E-commerce_product_recommendation_system

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# %matplotlib inline
plt.style.use("ggplot")


import sklearn
from sklearn.decomposition import TruncatedSVD
amazon_ratings = pd.read_csv('../input/amazon-ratings/ratings_Beauty.csv')
amazon_ratings = amazon_ratings.dropna()
amazon_ratings.head()
amazon_ratings.shape
popular_products = pd.DataFrame(amazon_ratings.groupby('ProductId')['Rating'].count())
most_popular = popular_products.sort_values('Rating', ascending=False)
most_popular.head(10)
most_popular.head(30).plot(kind = "bar")
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()
ratings_utility_matrix.shape
X = ratings_utility_matrix.T
X.head()
X.shape
X1 = X
SVD = TruncatedSVD(n_components=10)
decomposed_matrix = SVD.fit_transform(X)
decomposed_matrix.shape
correlation_matrix = np.corrcoef(decomposed_matrix)
correlation_matrix.shape
X.index[99]
i = "6117036094"


product_names = list(X.index)
```

```
product_ID = product_names.index(i)
product_ID
correlation_product_ID = correlation_matrix[product_ID]
correlation_product_ID.shape
Recommend = list(X.index[correlation_product_ID > 0.90])


# Removes the item already bought by the customer
Recommend.remove(i)



Recommend[0:9]
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
product_descriptions = pd.read_csv('../input/home-depot-product-search-
relevance/product_descriptions.csv')
product_descriptions.shape
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)
vectorizer = TfidfVectorizer(stop_words='english')
X1 = vectorizer.fit_transform(product_descriptions1["product_description"])
X1
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
```

```
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)
def show_recommendations(product):
#print("Cluster ID:")
Y = vectorizer.transform([product])
prediction = model.predict(Y)
#print(prediction)
print_cluster(prediction[0])
show_recommendations("cutting tool")
show_recommendations("spray paint")
show_recommendations("steel drill")
show_recommendations("water")
```