Navodita Infotech – Machine Learning

Sentiment Analysis Project Documentation

Table of Contents

1. Introduction

The Sentiment Analysis project aims to classify text sentiments into three categories: Positive, Negative, and Neutral. This document provides a detailed overview of the project, including data processing, model development, and training.

2. Data Overview

The project uses the Sentiment Analysis Dataset from Kaggle, consisting of tweets labeled with sentiments. The dataset contains various features, including 'text', 'sentiment', and additional information about the tweet.

Sentiment Distribution

- Neutral: 11117 instances
- Positive: 8582 instances
- Negative: 7781 instances

3. Data Preprocessing

Data preprocessing involves cleaning and transforming raw text data into a format suitable for model training. Key steps include:

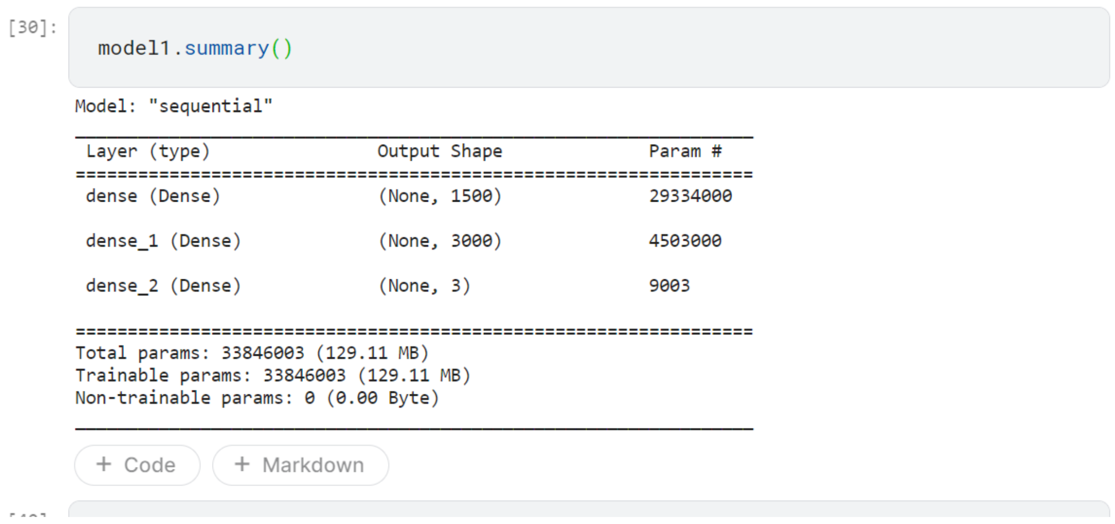
- Handling missing values
- Text cleaning (removing special characters, stemming)
- Text vectorization using CountVectorizer

4. Model Architecture

The neural network model is implemented using TensorFlow and Keras. The architecture consists of:

- Input layer with the number of features matching the vectorized text length
- Two dense hidden layers with ReLU activation functions
- Output layer with three units (softmax activation) representing the three sentiment classes

Model Summary:

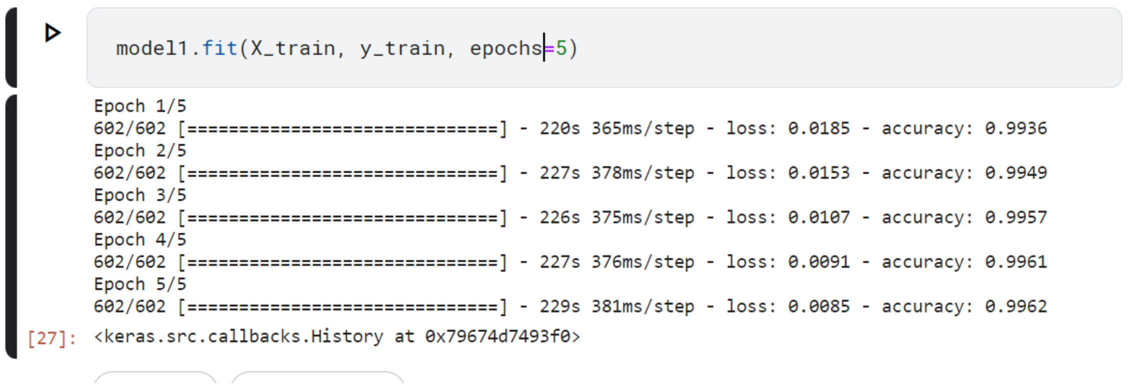


```
[30]: model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1500)              29334000

 dense_1 (Dense)             (None, 3000)              4503000

 dense_2 (Dense)             (None, 3)                 9003

=================================================================
Total params: 33846003 (129.11 MB)
Trainable params: 33846003 (129.11 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

5. Model Training

The model is trained using the training data split into training and validation sets. The compilation includes the Adam optimizer, categorical crossentropy loss, and accuracy as the evaluation metric. Class weights are used to handle class imbalance.

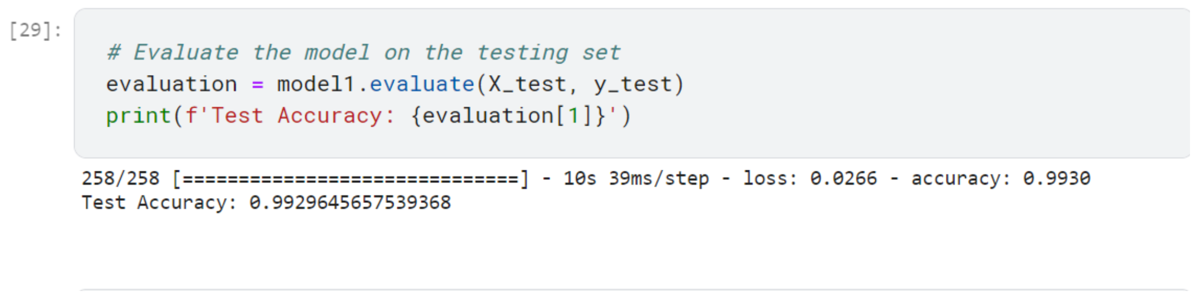


```
model1.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
602/602 [==============================] - 220s 365ms/step - loss: 0.0185 - accuracy: 0.9936
Epoch 2/5
602/602 [==============================] - 227s 378ms/step - loss: 0.0153 - accuracy: 0.9949
Epoch 3/5
602/602 [==============================] - 226s 375ms/step - loss: 0.0107 - accuracy: 0.9957
Epoch 4/5
602/602 [==============================] - 227s 376ms/step - loss: 0.0091 - accuracy: 0.9961
Epoch 5/5
602/602 [==============================] - 229s 381ms/step - loss: 0.0085 - accuracy: 0.9962
[27]: <keras.src.callbacks.History at 0x79674d7493f0>
```

6. Evaluation Metrics

Evaluation metrics are essential to assess model performance. Common metrics include accuracy, precision, recall, and F1-score. The model is evaluated on a separate test set.



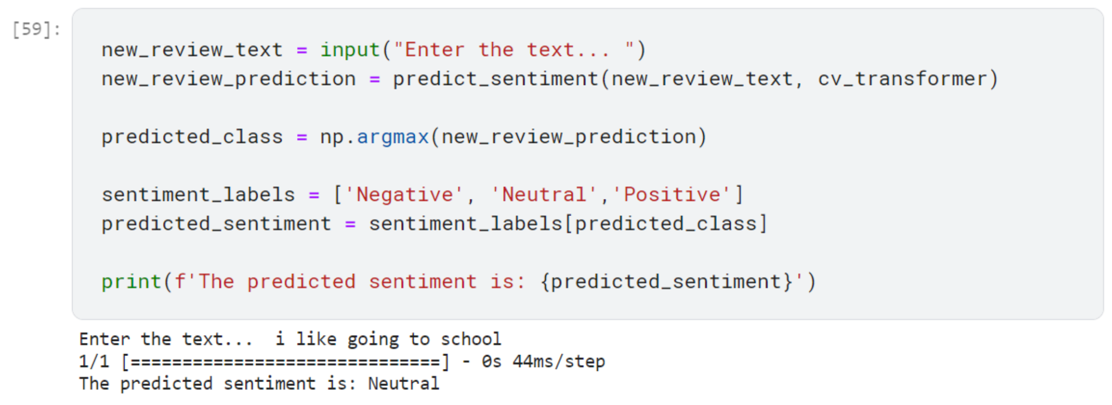
```
[29]:
# Evaluate the model on the testing set
evaluation = model1.evaluate(X_test, y_test)
print(f'Test Accuracy: {evaluation[1]}')
```

```
258/258 [==============================] - 10s 39ms/step - loss: 0.0266 - accuracy: 0.9930
Test Accuracy: 0.9929645657539368
```

7. Usage Example

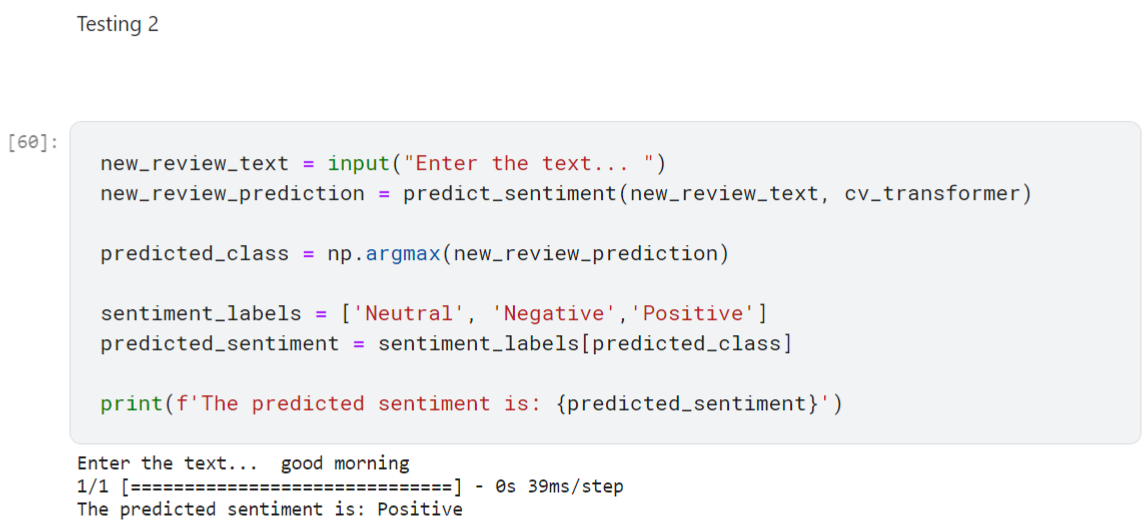
An example is provided for using the trained model to predict the sentiment of new text input.

Testing 1



```
[59]:
new_review_text = input("Enter the text... ")
new_review_prediction = predict_sentiment(new_review_text, cv_transformer)

predicted_class = np.argmax(new_review_prediction)

sentiment_labels = ['Negative', 'Neutral','Positive']
predicted_sentiment = sentiment_labels[predicted_class]

print(f'The predicted sentiment is: {predicted_sentiment}')
```

```
Enter the text...  i like going to school
1/1 [==============================] - 0s 44ms/step
The predicted sentiment is: Neutral
```



Testing 2

```
[60]:
new_review_text = input("Enter the text... ")
new_review_prediction = predict_sentiment(new_review_text, cv_transformer)

predicted_class = np.argmax(new_review_prediction)

sentiment_labels = ['Neutral', 'Negative','Positive']
predicted_sentiment = sentiment_labels[predicted_class]

print(f'The predicted sentiment is: {predicted_sentiment}')
```

```
Enter the text...  good morning
1/1 [==============================] - 0s 39ms/step
The predicted sentiment is: Positive
```

8. Conclusion

The Sentiment Analysis project demonstrates the process of building a neural network model for classifying text sentiments. Continuous improvement, such as fine-tuning the model, exploring advanced architectures, or incorporating additional features, can enhance the model's performance.

Links:

https://drive.google.com/drive/folders/125Lu95EV8VBQTf4HDulPPnZI6EDiZVf8?usp=sharing

https://www.kaggle.com/code/harshsangrulkar/sentiment-analysis/notebook

https://github.com/HarshSangrulkar/Sentiment-Analysis