**SVM for App Review Analysis – Madilyn Coulson**

**Motivation and Problem Description**

App reviews left by customers are crucial for app developers. The feedback can offer valuable insights, such as reporting missed bugs, giving suggestions for future updates, or requesting new features and functionality. However, app reviews may also be expressions of personal preference, like simple complaints or positive affirmation that the app is in good condition. App reviews can also be simple complaints that are moreover preference rather than actionable app functionality, or positive feedback stating that the app is in great condition with no changes needed. Developers need a way to determine whether an app review is actionable, and if so, what type of functionality fix is needed.

Hence, the goal of my project is to establish a way in which app reviews can be input into a program, and the program in return identifies actionable items and specifies corresponding functionality fixes required. To achieve this, I created a Python program with an imbedded Support Vector Machine (SVM), a machine learning model explained in the Support Vector Machine Details section, to assess if the input app reviews are actionable. Along with this, I developed four distinct SVMs to determine whether the actionable reviews are associated with crashes, speed issues, UI/UX concerns, or security bugs. The details and results of this process are described below.

**Support Vector Machine Details**

A Support Vector Machine (SVM) is a machine learning model designed to analyze data for classification and regression analysis. It falls under the category of supervised learning models, where input values and desired output values train the model. In implementing this, I utilized the scikit-learn (sklearn) module in Python, an open-source module used for a wide range of machine learning tasks, including SVMs. Within scikit-learn, I specifically utilized the SVC and TfidfVectorizer class and several metric functions. The SVC class executes the SVM algorithm for classification, and the TfidfVectorizer class converts raw text into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features which are used to train the SVM. The accuracy\_score, precision\_score, recall\_score, and f1\_score functions calculate the subset accuracy, precision, recall, and f-score by assessing the match between the set of prediction labels and the corresponding real labels.

In terms of the SVMs developed within this project, they were trained using datasets of 500 simulated app store reviews generated from ChatGPT. Each individual review in the datasets received a corresponding label - 1 denoting relevance to the specified field and 0 indicating lack of relevance to the specified field. For example, in the SVM evaluating whether the reviews are actionable, a label of 1 signifies the review is actionable, while a label of 0 indicates the review is not actionable. It is crucial to note that there were instances where labels had to be changed manually due to inaccuracies in ChatGPT. Therefore, every label was dually checked to ensure that it was correctly and accurately determined. This data underwent processing and utilization by the TfidfVectorizer class first and then the SVC class, as previously mentioned. Following this, the SVM’s accuracy was tested by implementing another set of 100 simulated app store reviews generated by ChatGPT. These new app store reviews were also assigned corresponding correct labels. The SVM was then employed to predict the labels for the new app store reviews, and the predicted labels were then compared to the actual labels with the accuracy\_score function as described above. This comparison provided the accuracy of the SVM on the dataset. The objective was to ensure each SVM achieved over 90% accurate before applying to real data, with a preference for accuracy levels of 95% or above. Along with this, the precision, recall, and f-score of the model was also calculated.

**Development Steps**

First, the actionable SVM was developed and tested as described above. Following this, the 100 most recent Google Play Store Reviews for the social media app Snapchat were manually pulled and placed into a text list (SnapchatReviews.txt), as well as in a Python array. The data that was returned as actionable was placed into a text list (ActionableSnapchatReviews.txt), as well as in a Python array. Following this, the SVM’s for crashes, security, speed, and user interface were developed and tested as well. The data identified as actionable was processed and returned by each corresponding SVM. The subsequent section details the outcomes of these runs and the associated actions.

To note, each general SVM file is named SVM*<type>*.py. Then, the file with the corresponding SVM used to test the Snapchat data is named SVM*<type>*Snapchat.py. Finally, the output of the Snapchat data returned from each SVM is named <type>SnapchatReviews.txt. For example, the actionable SVM’s are named SVMActionable.py and SVMActionableSnapchat.py, and the actionable text file is named ActionableSnapchatReviews.py. Along with this, each pair of SVM’s and its text file are in its type folder within src in the GitHub repository (the SVMs and text file listed above are in the directory src\Actionable).

**Results**

The following table depicts the accuracy of the trained SVM, resulting number of reviews returned from the Snapchat data, and the percentage of Snapchat data returned for each SVM:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Actionable SVM | Crash SVM | Security SVM | Speed SVM | User Interface SVM |
| Accuracy of Trained SVM | 94.0% | 100.0% | 100.0% | 98.0% | 98.0% |
| Precision of Trained SVM | 0.9464285714285714 | 1.0 | 1.0 | 1.0 | 0.9565217391304348 |
| Recall of Trained SVM | 0.9464285714285714 | 1.0 | 1.0 | 0.9393939393939394 | 1.0 |
| F-Score of Trained SVM | 0.9464285714285714 | 1.0 | 1.0 | 0.96875 | 0.9777777777777777 |
| Number of Reviews Returned | 63 | 2 | 0 | 7 | 27 |
| Percentage of Reviews Returned | 63% (63/100) | 3.17% (2/63) | 0% (0/63) | 11.11% (7/63) | 42.86% (27/63) |

As indicated in the table, the trained SVM’s achieved a 94% or higher accuracy when tested with 500 training data points and 100 test data points. The precision, recall, and f-score metrics were also substantially high and strong for each SVM. Upon applying the Snapchat data to the Actionable SVM, 63 of the 100 reviews were returned as actionable. Among the 63, 2 were identified as relating to crashes, 7 to speed, and 27 to UI/UX design. It was observed that 1 data point in the crash category overlapped in the UI data, and 3 speed-related data points were shared in the UI category. Therefore, there was 1 unique data point for crashes, 4 unique data points for speed, and 23 unique data points for UI, resulting in a total of 28 unique data points. This analysis shows that the SVMs were associated with 44.44% of the actionable data set.

**Results Analysis**

The results from the SVMs reveal several significant themes worth highlighting. Firstly, the SVMs exhibited high accuracy, precision, recall, and f-score on the training data, suggesting that the outcomes of the Snapchat review analysis are likely to be reliably accurate. In addition, the SVM successfully identifies 63 out of 100 of the Snapchat reviews as actionable, demonstrating its ability to pinpoint a substantial portion of reviews requiring further attention or analysis, which is critical for developers. Furthermore, the breakdowns provided into the categories of crashes, speed, and UI design offer valuable insight into prevalent problems that users encounter, UI being at the forefront of these issues. Ultimately, the SVMs were able to cover 44.44% of the actionable data set, indicating that they were generally effective in identifying and categorizing issues within the Snapchat reviews.

Although the SVMs demonstrated overall success, it's essential to acknowledge certain flaws. The analysis uncovered instances of overlap, where data belonged to multiple categories, suggesting potential interrelations between certain types of issues. Along with this, no issues were returned in the security category, which could indicate that this model was improperly trained. Lastly, some reviews returned in their designated categories lack accuracy in their descriptions. By manual evaluation, at least one crash review, three speed reviews, and three UI design reviews were reported incorrectly. This implies that the models may require stronger or more comprehensive training data to enhance their precision.

**Conclusion**

In conclusion, the purpose behind this project stemmed from the importance of app reviews for developers, as they provide valuable insights into user experiences and potential issues. The goal was to create a program using Support Vector Machines (SVMs) to identify actionable items in app reviews and specify corresponding functionality fixes. The SVMs, trained with a dataset of 500 simulated app store reviews, demonstrated high accuracy levels, with a preference for 95% or above. Applying these SVMs to Snapchat reviews revealed that 63 out of 100 were actionable, with a breakdown into categories such as crashes, speed issues, and UI/UX design. While the SVMs were generally effective, the analysis uncovered instances of overlap and indicated potential issues with the security category. Despite these findings, the project successfully showcased the potential of SVMs in automating the identification and categorization of actionable issues within app reviews, providing developers with valuable insights for improvement.

**GitHub Link**

<https://github.com/madicoulson/App-Review-Analysis>

Works Cited

GeeksforGeeks. "Learning Model Building in Scikit-learn: A Python Machine Learning Library." GeeksforGeeks, https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python- machine-learning-library/.

Wikipedia. "Support vector machine." Wikipedia, Wikimedia Foundation, 17 Nov. 2021, https://en.wikipedia.org/wiki/Support\_vector\_machine.