Homework - Datathinking

Shurakov Nikolai
Nikolai.Shurakov@ut.ee
student
University of Tartu

Coauthor ChatGPT
https://chat.openai.com/
LLM
Microsoft

Abstract

In this homework, we cleaned and analyzed a dataset of a Zulip chat using different machine-learning tools, such as linear regression, logistic regression, and embeddings. We visualized the results of these analyses using plots and tried to interpret each visualization. Finally, we learned how to create a report in Overleaf and how to include PDF figures with descriptions of each analysis and relevant equations.

1 Introduction

I tried to start doing this homework several times. I tried to use different IDEs: Thonny, VScode on my computer and Github Codespace. Both had its advantages and disadvantages, but I ended up with Github Codespace and copied the datathinking.org codespace project. The first two tasks were solved in the lecture on 30/03/2023, so I just repeated the steps from the part of the recording that I missed. I will title each section with what I was supposed to do and then add the outcome I got with some comments.

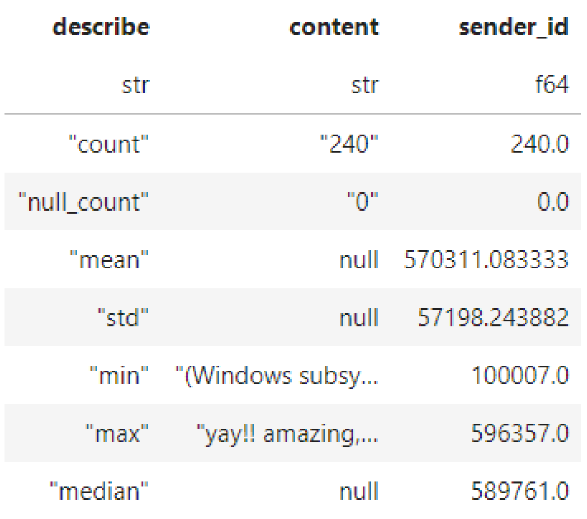
2 Clean Data Thinking Zulip chat data and compute summary statistics of the dataset

The genarated code imports the **polars** package and **json** module. It opens a JSON file ”*messages-000001.json*” in read mode using the **with** statement. It then loads the contents of the file into a Python dictionary called **data**. Two empty lists are created, *all_messages* and *all_senders*.

Next, a for loop is used to iterate over each message in the *zerver_message* column in *data dictionary*. The content of each message is extracted and appended to the *all_messages list*, while the sender is appended to the *all_senders list*.

A new dictionary **data** is created with two keys, *content* and *sender_id*, corresponding to the *all_messages* and *all_senders* lists, respectively.

Finally, a **polars DataFrame** is created from the data dictionary and summary statistics are computed using the **describe()** method.



describe	content	sender_id
str	str	f64
"count"	"240"	240.0
"null_count"	"0"	0.0
"mean"	null	570311.083333
"std"	null	57198.243882
"min"	"(Windows subsy...	100007.0
"max"	"yay!! amazing,...	596357.0
"median"	null	589761.0

Figure 1: Summary statistics of the dataset

3 Linear and logistic regressions

The next part of my code also uses a Polars DataFrame from the data dictionary. The text data is then preprocessed, and the **Word2Vec** model is used to convert the text data into numerical format. The model is used to make a list of embeddings for the first word in each message. The data is split into training and test sets, and logistic regression and linear regression models are trained on the training data. The logistic regression model is used to predict the sender of the messages in the test set and accuracy is computed. Similarly, linear regression model is used to predict the sender of the messages in the test set, and mean squared error is computed.

Logistic Regression Accuracy: 0.3125 Linear Regression Mean Squared Error: 46015520188.596924

Finally, the embeddings are visualized using t-SNE and scatter plot is used to represent embeddings in two dimensions.

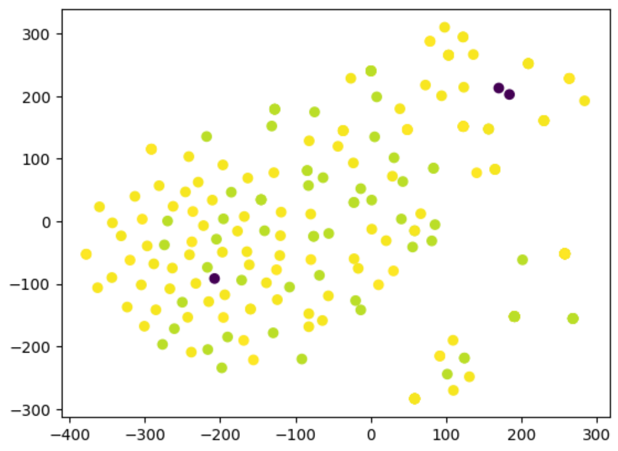


Figure 2: Comparison of Logistic and Linear Regression Models for Message Classification

4 Model for Message Senders

The code processes a JSON file containing chat messages and their corresponding sender IDs. The primary goal of this pipeline is to train a model that can predict the sender ID based on the content of the chat message using word embeddings.

To accomplish this, the text data is first preprocessed and tokenized, and then converted into numerical format using the **Word2Vec** algorithm. After obtaining the embeddings for the chat messages, a linear regression model is trained on a subset of the data and evaluated on a separate test set.

Once the model is trained and evaluated, the predicted sender IDs are compared against the actual sender IDs using a scatter plot. This plot helps visualize how well the linear regression model performs at predicting the sender IDs for the test set (not so well). The plot also includes a line for perfect correlation and a legend with appropriate labels, making it easier to interpret and analyze the results.

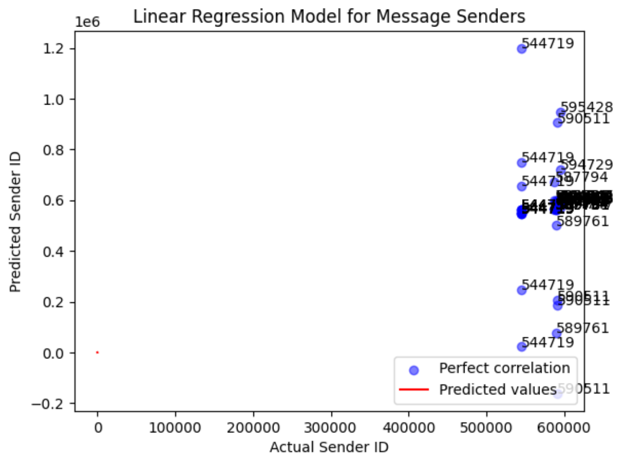


Figure 3: Linear Regression Model for Message Senders

5 Linear and Logical regressions compared

I was not satisfied with any of the previously generated plots. The data look noisy. Although, this is expected (there is no real correlation between one's Zulip chat ID and messages first words), I updated the code to get another plot. Tis time the logistic regression model is evaluated for accuracy on the test data and the linear regression model is evaluated for mean squared error. The performance of the two models is compared, and the embeddings are visualized using t-SNE. Finally, a scatter plot is generated with the logistic regression results using t-SNE.

Logistic Regression Accuracy: 0.3125

Linear Regression Mean Squared Error: 46407887333.514915

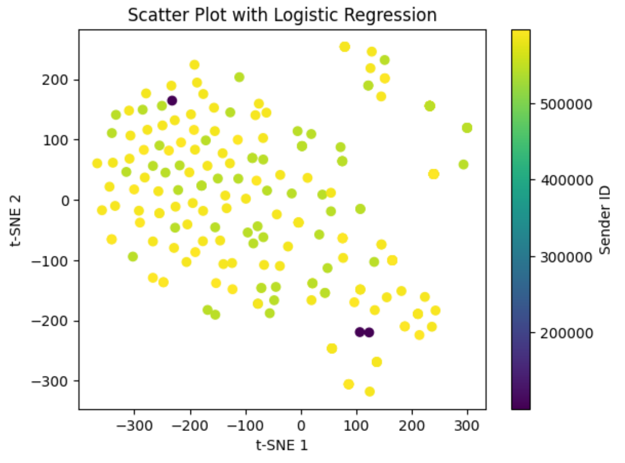


Figure 4: Scatter Plot with Logistic Regression

6 Plot with Embeddings

The embeddings are then used to visualize the messages in two dimensions using t-SNE, which is a technique for reducing high-dimensional data to a lower dimension while preserving the structure of the data. The resulting scatter plot shows the distribution of the messages in the two-dimensional space, with each point representing a message and the color indicating the sender of the message. The plot provides insight into the relationships between the messages and how they are distributed in the embedding space.

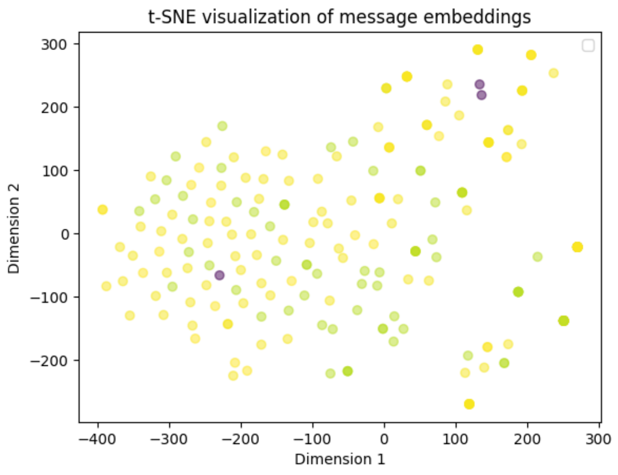


Figure 5: t-SNE visualization of message embeddings

7 Equations

Linear Regression:

The equation for simple linear regression is:

$$y = \beta_0 + \beta_1 x + \epsilon$$

where y is the dependent variable, x is the independent variable, β_0 is the intercept, β_1 is the slope, and ϵ is the error term.

In the code provided, the Linear Regression model is implemented using scikit-learn's LinearRegression class, which uses Ordinary Least Squares (OLS) method to fit a linear regression model.

Logistic Regression:

The equation for logistic regression is:

$$p(y = 1|x) = 1/(1 + exp(-z))$$

where p(y=1|x) is the probability of the dependent variable (y) being 1 given the independent variable (x), exp is the exponential function, and z is the weighted sum of the inputs.

In the code provided, the Logistic Regression model is implemented using scikit-learn's LogisticRegression class, which uses maximum likelihood estimation to fit a logistic regression model.

Embedding:

Word2Vec is a neural network-based algorithm that learns vector representations of words called word embeddings. The algorithm is based on the distributional hypothesis, which states that words that appear in similar contexts are likely to have similar meanings.

The equation for Word2Vec is based on the skip-gram model:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\ne 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

where J(⊠) is the objective function, T is the total number of words in the corpus, m is the size of the context window, wt is the center word, and wt+j is the context word.

In the code provided, the Word2Vec model is implemented using Gensim's Word2Vec class, which trains the word embeddings by maximizing the likelihood of the skip-gram model.

8 Conclusion

In conclusion, this homework involved cleaning and analyzing a dataset of a Zulip chat using different machine-learning tools such as linear regression, logistic regression, and embeddings. We visualized the results of these analyses using plots and tried to interpret each visualization. The data was processed using Polars and Word2Vec, and summary statistics were computed. The accuracy of the logistic regression model for predicting the sender of the messages was 0.3125, and the mean squared error of the linear regression model was 46015520188.596924. We also trained a linear regression model to predict the sender of the messages based on the content of the message, but the performance was not very good. Finally, we compared the performance of the logistic regression model and the linear regression model and observed that the logistic regression model performed better. Overall, this homework helped us to learn how to use different machine-learning tools and how to visualize the results of these analyses. Unfortunately, I didn't manage to follow Ismael's advice and didn't use vega or vega light, but I may try it later for some simpler tasks.

References

- json package: https://docs.python.org/3/library/json.html
- numpy package: https://numpy.org/doc/
- pandas package: https://pandas.pydata.org/docs/
- polars package: https://pola-rs.github.io/polars-book/
- scikit-learn package: https://scikit-learn.org/stable/
- LogisticRegression class: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- LinearRegression class: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- accuracy_score function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy.html
- mean_squared_error function: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html
- Word2Vec class: https://radimrehurek.com/gensim/models/word2vec.html
- TSNE class: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
- matplotlib.pyplot package: https://matplotlib.org/stable/api/pyplot_summary.html