**Please complete before submission:**

**Name: Zeyan Liang**

**Student Number: 2627888L**

Q1a:

After the histogram drawn from the statistics of the number and distribution of multi-labels in the different data sets(Train, Validation, Test), it is found that all data sets are relatively balanced with reasonable labels distribution. It can be noticed that the amount of data labeled soda is higher than that of other labels. An unbalanced training set may result in a model with high accuracy after training, but the model cannot accurately predict the labels of the test set. Therefore, if the training set data is found to be unbalanced, it is necessary to split the data and increase the feature and class re-weighting to solve the problem. This dataset does not have this problem, so we can start training the model.

Chart, bar chart

Description automatically generated

Q1b:

The dummy uses simple rules to make predictions. The main way it works is to completely ignore the input data for label prediction. This leads to poor accuracy of the dummy classifier, which is usually used as a simple baseline to compare with other classifiers. There is a parameter in the dummy classifier called strategy, and the input value of this parameter determines how the classifier works. For example, in Q2B we use dummy classifiers that work in two ways to train the model. The first is ‘most\_frequent’, the predicted value is the category with the highest frequency. This can be seen from the confusion matrix diagram I drew, tea is the most frequent label, so the prediction result of all data is tea.

Then the parameter input is ‘stratified’, and random prediction is given according to the frequency distribution in the training set. Its effect is much better than the previous model, but the accuracy is still not ideal. It can be seen from the drawn confusion matrix that the predicted label distribution is uniform just like the label distribution in the training set. The other two parameters of the dummy classifier are not input here, they are the default ‘random\_state’ and constant, including the parameter strategy, there are actually six options, and only two were tried to train the model in Q1B.

A picture containing text, display

Description automatically generated

Qr code

Description automatically generated with low confidence

The third classifier is LogisticRegression, which is more commonly used to classify linear data. Logistic regression is often used for binary classification. Its principle is that when the data conforms to this distribution, the value of the parameter will be estimated by the maximum likelihood value. In addition to this, the data needs to undergo specific processing before being put into model training. The data will be encoded first using One-hot vectorization. The principle of one-hot encoding is to assume that each feature has m number of possible values. After these features are encoded by one-hot, they become the same number of binary features. The main advantages of this method are as follows: it solves the shortcomings of classifiers in dealing with attribute problems, and on the other hand, the richness of features is also expanded.

The training results of this model are reasonable. The prediction scores for the test set are close to 100%. The precision, recall, and f1-score indicators in the test set are all between 76% and 77%, which also shows that after one-hot Coding the processing of the data, the logistic regression model has a good ability to learn on this dataset.

The confusion matrix plot shows that most of the categories have high accuracy predictions, and then the two categories of PS4 and PCgaming are sometimes wrong, and the reasons for these errors will be analyzed in Q2B.

Important parameters in logical learning such as penalty and C will also directly affect the training results of the model. The penalty determines which norm the model will use to calculate, and C is the Inverse of regularization strength. Usually smaller C results in stronger regularization.

A picture containing text

Description automatically generated

Qr code

Description automatically generated

IDF and TF are weighted as keywords. IDF is insufficient to explain the importance and distribution of words. Because of its own simple structure, it cannot perform effective weight adjustment after the text set has been classified, so that it cannot provide an objective Model accuracy improved. After processing the text using this encoding, the logistic regression model improved slightly on the test set, with scores ranging from 78% to 79%.

However, the distribution of tags is also similar, the main problem still lies in the data that should be pcgaming and ps4 tags. The model is still struggling to predict data for both labels.

In TF-IDF vectorization, three important parameters directly affect the data processing results, the choice of tokenizer, the number of max\_features and whether the sublinear\_tf affects the data set to a certain extent, and their impact will be analyzed in Q2A.

Calendar

Description automatically generated with medium confidence

Qr code

Description automatically generated

The last classifier selection is SVC, which uses One-hot vectorization to process the data. The trained model is similar to the previous one when predicting the test set data. The indexes are between 75% and 80%, and the prediction errors mainly occur. On the data tagged PS4 and pcgaming. C, kernel and degree in the SVC parameters are all important parameters that will directly affect the training results, but here only the kernel is selected as RBF, and its parameters are all default settings.

Calendar

Description automatically generated

Qr code

Description automatically generated

Calendar

Description automatically generated

Background pattern

Description automatically generated with medium confidence

Q1c:

I chose Random Forest as the classifier and CountVectorizer to process the dataset for model training. The test results are similar to the above three models, with scores between 76% and 78%. This time the errors are distributed on the datasets labeled PS4, pcgaming, hydrohomies and xbox. For the input of parameters, I only entered spacy as the tokenizer of the random forest classifier.

The reason I choose random forest is that it is a learning method based on bagging, it is not only used for regression models, but also for classification models with high adaptability. Faster training speed, parameter importance that can be sorted, and resistance to overfitting due to random references.

Q2a:

The first object to tune is C in the logistic regression classifier. C as the inverse of the regularization strength. The value range is greater than zero, and the default value in the model is 1.0. In other words, the ratio of the regular term and the loss function is also one to one. A smaller C results in a smaller loss function for the classifier, and on the contrary, it will have a stronger regularization effect.

Within the range of values, I set the initial C to 10^-3, and each time it is multiplied and incremented until the maximum value is 10^5. And the model training results of all different parameters of C are represented by a line graph.

In my opinion, if the value of C is large, our slack variable will be small, and our allowable error will be small, so the classifier will try to classify the samples correctly, and as a result we will get a hyperplane with small spacing, which may lead to the classification performance of the new sample is very poor, but the classification have a good performance on Training data set. When the value of C is small means we allow a large error, even if the classifier needs Misclassified samples, it will also look for hyperplanes with large spacing, in this case, if your training set is linearly separable, it is also possible to have misclassified samples.

Chart, line chart

Description automatically generated

As can be seen from the figure, before C is equal to 1, Both test score and train score has a significant increase with the increase of C, but after that the entire model has overfitting, so I set C= 1 is the best choice for the secondary parameter. But in practice the parameter C helps very little.

Next, simultaneously adjust the sublinear in the logistic regression classifier and the maximum number of features in the Tf-Idf vectorizer. Here I use the pipeline and GridsearchCV to assist in the parameter adjustment. The value of sublinear is true or false. The maximum number of features is an arithmetic sequence with an initial value of 500 and a step size of 500. The maximum value is 8781, which is also the default number in the vectorization.

A picture containing shape

Description automatically generated

According to the graph, it can be seen from both comparisons that when the sublinear parameter is true, the accuracy of the model is slightly higher than when sublinear is false. After the maximum number of features reaches 7500, the model starts to overfit. Therefore, in this parameter tuning section I think sublinear true and max feature number 7500 are the best choices.

Finally, I chose the tokenizer in TF-IDF. I chose spacy to process the data. I found that the accuracy gap between using spacy as tokenizer and not using spacy was about 4%. This is a significant improvement in the performance of the model. parameter. The frequency of stop words and punctuation marks in the text is very high. If these meaningless tokens are not removed, it will affect the prediction accuracy of the trained model on the test data set.

A picture containing calendar

Description automatically generated

Qr code

Description automatically generated with medium confidence

Q2b:

For the error analysis part, I chose the two labels with the worst predicted performance for analysis, I extracted all the data that should be pcgaming and PS4 that were wrongly predicted, a total of 35, I printed out their Title, wrong label, correct label, and content. I found three important trends and types of prediction errors.

The first is that the content of the post lacks important judgment information, and the title of the post contains this one or more information but is not added to the training set for model training. For example, in the 0th and 18th pieces of data I printed out, the content of the post is only discussing a game that is supported by all platforms, but it does not mention which platform it is. The title clearly indicates that it is the steam platform. This important information can be Judging this post should be classified as pcgaming, but they were both mispredicted as PS4.

The second is that neither the title nor the content of the post has important information that can be effectively used to determine the category, so the prediction accuracy of such posts is low, such as the 1st, 3rd, 6th and so on that I printed out. This type of post is the most, and their main content and title are discussing games or information about games, but these content may occur on all game platforms and are not exclusive. Therefore, this type of information is predicted irregularly, and the wrong labels are distributed irregularly.

The last one is that the content and title of the post have valid information that can be used for judgment, but they are still predicted wrong. For example, print out the 9th and 16th. Their text and title appear PC, or XBOX, but the correct label is not these. These posts are a bit too confusing to be predictable.

Q3a:

I choose to add post title, author as two new features and content in the form of a string combined into a query and input this entire query as the training set. I chose these two data as new features for two main reasons. First, after error analysis, I found that many titles have more effective information that is more important than the content, which can improve the accuracy of the model. The keyword duration is shown in the title, but not necessarily in the content of the post. Secondly, the author is also a very important source of information, because reddit is a social platform, and people who initiate posts always have their own areas and topics of interest. There is a high probability that they will post many posts about a field, so adding the author as a data feature can also help the model to learn better.

Q3b:

I used a method and process similar to Q2 to adjust the parameters of my new model, the parameters include tokenizer, binary, sublinear, C, the maximum number of features, and finally obtained the optimal parameter set.

max\_features=7000, tokenizer=text\_pipeline\_spacy, binary=True, sublinear\_tf=True

, C=1, Penalty=L2. I retrained the model with these parameters and plotted the confusion matrix and intrinsic evaluation metrics.

Q3c:

Calendar

Description automatically generated

Qr code

Description automatically generated

Accuracy is often used as a criterion to measure the quality of a classification model. It calculates the ratio of the total number of paired label predictions to the total number of labels that need to be predicted. Accuracy is not a good indicator if the distribution of this dataset is not balanced. From the label distribution map of Q1, the training data set is relatively balanced, which is why the prediction accuracy of the test set is very impressive.

From the data, it is clear that the two features added in Q3B are of significant help. The three used data are the subject of the post, and almost all important information is in these data, usually more Valid features and larger valid datasets tend to make the model's predictions more accurate, so theoretically, newly added features can optimize the model's accuracy. In addition to the title, author and content, there are several data in the dataset that have not been used for training, that is, the id and score. The id of each post is unique and has no use value. For the score, it is possible to add a weight to each data according to the score, and to give more weight to the data with higher scores when the model is trained. Because these high-scoring posts may be more representative.

Q4:

Text classification in natural language processing is a frequently researched direction, and both machine learning and deep learning using image convolutional networks have many academic outputs in this research area. However, more flexible, and robust graph convolutional neural networks such as non-grid convolutions or arbitrary graphs have not been fully explored. In this paper, Scholars build a corresponding text graph based on the number of co-occurrences of words in a window of a specified size and the relationship between these words and documents as a corpus, after training a text-image convolutional network model. Their purpose is to build a robust and more flexible TGCN.

Scholars have listed a variety of implementations of natural language processing, from the initial traditional text classification using bag of words features to filter news texts and detect spam, to later using deep learning word embedding and document embedding to do CNN, RNN, etc. Text classification, and later graph neural networks with rule-based grid structures. This type of graph neural network is favored by scholars, and methods such as semantic role labels and syntactic structure coding are added to the technology for machine translation.

Scholars have proposed a text classification method that has not been used on a large scale, called Text Graph Convolutional Networks. Different types of text corpora can be subdivided and branched into various node classification problems to deal with.

The main methods used in the experiment are one-hot encoding for vectorized text processing, and TGCN (Text Graph Convolutional Network) based on graph convolutional network. Their main experimental method is to compare their model with the experimental results of other mainstream models, such as the more commonly used TF-IDF + LR, CNN, LSTM and so on. The corpus they use is also very large and extensive to comprehensively test the feasibility of TGCN, and the datasets come from movie reviews, Reuters, the US National Library of Medicine, etc.

Although this model has good experimental results, it still lacks and develops opportunities. In the process of predicting unseen test documents, TGCN may not be able to generate embeddings quickly because there are no tags in the test documents. They also proposed solutions such as introducing induction and building fast GCN models. In addition, unsupervised learning frameworks for large batches of text input and attention mechanisms that can affect model classification performance are also identified as possible directions for development.