University of Mumbai

Stance Detection from Text

Submitted in partial fulfillment of requirements

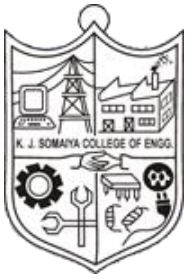
For the degree of

Bachelors in Technology

by

Name: Nishavak Naik
Roll No: 1814040
Name: Atharva Kitkaru
Roll No: 1814033
Name: Krisha Mehta
Roll No: 1814108

Guide: Dr. Irfan A Siddavatam



Department of Information Technology
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch 2018 - 2022

K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Certificate

This is to certify that the dissertation report entitled **Stance Detection from Text** is bona fide record of the dissertation work done by **Nishavak Naik, Atharva Kitkaru and Krisha Mehta** in the year 2021-22 under the guidance of **Dr. Irfan A Siddavatam** of Department of Information Technology in partial fulfillment of requirement for the Bachelors in Technology degree in Information Technology of University of Mumbai.

_____ _____

Guide Head of the Department

_____

Principal

Date:

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate of Approval of Examiners

We certify that this dissertation report entitled **Stance Detection from Text** is bona fide record of project work done by Nishavak Naik, Atharva Kitkaru and Krisha Mehta.

This project is approved for the award of Bachelors in Technology Degree in Information Technology of University of Mumbai.

_____

Internal Examiner

_____

External Examiner

Date:

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

DECLARATION

We declare that this written thesis submission represents the work done based on our and / or others' ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in our submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

Signature of the Student **Roll No.**	**Signature of the Student** **Roll No.**
Signature of the Student **Roll No.**	**Signature of the Student** **Roll No.**

Date:

Place: Mumbai-77

Dedicated to

……..……

Abstract

With the advent of social media, exchange of ideas and opinions on a spectrum of topics has been carried out at an unparalleled rate. Understanding people's opinions on this array of topics, especially the sensitive/controversial ones, holds the key to performing any social analytical studies. This is where stance detection comes into picture. It is the task of automatically determining from text whether the author of the text is in favor of, against, or neutral towards a proposition or target. The target may be a person, an organization, a government policy, a movement, a product, etc. Stance detection is different from Sentiment analysis, as the latter doesn't take into account the target towards which the statement is directed. Our system proposes a novel method for the detection of stance from text. Since there are thousands of potential topics to take a stance on, most with little to no training data, we utilize a new dataset for stance detection namely VAST (Varied Stance Topics) that captures a wider range of topics and lexical variation than in previous datasets. We propose to experiment with different models for stance detection and compare their results. Additionally, we will be providing a user friendly interface which can be utilized to interact with the models. A detailed description of the methodology is presented, i.e. the training process, processing of data and the outcomes generation. The accuracy, the efficiency and the performance of the approach under different real scenarios would also be evaluated.

***Key words*:** Stance Detection, Context-free, Contextual, BERT, word2vec, zero-shot, few-shot, Grouped Topic Representation

Contents

List of Figures

List of Tables

Nomenclature

API	Application programming interface
BERT	Bidirectional Encoder Representations from Transformers
C	Contextual
CF	Context-free
DT	Decision Tree
FN	False Negative
FP	False Positive
GB	Gradient Boosting
GNB	Gaussian Naive Bayes
KNN	K-Nearest Neighbors
LR	Logistic Regression
NN	Neural Network
RF	Random Forest
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
VAST	Varied Stance Topics

Chapter 1

Introduction

This chapter presents the overview of the thesis topic selected, the Motivation behind thesis topic selected and scope of thesis work. It also provides salient contribution through this thesis and finally explains the organization of the thesis

1.1 Problem Definition

Introduction of social media platforms has allowed people to express their opinions freely on variety of topics. Some topics tend to be controversial and garners a spectrum of opinions. Understanding the stance of people on these topics and messages becomes crucial to understand the general opinion of public which can be used for further studies and analysis.Our project aims at implementing a platform that allows for the detection of stance from text. To implement this, we will be using stance detection models which take the input, pre-process it in the format suitable, extract the relevant features and deliver the predictions.Furthermore, we will be comparing the results of different models to find the best performing one. Along with that, we will be providing a user interface so that users can interact with the trained models for detection of stance.

1.2 Motivation of the thesis

Stance detection plays a major role in analytical studies measuring public opinion on social media, particularly on political and social issues. The nature of these issues is usually controversial, wherein people express opposing opinions toward differentiable points.Social media heavily influences people and their perspective of the world.Understanding people's

opinions towards a particular entity becomes key to understanding trends. This is where stance detection comes in. Social issues such as abortion, climate change, and feminism have been heavily used as target topics for stance detection on social media.Similarly, political topics, such as referendums and elections, have always been hot topics that have been used in stance detection to study public opinion.Stance Detection helps us understand the stance of the general public towards the issues/ organizations/ person, etc.

1.3 Scope of the thesis

The main aim of the project is to pre-process the text data and detect stance from text.This project will detect the stance of the inputs (target and reply) provided and classify it into one of the following: 'favor', 'against', or 'neutral'. After classification,it will display the results using a graphical user-friendly interface and compare results generated by different algorithms.

1.4 Salient Contribution

Our Project has taken inspiration from [5] which introduces a new dataset VAST and a new method to evaluate targets in stance detection in the form of topic grouping. In their future scope, they have mentioned about trying new and complicated methods in topic grouping. Through this project, we have introduced a new method of topic grouping in the form of 'Bertopic' which shows improvement in results compared to other topic grouping method discussed in the paper. Additionally, we have compared different models which include baseline sklearn machine learning models as well as different neural network models. We have also introduced Gradient Boosting Classifier model which shows significantly good results.

1.5 Organisation of the Thesis

The report is organized into 8 chapters. Each chapter gives detailed information about the different aspects of the project.

- Chapter 1 presents the overview of the topic selected for the project. It contains information about the existing systems related to our project, our proposed system as well as the scope of the project.
- Chapter 2 presents the background research and study of existing systems and references.
- Chapter 3 lays out the project management plan which will be employed for the project. It gives a brief overview of the project organization and the project management plan.
- Chapter 4 presents the software requirements specifications for the project. It lists out the various requirements such as hardware, software and user interface requirements along with functional and non-functional requirements.
- Chapter 5 gives detailed information about the system architecture design, user interface design and the data flow diagram.
- Chapter 6 describes the technologies used, algorithm , and methodology for implementation of the project as well as results obtained.
- Chapter 7 describes the test plan and the test cases for the project. It provides an overview of the test approach used for the project.
- Chapter 8 provides the conclusion of the project along with the scope for future work.

In this chapter, we discussed the overview of the thesis topic selected, the Motivation behind thesis topic selected and scope of thesis work. It also provides salient contribution through this thesis and finally explains the organization of the thesis. Next chapter will cover the literature survey on Stance Detection.

Chapter 2

Literature Survey

This chapter presents the literature survey in the field of Stance Detection from text.

Stance Identification by Sentiment and Target Detection

The authors discussed the characteristics of different types of topics, and the interaction among sentiment, target, and stance in a sentence[1]. They proposed an approach without the need of stance-labeled data to identify stance incorporating the findings of their interaction. The proposed approach is topic independent and can be applied to individual topics flexibly. They evaluated their method on the SemEva1-2016 dataset for detecting stance in tweets, which contains six topics of two different types. Their experimental results show that the approach is promising even when stance-labeled data is not available.

BiLSTM-Autoencoder Architecture for Stance Prediction

The authors discuss a stance prediction technique using the Deep Learning approach, which can be used as a factor to determine the authenticity of news articles[2]. The Fake News Stance Prediction is the process of automatically classifying the stance of a news article towards a target into one of the following classes: Agree, Disagree, Discuss, Unrelated. The stance prediction task input is the news articles containing a pair: a headline as the target and a body as a claim. The paper proposes a deep learning architecture using Bi-directional

Long Short Term Memory and Autoencoder for stance prediction. They illustrate, through empirical studies, that the method is reasonably accurate at predicting stance, achieving a classification accuracy as high as 94%. The proposed stance detection method would be useful for assessing the credibility of news articles.

A Multilingual Multi-Target Dataset for Stance Detection

This paper focuses on the Multilingual Multi-Target dataset for stance detection where they proposed a much larger dataset that combined multilinguality and a multitude of topics and targets[3]. They created X-stance dataset based on BERT model which comprised of more than 150 questions about Swiss politics and more than 67k answers given by candidates running for political office in Switzerland. Questions are available in four languages: English, Swiss Standard German, French, and Italian. Here the language of a comment depended on the candidate's region of origin. They were successfully able to create dataset that extended over a broad range of topics and issues regarding national Swiss politics.

A Dataset for Multi-Target Stance Detection

The authors have focused on the problem of multi-target stance detection. They experimented with several neural models on the dataset and showed that they are more effective in jointly modeling the overall position towards two related targets compared to independent predictions and other models of joint learning, such as cascading classification[4]. Their goal was to provide a benchmark dataset to jointly learn subjectivities corresponding to related targets after which they investigated the problem of jointly predicting the stance expressed towards multiple targets (two at a time), in order to demonstrate the utility of the dataset. Finally they made the new dataset publicly available, in order to facilitate further research in multi-target stance classification.

Zero-Shot Stance Detection: A Dataset and Model using Generalized Topic Representations

The authors of the paper focus on zero-shot stance detection: classifying stance from no training examples. In the paper, they presented a new dataset for zero-shot stance detection that captured a wider range of topics and lexical variation than in previous datasets. Additionally, they proposed a new model for stance detection that implicitly captures relationships between topics using generalized topic representations and showed that the model improves performance on a number of challenging linguistic phenomena.
The developed model TGA Net, which uses generalized topic representations to implicitly capture relationships between topics, performed significantly better than BERT for stance detection on pro labels, and performed similarly on other labels. In addition, extensive analysis showed that the model provided substantial improvement on a number of challenging phenomena (e.g., sarcasm) and was less reliant on sentiment cues that tend to mislead the models. The models were evaluated on a new dataset, VAST, that has a large number of topics with wide linguistic variation and that they create and make available.

Stance Detection with bi-directional encoding

The authors felt a need to learn a model that interprets the tweet stance towards a target that might not be mentioned in the tweet itself. Secondly, to need to learn such a model without labeled training data for the target with respect to which we are predicting the stance. To address these challenges, the authors of [8] developed a neural network architecture based on conditional encoding. A long-short term memory (LSTM) network is used to encode the target, followed by a second LSTM that encodes the tweet using the encoding of the target as its initial state. They compared 3 techniques: target independent models, target dependent models and bi-directional encoding.It was found that bi-directional encoding delivered the best results.

Unsupervised User Stance Detection on Twitter

Unsupervised Learning is where the model is not provided with any labels. The paper [19] , Unsupervised User Stance Detection on Twitter, experiments with different methods and datasets and finally proposes a method for dimensionality reduction and then clustering to find clusters of data , which can be labeled with their corresponding stances manually later on. Given the data, the authors first calculate the cosine similarity between each pair of users, for features like number of unique tweets (feature space T), the number of unique hashtags (feature space H), or the number of unique retweeted accounts (feature space R). Based on the aforementioned cosine similarity between users, they experiment with different dimensionality reduction techniques : UMAP, t-SNE and FD. After projecting the user data into 2-dimensional space, they experimented with DBSCAN and Mean shift clustering techniques. These clusters then were manually labelled and evaluated. The datasets used were a combination of Kavanaugh Dataset (English), Trump Dataset (English) and Erdogan dataset (Turkish). Their most accurate setups use retweeted accounts as features, either the Fruchterman-Reingold force-directed algorithm or UMAP for dimensionality reduction, and Mean Shift for clustering, with UMAP being significantly faster than Fruchterman-Reingold. These setups were able to identify groups of users corresponding to the predominant stances on controversial topics with more than 98 percent purity based on our benchmark data.

In this chapter, we covered the literature review on the topic of Stance Detection. Next chapter covers project management plan which includes deliverables, process model, identification and division of roles and responsibilites,identification of tasks and mapping them into a timeline and the tools and technologies required for implementing these tasks.

Chapter 3

Sofware Project Management Plan

This chapter presents project management plan which includes deliverables, process model, identification and division of roles and responsibilites,identification of tasks and mapping them into a timeline and the tools and technologies required for implementing these tasks.

3.1 Introduction

This chapter establishes roles, responsibilities, processes and schedules for managing the software development process, in addition to outlining the tools, methods, and procedures to be used.This chapter contains a plan for the successful execution of the goals and lays out the project management plan which will be employed for the project. It gives a brief overview of the project organization and the project management plan.

3.1.1 Project Overview

Introduction of social media platforms has allowed people to express their opinions freely on variety of topics. Some topics tend to be controversial and garners a spectrum of opinions. Understanding the stance of people on these topics and messages becomes crucial to understand the general opinion of public which can be used for further studies and analysis. Our project aims at implementing a platform that allows for the detection of stance from text. To implement this, we will be using stance detection models which take the input, preprocess it in the format suitable, extract the relevant features and deliver the predictions. Furthermore, we will be comparing the results of different models to find the best performing one. Along

with that, we will be providing a user interface so that users can interact with the trained models for detection of stance.

3.1.2 Project Deliverables

Projects create deliverables, which are simply the results of the project or the processes in the project. That means a deliverable can be something as big as the objective of the project itself or the reporting that is part of the larger project Following are the deliverable of this project:

- Software Requirement Specification
- Software Project Management Plan
- Software Design Document
- Software Test Document
- Completed application
- Synopsis

3.2 Project Organisation

3.2.1 Software Process Model

The approach that will be utilized for development of Stance Detection System is Spiral Model

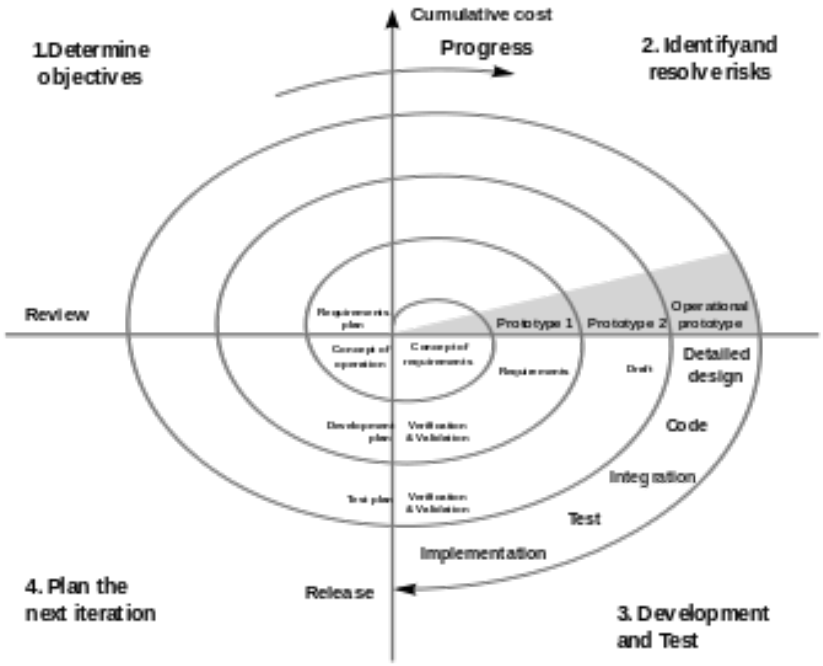


Figure 3.1: Spiral Model

Spiral model follows an evolutionary approach. It consists of spirals, each spiral consists of framework activites, for illustrative purposes, the figure contains generic 5 framework activites. Inner spirals focus on identifying software requirements and project risks; may also incorporate prototyping.Outer spirals take on a classical waterfall approach after requirements have been defined.We propose to use the Spiral Software Process Model in order to keep a constant sync in process of development along with planning,design ,maintenance, review and testing. It also allows us to assess the risks at the start of each loop and address them accordingly.

3.2.2 Roles and Responsibilities

- **Project Manager (Nishavak Naik)**:
  Role Description: Project Manager is responsible for working and assigning duties to the team members. Communication with faculty about progress and timely execution of the assignments. Establish the vision and scope of the project. Develop a plan to manage the project, gather the requirements, and document the plan.

- **Designer (Atharva Kitkaru, Nishavak Naik, Krisha Mehta)**:

Role Description: Designer works as part of a collaborative development team to manage the look-and feel of the project, and ensure that the project's GUI adheres to design standards, analyze user needs and developer requirement specifications and transform requirements into design of product.

- **Developer (Atharva Kitkaru, Nishavak Naik, Krisha Mehta)**:
  Role Description: Developers get familiar with applications and resources that may be available. Support to make the team more effective, and learn about the languages that the team will use. Code for application and programs for backend processing systems to build a working project fulfilling the requirements specified in System Requirements Specification.

- **Tester (Atharva Kitkaru, Nishavak Naik, Krisha Mehta)**:
  Role Description: Testers are responsible for documenting problems, defining methodologies for testing and responsibilities, and managing schedules for testing. Develop and describe the test plan of the system and prepare use case procedures.

3.2.3 Tools and Techniques

- Various text editors, Overleaf and ShareLaTeX for documentation.
- Draw.io to make all the UML diagrams.
- GanttPro (an online tool to make Gantt charts) to map the process scheduling Gantt chart.
- Visual Studio IDE / Sublime Text, Jupyter Notebook, Google Colab will be used for developing the application and testing of machine learning models
- ReactJS for frontend Development, Django for backend
- Python Libraries: Keras, Tensorflow, nltk, Numpy, Pandas, matplotlib, seaborn
- Selenium for interface testing.

3.3 Project Management Plan

3.3.1 Tasks

- Understanding basic project requirements
- Conducting Literature Survey
- Requirement Gathering and Analysis
- Planning
- Designing
- STD
- Dataset Exploration, Preprocessing and feature extraction
- Development of Models for Stance Detection
- Fine tuning hyperparameters
- Analysing, visualizing, and validating the models
- Creation of web interface and integration with models
- Testing
- Final Report Creation and Submission

3.3.1.1 Task name: Understanding basic project requirements

- **Description:**
  Formulate a proper problem definition by deciding need,objectives and outcomes of the project.
- **Deliverables and Milestones:**
  Problem Definition
- **Resources Needed:**
  Internet, Web Browser and Text Editor

- **Dependencies and Constraints:**
  None
- **Risks and Contingencies:**
  Lack of Domain Knowledge

3.3.1.2 Task name: Conducting Literature Survey

- **Description:**
  Go through all the research papers related to the topic to find out the background , current work and future scope of the project.
- **Deliverables and Milestones:**
  Literature review of topic
- **Resources Needed:**
  Internet, Web Browser and Text Editor
- **Dependencies and Constraints:**
  None
- **Risks and Contingencies:**
  None.

3.3.1.3 Task name: Research Requirement Gathering and Analysis

- **Description:**
  Decide Functional, Non-Functional, Software, Hardware and other requirements for the research project.
- **Deliverables and Milestones:**
  Software Requirement Specification
- **Resources Needed:**
  Latex
- **Dependencies and Constraints:**
  None

- **Risks and Contingencies:**
  Wrong Interpretation or ambiguity, Inconsistent SRS , Missing requirements.

3.3.1.4 Task name: Planning

- **Description:**
  Deciding on the deliverables, schedule, task assignments, resources required, understanding the risks and contengencies and preparing a detailed document covering all this with a Gantt chart.

- **Deliverables and Milestones:**
  Software Project Management Plan

- **Resources Needed:**
  Gantt Pro, Latex.

- **Dependencies and Constraints:**
  SRS

- **Risks and Contingencies:**
  Unrealistic / inconsistent planning.

3.3.1.5 Task name: Designing

- **Description:**
  In this task, we focus on developing the UI Designing which plays an important part in how the user will be interacting with the product software and also designing the UML diagrams which defines how the product will be created to capture the system's functionality and requirements along with its dynamic behavior.

- **Deliverables and Milestones:**
  UML Diagrams, Final Version of SDD

- **Resources Needed:**
  Draw.io, Latex.

- **Dependencies and Constraints:**
  SPMP

- **Risks and Contingencies:**
  Incorrect/ lack of mapping of requirements to design, incomplete UML diagrams.

3.3.1.6 Task name: System Test Document

- **Description:**
  Deciding testing strategies, tools and techniques, generation of test cases and preparation of STD

- **Deliverables and Milestones:**
  STD

- **Resources Needed:**
  Latex

- **Dependencies and Constraints:**
  SRS, SPMP and SDD

- **Risks and Contingencies:**
  Incorrect test cases, not enough test case coverage.

3.3.1.7 Task name: Dataset Exploration, Preprocessing and feature extraction

- **Description:**
  Exploring the dataset and performing operations on the dataset to make it ready for feature extraction. The preprocessed data is then used to extract important features which can be used to feed the models.

- **Deliverables and Milestones:**
  Pre-processed data along with features extracted.

- **Resources Needed:**
  Google colab/Jupyter Notebook, Google Drive

- **Risks and Contingencies:**
  Improper data cleansing/ preprocessing leading to improper features

3.3.1.8 Task name: Development of Models for Stance Detection

- **Description:**
  Training models using algorithms such as SVM, CNN, LSTM, Random Forrest for Stance Detection
- **Deliverables and Milestones:**
  Trained Models
- **Resources Needed:**
  Google colab/ Jupyter Notebook
- **Dependencies and Constraints:**
  Preprocessed data with features extracted.
- **Risks and Contingencies:**
  Overfitting and Underfitting of the models.

3.3.1.9 Task name: Fine tuning hyperparameters

- **Description:**
  Analysing and modifying the hyperparameters of the trained models to further enhance the accuracy of the models.
- **Deliverables and Milestones:**
  Trained Models with the best possible hyperparameters.
- **Resources Needed:**
  Google colab
- **Dependencies and Constraints:**
  Trained models.
- **Risks and Contingencies:**
  Overfitting and Underfitting of the models.

3.3.1.10 Task name: Analysing, visualizing, and validating the models

- **Description:**
  Analysing and visualizing the trained model to gain insights into its working and finding any areas for further improvement. Performing validation on the testing data using the trained models.
- **Deliverables and Milestones:**
  Visualization Results with analysis.
- **Resources Needed:**
  Google colab
- **Dependencies and Constraints:**
  None
- **Risks and Contingencies:**
  None.

3.3.1.11 Task name: Creation of web interface and integration with models

- **Description:**
  Build a complete software application with an intuitive UI which can interact with the models
- **Deliverables and Milestones:**
  Web interface that can interact with the models.
- **Resources Needed:**
  Google colab, VS Code
- **Dependencies and Constraints:**
  None
- **Risks and Contingencies:**
  None.

3.3.1.12 Task name: Testing

- **Description:**
  Testing the final application for bugs and updating the STD
- **Deliverables and Milestones:**
  Web interface that can interact with the models.
- **Resources Needed:**
  Selenium
- **Risks and Contingencies:**
  Incorrect execution of test cases, test cases are not exhaustive enough.

3.3.1.13 Task name: Final Report Creation and Submission

- **Description:**
  Submission of Final Report.
- **Deliverables and Milestones:**
  Web interface that can interact with the models.
- **Resources Needed:**
  Latex
- **Risks and Contingencies:**
  None.

3.3.2 Assignments

- **Understanding basic project requirements**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Conducting Literature Survey**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Requirement Gathering and Analysis**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Planning**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Designing**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta

- **STD**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Dataset Exploration, Preprocessing and feature extraction** : Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Development Models for Stance Detection**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Fine tuning hyperparameters**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Analysing, visualizing, and validating the models**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Creation of web interface and integration with models**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Testing** : Atharva Kitkaru, Nishavak Naik, Krisha Mehta
- **Final Report Creation and Submission**: Atharva Kitkaru, Nishavak Naik, Krisha Mehta

3.3.3 Timetable

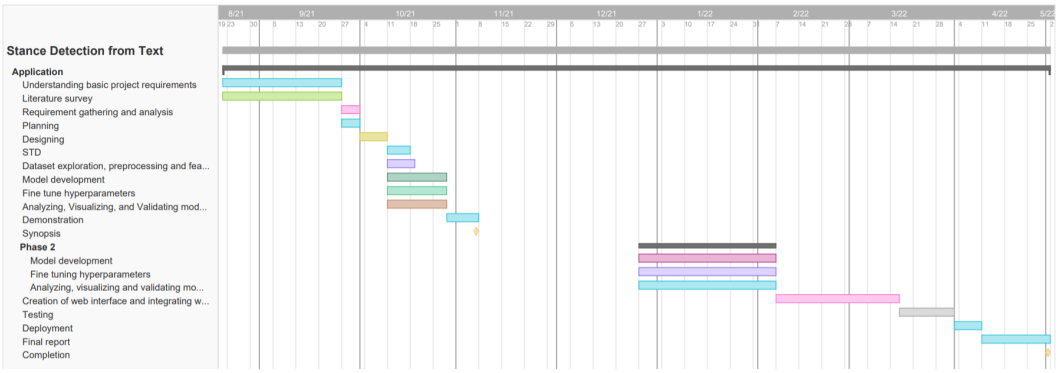


Figure 3.2: Gantt Chart

In this chapter, we discussed different elements of software management plan such as deliverables, tasks, timeline, tools and technologies required. The next chapter discusses the requirements in the form of software, hardware and interface requirements. It also discusses the functionalities of the product and the system attributes

Chapter 4

Software Requirements Specification

This chapter presents project management plan which includes deliverables, process model, identification and division of roles and responsibilites,identification of tasks and mapping them into a timeline and the tools and technologies required for implementing these tasks.

4.1 Introduction

This chapter presents the software requirements specifications for the project. It lists out the various requirements such as hardware, software and user interface requirements along with functional and non-functional requirements.

4.1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for Stance Detection System. It will explain system constraints, system features, interfaces and functionality of the system. This document is primarily intended to help the stakeholders with basic requirements of the software that are to be provided to the end users.

4.1.2 Overview

Introduction of social media platforms has allowed people to express their opinions freely on variety of topics. Some topics tend to be controversial and garners a spectrum of opinions.

Understanding the stance of people on these topics and messages becomes crucial to understand the general opinion of public which can be used for further studies and analysis. Our project aims at implementing a platform that allows for the detection of stance from text. To implement this, we will be using stance detection models which take the input, preprocess it in the format suitable, extract the relevant features and deliver the predictions. Furthermore, we will be comparing the results of different models to find the best performing one. Along with that, we will be providing a user interface so that users can interact with the trained models for detection of stance.

4.1.3 Objectives

- Implement machine learning algorithms to detect stance.
- Evaluation of results and comparison of different algorithms using different performance metrics
- Development of a user interface which allows users to detect stance with the use of trained models.

4.1.4 Important Terminologies

- Raw data: It's basically the input data which we will be fed to the model.
- Database: Collection of information on different topics related to each other.
- Preprocessing: Preprocessing includes various methods (tokenize, word check, stemming, tagging, topic collection, verb collection, topic vectorization) which will help us to convert the raw data into useful feature vectors.
- Linguistic approach: This approach (e.g., natural language processing or NLP) are focused on news content, and aim to investigate fake news patterns by analyzing underlying semantics.
- Classifier: An algorithm that implements classification, especially in a concrete datasets. The classifier will help to classify the input into various categories.

- Model: Model here basically can be machine learning model (SVM, Logistic Regression, Random Forest), Deep Learning Models like CNN, RNN, LSTM, pretrained models like BERT or a combination of different approaches which is used for classification.
- Word Embedding: Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. Different approaches for generation of word embeddings are Glove, Word2Vec, BERT, TF-IDF,etc.

4.2 Specific Requirements

4.2.1 External Interface Requirements

4.2.1.1 User Interfaces

The System should be user-friendly such that a user with no prior experience of using this software should be able to use and interact with the models to detect stance easily.

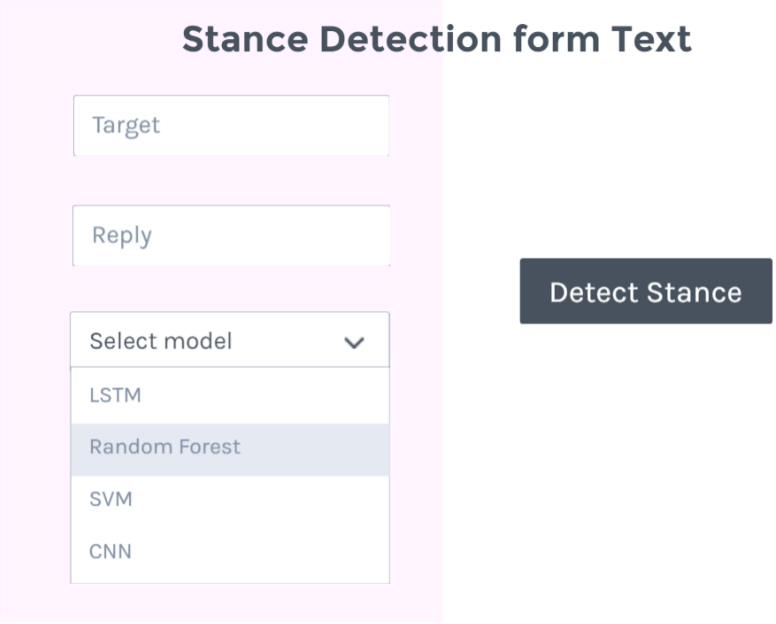


Figure 4.1: Home page

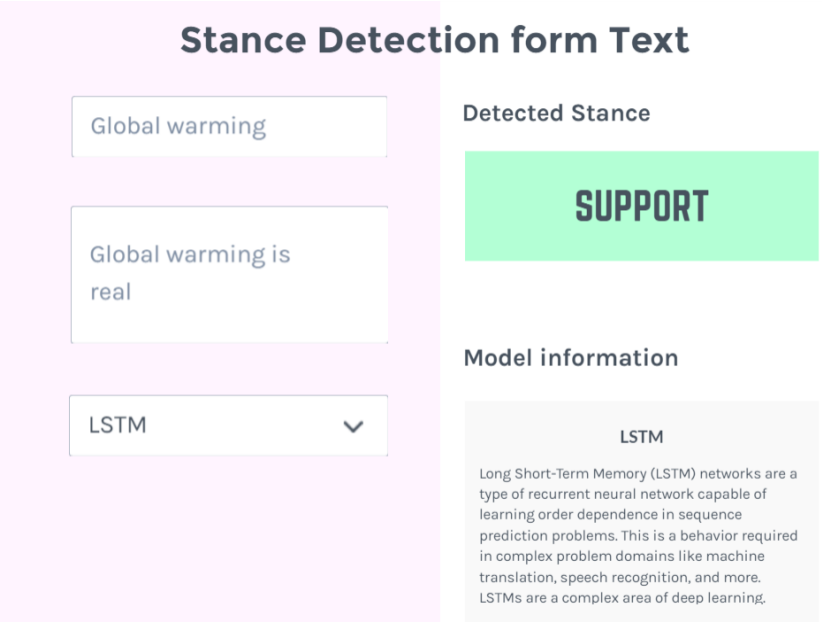


Figure 4.2: Model prediction

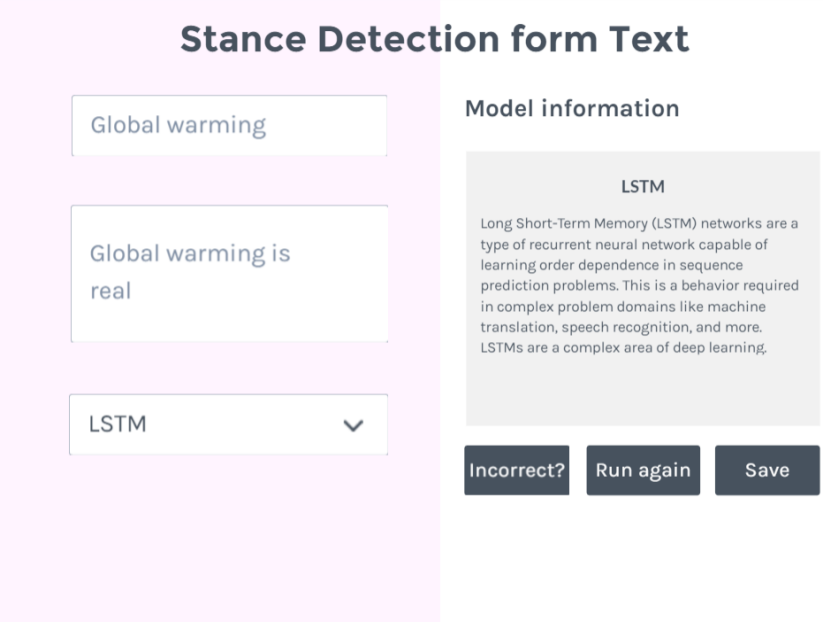


Figure 4.3: Model information and confidence

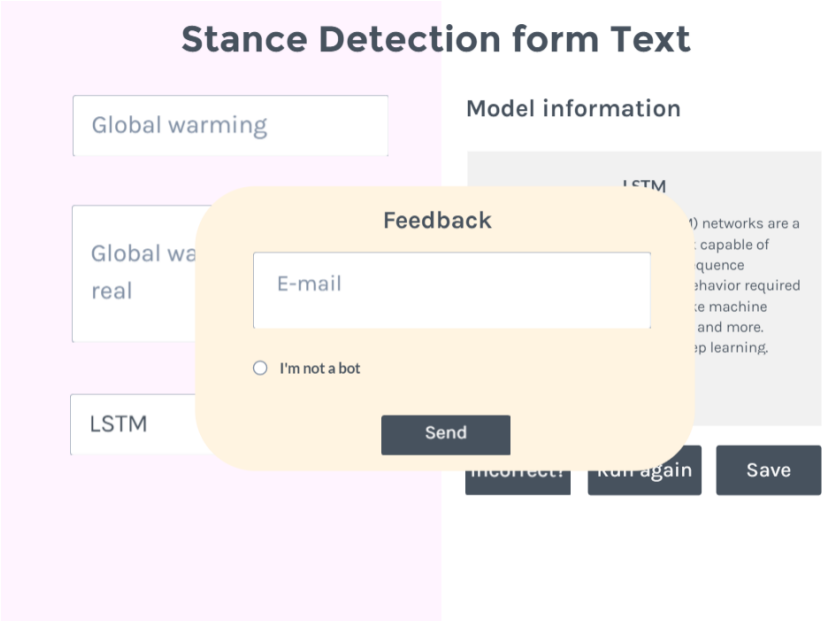


Figure 4.4: Reporting incorrect predictions

4.2.1.2 Hardware Requirements

The minimum Requirement needed to run the Stance Detection System seamlessly:

1. Personal Computer Laptop:
    - Windows 8 or later or MacOs 10.5.8 or later or Ubuntu Linux 8.04 or later
    - Processor: Pentium Dual Core 1.66 GHzand above
    - Active Internet Connection

4.2.1.3 Software Requirements

1. Web client: Chrome, Mozilla, Edge,etc
2. Programming language: Python, Javascript
3. Library : Tensorflow, keras, nltk, sklearn, React.js
4. Platform : Google colab, Jupyter notebook

4.2.1.4 Communication Protocol

The Main Communication Protocol that will be used is HTTP(HyperText Transfer Protocol). This will be used to transfer information back and forth from the client to server and vice versa. HTTP GET FTP and POST will be used to send information securely over the web browser.

4.2.2 Software Product Features

1. The system should be able to preprocess the text data and should be able to deal with symbols too.
2. The system should be able to detect the stance of the inputs (target and reply) provided and classify it into one of the following: 'favor', 'against', or 'neutral'.
3. The System will provide a graphical user-friendly interface for a better understanding of results.
4. It should be able to generate performance results with appropriate metrics.
5. It should be able to compare results generated by different algorithms.

4.2.3 Software System Attributes

4.2.3.1 Availability:

The system should be designed in such a way that it is available 24/7 to the users and they can access the system whenever they intend to.

4.2.3.2 Portability:

The system must operate in any operating system.

4.2.3.3 Reliability:

The system has to be 80 percent reliable i.e. lets say for a time span of month, 80percent reliability means that under normal usage conditions, there is 80 percent chance that the system wont experience a critical failure. 20 percent of unreliability can be attributed to bugs in

the software which got overlooked/weren't uncovered during development, misjudgements, changing environments in which system is going to function

4.2.3.4 Maintainability/Modifiability:

The system should be easily modifiable. Any fixes should involve modifications only to that part and not the entire system, which can be achieved by writing code in a modular fashion.

4.2.3.5 Security:

The system should handle the sensitive information with care and make sure it's not leaked to potential threats.

4.2.3.6 Usability:

The system should be easy to use and provide results in an easy-to-understand format to the user.

In this chapter, we discussed the hardware, software and interface requirements. We also discussed the functional and non-functional requirements. Next chapter talks about software design description which maps the requirements discussed in this chapter into design

Chapter 5

Software Design Description

This chapter maps the software requirements into design. It coves system architecture design, user interface design, description of individual components, use case scenarios and data flow diagram

5.1 Introduction

This chapter outlines the detailed structure of our project components and the precise details of implementation needed to fulfill the specifications set out in the Software Requirements Specification (SRS). It is assumed that the reader has read the SRS, since this document also defines the implementation details of the desired behaviour given the requirements within it. This document will build heavily on the Software Architecture and Design and so knowledge of the general system architecture is recommended prior to commencing this document.

5.1.1 Design Overview

The project has two dimensions: Client and Server. The client is an application which is responsible for taking the input target and reply as text from the user and displaying the result received from the server. The server receives the request from the client, it processes the input and applies the algorithm to the input data and then returns the output to the client accordingly.

5.1.2 Requirements Traceability Matrix

Table 5.1: Requirements Traceability Matrix

	User	Machine Learning Models	Database
Target and opinion input	X		
Model Selection	X		
Hyperparameter tuning		X	
Report Generation and analysis	X	X	X

5.2 System Architectural Design

5.2.1 Chosen System Architecture

The chosen system architecture for this project is MVC. MVC is a software design pattern for developing a web application. MVC Structure consists of three parts namely Model, View and Controller.

Model acts as the interface of the data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database.

The View is the user interface what we see in our browser when we render a website.It is coded using HTML/CSS/JavaScript/TypeScript. A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

5.2.2 Discussion of Alternative Designs

Client Server is an alternative option which can be used for implementation of the project.Client Server is the simplest and a commonly used architectural pattern for developing applications. The client is the application's User Interface which accepts inputs and displays outputs and the server is a machine learning algorithm used for processing the inputs.

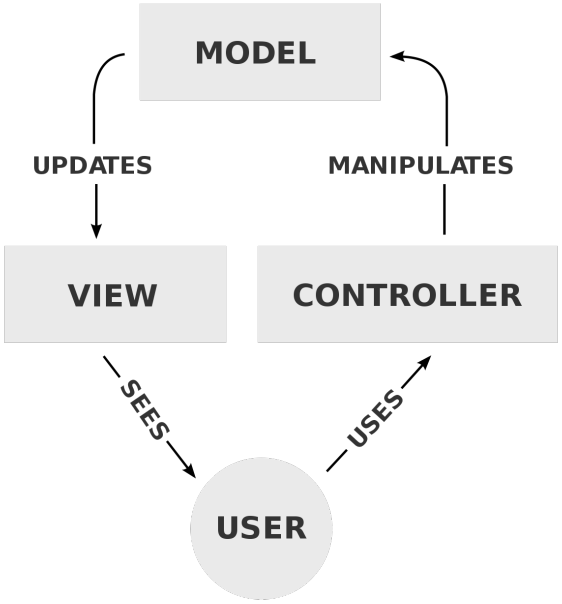


Figure 5.1: MVC Architecture

5.2.3 System Interface Description

5.2.3.1 User Interface

Stance detection from text application interface has input fields on the landing page where the users are supposed to enter the target and reply/opinion; after which the users are expected to select a machine learning model to proceed with the detection of stance from the input texts.

The data shall be then fed to the trained model and the detected stance shall be displayed in the output section. Furthermore, the verbose information of the used model will be proffered for getting insights of the model working and performance metrics.

Options are provided to the users to report an potentially incorrect prediction by the model, re-run the model prediction, and save the model outputs. The feedback section takes an e-mail input to identify the user who provided the feedback and let the user know about their contribution to the application's development. This is also to ensure protection against bot attacks.

5.2.3.2 Hardware Interface

Stance detection from text is an application which can run on any recent web browsers.

5.2.3.3 Software Interface

The software interface follows the Model-View-Controller (MVC) model for making and modeling data objects.

5.3 Detailed Description of the Components

5.3.1 User Interface

Table 5.2: User Interface

Responsibilities	1.To provide an interface using which user can access the respective application and its functionalities 2. Display the results
Constraints	It should be easy to use and responsive
Composition	ReactJS
Interaction	1. User inputs target, reply and chooses model 2. Users are displayed results 3. User may provide feedback on the predicted outputs

5.3.2 Machine Learning Model

Table 5.3: Machine Learning model

Responsibilities	To detect correct stance from the given inputs.
Constraints	Input being pre-processed based on considered model input parameter
Composition	Python ML libraries
Interaction	View of MVC architecture interacts with Model to produce and display outputs.

5.3.3 Database

Table 5.4: Database

Responsibilities	To store the results of the model and the feedbacks provided by the users
Constraints	Enforce proper data type values
Composition	SQLite
Interaction	Interaction with the system for user data

5.4 User Interface Design

5.4.1 Description of User Interface

5.4.1.1 Screen Images

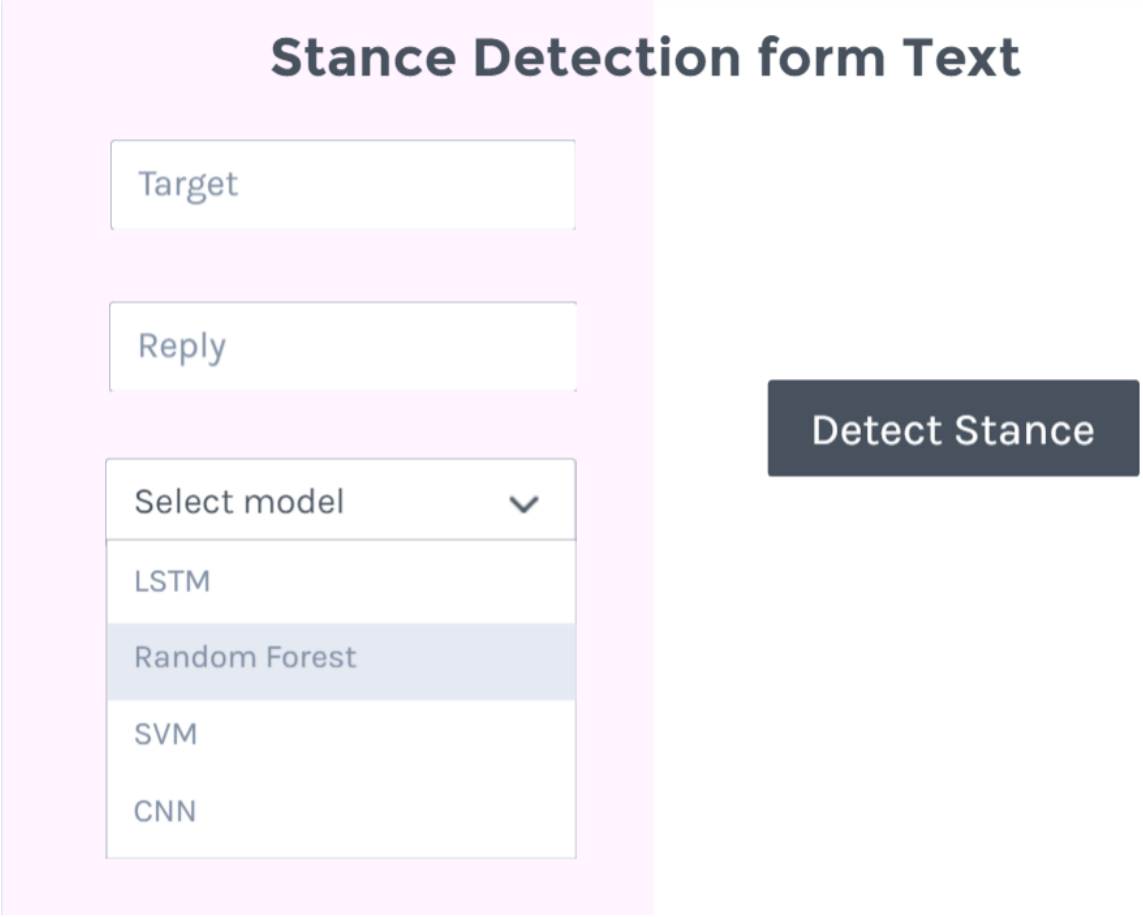


Figure 5.2: User Interface Prototype-Landing page

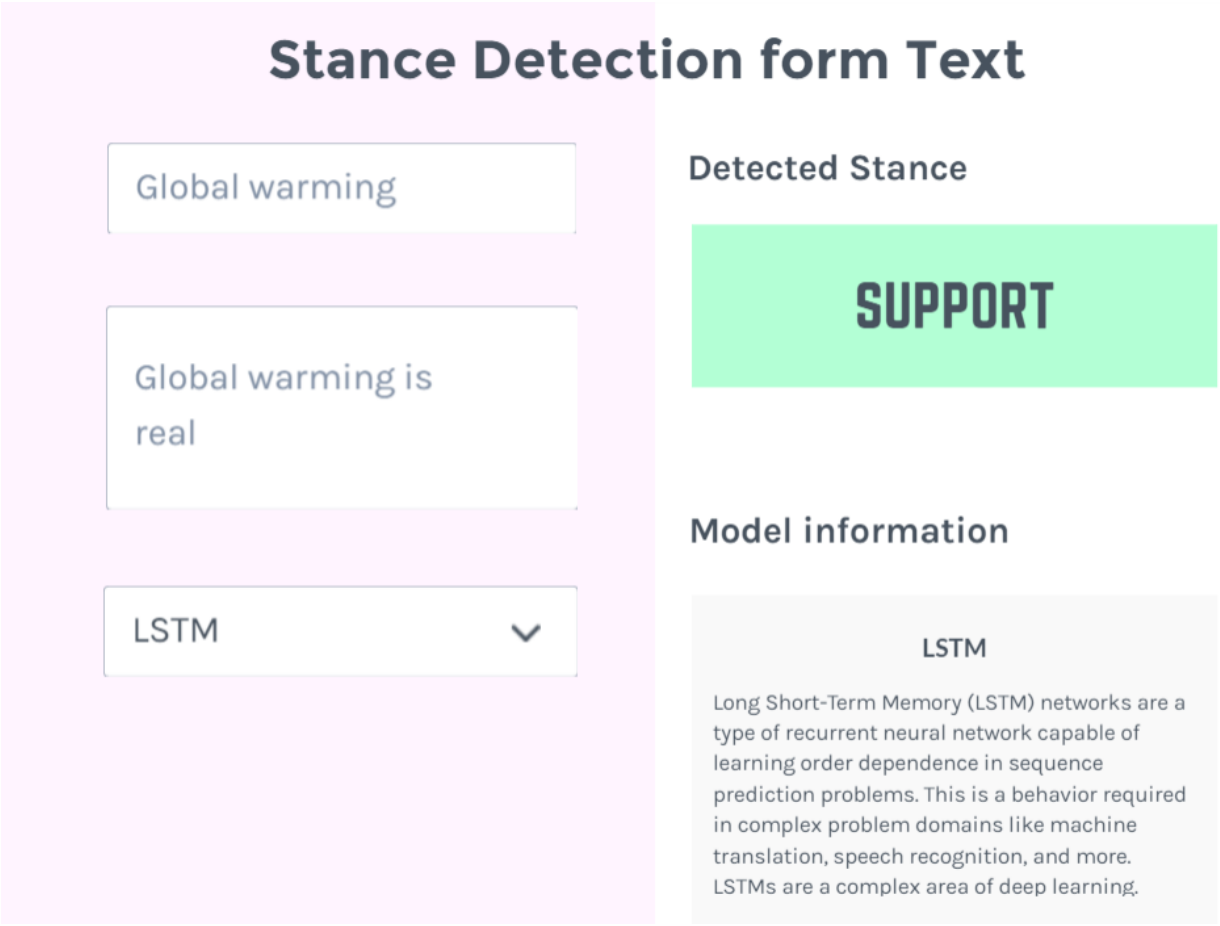


Figure 5.3: User Interface Prototype-Detected stance information

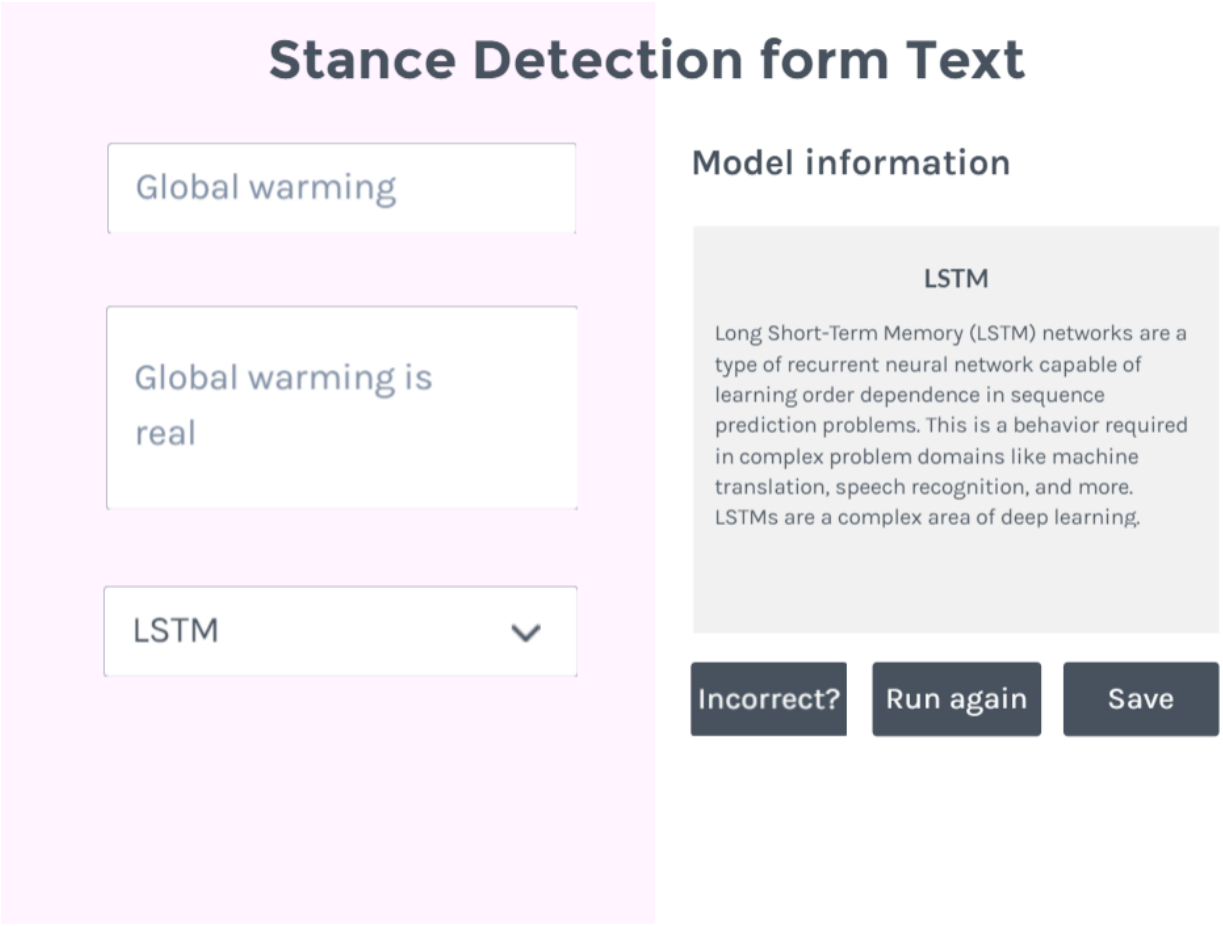


Figure 5.4: User Interface Prototype-Verbose details of the model

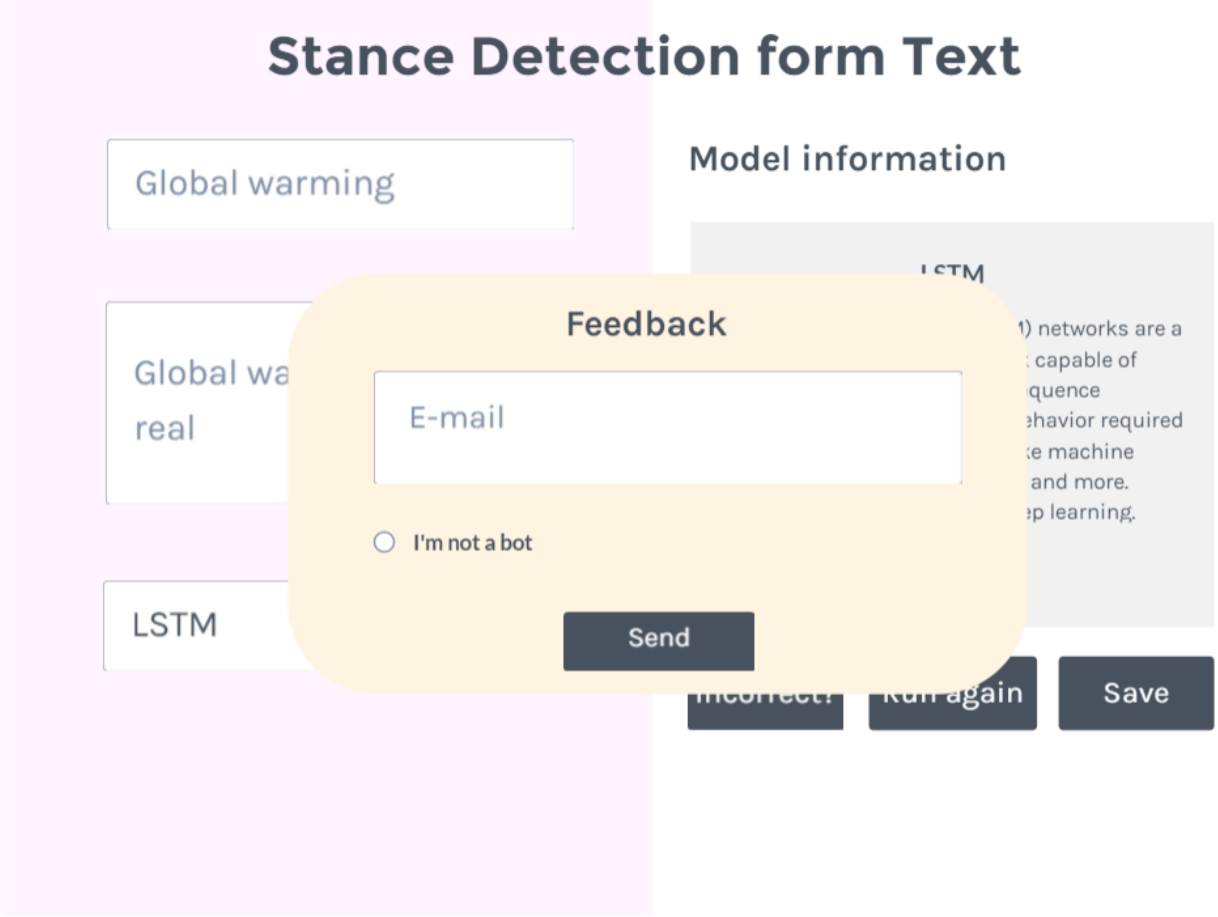


Figure 5.5: User Interface Prototype-Feedback

5.4.1.2 Objects And Actions

User Dashboard

The user can interact with different models using drop-downs and see the stance detected. Apart from that, user can also use auto-mode which uses a pool of good performing models to produce results.

Feedback Panel

Using this pane, the user can provide feedback about any discrepancy in the detected stance along with his email and optional feedback.

5.5 Use Case Specification

5.5.1 Use case 1

Table 5.5: Use Case: Fetch target and reply

Use Case ID:	1
Use Case name:	Fetch input
Primary Actor:	User
Description:	Used to insert data for stance detection
Trigger:	Clicking the "Detect Stance" button
Post conditions:	The detected stance shall be displayed in the results section
Normal flow:	1. User shall enter target and reply/opinion; select the machine learning model; and click on detect stance 2. After clicking the detect stance button, the loading screen appears whilst the model predicts the output. 3. The model's prediction is displayed in the results section.
Alternative flows:	If insufficient data is provided then the helper text asks user to enter the needed data.
Exceptions:	Data entered is not textual.
Priority:	High
Frequency of use:	High
Assumptions:	User has internet connectivity and recent web browser.

5.5.2 Use case 2

Table 5.6: Use Case name: Generate Report

Use Case ID:	2
Use Case name:	Generate output
Primary Actor:	User
Description:	The machine learning model predicts the detected stance based on the given inputs (target and reply/opinion). The additional model performance metrics is also provided along with the model to the user.
Trigger:	Click "Detect stance" button
Preconditions:	User has entered the target as well as the reply/opinion and has selected the desired machine learning model to use.
Post conditions:	The detected stance is displayed to the user.
Normal flow:	1. User clicks on "Detect Stance" button 2. The user inputs (target and reply/opinion) are processed. 3. Machine learning model is used to detect the output stance. 4. Generated report is displayed to the user.
Alternative flows:	If the data cannot be processed by the system, an error shall be displayed.
Exceptions:	Input data is not in text format
Priority:	High
Frequency of use:	High
Assumptions:	User has internet connectivity and recent web browser.

5.6 Data Flow Diagram

5.6.1 Context Level DFD

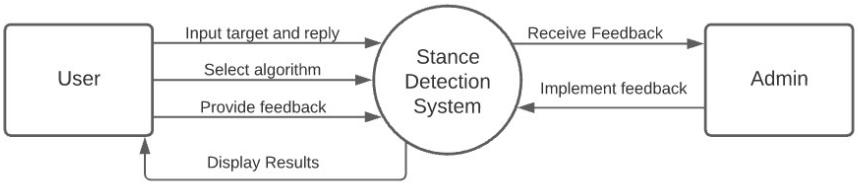


Figure 5.6: Context Level DFD

5.6.2 Level-0 DFD

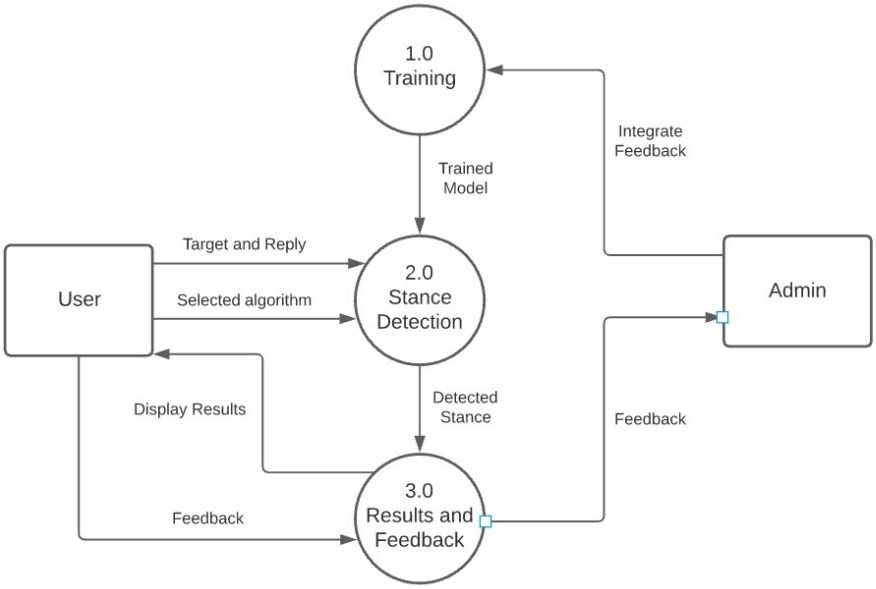


Figure 5.7: Level 0 DFD

5.6.3 Level-1 DFD

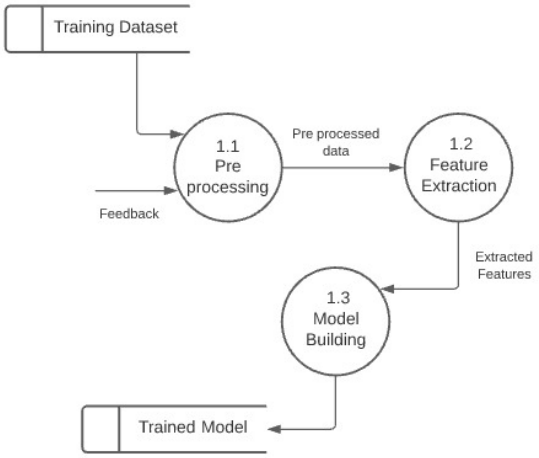


Figure 5.8: Level 1 DFD subprocess 1

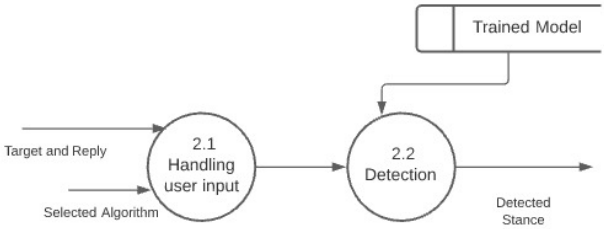


Figure 5.9: Level 1 DFD subprocess 2

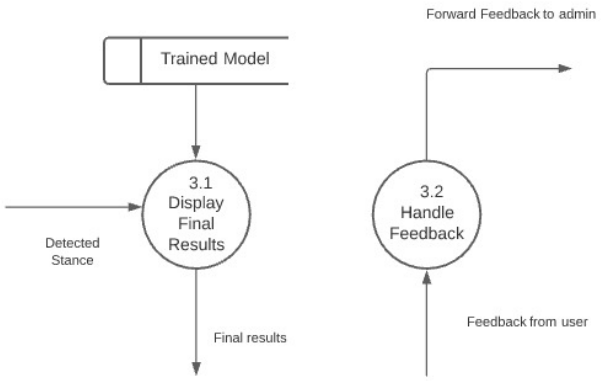


Figure 5.10: Level 1 DFD subprocess 3

In this chapter, we covered Design aspects of the product which include System Architecture, UI design, Use case specification, Data flow diagrams and description of components. Next chapter discusses the implementation aspect of the project.

Chapter 6

IMPLEMENTATION

This chapter presents technologies used, algorithm and implementation details of the project.

6.1 Technologies used

- Visual Studio Code as code editor and Overleaf for documentation.
- Draw.io to make all the UML diagrams.
- GanttPro (an online tool to make Gantt charts) to map the process scheduling Gantt chart.
- Visual Studio IDE / Sublime Text, Jupyter Notebook, Google Colab will be used for developing the application and testing of machine learning model.
- ReactJS for frontend and Django for backend development.
- Python Libraries: Keras, TensorFlow, NLTK, NumPy, Pandas, matplotlib, seaborn.
- Selenium for interface testing.

6.2 Algorithm

Stance Detection

1. Preprocessing of text (removal of specific stopwords, converting text to lower case, tokenization, lemmatization, removal of special characters).
2. Embedding text using Contextual SBERT encoder or Context-free Word2Vec encoder.
3. Grouping targets using Sklearn's Agglomerative clustering or BERTopic models.
4. Concatenating inputs and feeding them to the Sklearn and Neural Network models.
5. Fitting the models on the Training dataset.
6. Evaluating model using Validation and Testing dataset.
7. Fine-tuning model hyper-parameters to get better results.

Feedback

1. User clicks on the Report Incorrect Feedback button on the User Interface.
2. User fills out the form containing e-mail address, optional feedback message, and expected stance.
3. Feedback gets stored in the database.
4. The developer views the feedbacks and adds them to dataset if relevant.
5. Retrain the model with new dataset.

6.3 Implementation

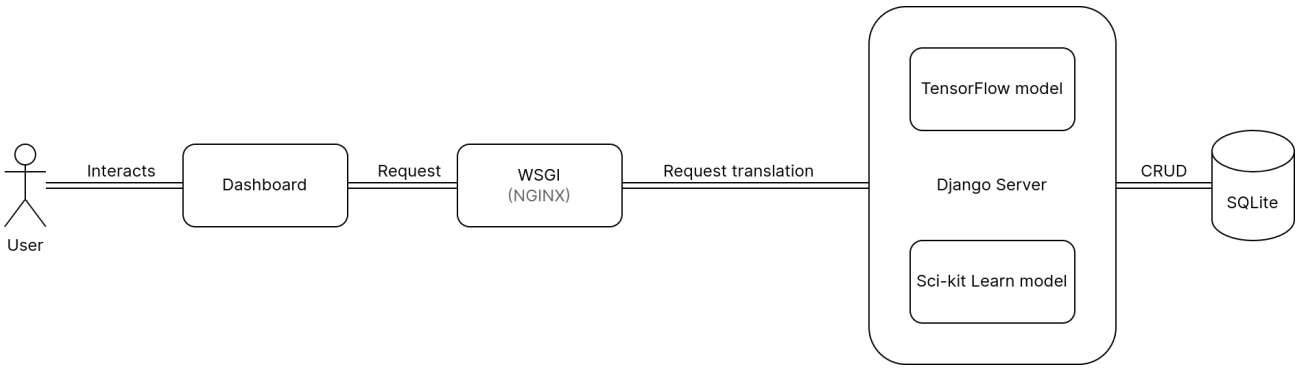


Figure 6.1: System Overview

6.3.1 Dashboard

6.3.1.1 Technologies used

We used React.js to build our client-side of the application. React is flexible due to its modular structure, thus easier to maintain. React is designed to provide high performance. The core of the library offered a virtual DOM program and server-side rendering, which made our complex application run extremely fast.
We used Sass to write our styles. Sass allowed us to compile styles which did not break across browsers. We also used Bootstrap CSS framework to reduce the code needed to build a visually appealing application interface.
React Toastify library allowed us to ping users with Execution completion, error encountered, or reported incorrect prediction notifications.
Voca library helped to transform text in the code.
Axios library allowed us to write Asynchronous network requests with ease.
React Router DOM made routing across the application components easy without needing to fetch entire application files from the internet again and again.

6.3.1.2 Flow

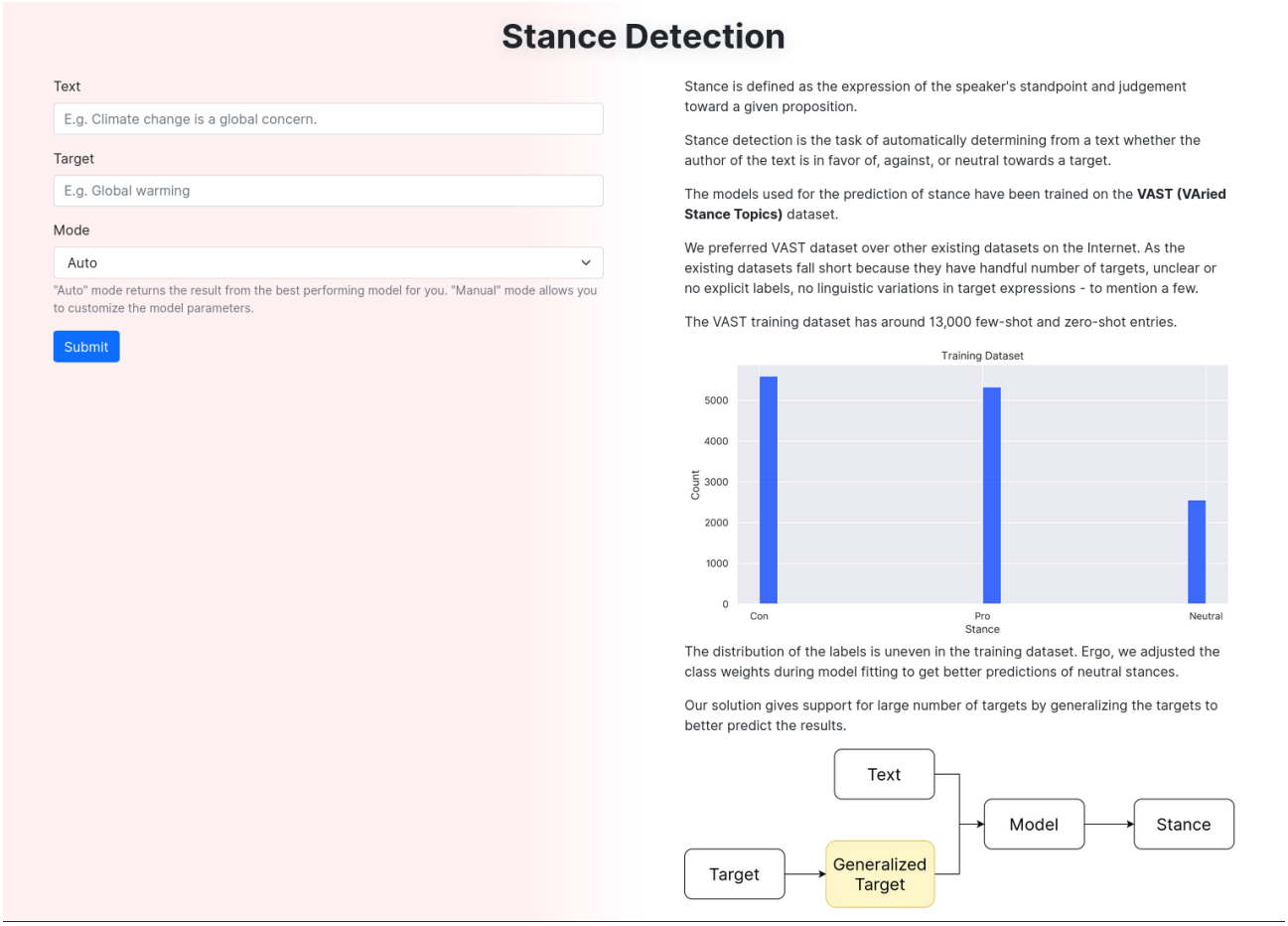


Figure 6.2: User lands on the Home page of the application

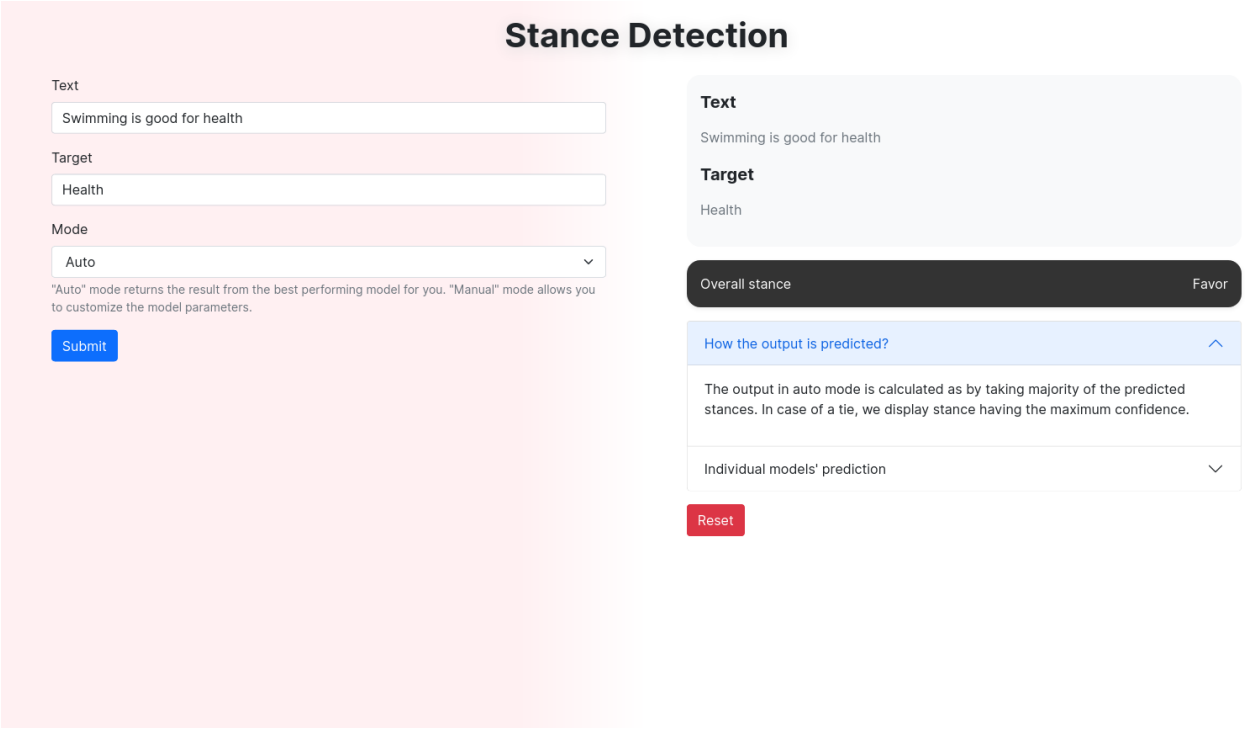


Figure 6.3: Detecting stance in Auto mode

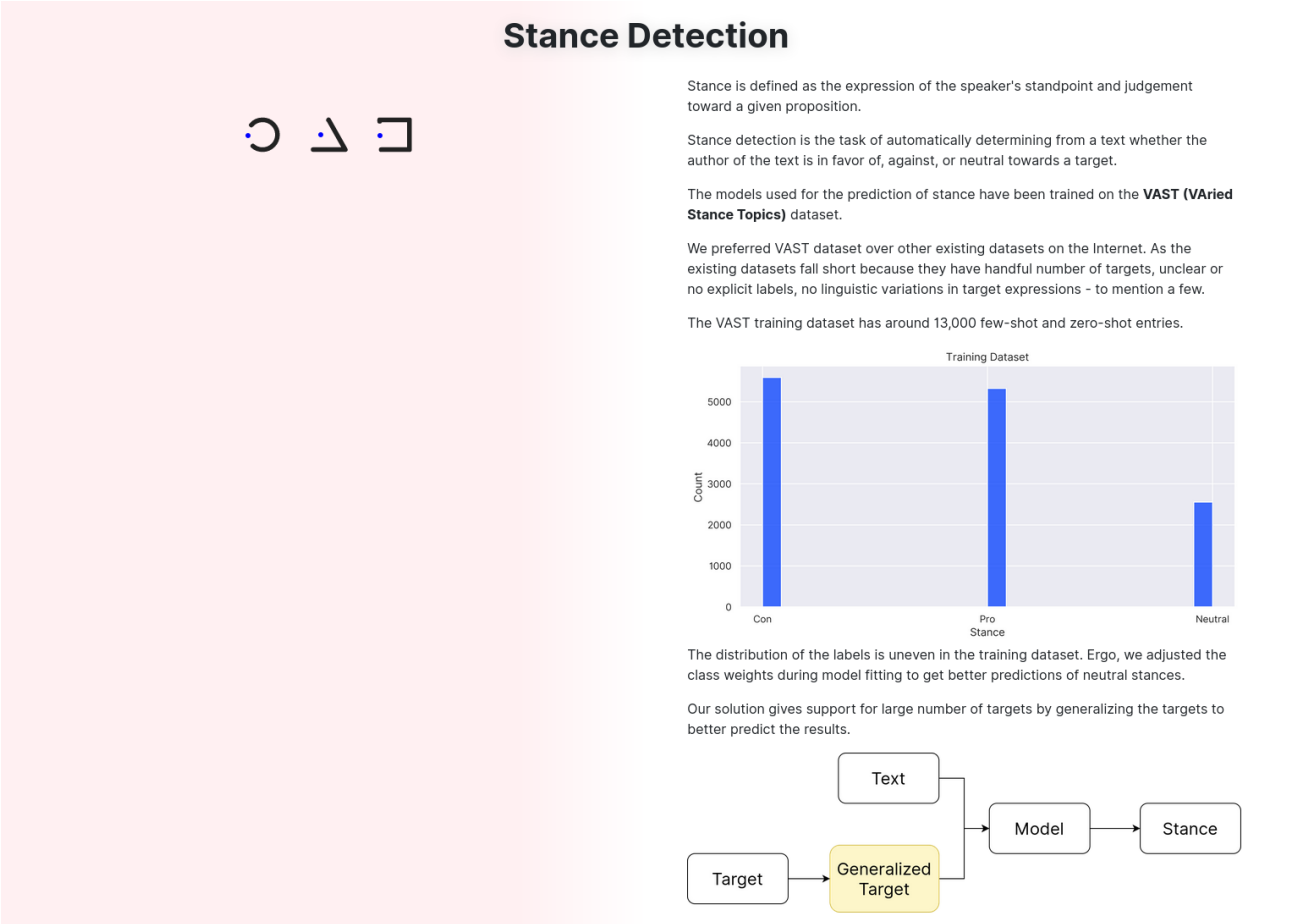


Figure 6.4: Loading the results

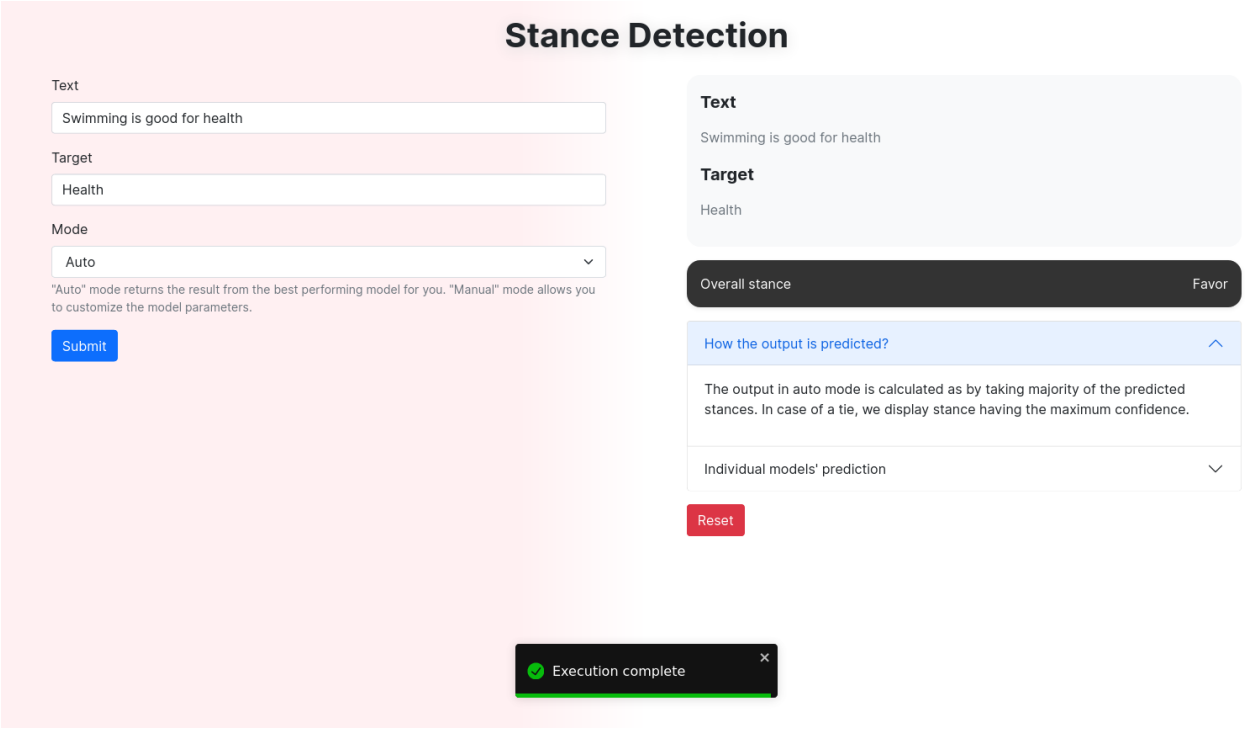


Figure 6.5: User notified about execution completion

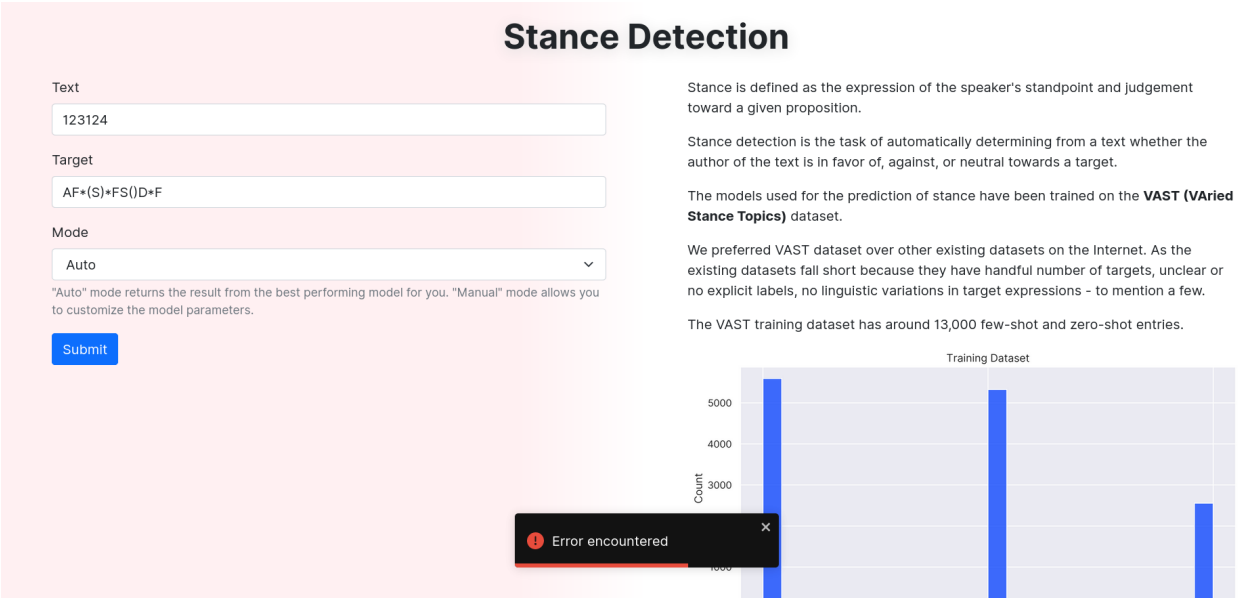


Figure 6.6: User notified of bad inputs

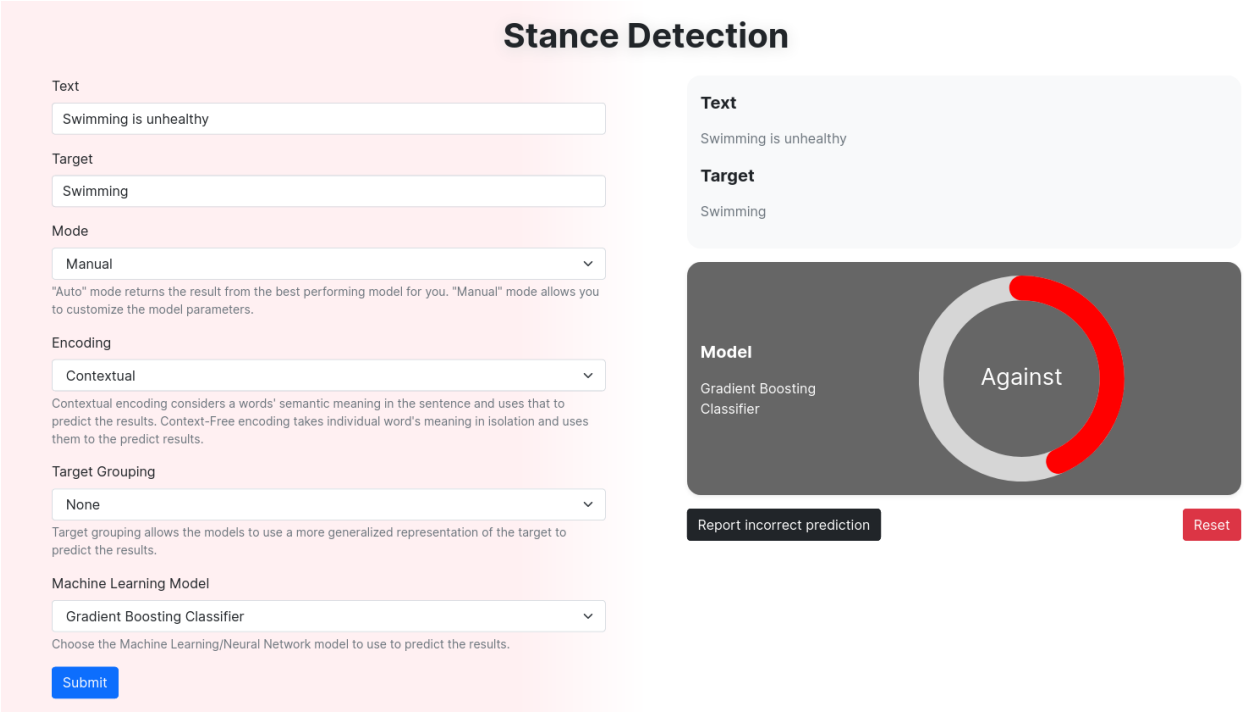


Figure 6.7: Manual mode

6.3.2 Django Server

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. It takes care of much of the hassle of web development, so we could focus on writing your app without needing to reinvent the wheel.

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that we could scale the application for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. It's also maintainable, highly secure and portable.

The Web Server Gateway Interface (WSGI) is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language. Our Django application server takes in the request from the WSGI and executes the calls to the models build using Sklearn and TensorFlow and returns the predicted stance to the user Dashboard via WSGI.

Model Building

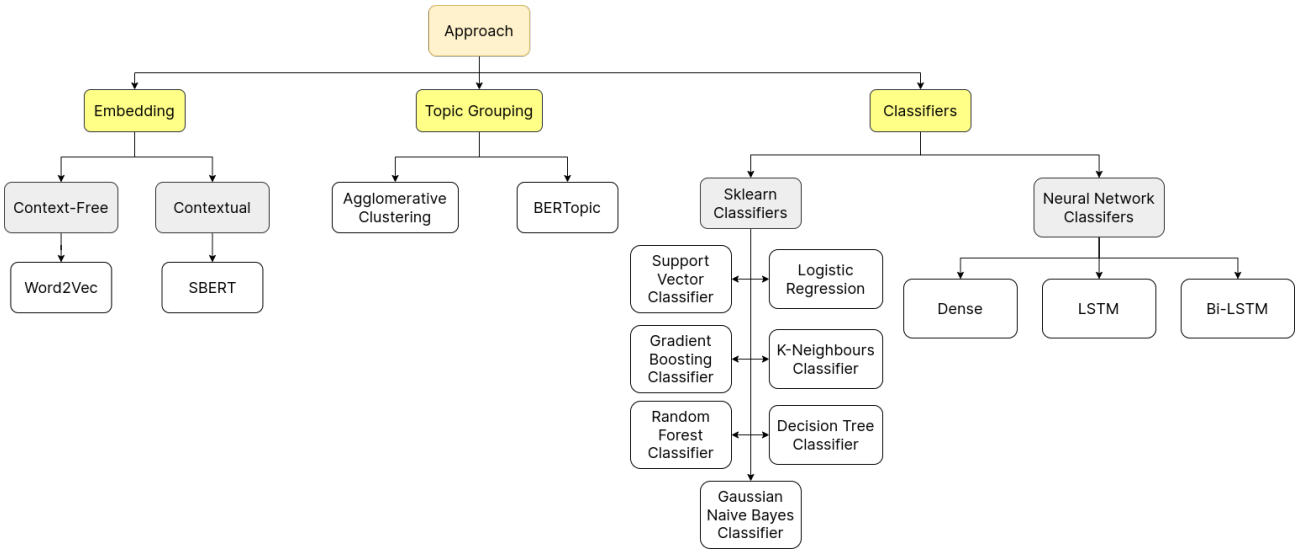


Figure 6.8: Implemented models

We used Google Colaboratory to build the machine learning models shown in the Figure 6.8. The flow of the model building is shown in the Figure 6.9

Figure 6.9: Procedure of model training and evaluation

Step 1: Preprocessing

We preprocessed the data of the VAST dataset. The steps involved in the preprocessing are - lowercasing, tokenization, lemmatization, and removal of specific stopwords in case of context-free embedding approach.

Step 2: Embedding

We have used 2 methods for generating word embeddings: Context-free (Word2vec) and Contextual(SBERT).

Word2vec is a word embedding method which is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. One

drawback with Word2vec is that it produces the same embedding for a word irrespective of the context in which it appears, thus called context-free.

SBERT also known as SentenceBERT, is a contextual word embedding method i.e. it generates embedding for each word depending upon the context in which that word is used in a particular sentence. SBERT is an extension of BERT. BERT makes up the base of this model, to which a pooling layer has been appended. This pooling layer enables us to create a fixed-size representation for input sentences of varying lengths.

Step 3: Target generalization

Since the dataset has few-shot (very small number of examples per target), and zero-shot (examples completely unseen by the model before), it becomes necessary to make the most of the target information we have. To extract more information from these targets, we are performing target grouping which also handles the problem of few examples per target and unseen examples.

We have used 2 target grouping approaches:

Agglomerative Clustering is a clustering algorithm which works in a "bottom-up" manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root)

BERTopic is a topic modeling technique that leverages transformers and c-TF-IDF to create dense clusters allowing for easily interpretable topics whilst keeping important words in the topic descriptions. It generates document embedding with pre-trained transformer-based language models, clusters these embeddings, and finally, generates topic representations.

The text, generalized target and stance labels are then provided as input to the machine learning models.

Step 4: ML models

For training the data, we have used sklearn classifiers as well as created neural network models using different combinations of Dense, LSTM and Bi-LSTM layers.

Following Sklearn Models have been used:

1. **Gradient Boosting** is machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.
2. **Support Vector Machine** is a supervised algorithm which creates the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
3. **Random Forest** is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time.
4. **Decision Tree** is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
5. **K-Nearest Neighbours** is a non-parametric algorithm which assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
6. **Logistic Regression** is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.
7. **Gaussian Naive Bayes** is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

Step 5: Evaluation

We have tried different combinations (total 60) of embedding, target grouping methods and machine learning models. The documented results for these models are as follows:

Stance Label	Best Performing Model	Confidence
Against	Context-free Bertopic gradient boosting classifier	0.51
Pro	Context-free no grouping gradient boosting classifier	0.6
Neutral	Context-free no grouping gradient boosting classifier	0.89
Overall	Context-free no grouping gradient boosting classifier	0.62

Figure 6.10: Best Performing Models

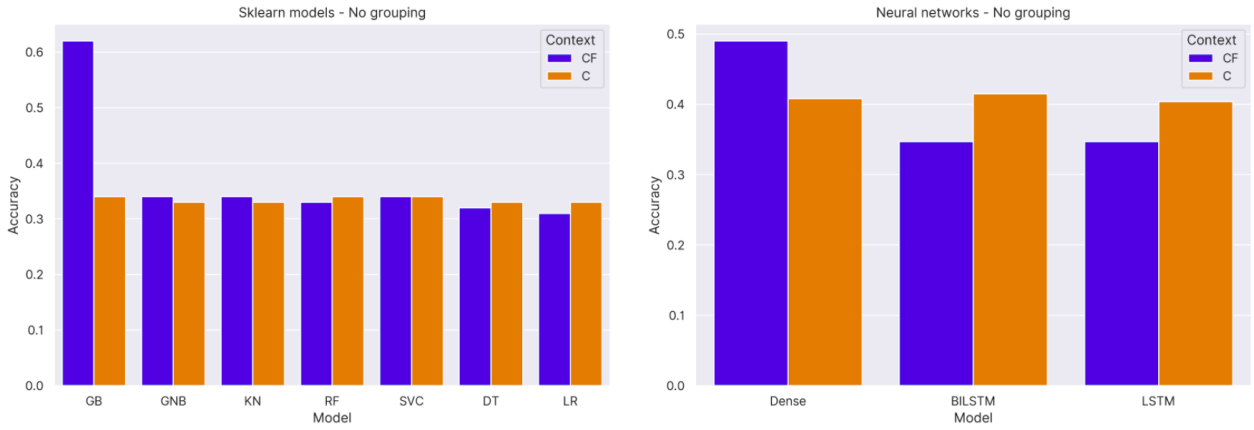


Figure 6.11: Contextual Vs Context-Free No Grouping

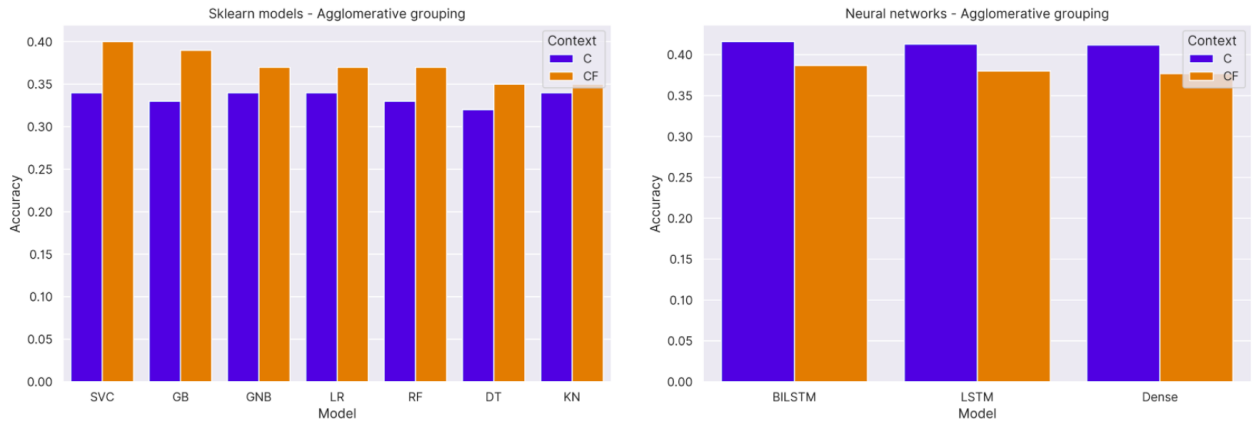


Figure 6.12: Contextual Vs Context-Free Agglomerative Grouping

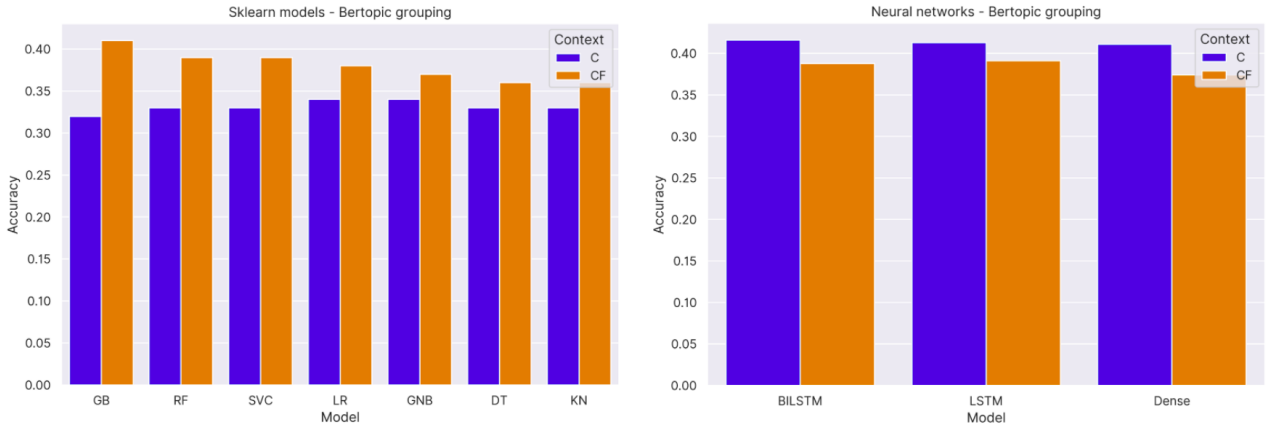


Figure 6.13: Contextual Vs Context-Free BERTopic Grouping

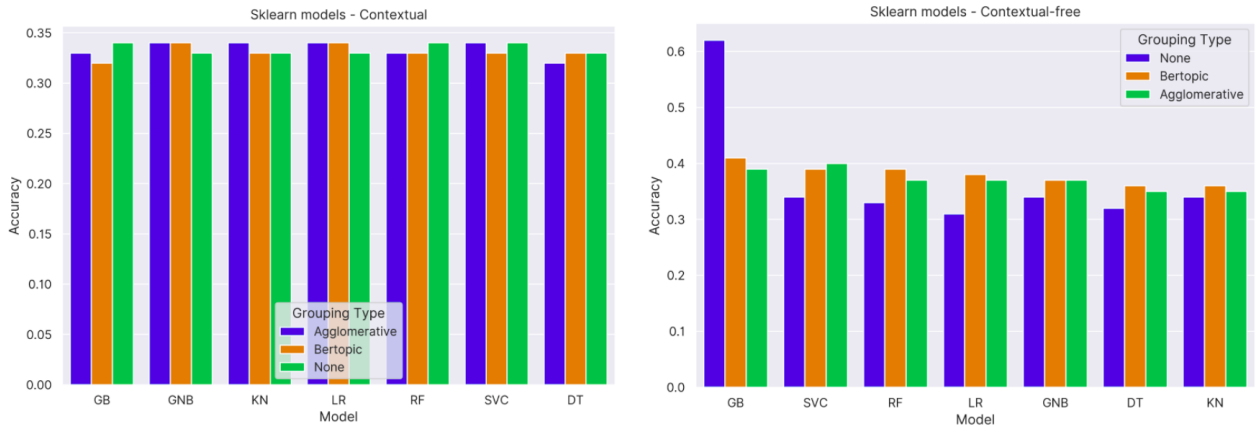


Figure 6.14: Comparison of grouping types: Sklearn models

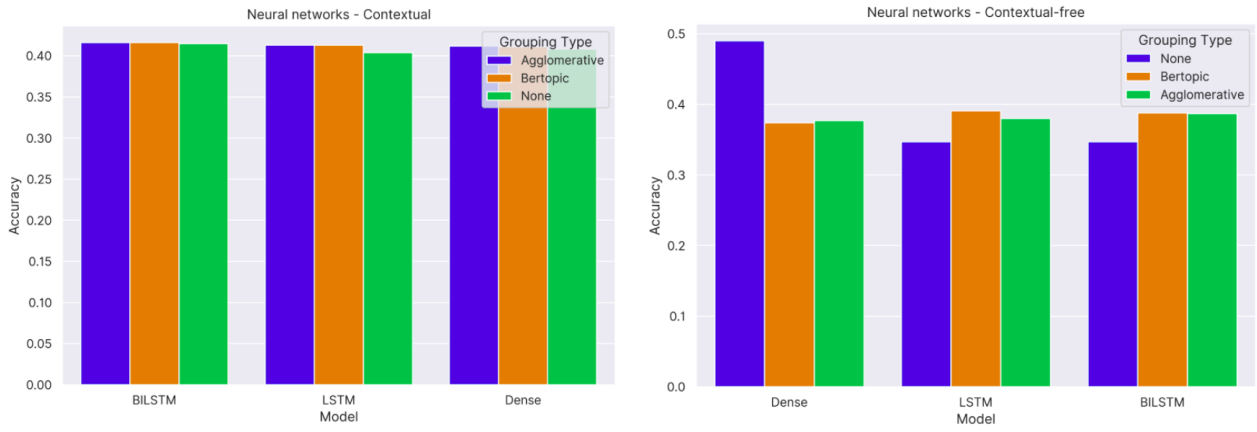


Figure 6.15: Comparison of grouping types: TensorFlow Neural Networks

Results observed indicate that :

1. Context-free no grouping Gradient Boosting Classifier model was the best performing model overall.
2. Comparison of contextual vs context-free embeddings shows that context-free overall performed better than contextual for sklearn models, whereas neural network contextual models showed slightly better results than context-free models.
3. Comparison of different grouping types showed that for context free models, Grouping performed better than no-grouping, whereas for contextual models, the effect of grouping was not much seen.

Checking VAST dataset usefulness for Sentiment Analysis

Since the results from stance detection were not satisfactory, we wanted to check whether the VAST dataset is more suitable for other NLP tasks such as Sentiment Analysis. Since the labels available with us were for stance detection and not sentiment analysis, we generated sentiment results using the following approach:

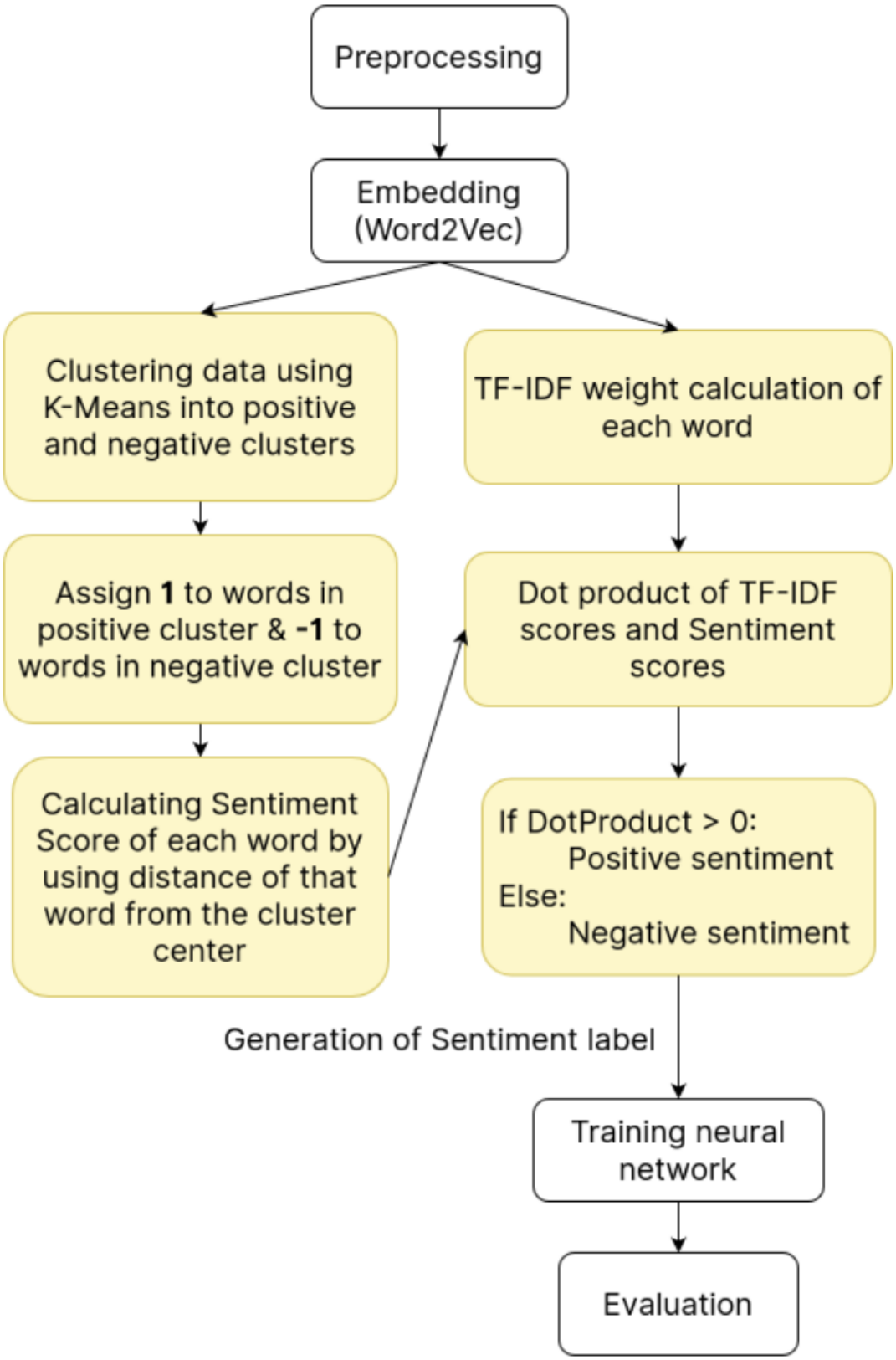


Figure 6.16: Sentiment Analysis Approach

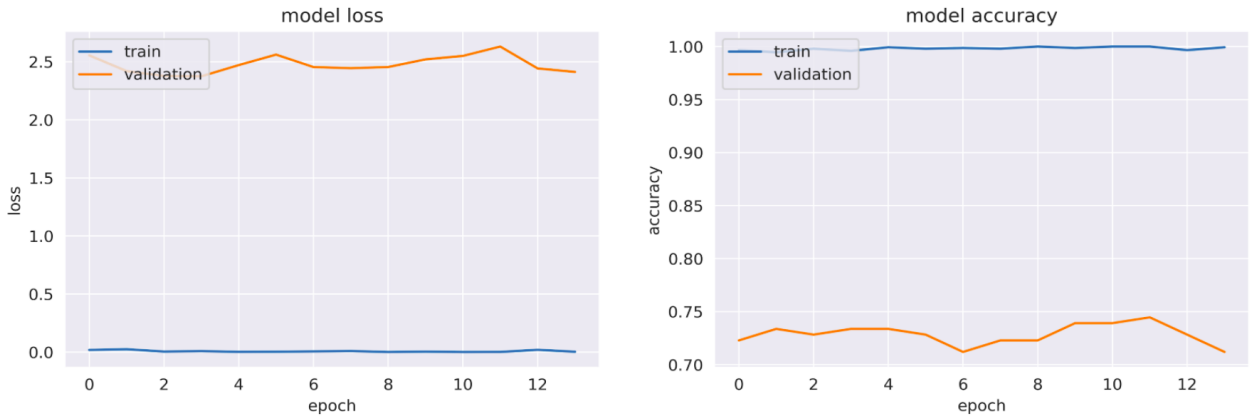


Figure 6.17: Sentiment Analysis Performance Plots

	precision	recall	f1-score	support
0	0.77	0.86	0.81	127
1	0.59	0.45	0.51	58
accuracy			0.73	185
macro avg	0.68	0.65	0.66	185
weighted avg	0.72	0.73	0.72	185

Test Set Results

	precision	recall	f1-score	support
0	0.72	0.90	0.80	107
1	0.78	0.51	0.61	77
accuracy			0.73	184
macro avg	0.75	0.70	0.71	184
weighted avg	0.74	0.73	0.72	184

Validation Set Results

Figure 6.18: Sentiment Analysis Results

We observed much better results for sentiment analysis on same dataset which indicates that the dataset could have been better used for sentiment analysis.

6.3.3 Sqlite Database

Sqlite Database stores the feedback information provided by the user. This includes user's email, optional message, predicted stance and correct stance according to the user. This feedback information will then be verified by developer for relevance and if found relevant, will be incorporated into the model.

Chapter 7

SOFTWARE TEST DOCUMENT

This chapter presents the overall testing approach, tools required for testing, features to be and not to be tested, test plan and results.

7.1 Introduction

This chapter documents and tracks the necessary information required to effectively define the approach to be used in the testing of the project's product. The Test Plan document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and testing team.

7.1.1 System Overview

Introduction of social media platforms has allowed people to express their opinions freely on variety of topics. Some topics tend to be controversial and garners a spectrum of opinions. Understanding the stance of people on these topics and messages becomes crucial to understand the general opinion of public which can be used for further studies and analysis. Our project aims at implementing a platform that allows for the detection of stance from text. To implement this, we will be using stance detection models which take the input, preprocess it in the format suitable, extract the relevant features and deliver the predictions. Furthermore, we will be comparing the results of different models to find the best performing one. Along with that, we will be providing a user interface so that users can interact with the trained models for detection of stance.

7.1.2 Test Approach

1. **Unit Testing:** Unit Testing is a method of testing that verifies the individual units of source code are working properly. It is a major validation effort performed on the smallest module of the system. Unit testing is a basic level of testing which cannot be overlooked, and confirms the behaviour of a single module according to its functional specifications.

2. **Integration Testing:** Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

3. **Acceptance Testing:** Acceptance Testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.

4. **Usability testing:**The Data must be correct. The user interface should be easily accessible, user-friendly and content must be correct.

5. **Invariance Testing:**Invariance tests allow us to describe a set of perturbations we should be able to make to the input without affecting the model's output. We can use these perturbations to produce pairs of input examples (original and perturbed) and check for consistency in the model predictions.

6. **Directional Expectation Testing:** Directional expectation tests, on the other hand, allow us to define a set of perturbations to the input which should have a predictable effect on the model output.

7.2 Introduction

7.2.1 Features to be tested

Feature	Corresponding Test Cases
User Interface testing 1. Testing input interface(target, opinion, algorithm droplist, feedback) 2. Testing results display 3. Testing Layout	TC-1, TC-2, TC-3, TC-4
Pre-training tests 1. Checking the consistency of features and target variable and ensuring they are in appropriate format and range in the dataset. 2. Checking for redundant and missing values in dataset. 3. Checking the shape of model output and ensure it aligns with input features 4. checking the output ranges and ensure it aligns with our expectations 5. Checking for test set leakage i.e. duplicate rows in train/test splits by concatenating train and test data, dropping duplicates, and checking the number of rows	TC-5, TC-6, TC-7, TC-9, TC-9
Testing Model for Stance Detection (Post-train tests) 1. Testing Model results compared to actual output(will also include invariance and directional expectation testing) 2. Testing Hyper-parameters	TC-10, TC-11, TC-12, TC-13

Feedback	TC-14

7.2.2 Features not to be tested

7.2.3 Testing Tools and Environment

Requirements:

1. Hardware: Windows/Linux/Mac Laptop or Desktop with minimum 4GB RAM
2. Software:
   - Drivers
     (a) Chrome Driver
     (b) Mozilla -GeckoDriver
     (c) Internet Explorer Driver
   - Web Browsers
     (a) Chrome
     (b) Internet Explorer
     (c) Firefox
     (d) Safari
3. Tools : Selenium IDE for automated Interface testing. Python modules for testing of models.

7.3 Test Cases

7.3.1 Interface Testing

7.3.1.1 Test Case TC-1

Purpose	Input	Expected Output	Test Procedure	Result
To test if the system accepts inputs in correct format for target, opinion, and algorithm	Target, Opinion, Algorithm	Results should be displayed for specific input without any error message	Validation Tests will be done using Selenium. Correct inputs will be provided and output will be checked accordingly	Pass

7.3.1.2 Test Case TC-2

Purpose	Input	Expected Output	Test Procedure	Result
To test if the system rejects inputs in in-correct format for target, opinion, and algorithm with respective error message being displayed	Incorrect Target, Opinion, Algorithm	Error message informing why the input is incorrect	Validation Tests will be done using Selenium. Incorrect inputs will be provided and output will be checked accordingly	Pass

7.3.1.3 Test Case TC-3

Purpose	Input	Expected Output	Test Procedure	Result
To test if the system displays the results in appropriate format which will include the detected stance, model information, and option for feedback	Target, Opinion, Algorithm	Results in appropriate format which will include the detected stance, model information, and option for feedback	Validation Testing will be done using Selenium.	Pass

7.3.1.4 Test Case TC-4

Purpose	Input	Expected Output	Test Procedure	Result
To test if the layout of Interface includes the essential features specified in requirements and maintains the essence	UI designs and interface	User Interface which meets the functional requirements and follows the essence of designed layout	This has to be tested manually.	Pass

7.3.2 Testing Data(pre-training tests)

7.3.2.1 Test Case TC-5

Purpose	Input	Expected Output	Test Procedure	Result
To check the consistency of features and target variable and ensuring they are in appropriate format and range in the dataset.	Dataset features and target	Features and target which are in appropriate format and range	Manually testing individual features and target for format consistency and within limit range	Pass

7.3.2.2 Test Case TC-6

Purpose	Input	Expected Output	Test Procedure	Result
To check for redundant and missing values in dataset.	Dataset features and target	Dataset with no missing and redundant values	1. Testing individual features and target for redundant and missing values with the help of python modules	Pass

7.3.2.3 Test Case TC-7

Purpose	Input	Expected Output	Test Procedure	Result
To check the shape of model output and ensure it aligns with input features	Dataset and model output	Pass if the shape matches else Fail	1. Testing shape to be done with the help of python modules	Pass

7.3.2.4 Test Case TC-8

Purpose	Input	Expected Output	Test Procedure	Result
To check for test set leakage i.e. duplicate rows in train/test splits by concatenating train and test data, dropping duplicates, and checking the number of rows	Train test split data	Pass if no duplicate rows in split data else Fail	1. This can be done by concatenating train and test data, dropping duplicates, and checking the number of rows	Pass

7.3.3 Testing Model for Stance Detection (post-train tests)

7.3.3.1 Test Case TC-9

Purpose	Input	Expected Output	Test Procedure	Result
To check the model output ranges and ensure it aligns with our expectations	Dataset and model output	Pass if the output ranges align with our expectations else Fail	1. Testing to be done with the help of python modules	Pass

7.3.3.2 Test Case TC-10

Purpose	Input	Expected Output	Test Procedure	Result
To test if the model predicts the stance and the predicted stance is either 'favor', 'against' or 'neutral'	Model output i.e predicted stance	Pass if model predicts without any error and belongs to one of the 3 classes else Fail	1. This testing will be done using python machine learning libraries	Pass

7.3.3.3 Test Case TC-11

Purpose	Input	Expected Output	Test Procedure	Result
To test for invariance in model predictions i.e introducing small set or perturbances to the input which should not affect the model's output	Original and Perturbed input examples	Model output should be the same for both examples	1. Generate a set of pairs of original and perturbed input examples 2. Run each of the pairs as input to the model to test invariance	Invariance Testing results showed that some models showed passing the test for some sentences whereas some models failing the test for some sentences. The test results section contains examples for passed as well as failed test case

7.3.3.4 Test Case TC-12

Purpose	Input	Expected Output	Test Procedure	Result
To perform directional expectation testing i.e introducing small set or perturbances to the input which should have a predictable effect on the model's output	Original and Perturbed input examples	Model's output for the perturbed example should be as predicted	1. Generate a set of pairs of original and perturbed input examples 2. Run each of the pairs as input to the model	Directional Expectation Testing results showed that some models showed passing the test for some sentences whereas some models failing the test for some sentences. The test results section contains examples for passed as well as failed test case

7.3.3.5 Test Case TC-13

Purpose	Input	Expected Output	Test Procedure	Result
To test the same model with hyper parameter tuning	Hyper-parameters	Better performance results than previously trained model on different parameters	1. This module needs to be tested manually by changing hyper-parameters	Pass

7.3.4 Feedback

7.3.4.1 Test Case TC-14

Purpose	Input	Expected Output	Test Procedure	Result
To test if the model's detected stance feedback system works	Feedback on detected stance as provided by user	Feedback is reflected on developer's portal	1. Check developer's portal for the feedback entry	Pass

7.4 Test Logs

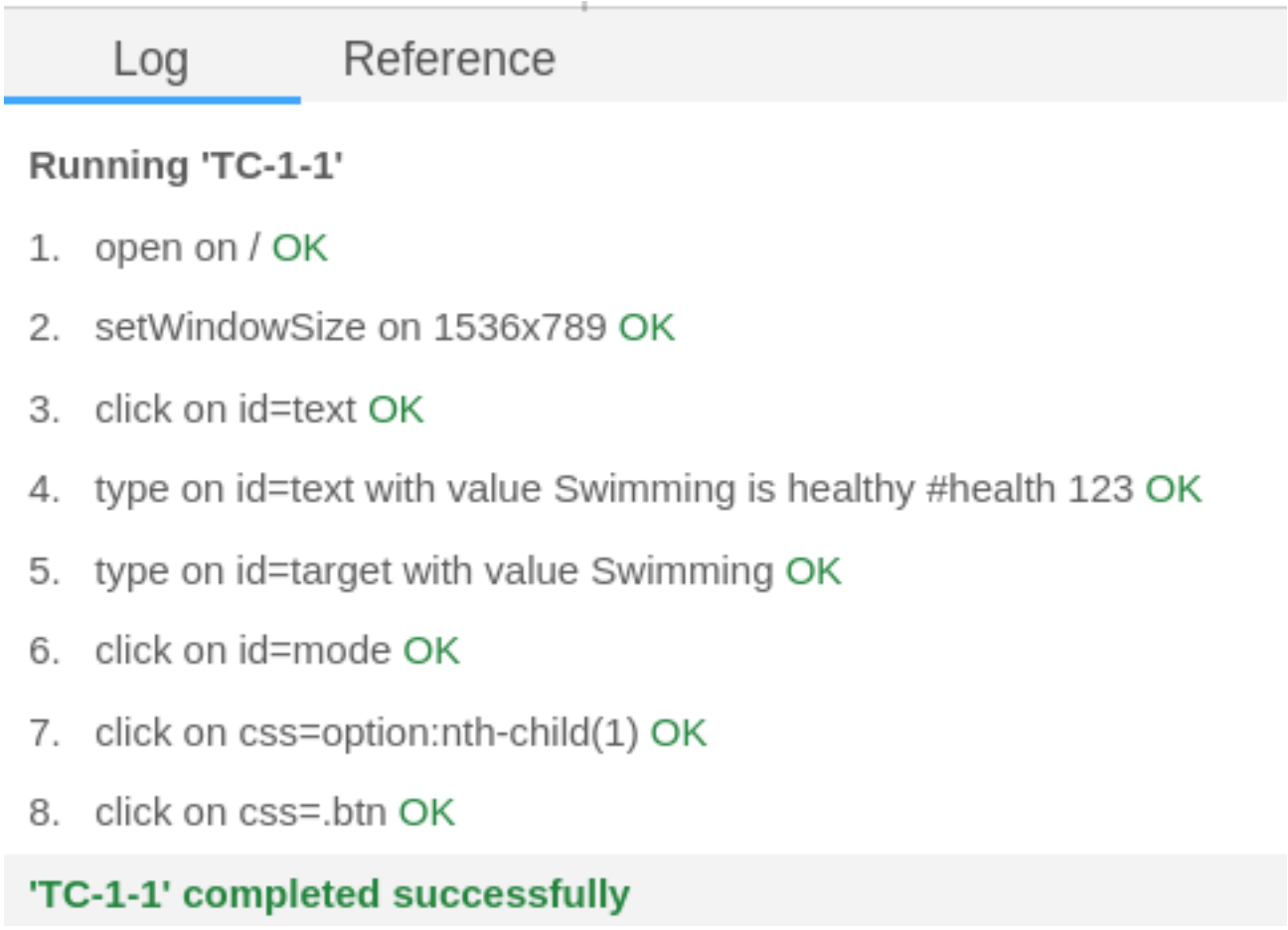


Figure 7.1: TC-1

Running 'TC-1-2'

1. open on / OK
2. setWindowSize on 534x759 OK
3. click on id=mode OK
4. click on id=text OK
5. type on id=text with value Swimming is healthy #health 123 OK
6. type on id=target with value Swimming OK
7. select on id=mode with value label=Manual OK
8. click on id=encoding OK
9. select on id=encoding with value label=Context-Free OK
10. click on css=#encoding > option:nth-child(2) OK
11. click on id=targetGrouping OK
12. click on css=#targetGrouping > option:nth-child(1) OK
13. click on id=modelName OK
14. click on css=#modelName > option:nth-child(1) OK
15. click on css=.btn OK
16. runScript on window.scrollTo(0,128.8000030517578) OK

'TC-1-2' completed successfully

Figure 7.2: TC-1

Running 'TC-1-3'

1. open on / OK
2. setWindowSize on 534x759 OK
3. type on id=text with value Swimming is healthy OK
4. type on id=target with value Swimming OK
5. click on id=mode OK
6. select on id=mode with value label=Manual OK
7. click on css=#mode > option:nth-child(2) OK
8. click on css=.btn OK
9. runScript on window.scrollTo(0,0) OK
10. click on css=.btn-dark OK
11. click on id=email OK
12. type on id=email with value nishavak.n@somaiya.edu OK
13. click on id=expectedStance OK
14. click on css=#expectedStance > option:nth-child(1) OK
15. click on id=feedback OK
16. click on css=.modal-footer > .btn OK

'TC-1-3' completed successfully

Figure 7.3: TC-1

Running 'TC-2-1'

1. open on / OK
2. setWindowSize on 768x789 OK
3. click on id=text OK
4. type on id=text with value 24324 OK
5. type on id=target with value @$$ OK
6. click on id=mode OK
7. select on id=mode with value label=Manual OK
8. click on css=#mode > option:nth-child(2) OK
9. click on id=encoding OK
10. select on id=encoding with value label=Context-Free OK
11. click on css=.btn OK
12. runScript on window.scrollTo(0,0) OK
13. click on id=app OK

'TC-2-1' completed successfully

Figure 7.4: TC-2

Running 'TC-2-2'

1. open on / OK
2. setWindowSize on 534x759 OK
3. click on id=text OK
4. type on id=text with value Swimming is healthy OK
5. type on id=target with value Swimming OK
6. select on id=mode with value label=Manual OK
7. click on css=.btn OK
8. runScript on window.scrollTo(0,0) OK
9. click on css=.btn-dark OK
10. click on id=email OK
11. type on id=email with value sdfnsdijfiodsj OK
12. click on css=.modal-footer > .btn OK

'TC-2-2' completed successfully

Figure 7.5: TC-2

Running 'TC-3'

1. open on / OK
2. setWindowSize on 1536x789 OK
3. click on id=text OK
4. type on id=text with value Swimming is healthy OK
5. type on id=target with value Swimming OK
6. select on id=mode with value label=Manual OK
7. click on id=modelName OK
8. select on id=modelName with value label=K-Nearest Neighbors Classifier OK
9. click on css=option:nth-child(7) OK
10. click on css=.btn OK
11. runScript on window.scrollTo(0,128.8000030517578) OK

'TC-3' completed successfully

Figure 7.6: TC-3

7.5 Test Results

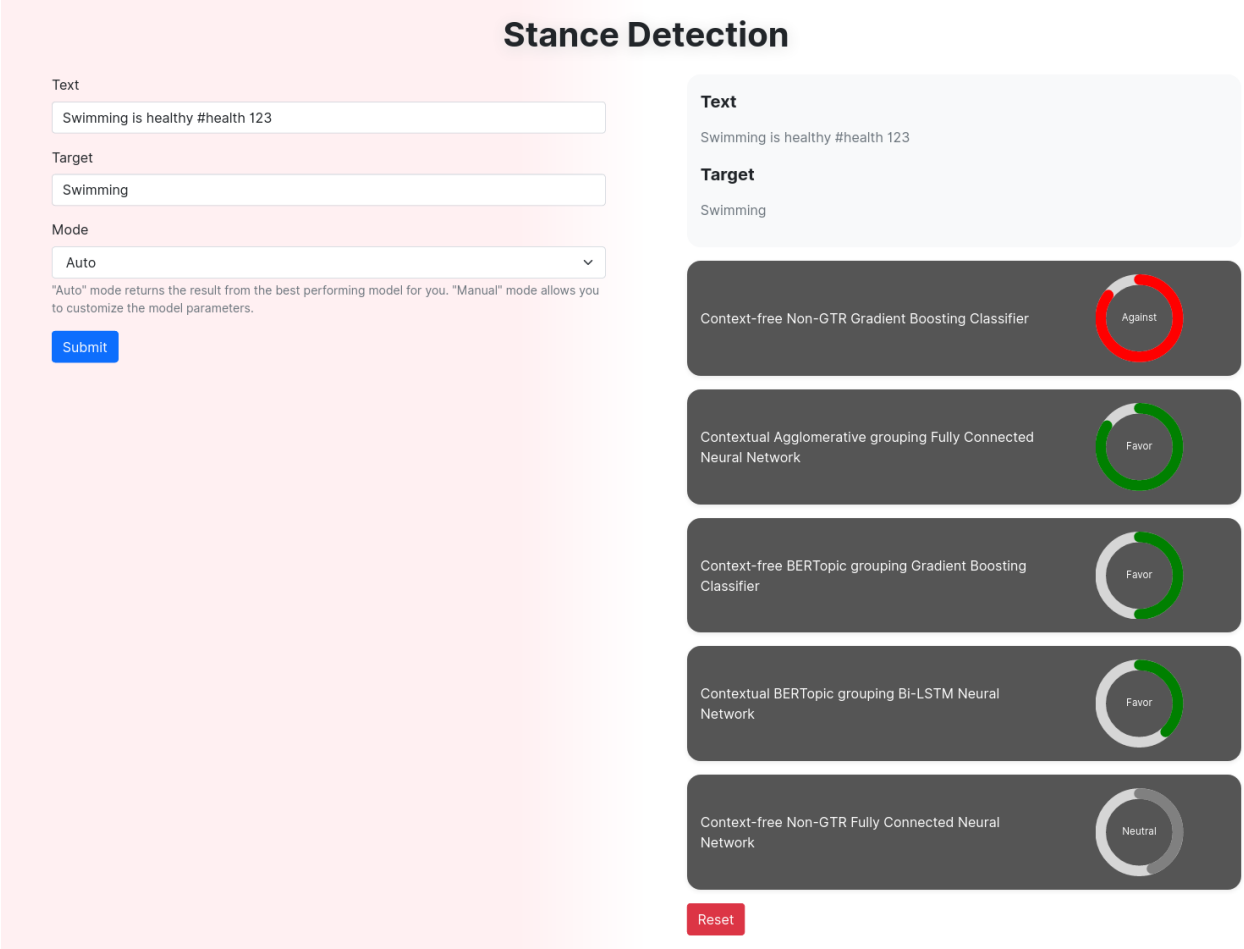


Figure 7.7: TC-1

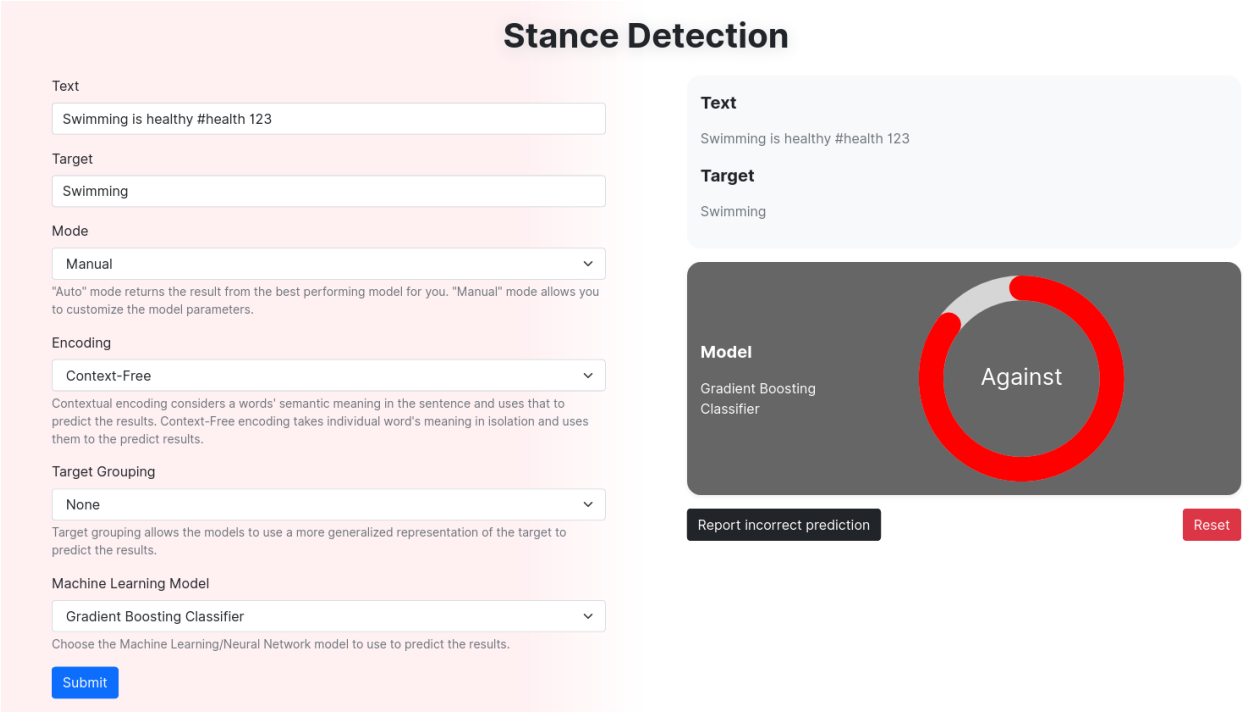


Figure 7.8: TC-1

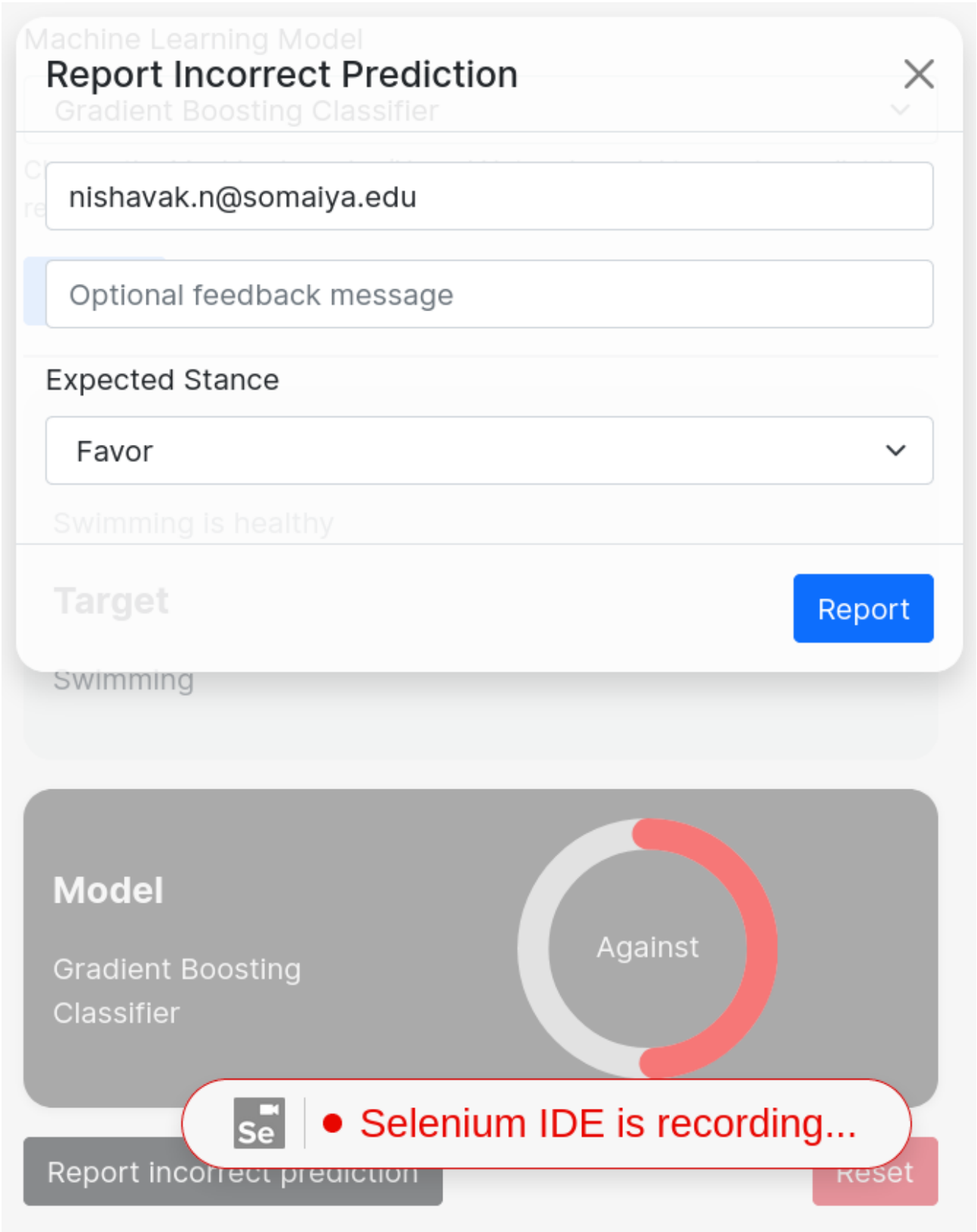


Figure 7.9: TC-1

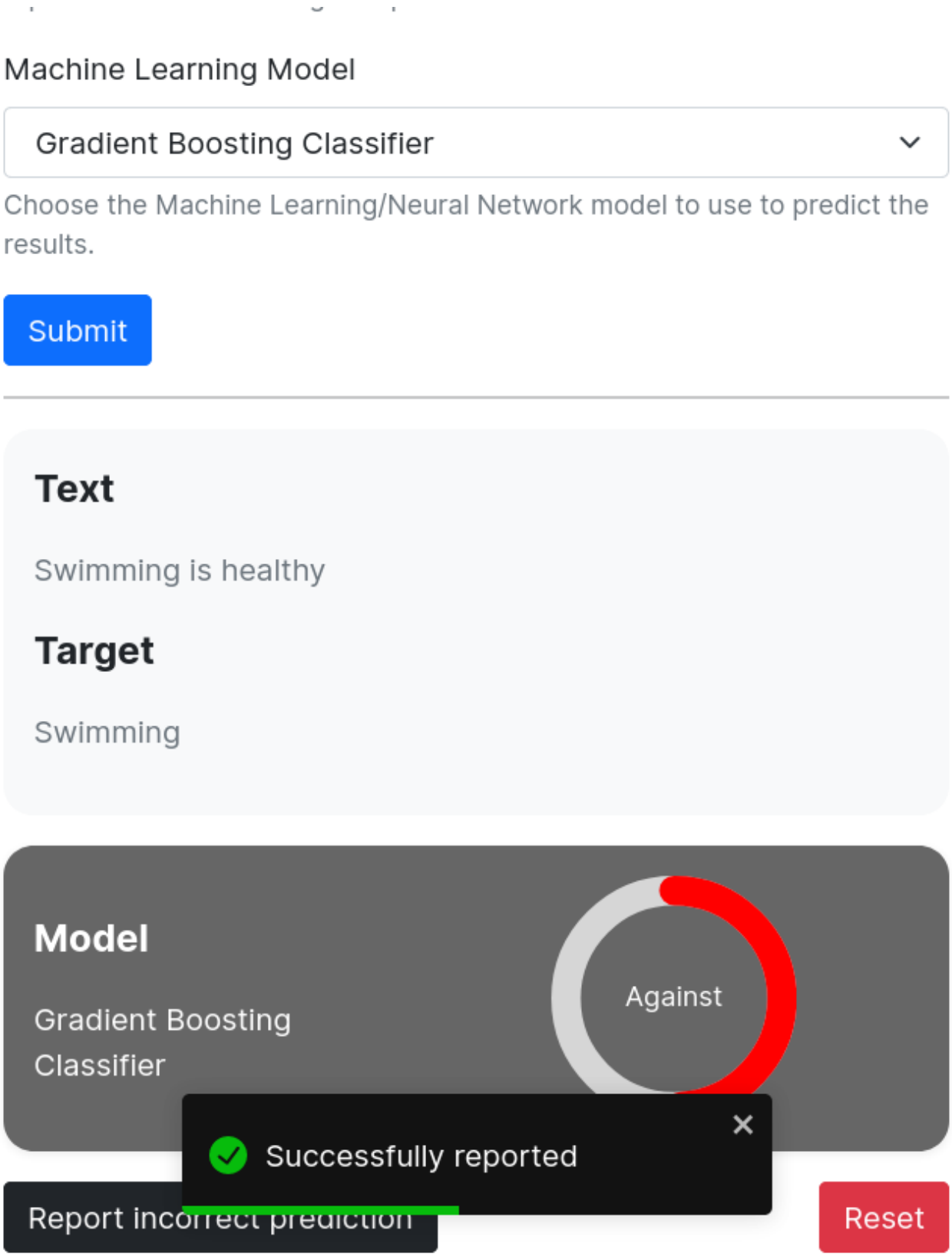


Figure 7.10: TC-1

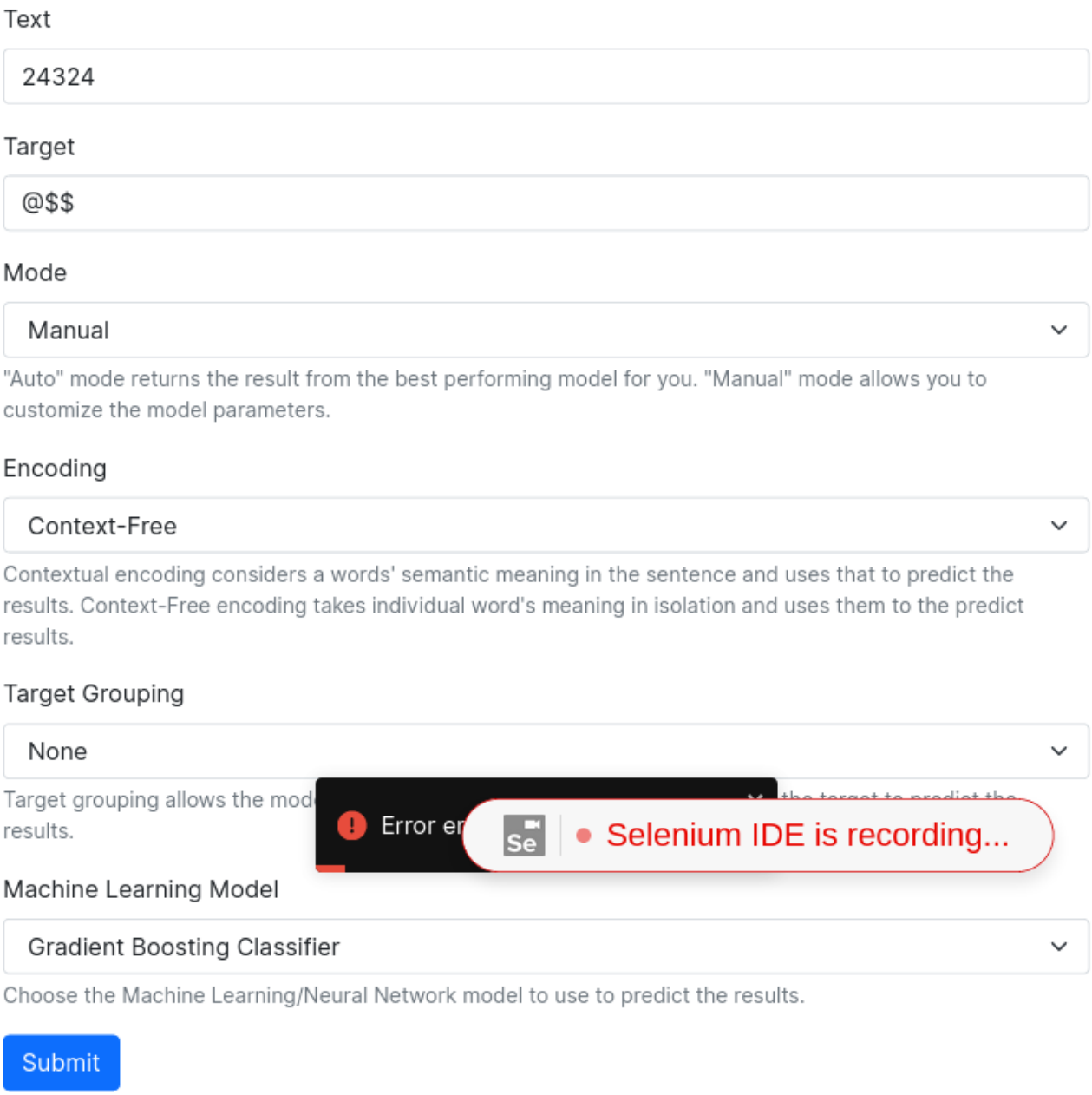


Figure 7.11: TC-2

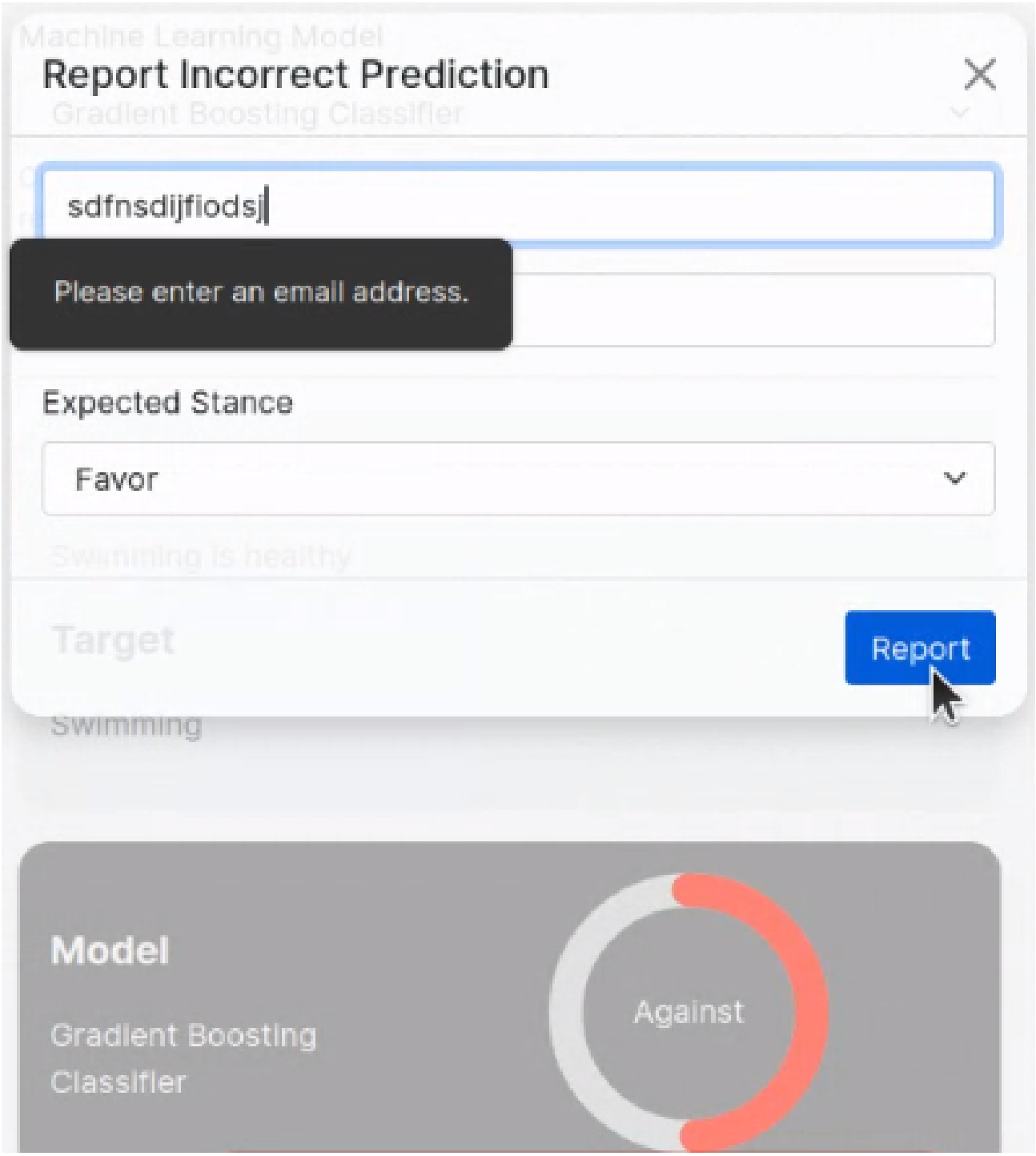


Figure 7.12: TC-2

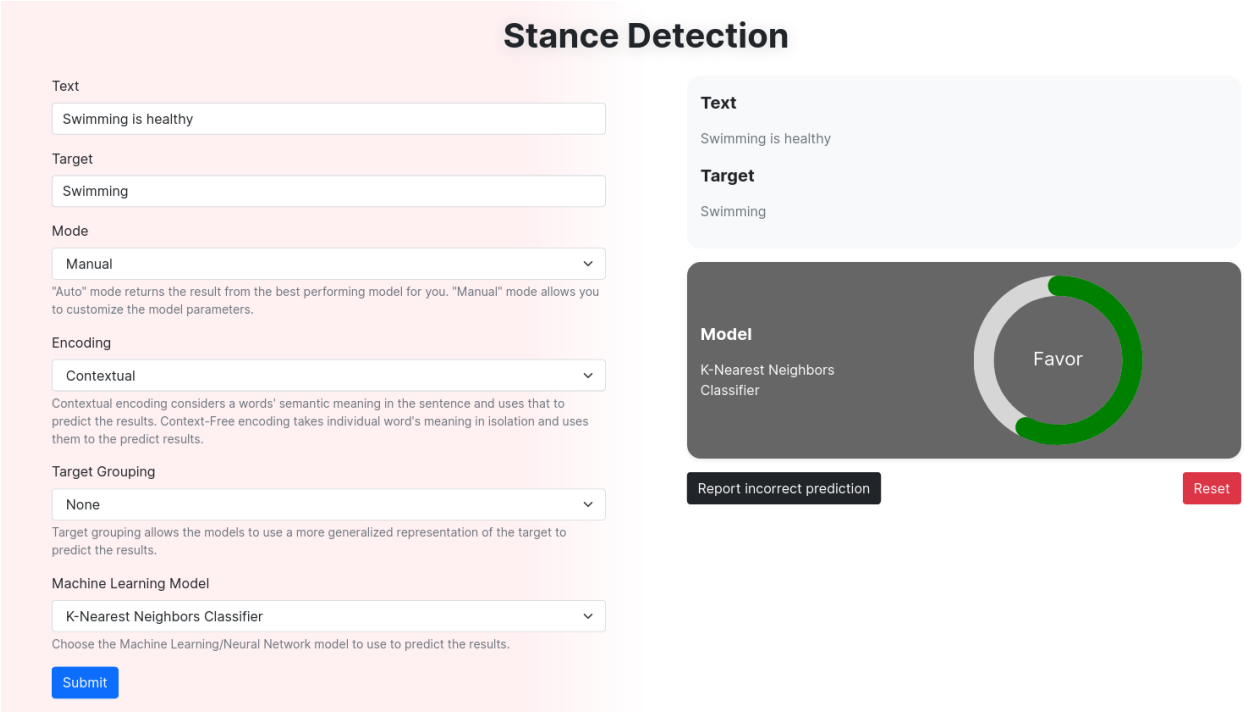


Figure 7.13: TC-3

```
Epoch 1/10
421/421 [==============================] - 268s 627ms/step - loss: 1.0521 - accuracy: 0.4747 - val_loss: 1.0778 - val_accuracy: 0.3882
Epoch 2/10
421/421 [==============================] - 263s 624ms/step - loss: 1.0178 - accuracy: 0.5053 - val_loss: 1.0854 - val_accuracy: 0.3940
Epoch 3/10
421/421 [==============================] - 270s 641ms/step - loss: 1.0024 - accuracy: 0.5134 - val_loss: 1.0929 - val_accuracy: 0.3823
Epoch 4/10
421/421 [==============================] - ETA: 0s - loss: 0.9897 - accuracy: 0.5243Restoring model weights from the end of the best epoch: 1.
421/421 [==============================] - 270s 642ms/step - loss: 0.9897 - accuracy: 0.5243 - val_loss: 1.1011 - val_accuracy: 0.3843
Epoch 4: early stopping
```

Figure 7.14: TC-5 : Model Trained successfully indicating inputs are consistent and within range

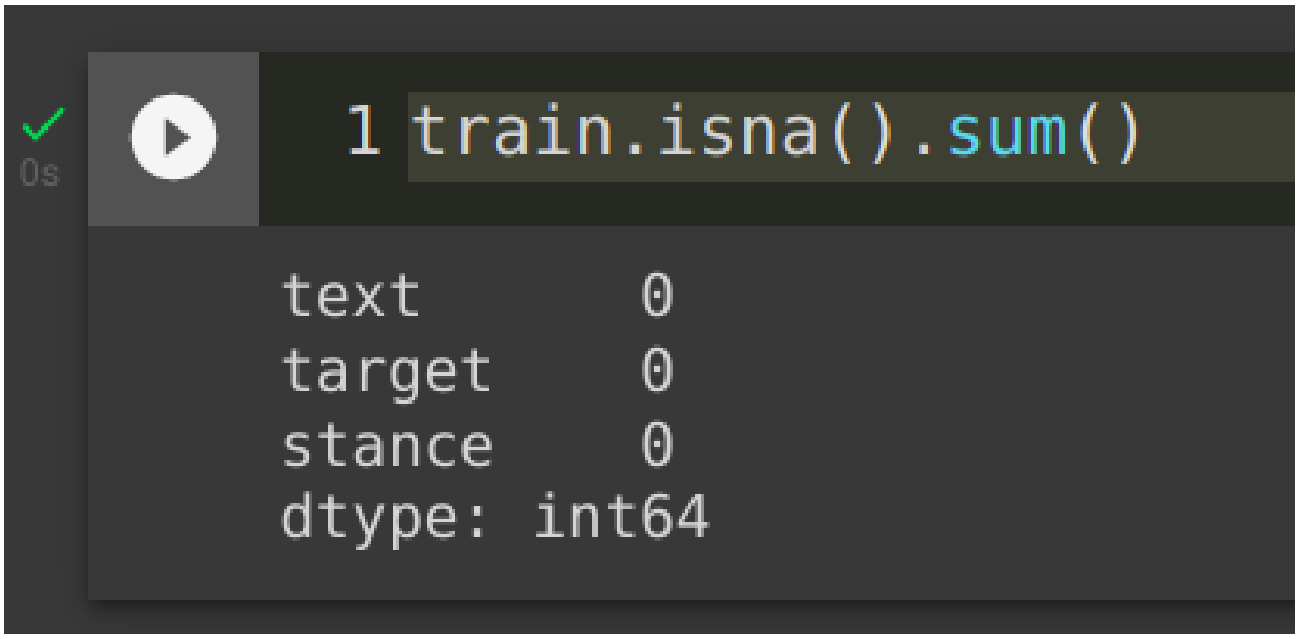


Figure 7.15: TC-6: No null rows

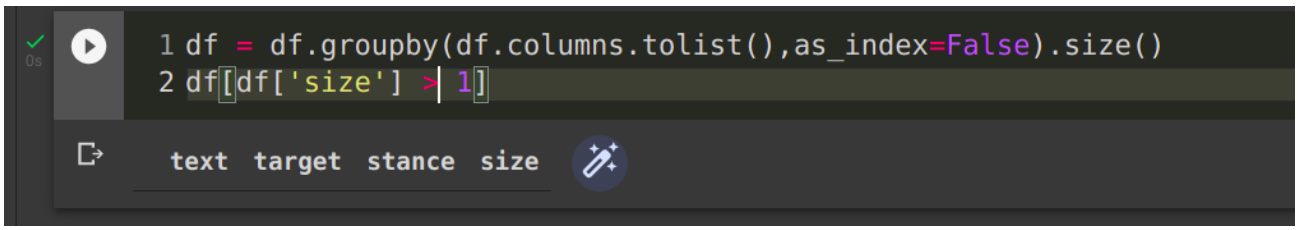


Figure 7.16: TC-6: No duplicate / redundant rows

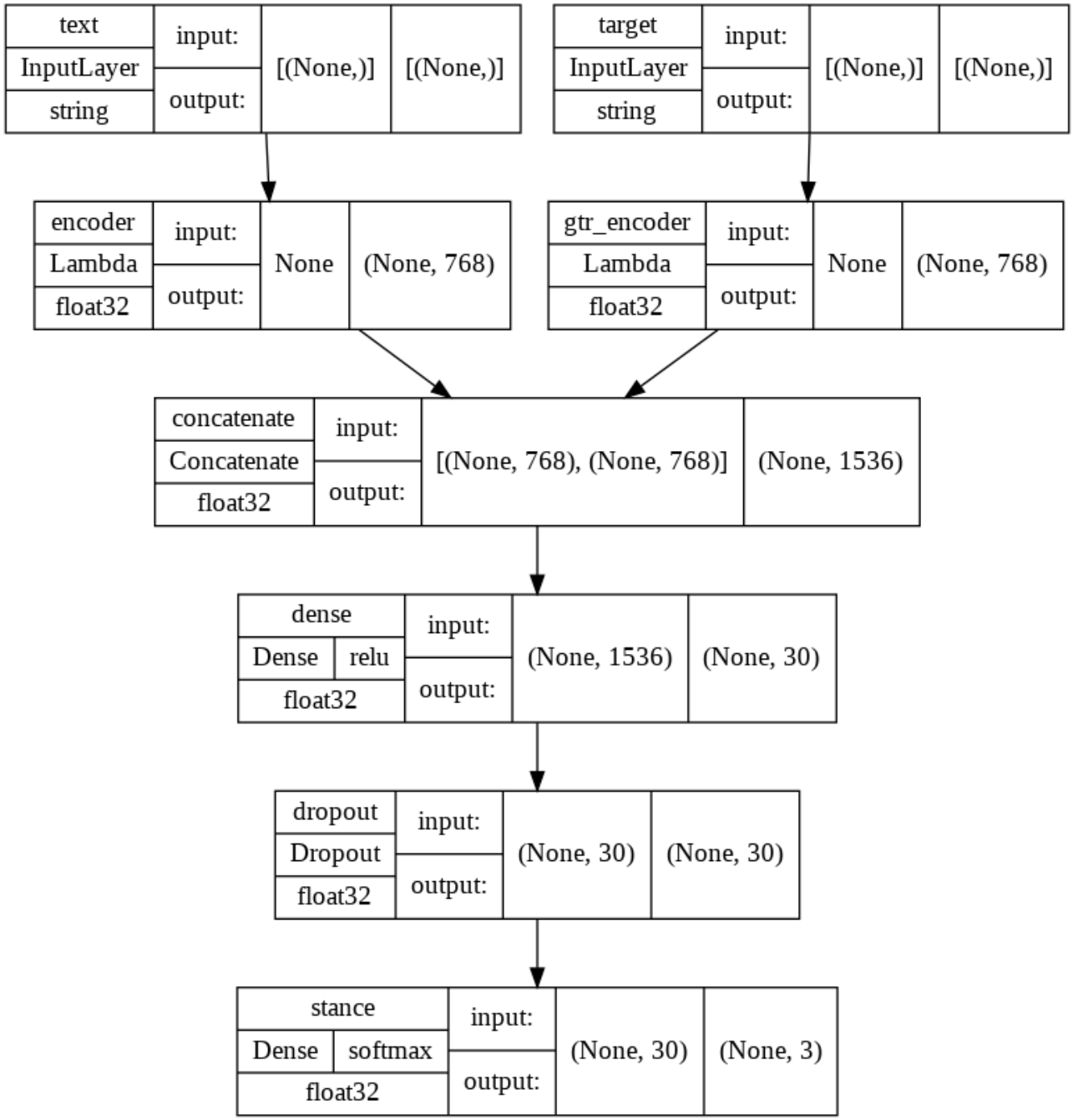


Figure 7.17: TC-7: Model diagram indicates input features are in correct shape

```
1 df = pd.concat([train, test])
2 df = df.groupby(df.columns.tolist(),as_index=False).size()
3 df[df['size'] > 1]
```

text	target	stance	size

Figure 7.18: TC-8: Checking for set leakage by combining training and testing set and seeing if there are any redundant rows

```
1 y_true = np.array(test['stance'])
2 y_pred = stance_detection.predict(Test)
3 y_pred = np.argmax(y_pred, axis=1)
4 print(classification_report(y_true, y_pred, digits=3))
```

	precision	recall	f1-score	support
0	0.437	0.402	0.419	1018
1	0.428	0.615	0.505	944
2	0.364	0.248	0.295	1044
accuracy			0.416	3006
macro avg	0.410	0.422	0.406	3006
weighted avg	0.409	0.416	0.403	3006

Figure 7.19: TC-9: Classification report shows that model outputs are in expected range 0-2

```
212 # models
213 stanceMap = {0: "against", 1: "favor", 2: "neutral"}
214
```

Figure 7.20: TC-10

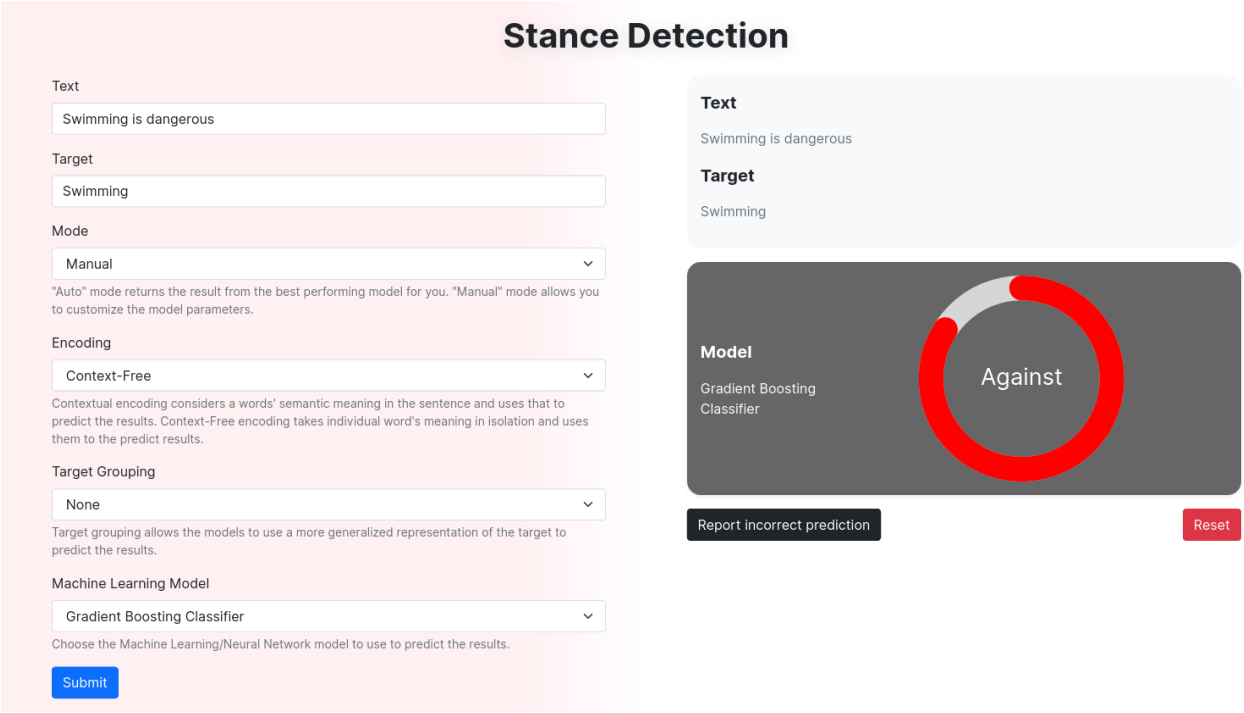


Figure 7.21: TC-11: Invariance Testing Passed Case - original sentence

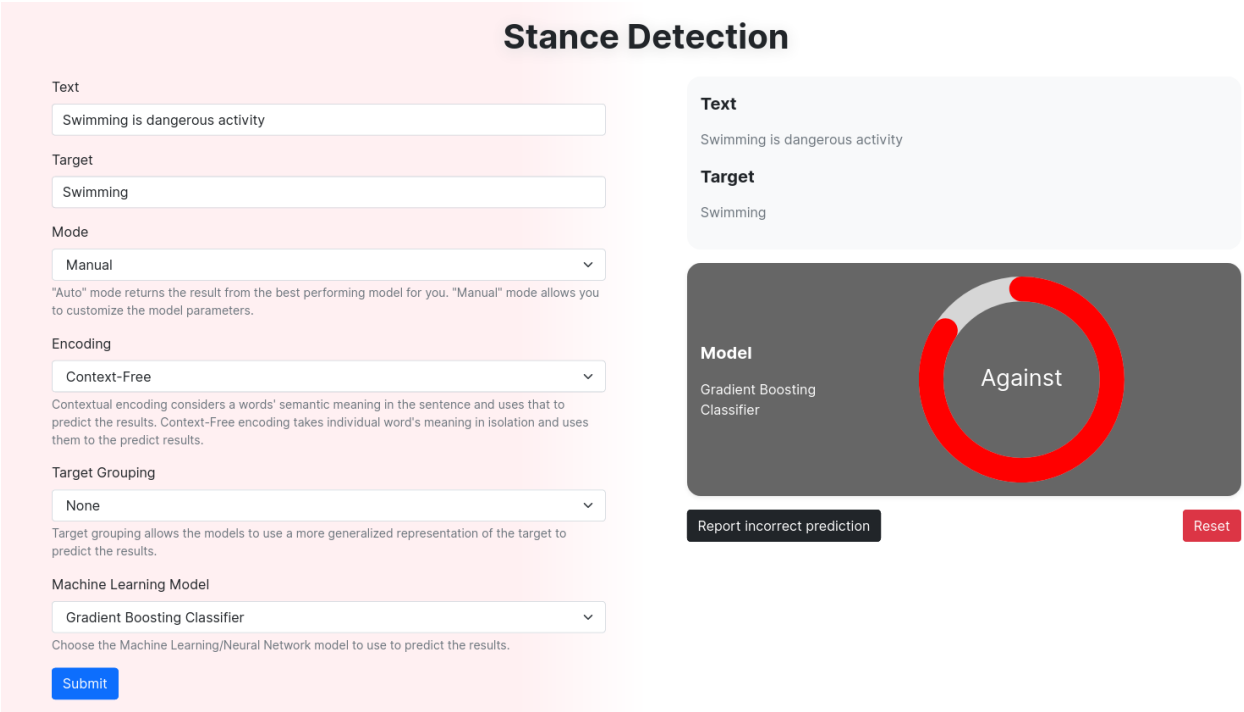


Figure 7.22: TC-11: Invariance Testing Passed Case - perturbed sentence

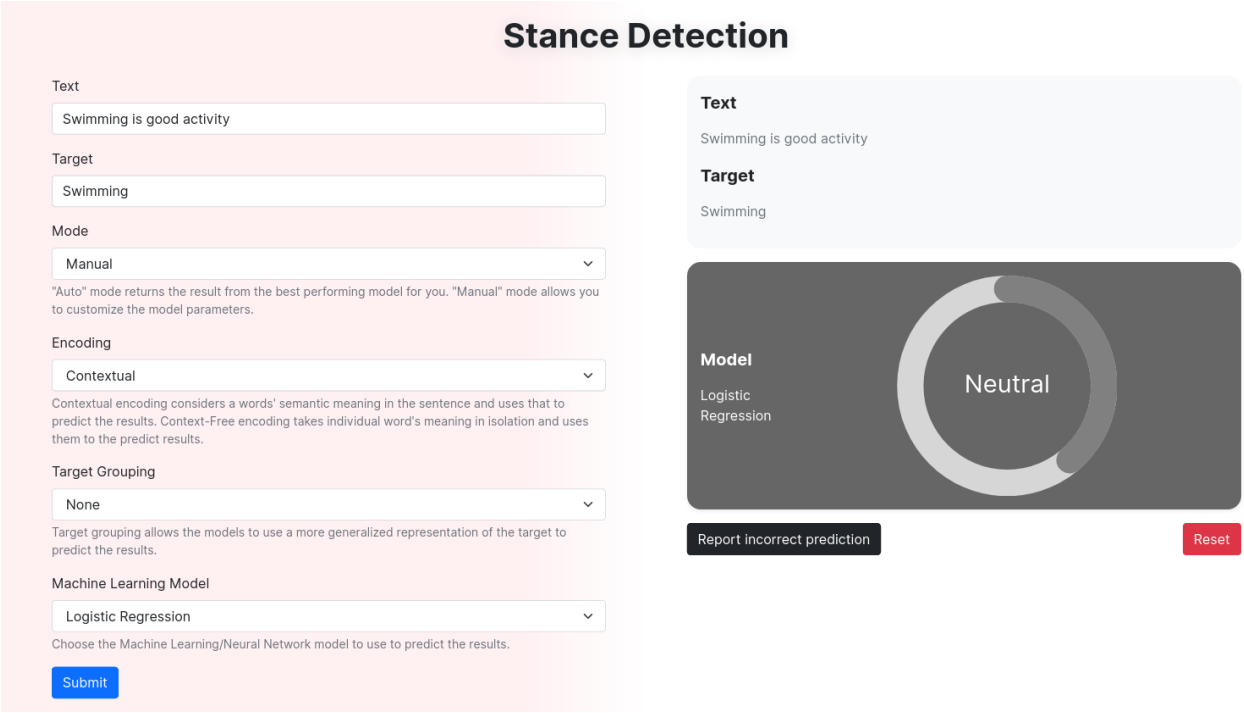


Figure 7.23: TC-11: Invariance Testing Failed Case - original sentence

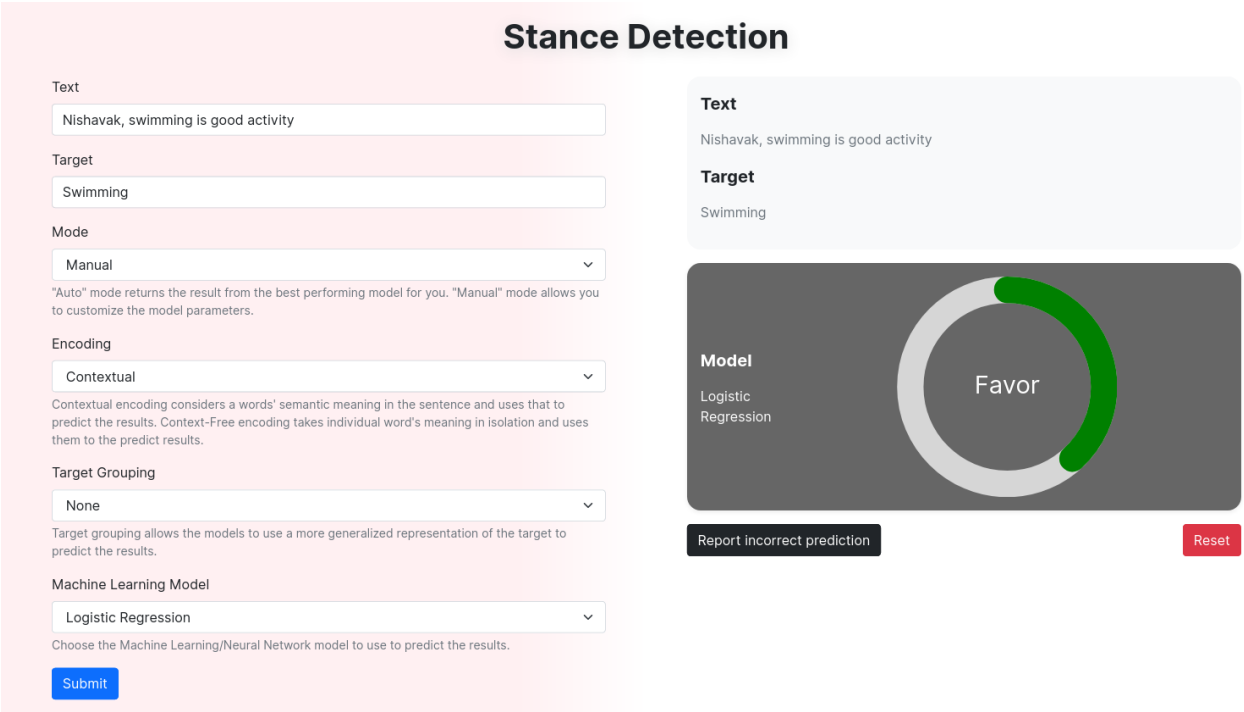


Figure 7.24: TC-11: Invariance Testing Failed Case - perturbed sentence

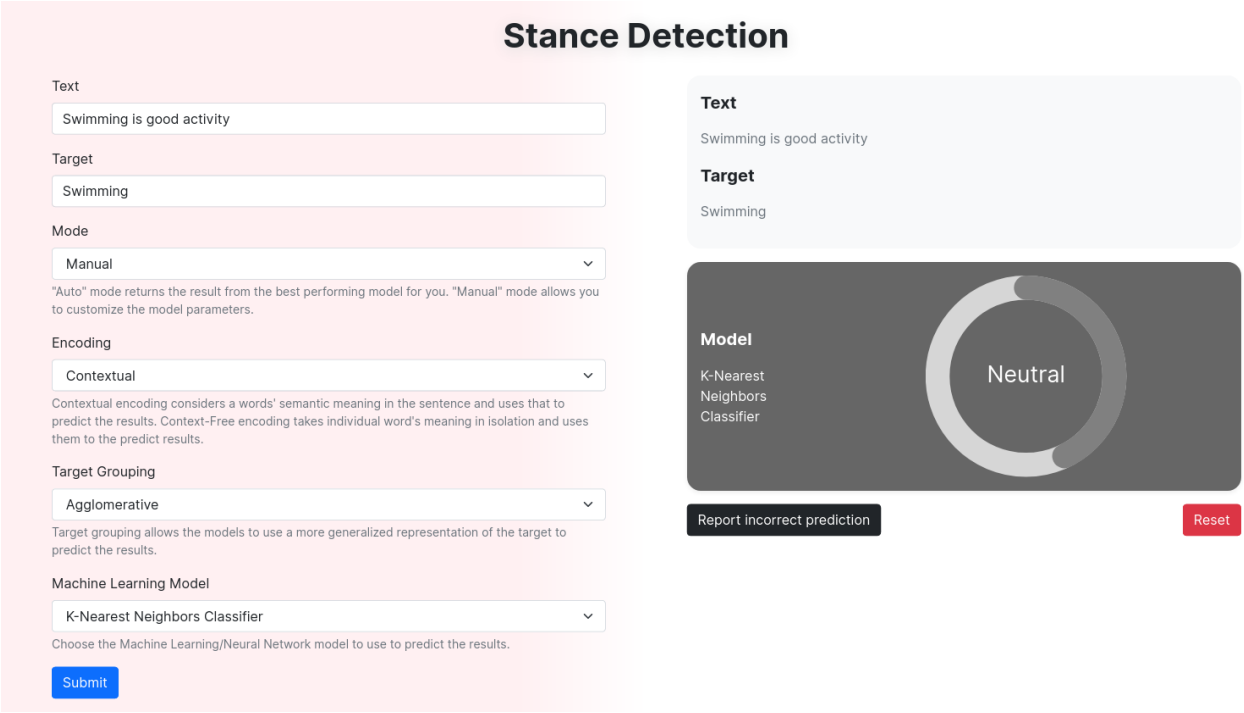


Figure 7.25: TC-12: Directional Expectation Testing Passed Case - original sentence

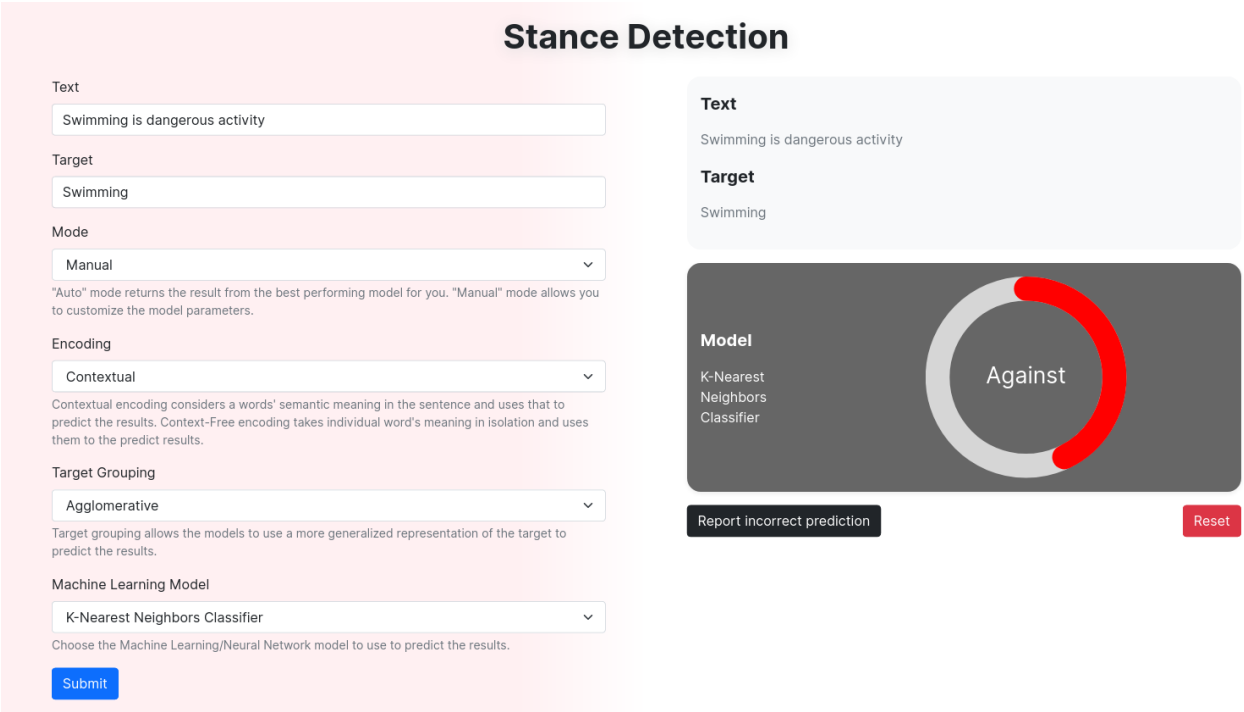


Figure 7.26: TC-12: Directional Expectation Testing Passed Case - perturbed sentence

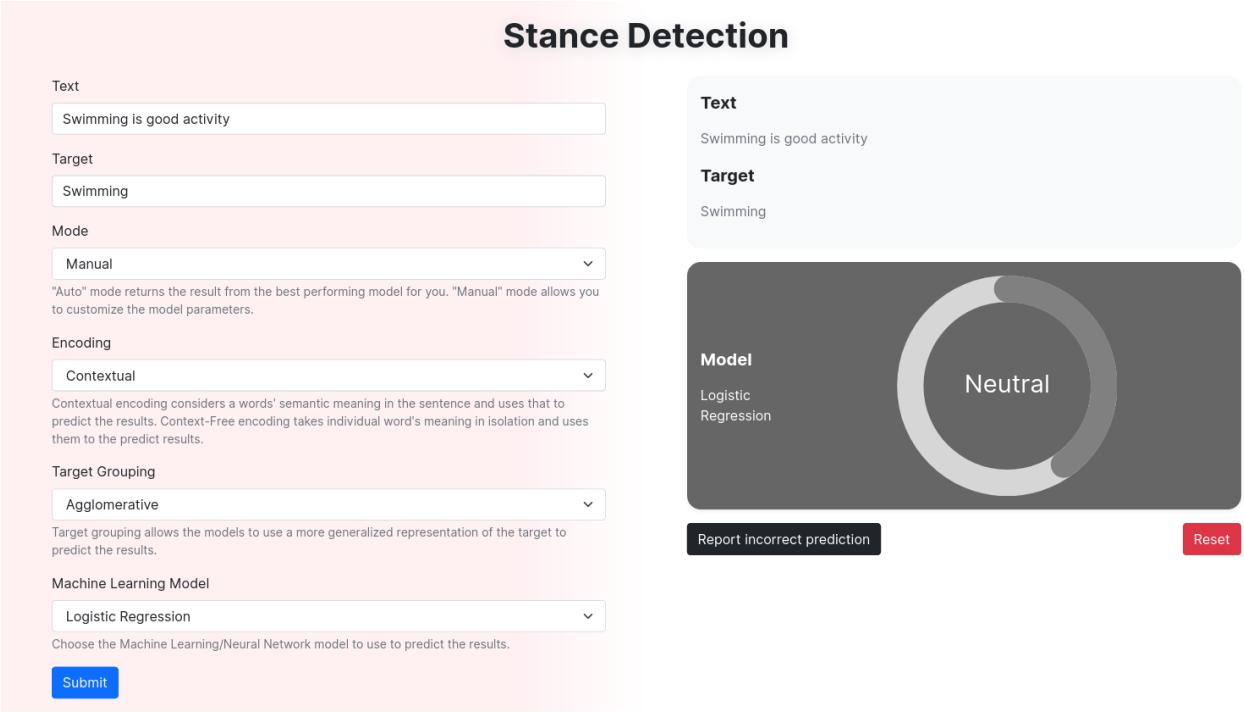


Figure 7.27: TC-12: Directional Expectation Testing Failed Case - original sentence

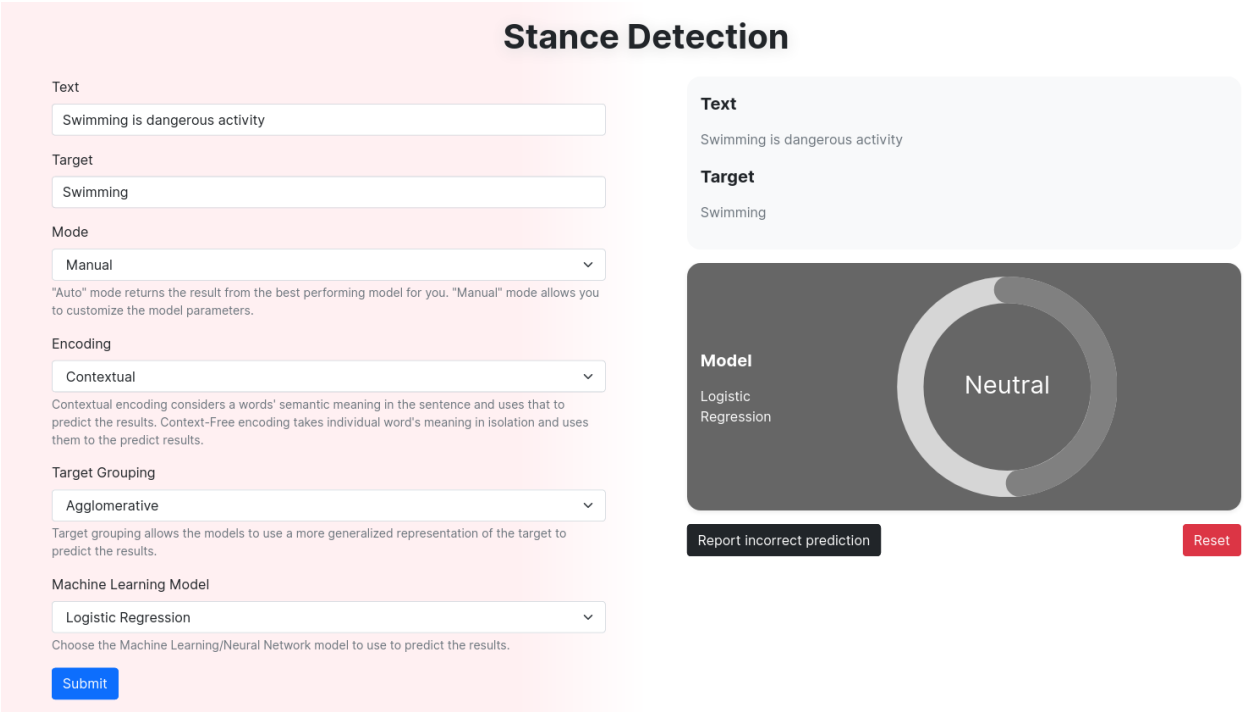


Figure 7.28: TC-12: Directional Expectation Testing Failed Case - perturbed sentence

Architecture	F1	P	R	Acc	batch
no hidden	0.391	0.397	0.391	0.391	32
dense 40	0.387	0.396	0.39	0.39	32
dense 40 - dropout 50	0.396	0.404	0.405	0.405	32

Figure 7.29: TC-13: Hyper parameter tuning results

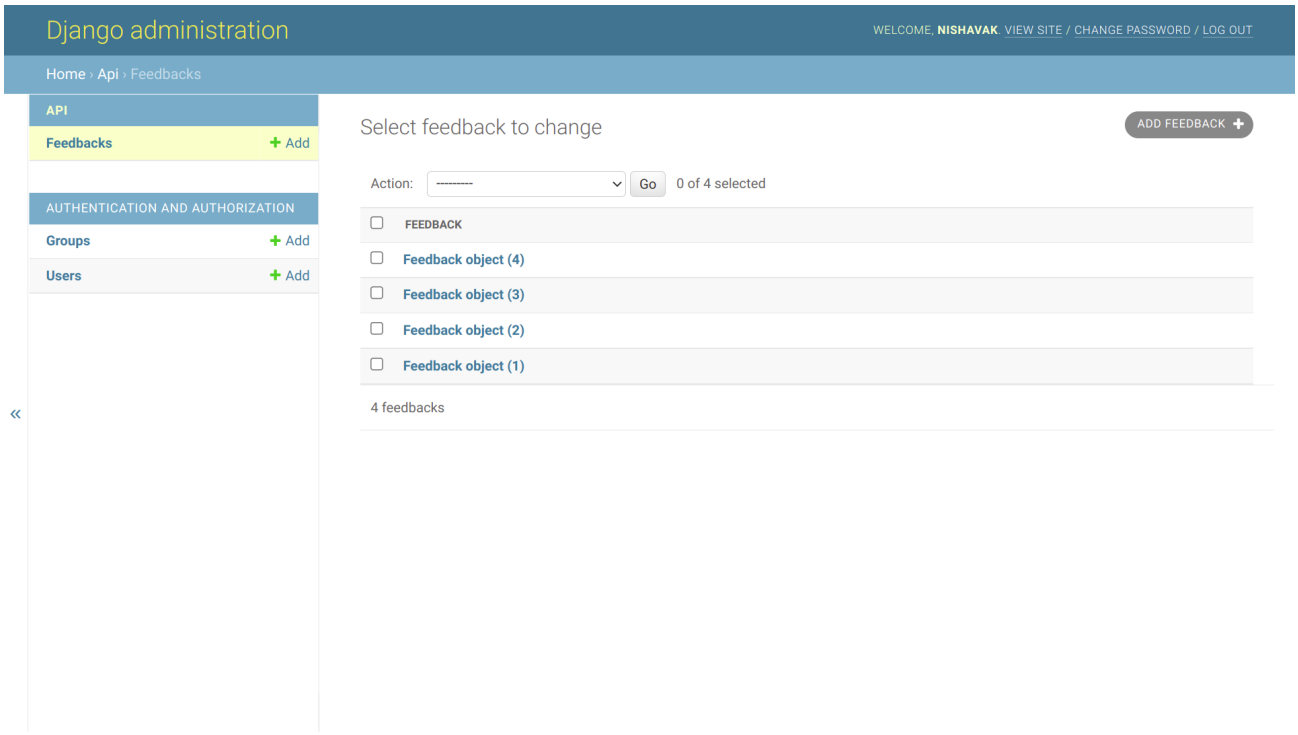


Figure 7.30: TC-14: Django backend feedback object created

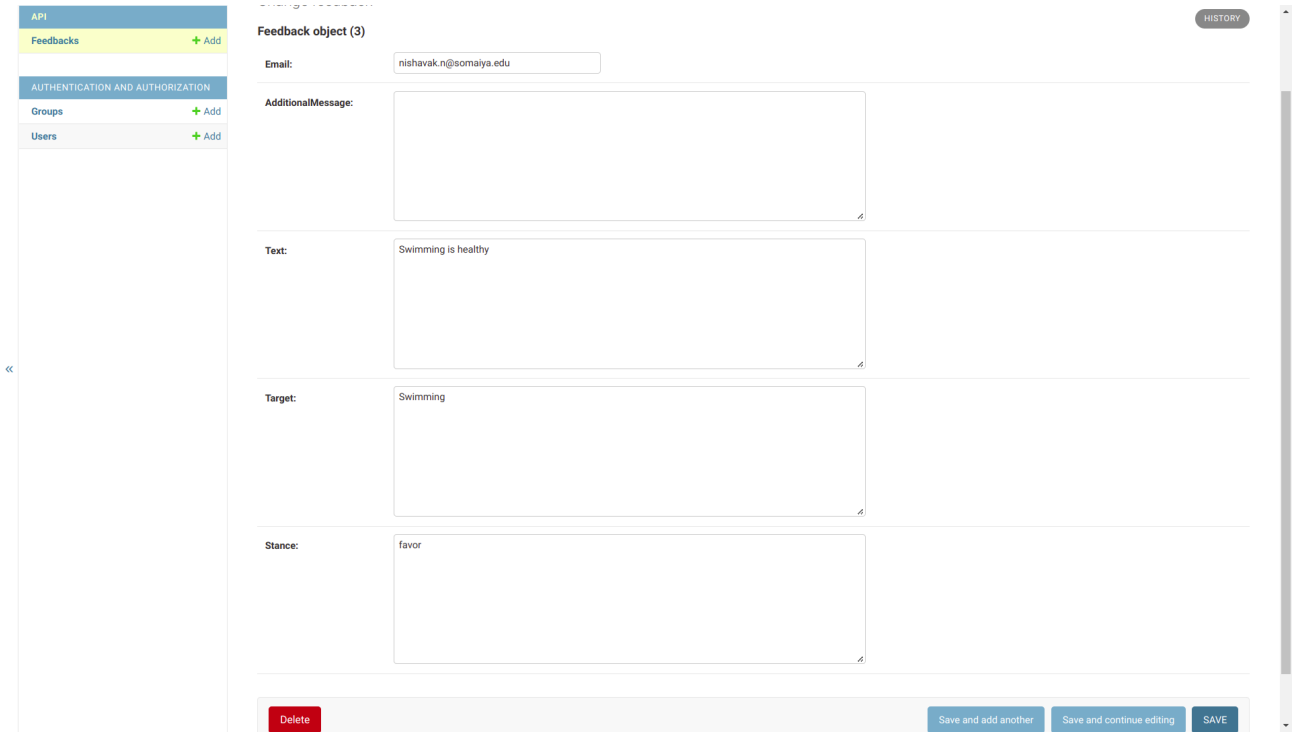


Figure 7.31: TC-14: Django backend feedback object view

In this chapter, we covered the overall testing approach, tools required for testing, features to be and not to be tested, test plan and result. Coming to the end of the thesis, next chapter discusses the conclusions we drew from our results and the scope for future work.

Chapter 8

CONCLUSION AND FUTURE WORK

8.0.1 Conclusions

Understand people's opinions on different topics plays a crucial role in different analytical studies. To this end, we have proposed a system of stance detection. We first started with the topic selection followed by extensive literature survey to get a good understanding of the problem and the existing solutions. After that, we started the documentation work by implementing the SRS, SPMP, SDD, and STD. The prototype design for the project was developed to get a feel of the final interfaces that we would be delivered. Sticking to the project schedule we started the implementation through the pre-processing of the text data from VAST dataset. Preprocessing of text involved removal of specific stopwords, tokenization, lowercasing - to mention a few. Then we generated word embeddings for the text data using 2 approaches: Contextual (SBERT) and context-free (Word2Vec). We introduced new method for generalization of the targets to improve the results. Target generalization methods used were namely BERTopic and Sci-kit learn's Agglomerative clustering. This was followed by the development of different machine learning models for stance detection. The inputs to the model includes target and text which were further classified by the models into one of the following stances: 'favor', 'against', or 'neutral'. The machine learning models which we utilized included sklearn models and neural networks (Dense, LSTM, Bi-LSTM) built using TensorFlow. Hyper-parameter tuning was performed on different combinations of word embeddings, target generalization methods and machine learning models. Results from the evaluation of the models indicated that:

1. Context-free no grouping Gradient Boosting Classifier model was the best performing

model overall.

2. Comparison of contextual vs context-free embeddings shows that context-free overall performed better than contextual for sklearn models, whereas neural network contextual models showed slightly better results than context-free models.

3. Comparison of different grouping types showed that for context free models, Grouping performed better than no-grouping, whereas for contextual models, the effect of grouping was not much seen.

Furthermore, we performed Sentiment Analysis on the same dataset to check the suitability of dataset for another NLP task; since good results were not observed for stance. We found that the model trained on the VAST dataset performed exceedingly well on sentiment analysis. Finally, we developed a user interface which allows interaction with different embedding techniques, target generalization methods and machine learning models.

8.0.2 Scope for Future Work

In future, this project can be extended further to detect stance from texts of different languages apart from English, especially native languages, given the appropriate dataset. Furthermore, multi-target stance detection is another possibility that can be explored. Multi-target stance detection is where the user provides multiple targets and stance is detected for each target individually. Another possible extension is detecting more than 3 categories of stances, given the appropriate data. To improve the performance of models, understanding the trade-off between generalization and specialization of target grouping and different combination approaches of Text and Target to feed to the models can provide assistance.

Bibliography

[1] C. -C. Chen, H. -Y. Tsai, H. -H. Huang and H. -H. Chen, "Stance Identification by Sentiment and Target Detection," 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 2020, pp. 331-338, doi: 10.1109/WIIAT50758.2020.00047.

[2] S. M. Padnekar, G. S. Kumar and P. Deepak, "BiLSTM-Autoencoder Architecture for Stance Prediction," 2020 International Conference on Data Science and Engineering (ICDSE), 2020, pp. 1-5,doi:10.1109/ICDSE50459.2020.9310133.

[3] Vamvas, Jannis and Rico Sennrich. “X -stance: A Multilingual Multi-Target Dataset for Stance Detection.” ArXiv abs/2003.08385 (2020): n. pag

[4] P. Sobhani, D. Inkpen, en X. Zhu, “A Dataset for Multi-Target Stance Detection”, in Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, 2017, bll 551–557.

[5] E. Allaway and K. McKeown, Zero-Shot Stance Detection: A Dataset and Model using Generalized Topic Representations. 2020.

[6] A. Vaswani et al., “Attention Is All You Need”, CoRR, vol abs/1706.03762, 2017.

[7] J. H. Ward, “Hierarchical Grouping to Optimize an Objective Function”, Journal of the American Statistical Association, vol 58, no 301, bll 236–244, 1963.

[8] Isabelle Augenstein, Tim Rocktäschel, Andreas Vla- ¨ chos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In EMNLP.

[9] Chang Xu, Cecile Paris, Surya Nepal, and Ross Sparks. 2018. Cross-target stance classification with self attention networks In ACL.

[10] J. Pennington, R. Socher, en C. Manning, “GloVe: Global Vectors for Word Representation”, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, bll 1532–1543.

[11] J. Devlin, M.-W. Chang, K. Lee, en K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, CoRR, vol abs/1810.04805, 2018.

[12] D. P. Kingma en J. Ba, “Adam: A Method for Stochastic Optimization”, arXiv [cs.LG].

[13] Siddiqua, Umme Chy, Abu Nowshed Aono, Masaki. (2019). Tweet Stance Detection Using Multi-Kernel Convolution and Attentive LSTM Variants. IEICE Transactions on Information and Systems. E102.D. 2493-2503. 10.1587/transinf.2019EDP7080.

[14] S. Ruder, “An overview of gradient descent optimization algorithms”, CoRR, vol abs/1609.04747, 2016.

[15] V. Sanh, L. Debut, J. Chaumond, en T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”, CoRR, vol abs/1910.01108, 2019.

[16] H. Karande, R. Walambe, V. Benjamin, K. Kotecha, en T. S. Raghu, “Stance Detection with BERT Embeddings for Credibility Analysis of Information on Social Media”, CoRR, vol abs/2105.10272, 2021.

[17] S. Lin, B. Wu, T. Chou, Y. Lin and H. Kao, "Bidirectional Perspective with Topic Information for Stance Detection," in 2020 International Conference on Pervasive Artificial Intelligence (ICPAI), Taipei, Taiwan, 2020 pp. 1-8.doi: 10.1109/ICPAI51961.2020.00009

[18] Z. Wang, Q. Sun, S. Li, Q. Zhu and G. Zhou, "Neural Stance Detection With Hierarchical Linguistic Representations," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 635-645, 2020, doi: 10.1109/TASLP.2020.2963954.

[19] Kareem Darwish , Peter Stefanov , Michael Aupetit , Preslav Nakov , Qatar Computing Research Institute Hamad bin Khalifa University, Doha, Qatar Faculty of Mathematics and Informatics, Sofia University ”St. Kliment Ohridski”, Sofia, Bulgaria.

Appendix A

VAST Dataset

VAST dataset is a dataset used for stance detection, first introduced in [5]. VAST consists of a wide range of topics, and covers a broad range of topics, from politics (e.g., 'a Palestinian state'), public health (e.g., 'childhood vaccination') to education (e.g., 'charter schools') . I n addition, the data includes a wide range of similar expressions (e.g., 'guns on campus' versus 'firearms on campus'). This variation captures how humans might realistically describe the same topic and contrasts with the lack of variation in existing datasets. It contains zero-shot(unseen targets by the model in the data) and few-shot data(very few examples per target). The authors of the dataset generated annotations on comments collected from The New York Times 'Room for Debate' section, part of the Argument Reasoning Comprehension (ARC) Corpus.

	post	new_topic	label
0	Regulation of corporations has been subverted ...	companies	0
1	Regulation of corporations has been subverted ...	regulation	0
2	Regulation of corporations has been subverted ...	corporate regulation	0
3	Regulation of corporations has been subverted ...	regulation of corporations	0
4	Absolutely it's needs to be defined and regula...	food labels	0

Figure 8.1: First 5 rows of VAST dataset

Author's Publication

Acknowledgement

We are very pleased to present to one and all our project synopsis on Stance Detection from Text. We are very thankful to K.J Somaiya College Of Engineering, Vidyavihar for supporting our ideas and giving us an opportunity to express them. We would also like to express our special gratitude towards Dr. Shubha Pandit, Principal, K.J Somaiya College of Engineering, Vidyavihar. We are also very thankful to our guide Dr. Irfan A Siddavaam, KJSCE for his continuous guidance and support throughout the project.