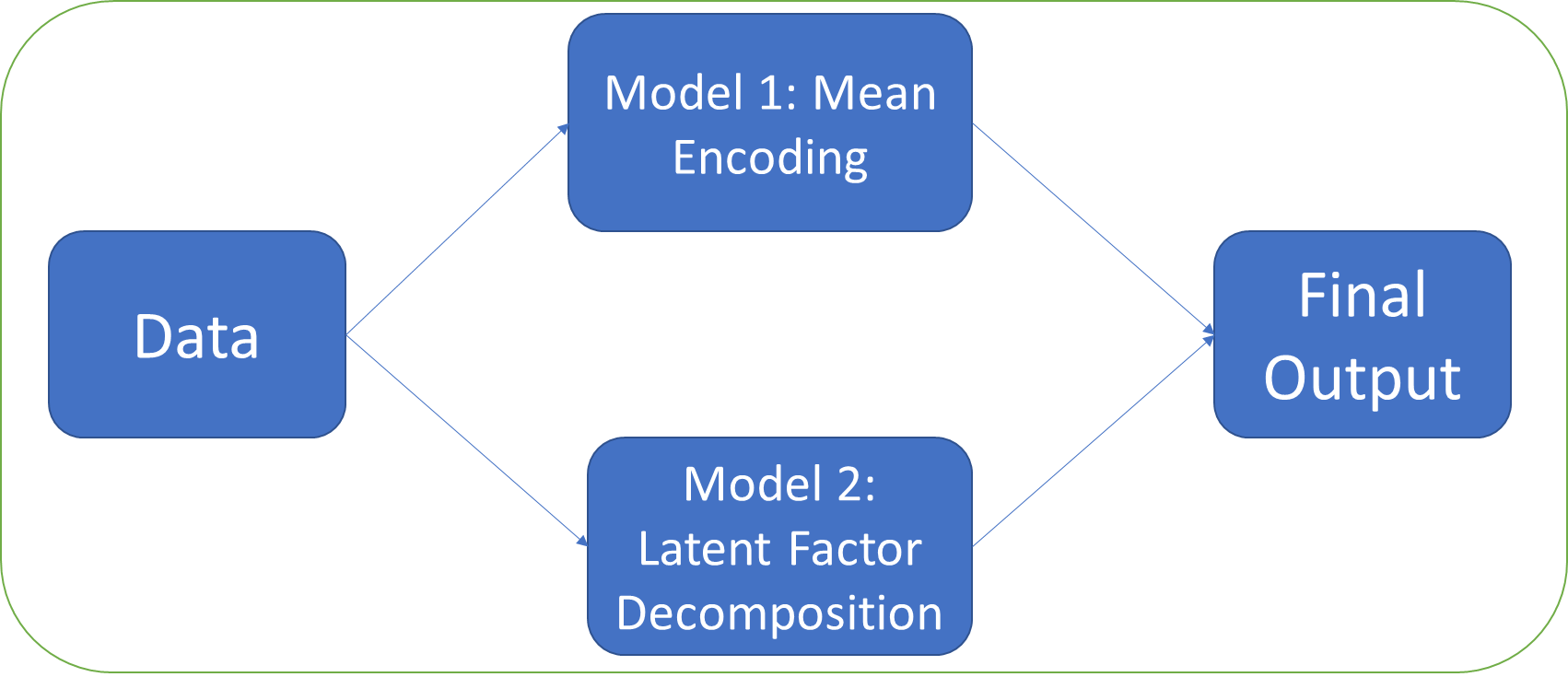
Felicity: Kings of ML

# Problem Statement

Determine kda\_ratio for user\_id, hero\_id combination.

# High Level Approach:



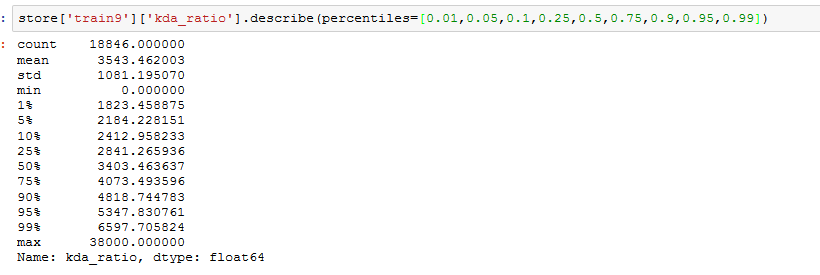
Predict the kda\_ratio using below two models:

1. Mean Encoding
2. Latent Factor Decomposition

Take an ensemble of both these models and get the final output.

# Data Pre processing:

Below is the univariate distribution of kda\_ratio.



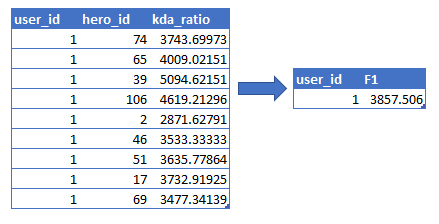
Outliers from kda\_ratio are removed by clipping the values. The kda\_ratio is capped and floored between 2000-6000 for all the further steps.

# Mean Encoding:

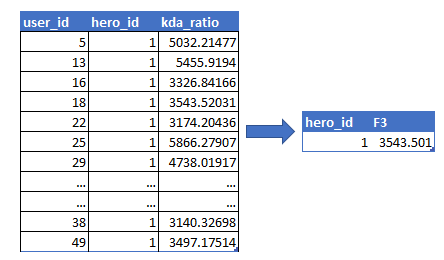
1. Calculate the below features for each data point in training set:
   1. F1: Average kda\_score for each user
   2. F3: Average kda\_score for each hero
   3. F2: Average kda\_score for each user after removing the hero bias (F2)
   4. F4: Average kds\_score for each hero after removing the user bias (F1)
2. Encode the above features for each data point in the test set
3. Final Score for each test sample = ((F1+F4)+(F2+F3))/2

## Calculations:

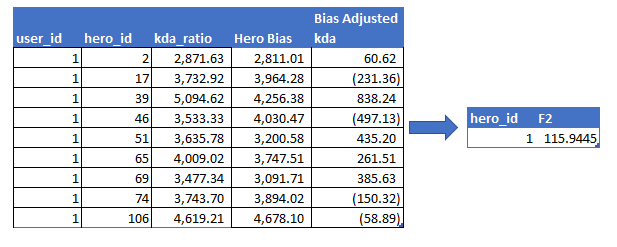
1. F1:



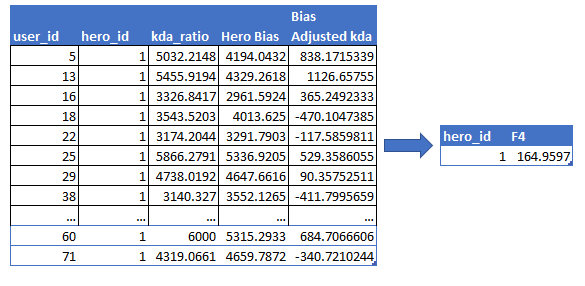
1. F3:



1. F2:



1. F4:



Score 1: (F1+F2+F3+F4)/2

# Latent Factor Decomposition:

Latent Factor Decomposition is an unsupervised learning approach where User-Hero-kda\_ratio matrix is decomposed to two rectangular matrices. An intermediate dimension is learnt using the data which represents the latent/interaction factors between user and hero. The algorithm used is a tweaked version of SVD. Least squares is used to converge to the final matrices. The kda\_ratio’s which are unknown are not used in least squares calculation.

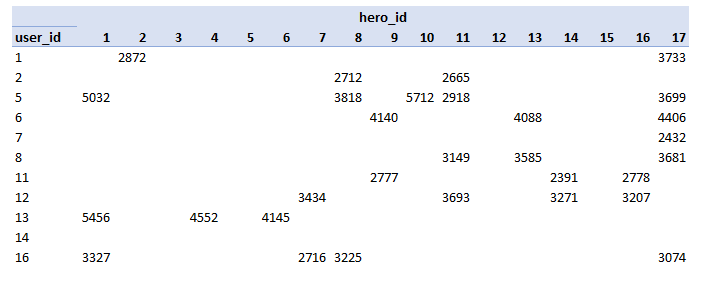
X: User-Hero-kda\_ratio matrix.

A: User-Latent Factors

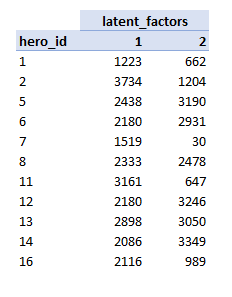
B: Latent Factors-Item

**X=A\*B**

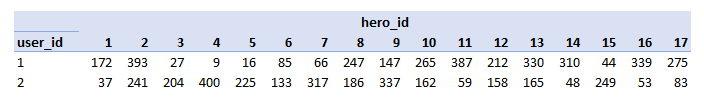
Below is an example of the process (data only for illustration).



**X =**

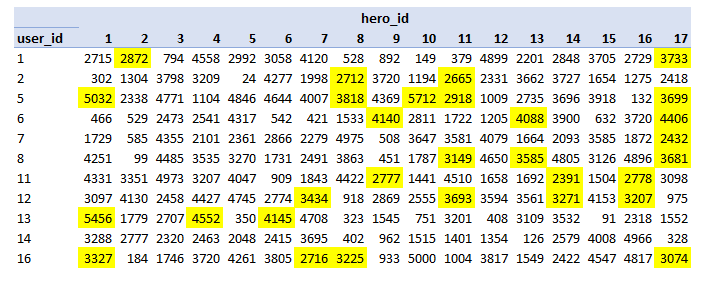


**A** =



**B** =

After decomposition of X, X is reconstructed using dot product of A,B. This also helps fill up the missing kda\_ratio’s which are not available.



XReconstructed=

The reconstructed X is used to pick the kda\_rating of the test data points.

## Tuning the Matrix Factorization

The matrix factorization algorithm has number of latent spaces as a hyperparameter and number of iterations used to get the decomposed matrices. Validation on ‘train\_1’ dataset is used to tune these hyperparameters. Below is the relation of these hyperparameters on the model performance:

1. #Latent Factors: Increasing the latent factors increases the amount of information captured for the input matrix. Higher the latent factors, lower the train error. Upto a particular limit, the validation error also decreases. Increasing the number of latent factors further will lead to overfitting and we will observe increasing validation error
2. #Iterations: More the number of iterations, more information will be learnt by our model. As we learn only on the available values, increasing the number of iterations will learn to overfitting on training data and not the corresponding gain in the validation data.

Using validation on the ‘train\_1’ dataset, we get the below parameters for our model:

#Latent Factors: 2

#Iterations: 10

# Final Score

To get the final kda\_ratio, the above two models are ensembled. We take an average of kda\_ratios of both the models to get the final score.

kda\_ratio = (Mean Encoding kda + Latent Factor kda)/2