INFO 371 PS3 Report

Question 1

a. (2 pts) Explain how to turn P(E|H) to P(H|E), with E=Evidence and H=Hypothesis in layman's terms.

Bayes' Theorem is a concept in probability that fine-tunes our belief about a hypothesis (H) upon encountering new **evidence** (E). This theorem lets us update the initial likelihood of a hypothesis (P(H)), known as the **prior**, by considering the frequency at which the evidence is associated with the hypothesis—this is the **likelihood** (P(E|H)). By factoring in the general probability of encountering the evidence (P(E)), we revise our stance to arrive at a more informed belief, termed the **posterior probability** (P(H|E)). In a Naive Bayes setting, this approach is harnessed to forecast outcomes, treating each piece of evidence as an independent factor that impacts the hypothesis's likelihood, thus refining our predictions by integrating all available evidence.

b. (2 pts) Use an example from real life to ground the explanation.

Imagine you're a doctor trying to diagnose a patient. You have a **hypothesis** that the patient has a certain disease (let's call it Disease A). Bayes' Theorem helps you update the likelihood of the patient actually having Disease A after you get a new test result (**the evidence**). Initially, you have a general idea of how common Disease A is among patients (**this is the prior probability**). When the test results come in, you consider how likely it is to get such a result if the patient really has Disease A (**this is the likelihood**). Bayes' Theorem combines your initial assumption about Disease A's prevalence with the new test result to give you a revised probability that the patient has Disease A. This revised probability **(the posterior**) is a more informed assessment, considering both the general prevalence of the disease and the specific test results for this patient. In essence, Bayes' Theorem helps the doctor go from a broad assumption to a personalized diagnosis by incorporating new, specific evidence.

Question 2

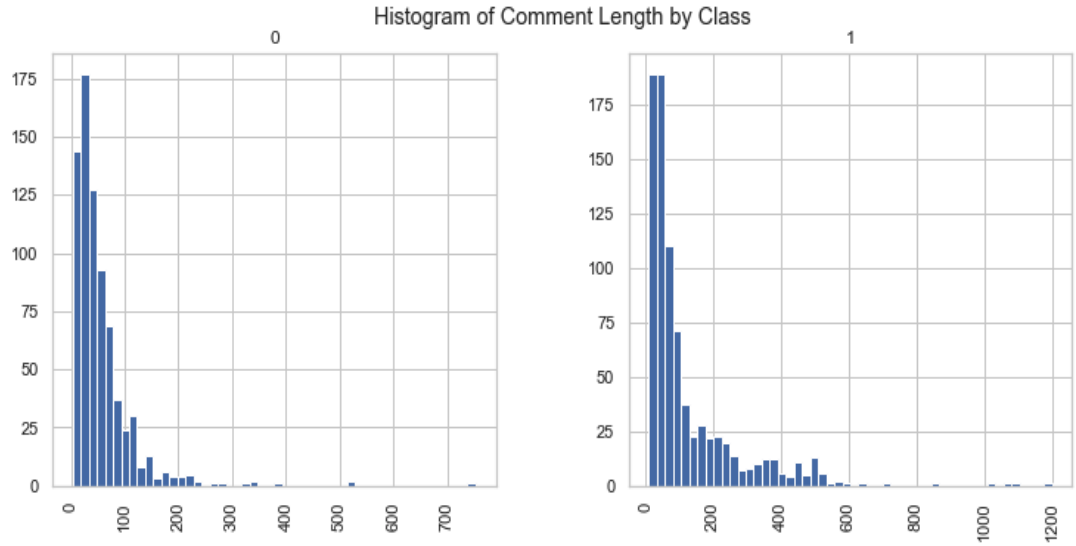
For this exercise, use any four of these five datasets to build a spam filter with the Naïve Bayes approach. Use that filter to check the accuracy of the remaining dataset. Make sure to report the details of the training process and the model.

a. Use any four of these five datasets to build a spam filter with the Naïve Bayes approach.

- **Title:** Spam Filter Training Report for YouTube Spam Collection Datasets
- **Introduction:** This report outlines developing a spam filter for YouTube comments using a Naïve Bayes classification approach. The goal of this filter is to accurately distinguish between spam (class 1) and non-spam (class 0) comments.
- **Training Datasets Used:**

- Four datasets were combined for this task:
    a. Youtube01-Psy.csv
    b. Youtube02-KatyPerry.csv
    c. Youtube03-LMFAO.csv
    d. Youtube04-Eminem.csv

- **Step 1: Load training data, look around**
  The following steps were taken to preprocess the data:
    - **Loading Data:** The datasets were loaded into Pandas dataframes and concatenated into a single dataset with 1586 entries and 5 columns.
    - **Initial Exploration:** The dataset was checked for missing values, and basic statistics were gathered. Notably, the 'DATE' column had 245 missing entries. Since the 'DATE' column is not used when building and training the model, it is not necessary to fill in these missing values. Typically, feature selection is based on data that has a direct impact on the model's predictive power. If the date information is not relevant to our prediction goals or does not provide valuable additional information about the prediction results, this feature can be omitted.
    - **Descriptive Statistics and Content-Length Analysis:**
        a. **Class Count:** The 'CLASS' column, which identifies whether a comment is spam (1) or not spam (0). There are 1586 values in the 'CLASS' column, including 755 not-spam content and 831 spam content.
        b. **Comment Frequency:** Comments in the dataset show varied frequency, with the possibility of duplicates. The dataset's most frequent spam comment occurred 93 times, signaling potential common spam phrases or links.
        c. **Content-Length:** A 'CONTENT_LENGTH' column was introduced to represent the number of characters in each comment. On average, comments are about 91 characters long. The comment lengths vary widely, stretching up to 1,200 characters for the longest comment.



        d. **Visual Analysis:** Histograms reveal that most comments, whether spam or not, are concise. There's a noticeable occurrence of shorter comments in both categories, but non-spam comments show a slightly wider distribution in length.

The left histogram (Class 0) demonstrates a high frequency of short comments, peaking at under 50 characters. The right histogram (Class 1) shows that spam comments are generally brief, with the majority under 200 characters. Both histograms exhibit a tail-off as comment length increases, with very few comments approaching the 1,200-character maximum.

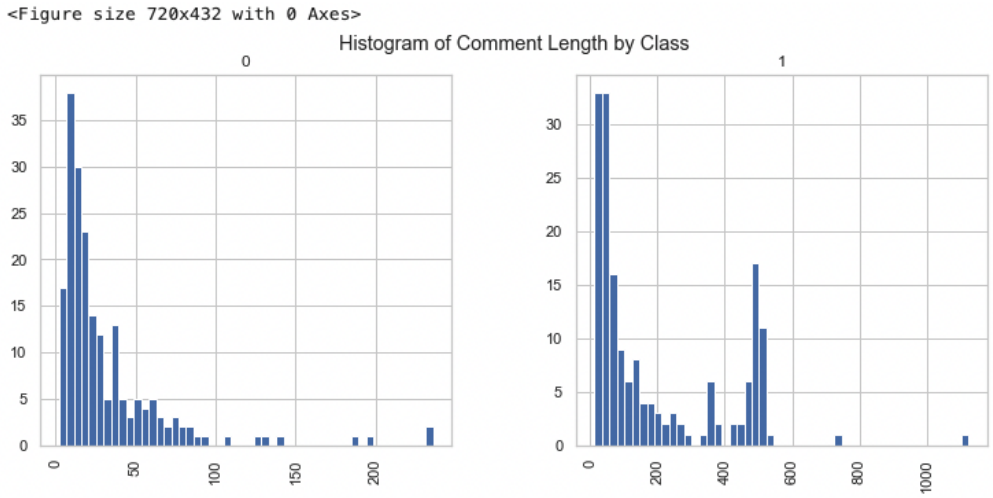
- **Step 2: Data preprocessing**
  In terms of the data preprocessing, we employed the Natural Language Processing library NLTK for text data and created a new function called "nltk_preprocess" to perform the following actions:
  - **Tokenization:** We utilized word_tokenize to break down comments into lists of words.
  - **Lowercasing:** All words were converted to lowercase for format consistency.
  - **Stop Words Removal:** English stop words, which are common but not significant for meaning classification, such as "is," "and," "the," etc., were removed from the text.
  - **Lemmatization:** We applied WordNetLemmatizer for word lemmatization to ensure uniformity in word forms.

  These steps helped us to reduce the number of features for the model to process and increase feature quality.
- **Step 3: Data to vectors**
  Textual data cannot be directly processed by machine learning algorithms, which necessitate numerical input. To convert the raw 'CONTENT' into a usable format for algorithmic processing, two main transformations were employed.
  - **Bag-of-Words Model:** Utilized CountVectorizer to transform the 'CONTENT' column into a Bag-of-Words (BoW) model. This model counts the number of times each word appears in a document, constructing a vocabulary of 3,883 unique words from our dataset. The analyzer=nltk_preprocess parameter ensures that our text data is first processed through the custom nltk_preprocess function. This function tokenizes the text, converts it to lowercase, removes stopwords, and applies lemmatization to condense words to their base or root form.
  - **TF-IDF Transformation:** Following the creation of the BoW model, TfidfTransformer was applied. TF-IDF stands for Term Frequency-Inverse Document Frequency. This weighting scheme evaluates how important a word is to a document in a collection or corpus. The fit process computed the IDF values across our text data, which reduces the weight of terms that occur very frequently and increases the importance of terms that occur rarely. The resulting TF-IDF matrix has dimensions of 1,586 (the number of documents) by 3,883 (the number of unique words), reflecting the weighted frequency of each word in each document. The dimensionality of the TF-IDF matrix implies that we have a rich feature space to represent the nuances of the text data. Each of the 1,586 documents is now represented as a vector in a 3,883-dimensional space, where each dimension corresponds to a term from our processed vocabulary.
- **Step 4: Training a model, detecting spam**
  - **Training the Multinomial Naïve Bayes Classifier:** The classifier was trained using the TF-IDF matrix as its input, where each row represents a comment, and each column represents the TF-IDF score for a specific word in the vocabulary. The 'CLASS' column,

which labels comments as spam (1) or not spam (0), served as the target variable during training.

b. Use that filter to check the accuracy of the remaining dataset.

- **Step 1: Load and Preliminary Exploration of Test Data**
  - **Loading the Test Dataset:** The test dataset, 'Youtube05-Shakira.csv', was loaded into a pandas DataFrame. The data comprised 370 entries, each with 5 features: 'COMMENT_ID,' 'AUTHOR,' 'DATE,' 'CONTENT,' and 'CLASS.' An initial assessment was conducted to check for missing values. The dataset proved complete, with no missing entries across all features.
  - **Class Count:** The 'CLASS' column, which identifies whether a comment is spam (1) or not spam (0). There are 370 values in the 'CLASS' column.
  - **Content Analysis:** The descriptive statistics for the 'CONTENT' feature were stratified by 'CLASS.' Among non-spam comments (Class 0), there were 196 unique entries, with the most common comment related to admiration for Shakira, appearing 4 times. In contrast, the spam category (Class 1) had 174 unique entries, with the top spam content appearing 4 times, potentially indicating spam patterns.
  - **Comment Length Exploration:** A new column, 'CONTENT_LENGTH,' was created to quantify the length of each comment in terms of the number of characters. The average comment length was approximately 109 characters, with a significant standard deviation(162), suggesting considerable variability in comment lengths. The minimum comment length was 2 characters, and the maximum reached 1,125 characters. The 25th percentile was at 17 characters, the median at 37.5 characters, and the 75th percentile at 98 characters, highlighting that most of the comments are relatively short.



  - **Visual Representation of Data:** Histograms of 'CONTENT_LENGTH' stratified by 'CLASS' were plotted. The histograms indicated that most comments are concise, with a majority falling within shorter length intervals. The histogram for non-spam comments exhibited a broader length spread than spam comments, with a noticeable peak for shorter comments. The tail of the distribution for both classes confirmed that longer comments

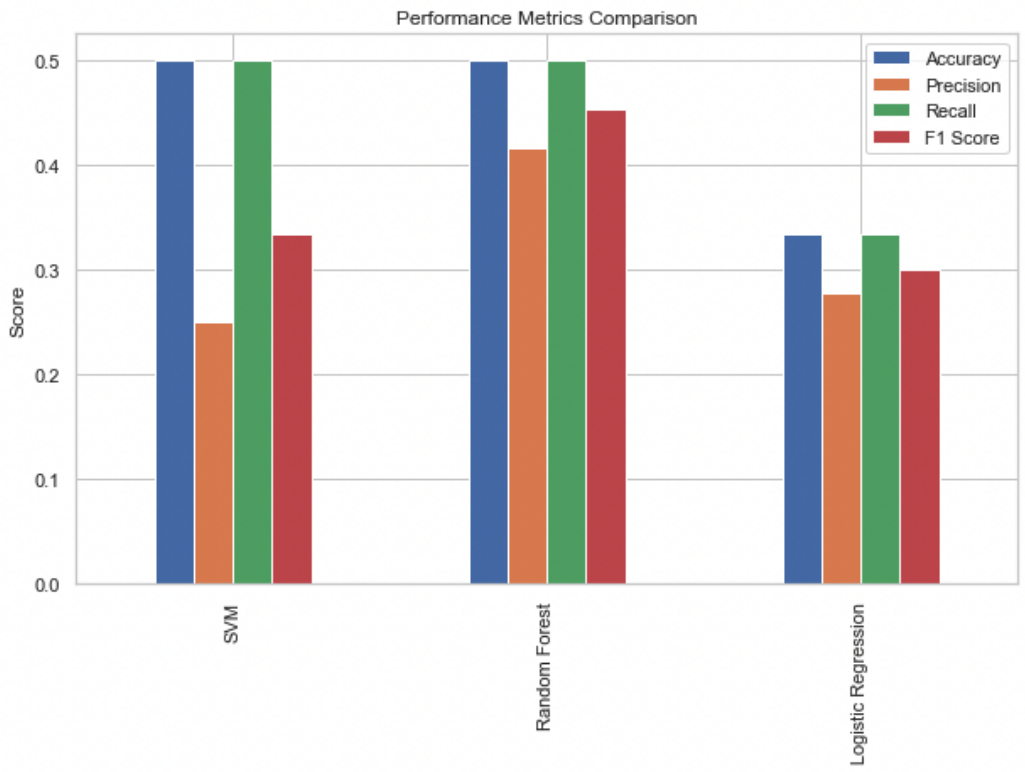
are less frequent, reinforcing the observation that both spam and non-spam comments are typically brief.

- **Step 2: Model Evaluation**
  The model's performance was evaluated on the test dataset 'Youtube05-Shakira.csv'. Predictions were made using a pipeline that combined a TF-IDF vectorizer and a Multinomial Naive Bayes classifier, both of which had been fitted on the combined training data.
  - **Understanding Performance Metrics:**
    a. Accuracy (88.92%): The accuracy of the model on the test data was 88.92%, indicating that the model correctly identified spam and non-spam comments with a high rate of success.
    b. Precision (85.95%): Precision, which measures the proportion of true positive predictions in all positive predictions, was at 85.95%. This suggests that when the model predicted a comment as spam, it was correct 85.95% of the time.
    c. Recall (91.38%): Recall, assessing the proportion of actual positives correctly identified, stood at 91.38%, showing that the model captured a high percentage of actual spam comments.
    d. F1 Score (88.58%): The F1 score, a harmonic mean of precision and recall, was 88.58%, reflecting a balanced model with high precision and recall.

    The high values across these metrics suggest that the model is accurate and robust in distinguishing between spam and non-spam comments.
  - **Confusion Matrix Analysis:** The confusion matrix provided further insights into the model's performance, revealing the number of true positives, false positives, true negatives, and false negatives. Specifically, the matrix [[170, 26], [15, 159]] implies that there were 170 true negatives (correctly identified non-spam), 159 true positives (correctly identified spam), 26 false positives (non-spam incorrectly labeled as spam), and 15 false negatives (spam not detected).

Question 3

Use an SVM-based model to predict the Portuguese outcome of the battle from the number of ships involved on all sides and Spanish involvement. Try solving the same problem using two other classifiers that you know. Report and compare their results with those from SVM.



Here is the summary of the performance metrics obtained from the SVM, Random Forest, and Logistic regression models:

- SVM:
    a. Accuracy: 0.5
    b. Precision: 0.25
    c. Recall: 0.5
    d. F1 Score: approximately 0.333
- Random Forest:
    a. Accuracy: 0.5
    b. Precision: approximately 0.417
    c. Recall: 0.5
    d. F1 Score: approximately 0.452
- Logistic Regression:
    a. Accuracy: approximately 0.333
    b. Precision: approximately 0.278
    c. Recall: approximately 0.333
    d. F1 Score: 0.3

From our output and the plot, we can know that:

- The SVM model has an accuracy and recall of 0.5, but it has the lowest precision (0.25) and the lower F1 Score (approximately 0.333) among the three models.

- The Random Forest classifier performs the best out of the three models. It shares the same accuracy with the SVM at 0.5 but has improved precision (around 0.417) and F1 Score (approximately 0.452). This indicates that the Random Forest classifier better balances precision and recall for this particular task.
- The Logistic Regression classifier shows the poorest performance on this task with the lowest accuracy (around 0.333), lower precision (approximately 0.278), lowest recall (around 0.333), and the lowest F1 Score (0.3). This means the model is the least accurate at predicting the outcome of the Portuguese naval battles compared to the other two models.

In conclusion, the Random Forest model has the highest accuracy score, balancing precision, and recall better, providing the highest F1 Score, suggesting it might be the best choice for this particular dataset. The SVM, while having comparable accuracy to Random Forest, has relatively lower precision and F1 Score, indicating it may not be the best model for predicting the battle outcomes. Logistic Regression performs worse across all metrics than the other models, indicating it might not be suitable for this particular problem.

Reflection on problem 3: The performance of the SVM, Random Forest, and Logistic Regression models is less than ideal, which could largely be due to the quality and scope of the dataset. The potential issues include imbalanced classes, sparse data, and a lack of more nuanced features that could provide a deeper context for each battle, such as crew experience or ship condition. Small dataset sizes and possible anomalies could also hinder the models' ability to learn and generalize. Enhancing the dataset and incorporating more complex features might improve model performance.