Development phase -2

# Machine Learning model deployment using IBM Cloud Watson Studio

## Product Recommendation System for E-commerce

# Part II: Model-Based Collaborative Filtering System

A model-based collaborative filtering system is used to recommend items to users by analyzing their purchase history and comparing the ratings given by other users who have bought similar items. This approach leverages patterns in user preferences from a diverse set of user data.

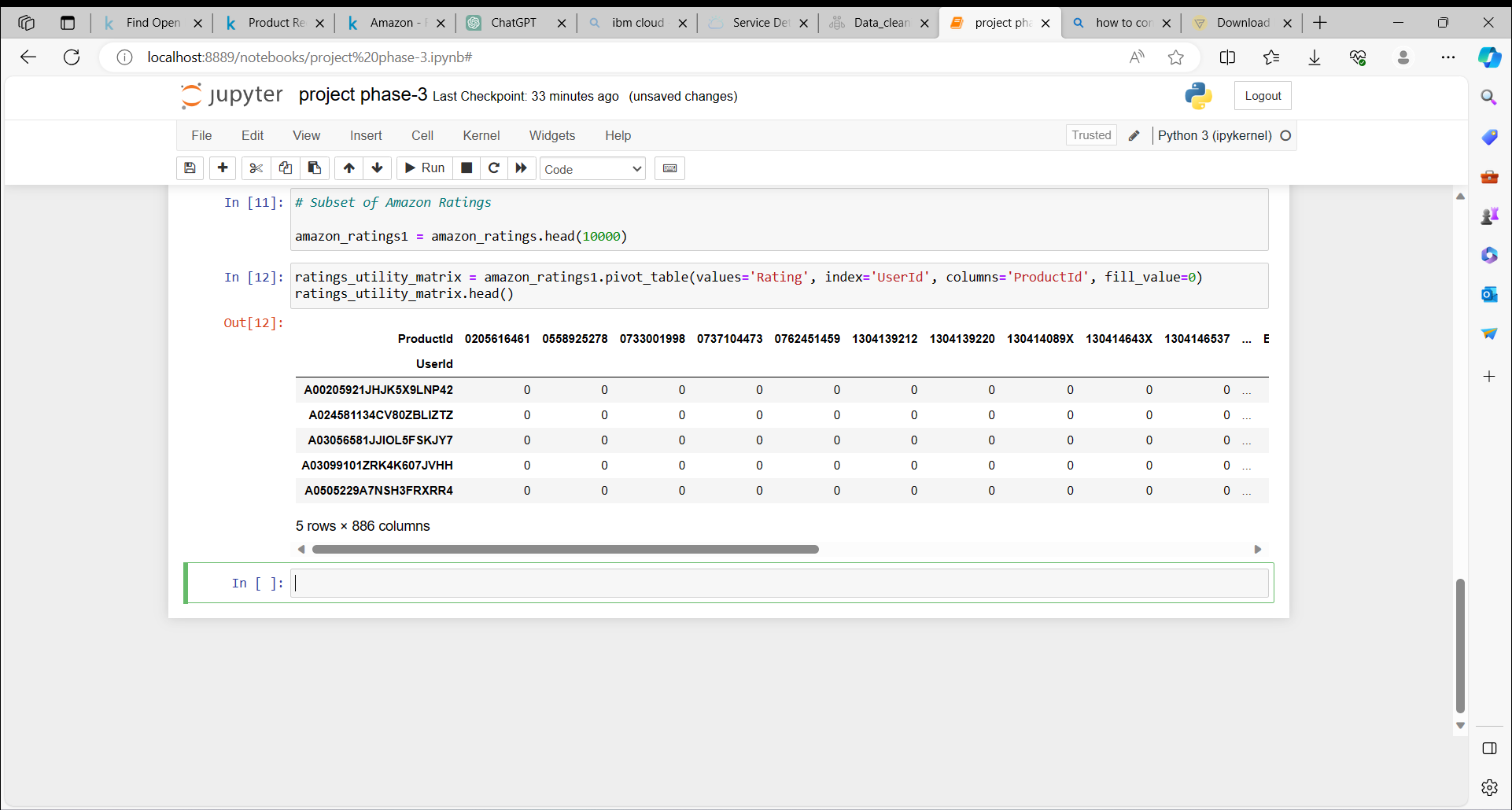
The utility matrix, which is the foundation of this system, is a representation of user-item preferences in the form of a matrix. This matrix contains information about the ratings and preferences of users for various items. Since users typically do not purchase or rate all the items available, the utility matrix is sparse, with many values remaining unknown.

# Subset of Amazon Ratings

amazon\_ratings1 = amazon\_ratings.head(10000)

ratings\_utility\_matrix = amazon\_ratings1.pivot\_table(values='Rating', index='UserId', columns='ProductId', fill\_value=0)

ratings\_utility\_matrix.head()



As expected, the utility matrix obtaned above is sparce, I have filled up the unknown values wth 0.

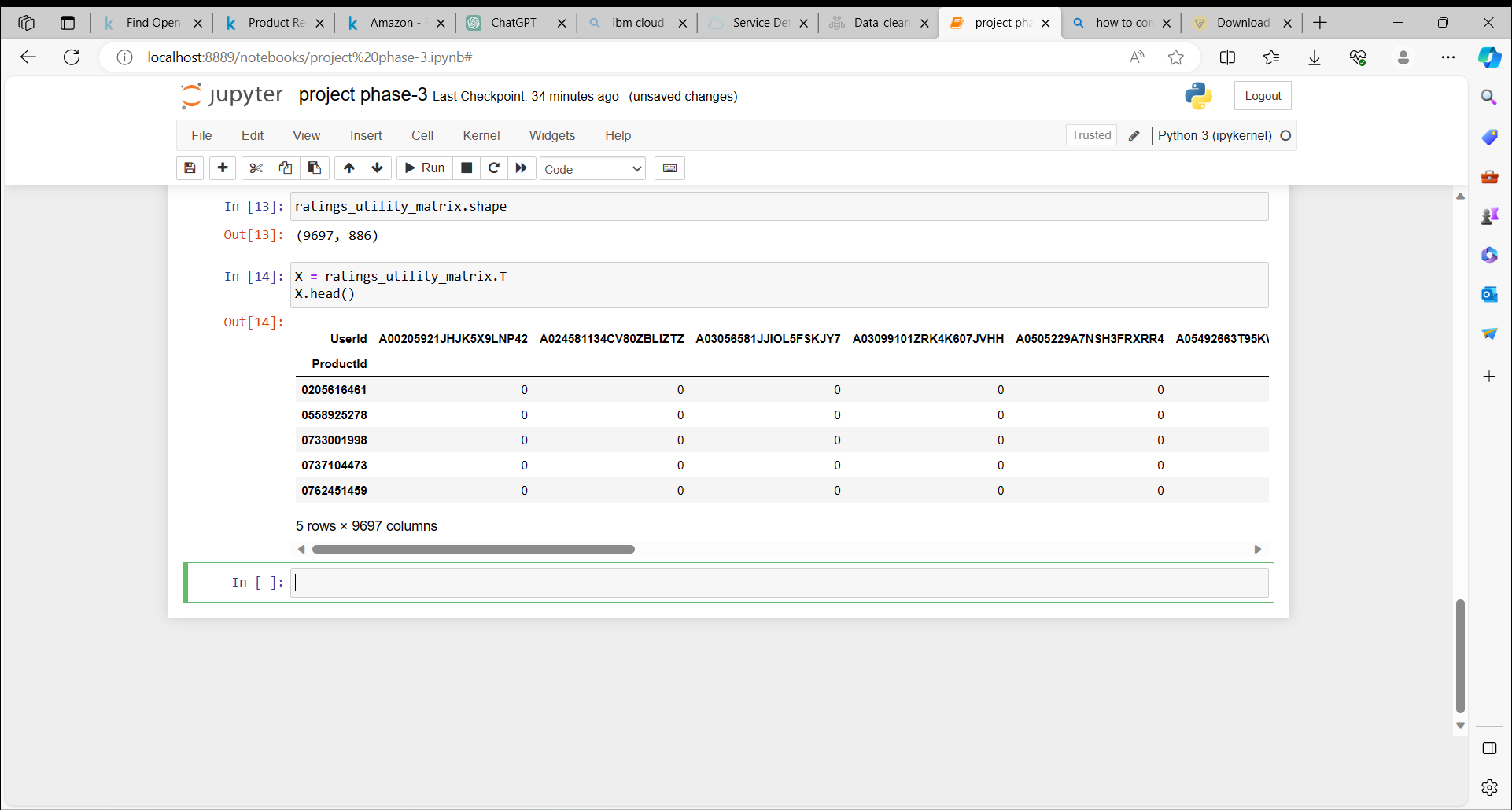
ratings\_utility\_matrix.shape

(9697, 886)

Transposing the matrix

X = ratings\_utility\_matrix.T

X.head()



X.shape

(886, 9697)

Unique products in subset of data

X1 = X

### **Decomposing the Matrix**

SVD = TruncatedSVD(n\_components=10)

decomposed\_matrix = SVD.fit\_transform(X)

decomposed\_matrix.shape

(886, 10)

### **Correlation Matrix**

correlation\_matrix = np.corrcoef(decomposed\_matrix)

correlation\_matrix.shape

(886, 886)

### **Isolating Product ID # 6117036094 from the Correlation Matrix**

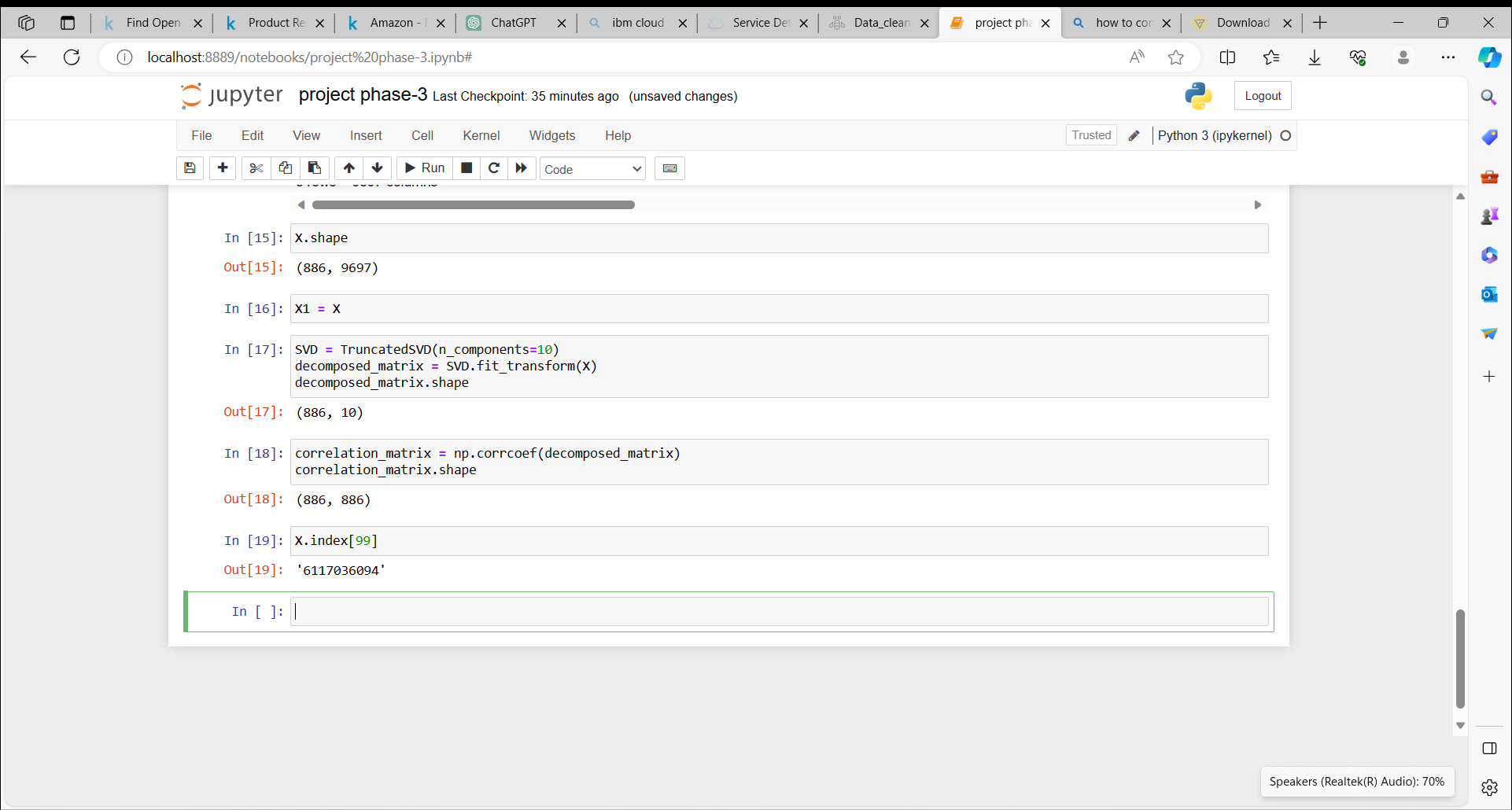
Assuming the customer buys Product ID # 6117036094 (randomly chosen)

X.index[99]

'6117036094'

linkcode

Index # of product ID purchased by customer



i = "6117036094"

product\_names = list(X.index)

product\_ID = product\_names.index(i)

product\_ID

99

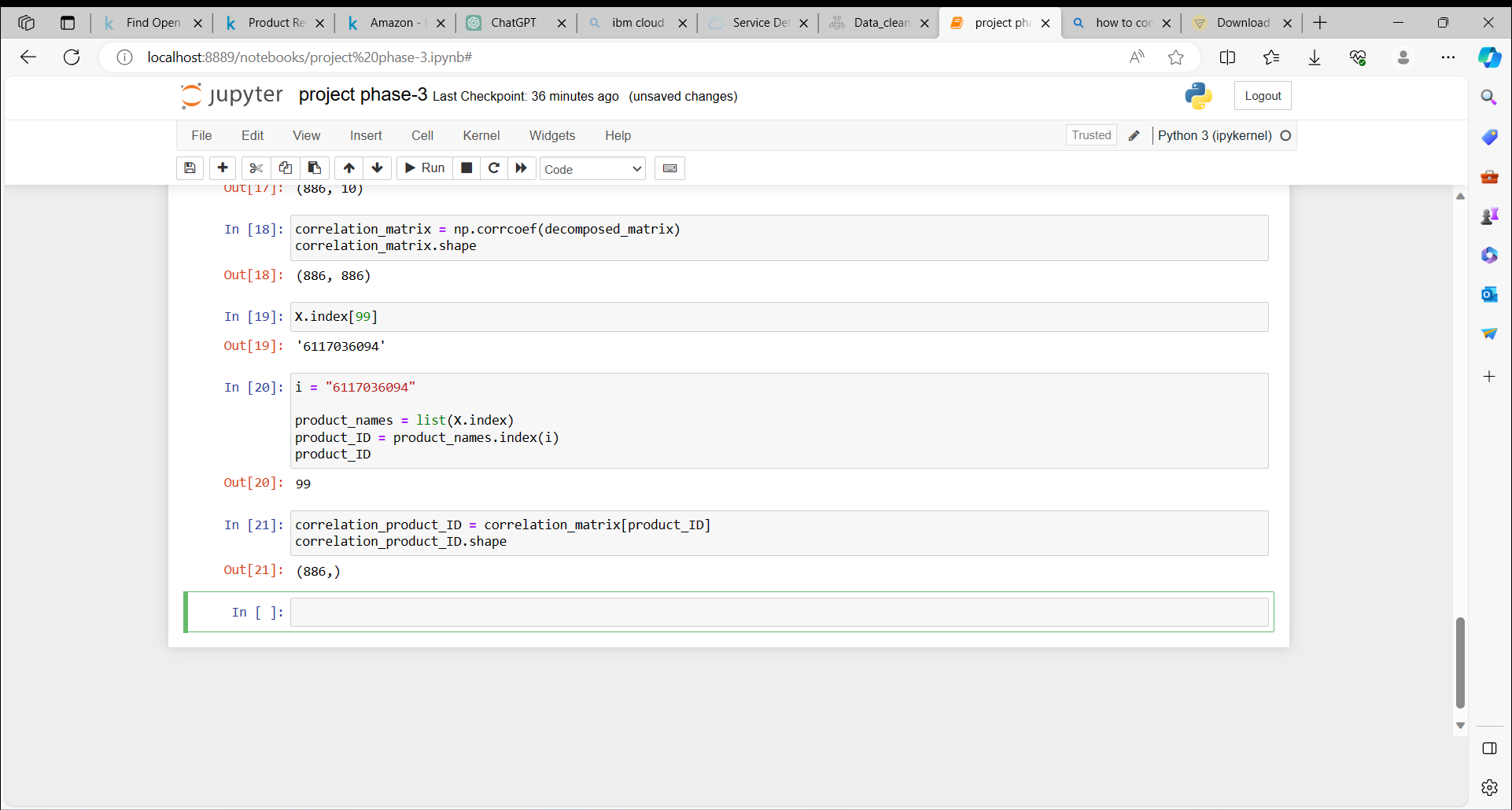
Linkcode

Correlation for all items with the item purchased by this customer based on items rated by other customers people who bought the same product

correlation\_product\_ID = correlation\_matrix[product\_ID]

correlation\_product\_ID.shape

(886,)



### **Recommending top 10 highly correlated products in sequence**

Recommend = list(X.index[correlation\_product\_ID > 0.90])

*# Removes the item already bought by the customer*

Recommend.remove(i)

Recommend[0:9]

['0733001998',

'1304139212',

'1304139220',

'130414089X',

'130414643X',

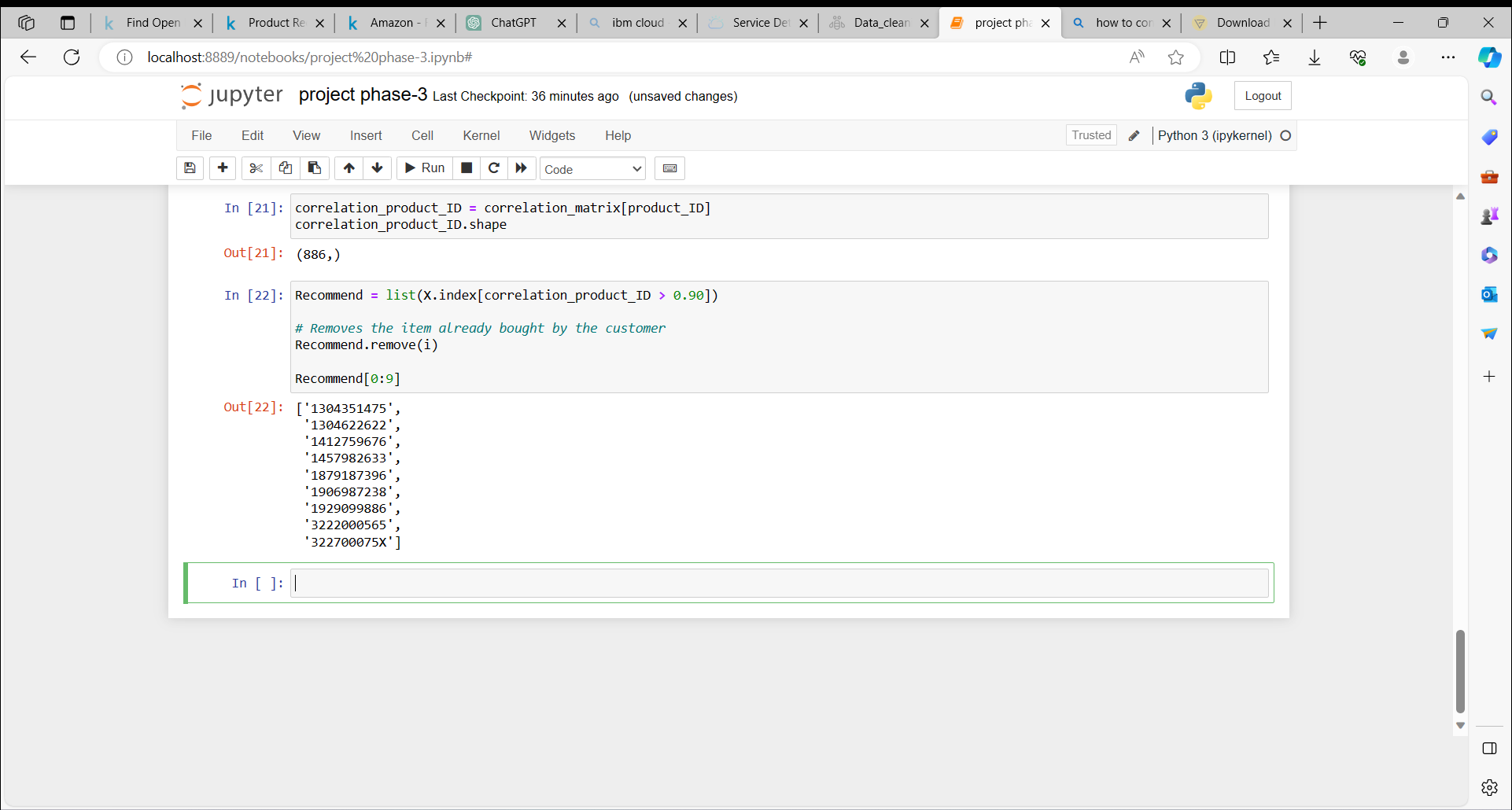
'130414674X',

'1304174778',

'1304174867',

'1304174905']

**Product Id #**Here are the top 10 products to be displayed by the recommendation system to the above customer based on the purchase history of other customers in the website.



# Part III: Recommendations for Businesses Without Product Ratings

In the absence of user-item purchase history, a recommendation system for a business can be created using a search engine approach. This system generates product recommendations by analyzing and clustering textual information provided in product descriptions.

To accomplish this, a dataset from Home Depot, containing product information, is utilized as the source of data. This dataset serves as the foundation for generating recommendations based on the textual content and characteristics of the products.

# Importing libraries

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

from sklearn.neighbors import NearestNeighbors

from sklearn.cluster import KMeans

from sklearn.metrics import adjusted\_rand\_score

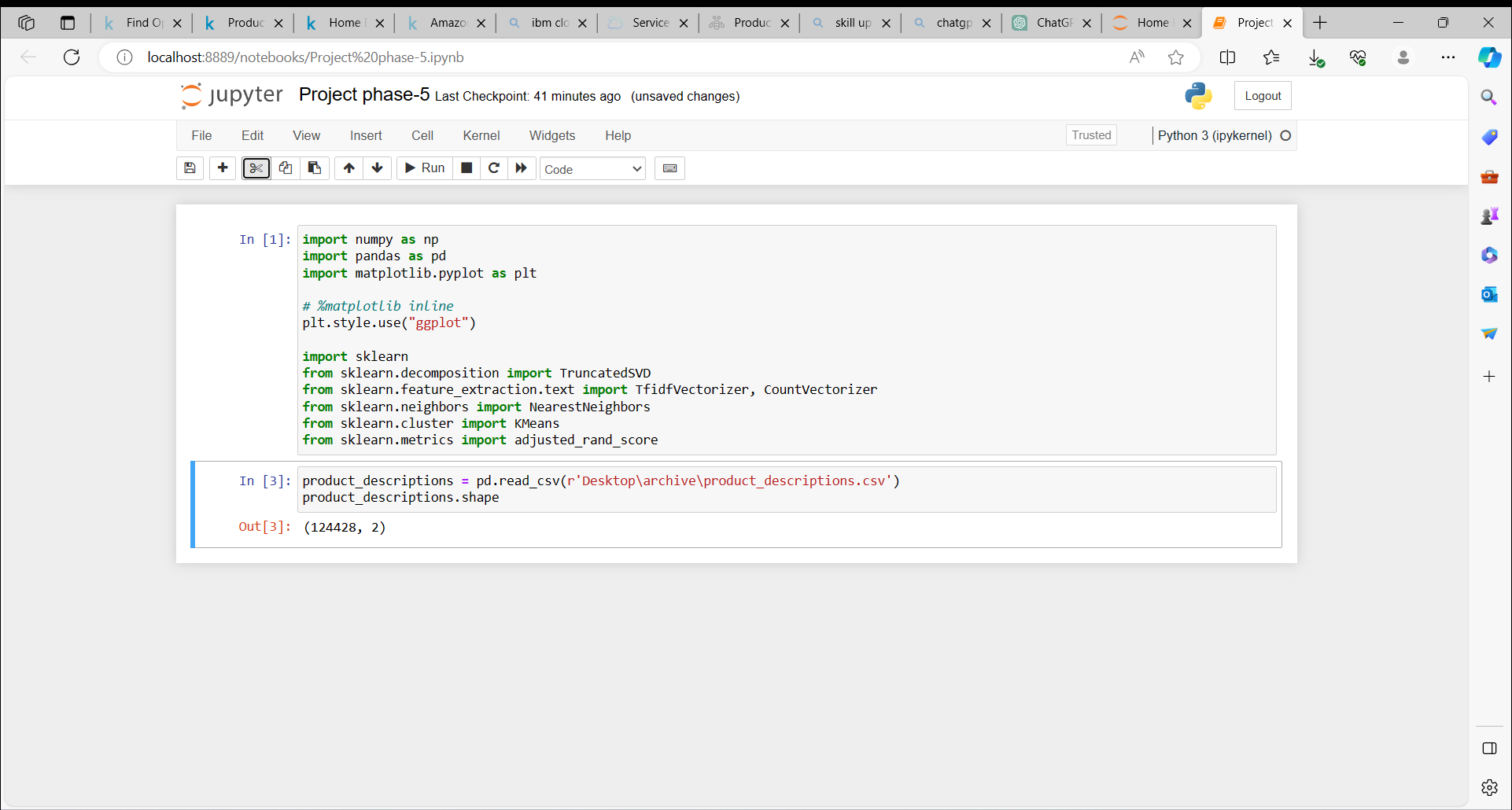
### **Item to item based recommendation system based on product description**

Applicable when business is setting up its E-commerce website for the first time

product\_descriptions = pd.read\_csv(r’Desktop\archive\product\_descriptions.csv')

product\_descriptions.shape

(124428, 2)



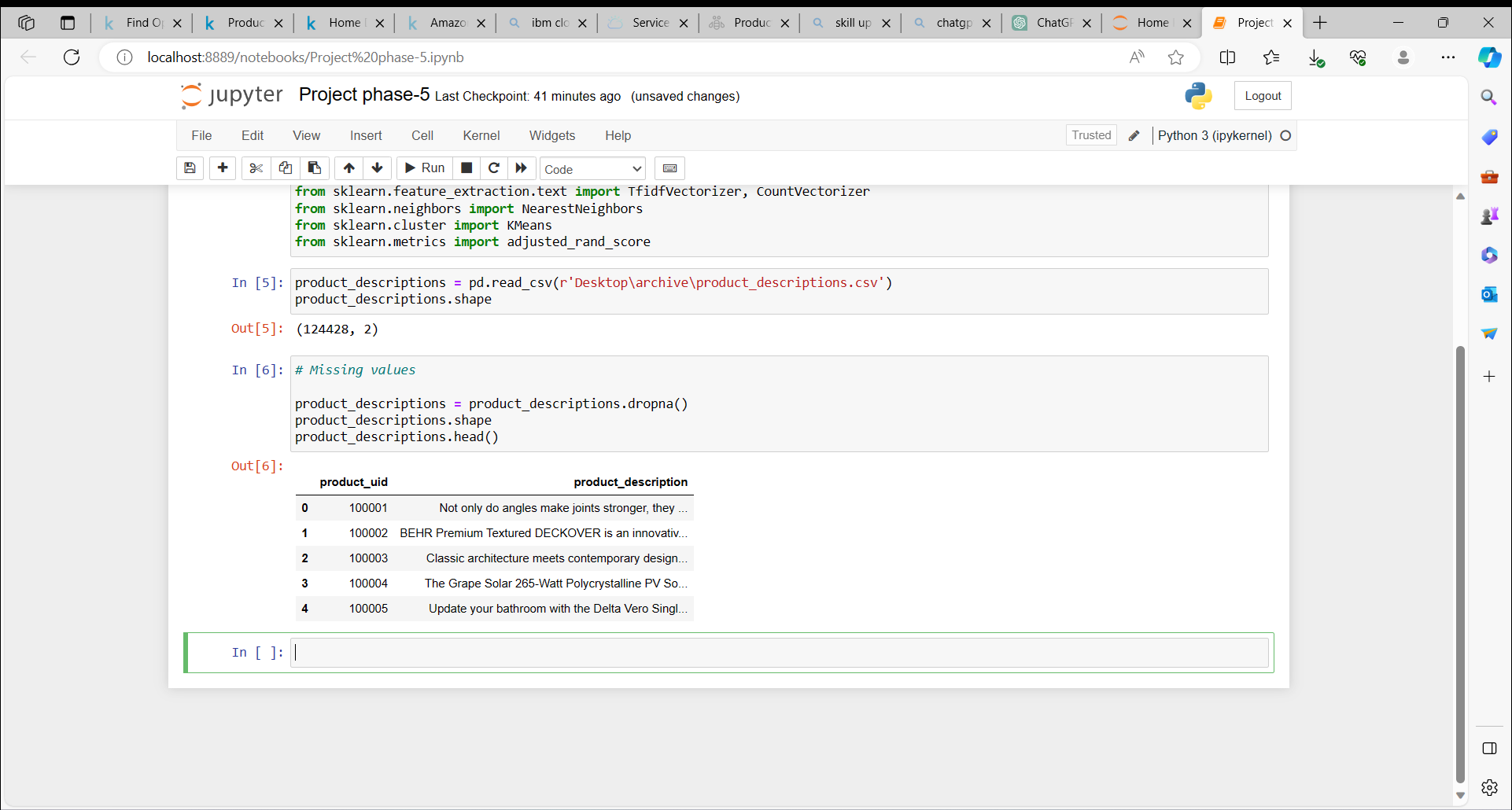
#### **Checking for missing values**

# Missing values

product\_descriptions = product\_descriptions.dropna()

product\_descriptions.shape

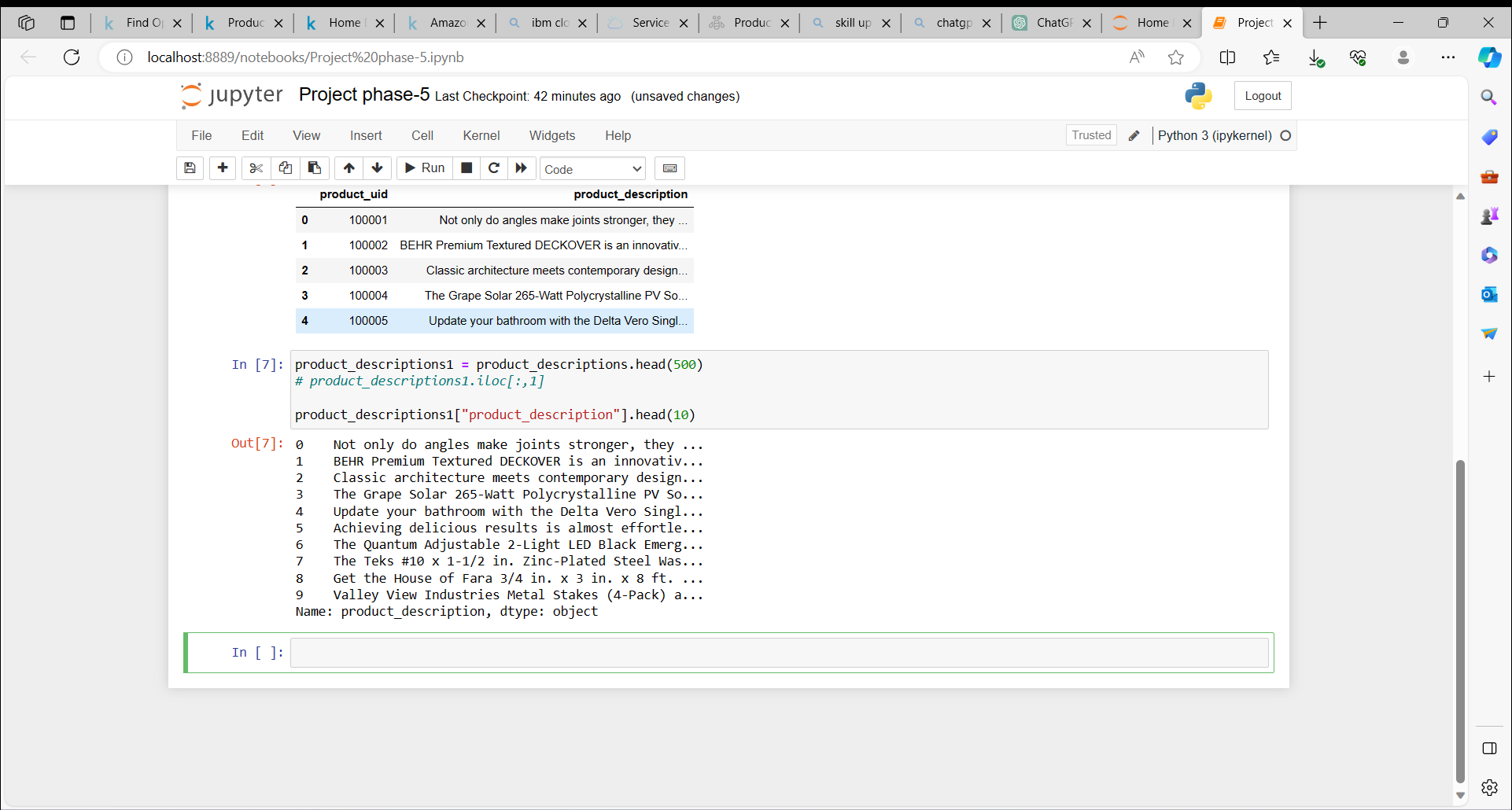
product\_descriptions.head()



product\_descriptions1 = product\_descriptions.head(500)

# product\_descriptions1.iloc[:,1]

product\_descriptions1["product\_description"].head(10)



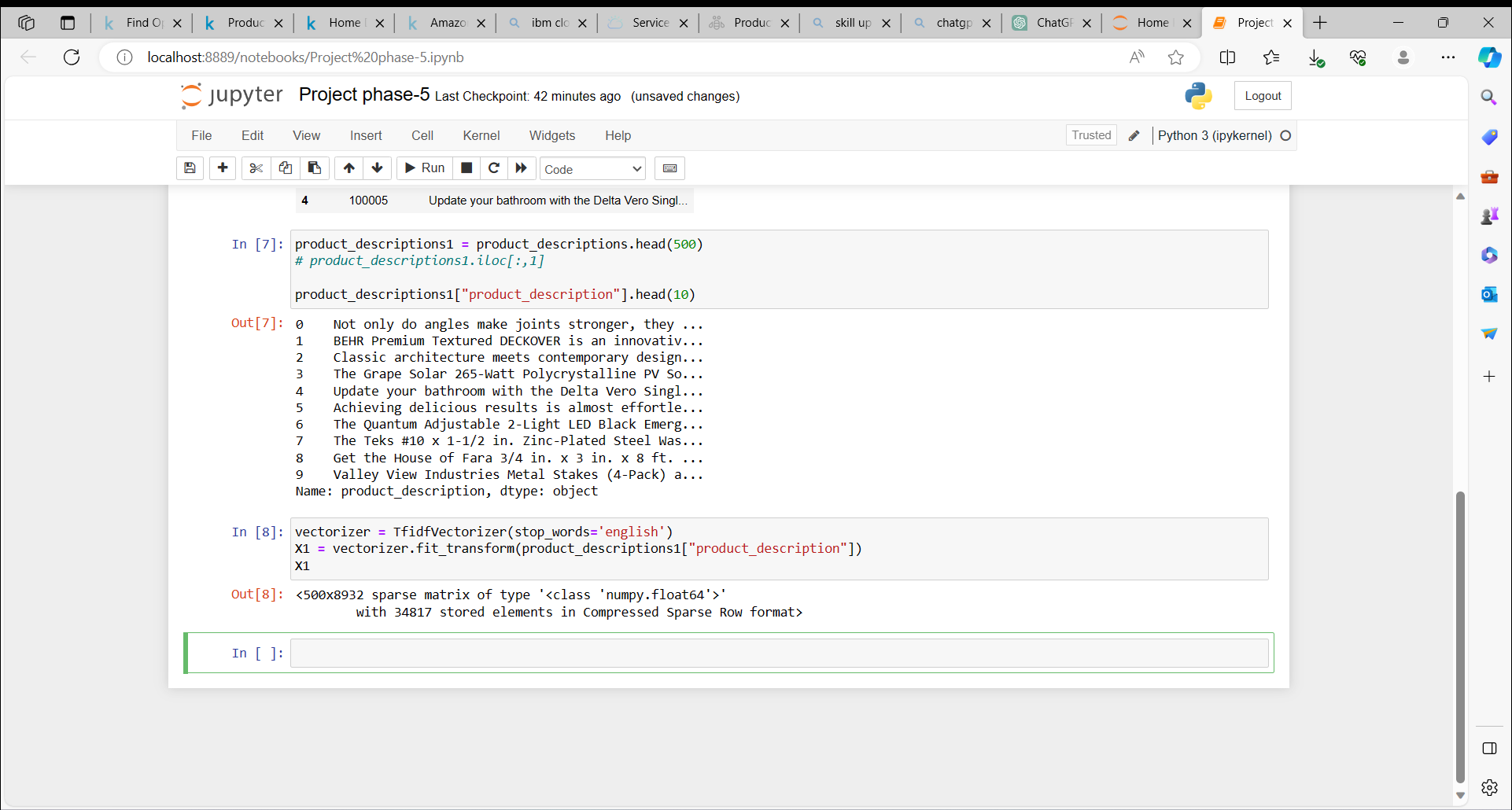
#### **Feature extraction from product descriptions**

Converting the text in product description into numerical data for analysis

vectorizer = TfidfVectorizer(stop\_words='english')

X1 = vectorizer.fit\_transform(product\_descriptions1["product\_description"])

X1



#### **Visualizing product clusters in subset of data**

linkcode

*# Fitting K-Means to the dataset*

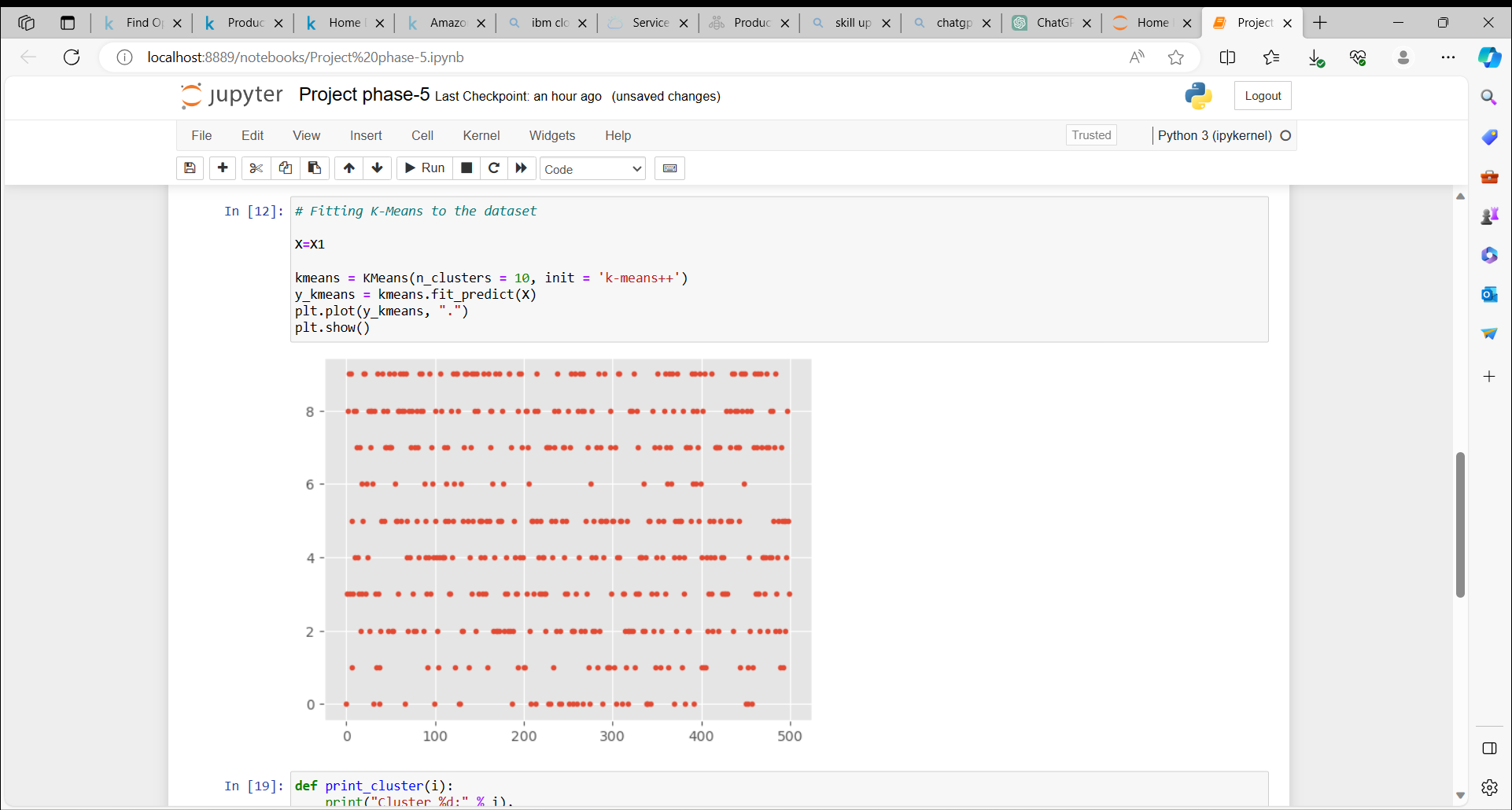
X=X1

kmeans = KMeans(n\_clusters = 10, init = 'k-means++')

y\_kmeans = kmeans.fit\_predict(X)

plt.plot(y\_kmeans, ".")

plt.show()



def print\_cluster(i):

print("Cluster %d:" % i),

for ind in order\_centroids[i, :10]:

print(' %s' % terms[ind]),

print

* Recommendation of product based on the current product selected by user.
* To recommend related product based on, Frequently bought together.

#### **Top words in each cluster based on product description**

# # Optimal clusters is

true\_k = 10

model = KMeans(n\_clusters=true\_k, init='k-means++', max\_iter=100, n\_init=1)

model.fit(X1)

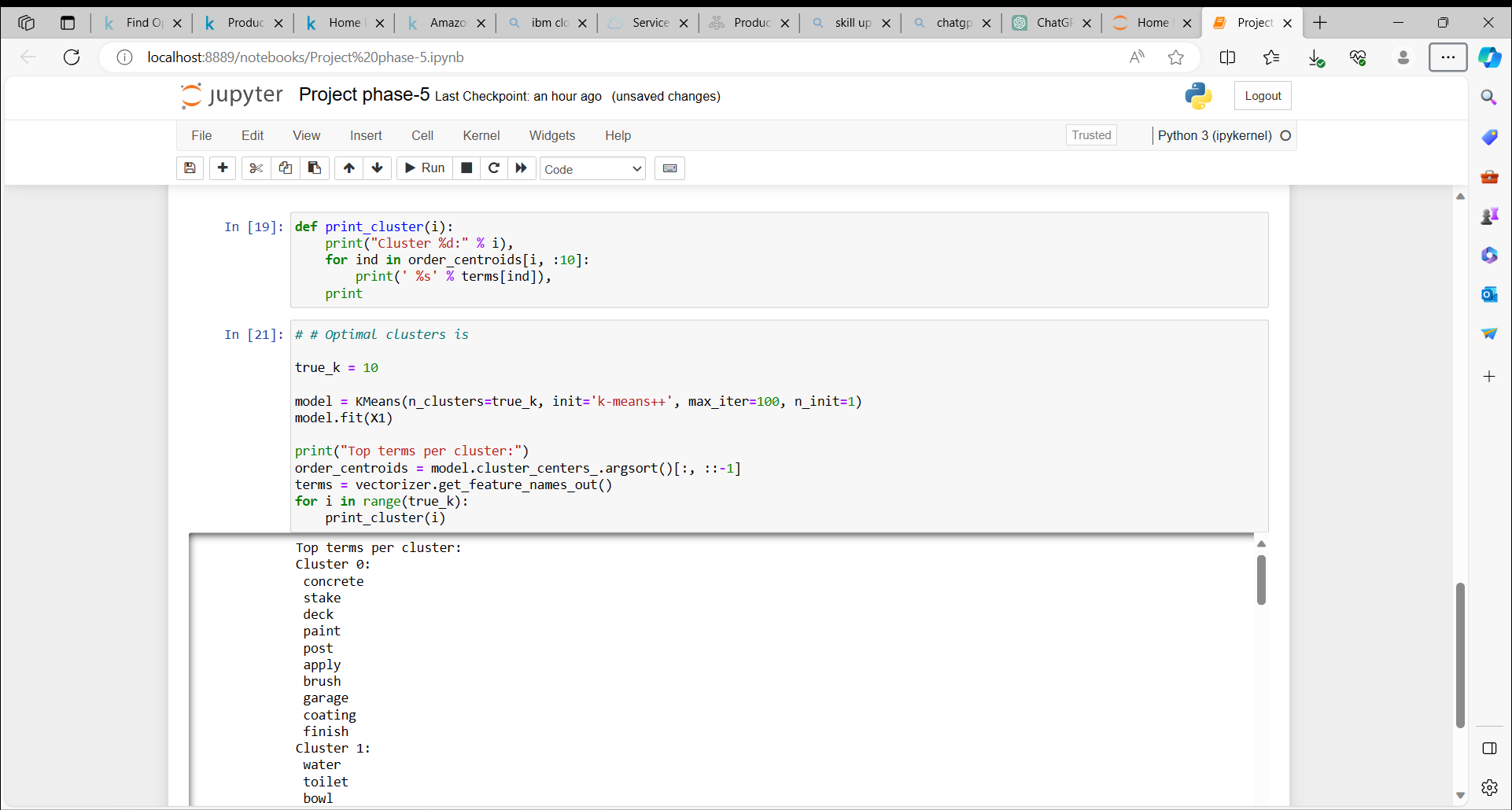
print("Top terms per cluster:")

order\_centroids = model.cluster\_centers\_.argsort()[:, ::-1]

terms = vectorizer.get\_feature\_names()

for i in range(true\_k):

print\_cluster(i)



#### **Predicting clusters based on key search words**

def show\_recommendations(product):

#print("Cluster ID:")

Y = vectorizer.transform([product])

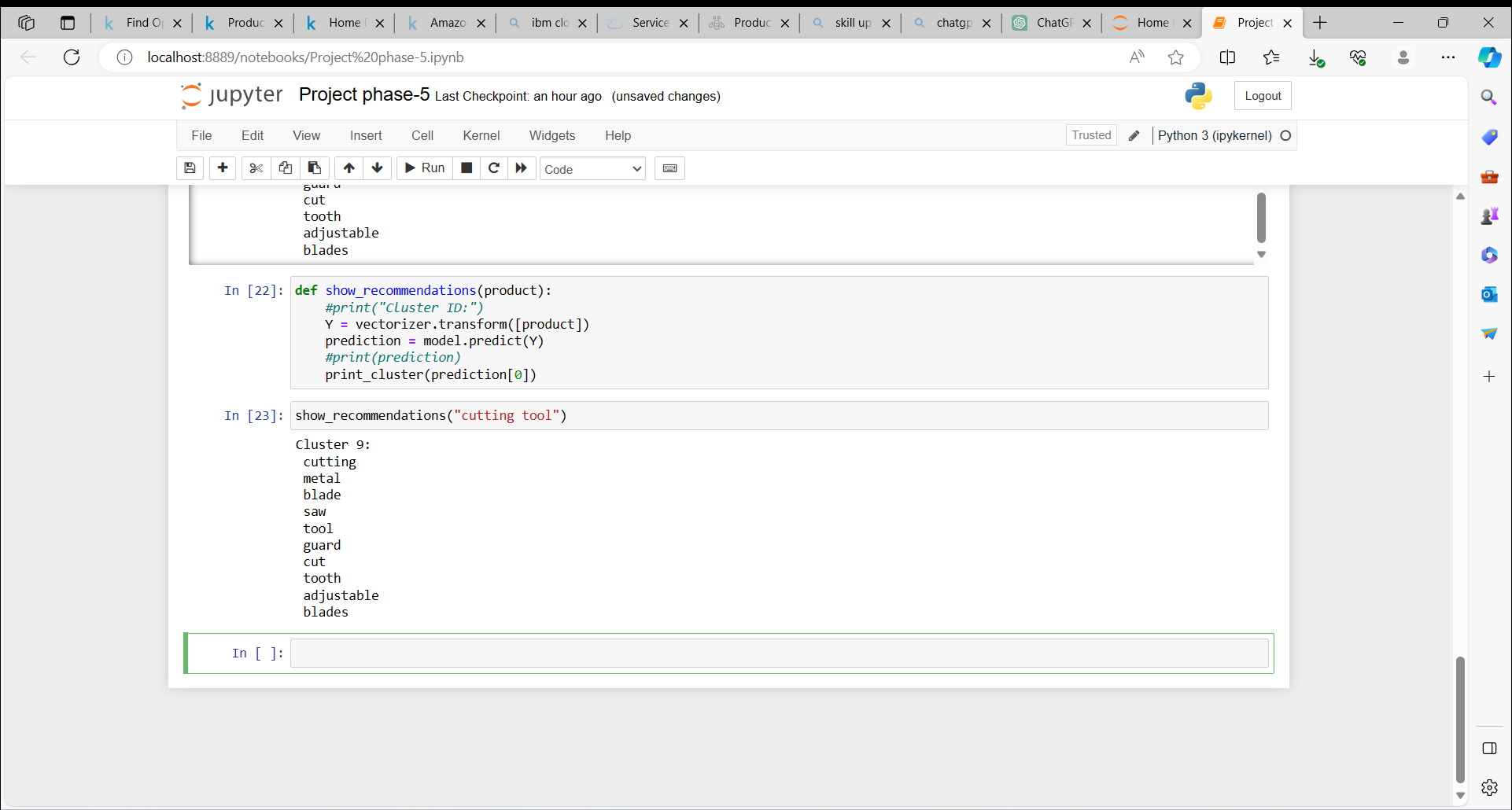
prediction = model.predict(Y)

#print(prediction)

print\_cluster(prediction[0])

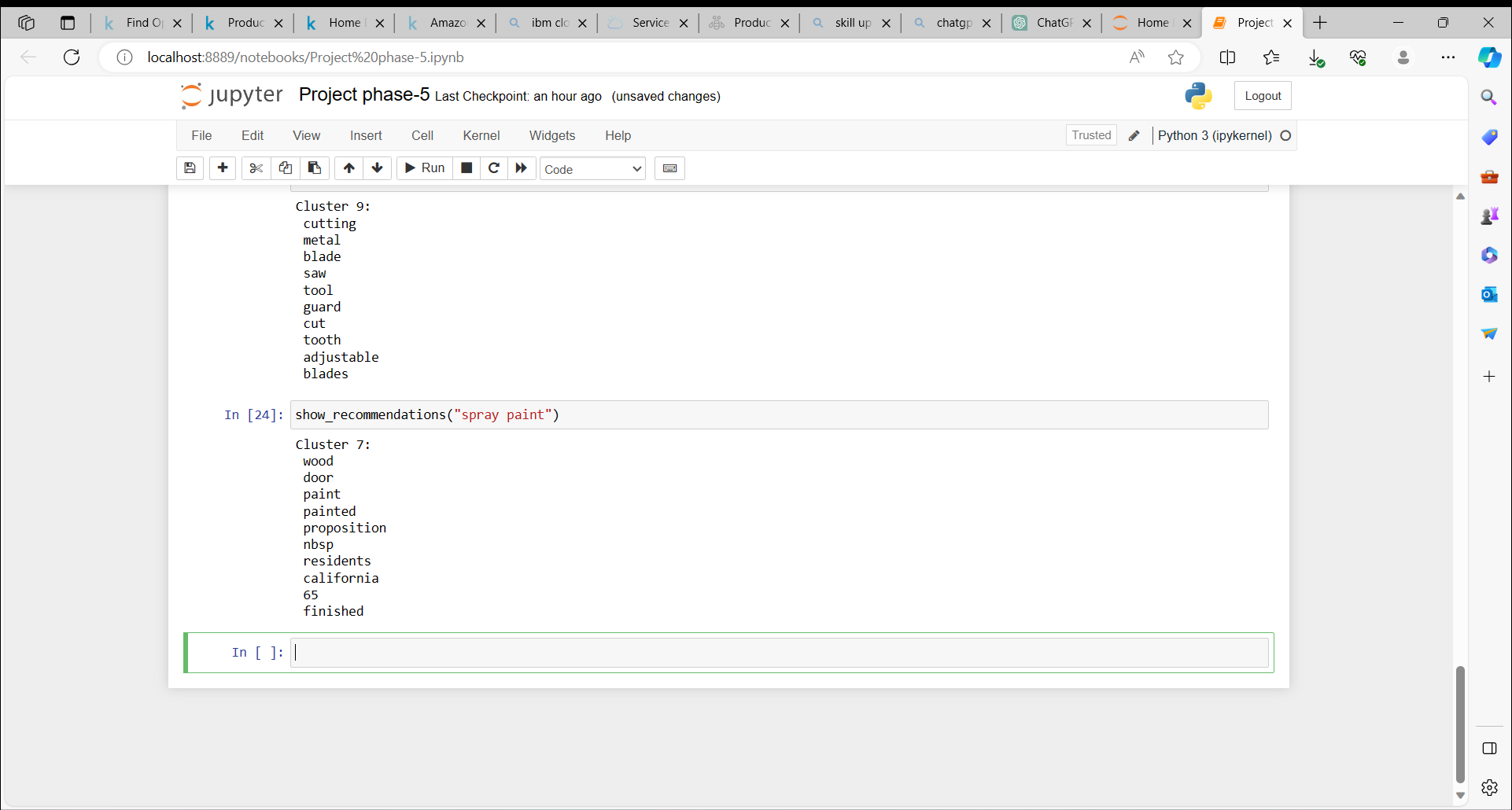
**Keyword :**cutting tool

show\_recommendations("cutting tool")



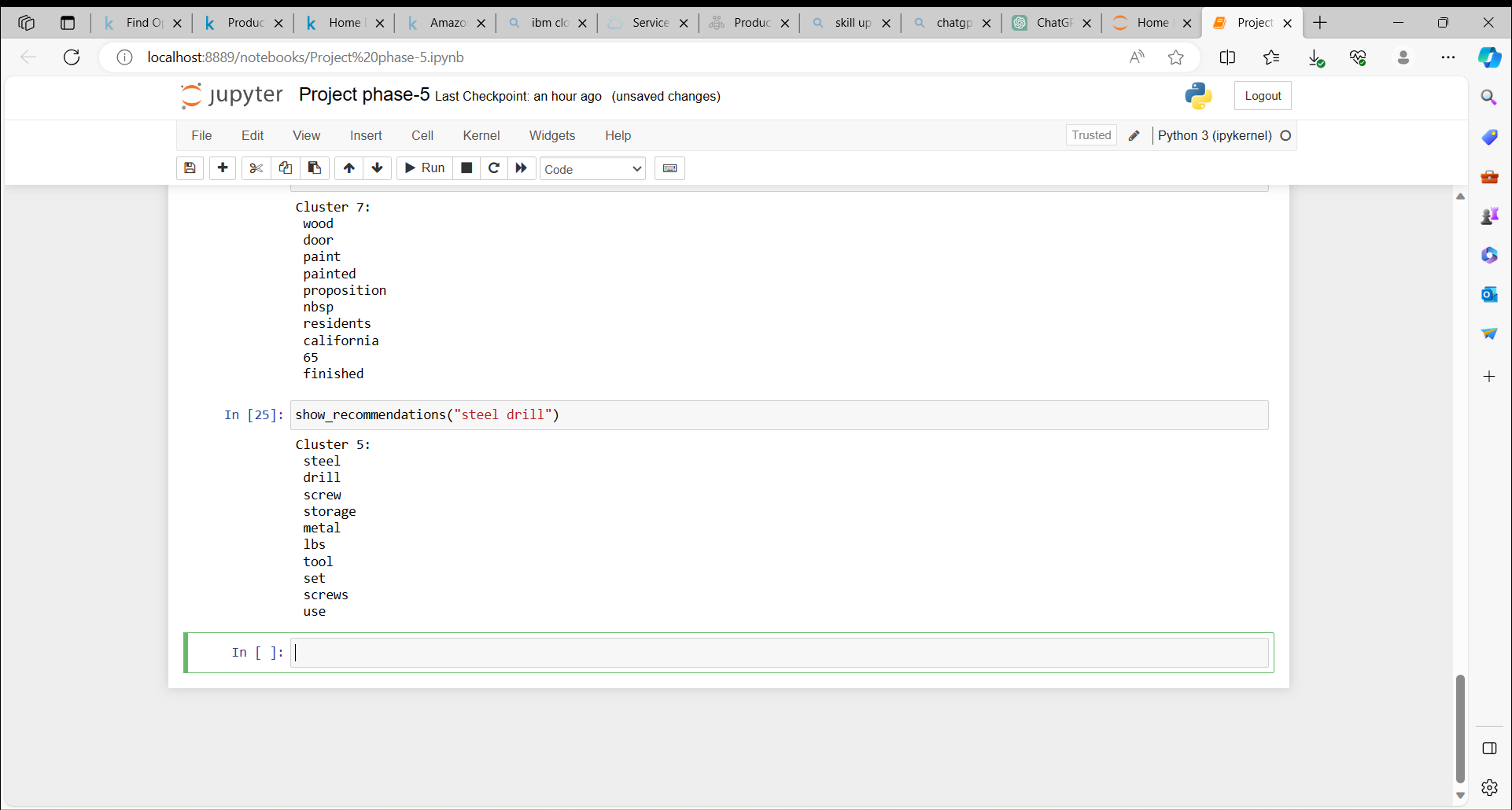
**Keyword :**spray paint

show\_recommendations("spray paint")



**Keyword :**steel drill

show\_recommendations("steel drill")



Once a cluster is identified based on the user's search words, the recommendation system can display items from the corresponding product clusters based on the product descriptions.

#### **Summary:**

This approach is most effective when a business is establishing its e-commerce website from scratch and lacks any initial user-item purchase or rating history. In such cases, this recommendation system provides valuable initial recommendations to assist users. As buyers accumulate a purchase history over time, the recommendation engine can transition to employing model-based collaborative filtering techniques for even more personalized and accurate recommendations.