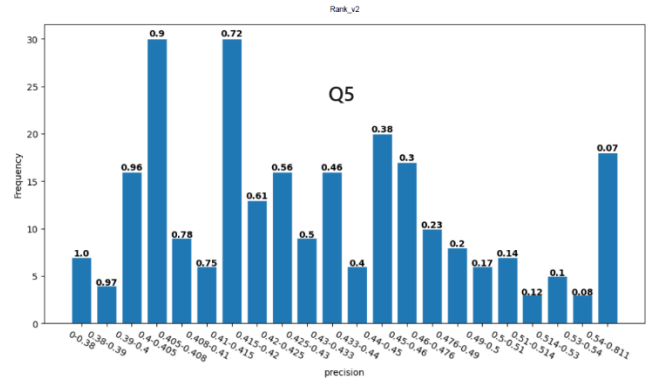
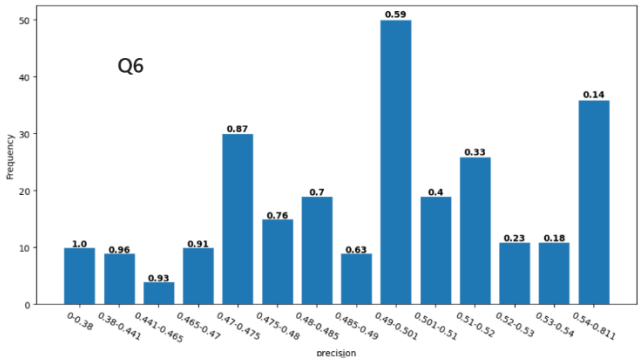
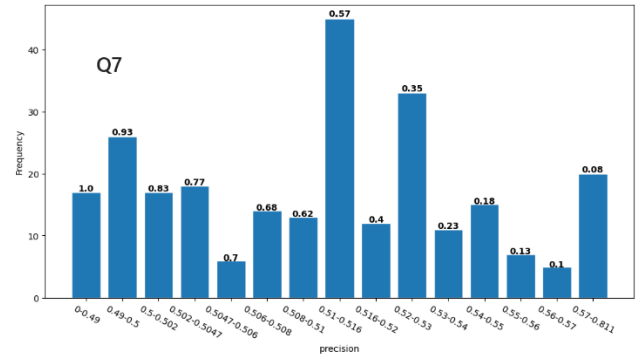
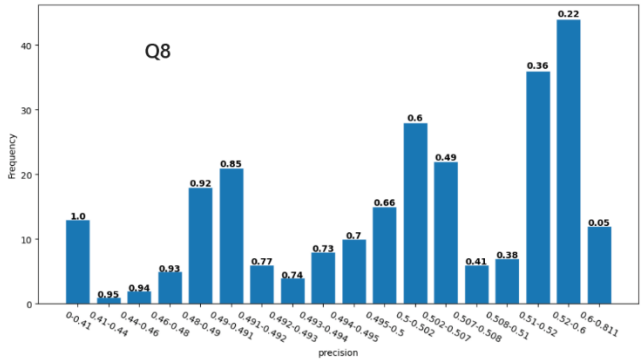
Distribution of Results









Top Students and their approach

1.

Cleaning:
1. nan is drooped from data to prevent error downstream.
2. Then it is converted to lower cases using str. lower
3. Url is removed using replace function in pandas and regular expression.
4. Nonalphabetical is removed using replace too.
5. Extra spaces are removed using replace
6. Contraction is removed using contractions.fix function from lib contractions.

Pre-processing:

1. remove the stop words
2. Perform the lemmatization

Grid search:

Perceptron is then fitted and applied while searching for optimal alpha and l1 ratio using Grid search cv. The optimizer cannot seem to find any difference using different values. The weighted average f1 was 0.61

SVM is then fitted using both the RBF and poly kernel. Gridsearch cv is applied and optimal value was found at (C:1.6,Gamma:0.8) for RBF and (Degree:1,Gamma:1.4) for poly. Generally, a polynomial kernel will have a slightly better f1 score than RBF kernel. But still, both best kernels were constructed and the one with a better f1 score was chosen for the result. Weighted average f1 was 0.65 for poly.(0.64 for RBF)

Logistic regression was then applied, and an optimal C value of 2.0 was found. Weighted average
f1 was 0.65

multinomial naive Bayes was then applied, and an optimal alpha of 0.08 was found. Weighted average f1 was 0.59

2.

Cleaning:

1. Removing HTML Tags
2. convert uppercase to lowercase
3. Tokenize and Remove non-alphabetical characters
4. Perform contractions on the reviews

Preprocessing:

1. Remove stop words
2. Perform lemmatization

Feature Extraction:

1. Remove low variance features
    a. from sklearn.feature_selection import VarianceThreshold
    b. Remove all features retrieved from TFIDF that have very low variance (i.2 less than 0.00002)
    c. It also considerably reduces the training and testing time
2. Dimensionality Reduction
    a. from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
    b. We use Linear Discriminant Analysis (LDA) to reduce dimensionality to the features which have the maximum impact. LDA uses a matrix factorization method to achieve this. We use LDA because it maximizes the separability between the classes.

Model:

1. Grid_search_best_model: Finds the best fit model by searching through all the parameters

3.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

wordVector = CountVectorizer()
finalWordVocab = wordVector.fit( df.review )
bagOfWords = finalWordVocab.transform( df.review )
tfIdfObject = TfidfTransformer().fit( bagOfWords )
finalFeature = tfIdfObject.transform( bagOfWords )

xtrain, xtest, ytrain, ytest = train_test_split(finalFeature, star, test_size = 0.2)
```

It looks like he uses the bag of words first and then Tfidf to get higher results

4.

Cleaning:

1. Removing all URLs from the data - Here I identify any sentence starting with HTTPS, HTTP or www and I remove this sentence so as to ensure there are no links within the data
2. Removing HTML - Using the BeautifulSoup Library I remove HTML tags from the data such as and more. I have used the IXML parser since it's extremely fast compared to the default HTML parser in beautifulsoup which ensures speed.
3. Converting to Lower - This step is performed to convert all text to lowercase.
4. Perform Contractions - I perform contractions on the data such as converting wouldn't to would not using the contractions library using contractions. fix.
5. Removing Non-Alphabetical Characters - Using regex I remove all characters which are not A-Z or a-z.
6. Removing Multiple Spaces - I remove multiple spaces since there is a possibility that the data has two or more spaces in between or at the start/end of the string.

7. Dropping Null Values - After performing intensive cleaning tasks such as Removing URLs, HTML, non-alphabetical characters, and more there could be a possibility that the data has now some NA values which are not required anymore thus I have dropped them

8. Remove Strings with 0 length - After some analysis, there were some strings that did not have any characters thu

Preprocessing:

1. Finding and Removing Stopwords
2. Performing Lemmatization

Tfidf:

1. For Tf-idf rather than just using the basic version, an n-gram approach is taken for 1 to 3 n-grams. After performing ngrams we also perform the select k-best to find the best features according to the k. Hereafter performing analysis on k=50k, 100k, 200k, 250k, and 300k the best result was at 250k and that was selected

5.

Cleaning:

1. Converting text to lowercase
2. Using BeautifulSoup to extract text removing any HTML tags
3. Regex is used to remove any URLs/ Links
4. Contractions are performed using the existing Python library. Contractions are performed before removing non-alphabetical characters to avoid removing apostrophes ahead since after removing we won't be able to expand the word. (don't => do not instead of dont) •
5. Non-alphabetical characters are removed using regex • Extra spaces are removed

Tf-IDF::

In order to get better results, the SelectKBest method is used to extract top K features. After trial and error, the optimal k value comes out to be 2^15 = 262144 Additionally, passed the ngram_range argument with value as (1, 3) to include bigram and trigrams along with unigram. Using higher n-grams can be helpful in identifying that a certain multi-word expression occurs in the text that the BoW model will ignore.