




# SPECIAL TOPICS PROJECT



Makade, Mishi

## Data Preparation

We began by bringing our data into Python, which is a computer language, using a helpful tool called Pandas. This was really important because our main goal was to use the words and ratings to understand people's feelings. To make things simpler, we carefully looked at our data and removed some parts that didn't help with our study. We kept only the ratings and the words, which were the most important.

First, we checked if there were any missing pieces of information in our data. If we found any, we threw them away because they didn't tell us anything useful. This made our data better.

Then, we turned our attention to the words in our data. The first thing we did was to find and take out any special symbols that might be in the words. We used a tool called Regex for this job because it's good at finding and fixing these special symbols.

At the same time, we made sure all the letters in the words were the same by changing the big letters to small ones. We also removed common words like "the," "on," "in," and others that don't tell us much. To help us with this, we used something called the NLTK library, which is like a helper for understanding words in a computer.

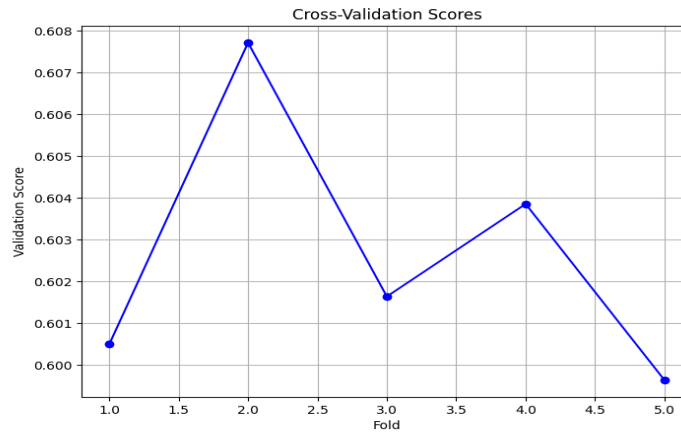
We also got rid of any strange symbols in the words using Regex. Lastly, we took out any numbers from the words so we could focus only on the words themselves. This way, we had a clean and clear set of words to study.

Then the data was split into features and target variable. The features are the text, and the target variable is the ratings. Before training the model, the data was split into train and test set which is a very important step. The data was split such the train set is 70% and the test set is 30%.

The text was pre-processed further. Since ML models require only numerical data, the text was vectorized to make it numerical.

## Model Training and Evaluation

For the modelling phase, a Logistic Regression model was used. The Logistic Regression model was configured such that the penalty was l2. The C was set to be 1 with the aim of preventing the model from overfitting. The model was fit on the training data where the text was the features, and the ratings were the labels. The model was cross-validated. The cross-validation was done with 5 folds, and the accuracy score for each fold was recorded. The accuracy scores for the cross-validation folds are shown in the plot below.



**Figure 1.** Validation Scores

Then the model was trained on the whole training data. The trained model and the vectorizer used to vectorise the texts are saved using the pickle package. To evaluate the performance of the model the model was used to make predictions on new unseen data which is the test dataset. The performance was evaluated using accuracy and a confusion matrix. The confusion matrix is shown below, and it shows most of the instances were correctly predicted, especially for the 2<sup>nd</sup> class.

True Labels	0	1	2	3	4
0	2405	3571	0	0	0
1	2422	3601	0	0	0
2	0	0	6005	0	0
3	0	0	0	3554	2434
4	0	0	0	3594	2414
	0	1	2	3	4
	Predicted Labels				

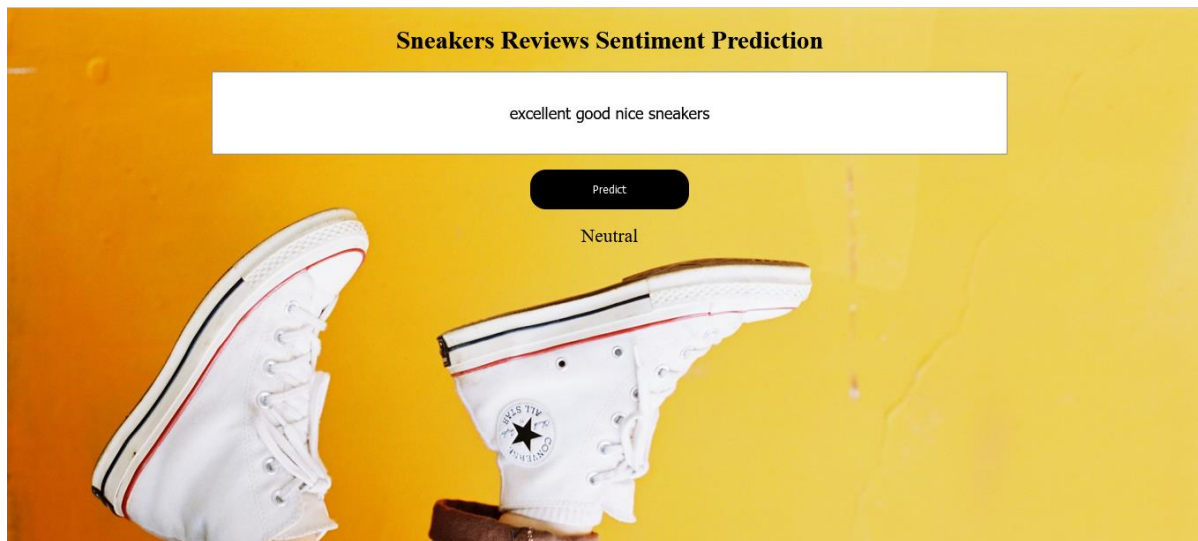
**Figure 2.** Confusion matrix

The accuracy obtained for the test dataset is 60%. After obtaining the predictions the rating predicted is changed to a sentiment. This is done such that the ratings 0 or 1 are interpreted as a negative sentiment. And a rating of 3 is interpreted as a neutral sentiment. Then a rating of 4 or 5 is interpreted as a positive sentiment. After a prediction is made by the model, the prediction is turned to a sentiment using aforementioned criteria.

## API

### Introduction to the API

The API created is used to take user text reviews and produce a sentiment of the given text. A front-end UI was created which takes the text and produces the sentiment. The UI is shown in the picture below. A user has an input box where they can enter the text they want to predict, then a button is also provided which allows the user to obtain the sentiment prediction for the text they entered. A prediction is shown below the button.



**Figure 3.** Front-end UI

### Endpoint and request information

The back end of the API consists of two endpoints.

1. Admin endpoint: The first endpoint is the admin endpoint. This endpoint allows one to interact directly with the data of the API without writing code. You can create, delete, and update the data stored in the database models.
2. Predict endpoint: Then the API also has the predict endpoint. The predict endpoint is where a user can enter their data and see the sentiment prediction. It takes input data, puts it through the model. Then a prediction is given and shown. This endpoint supports a POST request which is specified using the decorators. It is linked to the frontend part. The front-end part takes the user data, sends it to this endpoint, and the endpoint gives a prediction for the data. Then the prediction is sent and displayed on the user interface.

### Processing of user input

Serializers play an important role in the processing of input data, as demonstrated in this assignment. Their usage involved the following key functions:

1. Deserialization: Serializers were employed to deserialize data. In essence, they facilitated the conversion of data input by the user through the API/UI, which was formatted in JSON, into a Python object that Python could effectively process.
2. Serialization: Serializers were also utilized for data serialization. As per the API's requirements, data from the UI needed to be sent to the model for predictions. However, the predictions generated were in the form of Python objects, which were incompatible with the API and UI. Therefore, serializers played a crucial role in transforming these predictions from Python objects into a JSON format that could be consumed by the API and website.