both systems:** By combining both the summary-based and feature-based systems, a comprehensive movie recommender system could be developed that accounts for multiple factors in recommending movies. This could provide a more accurate and personalized recommendation to the user.

Face Mask Detection System

# Data Acquisition and Description

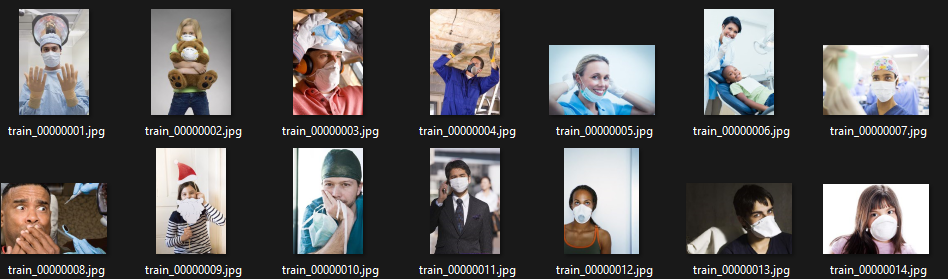
Our system relies on two essential datasets to perform its tasks: the WIDER FACE dataset and the MAFA dataset.

**The WIDER FACE dataset**: (Yang, n.d.) is a widely used benchmark dataset for face detection. It contains a diverse range of images, selected from the publicly available WIDER dataset, with a total of 32,203 images and 393,703 labelled faces. These faces exhibit high variability in terms of scale, pose, and occlusion, as shown in sample images.



To detect faces from images in our system, we will use a highly accurate pre-trained model called FaceBoxes, which was trained on the WIDER FACE dataset. This pre-trained model can be downloaded from <https://github.com/sfzhang15/FaceBoxes>.

**The MAFA dataset**: (Shiming Ge J. L., n.d.) on the other hand, contains 30,811 internet images with 35,806 masked faces. The faces in this dataset exhibit varying degrees of orientation and occlusion, with at least one part of each face occluded by a mask. The MAFA dataset is crucial for our system as it allows us to detect whether a face wears a mask or not.



The Dataset contains 4 Directories:

1. train images: a directory that contains all the trained images.
2. test images: a directory that contains all the test images.
3. MAFA-Label-Train: contains the .mat file of the train dataset.
4. MAFA-Label-Test: contains the .mat file of the test dataset.

The MAFA dataset contains information about:

1. Bounding box of a face
2. Position of two eyes.
3. Bounding box of the occlude.
4. Occlude type: 1 for simple, 2 for complex and 3 for human body.
5. Occlude Degree stands for the number of occluded face parts
6. Gender and race stand for the gender and race of one face.
7. Orientation stands for the face orientation/pose, and has: 1-left, 2-left frontal, 3-frontal, 4-right frontal, 5-right.
8. Bounding box of the glasses and is set to (-1,-1,-1,-1) when no glasses.

# Reading the MAFA Dataset

In this section of our code, we are working with the MAFA dataset and performing several essential tasks to prepare the data for our model.

We have defined three functions that will help us to read and parse the data from the train mat file and create the required dataframe.

Text

Description automatically generated

There are 3 functions:

1. **load\_train\_data**, reads the train mat file and parses it into two lists - images and labels. This function is essential to load the training data from the dataset and prepare it for further processing.
2. **parse\_train\_labels**, is used to parse the labels returned from the mat file. These labels are in a specific format that needs to be converted into a dictionary format.  
   A picture containing text, wall, indoor

   Description automatically generated  
   so we can later use it to create the required dataframe. This function takes the output of the first function and converts the labels into a dictionary format.
3. **to\_dataframe,** creates the required dataframe using the images list from the first function and the labels dictionary from the second function. This function is crucial for our system as it prepares the data in the required format for our model to process.

Graphical user interface

Description automatically generated with low confidence

# Creating the Class (Label) Column

Upon examining the MAFA dataset, we observed that it contains two important columns, namely, "occlude type" and "occlude degree". After analyzing the dataset images, we discovered that the faces wearing masks have "occlude type" values of either 1 or 2 and "occlude degree" values is 3. On the other hand, faces not wearing masks have values other than 1 or 2 in either of these columns. This information is critical for our model as it allows us to accurately classify whether a face is wearing a mask or not based on the values in these column A picture containing table

Description automatically generated

The following are some sample images with occlude type and degrees.

A screenshot of a computer

Description automatically generated with medium confidence



# Preparing the Train/Test Directories

The process of preparing directories for the train and test images is a critical step in our system. In this part, we aim to preprocess the images and extract faces from each image.

To achieve this, we will use the "face" column in the dataset, which provides the bounding box of the face in each image. Using this information, we can crop the face from each image and save the resulting images in a new directory.  
Text

Description automatically generated

A collage of a person's face

Description automatically generated with medium confidence

Once the face images have been extracted, we will create two directories, namely "full\_mask" and "no\_mask," to represent the two classes in our classification task. We will then using shutil library move each image to its respective directory based on its class in the dataframe.

Text

Description automatically generated

# Building the Model

In this step, we are building a model for our mask detection system using a pre-trained MobileNet V2 model with the TensorFlow framework (Tsang, n.d.).

Firstly, we import the necessary libraries:

Graphical user interface, text

Description automatically generated

Then, we load the directories we prepared into datasets.  
Graphical user interface, text, application, email

Description automatically generated

It's worth noting that the input shape of MobileNet V2 is (160, 160, 3), so we resize our images accordingly.

After loading the data, we preprocess it using Auto-tune and prefetch techniques. Auto-tune optimizes the performance of the data pipeline by automatically tuning the data loading parameters, and prefetching the data ensures that it is loaded and ready for processing by the GPU.

Graphical user interface, text, application

Description automatically generated

To increase the size and diversity of our training dataset, we use data augmentation. This technique applies a set of random transformations such as rotations, translations, zooming, flipping, and brightness adjustments to the original data, which helps to overcome the problem of overfitting.Text

Description automatically generated

Next, we load the pre-trained MobileNet V2 model. This model is designed for mobile and embedded devices, making it well-suited for our mask detection system. We freeze all the layers of the pre-trained model so they will not be updated during the training process.Graphical user interface, text

Description automatically generated

Finally, we build our model by using an input shape of (160, 160, 3) and appending the MobileNet V2 base model with an average pooling and Dropout layer to avoid overfitting. We add a prediction layer consisting of a Dense layer with 1 cell, which outputs a probability from 0 to 1, indicating whether the image is wearing a mask or not.

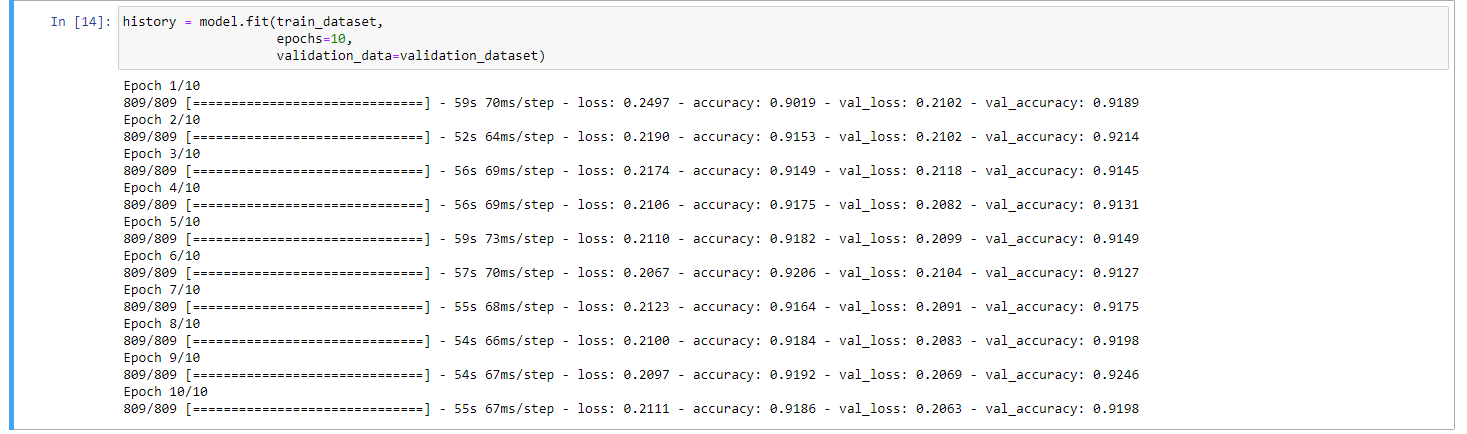
Graphical user interface, text, application

Description automatically generated

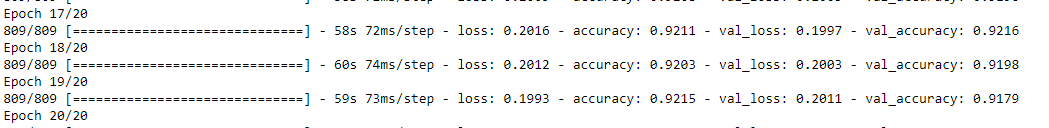
Graphical user interface, text

Description automatically generated

# Compiling the Model

After building the model, we compiled it using a learning rate of 0.001. However, during training, we noticed that the loss value was fluctuating, increasing, and decreasing. 

To address this issue, we decided to reduce the learning rate to 0.0001 and and train the model again over 20 epochs. As a result, we achieved better performance and accuracy of 92.5% with better validation loss.



Once we built the model, we evaluated its performance on the Test Dataset using the Confusion Matrix. Our analysis showed that the model performed exceptionally well on images with Full Mask data. However, we also discovered that the data was unbalanced, which means that we need to include more images with no mask data in future training iterations.Chart, treemap chart

Description automatically generated

After evaluating the model's performance, we saved it so that we can load and use it later for the classification of faces wearing masks or not.

# Building the Real-Time Face Mask Detection System

We have built a Real-Time Face Mask Detection System that integrates the FaceBoxes model to detect faces and our trained Face mask detection model. To implement this, we created a class that takes in a frame and applies the FaceBoxes detector to return both the bounding box of the face from that frame and the cropped face. The class also has a threshold score used by the model to detect faces.

Graphical user interface, text, application

Description automatically generated

Next, we loaded both the FaceBoxes model and our trained Face mask detection model in the system. The system utilizes the camera opened by the OpenCV2 library to read frames and send them to the face detector. The face detector returns the faces and their locations in the frame. These faces are then sent to the face mask classifier to predict whether the face is wearing a mask or not. Finally, the frame adds a rectangle over the face and a text declaring if there is a mask or not.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

With this Real-Time Face Mask Detection System, we can detect if a person is wearing a mask or not, which is crucial in promoting safety measures and preventing the spread of diseases.

TMDB5000 Movies Dataset Recommender System

# Data Acquisition and Description

Our system relies on the TMDB5000 Movies Dataset (Database, n.d.)which is released from The Movie Database website [The Movie Database (TMDB) (themoviedb.org)](https://www.themoviedb.org/)

The dataset contains two .csv files, each containing features for the movie.

1. credits.csv which contains:
   1. movie\_id - A unique identifier for each movie.
   2. cast - The name of lead and supporting actors.
   3. crew - The name of Director, Editor, Composer, Writer etc.
2. movies.csv which contains:
   1. budget - The budget in which the movie was made.
   2. genre - The genre of the movie, Action, Comedy ,Thriller etc.
   3. homepage - A link to the homepage of the movie.
   4. id - This is infact the movie\_id as in the first dataset.
   5. keywords - The keywords or tags related to the movie.
   6. original\_language - The language in which the movie was made.
   7. original\_title - The title of the movie before translation or adaptation.
   8. overview - A brief description of the movie.
   9. popularity - A numeric quantity specifying the movie popularity.
   10. production\_companies - The production house of the movie.
   11. production\_countries - The country in which it was produced.
   12. release\_date - The date on which it was released.
   13. revenue - The worldwide revenue generated by the movie.
   14. runtime - The running time of the movie in minutes.
   15. status - "Released" or "Rumored".
   16. tagline - Movie's tagline.
   17. title - Title of the movie.
   18. vote\_average - average ratings the movie recieved.
   19. vote\_count - the count of votes recieved.

# Data Loading and Pre-processing

Firstly, we load the two datasets, and join them on the ‘id’ column

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

The shapes of the datasets:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

# Creating Content-Based Recommender based on movie plot summary ‘overview.’

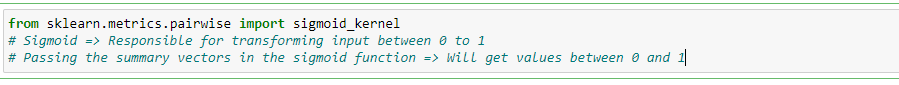
For each movie, we need to represent its summary as a vector, so later we can find its similarity with other movies’ summaries. To obtain this we used the NLP concept of TFidf Vectorizer.

Graphical user interface, text, application

Description automatically generated

Since we are applying content-based recommendation on the Summary of the movie, so using pair-wise similarity is the best suited for this function as it is more robust to outliers and noise.

The **sigmoid\_kernel** function from the **sklearn.metrics.pairwise** module is used in recommender systems to compute the pairwise similarity between items in a dataset. It is based on the sigmoid function, which maps the similarity values to a range of [0, 1].



# Creating a Function to get recommendations for a movie based on the summaries.

To get Movie Recommendation, there are some steps:

1. Get corresponding index of the movie title.
2. Get pairwise similarity scores with all other movies.
3. Sort the movies according to the highest pairwise scores.
4. Get the top 10 most similar movies.
5. Get the movies indices and return their corresponding titles.

Graphical user interface, text, application, email

Description automatically generated

# Creating a Function to get recommendations for a movie based on the Genres, Keywords and Credits.

The quality of our recommender would be increased with the usage of more features, we will build our recommender based on the three most important actors, the director and the keywords associated with that movie.

Building methods to return the directors and the top three actors.

Graphical user interface, text, application, email

Description automatically generated

Applying on our dataset:

Text

Description automatically generated

We are creating a function to clean the data by removing spaces and converting all characters to lower case. This process can aid in finding similarities between items by eliminating differences caused by capitalization and white spaces.Graphical user interface, text, application

Description automatically generated

Combining all features in one column ‘soup’ to easily create the Vectorizer Matrix

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

We have developed a recommendation function that utilizes cosine similarity. Cosine similarity is particularly suitable for our purpose as we are looking to identify similarities based on individual words, rather than entire sentences.  
Text

Description automatically generated

# Combination of Both Developed Content-Based Recommender Functions

In this part, we will combine the two previously developed functions, each of which takes a movie title and returns a list of top 10 similar movies based on either the summary or a combination of genre, actors, and credits of the movie. We will do this using a simple scoring system to generate a final list of top 10 recommended movies.

The system combines the top 10 movie recommendation lists from each function, assigning scores based on the order of the movies in the combined list. If a movie appears in both lists, its score is incremented to give it a higher priority. Finally, the system returns a list of the movies with the highest scores, representing the top recommendations.

Text

Description automatically generated

Text

Description automatically generated

# References

*Database, T. M. (n.d.). TMDB 5000 Movie Dataset. From Kaggle: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata*

*Jianfeng Wang, Y. Y. (n.d.). Face Attention Network: An Effective Face Detector for the Occluded Faces. From arxiv: https://arxiv.org/abs/1711.07246*

*Shifeng Zhang, X. Z. (n.d.). FaceBoxes: A CPU Real-time Face Detector with High Accuracy. From arxiv: https://arxiv.org/abs/1708.05234*

*Shiming Ge, J. L. (n.d.). Detecting Masked Faces in the Wild with LLE-CNNs. From imsg: https://imsg.ac.cn/research/maskedface.html*

*Shiming Ge, J. L. (n.d.). Detecting Masked Faces in the Wild With LLE-CNNs. From Papers with Code: https://paperswithcode.com/paper/detecting-masked-faces-in-the-wild-with-lle*

*Tsang, S.-H. (n.d.). Review: MobileNetV2 — Light Weight Model (Image Classification). From Towards Data Science: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c*

*Yang, S. (n.d.). WIDER FACE: A Face Detection Benchmark. From Wider Face: http://shuoyang1213.me/WIDERFACE/*