CU INNOVATION HUB

Siam Innovation Distr

DESIGN THINKI

FOR BUSINESS INNOVATION

Kaweewut Temphuwapat
Team Lead, Innovation Lab and CVC,
PTT Group Design Leader,
MBA, Stanford University d.leader, Stanford d.school

pply Now – Aug 31, 2017 : Open for Public

t Course : SEP 9 – OCT 7, Every Saturday, 09.00 - 12.00

oem Rajakumari 60 Building (Chamchuri 10 Bldg), Fl 4

https://goo.gl/forms/C

ted Seats! Selected candidates will be fully-funded towards the course fee, worth 35,000 baht

NTACT US ☎ 093-725-0808 02-218-3106-7 www.facebook.com/cuihub @cuinnovationhub innovationh cu.innovati

CU INNOVATION HUB

MACHINE LEARNING

Siam Innovation District

Tech Talent

Machine Learning

- Essential tools and libraries: Python, Jupyter Notebook, NumPy, Pandas, SciPy, Scikit-Learn, MatPlotLib,and Seaborn
- Data collection through API and web scraping
- Machine Learning Algorithms reviews

Warodom Khamphanchai, Ph.D.
Bangkok AI Ambassador,
Ex-Software Developer at
Samsung SmartThings in Palo Alto,
CA. Ex-Full Stack Developer at

Sorawit Saengkyongam
Data Scientist at Agoda,
Google Developer Expert in
Machine Learning.

Agenda ครับ ------
1. Meet guest developers from Silicon Valle
2. What you need to know to convince your prospective CTO or developer?
3. How can you communicate your ideas to your CTO or developer?

ION

24 Steps to
uccessful Startup
Course

want to make happen? Join this comprehensive startup t will take you from idea to product in 6-weeks!

sh course to take your startup idea
Research through to Building your

tartup pitch deck and get ready for

artup building course from the MIT

MARTIN TRUST CENTER FOR MIT ENTREPRENEURSHIP

Tareef Jafferi
Ex-teaching staff at MIT,
Ex-Googler, Serial Entrepreneur

ug 31, 2017 : Open for Public

14, Every Saturday, 9.00 - 13.00

ding (Chamchuri 10 Bldg)., Fl 4

didates will be fully-funded towards the course fee, worth 35,000 baht

http://bit.ly/cueDE

93-725-0808 2-218-3106-7 www.facebook.com/cuihub @cuinnovationhub innovationhub.chula.ac.th cu.innovation.hub@gmail.com

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM

We will build a web-based wine review and recommendation system using Python technologies such as Django, Pandas, SciPy, and Scikit-learn.

Step 1
Start Django Web App

(tag : 1)

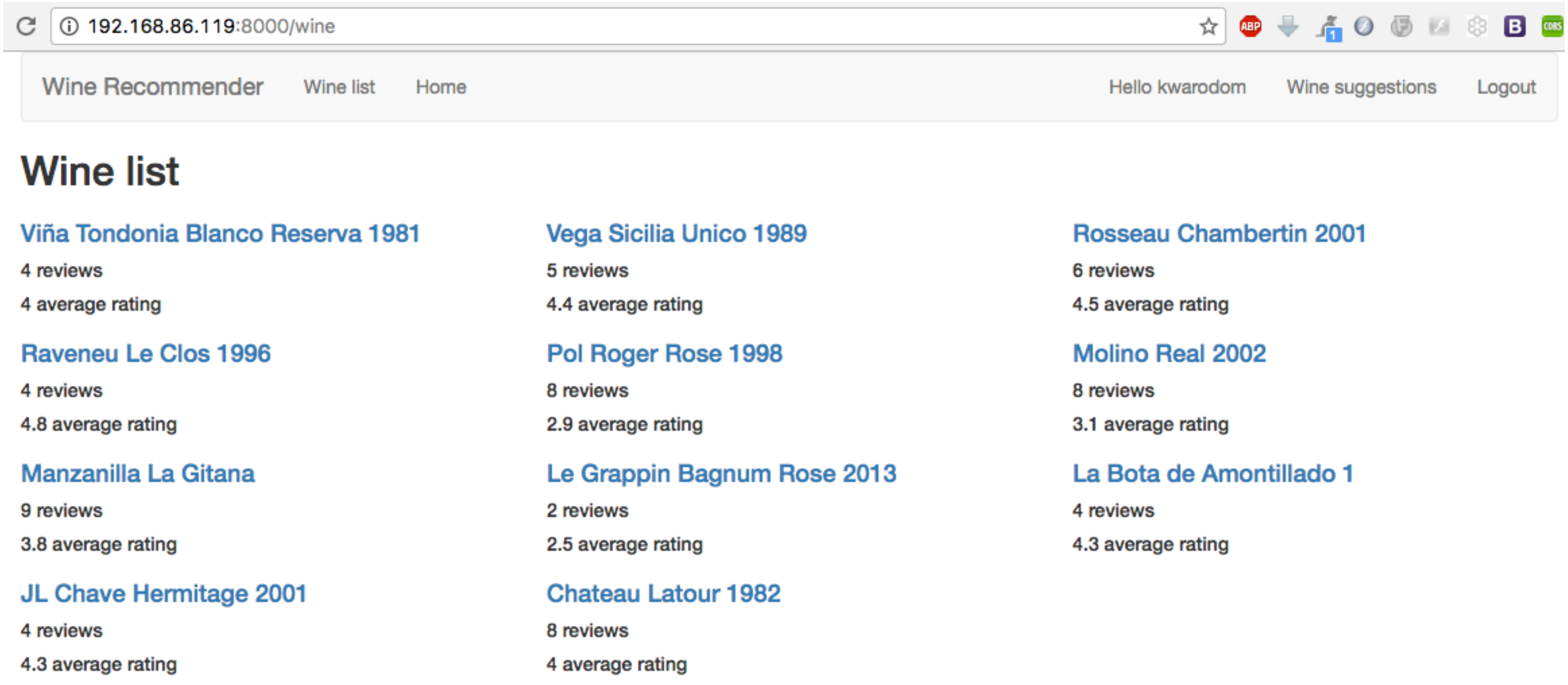
Step 2
Add User Management

(tag : 2)

Step 3
Generate Recommendations

(tag : 3)

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM



Wine list

Viña Tondonia Blanco Reserva 1981

4 reviews

4 average rating

Vega Sicilia Unico 1989

5 reviews

4.4 average rating

Rosseau Chambertin 2001

6 reviews

4.5 average rating

Raveneu Le Clos 1996

4 reviews

4.8 average rating

Pol Roger Rose 1998

8 reviews

2.9 average rating

Molino Real 2002

8 reviews

3.1 average rating

Manzanilla La Gitana

9 reviews

3.8 average rating

Le Grappin Bagnum Rose 2013

2 reviews

2.5 average rating

La Bota de Amontillado 1

4 reviews

4.3 average rating

JL Chave Hermitage 2001

4 reviews

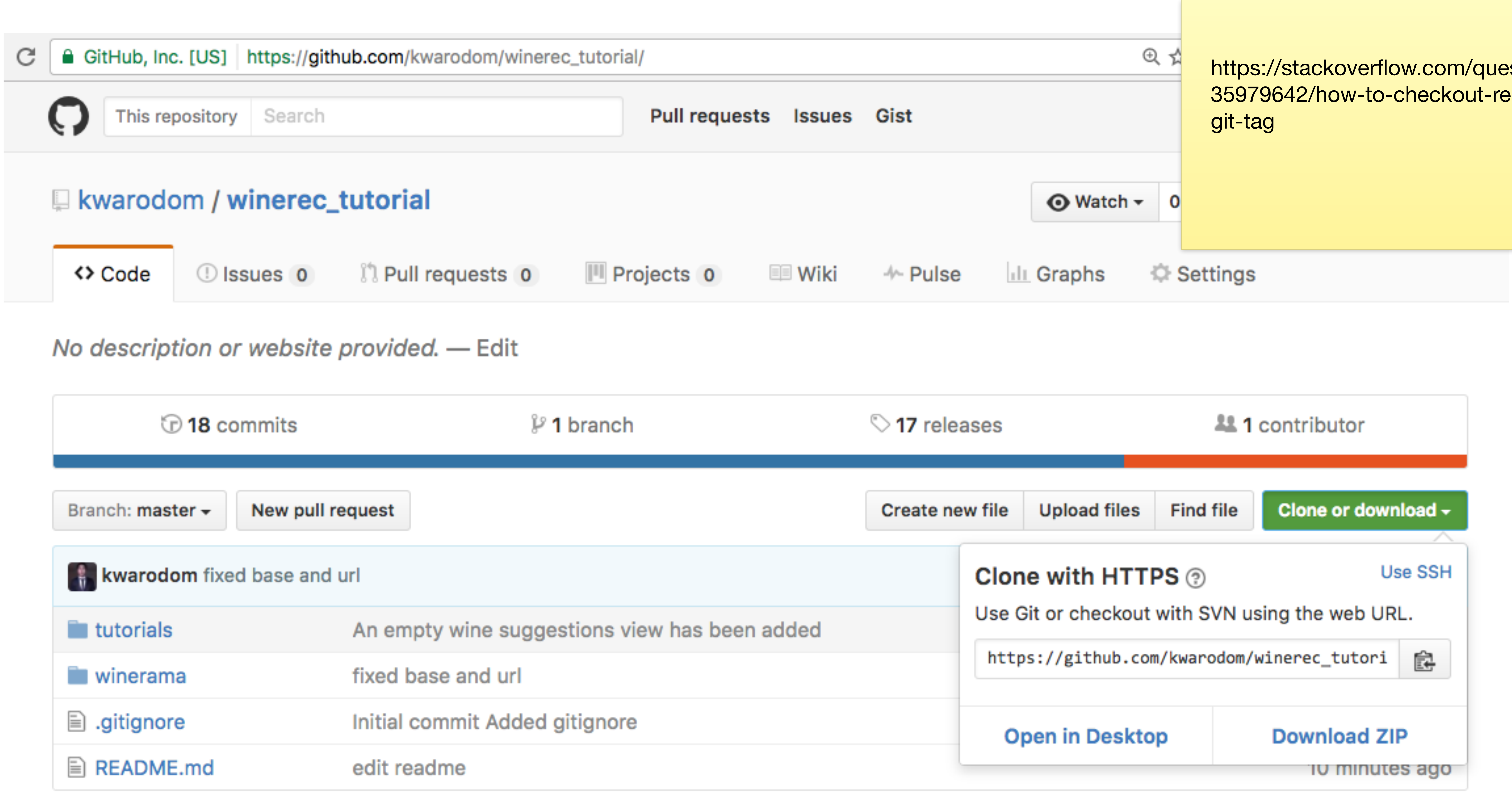
4.3 average rating

Chateau Latour 1982

8 reviews

4 average rating

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM



No description or website provided. — Edit

kwarodom fixed base and url

tutorials	An empty wine suggestions view has been added
winerama	fixed base and url
.gitignore	Initial commit Added gitignore
README.md	edit readme

10 minutes ago

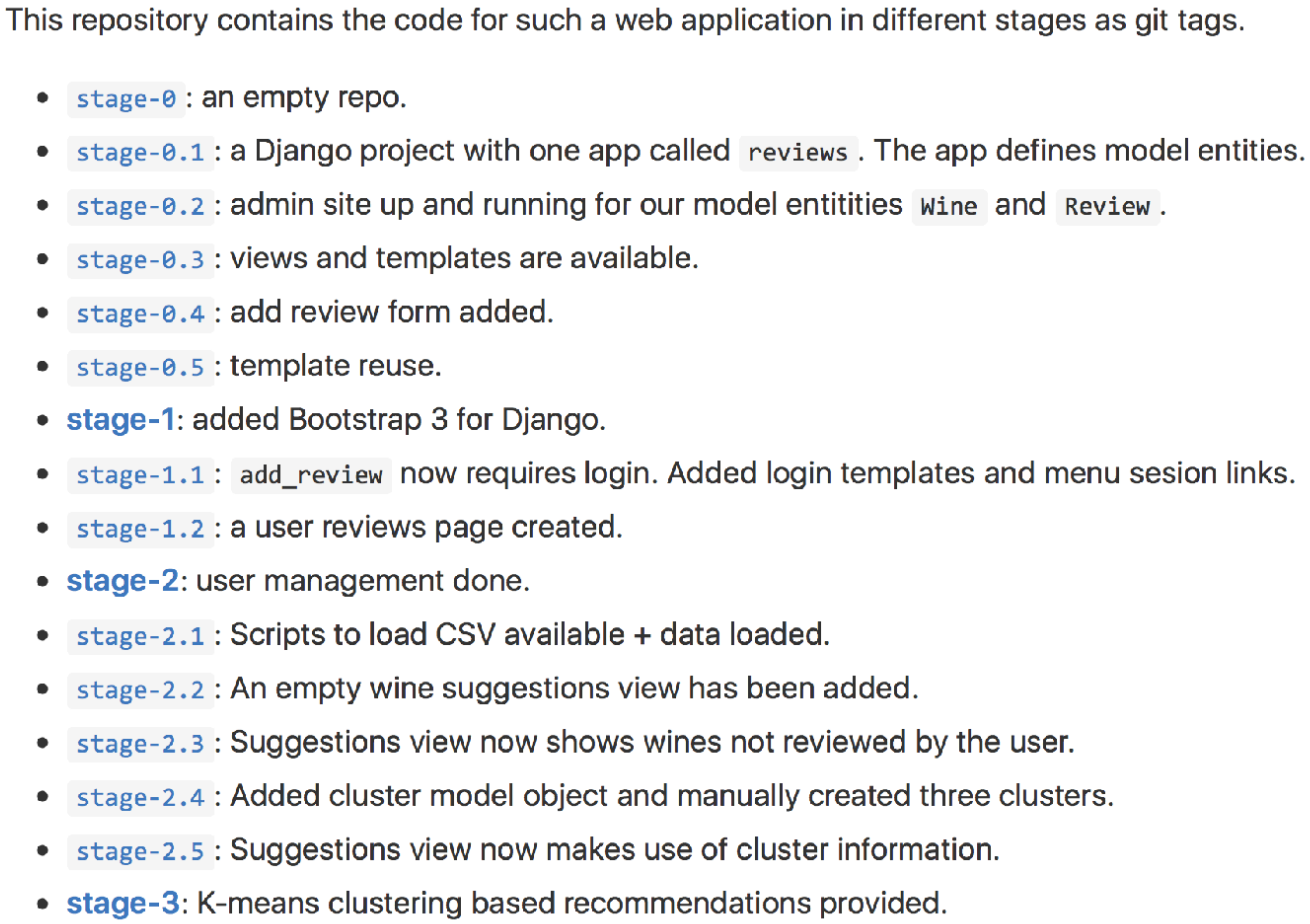
Use Git or checkout with SVN using the web URL.

```
https://github.com/kwarodom/winerec_tutori
```

https://stackoverflow.com/questions/35979642/how-to-checkout-remote-git-tag

At terminal type: git clone https://github.com/kwarodom/winerec_tutorial.git

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM

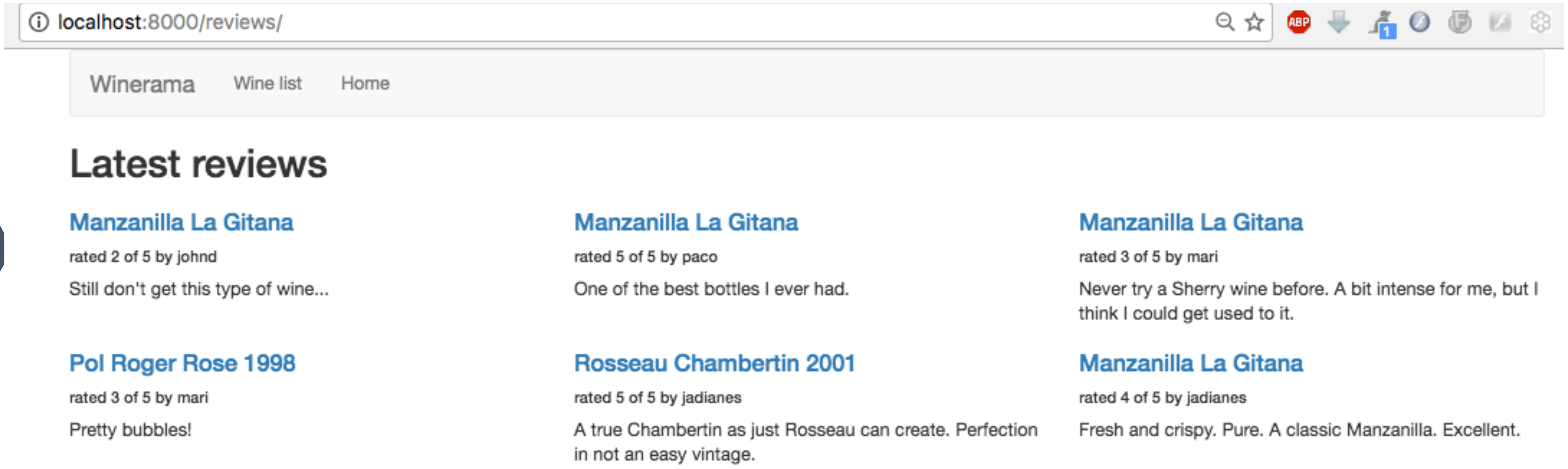


This repository contains the code for such a web application in different stages as git tags.

- `stage-0` : an empty repo.
- `stage-0.1` : a Django project with one app called `reviews` . The app defines model entities.
- `stage-0.2` : admin site up and running for our model entitities `Wine` and `Review` .
- `stage-0.3` : views and templates are available.
- `stage-0.4` : add review form added.
- `stage-0.5` : template reuse.
- **stage-1**: added Bootstrap 3 for Django.
- `stage-1.1` : `add_review` now requires login. Added login templates and menu sesion links.
- `stage-1.2` : a user reviews page created.
- **stage-2**: user management done.
- `stage-2.1` : Scripts to load CSV available + data loaded.
- `stage-2.2` : An empty wine suggestions view has been added.
- `stage-2.3` : Suggestions view now shows wines not reviewed by the user.
- `stage-2.4` : Added cluster model object and manually created three clusters.
- `stage-2.5` : Suggestions view now makes use of cluster information.
- **stage-3**: K-means clustering based recommendations provided.

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM

We will build a web-based wine review and recommendation system using Python technologies such as Django, Pandas, SciPy, and Scikit-learn.



Latest reviews

Manzanilla La Gitana
rated 2 of 5 by johnd
Still don't get this type of wine...

Manzanilla La Gitana
rated 5 of 5 by paco
One of the best bottles I ever had.

Manzanilla La Gitana
rated 3 of 5 by mari
Never try a Sherry wine before. A bit intense for me, but I think I could get used to it.

Pol Roger Rose 1998
rated 3 of 5 by mari
Pretty bubbles!

Rosseau Chambertin 2001
rated 5 of 5 by jadianes
A true Chambertin as just Rosseau can create. Perfection in not an easy vintage.

Manzanilla La Gitana
rated 4 of 5 by jadianes
Fresh and crispy. Pure. A classic Manzanilla. Excellent.

Step 1

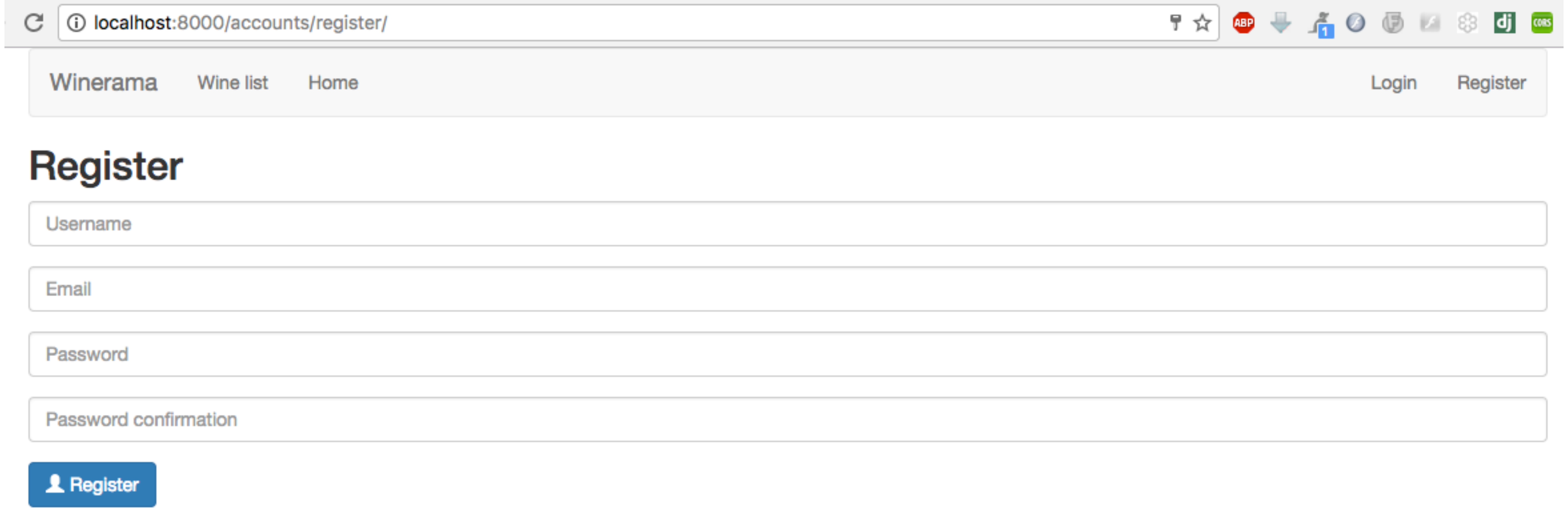
Start Django Web App

(tag : stage-1)

Start Django project and Django app for wine recommendation web

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM

We will build a web-based wine review and recommendation system using Python technologies such as Django, Pandas, SciPy, and Scikit-learn.

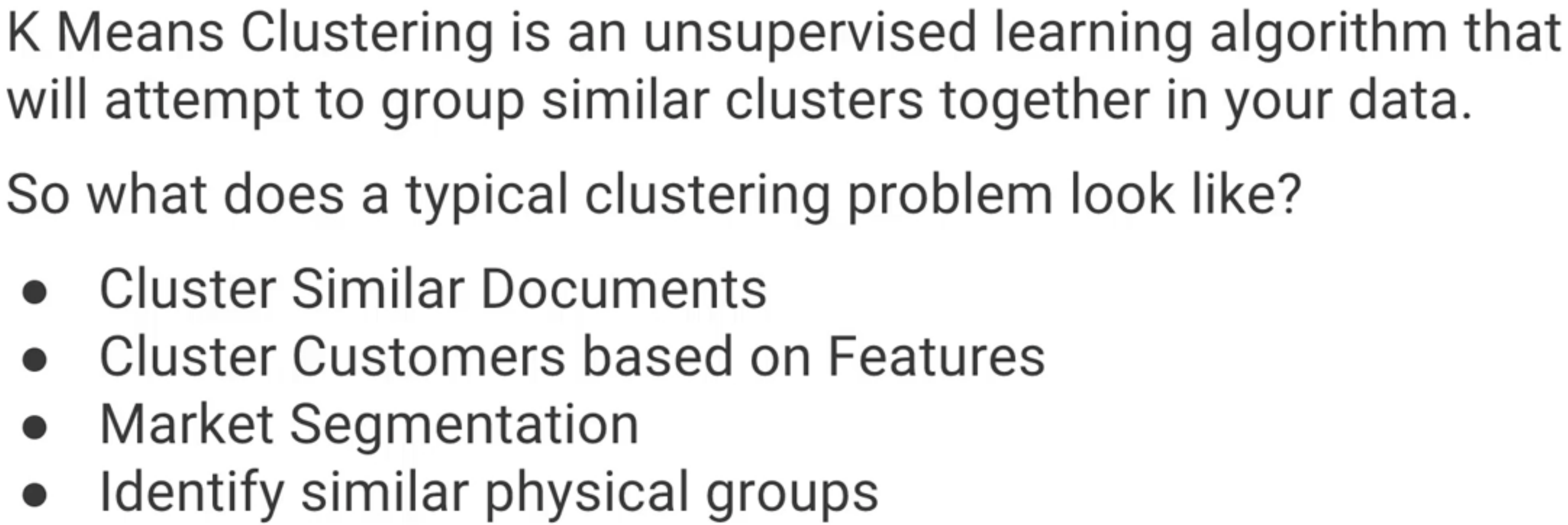


Step 2

Add User Management

(tag : stage-2)

Add user management and, once we have users identified, proceed to generate user recommendations using machine learning.

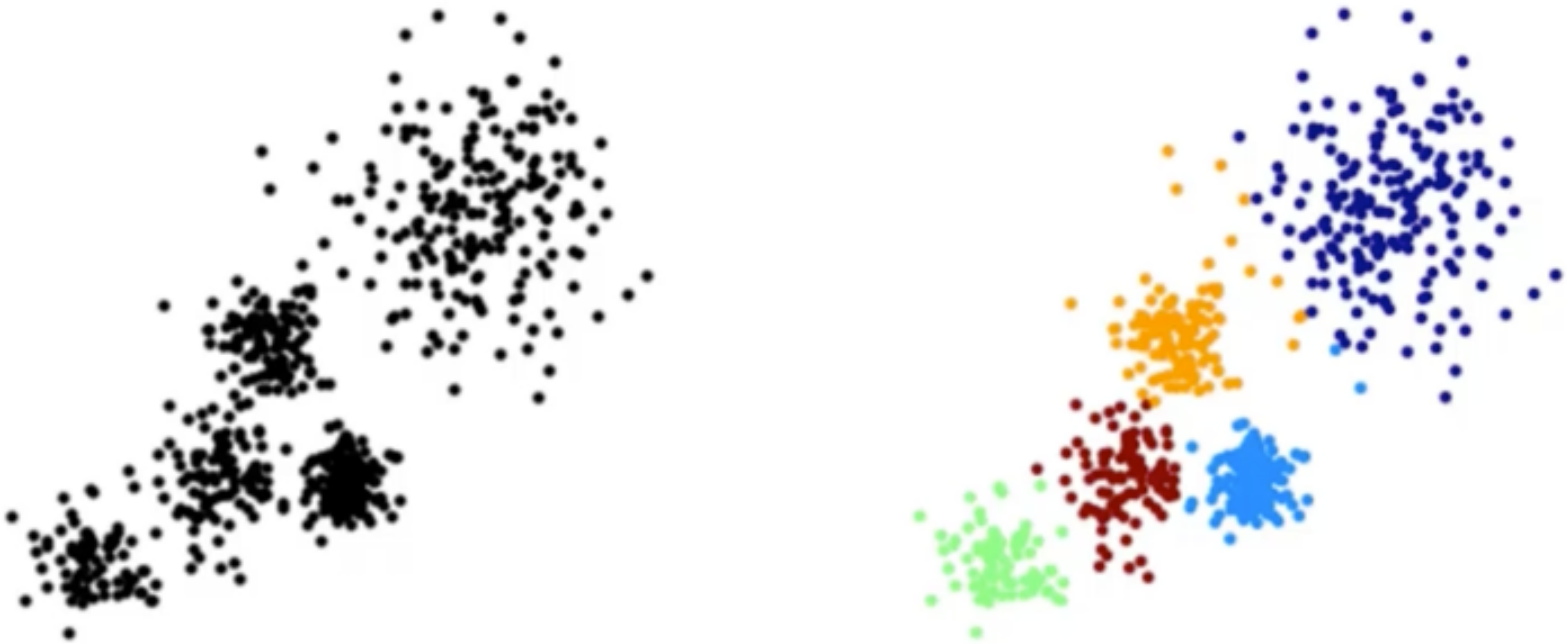


K Means Clustering is an unsupervised learning algorithm that will attempt to group similar clusters together in your data.

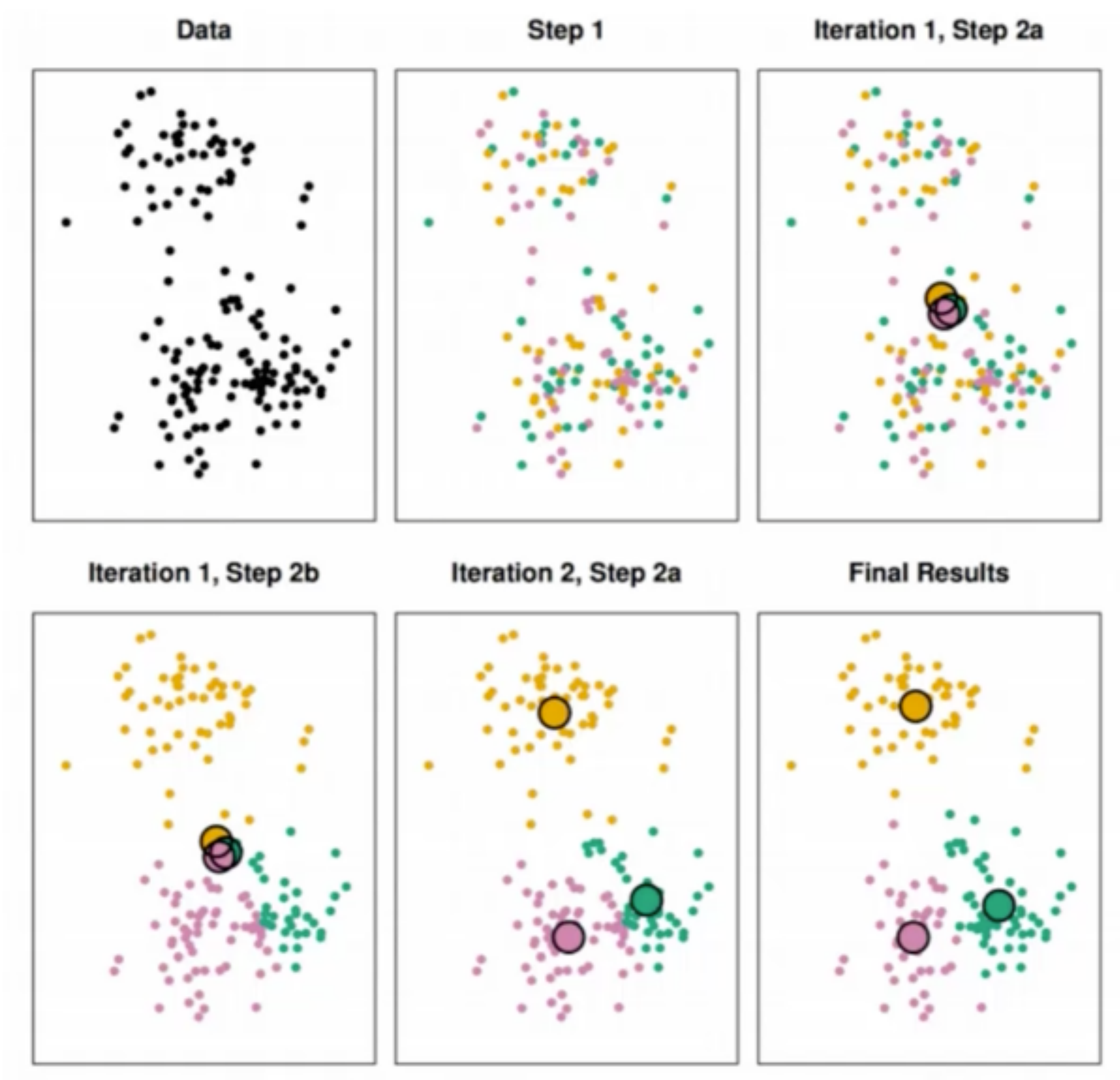
So what does a typical clustering problem look like?

- Cluster Similar Documents
- Cluster Customers based on Features
- Market Segmentation
- Identify similar physical groups

The overall goal is to divide data into distinct groups such that observations within each group are similar

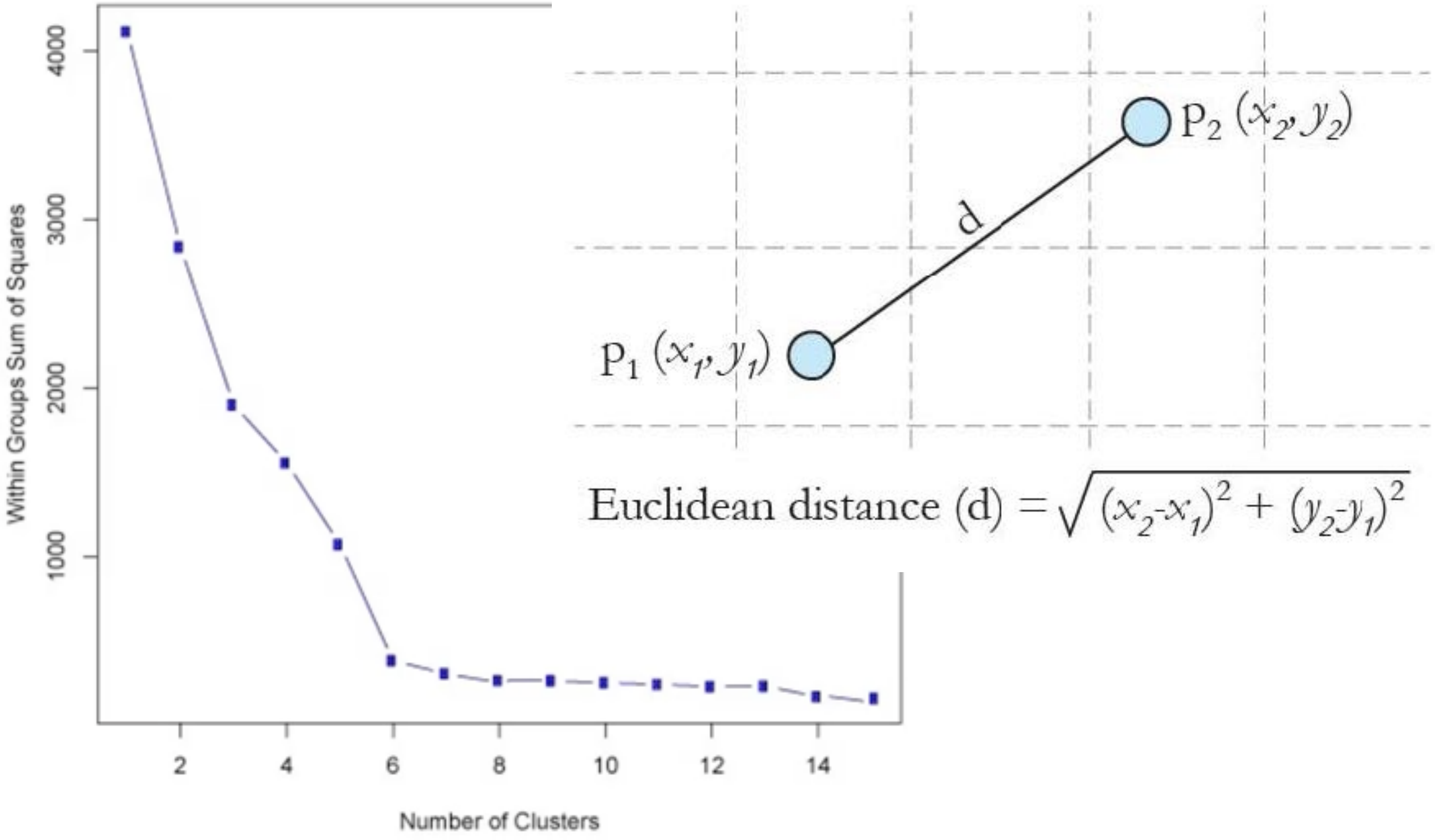


UNSUPERVISED LEARNING -> K-MEANS CLUSTERING



The K Means Algorithm

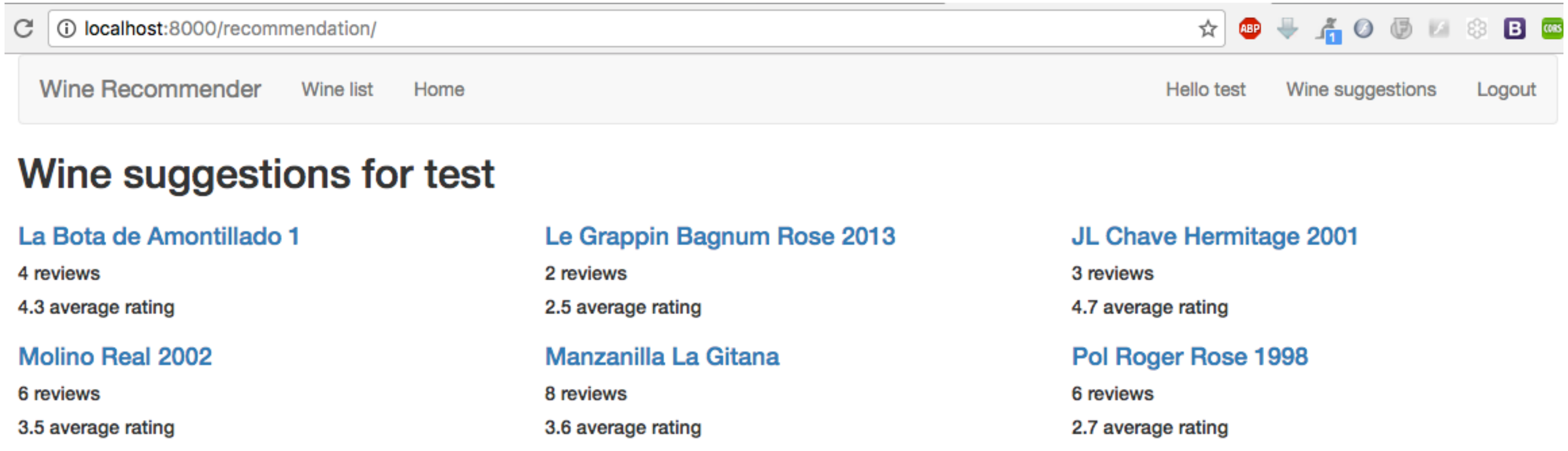
- Choose a number of Clusters “K”
- Randomly assign each point to a cluster
- Until clusters stop changing, repeat the following:
  - For each cluster, compute the cluster centroid by taking the mean vector of points in the cluster
  - Assign each data point to the cluster for which the centroid is the closest



Euclidean distance (d) $= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM

We will build a web-based wine review and recommendation system using Python technologies such as Django, Pandas, SciPy, and Scikit-learn.



Wine suggestions for test

La Bota de Amontillado 1
4 reviews
4.3 average rating

Le Grappin Bagnum Rose 2013
2 reviews
2.5 average rating

JL Chave Hermitage 2001
3 reviews
4.7 average rating

Molino Real 2002
6 reviews
3.5 average rating

Manzanilla La Gitana
8 reviews
3.6 average rating

Pol Roger Rose 1998
6 reviews
2.7 average rating

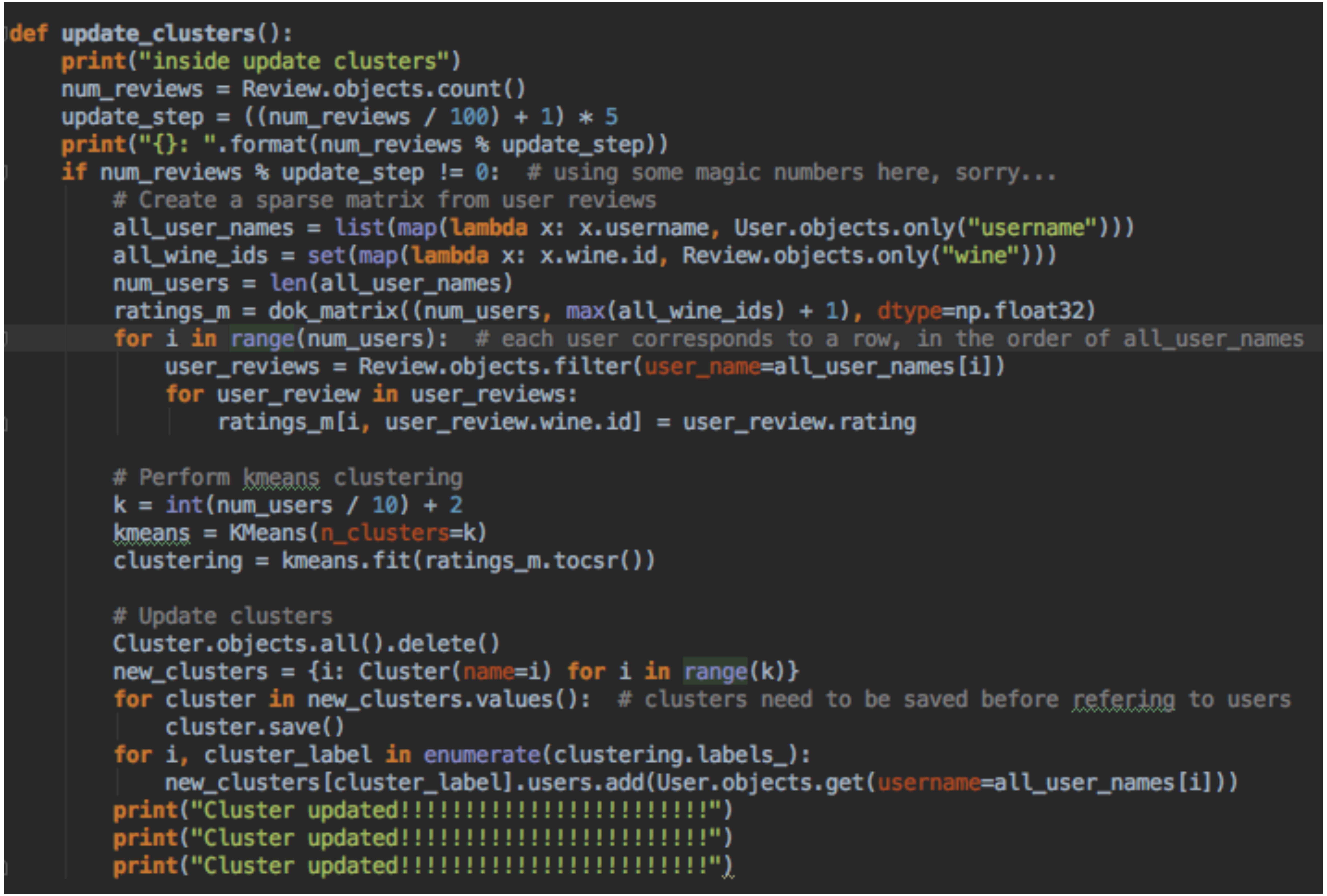
Step 3

Generate Recommendations

(tag : stage-3.3)

Use machine learning to provide wine suggestions to our users, using k-means clustering.

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM



```python
def update_clusters():
    print("inside update clusters")
    num_reviews = Review.objects.count()
    update_step = ((num_reviews / 100) + 1) * 5
    print("{}: ".format(num_reviews % update_step))
    if num_reviews % update_step != 0:  # using some magic numbers here, sorry...
        # Create a sparse matrix from user reviews
        all_user_names = list(map(lambda x: x.username, User.objects.only("username")))
        all_wine_ids = set(map(lambda x: x.wine.id, Review.objects.only("wine")))
        num_users = len(all_user_names)
        ratings_m = dok_matrix((num_users, max(all_wine_ids) + 1), dtype=np.float32)
        for i in range(num_users):  # each user corresponds to a row, in the order of all_user_names
            user_reviews = Review.objects.filter(user_name=all_user_names[i])
            for user_review in user_reviews:
                ratings_m[i, user_review.wine.id] = user_review.rating

        # Perform kmeans clustering
        k = int(num_users / 10) + 2
        kmeans = KMeans(n_clusters=k)
        clustering = kmeans.fit(ratings_m.tocsr())

        # Update clusters
        Cluster.objects.all().delete()
        new_clusters = {i: Cluster(name=i) for i in range(k)}
        for cluster in new_clusters.values():  # clusters need to be saved before refering to users
            cluster.save()
        for i, cluster_label in enumerate(clustering.labels_):
            new_clusters[cluster_label].users.add(User.objects.get(username=all_user_names[i]))
        print("Cluster updated!!!!!!!!!!!!!!!!!!!!!!!!")
        print("Cluster updated!!!!!!!!!!!!!!!!!!!!!!!!")
        print("Cluster updated!!!!!!!!!!!!!!!!!!!!!!!!")
```

Step 3

Generate Recommendations

(tag : stage-3.3)

LEARN TO BUILD A WEB-BASED RECOMMENDER SYSTEM

3 Steps of Clustering Process

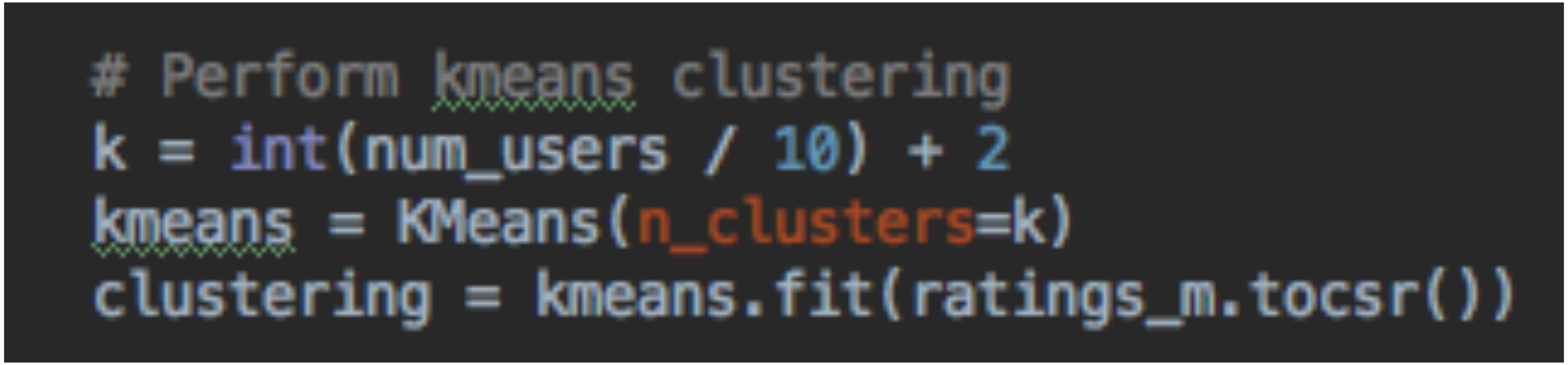
Step 3.1 create sparse matrix using user reviews ratings

	wine_id$_1$	wine_id$_2$	• • •		wine_id$_n$
username$_1$	rating$_{(i,j)}$				
username$_2$					
• • •					
username$_n$					

*in order to easily build a sparse matrix, dok_matrix from scipy is used

3 Steps of Clustering Process

Step 3.2 Perform K-Means Clustering

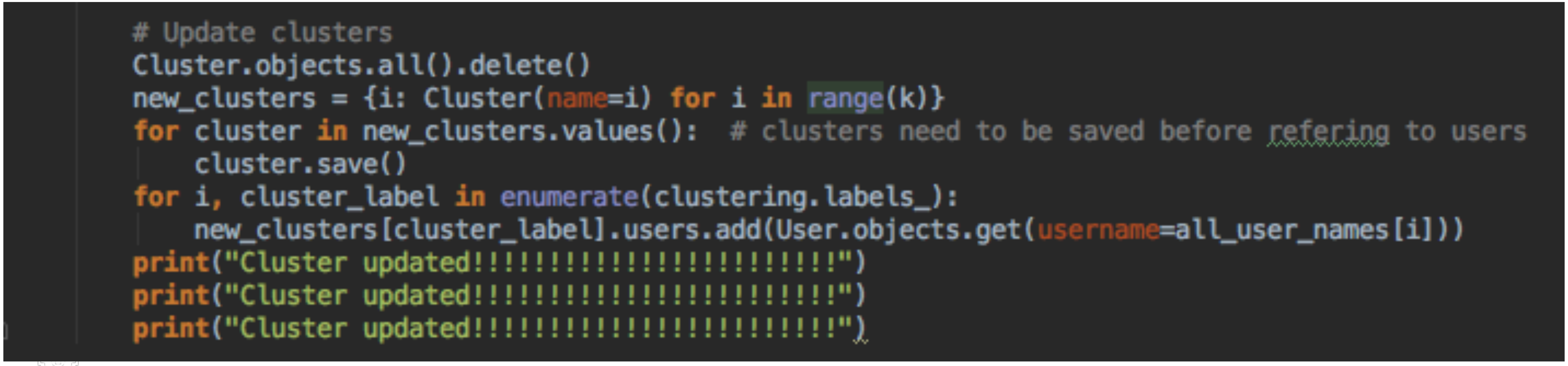


```
# Perform kmeans clustering
k = int(num_users / 10) + 2
kmeans = KMeans(n_clusters=k)
clustering = kmeans.fit(ratings_m.tocsr())
```

*convert dok_matrix into a csr_matrix that is better for the calculation needed in k-means

3 Steps of Clustering Process

Step 3.3 Update cluster assignments in the database



```
# Update clusters
Cluster.objects.all().delete()
new_clusters = {i: Cluster(name=i) for i in range(k)}
for cluster in new_clusters.values():  # clusters need to be saved before refering to users
    cluster.save()
for i, cluster_label in enumerate(clustering.labels_):
    new_clusters[cluster_label].users.add(User.objects.get(username=all_user_names[i]))
print("Cluster updated!!!!!!!!!!!!!!!!!!!!!!!!")
print("Cluster updated!!!!!!!!!!!!!!!!!!!!!!!!")
print("Cluster updated!!!!!!!!!!!!!!!!!!!!!!!!")
```