



# Face Value: Using AI to Predict Stardom in South Korean Drama

Jackson Tin,  
General Assembly

# Problem



South Korean entertainment industry is fiercely competitive and burgeoning with potential talents. In 2022, the Korean film industry is worth **USD 1.3 billion**.



Current manual scouting methods may not fully capture the breadth of this potential



Subject to the bias of the individual talent scouts themselves

# Motivation



Develop a predictive model that uses face images to assess potential success in the Korean entertainment industry



Automate screening of candidates by feeding the face images to the model to predict potential success based on data



For myself, a fun project to take on

# What is **Success**?

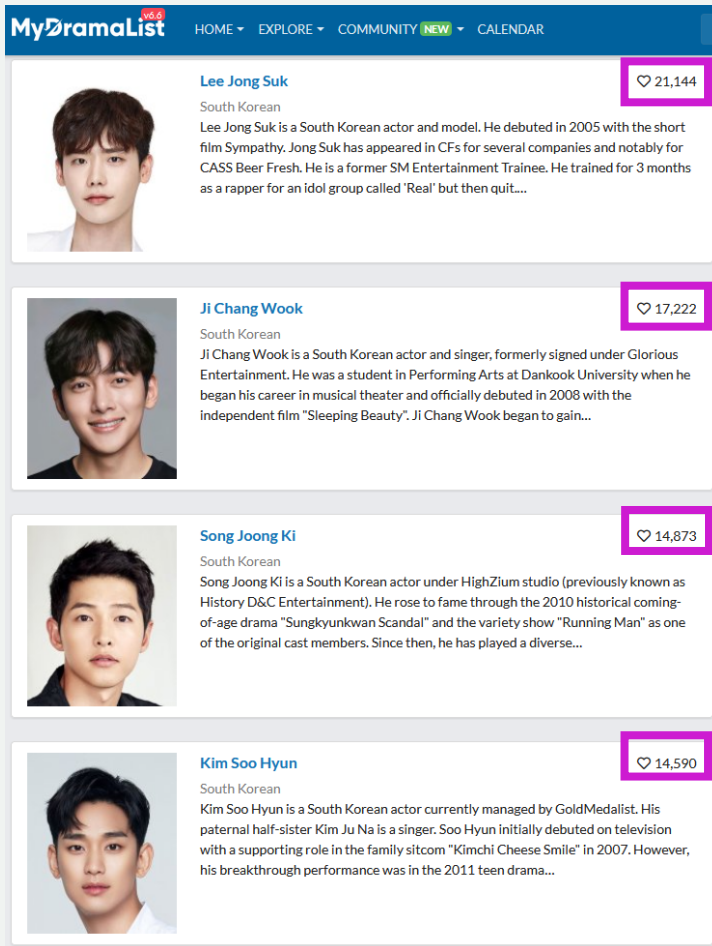
For the purposes of developing a prototype:

Popularity level of **male South Korean** actors based on mydramalist.com will be used to gauge success



For future works, other metrics such as net worth, brand endorsements, drama ratings etc. can be factored in when resources are available

# Popularity on mydramalist.com

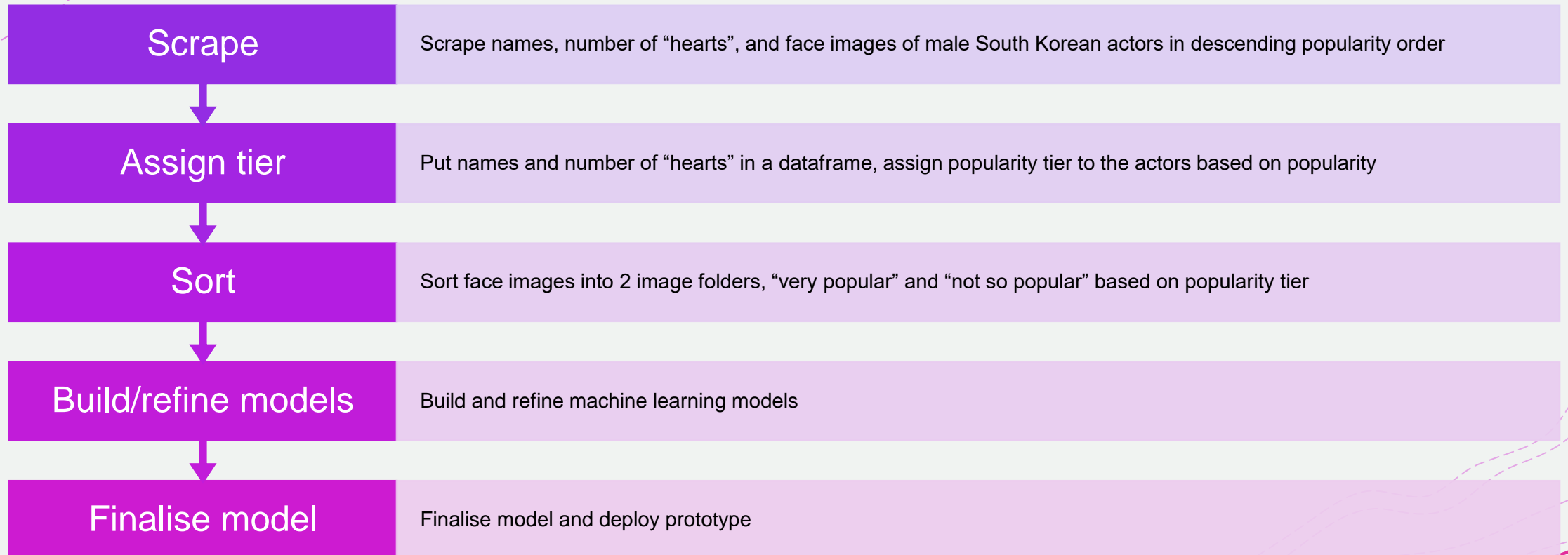


The screenshot shows the MyDramaList website interface. At the top is a navigation bar with links: HOME, EXPLORE, COMMUNITY (with a 'NEW' badge), and CALENDAR. Below the navigation bar, four actor profiles are listed. Each profile includes a portrait photo, the actor's name, their nationality, and a brief biography. A red box highlights the heart count for each actor: Lee Jong Suk (21,144), Ji Chang Wook (17,222), Song Joong Ki (14,873), and Kim Soo Hyun (14,590).

Actor	Heart Count
Lee Jong Suk	21,144
Ji Chang Wook	17,222
Song Joong Ki	14,873
Kim Soo Hyun	14,590

- + Users give a “heart” to actors they like
- + For each actor, users can only give a “heart” once
- + Users can give a “heart” to more than one actor
- + The higher the “hearts”, the more popular the actor

# General Workflow



# Scraping and Data Collection

1. Use `BeautifulSoup` and `requests` to scrape names, number of “hearts” and hyperlinks of the face images for South Korean Male actors in descending order of number of “hearts”
2. Remove duplicate names, keep first entry
3. Modify the last letter of each hyperlink to access the higher resolution version of the face images
4. Download all face images with `requests.get()`
5. Assign actors in dataframe to “very popular” tier or “not so popular” tier based on number of “hearts”

# Scraping and Data Collection

241

actors in very popular tier

249

actors in not so popular tier



# Very Popular Tier



Lee Jong Suk



Ji Chang Wook



Park Seo Hoon



Song Joong Ki



Kim Soo Hyun



Lee Min Ho



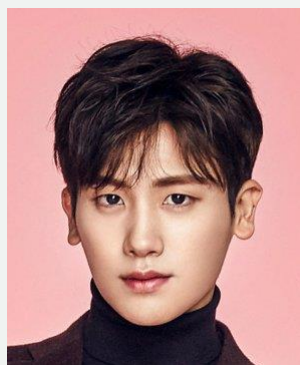
Nam Joo Hyuk



Lee Joon Gi



Lee Dong Wook



Park Hyung Sik



Ji Sung



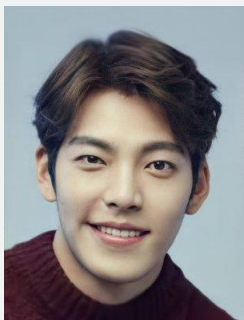
Gong Yoo



Park Bo Gum



Seo In Guk



Kim Woo Bin



Lee Seung Gi



Cha Eun Woo



Hyun Bin



Yoo Seung Ho



Seo Kang Joon



# Not So Popular Tier



Kang Han Saem



Yoo Byung Jae



Jang Won Young



Park Sang Won



Choi Dong Goo



Ahn Sung Ki



B-Bomb



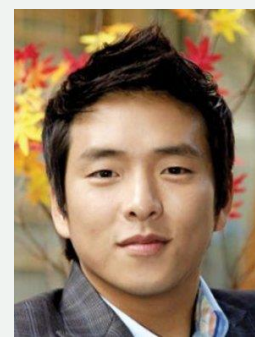
Kim Chung Soon



Tony Ahn



Son Ji Chang



Kim Dong Yoon



Kim Min Kyo



Kim Tae Hyung



Kwon Hyuk Soo



Lee Seung Hyo



Lee Young Hoon



Park Geun Hyung



Heo Dong Won



Park Yong Woo

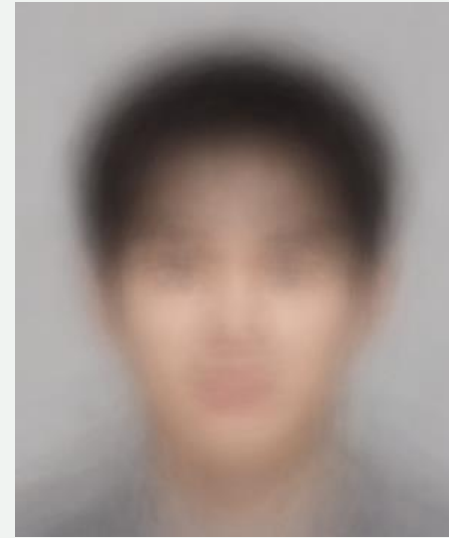


Shin Seung Hwan

Average face  
of top 20 very  
popular actors



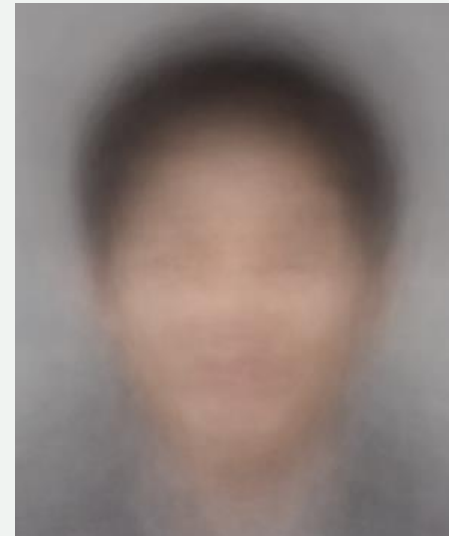
Average face  
of all 241 very  
popular actors



Average face  
of 20 not so  
popular actors



Average face  
of all 249 not  
so popular  
actors





# Why do looks matter?

Looks are not everything, but..

- + Your visuals make the first impression, more so in the entertainment industry
- + Looks also contribute to the halo effect, where good-looking people are perceived as a good or talented person
- + For other things like acting skills, dancing skills, vocals, these can be trained



# What constitutes facial attractiveness?

- + **Facial averageness**

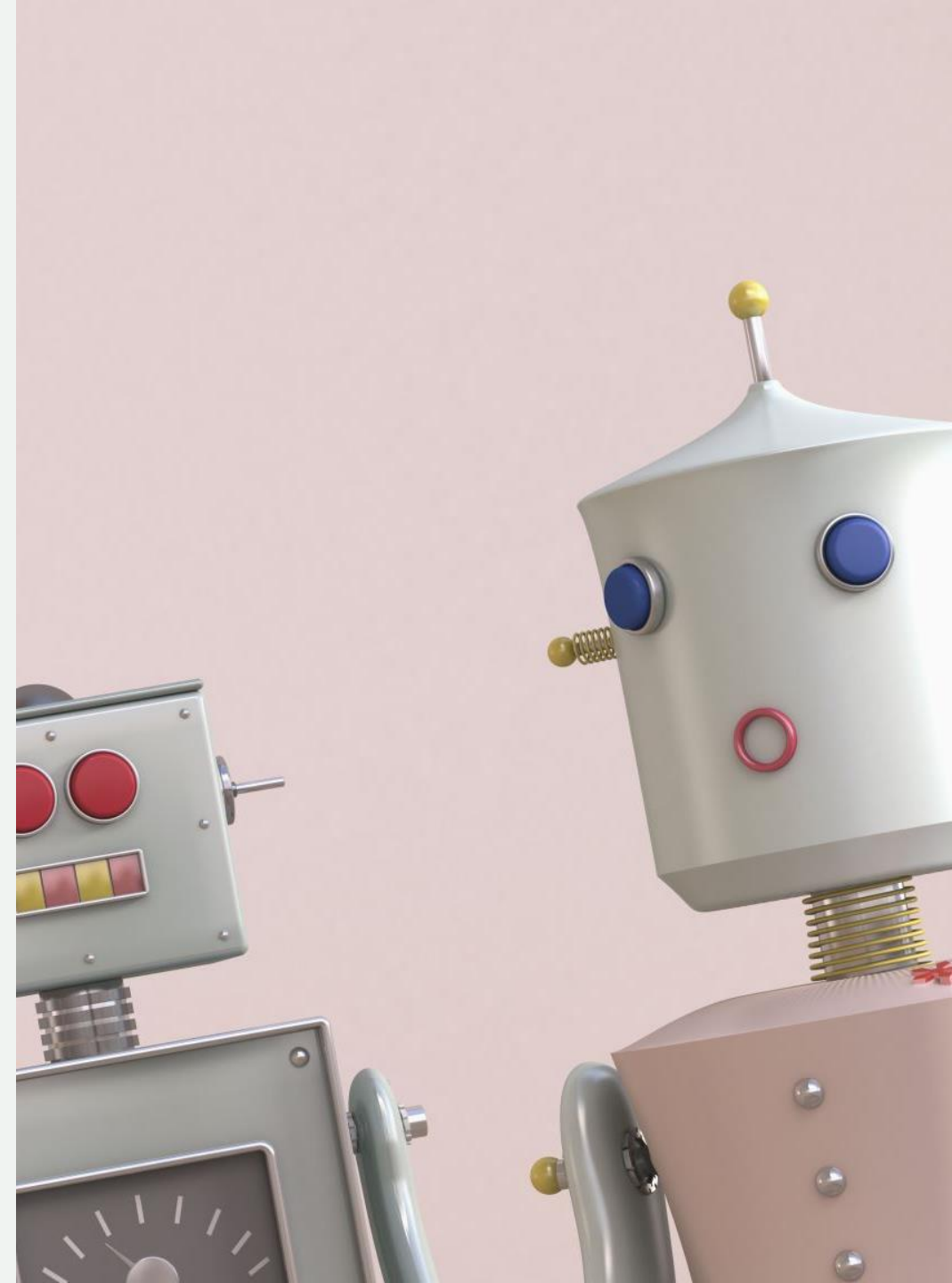
- + With some parameters, it's beneficial to be above average or away from the norm to achieve that 'model tier' look, while for others a sizable deviation can exponentially hurt the subject's aesthetic harmony

- + **Symmetry**

- + **Youthfulness**

- + **Sexual dimorphism**

- + Facial sexual dimorphism emerges at puberty: as the size and shape of the male and female faces increase with age, faces begin to show different secondary sexual characteristics (i.e., **masculine** or **feminine**)
- + For example, male jawbones become larger, cheekbones more prominent, cheeks and lips thinner



# Youthfulness - Very Popular Tier

+ Significant emphasis on maintaining a youthful, almost boyish look



Lee Jong Suk



Ji Chang Wook



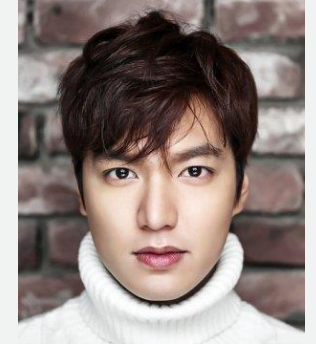
Park Seo Hoon



Song Joong Ki



Kim Soo Hyun



Lee Min Ho



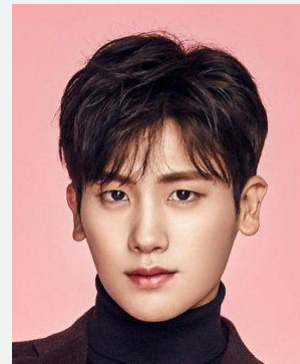
Nam Joo Hyuk



Lee Joon Gi



Lee Dong Wook



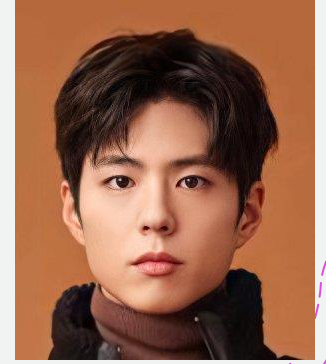
Park Hyung Sik



Ji Sung



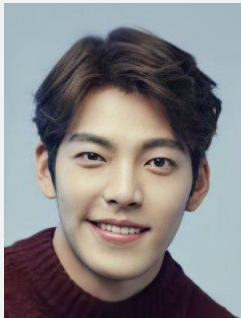
Gong Yoo



Park Bo Gum



Seo In Guk



Kim Woo Bin



Lee Seung Gi



Cha Eun Woo



Hyun Bin



Yoo Seung Ho



Seo Kang Joon



# Sexual dimorphism

+ South Korean male beauty standards are unique

## **Masculine** Features

- Sharp, V-shaped or a slightly rounded jawline
- High, defined cheekbones

## **Non-Traditional/Masculine** Features

- Pale and flawless skin
- Plump lips
- Slim face
- Cosmetics

## **Other Societal/Trendy** Features

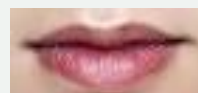
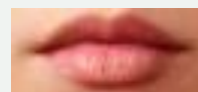
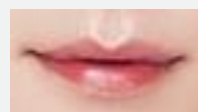
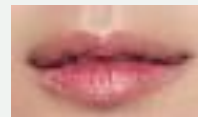
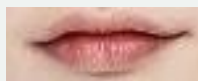
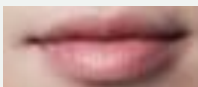
- Double eyelids, big eyes
- Well-defined nose, high nose bridge
- Clean look
- Hairstyle

- **Sharp jawline**
- **High, defined cheekbones**





- **Plump lips**
- **Cosmetics**





- **Hairstyle**
- **Clean-shaved look**





# Not So Popular Tier



Kang Han Saem



Yoo Byung Jae



Jang Won Young



Park Sang Won



Choi Dong Goo



Ahn Sung Ki



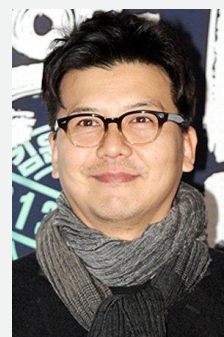
B-Bomb



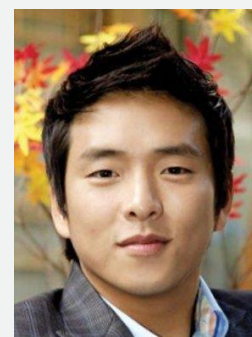
Kim Chung Soon



Tony Ahn



Son Ji Chang



Kim Dong Yoon



Kim Min Kyo



Kim Tae Hyung



Kwon Hyuk Soo



Lee Seung Hyo



Lee Young Hoon



Park Geun Hyung



Heo Dong Won



Park Yong Woo



Shin Seung Hwan

# Feature Extraction – VGG16

- VGG16 is a type of pretrained convolutional neural network (CNN) model that is commonly used for object detection and classification
- VGG16 takes an image as input and passes it through its network layers. Each layer learns to recognize different features in the image.
  - Early layers detect simple features like edges and curves,
  - Deeper layers recognize more complex features like shapes, patterns, or objects.
- Take out the top layer, i.e. the classification layer, to output numerical representation of the image that encapsulates the features the model has learned

col_9	...	col_25079	col_25080	col_25081	col_25082	col_25083	col_25084	col_25085	col_25086	col_25087	col_25088
0.0	...	0.0	0.0	0.0	0.0	3.9212093	0.0	0.0	0.027669907	0.0	very_popular
0.0	...	0.0	0.0	5.1023827	1.3957657	0.0	0.0	0.0	17.52898	0.0	very_popular
0.0	...	0.0	0.0	0.0	98.71404	0.0	0.0	0.0	0.0	0.0	very_popular
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.752463	0.0	very_popular
0.0	...	0.0	6.256437	3.4620895	0.0	0.0	0.0	0.0	0.0	0.0	very_popular
...	...	...	...	...	...	...	...	...	...	...	...
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	20.953001	0.0	not_so_popular
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	not_so_popular
0.0	...	0.0	0.0	0.0	0.0	16.664928	0.0	0.0	7.663937	0.0	not_so_popular
0.0	...	7.1930723	0.0	9.211779	0.0	0.0	15.40993	0.0	0.0	0.0	not_so_popular
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19.3946	0.0	not_so_popular

- Each column is a feature

# Building the Models

1. Logistic Regression (lr)
2. K Neighbours Classifier (knn)
3. Ada Boost Classifier (ada)
4. Extreme Gradient Boosting (xgboost)
5. Gradient Boosting Classifier (gbc)
6. Light Gradient Boosting Machine (lightgbm)
7. Random Forest Classifier (rf)
8. SVM
9. Decision Tree Classifier (dt)
10. Extra Trees Classifier (et)
11. Linear Discriminant Analysis (lda)
12. Quadratic Discriminant Analysis (qda)
13. Ridge Classifier (ridge)

# VGG16 Base Feature Extraction Model

	Best Model		Accuracy	AUC	Recall	Precision	f1-score
Train Size 0.7, 10 folds	xgboost	Train	0.8649	0.9262	0.8501	0.8726	0.8611
		Test	0.7957	0.8637	0.7868	0.8023	0.7917
Train Size 0.7, 5 folds	xgboost	Train	0.8571	0.9266	0.8432	0.8636	0.8532
		Test	0.7960	0.8622	0.7870	0.7975	0.7920
Train Size 0.65, 10 folds	xgboost	Train	0.8354	0.9017	0.8269	0.8363	0.8314
		Test	0.7674	0.8441	0.7687	0.7631	0.7631
Train Size 0.65, 5 folds	lightgbm	Train	0.8593	0.9205	0.8397	0.8689	0.8539
		Test	0.7925	0.8540	0.7562	0.8097	0.7811

Problems:

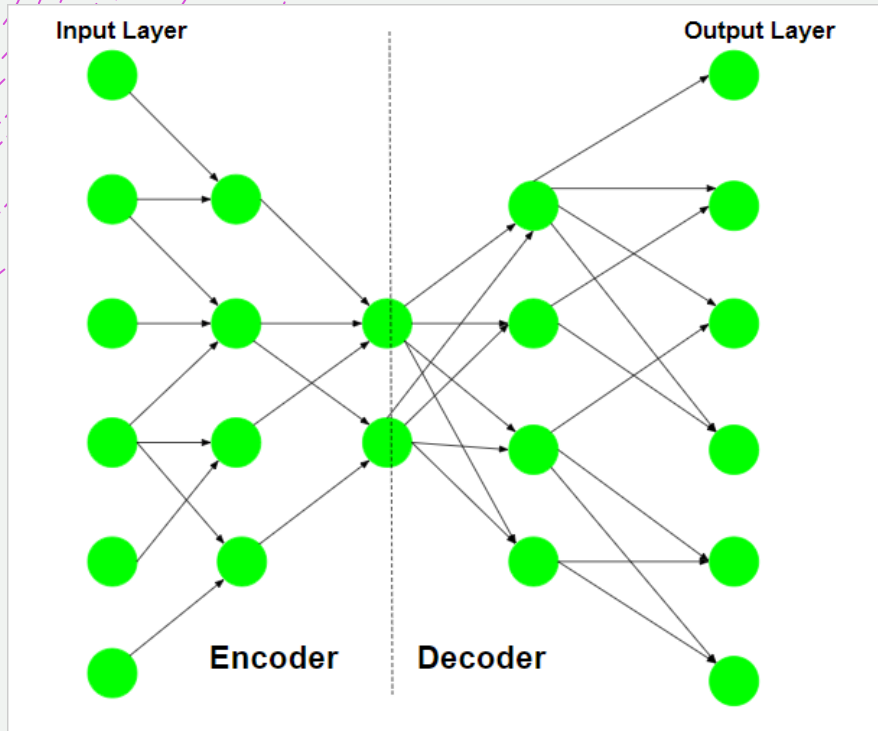
- Models take hours to train
- When model is deployed, each prediction takes a long time



# VGGFace Base Feature Extraction Model

	Best Model		Accuracy	AUC	Recall	Precision	f1-score
Train Size 0.7, 10 folds	et	Train	1.0000	1.0000	1.0000	1.0000	1.0000
		Test	0.7845	0.8699	0.7518	0.8172	0.7775
Train Size 0.7, 5 folds	xgboost	Train	0.8535	0.9441	0.8151	0.8787	0.8455
		Test	0.7347	0.8422	0.7046	0.7466	0.7232
Train Size 0.6, 10 folds	lr	Train	1.0000	1.0000	1.0000	1.0000	1.0000
		Test	0.7721	0.8498	0.7457	0.7926	0.7533
Train Size 0.6, 5 folds	xgboost	Train	0.8563	0.9416	0.8138	0.8860	0.8481
		Test	0.7724	0.8433	0.7241	0.7947	0.7567

# Feature Extraction – VGG16 + Autoencoder



- An autoencoder is a type of artificial neural network used for learning efficient codings of input data, i.e.:
  - Reduces dimensionality of the data
  - Retain only the most important features of the data
- Consists of two main parts:
  - **Encoder**: Encodes the input image as a compressed representation in a reduced dimension. The compressed image typically contains the main attributes of the original image.
  - **Decoder**: Aims to reconstruct the input and decodes the compressed image back into the original image
- Goal is to make our image classification model simpler (as it takes input of lower dimension) and faster (as there's less data to process) without sacrificing too much performance



# Feature Extraction – VGG16 + Autoencoder

	Best Model		Accuracy	AUC	Recall	Precision	f1-score
Batch Size 256, 5 folds, 50 epochs	gbc	Train	1.0000	1.0000	1.0000	1.0000	1.0000
		Test	0.7725	0.8369	0.7756	0.7696	0.7708
Batch Size 512, 5 folds, 50 epochs	lightgbm	Train	0.9971	1.0000	1.0000	0.9941	0.9971
		Test	0.7697	0.8512	0.7873	0.7611	0.7717
Batch Size 512, 5 folds, 50 epochs	xgboost	Train	0.7752	0.8806	0.7640	0.7765	0.7700
		Test	0.7520	0.8451	0.7456	0.7611	0.7498

# Feature Extraction – VGGFace + Autoencoder

	Best Model		Accuracy	AUC	Recall	Precision	f1-score
Batch Size 32, 10 folds, 100 epochs	xgboost	Train	0.8478	0.9318	0.789	0.8901	0.8361
		Test	0.7550	0.8342	0.6923	0.7914	0.7333
Batch Size 32, 10 folds, 50 epochs	lightgbm	Train	0.8322	0.9165	0.8225	0.8347	0.8284
		Test	0.7642	0.8279	0.7643	0.7598	0.7586

# Feature Extraction – EfficientNetB0 + Autoencoder

	Best Model		Accuracy	AUC	Recall	Precision	f1-score
Batch Size 512, 10 folds, 50 epoch	lr	Train	0.8452	0.9288	0.8573	0.8333	0.8451
		Test	0.8165	0.8847	0.8346	0.8101	0.8170
Batch Size 512, 5 folds, 50 epochs	lda	Train	0.8513	0.9232	0.8521	0.8472	0.8496
		Test	0.8194	0.8853	0.8226	0.8137	0.8163
Batch Size 512, 10 folds, 25 epochs	et	Train	0.8141	0.8831	0.8481	0.7904	0.8181
		Test	0.8018	0.8605	0.8456	0.7775	0.8080
Batch Size 512, 5 folds, 25 epochs	et	Train	0.8455	0.9236	0.8772	0.8214	0.8483
		Test	0.8046	0.8740	0.8342	0.7849	0.808
Batch Size 256, 10 folds, 25 epochs	et	Train	0.9129	0.9842	0.9329	0.8947	0.9134
		Test	0.8191	0.8790	0.8338	0.8174	0.8206
Batch Size 128, 10 folds, 100 epochs	ada	Train	0.8795	0.9610	0.8843	0.8729	0.8785
		Test	0.8022	0.8871	0.7926	0.8101	0.7976
Batch Size 32, 10 folds, 50 epochs	lda	Train	0.8264	0.9072	0.8448	0.8108	0.8274
		Test	0.8134	0.8872	0.8283	0.8091	0.8149

# Feature Extraction – MobileNet + Autoencoder

	Best Model		Accuracy	AUC	Recall	Precision	f1-score
Batch Size 512, 10 folds, 50 epochs	rf	Train	0.8992	0.9678	0.9277	0.8755	0.9007
		Test	0.7813	0.8347	0.8169	0.7621	0.7851
Batch Size 512, 5 folds, 50 epochs	et	Train	0.8105	0.8875	0.8402	0.7891	0.8137
		Test	0.7725	0.8389	0.7868	0.7612	0.7735

# Best Model

Model Type		Best Model		Accuracy	AUC	Recall	Precision	f1-score
EfficientNetB0 + Autoencoder	Batch Size 512, 5 folds, 50 epochs	Ida	Train	0.8513	0.9232	0.8521	0.8472	0.8496
			Test	0.8194	0.8853	0.8226	0.8137	0.8163
VGG16	Train Size 0.7, 5 folds	xgboost	Train	0.8571	0.9266	0.8432	0.8636	0.8532
			Test	0.7960	0.8622	0.7870	0.7975	0.7920
VGG16 + Autoencoder	Batch Size 512, 5 folds, 50 epochs	xgboost	Train	0.7752	0.8806	0.7640	0.7765	0.7700
			Test	0.7520	0.8451	0.7456	0.7611	0.7498
VGGFace	Train Size 0.6, 5 folds	xgboost	Train	0.8563	0.9416	0.8138	0.8860	0.8481
			Test	0.7724	0.8433	0.7241	0.7947	0.7567
MobileNet + Autoencoder	Batch Size 512, 10 folds, 50 epochs	rf	Train	0.8105	0.8875	0.8402	0.7891	0.8137
			Test	0.7725	0.8389	0.7868	0.7612	0.7735

EfficientNetB0 + Autoencoder gives the best test accuracy, ~**3-4%** more than others

A photograph of a Space Shuttle launching at night. The shuttle is positioned in the center-right of the frame, ascending vertically. A large, bright plume of fire and white smoke trails from its base. To the right of the shuttle, a massive, dark, billowing cloud of smoke and vapor rises into the dark blue night sky. On the far right edge, a portion of a launch tower's metal structure is visible. The foreground is a solid black horizontal band. In the bottom-left corner, the word "Deployment" is written in a white, serif font. The image is decorated with white dashed wavy lines in the top-left corner and a solid pink wavy line in the bottom-right corner.

# Deployment

# Future Work



Expand definition of success to include more metrics including **net worth**, **brand endorsements**, **drama ratings** and others



Expand **regional scope** to Japan, Taiwan, outside of Asia



Add **more actors**, as well as many **more images** per actor



Train using **neural networks** when there are more data, and explore adding a third class of popularity



Deploy model on a platform with **more capabilities** such as **batch upload** and **dashboard**