Case Study 9:Netflix Movie Recommendation System (Collaborative based recommendation)

7.1 Business/Real world problem : Problem definition

This case study is based on the competition hosted by Netflix.
This competition began on October 2,2008 and ended on October 2,2011.
The prize amount was 1 million US dollars. It’s a huge cash prize.

What is Netflix ?
Netflix is an online streaming platform where the users can stream movies,web series and much more. Netflix would recommend movies or web series or any other video content based on your watch history. For example, if you use Netflix more to watch horror movies, then when you visit the Netflix next time, it will recommend you top horror movies. This is one such simple example. But, we tend to watch multiple genres of movies based on our current mood,references made by our friends,bias towards certain actors, etc. So, Netflix needs to encompass all of these and need to give you good recommendations.

Let’s make it formal

1.) We have a set of users where a ith user is denoted as ui.
2.) We have a set of movies where a ith movie is denoted as mi.
3.) We have a set of ratings where a i,j rating is defined as the rating given by user ui on movie mj. The possible values for ratings are [1,2,3,4,5].
4.) Netflix collects all those ratings data which are provided by the users.
5.) Netflix internally uses a system called **Cinematch** for recommendations. Cinematch's job is to predict whether someone will enjoy a movie based on how much they liked or disliked other movies.

6.) The aim of this competition is to do better than the existing system called Cinematch which Netflix uses. If any participant surpassed 10% of Cinematch performance, then the participant would be declared as the winner. The performance metric used here is Root Mean Squared Error.
7.) For more details on this competition, please refer to this

7.2 Objectives and constraints

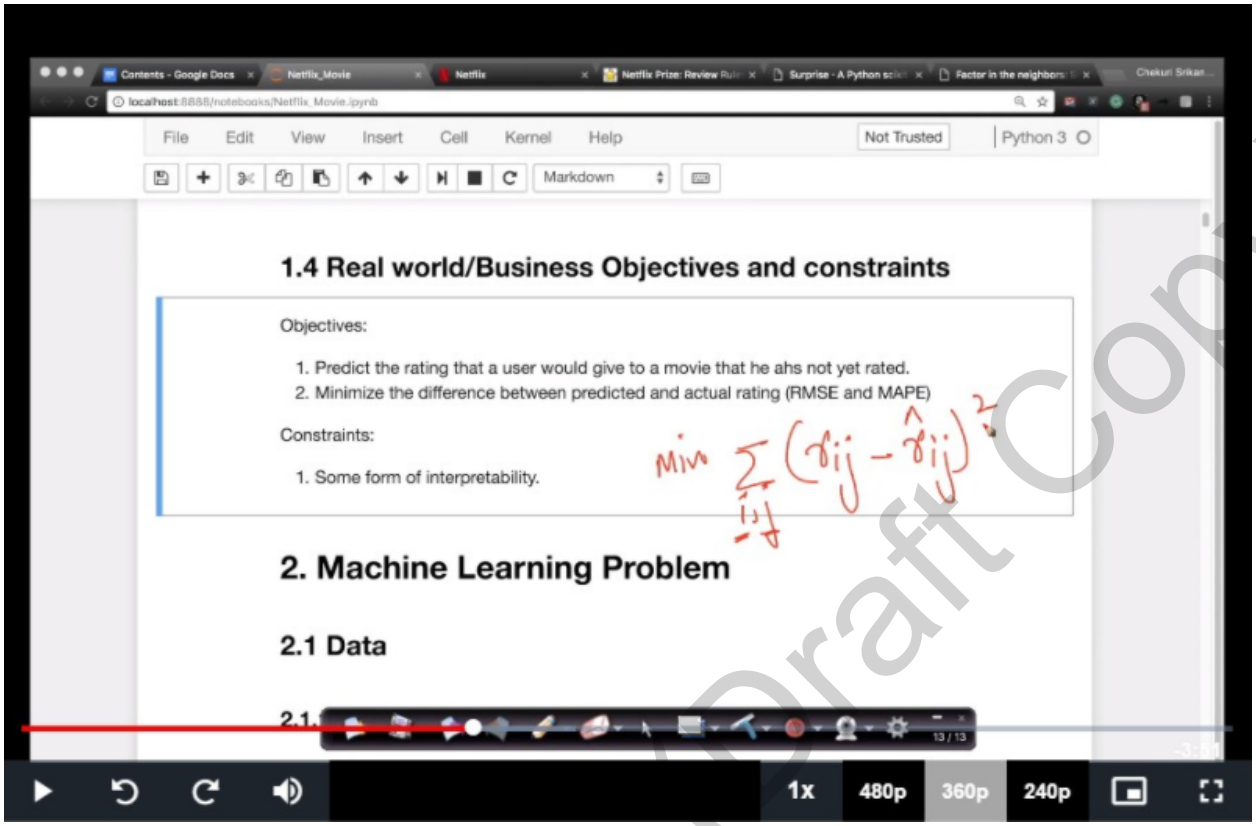
Objectives :

1.) Predict the rating that a user would give to a movie that he has not yet rated.
2.) Minimize the difference between predicted and actual rating (RMSE and MAPE). We will use RMSE here.

Constraints :

1.) Some form of interpretability. Our model should explain why a particular user ui given a rating rij to movie mj i.e, if a user gives a higher rating to a particular movie, then our algorithm or model should explain the reason behind it.

Let's look at the goal of our solution.



Timestamp : 02:17

1.) rij represents the actual rating given by user ui on movie mj.
2.) rij (with a cap) represents the predicted rating by our model or system or algorithm given by user ui on movie mj.
3.) The goal is to minimize the difference across all the ratings given by ser ui on movie mj i.e, for all possible combinations of i and j.
4.) If we minimize this difference squared term, then we are doing better. This is because the difference squared term is minimized only if the two terms are close enough to each other or rij (with a cap) is close to rij.

7.3 Mapping it to the ML problem:Data overview.

Data :

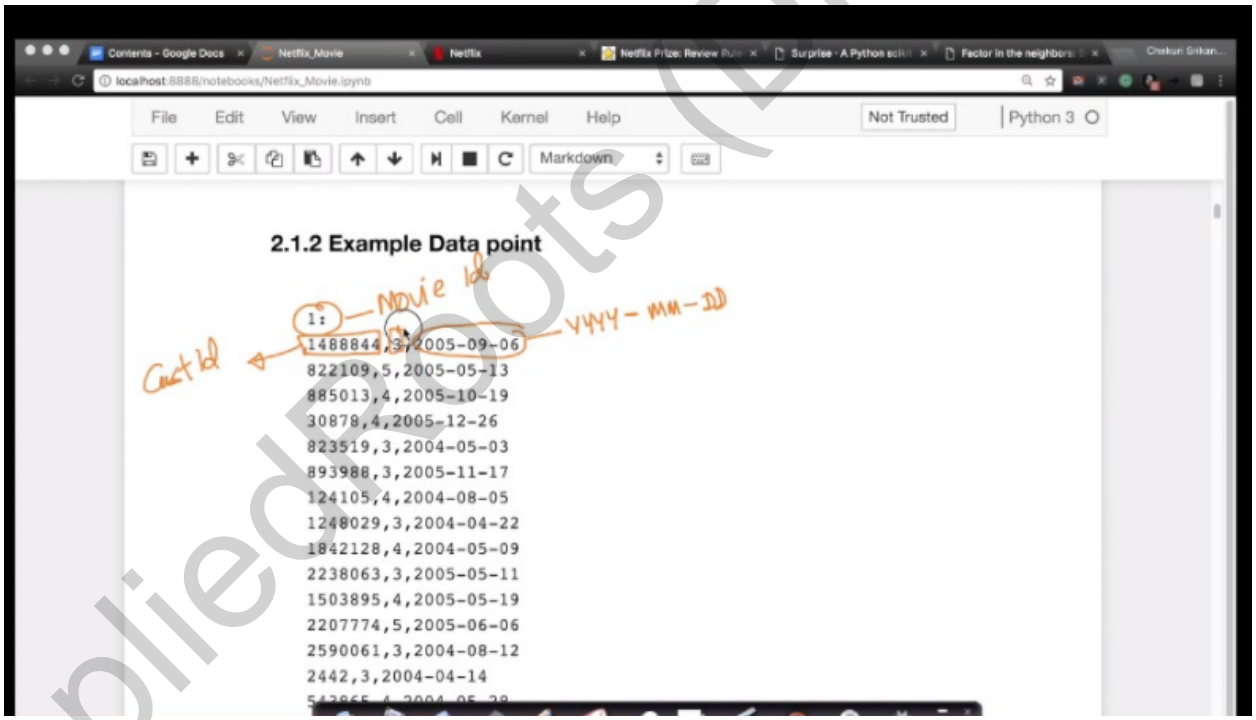
Data files : Get the data from here

1.) combined_data_1.txt
2.) combined_data_2.txt
3.) combined_data_3.txt
4.) combined_data_4.txt
5.) movie_titles.csv : It contains movie_id and the movie name

Example data point

1: -> movie_id
148844,3,2005-09-06

The above says that user id of 148844 watched the movie with movie id 1 and gave a rating 3. This was done on 2005-09-06 (YYYY-MM-DD).



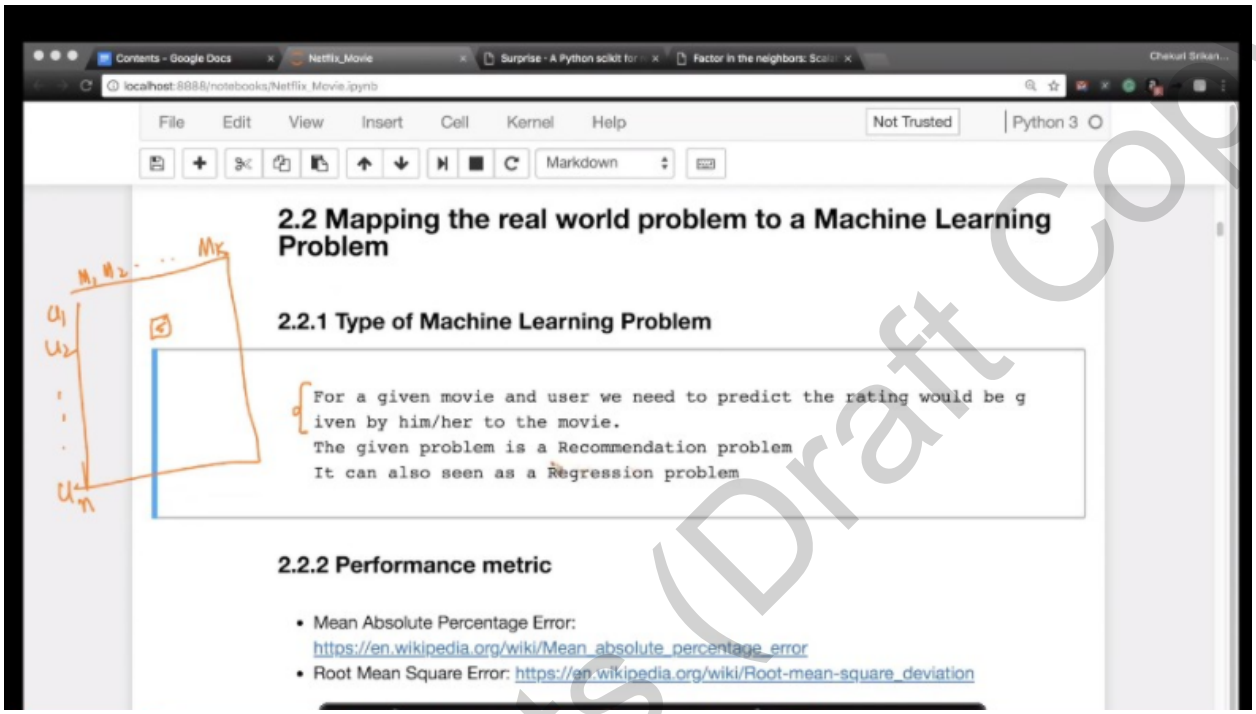
Timestamp : 01:50

Note : Ratings are from 1 to 5 i.e, {1,2,3,4,5}

MovieIDs range from 1 to 17,770 in a sequence.

CustomerIDs range from 1 to 2649429 with gaps. There are 480189 users.

7.4 Mapping to an ML problem:ML problem formulation



Timestamp : 0:29

The above is a matrix where a cell uimj contains the rating rij given by user ui on movie mj.

We have some training data, where some ratings are given. Our task is to fill the rest of the matrix.

It can be seen as a regression problem. Here the yi's are the ratings. But xi's are not explicitly given as features.

We will use that matrix and somehow invent the features xi's.

So, now it can be treated as a regression problem as we have xi's and yi's.

Performance Metric :

1.) Mean Absolute Percentage Error
2.) Root Mean Square Error

We will minimize the RMSE.

We will try to provide some interpretability.

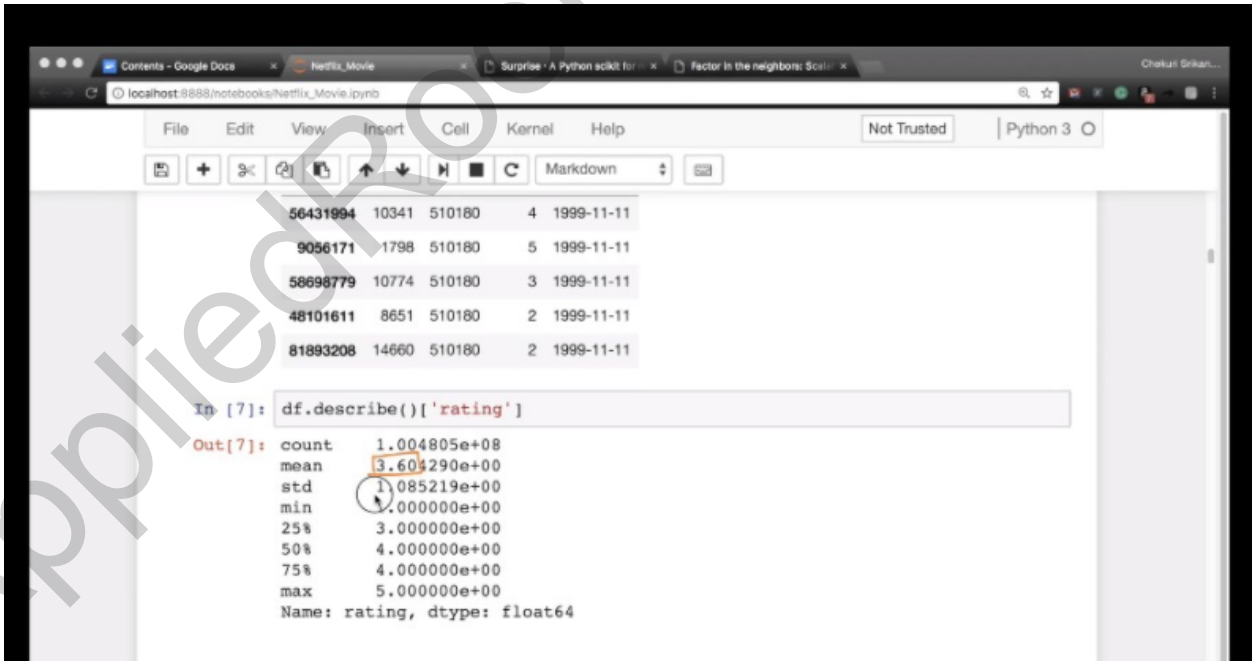
7.5 Exploratory Data Analysis:Data preprocessing

In the data.csv file , you will see the data in the format which is given below.

Movie_id,User_id,Rating,Date.

For example, 1,1488844,3,2005-09-06.

Some statistics about the ratings.



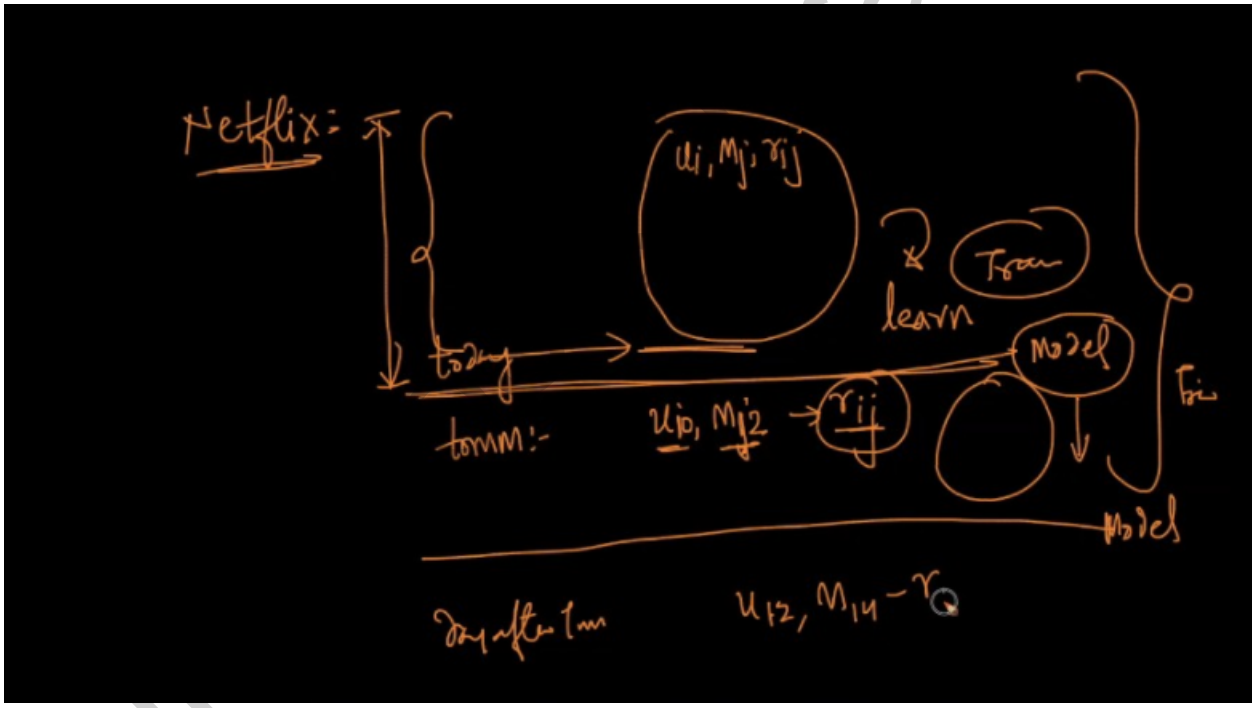
Timestamp : 04:42

The average rating is 3.6. This means on average the ratings are above and below 3.6.

Then , we remove the NaN and duplicate values.

Total number of ratings : 100480507
Total number of users : 480189
Total number of movies : 17770

7.6 Exploratory Data Analysis:Temporal Train-Test split.

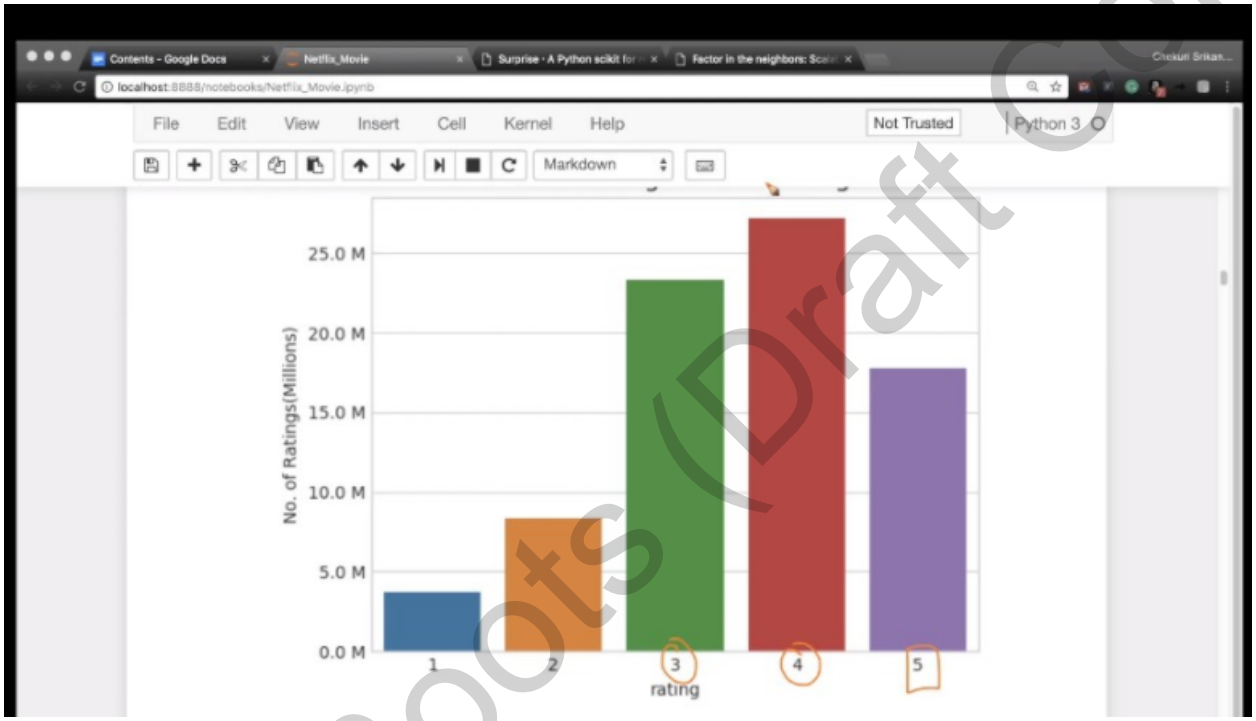


Timestamp : 02:23

As the ratings provided by the user ui on movie mj also depends on the time. In simple terms, the data has temporal nature. So, while splitting the data, we need to incorporate this.

So, basically sort the data in ascending order of dates i.e, the oldest date first. Then simply split in the ratio of 80:20.

7.7 Exploratory Data Analysis:Preliminary data analysis.

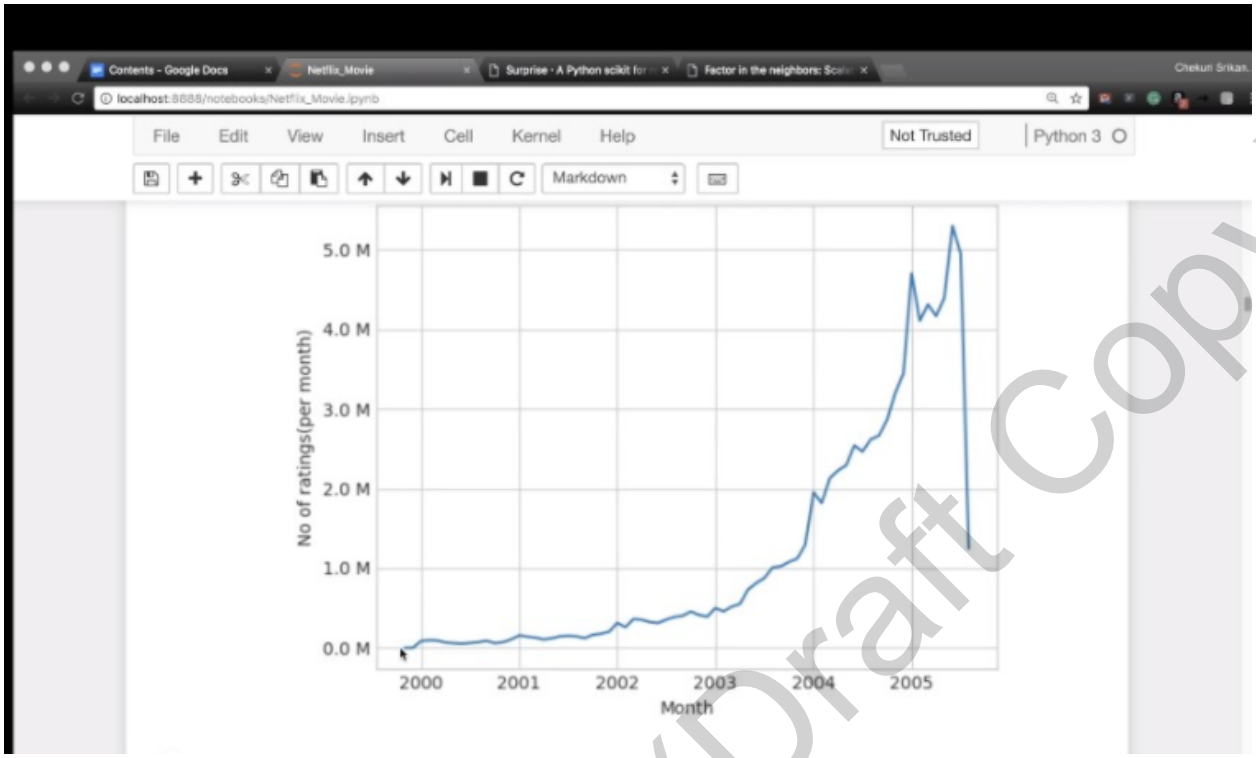


Timestamp : 00:52

Most of the users gave a rating of 4. This is clearly evident as the average rating is 3.6 which we saw before.

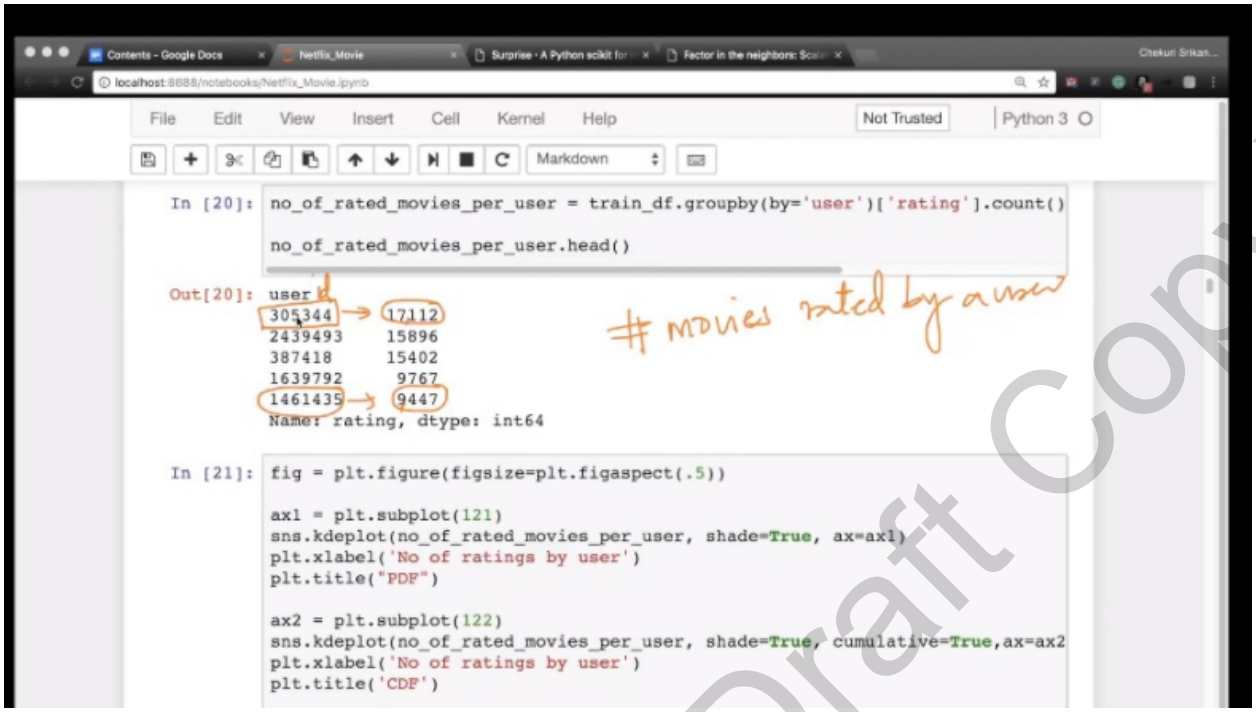
Only very few of the users gave a rating of 1.

So, typically users tend to give higher ratings.



Timestamp : 01:54

The ratings tend to increase as we progress through the years and then flattens out. This means,netflix as a company gave better content over the years as users liked it much.

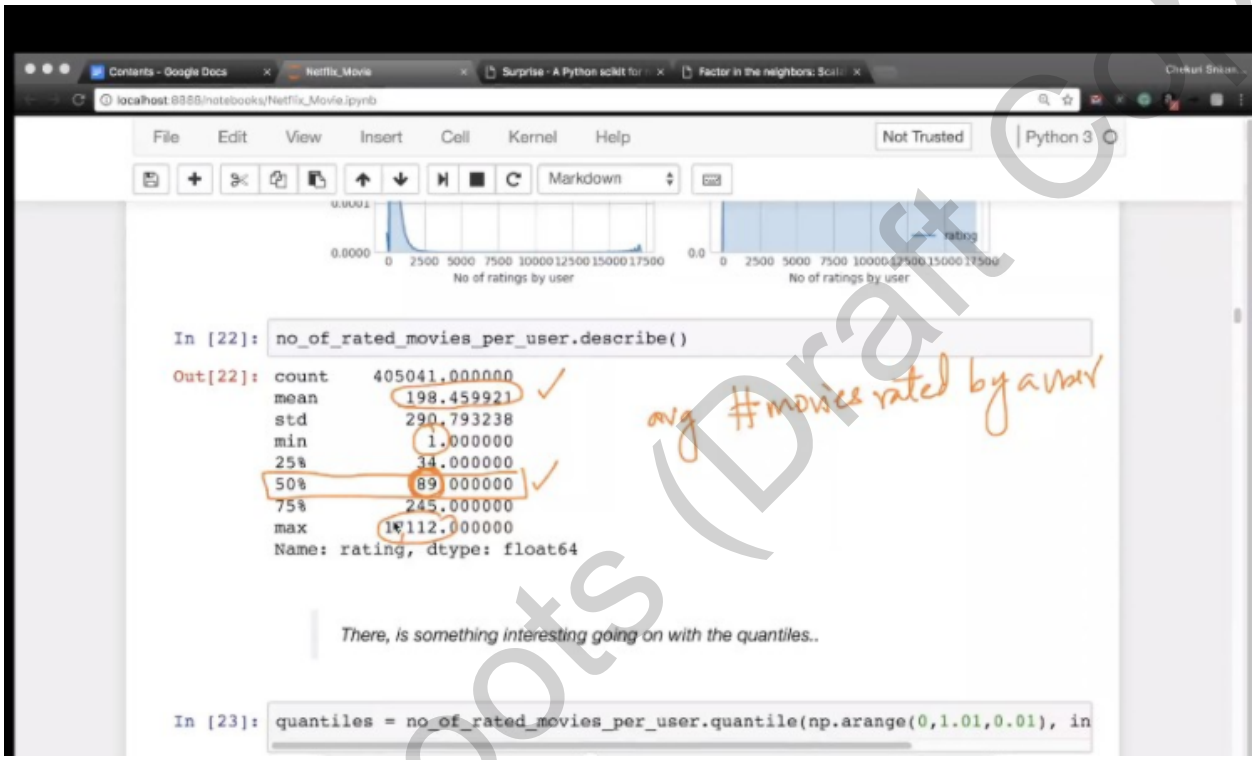


Timestamp : 04:30

The above data shows how many movies are rated by a user. The first record seems to be abnormal as the user rated for 17,112 movies.

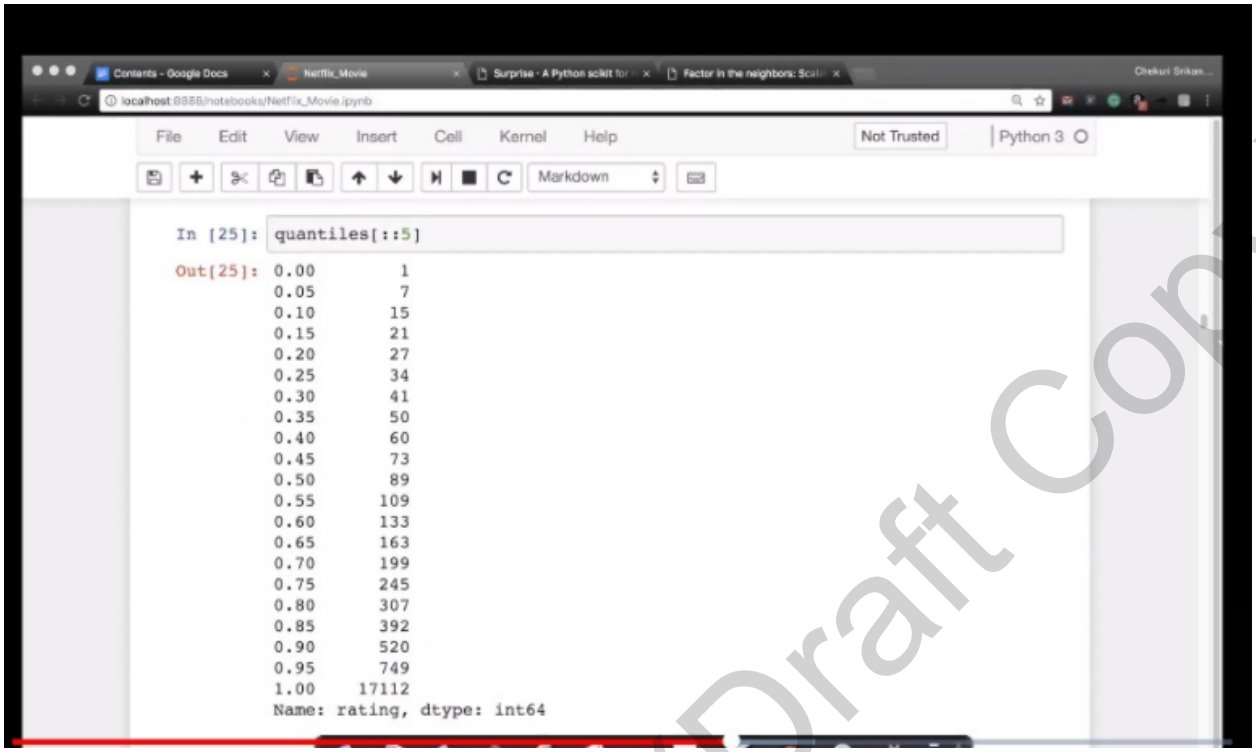
Timestamp : 05:02

The above is the pdf where the x axis represents the number of ratings by user and the y-axis represents the frequency of it. We can observe that only very few gave higher ratings.



Timestamp : 06:41

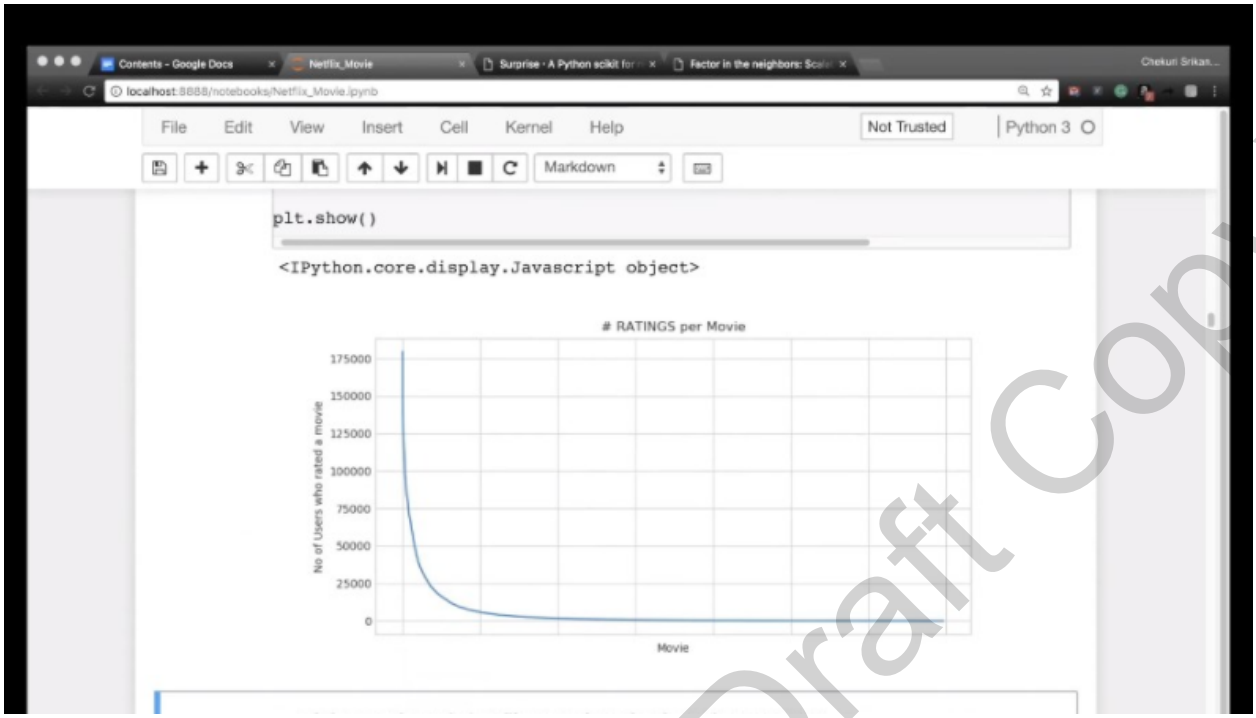
Some of the statistics are shown above.



Timestamp : 08:29

We can see only 5% of the users gave the number of ratings more than 749 and the maximum is 17112 which is abnormal.

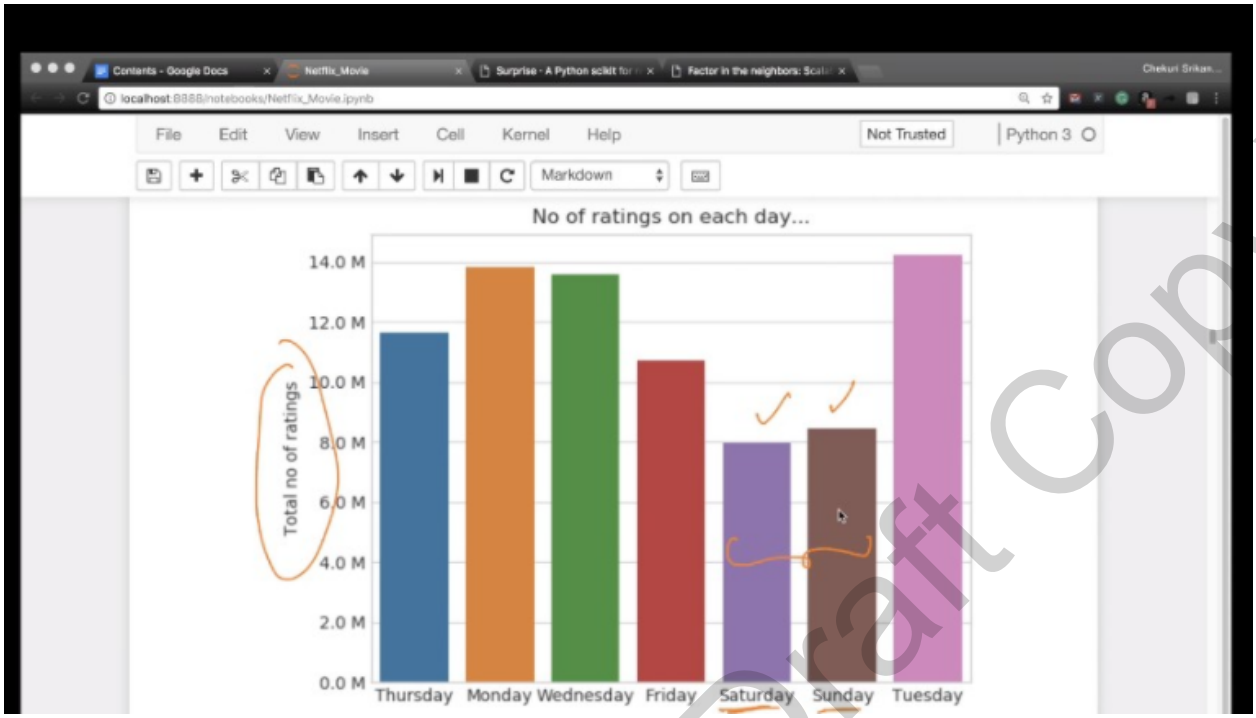
Analysis of ratings of a movie given by a user.



Timestamp : 09:47

X-axis : Movie_id and y-axis : Number of users who rated that movie.

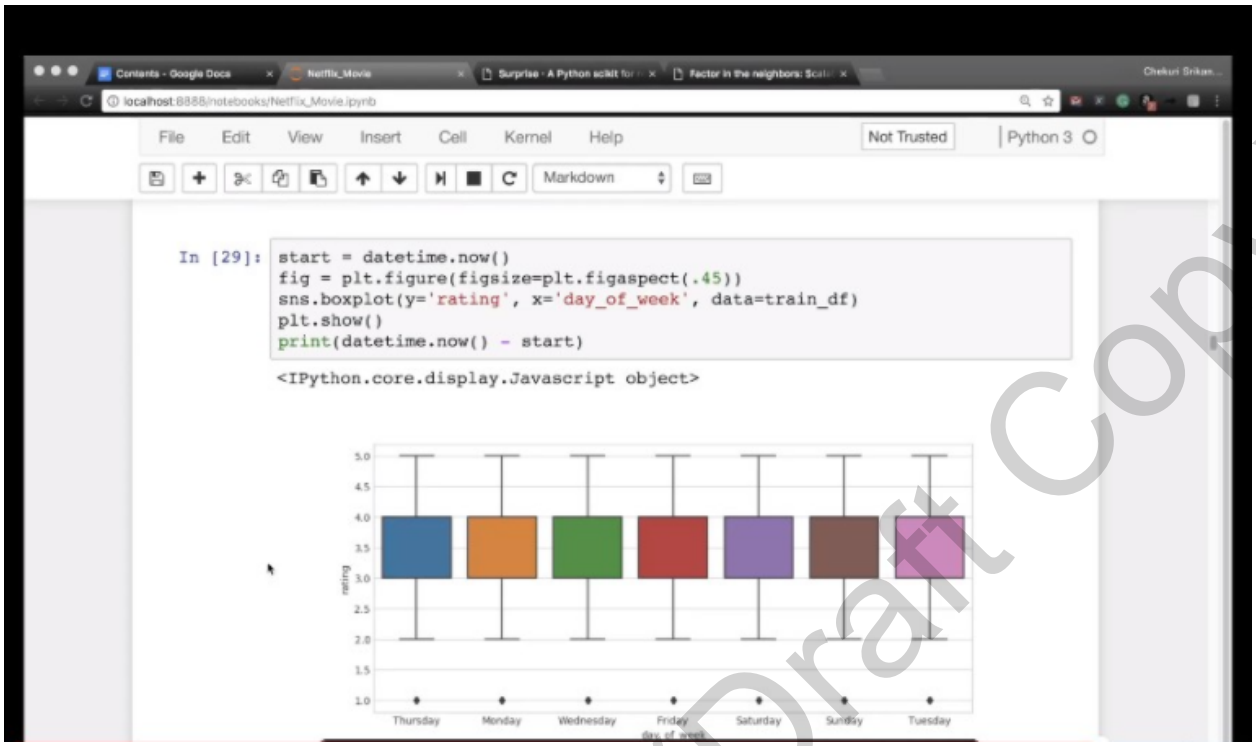
It’s a skewed distribution.



Timestamp : 11:24

Number of ratings on each day. On Tuesday, the users tend to rate more movies.

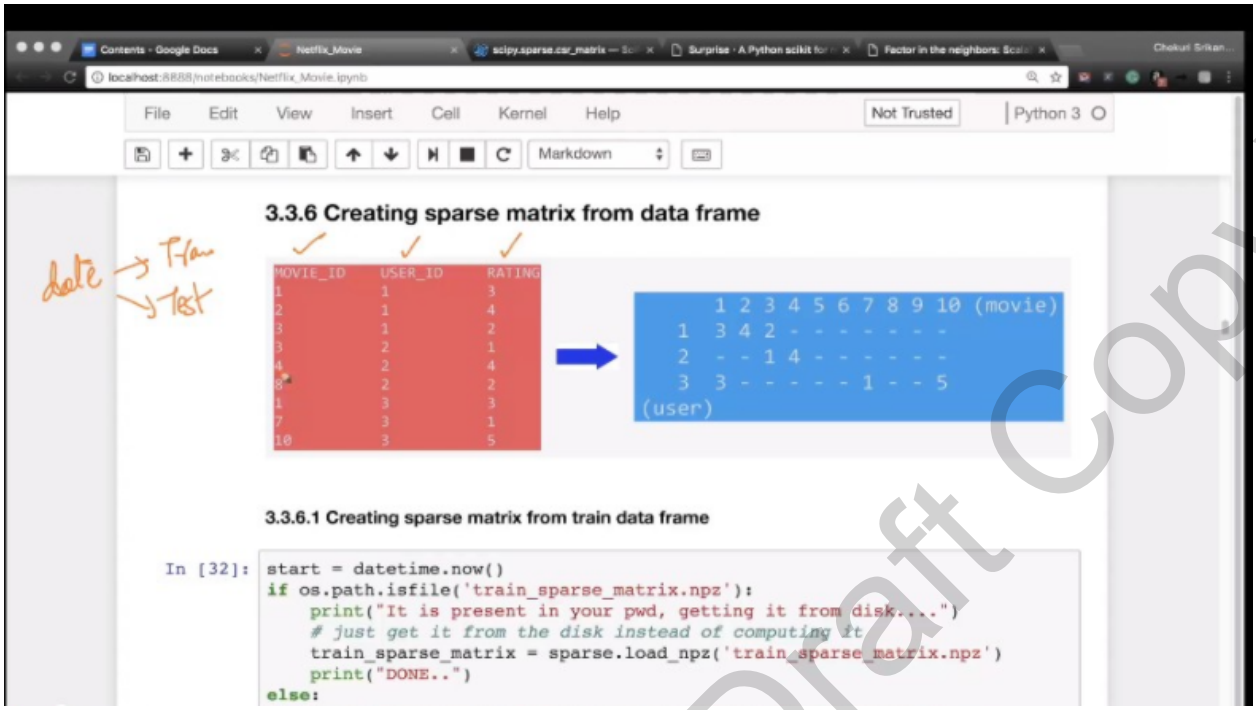
Can we use day of week for predicting the rating ?



Timestamp :12:45

As we can see from the box plot above, it's not a good feature to work on.

7.8 Exploratory Data Analysis:Sparse matrix representation

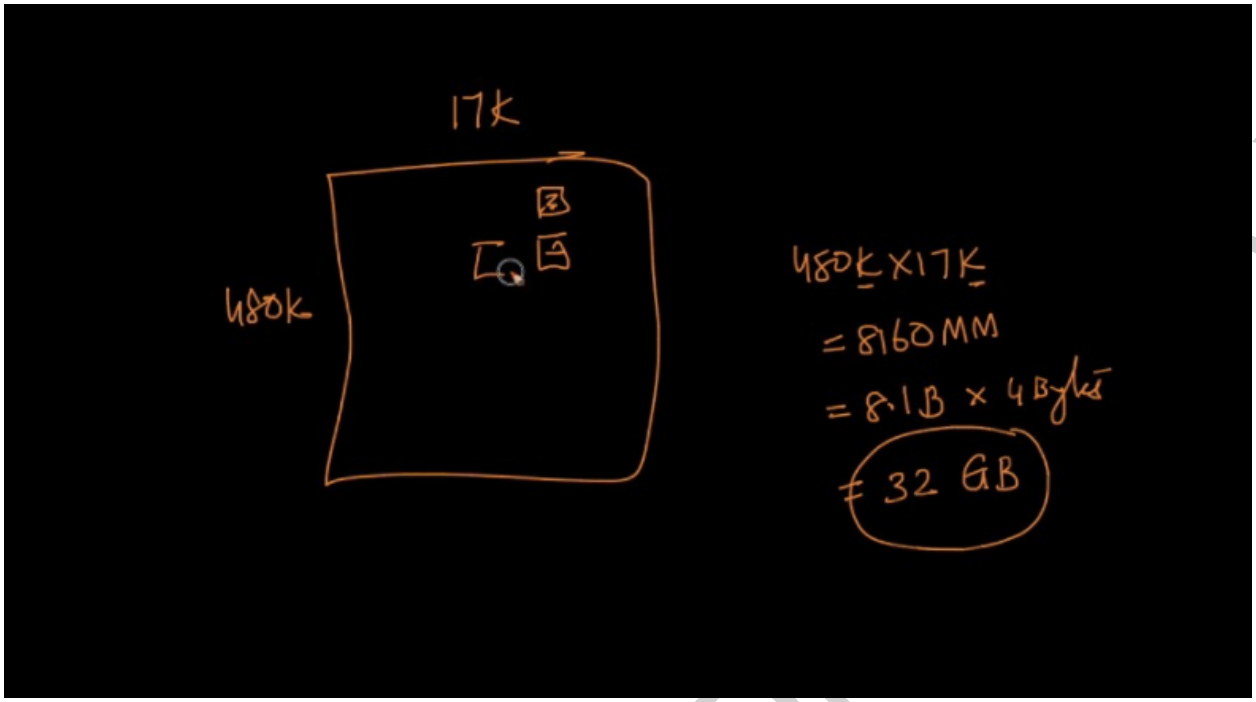


Timestamp : 0:18

We are simply converting it to a matrix where each cell represents the rating given by a user ui on movie mj. This is a sparse matrix since some of the entries are missing. This is because a particular user can't give ratings to all movies. The user can only give ratings to a subset of movies.

Using the csr_matrix() function, we can convert it to a compressed sparse matrix.

The data is extremely sparse.



Timestamp : 03:36

Loading the entire matrix requires 32 GB of memory. But, we know that some of the entries are not filled. So, we can use this property and store and retrieve it much more efficiently.

7.9 Exploratory Data Analysis:Average ratings for various slices

1.) Global mean of all ratings.
2.) User averages : For each user, compute the average of the ratings provided by that user.
3.) Movie averages : For each movie, compute the average of the ratings.

Timestamp : 05:13

The above resembles an approximation of gaussian distribution.

7.10 Exploratory Data Analysis:Cold start problem

Since the data is temporal in nature, we splitted accordingly. But there is a catch. There may be some users who are present in the training data but not in the test data. For these users, we don't have any information. Note : New movies are going to be released and new users also sign up on the website.

There are a total of 480189 users.

Number of users in train data : 405041

Number of users in test but not in train : 75,148 (Roughly 15% of the data).

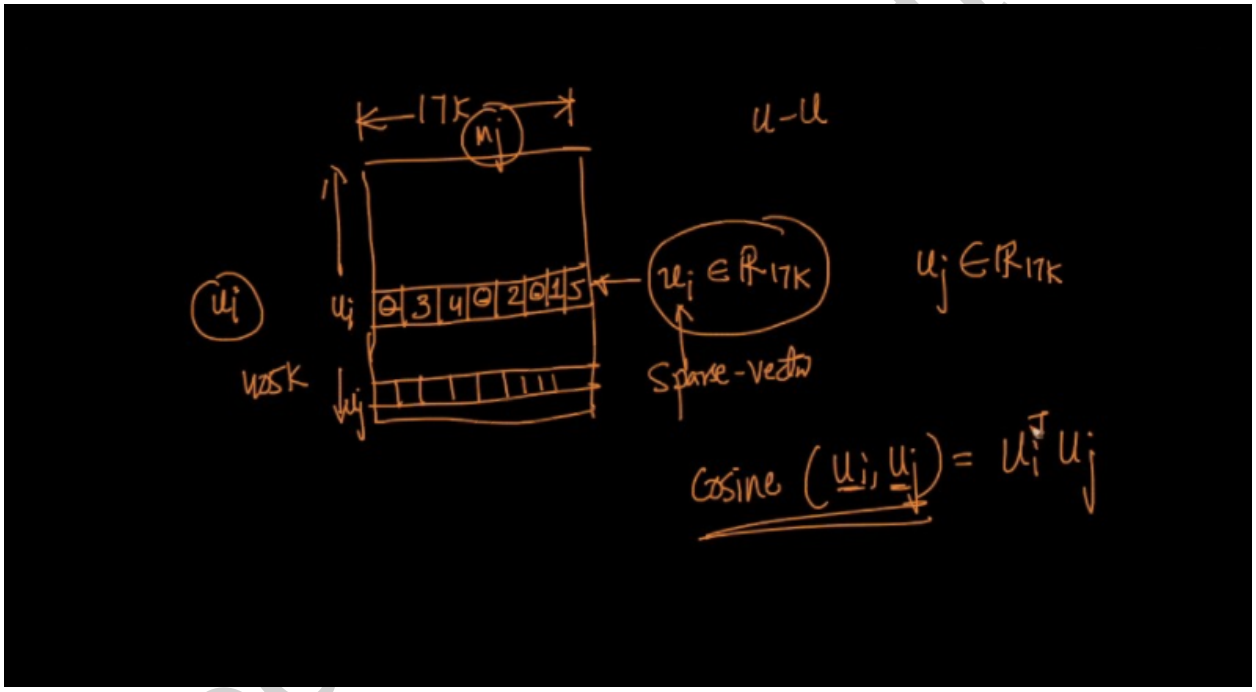
There are a total of 17,770 movies.

Number of users in train data : 17424

Number of users in test but not in train : 346 (Roughly 1.95 % of the data).

7.11 Computing Similarity matrices:User-User similarity matrix

Computing the user-user similarity matrix is hard and computationally infeasible as we have more number of users.



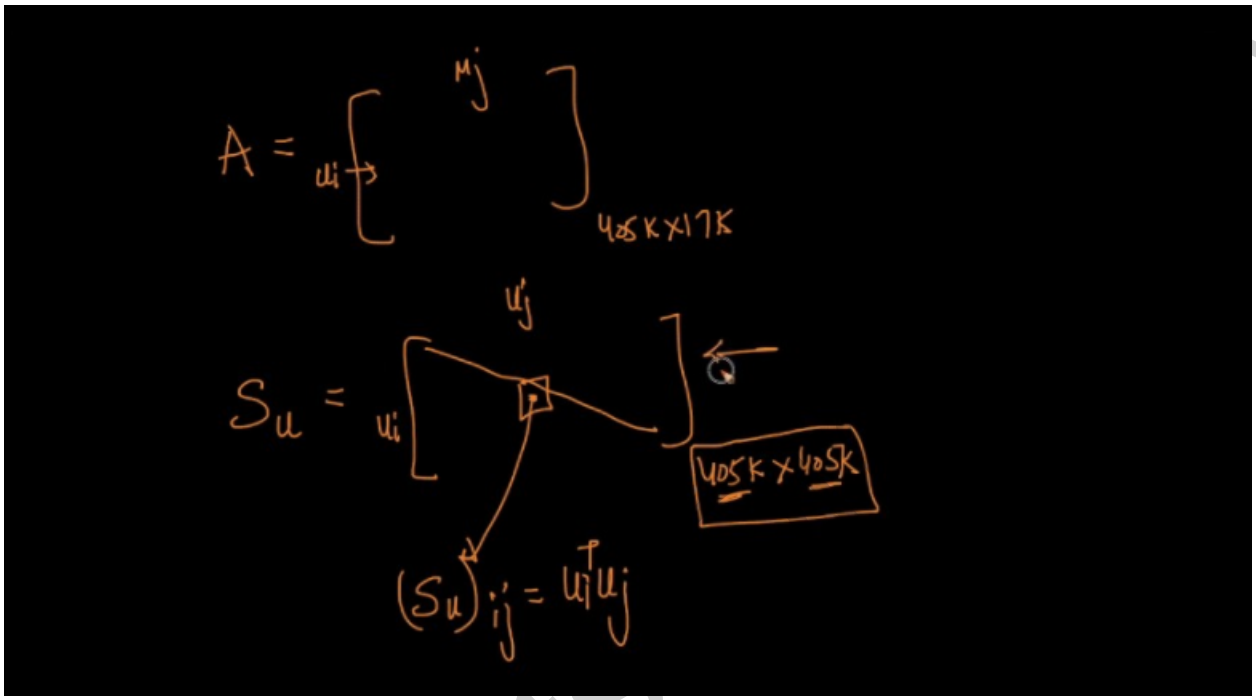
Timestamp : 02:26

Ui belongs to a 17k dimensional space. Uj belongs to a 17k dimensional space.

The similarity between Ui and Uj is defined as cosine(Ui,Uj) = UiTUj (dot product).

Both Ui and Uj are sparse vectors.

User-User similarity matrix. There are 405k users. So, the similarity matrix would be of shape 405k*405k since for every pair of users, we need the similarity value.



Timestamp : 04:06

The above S matrix is the user-user similarity matrix. Each cell uiuj contains the similarity value between ui and uj. This matrix is not sparse.

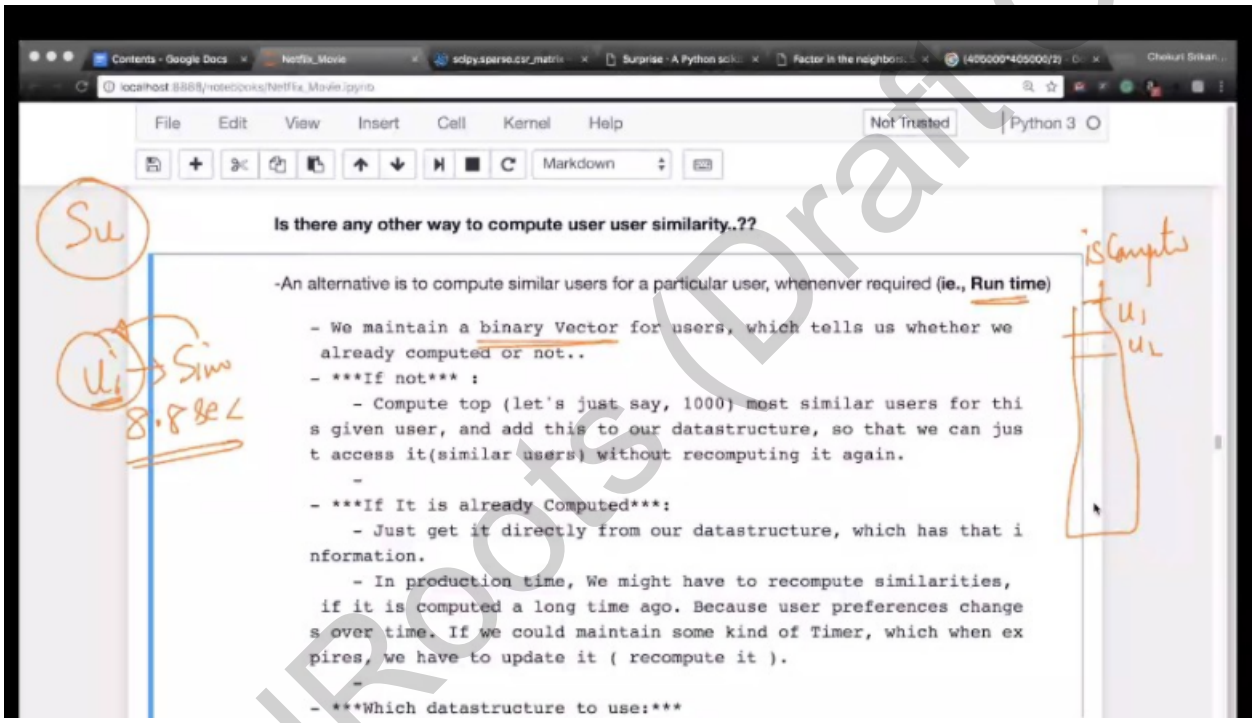
The above matrix is symmetric i.e, Suiuj = Sujui i.e, the value in the cell uiuj is equal to the value in the cell ujui. This is because cosine(uiuj) = cosine(ujui). So, computing Suiuj is enough and we don't need to compute Sujui. So, there are 405k*405k/2 values we need to compute.

We can parallelize the operation of computing the similarity matrix. But still,it's computationally infeasible.

Idea : The user vector belongs to 17k dimensional space. Can we use any dimensionality reduction techniques to reduce the dimensionality ? But still , it doesn't work.

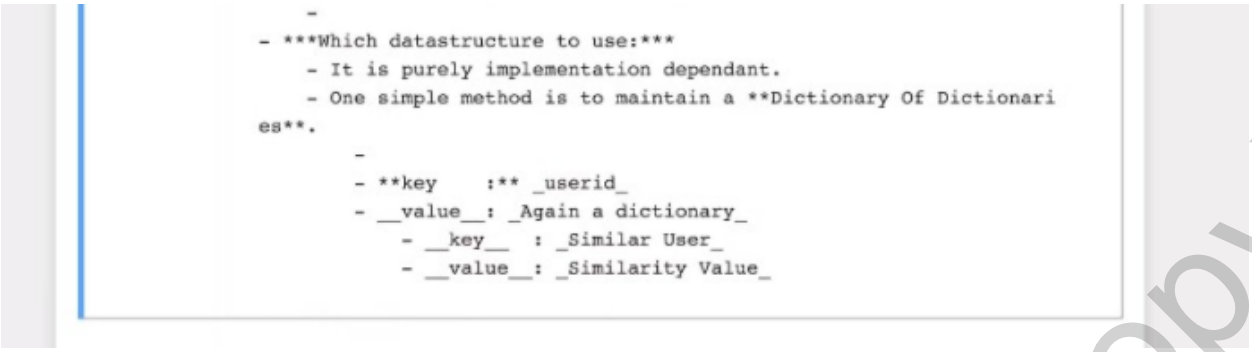
Why ?
Suppose if we reduce the dimension of a user vector from 17k to 500. Then, the values in the vector would be dense rather than sparse. So, we still need to do 500 multiplication and addition in the computation of dot product. Earlier, the user vector with 17k was sparse, so some of the entries were zero. This reduced the computation.

An alternative approach is as follows :



Timestamp : 16:50

Which data structure to use ?



Timestamp : 17:14

7.12 Computing Similarity matrices:Movie-Movie similarity

Each mi (movie vector) is of 405k dimensions and it's sparse too. We have a total of 17k movies. So, the shape of the similarity matrix is 17k * 17k.

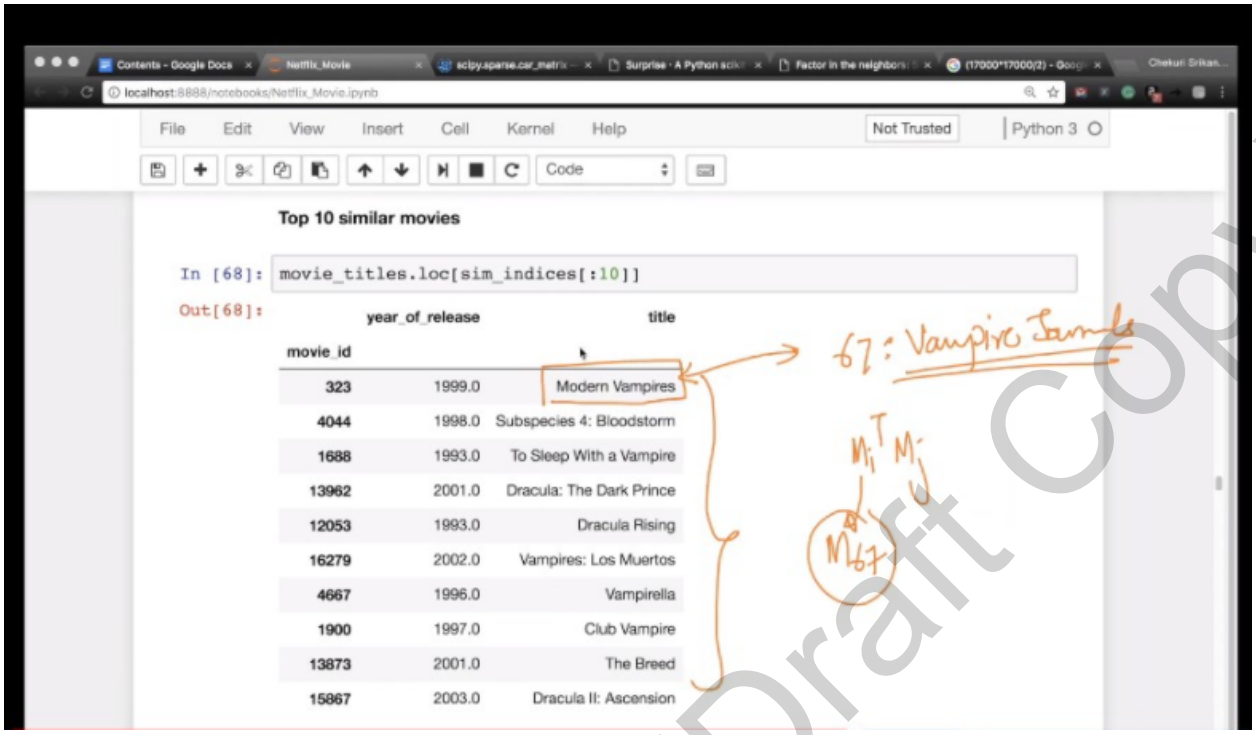
It's a symmetric matrix. So, we need to perform 17k *17k/2 values.

How to find top similar movies to a particular movie ?

It's simple to find. Let's say we want to find top similar movies to movie id 3. We have the similarity matrix right ? Take the 3rd row from the similarity matrix. This row contains the similarity values between movie id 3 and all other movies. Now, sort it in descending order. Now, choose the top k values. So, these movies are most similar to movie id 3.

7.13 Computing Similarity matrices:Does movie-movie similarity work?

Yes, it works.

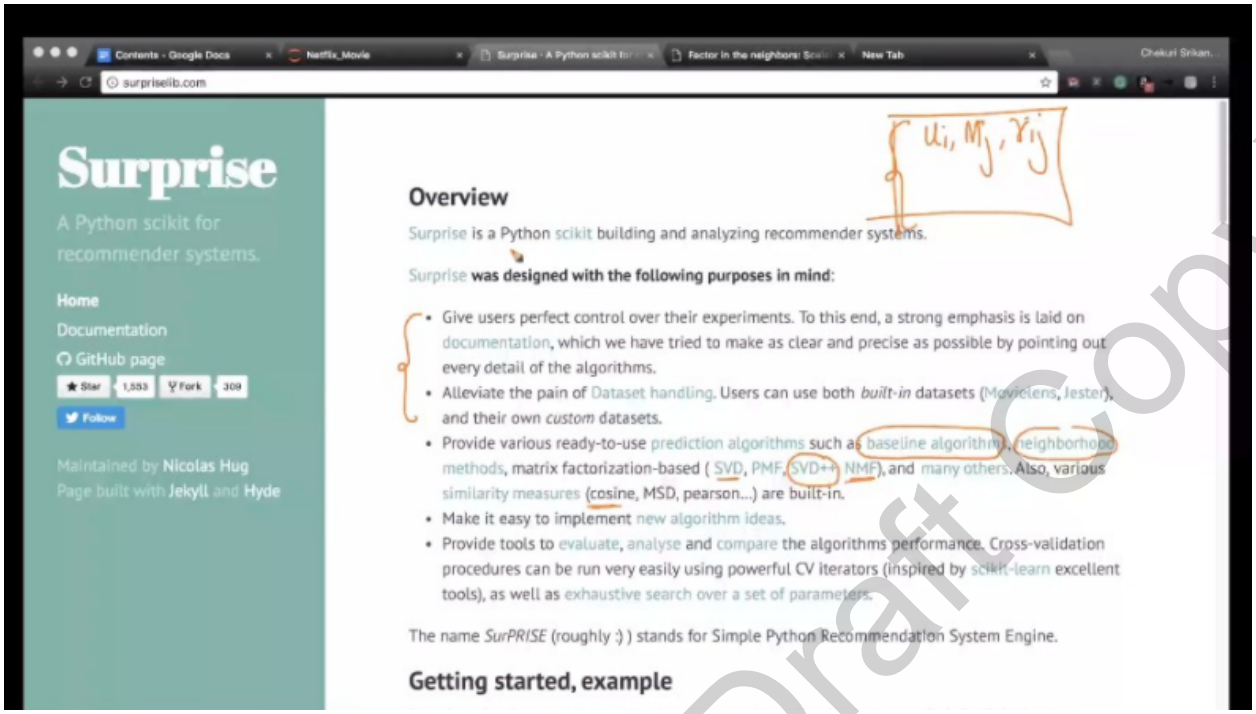


Timestamp : 03:59

We are printing the top similar movies to Vampire Journals. We can see that those sets of movies are very similar.

7.14 ML Models:Surprise library

Surprise is a Python scikit building and analyzing recommender systems.

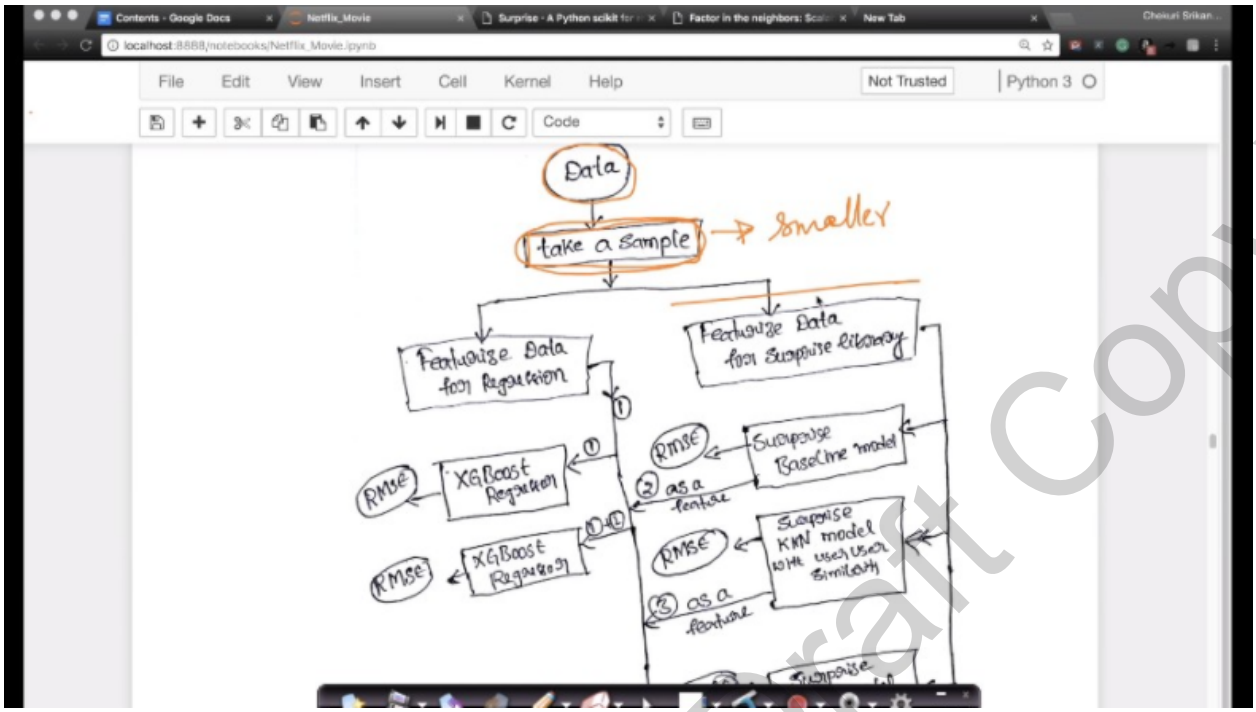


Timestamp : 01:34

Surprise stands for Simple Python Recommendation System Engine.

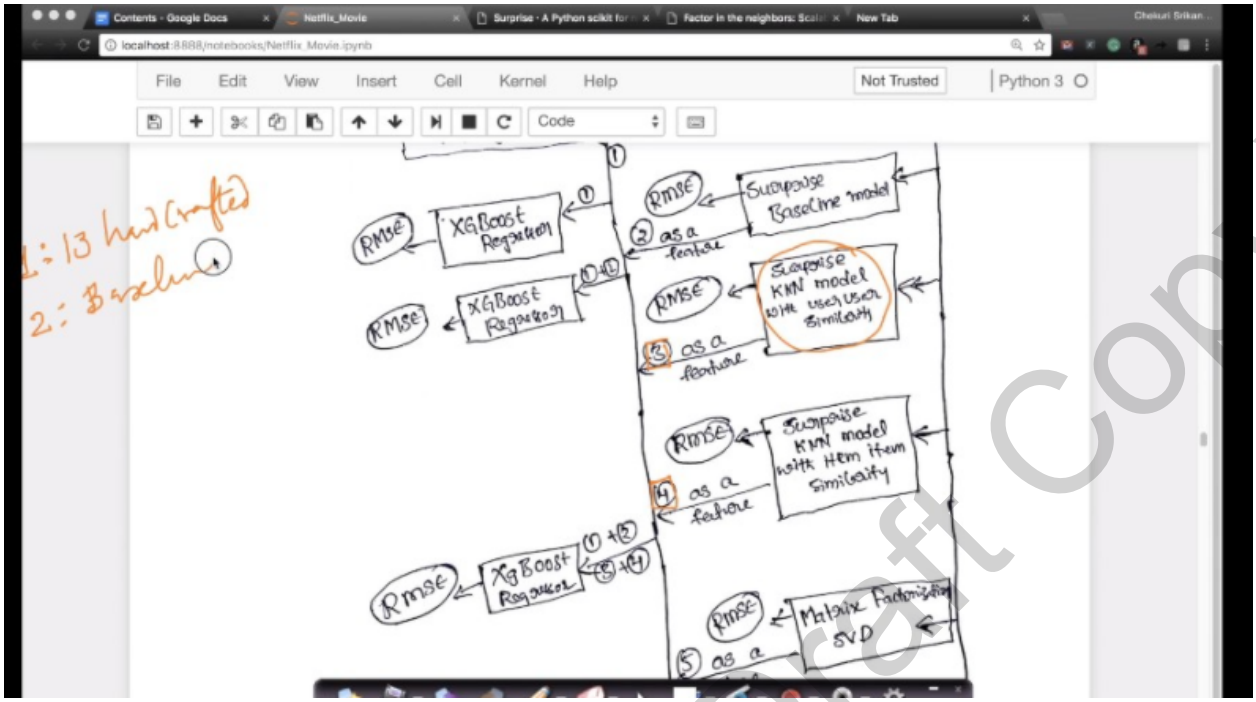
pip install numpy
pip install scikit-surprise (For installing the library)

7.15 Overview of the modelling strategy.



Timestamp : 00:46

1.) Take a subset of samples from the data as working with entire data is computationally infeasible.
2.) Come up with some features for regression algorithms.
3.) Also, in parallel, featurize the data for surprise baseline model,knn model with user user similarity and with item item similarity.
4.) The output predicted by the baseline model is also used for featurization in regression algorithms. This is similar to stacking of models.



Timestamp : 05:01

We also use SVD and SVD++.

7.16 Data Sampling.

Dataset is splitted into train and test data.

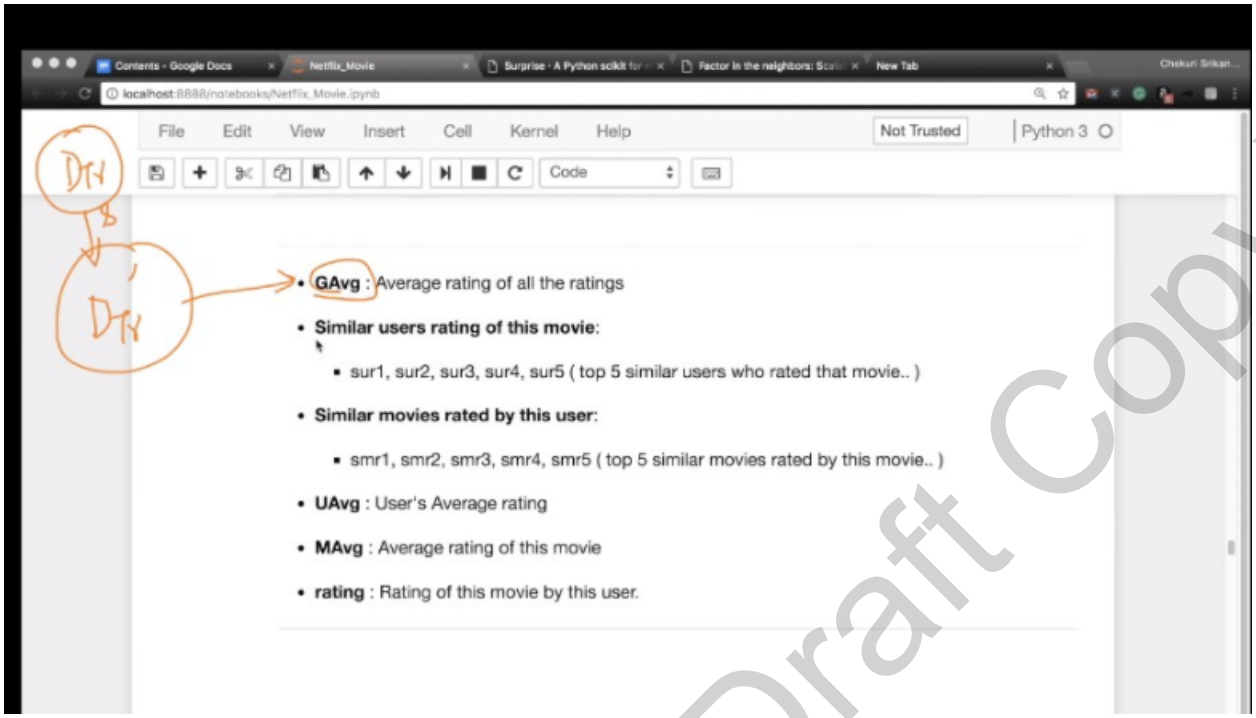
Train data : 405k * 17k
Test data : 348k * 17k

We take a subset of data from both train and test.

In train, we sampled 10k * 1k and from test we took 5k*500.

Let's build the models on the sampled data. We will find the best model out of it. Then, train the best model on the whole data.

7.18 Featurizations for regression.



Timestamp : 01:29

The above are self explanatory.

Sur1 ?

Let's pick a user ui. Now , find top k similar users to ui (Assume k = 5). Let those sets of users be u1,u2,u3,u4,u5.

Now, check whether u1 has watched mj or not. Similarly for others, perform the same.

Now, sur1 is the rating given by u1 on mj.

If any users have not watched movie mj, we don't consider them.

Similarly perform the same for movies too which is smr1,smr2,etc.

There are multiple ways people handle the Cold Start problem. Those are

1. You take the features of the movies based on its content and then evaluate the similar type of movies of the new user based on 2 to 3 movies he watched.

2. You recommend globally top movies initially to a new user who is a new user.

3. You try to show movies which are recently being popular from the region where your IP address is pointing to.

etc etc..

As the rest of the video lectures are only about the code sample discussion, we are not providing any notes for it. For any queries regarding the code samples, please feel free to post them in the comments section below the video lecture (or) you can mail us at mentors.diploma@appliedroots.com