CS 464 Group 10 Final Project Presentation

Sentiment Analysis on Amazon Product Reviews

Mehmet Berk Türkçapar 21902570
Emir Türkölmez 21901626
M. Haider Akbar 21901132
Afşın Arpad Demirkasımoğlu 21903470
Övgüm Can Sezen 21902418

Problem Definition

- Sentiment analysis is an important area of machine learning, as it enables businesses to understand how customers feel about their products and services.
- Amazon.com, one of the largest online retailers in the world, has millions of customer reviews for millions of products.
- By analyzing these reviews, businesses can gain insights into customer preferences, identify areas for improvement, and make data-driven decisions to enhance the customer experience.
- By analyzing customer feedback, businesses can improve their products, services, and customer engagement.
- So, in this project we aim to automate classification of product reviews using machine learning and deep learning algorithms.

Description of Data

- The dataset we have selected is called Amazon Product Reviews and is available on Kaggle [1].
- The dataset contains over 568.000 Amazon product reviews from a variety of categories, including electronics, clothing, home appliances, and more. It is in CSV form and contains a lot of information for each review.
- These information include review text, the score, review title, product category, review date and more. For the purposes of this project the review text and the score given (out of 5) will be important.
- We are planning to categorize each review based on the scores. Reviews with a score 5 or 4 will be considered as positive, with a score 3 will be considered neutral and with a score 2 or 1 will be considered as negative.

Description of Data (Sample)

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...

Project Plan

- Initially, the dataset is needed to be labeled based on their scores. Also, we will be dividing our labeled data into three groups for training, evaluation and testing our models.
- Next, in terms of data processing, the review text may require some preprocessing such as tokenization, stemming and removal of stop words.
- For embedding:
  - word2vec
- Two machine learning algorithms:
  - Gaussian Naive Bayes
  - Logistic Regression
- Deep learning model:
  - Fully connected layer

Data Preprocessing

Label the data according to ‘Score’

- Score > 3 => Positive
- Score = 3 => Neutral
- Score < 3 => Negative

NLP (Natural Language Processing)

- Stop Word Removal (Removing words like “a”, “an”, “the”)
- Punctuation Removal
- Converting all text to lowercase
- Tokenize the data
- Apply Stemming

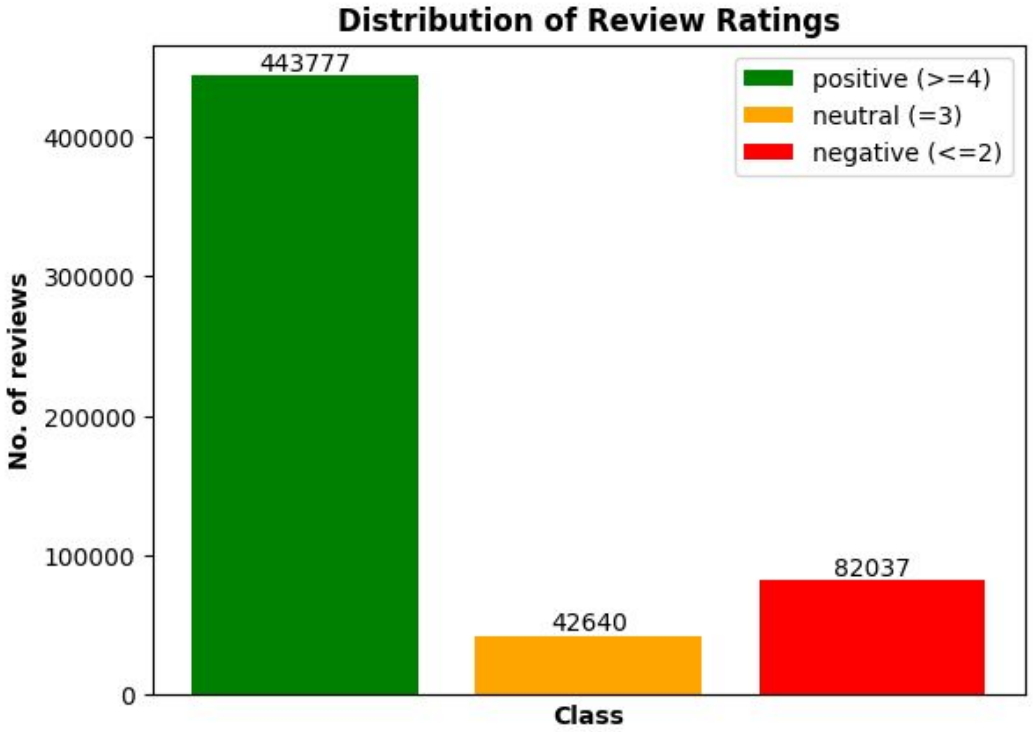
Data Preprocessing (Labeled Data)

	Score	Text	Label
0	5	I have bought several of the Vitality canned d...	positive
1	1	Product arrived labeled as Jumbo Salted Peanut...	negative
2	4	This is a confection that has been around a fe...	positive
3	2	If you are looking for the secret ingredient i...	negative
4	5	Great taffy at a great price. There was a wid...	positive

Analysis of the Dataset (Histogram)

- In total there are 568454 entries in the dataset.
- An imbalanced dataset.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$



Data Preprocessing

```
stemmer = PorterStemmer()

# Remove punctuation
df['Text'] = df['Text'].apply(lambda x: x.translate(str.maketrans('', '', string.punctuation)))

# Convert all text to lowercase
df['Text'] = df['Text'].apply(lambda x: x.lower())

# Tokenize the text data
df['Text'] = df['Text'].apply(lambda x: word_tokenize(x))

# Remove stop words
stop_words = set(stopwords.words('english'))
df['Text'] = df['Text'].apply(lambda x: [word for word in x if word not in stop_words])

# Stemming
df['Text'] = df['Text'].apply(lambda x: [stemmer.stem(word) for word in x])
```

Data Preprocessing (Without Stemming)

	Score	Text	Label
0	5	['bought', 'several', 'vitality', 'canned', 'd...	positive
1	1	['product', 'arrived', 'labeled', 'jumbo', 'sa...	negative
2	4	['confection', 'around', 'centuries', 'light',...	positive
3	2	['looking', 'secret', 'ingredient', 'robitussi...	negative
4	5	['great', 'taffy', 'great', 'price', 'wide', '...	positive

Data Preprocessing (With Stemming)

	Score	Text	Label
0	5	['bought', 'sever', 'vital', 'can', 'dog', 'fo...	positive
1	1	['product', 'arriv', 'label', 'jumbo', 'salt',...	negative
2	4	['confect', 'around', 'centuri', 'light', 'pil...	positive
3	2	['look', 'secret', 'ingredi', 'robitussin', 'b...	negative
4	5	['great', 'taffi', 'great', 'price', 'wide', '...	positive

Dataset Split

Dataset Split:

- 80% for training
  - 10% of training for validation
  - 90% for training
- 20% for testing

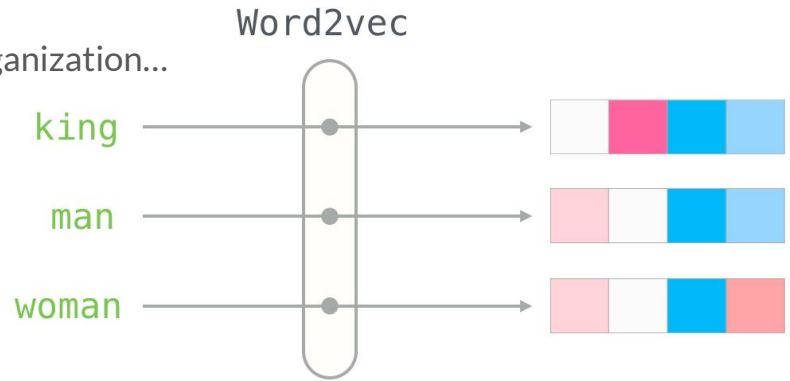
```
# Split into train and test sets (80-20 split)
train_data, test_data, train_labels, test_labels = train_test_split(df_stem['Text'], df_stem['Label'], test_size=0.2, random_state=42)

# Split train data into train and validation sets (90-10 split)
train_data, val_data, train_labels, val_labels = train_test_split(train_data, train_labels, test_size=0.1, random_state=42)
```

Word2vec

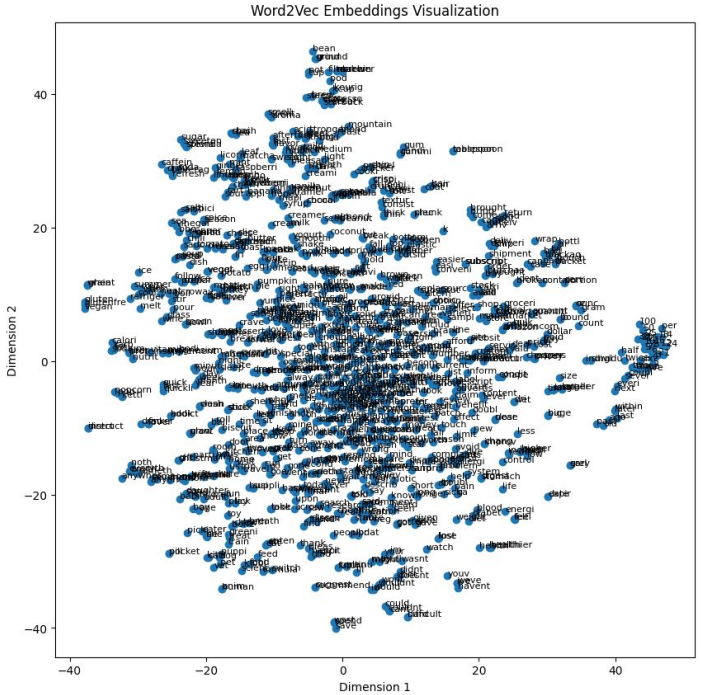
- Word2vec is a technique for replacing words to numbers
- Uses Neural Network to assign numbers to words
- Not context related, uses neighbouring words to determine context

Why word2vec?

- simple, fast to train and powerful
- semantic relatedness, synonym detection,concept organization…



Word2vec Embeddings




```
word_embeddings.most_similar(positive="great")
```

```
[('fantast', 0.8169080018997192),
 ('excel', 0.80169278383255),
 ('good', 0.7968287467956543),
 ('awesom', 0.7892809510231018),
 ('terrif', 0.7513600587844849),
 ('wonder', 0.7494795322418213),
 ('nice', 0.7204874753952026),
 ('perfect', 0.6945949196815491),
 ('fabul', 0.6798199415206909),
 ('amaz', 0.6281005144119263)]
```

Gaussian Naive Bayes

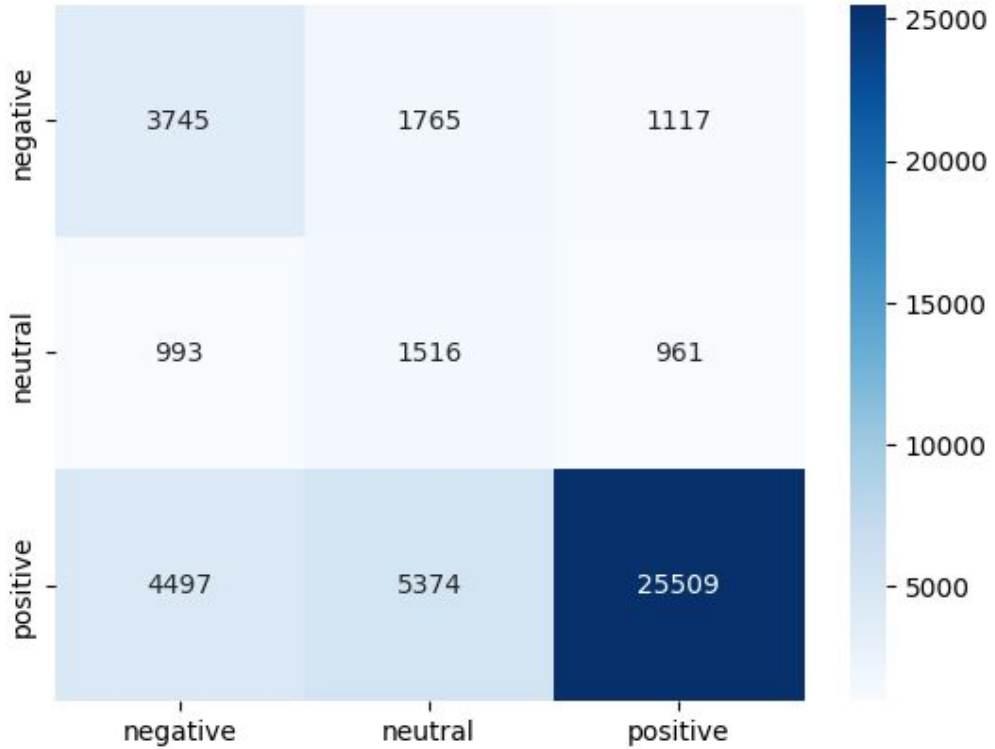
Why Gaussian Naive Bayes?

- Continuous data
- Handling continuous variables
- Multinomial Naive Bayes limitations

Gaussian Naive Bayes

Accuracy 67.7%

F1 Score 0.7182



Logistic Regression

Why logistic regression?

- Computationally efficient ⇒ Simple model AND gradient based optimization AND no feature interaction assumption
- Well established method for classification tasks with a large variety of support

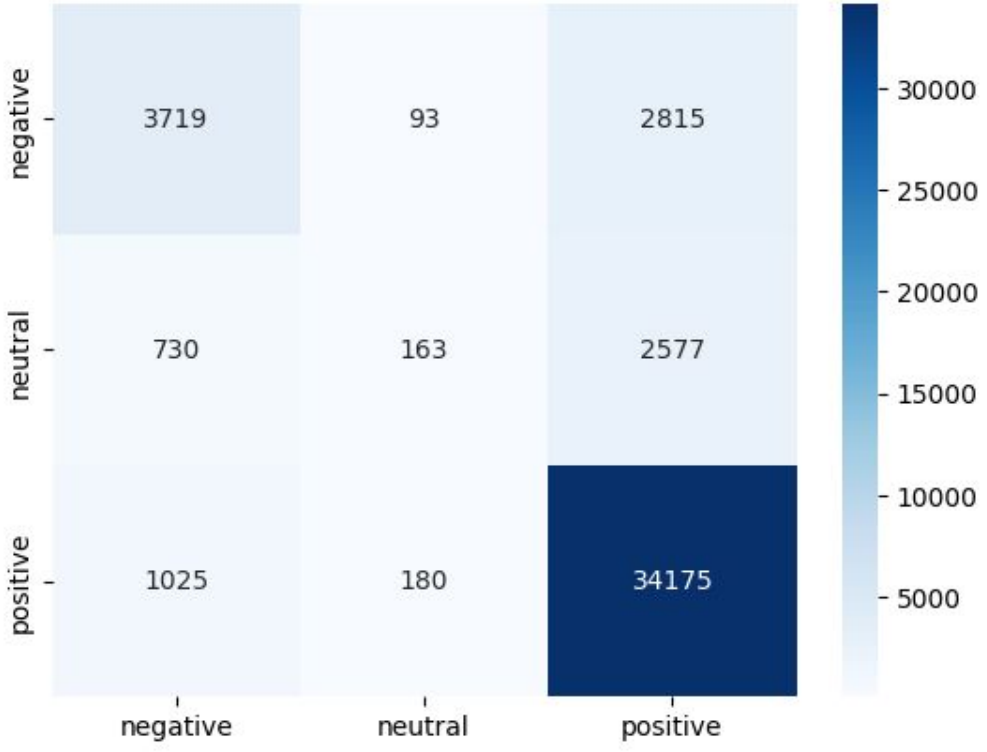
Why scikit-learn for logistic regression?

- Computationally efficient ⇒ Utilizes efficient numerical algorithms AND supports parallel processing
- Well integrated with the scikit-learn ecosystem, simplifies the evaluation and model validation processes too

Logistic Regression

Accuracy 0.8368

F1 Score 0.8054



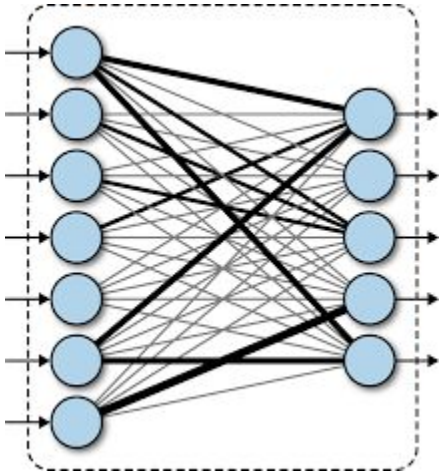
Fully Connected Layer

Optimizer: Adam

Loss Function: Cross Entropy

Different configurations:

- Activation Function
  - ReLU
  - Sigmoid
  - Tanh
- Learning Rate
  - 0.01
  - 0.001
  - 0.0001
- Hidden Layer Size
  - 128
  - 256
  - 512
- # of hidden layers



Fully Connected Layer (Activation Function)

- 25 epochs
- Hidden Layer Size = 128
- Batch Size = 32
- Learning Rate = 0.001

Activation Function	**F1 Score**
ReLU	0.8402
Sigmoid	0.8387
Tanh	0.8286

Fully Connected Layer (Learning Rate)

- 25 epochs
- Hidden Layer Size = 128
- Batch Size = 32
- Activation Function = ReLU

Learning Rate	F1 Score
0.0001	0.8344
0.001	0.8402
0.01	0.7942

Fully Connected Layer (Hidden Layer Size)

- 25 epochs
- Learning Rate = 0.001
- Single Layer used in each trial
- Batch Size = 32
- Activation Function = ReLU

Hidden Layer Size (number of neurons)	F1 Score
64	0.8304
128	0.8402
256	0.8482

Fully Connected Layer (Number Of Hidden Layers)

- 25 epochs
- Learning Rate = 0.001

- Batch Size = 32
- Activation Function = ReLU
- Hidden Layer Size = 256

Number of Hidden Layers	F1 Score
1	0.8482
2	0.8635
3	0.8639

Performances with Test Data

Model	Accuracy	F1 Score
Gaussian Naive Bayes	0.6772	0.7196
Logistic Regression	0.8399	0.8095
Fully Connected Layer	0.8755	0.8633

References

[1] A. Rumi, “Amazon product reviews,” *Kaggle*, 11-Jul-2021. [Online]. Available: https://www.kaggle.com/datasets/arhamrumi/amazon-product-reviews. [Accessed: 13-Mar-2023].