# Fake Review Detection

This involves automatically identifying and flagging fake or misleading reviews posted online. These reviews may be fabricated by individuals or generated by bots with the intention to manipulate consumer opinions.

**Dataset:**



**Features:**

[DOC\_ID, RATING, VERIFIED\_PURCHASE, PRODUCT\_CATEGORY, PRODUCT\_ID, PRODUCT\_TITLE, REVIEW\_TITLE, REVIEW\_TEXT]

**Target:**

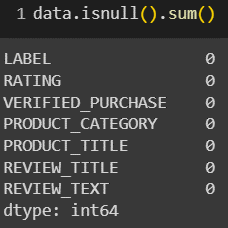
LABEL:

\_\_label1\_\_ -fake

\_\_label2\_\_ - real

## Preprocessing Data Set

* Removed unnecessary features like IDs.
* Handling missing values: There are no missing values.



* Class imbalance: There is no class imbalance.

A chart of a class distribution

Description automatically generated

* Separated the dataset into 2 parts

1. **data\_meta**: consists of features that represent the metadata of review

[RATING VERIFIED\_PURCHASE PRODUCT\_CATEGORY]

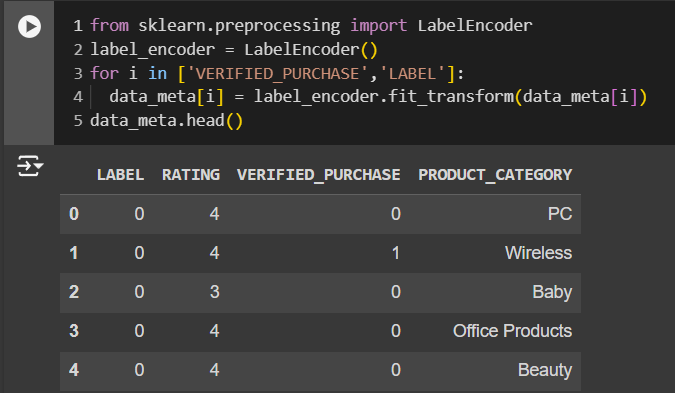
1. **data\_text**: consists of features that are actually review text

[PRODUCT\_TITLE REVIEW\_TITLE REVIEW\_TEXT]

### Preprocessing data\_meta:

1. **Label Encoding:**

Rating, Verified\_purchase features have inherent order among the categories. So, used LabelEncoder to encode them and also target LABEL into numerical values each value represents each category respectively.



1. **One Hot Encoding for Product\_category:**

Since there is no inherent order among values of product category one hot encoding is performed.

1. **Feature Construction:**

Constructed features like

'review\_length', 'title\_length', 'num\_words', 'num\_sentences',

'avg\_word\_length', 'num\_unique\_words', 'num\_stop\_words',

'punctuation\_count', 'num\_capitalized\_words', 'sentiment\_score',

'review\_length\_x\_rating', 'verified\_purchase\_x\_rating'.

1. **Scaling using Standard Scaler**

### PreProcessing data\_text:

Conversion all letters to lower case, removal of stop words, removal of punctuation.

## Application of ML Models:

1. Tokenized the data\_text using Tokenizer of tensorflow and concatenated both data\_meta and data\_text to apply following ml models

|  |  |
| --- | --- |
| Model | Accuracy |
| Logistic Regression | 0.7892857142857143 |
| Random Forest | 0.8161904761904762 |
| Gradient Boost | 0.8102380952380952 |
| LSTM | 0.8515, val\_accuracy: 0.7621, loss: 0.3236, val\_loss: 0.4933 |

1. Instead of combining giving PRODUCT\_TITLE, REVIEW\_TITLE, REVIEW\_TEXT to a single LSTM created a LSTM layer for each and a network of Dense layers is used for training data\_meta. This improved accuracy further to 0.8806 and decreased loss to 0.3222 and validation accuracy to 0.8260 and validation loss to 0.4862.
2. Then Implemented BERT embeddings instead of LSTM and other boosting techniques.

|  |  |
| --- | --- |
| Model | Accuracy Metric |
| BERT + Adaboost | 0.8088 |
| BERT + XGBOOST | 0.8283 |

## Hyper parameter Tuning:

GridSearch CV for searching best parameters for XGBoost model this lead to increase the accuracy of classification model to 0.8607.

## Gradio for Demo UI:

Employed Gradio package to deploy ui for testing the model.

A screenshot of a computer

Description automatically generated

## Future work:

Creation of web application for actual User Interface.

Frontend: React JS.

Backend: FLASK.