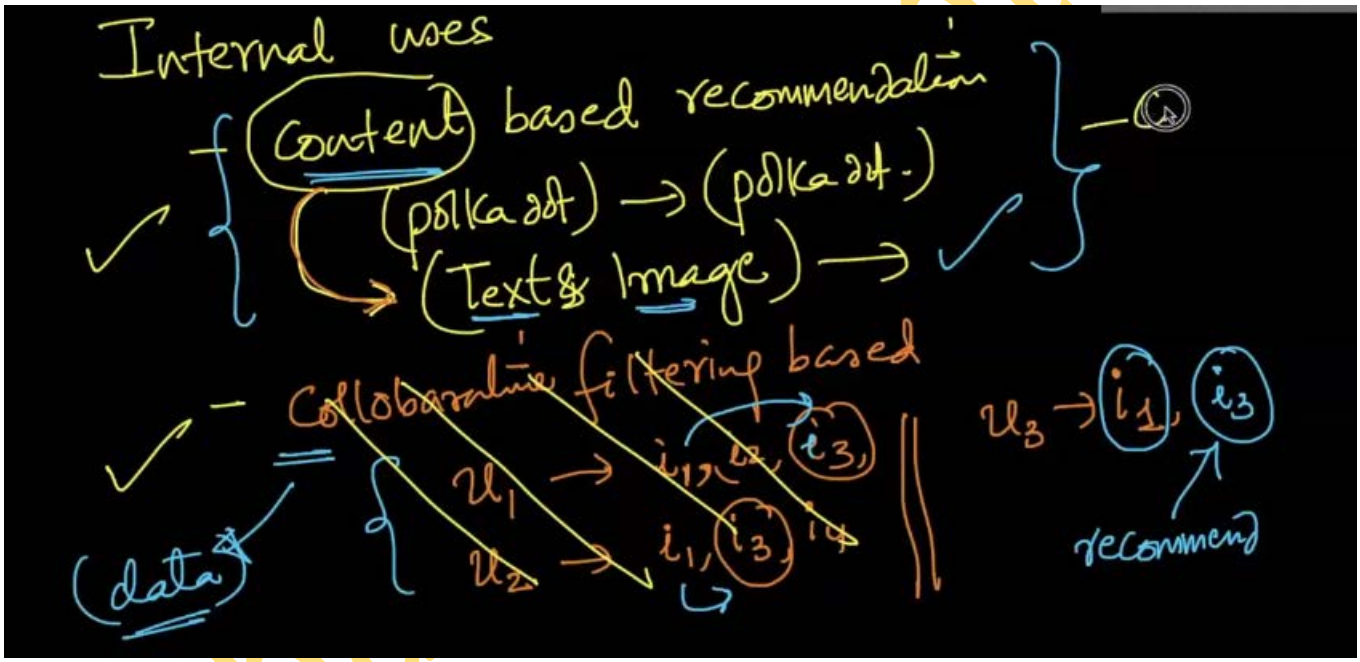
APPAREL RECOMMENDATION ON AMAZON-

Filtering is of 2 types-

1. Content based- Uses similarity between product images, brand, product title, product description etc to find similarity between products. I use this data in our project.

2. Collaborative filtering- Recommends products based on choices of users who visited the target product and later visited other products too. Data highly confidential.



Process pipeline used-

1. Data acquisition
2. Data cleaning
3. Text pre-processing
4. Text-based product recommendations (BoW, Tf-Idf, Word2Vec)
5. Deep learning image-based product recommendation
6. A/B Testing

1. Data acquisition-

- Web scraping on Amazon website is not policy compliant.
- So, Amazon's Product Advertising API used.\ \

- Data for women's tops was acquired. (183000 policy compliant product data obtained).
- I acquired the dataset from a certain ML workshop.

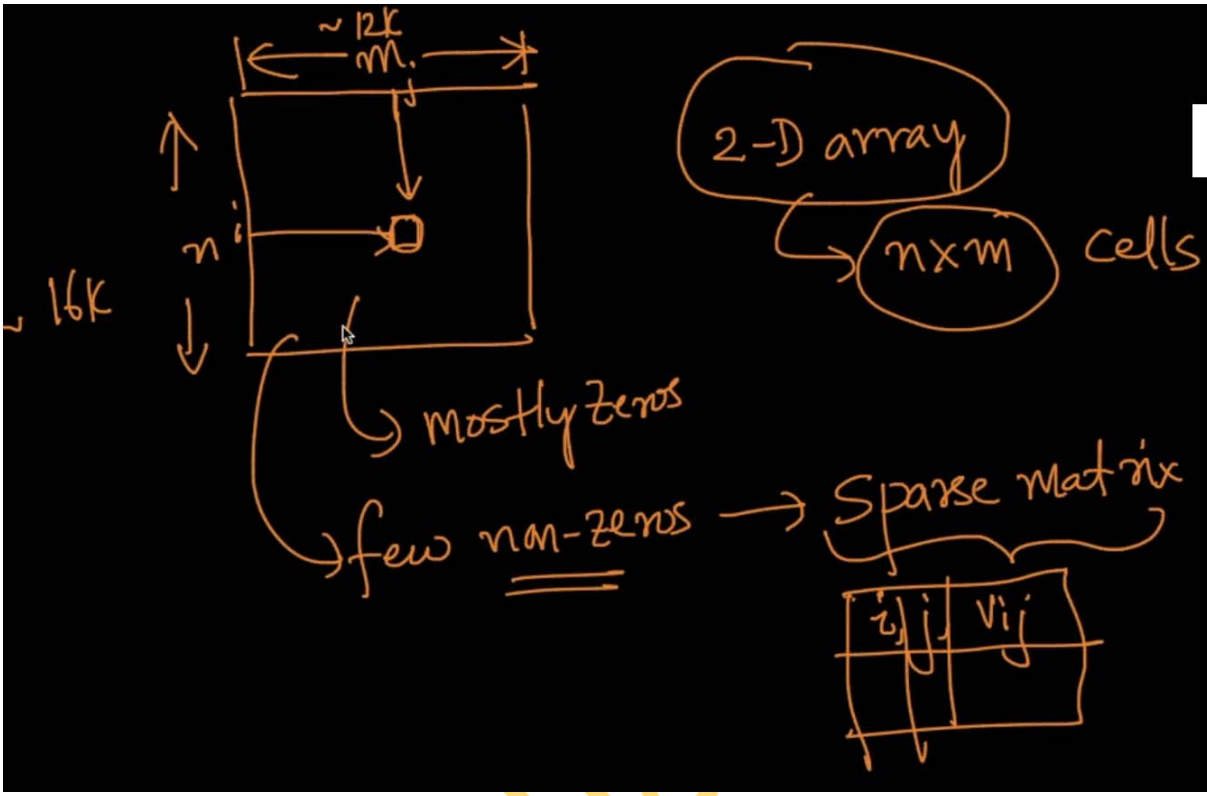
2. Out of 19 features, 7 have been used. They are described in the notebook. Title is a short yet the most useful text feature used. Product description has been avoided due to less information density in its text and avoiding high compute complexity.

3. EDA- column.describe(), column.unique() and column.most_common().

4. Price and colour have several missing values. Missing values are removed from the data.

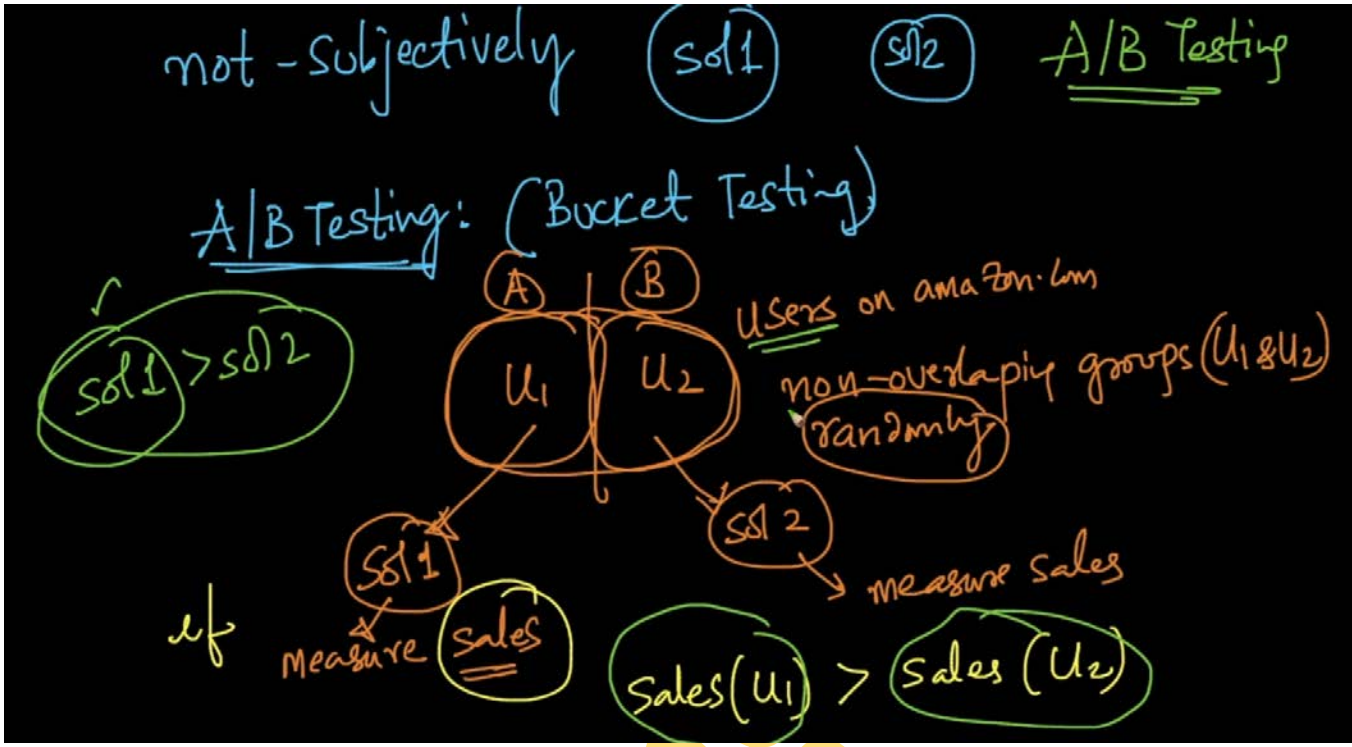
5. Several duplicate values in title is observed. 2325 approx.

6. Same shirt but different sizes have differing ASIN, hence are considered different data points. These need to be de-duped for good customer exp.
7. First the points are sorted alphabetically (title). Successive points having differing words <2 are eliminated
8. Second de-dup: O(n2) time. All data points are considered using twin loops. If differing words<3, the index is removed. (When dissimilar words are not at the end but in middle)
9. Pre-processing text: Stopword removal, non alphanumeric character removal and lowercase conversion.
10. Now, I explore text-based similarity.
11. BoW model does not return a 2-D matrix but a sparse matrix where each NON-ZERO value is stored in a tuple containing i, j and a[i][j]. This takes less space.

12. 

2. Utility functions are written. One for displaying image. One for plotting a plot divided into two subplots.
3. 'Plot heatmap' plots a frequency heatmap based on keys (common words), values (occurrences of common words), labels (model predicted values of common words). It also plots the image from the URL.
4. 'Plot heatmap image' first computes common words and passes on the keys and values to 'plot heatmap function'. 'Text to vector' returns a counter containing word-occurrences pair.
5. Tf-idf, BoW and idf models are explored.
6. Using scikit learn.metrics pairwise distances function, the indices and values of the smallest Euclidean distances are calculated and noted. These are passed to the get results function which plots the corresponding heatmap for words.
7. In the heatmap, black colouring shows that the word is not common between the current and query point. Deep pink shows that the common word is present for >2 times among the two points. The heatmap is annotated with BoW or Tf-idf values.
8. For tf-idf heatmap, for 1st row, the annotated labels literally specify the IDF as the TF for all the words is same.
9. TF-IDF gave better results in general than BoW.

10. However, TF-IDF gives more preference to shorter data points [inverse of no of terms]. It is good for webpages and other long docs etc. IDF was used but not for better results.
11. Now Word2Vec for semantic similarity retention was used.
12. Utility functions are defined. get_word_vec returns w2v (weighted or normal) of each word in sentence
13. get_distance computes distance between each word of 2 sentences (in w2v form) and returns the corresponding distance matrix.
14. plot_w2v_heatmap plots the heatmap corresponding to the dist matrix. Smaller distance corresponds to more similarity. XXL and 12 are similar, tiger and tigers are similar etc.
15. 'build_avg_vec' function converts each sentence in data[title] into its corr avg w2v vector (sentence2vec).
16. 'avg_w2v_model' uses the converted word2vec of titles to build the heatmap.
17. The same is done for idf-weighted w2v/
18. Next one-hot encoding of brand, product_type and colours columns is done to add extra features.
19. Na Values are filled with string and " " replaced with hyphen.
20. They are horizontally stacked
21. A heatmap is plotted with these features.
22. In industry, business rules are applied after considering all outputs from the models. [eg. same brand shld not occur more than twice].
23. Image features are extracted using VGG-16 CNN model. Keras Preprocessing Io used.
24. Pairwise distance between query image and recommended images are used to compute top images.
25. A/B Testing used for knowing which model is better.


not - subjectively
sol1
sol2
A/B Testing
A/B Testing: (Bucket Testing)
A
B
Users on amazon.com
sol1 > sol2
U1
U2
non-overlaping groups (U1 & U2)
randomly
sol1
sol2
measure sales
if
measure
sales
Sales (U1) > Sales (U2)