**HW1 Movie Review Classification**

**Report**

Full name: FNU Krithika Jain

G Number: G01462514

MinerUserName: fkrithik

Rank: 108

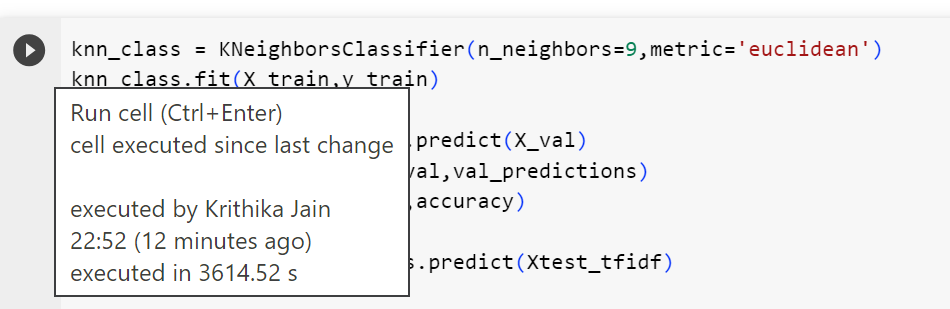
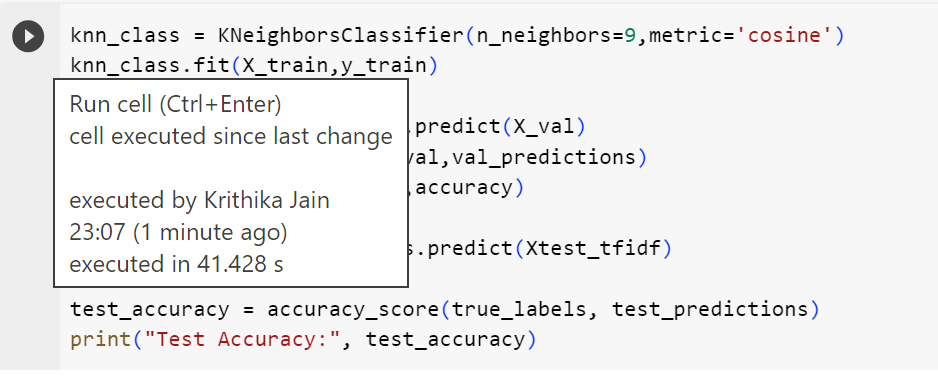
Accuracy: 0.80

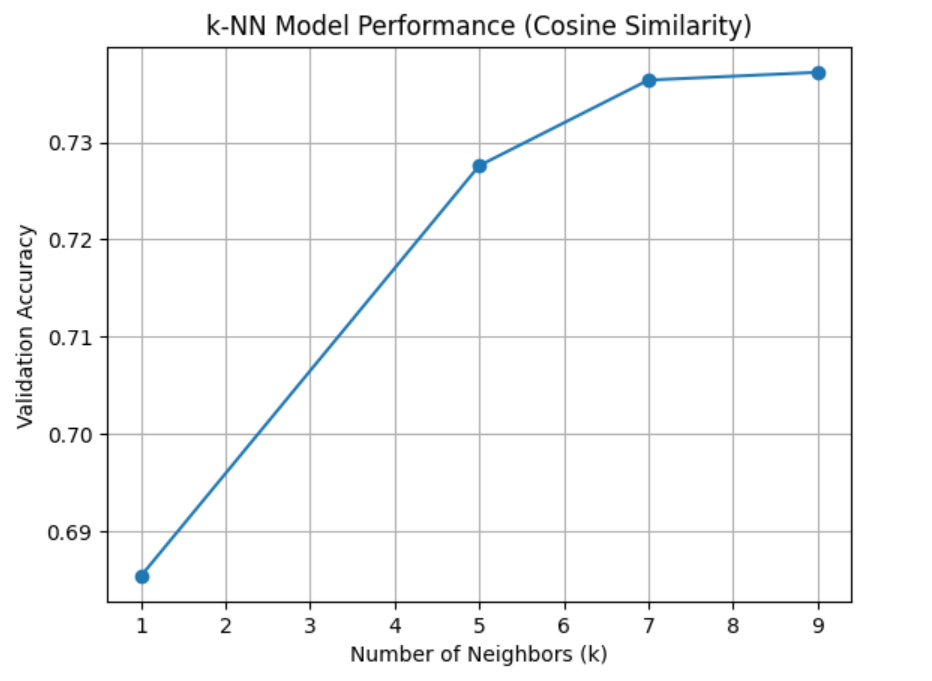
Instruction to run the code:

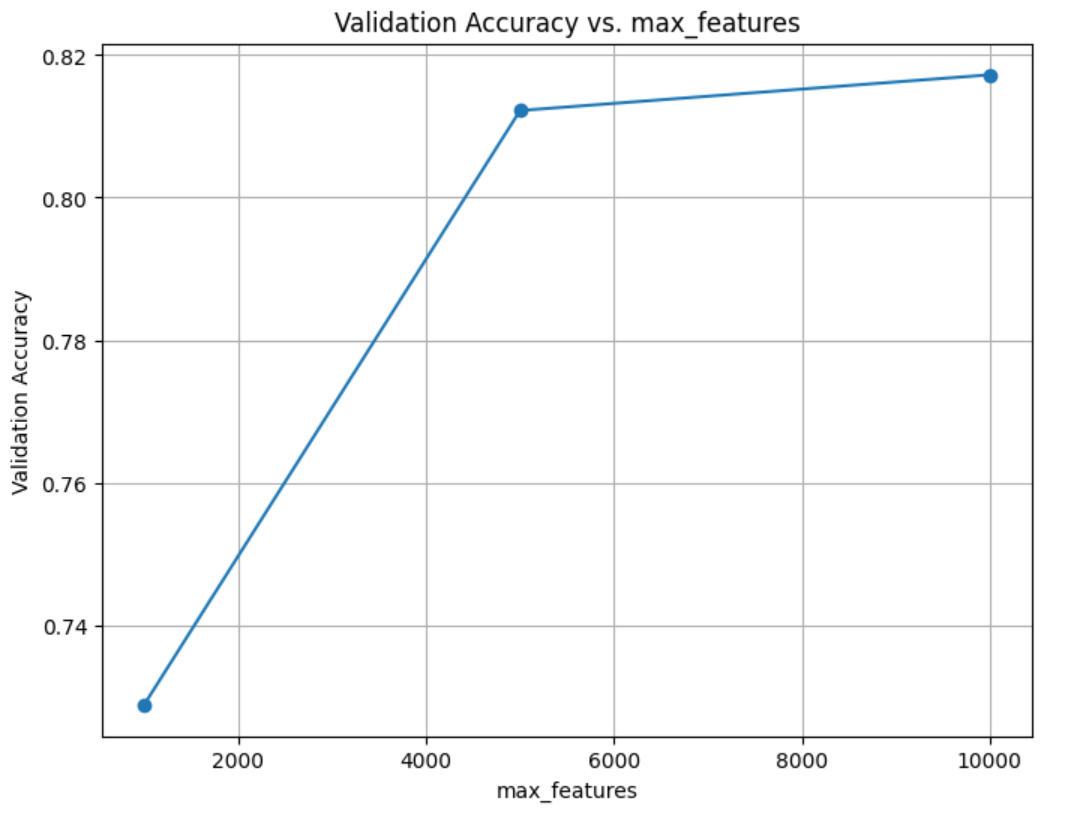
* The code is implemented using python on Google Colab platform, and various parts of code is divided in each cell that are to be run in order.
* To upload the dataset files, click on the files folder on the left hand side 🡪upload to session storage

Approach

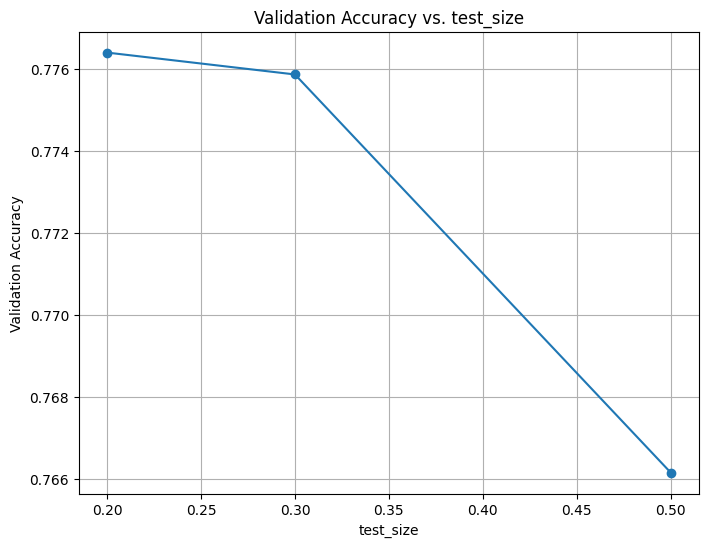
* The datasets were loaded and read into lists, the training dataset was split into reviews and sentiments using string and list manipulation.
* I have pre-processed the train and test dataset provided, using the regular expression to remove the unnecessary characters, libraries from nltk package to remove stopwords, performing stemming and lemmatization.
* First, ran the in-built k-nn classifier with different metric approaches in my experiment it includes the Euclidean distance and Cosine Similarity. Found out that cosine similarity ran faster for text data than the Euclidean distance. As shown in the image below

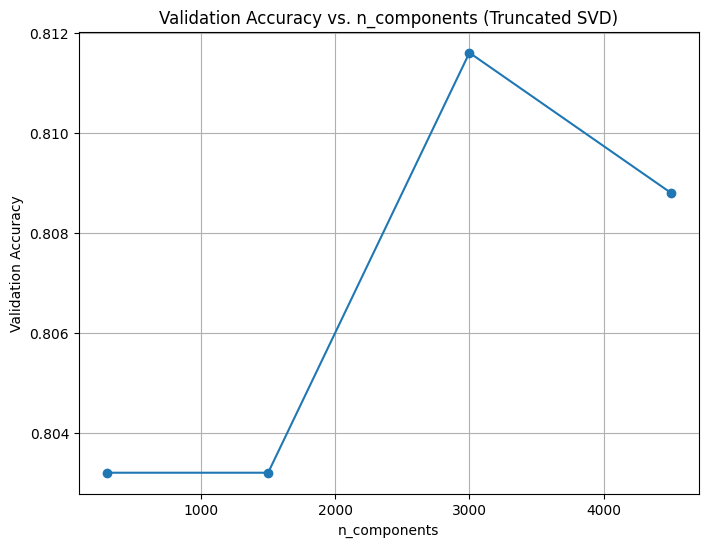
* Built my own knn-classifier, to identify the best nearest neighbor that helped to predict the reviews with the accuracy measured on validation data. Validation data is obtained from training data by splitting it into 80-20 ratio. 
* Used TF-IDF vectorization technique to convert the pre-processed data into numerical features. Controlled the maximun number of features setting it to 10000 (since it showed a higher prediction), min\_df as 2% and max\_df as 95% (filtering based on document frequency) uses default n\_gram as(1,1).



* Train-test Split – Here the training data is split into training and validation datasets with the most common split 80-20. This enables to evaluate the model's performance on a validation set before making predictions on the test set. The test-size I experimented with were 0.2,0.3 and 0.5. The accuracy of 0.2 showed higher prediction.



* Dimensionality Reduction here I’m using Truncated SVD that was beneficial in reducing the dimensionality of the data while retaining most of the variance. I also tried to implement the PCA technique but due to the sparse nature of the matrix, SVD seemed more reliable. Here I have experimented with different values to find the optimal one for my use case. (components is set to 3000)



* Throughout the training phase, I experimented with different values of K and evaluated their impact on prediction accuracy. After testing various values, I observed that the best performance was achieved when setting K to 157. This particular value appeared to strike a balance and provided an accuracy that was representative of the dataset's characteristics.
* Model Training and Evaluation: I have trained the KNN classifier using the training data and evaluate its performance on a validation set, measuring accuracy. And after hyper tuning and Cross Validating the validation dataset the final predictions on the test data were made, along with it’s accuracy compared with the ground true values. The Predictions were then written to an output file named test\_predictions.txt.