Discussing what we have done in the assignment:

1. Firstly we did the pre processing of the documents by removing the punctuations, special characters, and digits. Then further tokenised them and applied PORTER STEMMING Algo to do the required stemming.
2. Repeated the same task of pre processing with the queries as well.
3. Then using the inbuilt functionality of scikit-learn, i.e. TfidfVectorizer, we calculated the TF-IDF matrix where each row represents the document and columns represent the set of unique words in the entire corpus of documents.
4. Again we used the inbuilt functionality of rank_bm25 i.e. BM25Okapi to calculate the BM25 matrix where each row represents the documents and each column represents a query. Here is the thing, we first went with the standard values of k1 = 1.2 and b = 0.75, but since documents are not very long and at the same time,
5. Now, we calculated the top p stems i.e. terms across the entire corpus of documents. The value of p was kept as 6 because as we kept it 7, no query vector was having all the values as 1s, which will ultimately lead to zero number of document vectors matching the query vectors, so value of p was taken to be 6.
6. Further converted the documents and queries as binary vectors keeping the vocabulary as these top p stems and calculated logical AND, further stored those document vectors which are giving all the values of 1s in the final vector with the query vectors.
7. In the next question, it was asked to represent the documents and queries in the vector space model and using TF-IDF weights and BM25 weights as well. But that is what we get when we calculate their respective matrices as each row is the representation of the document vector in the given dimensions.
8. Further, we calculated the representations of the query in the same space by finding their matrices in the same space and applied cosine similarity to find the relevant documents for each query using both the TF-IDF spaces and BM25 spaces.

9. Now, we move on to the 2nd part of the assignment, where we remove the stop words from the documents and queries and then repeat all the above steps.
10. Now, we calculated both the TF-IDF and BM25 matrices again but this time, the stop words were removed.
11. Coming to the comparison, there was one major but obvious difference and that was the decrease in the vocabulary. This led to the better computation efficiency as well because we are not working with these common words, and this way we can capture a better relation between the document and the query.
12. On the boolean model, the impact was significant, because value of p where we can get any document to match with query dropped drastically. This can be attributed to the reason that most of the words, query and documents had in common were mostly stop words only.
13. On the vector model, for TF-IDF space, the rank of relevant documents did not change much but as we keep on progressing in the list further, the rank of later documents is affected a lot. In BM25 space, the results of relevant documents significantly changed because of the reason, now stop words are not influencing the results any more. So, now we are having more meaningful results. Moreover, overall performance for both of the spaces will increase.
14. Further, we did the document clustering on both the matrices by using K-means clustering, to cluster the similar documents together and results of that have been printed and displayed as well using a graph by reducing the dimensions using PCA.

KEY FINDINGS:

1. First of all, the removal of stop words is necessary because in majority cases, they don’t provide any meaningful context to the things.

2. Rather than using the Porter Stemming Algo, we could have used KROVERTZ Stemmer.

BOOLEAN MODEL:

1. This basically is very efficient in the terms of computations.
2. Very easy to represent as well.
3. But not very user friendly, because user needs to have a very good understanding of the Boolean operations in order to get the desired results. If not done properly, can result in too many or too few of the results.
4. Doesn't capture the context of the queries and the documents and does the retrieval only on the basis of the presence or absence of a particular term.
5. Doesn't keep the account of number of terms present in the document.

VECTOR MODEL:

1. Very good for finding the relevant documents for the set of queries.
2. But takes a lot of time to implement as each document and query has to be represented in the vector form and further COSINE similarity is applied which is computationally expensive.
3. Depends on what kind of weighing scheme one has used which gives different results as shown in the notebook.

TF_IDF:

1. Keeps the term frequency and rarity of words in the final output as well.
2. Takes a lot of time to compute.
3. Doesn't account for Term Saturation.
4. Doesn't penalise for Document Length.

5. Doesn't keep the context of the queries and the documents like Deep Leaning Models like BERT, etc.

BM25:

1. It is same as TF-IDF but here is the thing, it takes the Term Saturation and Document Length in account.

2. Still takes a lot of time to compute because can't utilise GPU for parallel processing.
3. This also doesn't have the context of the sentences like we have in Deep Learning Models.

Stemming:

1. This is important or we will have the same variations of a single word which will increase the vocabulary by many folds increasing the computation time.
2. Words come on to the same root word which increase the term frequency of a particular word telling more about its context.

Stop Word Removal:

1. This also becomes necessary as mentioned earlier, that majority of the times, they are not even required as they provide no meaning to the documents and queries.
2. They just increase the vocabulary size which is not required.
3. Moreover, this increases computation time as well and in return, it doesn't contribute anything.
4. Removal of stop words actually leads to better results as we need not keep the account of these words which means we can focus on the words which actually are important for the terms of Information Retrieval.

Document Clustering:

1. Here basically the results can be attributed to the facts that grouping is done on the basis of similarity of the documents.
2. The documents will be basically clustered on the basis of the vectors in the matrix.
3. This attributes to the fact that the documents which are closer to each other on the data space will be assigned in the same cluster.

Rest of the things about the model and other things are already mentioned in this pdf above.