**Predict-Closed-Question-Stack Overflow**

Millions of programmers use Stack Overflow to get high quality answers to their programming questions every day. We take quality very seriously, and have evolved an effective culture of moderation to safe-guard it.

With more than six thousand new questions asked on Stack Overflow every weekday we're looking to add more sophisticated software solutions to our moderation toolbox.

Closing Questions

Currently about 6% of all new questions end up "closed". Questions can be closed as off topic, not constructive, not a real question, or too localized. More in depth descriptions of each reason can be found in the Stack Overflow FAQ. The exact duplicate close reason has been excluded from this contest, since it depends on previous questions.

Your goal is to build a classifier that predicts whether or not a question will be closed given the question as submitted, along with the reason that the question was closed. Additional data about the user at question creation time is also available.

**Dataset Link**

<https://www.kaggle.com/competitions/predict-closed-questions-on-stack-overflow/data>

**Dataset Description**

The training data contains data through July 31st UTC, and the public leaderboard data goes from August 1 UTC to August 14 UTC.

The train.csv file contains post text and associated metadata at the time of post creation which will serve as inputs to your solution. The state of the post as of July 31st is also included. It contains the following fields (not in this order):

Input

PostCreationDate

OwnerUserId

OwnerCreationDate

ReputationAtPostCreation

OwnerUndeletedAnswerCountAtPostTime

Title

BodyMarkdown

Tag1

Tag2

Tag3

Tag4

Tag5

Output

OpenStatus

Additional Data

PostId

PostClosedDate

The public leaderboard data contains all of the above fields, except for the target field OpenStatus and PostClosedDate.

The file train-sample.csv is a stratified sample of the training data: it contains every closed question and an equally-sized random sample of the open questions in the training data.

All questions will have a value in Tag1, but Tags 2 through 5 are optional.

To convert the user submitted Markdown found in BodyMarkdown to HTML if desired, our open source implementations in C# and Javascript may be useful.

Additional data can be found in "2012-07 Stack Overflow.7z", which contains an entire public data dump of Stack Overflow. Descriptions of the values can be found in the archive itself as well as on Meta Stack Overflow. This data will not be available as inputs, but may be useful in building your solution. As it is rather large (6GB) you may find it easier to download as a torrent, more details can be found in this forum post.

Using 8000 smaples only from data to can fit in the memory

**CODE IMPLEMENTATION**

**Steps involved**

**Step-1: Basic imports**

**Step-2:Exploratory Data Analysis**

**Step-3:Feature Engineering**

**Step-4:Generate Test & Train**

**Step-5:Training Model**

**Step-6:Model Evalution**

**Feature Engineering:**

* Drop nulls
* Extraxt important features as title, BodyMarkdown and OpenStatue
* Remove Puncatution
* Encode Labels using LabelEncoder
* Use TfidfVectorizer to extract features from data

**Models:**

1. **NAÏVE BAYES:**

Naive Bayes is a probabilistic machine learning algorithm that can be used for classification tasks, including predicting whether a question on StackOverflow is likely to be closed.

To use Naive Bayes for this task, you would need to start by collecting a labeled dataset of questions that have been closed and questions that have not been closed. The features of each question (such as the title, tags, body, etc.) would then be extracted and transformed into a format that can be used by the algorithm.

Next, the Naive Bayes algorithm would be trained on this dataset, learning the conditional probabilities of each feature given each class (closed or not closed). The "naive" assumption in Naive Bayes is that all features are independent of each other, which simplifies the computation of probabilities.

Once the algorithm has been trained, you can use it to predict whether a new question is likely to be closed based on its features. The algorithm calculates the probability of the question belonging to each class (closed or not closed) and assigns the class with the higher probability as the predicted class.

One of the advantages of Naive Bayes is that it is relatively fast and can work well with small amounts of training data. However, its assumption of feature independence may not always hold true in practice, which can lead to decreased performance in some cases.

**B)MLP Classifier:**

Multilayer Perceptron (MLP) is a type of neural network that can be used for classification tasks, including predicting whether a question on StackOverflow is likely to be closed.

To use MLP for this task, you would need to start by collecting a labeled dataset of questions that have been closed and questions that have not been closed. The features of each question (such as the title, tags, body, etc.) would then be extracted and transformed into a format that can be used by the algorithm.

Next, the MLP algorithm would be trained on this dataset using backpropagation to update the weights and biases of the network. The MLP consists of an input layer, one or more hidden layers, and an output layer. Each layer is made up of multiple nodes (also known as neurons), which perform a weighted sum of their inputs and apply a non-linear activation function to the result.

During training, the MLP learns to map the input features to the correct output class (closed or not closed) by adjusting the weights and biases to minimize the error between the predicted output and the true output.

Once the MLP has been trained, you can use it to predict whether a new question is likely to be closed based on its features. The algorithm calculates the output of the MLP for the new question, which is a probability distribution over the output classes. The predicted class is then assigned as the one with the highest probability.

One of the advantages of MLP is its ability to learn complex non-linear relationships between the input features and the output class. However, it can be computationally expensive and requires a larger amount of training data compared to some other algorithms like Naive Bayes.

**C)RANDOM FOREST:**

Random Forest is an ensemble learning algorithm that can be used for classification tasks, including predicting whether a question on StackOverflow is likely to be closed.

To use Random Forest for this task, you would need to start by collecting a labeled dataset of questions that have been closed and questions that have not been closed. The features of each question (such as the title, tags, body, etc.) would then be extracted and transformed into a format that can be used by the algorithm.

Next, the Random Forest algorithm would be trained on this dataset by constructing a collection of decision trees. Each decision tree is trained on a random subset of the training data and a random subset of the features. During training, the decision tree learns to split the data based on the values of the input features to maximize the separation between the two classes (closed and not closed).

Once the Random Forest has been trained, you can use it to predict whether a new question is likely to be closed based on its features. The algorithm calculates the output of each decision tree in the forest and assigns the class with the most votes as the predicted class.

One of the advantages of Random Forest is its ability to handle high-dimensional data and to capture complex non-linear relationships between the input features and the output class. It is also less prone to overfitting compared to a single decision tree. However, it can be computationally expensive and requires more memory than some other algorithms like Naive Bayes.

**D)ANN**

Artificial Neural Networks (ANNs) are a class of machine learning algorithms that can be used for classification tasks, including predicting whether a question on StackOverflow is likely to be closed.

To use ANNs for this task, you would start by collecting a labeled dataset of questions that have been closed and questions that have not been closed. The features of each question (such as the title, tags, body, etc.) would then be extracted and transformed into a format that can be used by the algorithm.

Next, you would design an ANN architecture that consists of an input layer, one or more hidden layers, and an output layer. Each layer consists of multiple nodes (also known as neurons), which perform a weighted sum of their inputs and apply a non-linear activation function to the result.

During training, the ANN learns to map the input features to the correct output class (closed or not closed) by adjusting the weights and biases of the network using backpropagation. Backpropagation is an optimization algorithm that computes the gradient of the loss function with respect to the weights and biases, and then updates them in the direction that reduces the loss.

Once the ANN has been trained, you can use it to predict whether a new question is likely to be closed based on its features. The algorithm calculates the output of the ANN for the new question, which is a probability distribution over the output classes. The predicted class is then assigned as the one with the highest probability.

One of the advantages of ANNs is their ability to learn complex non-linear relationships between the input features and the output class. However, they can be computationally expensive and require a larger amount of training data compared to some other algorithms like Naive Bayes. Additionally, designing the optimal architecture of the ANN can require a significant amount of trial and error.

**E)CNN:**

Convolutional Neural Networks (CNNs) are a type of neural network that are particularly well-suited for image and text classification tasks, including predicting whether a question on StackOverflow is likely to be closed.

To use CNNs for this task, you would start by collecting a labeled dataset of questions that have been closed and questions that have not been closed. The features of each question (such as the title, tags, body, etc.) would then be extracted and transformed into a format that can be used by the algorithm.

Next, you would design a CNN architecture that consists of multiple convolutional layers and pooling layers, followed by one or more fully connected layers and an output layer. Each convolutional layer applies a set of filters to the input features, which are then downsampled by the pooling layers to reduce the dimensionality of the data. The fully connected layers at the end of the network process the output of the convolutional layers and produce a final prediction.

During training, the CNN learns to extract relevant features from the input data and use them to predict the output class (closed or not closed). This is accomplished by adjusting the weights and biases of the network using backpropagation and stochastic gradient descent.

Once the CNN has been trained, you can use it to predict whether a new question is likely to be closed based on its features. The algorithm calculates the output of the CNN for the new question, which is a probability distribution over the output classes. The predicted class is then assigned as the one with the highest probability.

One of the advantages of CNNs is their ability to automatically learn features from the input data, which can reduce the need for feature engineering. Additionally, they can handle high-dimensional input data such as text and images. However, they can be computationally expensive and require a larger amount of training data compared to some other algorithms like Naive Bayes.

**F)LSTM:**

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is well-suited for sequence-to-sequence prediction tasks, including predicting whether a question on StackOverflow is likely to be closed.

To use LSTMs for this task, you would start by collecting a labeled dataset of questions that have been closed and questions that have not been closed. The features of each question (such as the title, tags, body, etc.) would then be extracted and transformed into a sequence of word embeddings, which capture the semantic meaning of the words in the text.

Next, you would design an LSTM architecture that consists of multiple LSTM layers and one or more fully connected layers and an output layer. Each LSTM layer has a set of memory cells and gates that control the flow of information through the network. During training, the LSTM learns to process the input sequence and generate an output sequence that predicts the output class (closed or not closed).

During training, the LSTM learns to process the input sequence and generate an output sequence that predicts the output class (closed or not closed). This is accomplished by adjusting the weights and biases of the network using backpropagation through time (BPTT), which computes the gradient of the loss function with respect to the weights and biases across all time steps.

Once the LSTM has been trained, you can use it to predict whether a new question is likely to be closed based on its features. The algorithm processes the input sequence through the LSTM and calculates the output of the final fully connected layer, which is a probability distribution over the output classes. The predicted class is then assigned as the one with the highest probability.

One of the advantages of LSTMs is their ability to model long-term dependencies in the input sequence, which can be useful for understanding the context of a text. However, they can be computationally expensive and require a larger amount of training data compared to some other algorithms like Naive Bayes. Additionally, designing the optimal architecture of the LSTM can require a significant amount of trial and error.

**MODELS EVALUTION:**

1)MLP Classifier

Accuracy = 62.5 %

2)NAIVE

Accuracy = 36.9 %

3)ANN

Accuracy = 59.1 %

4)CNN

Accuracy = 11.07%

5)RANDOM FOREST

Accuracy = 57.8%

6)LSTM

Accuracy =11.07%

**CONCLUSION:**

In this Notebook, I demonstrated various machine learning & deep learning algorithms, I have used MLP, NAIVE, ANN, RANDOM FOREST,CNN, LSTM mechanism for the above the dataset. You can further enhance the performance of your model

->Using other deep learning algorithms like GRU and BERT which are more suited for text classification problems.

->Using advanced word-embedding methods like GloVe and BER**T.**