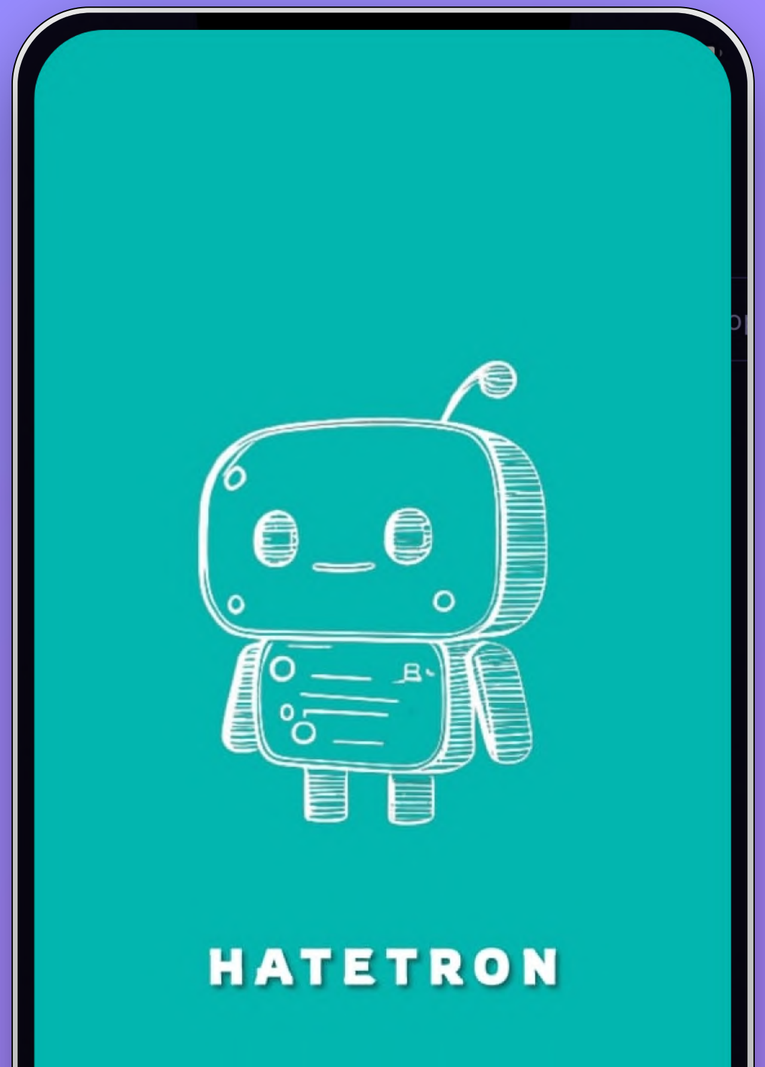
HATETRON

~A HATESPEECH DETECTION CHATBOT

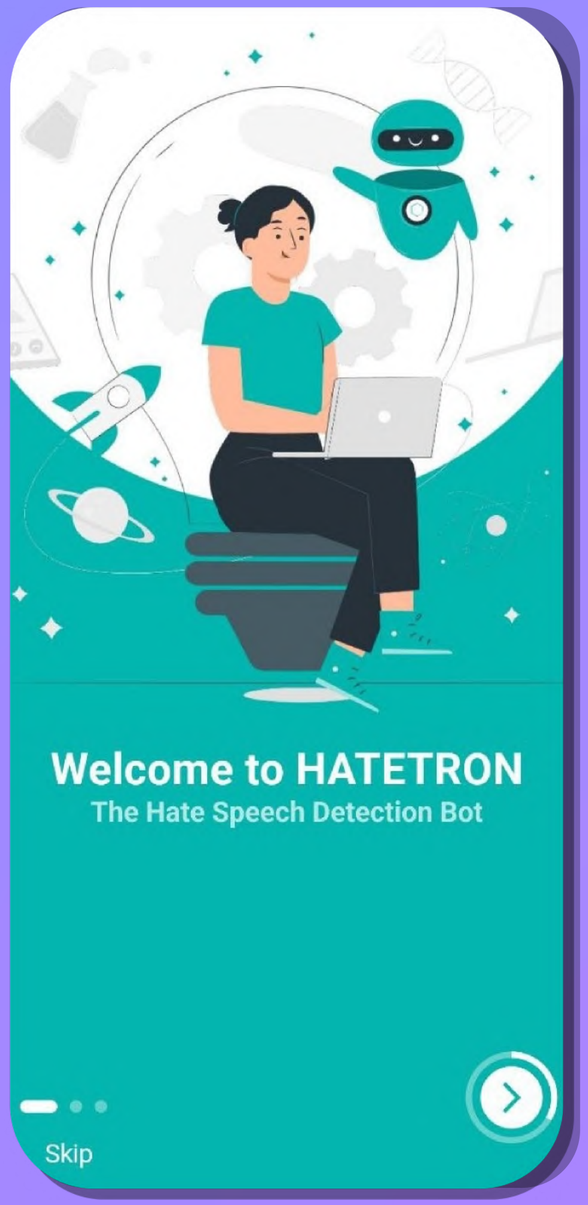


TEAM (GROUP 8)

- Anoushka Srivastava
- Anish Borkar
- Ankur Kaushal
- Dyuti Dasmahapatra
- D Veera Harsha Vardhan Reddy
- Godavarthi Sai Nikhil
- Himanshu Sharma
- Nichenametla Karthik Raja
- Vandamasu Sai Sumanth

PROBLEM STATEMENT

- Hate speech is a growing concern on the internet and social media platforms, as it can lead to harm, marginalization, and discrimination towards individuals and communities based on their ethnicity, race, religion, gender, sexual orientation, or other characteristics and can hurt others emotions also.
- The challenge is to develop a solution that can accurately identify hate speech in real-time and mitigate its harmful effects.
- The goal of hate speech detection is to provide a safe and inclusive online environment by reducing the spread of hate speech and promoting respectful communication.
- To develop a system that can accurately identify and respond to instances of hate speech in real-time communications, while also respecting cultural and contextual nuances and protecting freedom of expression.



COMPETETION

1. **Perspective by google:**
This is an api based on the machine learning models to detect toxic language in text based content.

2. **Hatesonar by hatelab:**
Along with machine learning this also uses natural language processing to prevent online hate.

3. **Clarifai by Clarifai Inc.:**
This is an AI moderation model that can be trained to identify flag hate speech and other form of abusive language in text, images as well as videos

METHODOLOGY

ML CRISP-DM

Experiment Design

Flutter Widget Use

Business Understanding

Problem Statement

Hate speech and offensive language are common on social media platforms, hindering constructive conversations.
The aim is to develop a real-time chatbot to detect hate speech.
The chatbot should flag messages containing offensive language, slurs, or discriminatory remarks.
The goal is to improve online conversations by reducing hate speech and promoting respect.

Target Audience

Hate speech detection targets those who want to prevent it on their online platforms.
The audience includes social media companies, news outlets, schools, government agencies, and non-profits.
It is also relevant for influencers, students, and teachers.
The goal is to monitor and prevent hate speech in online spaces.

Data Understanding

	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she
3	3	0	2	1	1	!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might
5	3	1	2	0	1	!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down t
6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too mu
7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of you big bitches coming for us skinny
8	3	0	3	0	1	" & you might not get ya bitch back & thats that "
9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Mariam" bitch
10	3	0	3	0	1	" Keeks is a bitch she curves everyone " lol I walked into a conversation like this. Su
11	3	0	3	0	1	" Murda Gang bitch its Gang Land "

Screenshot of hate speech dataset

Data Understanding

The dataset has 24,783 rows and 7 columns: index, count, hate_speech, offensive_language, neither, class and tweet.

- Index
- Count: number of CrowdFlower users who coded each tweet (min is 3, sometimes more users)
- Hate_speech: number of CF users who judged the tweet to be hate speech
- Offensive_language: number of CF users who judged the tweet to be offensive
- Neither: number of CF users who judged the tweet to be neither offensive nor hate
- Class : class label for majority of CF users.(0 -> Hate, 1 -> Offensive, 2 -> Safe)
- Tweet: text tweet

```
#dimensionality of dataset
tweet_df.shape
```

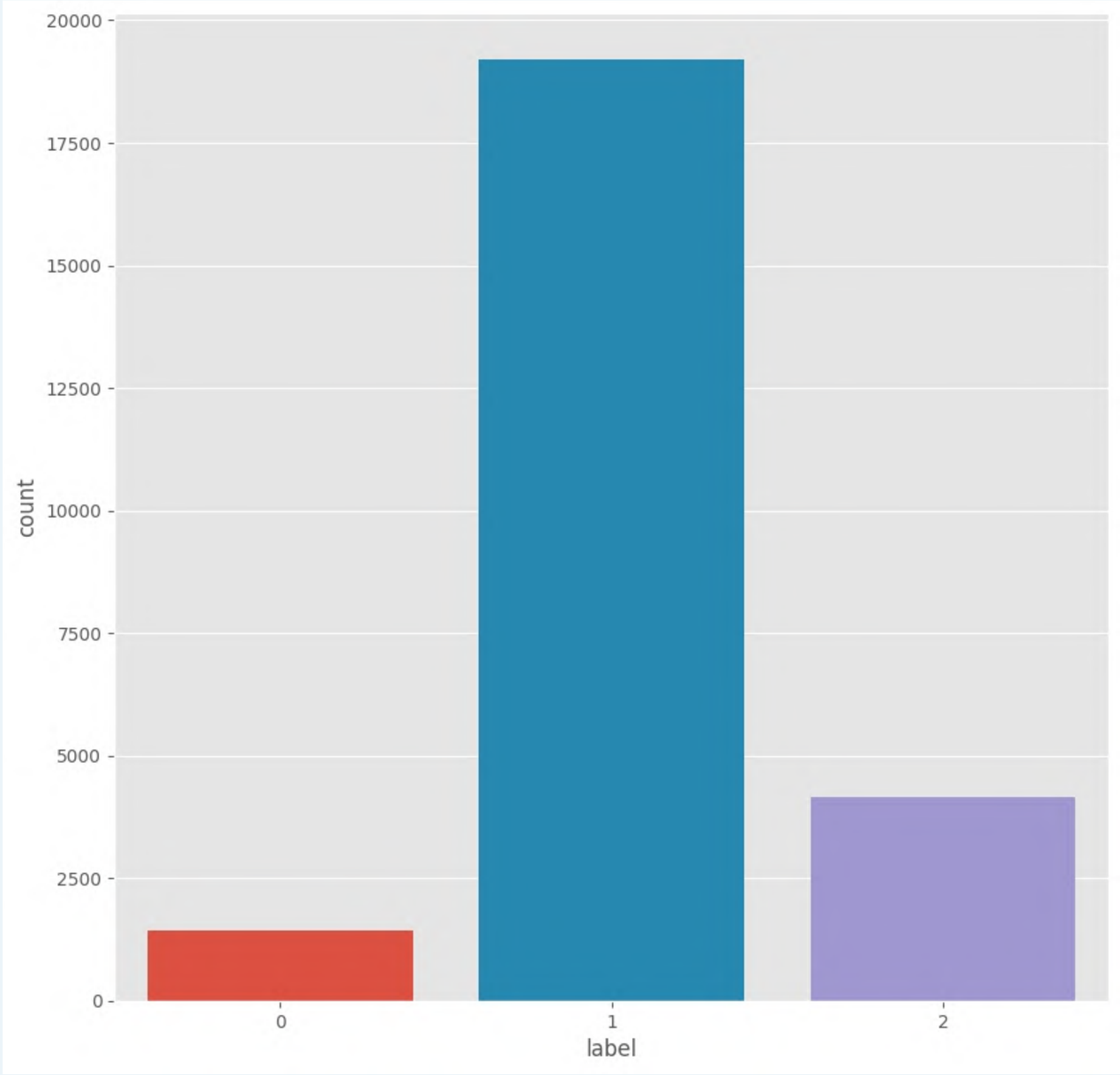
```
(24783, 7)
```

Data Understanding

We can observe that the dataset has an imbalance in the number of rows for offensive speech compared to safe and hate speech. This could lead to very poor predictions for hate speech and safe speech.

```
tweet_df['class'].value_counts()
```

```
1    19190
2     4163
0     1430
Name: class, dtype: int64
```



Data Preprocessing

	count	hate_speech	offensive_language	neither	class	tweet
0	3.0	0.0	0.0	3.0	Safe_Speech	!!! RT @mayasolovely: As a woman you shouldn't...
1	3.0	0.0	3.0	0.0	Offensive_Speech	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	3.0	0.0	3.0	0.0	Offensive_Speech	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3.0	0.0	2.0	1.0	Offensive_Speech	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	6.0	0.0	6.0	0.0	Offensive_Speech	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

```
# dimensionality of dataset
tweet_df.shape
```

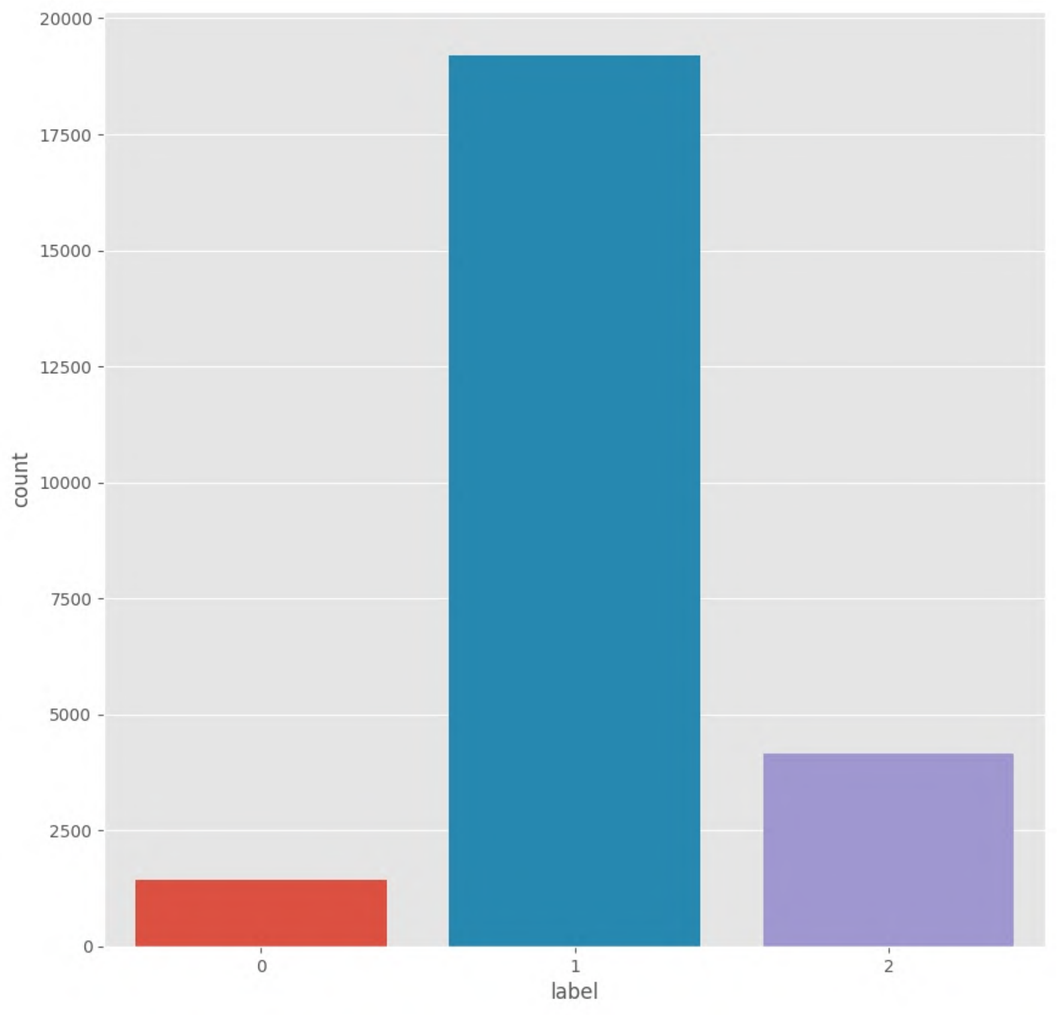
```
(44728, 6)
```

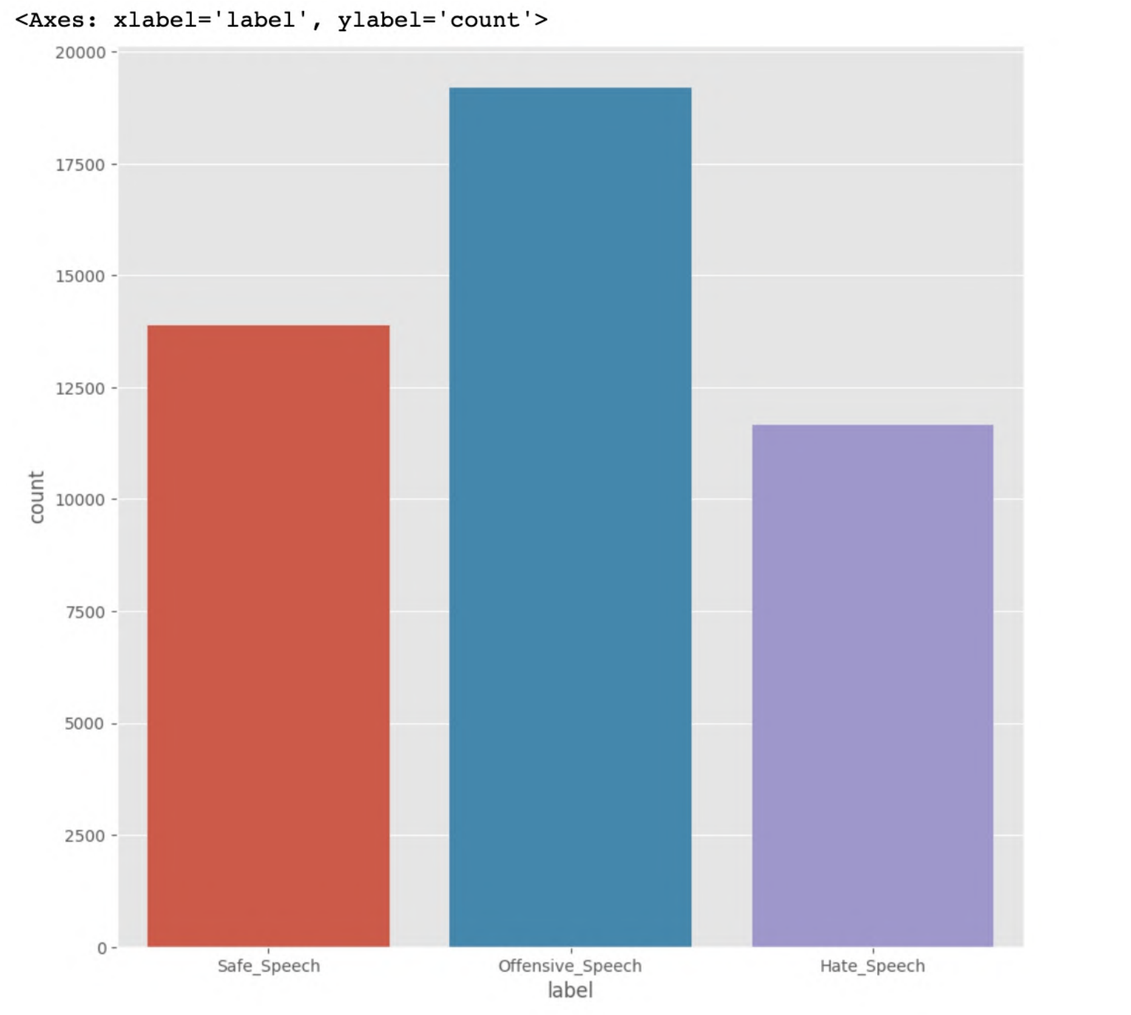
```
# Frequency of each label
tweet_df['class'].value_counts()
```

```
Offensive_Speech    19190
Safe_Speech         13882
Hate_Speech         11656
Name: class, dtype: int64
```

The dataset has three classes:
19190 is classified as Offensive_Speech
13882 is classified as Safe_Speech
11656 is classified as Hate_Speech

Data Preprocessing





Data Preprocessing

```
# Renaming 'class' column to 'label'
tweet_df.rename(columns = {'class':'label'}, inplace = True)
```

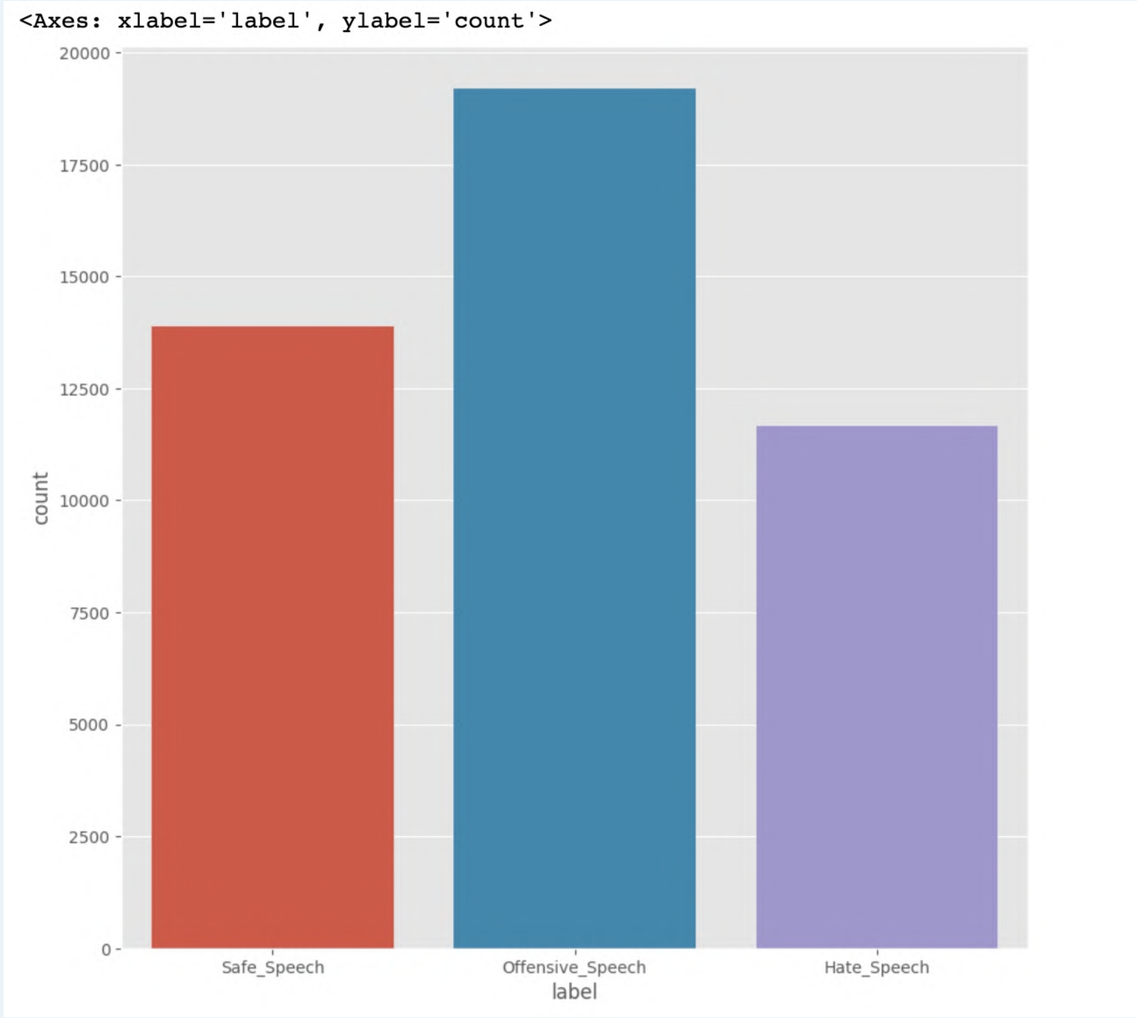
```
# Extracting columns 'label' and 'tweet'
tweet_df = tweet_df[['label','tweet']]
```

```
def data_processing(text):
    pattern = r'https?://\S+|www\.\S+'
    text = re.sub(pattern, '', text)
    text = text.lower()
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    text = re.sub(r'#', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'ð', '', text)
    text = re.sub(r'rt', '', text)

    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
```

```
# apply data_processing function to each tweet
tweet_df['tweet'].head(10).apply(data_processing)
```

```
0    [woman, shouldnt, complain, cleaning, house, a...
1    [boy, dat, coldtyga, dwn, bad, cuffin, dat, ho...
2    [dawg, ever, fuck, bitch, sta, cry, confused, ...
3                                [look, like, tranny]
4    [shit, hear, might, true, might, faker, bitch,...
```



Data Modeling

```
Tweet_train, Tweet_test, Label_train, Label_test = train_test_split(tweet_df['tweet'],
                                                    tweet_df['label'], test_size = 0.3, random_state = 101)
```

```
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(analyzer = data_processing)),
    ('classifier', RandomForestClassifier(random_state = 101))
])
```

```
p1 = Pipeline([
     ('tfidf', TfidfVectorizer(analyzer = data_processing)),
    ('classifier', LogisticRegression(max_iter=1000,random_state = 101))
])
```

```
p2 = Pipeline([
     ('tfidf', TfidfVectorizer(analyzer = data_processing)),
    ('classifier', MultinomialNB())
])
```

Data Modeling

	precision	recall	f1-score	support
Hate_Speech	0.89	0.82	0.86	3405
Offensive_Speech	0.92	0.94	0.93	5853
Safe_Speech	0.88	0.90	0.89	4161
accuracy			0.90	13419
macro avg	0.90	0.89	0.89	13419
weighted avg	0.90	0.90	0.90	13419

Classification report of Random Forest

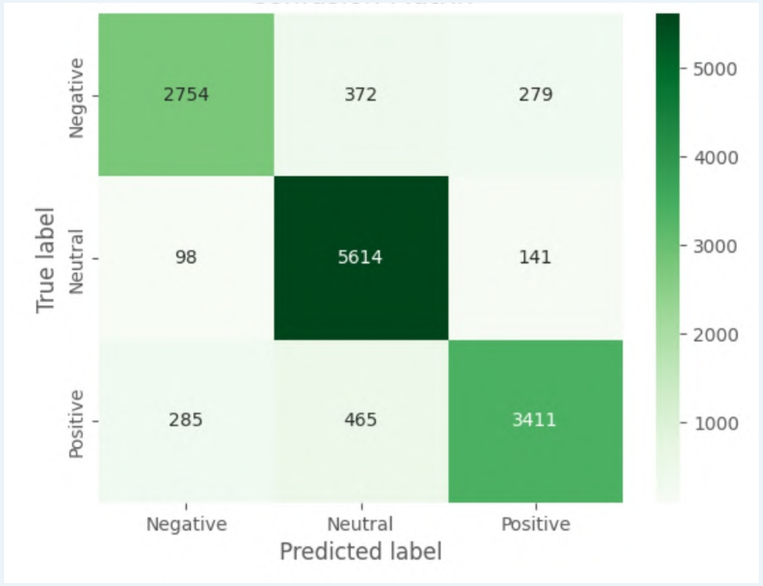
	precision	recall	f1-score	support
Hate_Speech	0.88	0.82	0.85	3405
Offensive_Speech	0.94	0.92	0.93	5853
Safe_Speech	0.86	0.93	0.90	4161
accuracy			0.90	13419
macro avg	0.89	0.89	0.89	13419
weighted avg	0.90	0.90	0.90	13419

Classification report of Logistic Regression

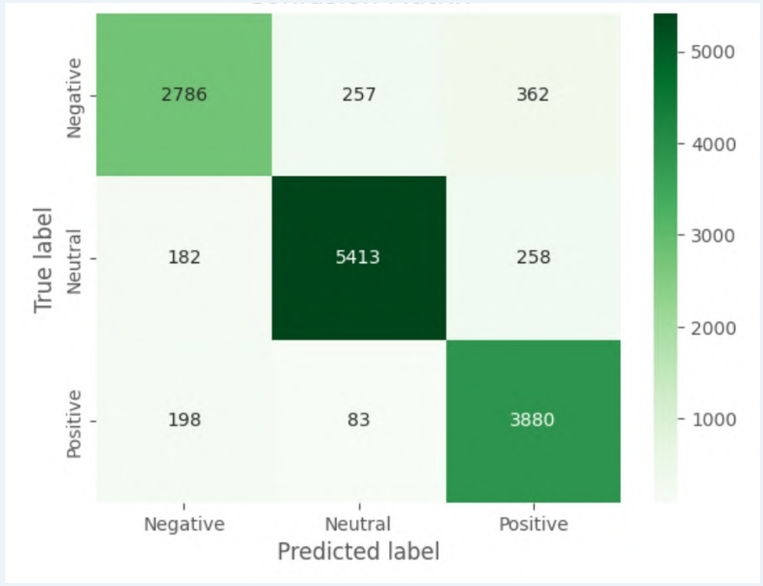
	precision	recall	f1-score	support
Hate_Speech	0.92	0.76	0.83	3405
Offensive_Speech	0.75	0.98	0.85	5853
Safe_Speech	0.93	0.66	0.77	4161
accuracy			0.82	13419
macro avg	0.87	0.80	0.82	13419
weighted avg	0.85	0.82	0.82	13419

Classification report of Multinomial Naive Bayes

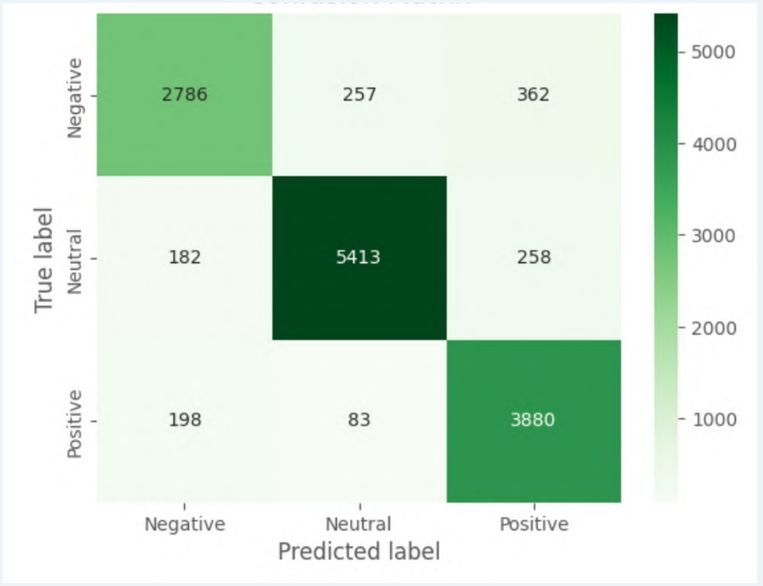
Data Modeling



Confusion Matrix of Random Forest

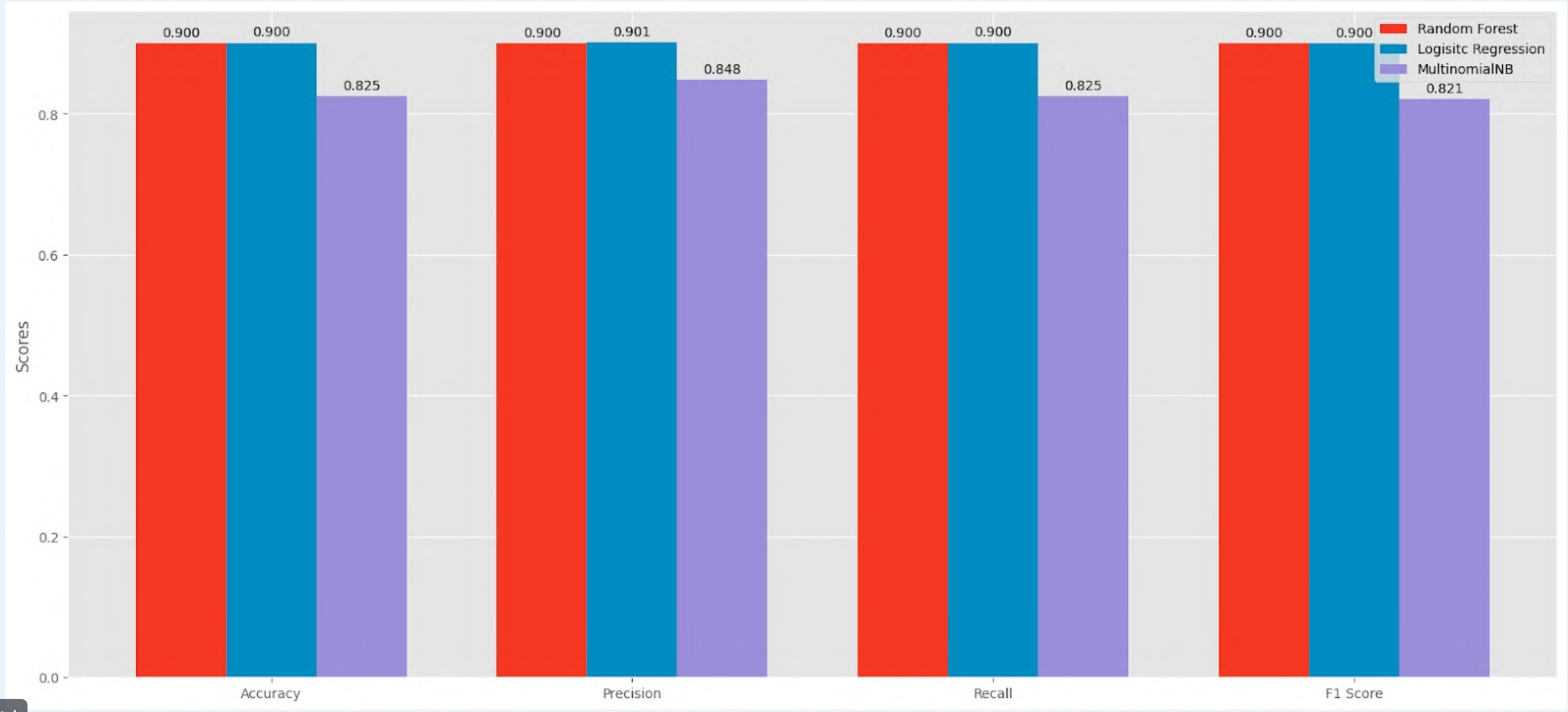


Confusion Matrix of Logistic Regression

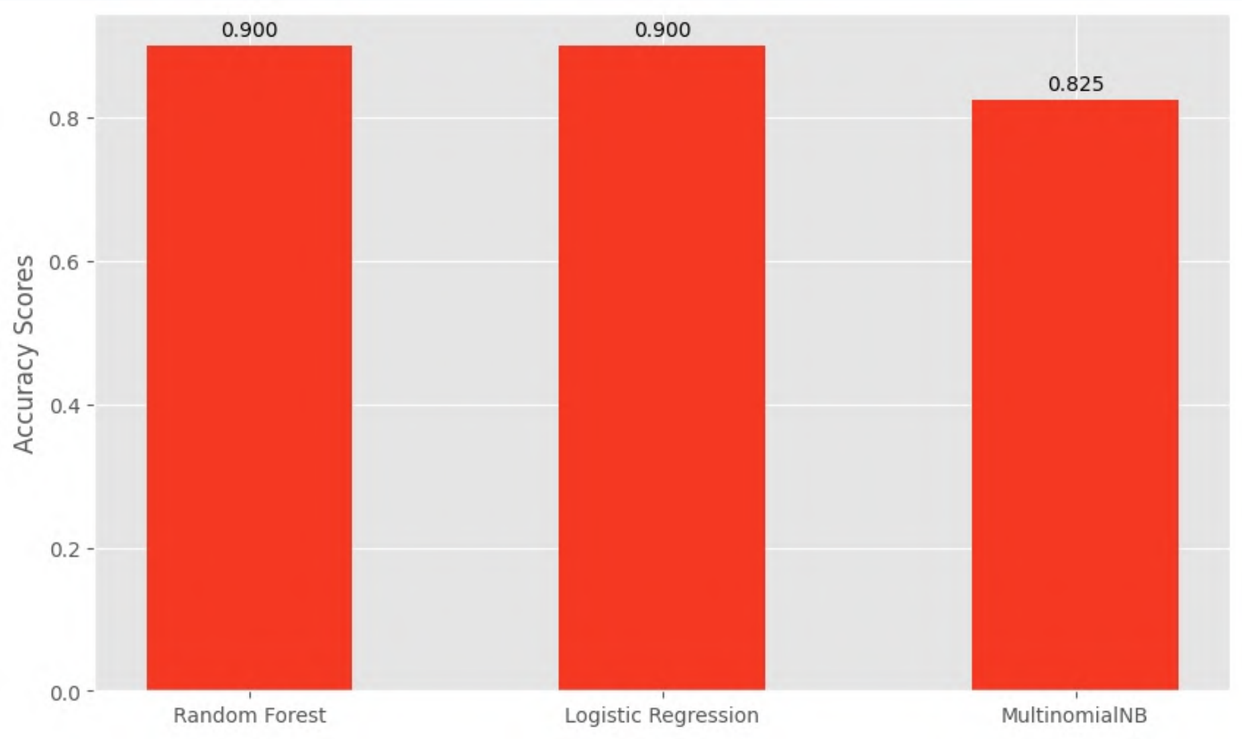


Confusion Matrix of Multinomial Naive Bayes

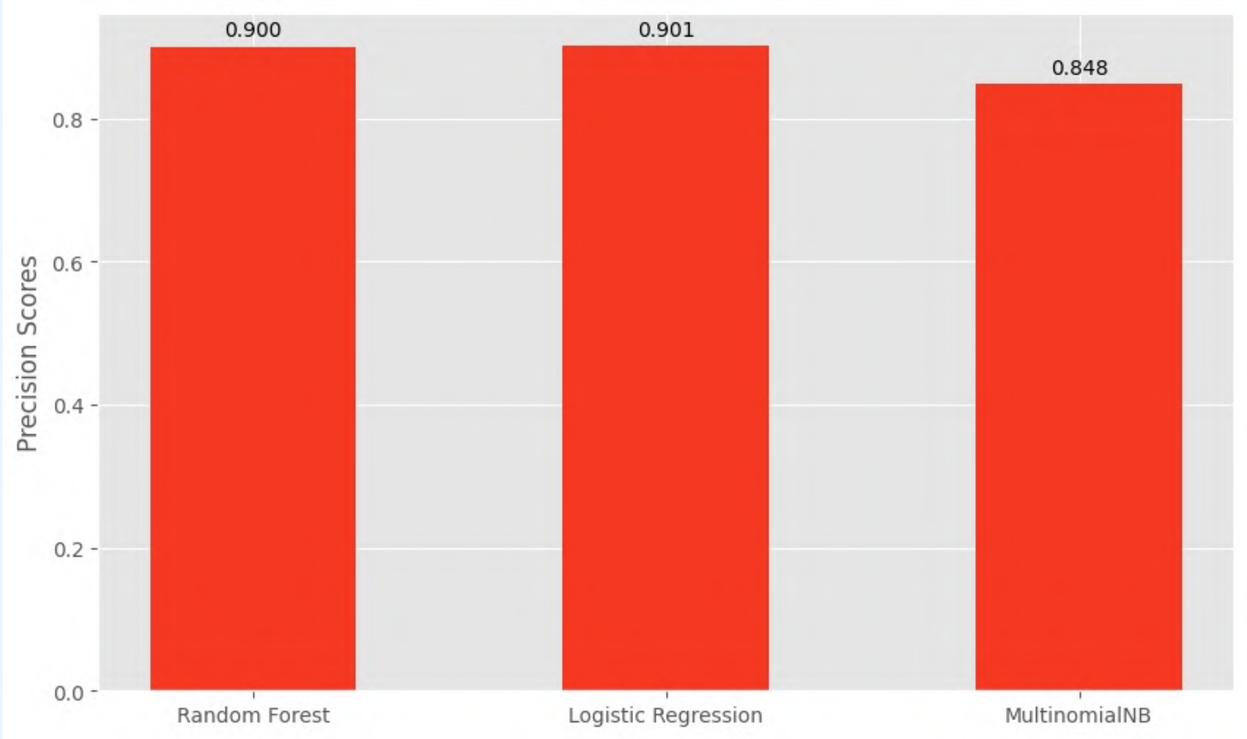
Data Validation



Data Validation

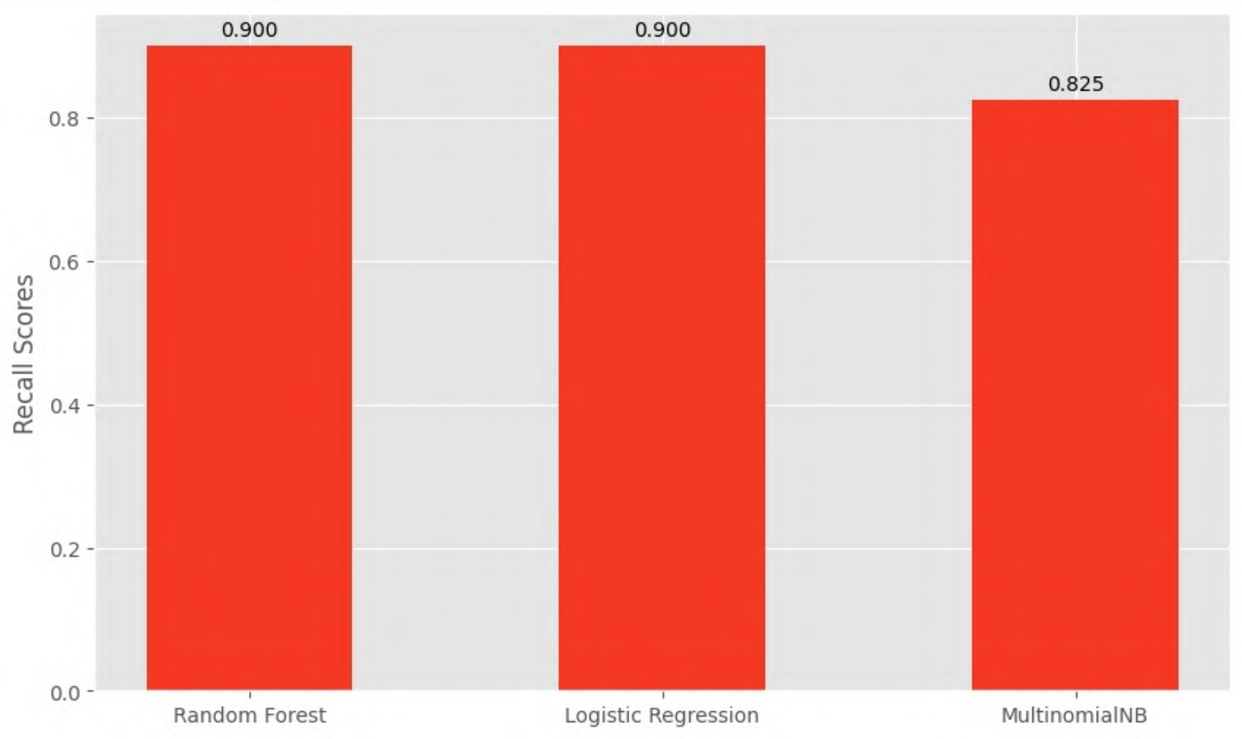


Comparison on the basis of Accuracy Score

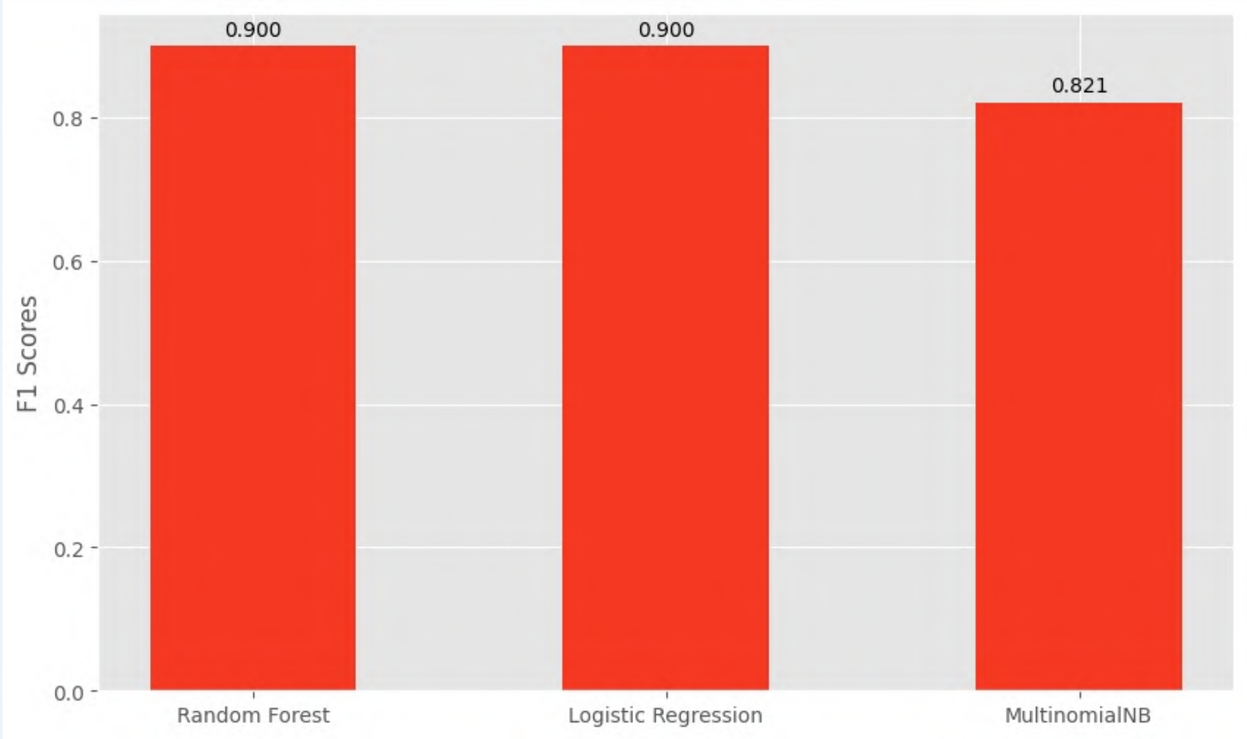


Comparison on the basis of Precision Score

Data Validation

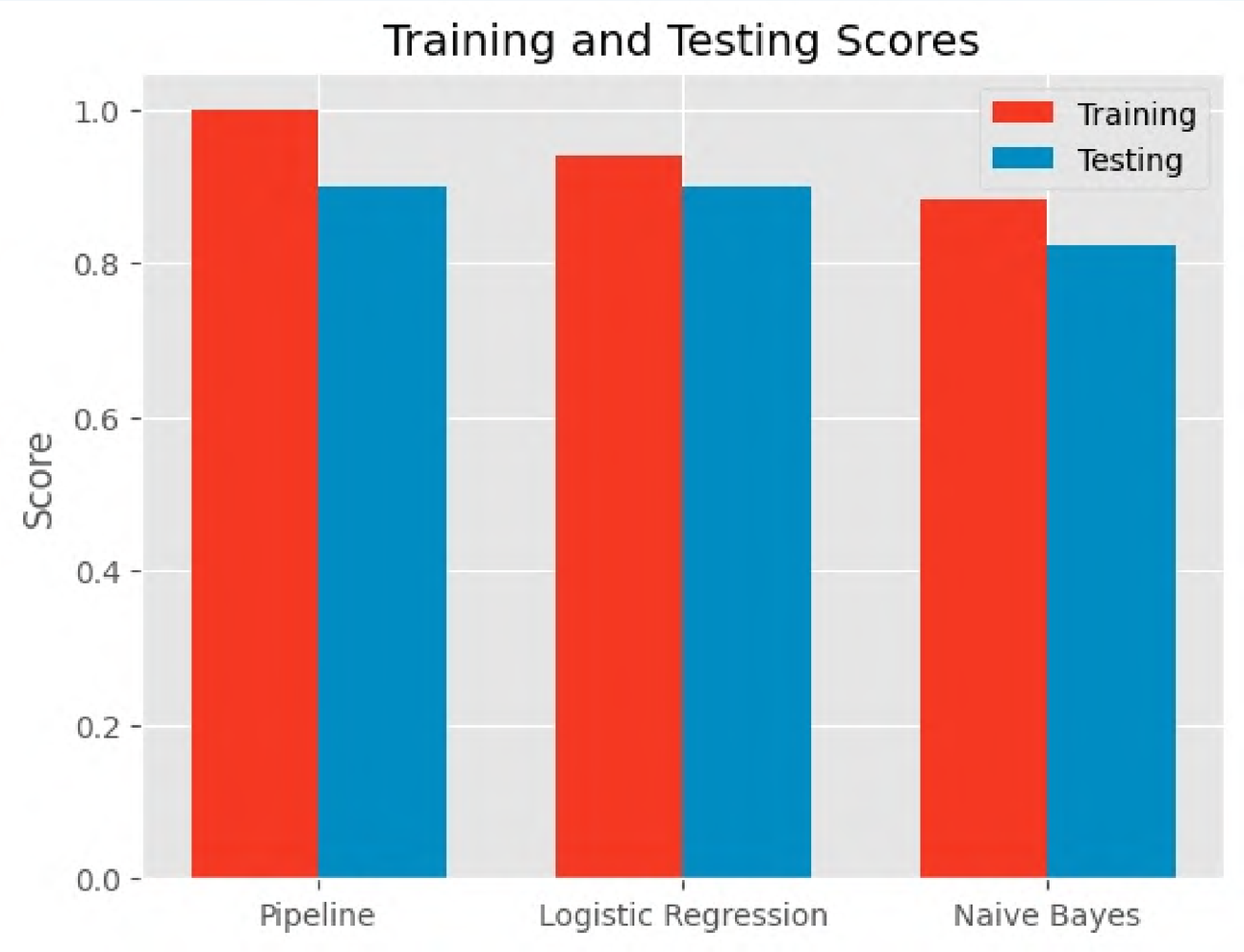


Comparison on the basis of Recall Score



Comparison on the basis of F1 Score

Data Validation



Comparison on the basis of Training & Testing Scores

```
Training and Testing Score of Random Forest
Training score: 0.999
Testing score: 0.900

Training and Testing Score of Logistic Regression
Training score: 0.939
Testing score: 0.900


Training and Testing Score of Multinomial Naive Bayes
Training score: 0.882
Testing score: 0.825
```

Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid to search over
param_grid = {
    'tfidf__max_features': [5000, 10000, 30000],
    'tfidf__ngram_range': [(1,1), (1,2)],
    'classifier__penalty': ['l1', 'l2'],
    'classifier__C': [0.01, 0.1, 1, 10, 100]
}

# Perform grid search cross-validation
grid = GridSearchCV(logreg, param_grid, cv=5, n_jobs=-1)
grid.fit(Tweet_train, Label_train)
```

- The max features set to be either 5000, 10000 or 30,000
- The range of n-grams set to be either unigram or bigram
- Type of regularisation used with options L1 and L2
- Regularisation strength to be either 0.01, 0.1, 1 or 100.

```
Best parameters:  {'classifier__C': 10, 'classifier__penalty': 'l2',
                   'tfidf__max_features': 30000, 'tfidf__ngram_range': (1, 2)}
Best score:  0.9007881248119223
```

	precision	recall	f1-score	support
Hate_Speech	0.89	0.85	0.87	2338
Offensive_Speech	0.92	0.93	0.93	3838
Safe_Speech	0.89	0.92	0.91	2770
accuracy			0.90	8946
macro avg	0.90	0.90	0.90	8946
weighted avg	0.90	0.90	0.90	8946

Model Deployment

https://hatespeech.azurewebsites.net

```
// 20230430181106
// http://3.126.139.154/?text=hello%20there

{
  "result": "This is a safe speech."
}
```

```
// 20230430181145
// http://3.126.139.154/?text=black%20people%20should%20be%20killed

{
  "result": "This is a hate speech."
}
```

Screenshots of api on giving two different inputs.

METHODOLOGY

MAD :

Login Page:

The programme writes two properties for the login page and a method to show the registration page. There are text areas for the password and email, as well as a login button.Future class is also used Widgets like Scaffold, SafeArea, Centre, SingleChildScrollView, Text, Container, Padding, TextField, and GestureDetector are used to organise the layout.

Registration Page:

In order to create a new user account using Firebase Authentication and store the user's data in Firestore, the app develops a user registration page that asks users to provide their personal information. The widgets 'RegisterPage' and '_RegisterPageState' are used for this, while the 'TextEditingController' widget controls the input fields. To establish a new user account and add user information to Firestore, use and to'signUp()' method.

Forgot Password:

The code implements a "Forgot Password" function using Flutter and Firebase authentication. Users enter their email address and receive a password reset link via email. The 'ForgotPasswordPage' stateful widget displays a form with a Material AppBar, a Column, a Padding widget, a Text widget, a SizedBox widget, a Container widget, a TextField widget, and a MaterialButton widget. An AlertDialog shows a success message or an error message.

METHODOLOGY

Home Page:

This code enables the "Forgot Password" function for a Flutter mobile app using Firebase authentication. The 'ForgotPasswordPage' stateful widget shows a password reset form with a TextField and MaterialButton, and the 'passwordReset()' function sends a password reset email to the entered email address, displaying an AlertDialog for success or error messages.

Chat Area:

This code creates a Flutter chatbot using DialogFlowtter package and API to give predictions and warnings based on user messages. The Home class initialises DialogFlowtter instance, a TextEditingController, and a list of messages to display on the screen. It uses conditional rendering to display UI components and alerts for risky messages.

Logout:

The "LogoutScreen" is a stateless widget that displays a confirmation dialog when the user chooses to log out of the app. It includes a message asking for confirmation, two buttons, and navigates to the login screen when the user confirms the log out action. The widget is used to handle logout actions throughout the app.

EXPERIMENT DESIGN

- Eight different iterations are done in usability testing and improvements are done for each specific interface. The tests were conducted with different types of participants, including students, general public, and design professionals.
- The feedback was collected through different methods such as SUS, UX Heatmap, Focus Groups, Survey, and Expert Review. The usability issues encountered were related to navigation difficulties, poor terminology, confusing feedback, poor color contrast and design, lack of features, and difficulty in accessing/viewing previous records
- The improvements made in each iteration were based on the feedback collected and included simplifying the interface design, redesigning icons, simplifying navigation, using consistent typography, improving color contrast and design, and adding new features like a scan the text option.
- Links to artifacts collected such as SUS reports, Google Forms, and annotated screenshots were provided. Overall, the iterative testing and improvement process helped in identifying and resolving usability issues, resulting in an improved user experience.

Usability Evaluations:

Learnability:

To improve learnability, the app includes:

- A brief tutorial for newcomers
- Buttons with clear labels to indicate their purpose
- Icons that closely match their intended function which helps users quickly understand the app's features and functionality.

Efficiency:

Efficiency is improved by:

- Bottom Navigation Bar for easy screen browsing
- App doing most of the work with clear instructions
- Image to text converter to save time
- Color-coded warnings for efficient identification and response to hate speech.

Memorability:

- Clearly labeled buttons to indicate their functionality
- Icons that closely match their intended function
- Contextual guidance and hints at each step to assist users in understanding the required information
- Assistance in providing accurate hate speech detection information

Satisfaction:

To improve satisfaction, the app includes:

- Cross-platform responsive design for a seamless user experience
- Accessibility for all users, including those with disabilities
- User-friendly and intuitive design with clear instructions and prompts
- Clutter-free interface with only necessary components
- Quick and accurate hate speech detection with engaging conversational experience.

Error Protection

To protect against errors, the app includes:

- Color-coded buttons for clear visual cues
- Warning prompt before logging out to prevent data loss
- Confirmation prompt before exiting an ongoing chat to prevent accidental exits.

Utility:

The app's utility is in accurately detecting and flagging hate and offensive speech content, which can be achieved through advanced algorithms capable of identifying such content.
The app's utility relies on its ability to provide timely and accurate warnings or alerts to users when hate speech is detected.

RESULTS

ML

Usability Evaluation

UI Screenshots

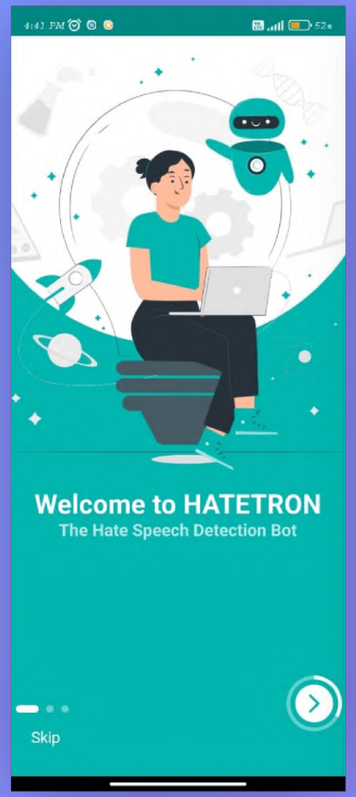
ML RESULT

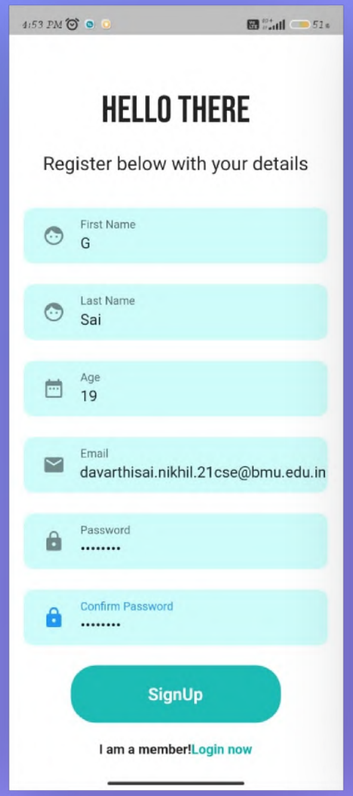
Model	Accuracy	Precision	Recall	F1 Score	Overfitting?
Random Forest Classifier	90.0%	90.0%	90.0%	90.0%	YES
Logistic Regression	90.0%	90.1%	90.0%	90.0%	NO
Multinomial Naive Bayes	82.5%	84.8%	82.5%	82.1%	NO

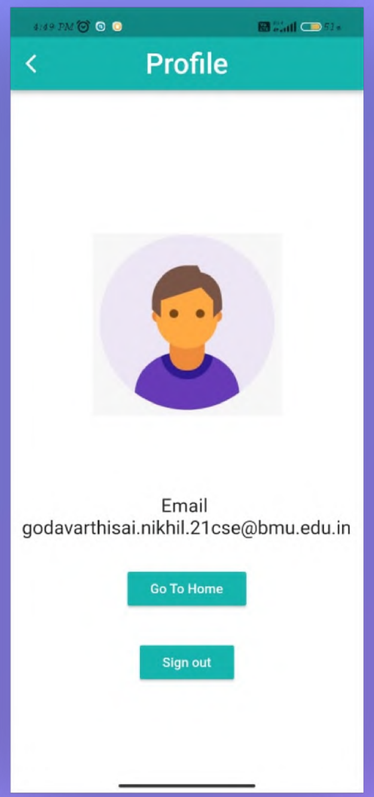
USABILITY EVALUATION RESULT

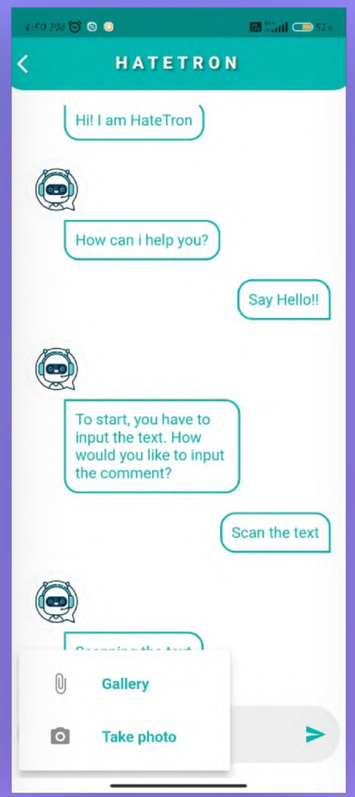
Iteration	Date/ Month	Duration	Number of Participants	Type of Participants	Method of Feedback	Type of usability issue encountered	Improvement from previous iteration	Link to artifacts collected
1	2nd March	1 week	12	Students	Usability Testing and SUS	Difficulty in using the interface and unclear icons	N/A - First iteration	Link for SUS Result
2	10th March	3 days	4	Students	UX Heatmap	Navigation difficulties	Simplified interface design, redesigned icons	Screenshots of user testing UX Heatmap
3	13th March	1 day	5	General Public	Focus Groups	Poor terminology and poor typography	Simplified navigation	N/A
4	15th March	2 days	10	Students	Survey	Confusing feedback	Simplified terminology with consistent typography	Link to Google Forms
5	18th March	1 day	2	Design professionals	Expert Review	Poor color contrast and design	More intuitive feedback	Annotated Screenshots of design changes
6	20th March	1 week	6	All users	Focus Groups	Lack of features to complete key tasks	Improved color contrast and design	Feedback mentioned below
7	28th March	1 day	11	Students	SUS Report	Bad user experience due to long steps for completing key tasks	Improved by adding scan the text option	Link for SUS Report
8	1st April	3 days	16	All	Survey	difficulty in accessing/viewing previous records.	Improved user experience	Link to Google Forms

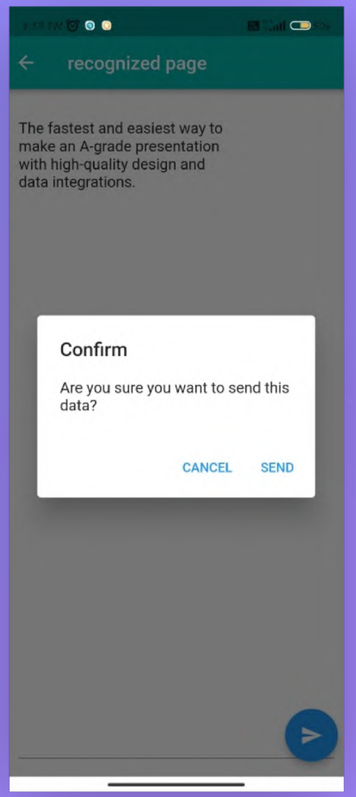
UI SCREENSHOTS

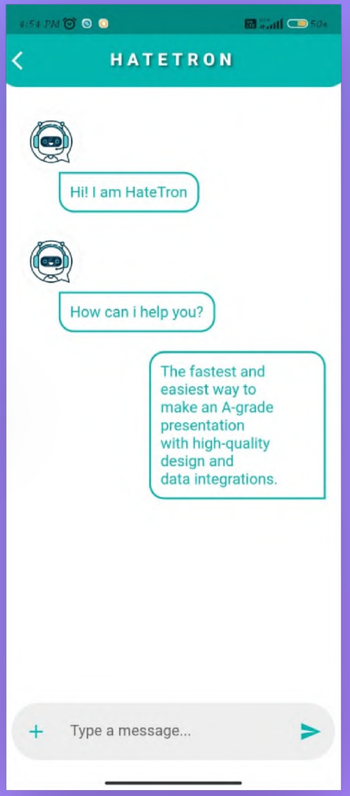






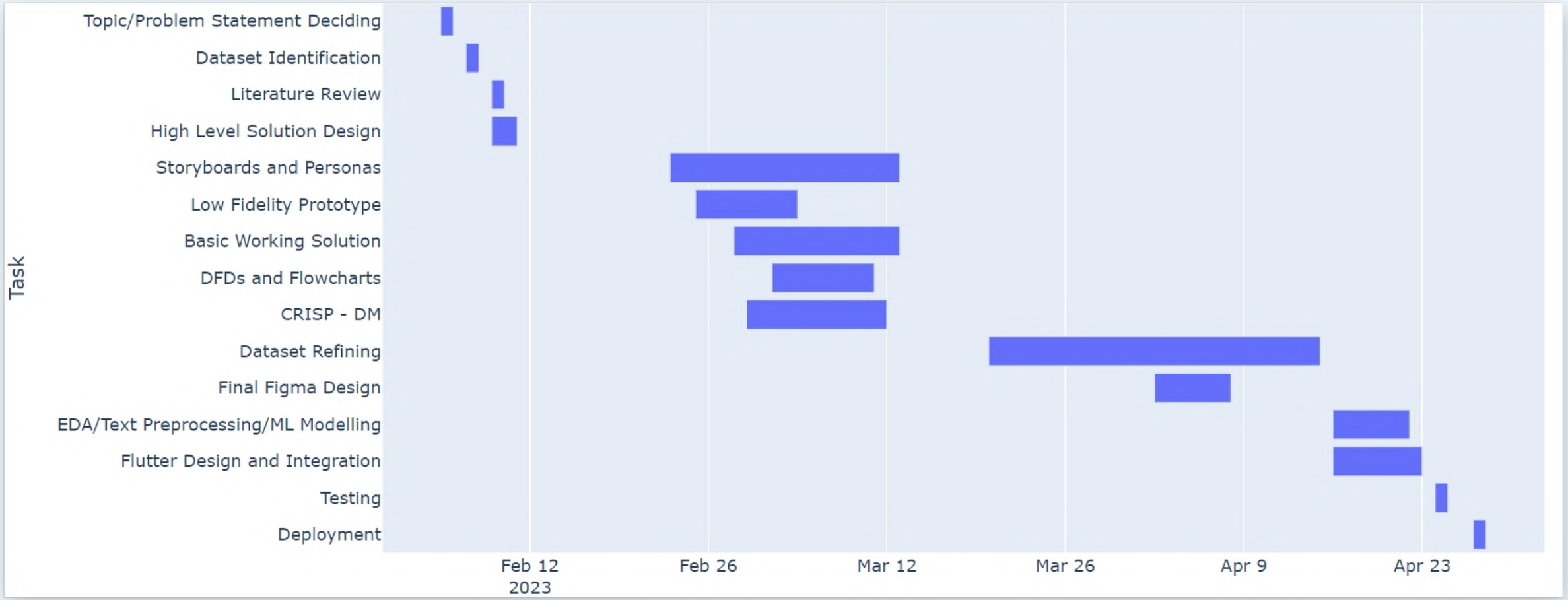






PROJECT MANAGEMENT

GANTT CHART



PROJECT MANAGEMENT

- The group divided tasks based on members' strengths and expertise.
- All nine members contributed equally to designing a basic working solution.
- The group divided into subgroups for the second evaluation, with specific tasks assigned to each subgroup.
- The group continued to refine and improve the project as they went along.
- The group divided into two parts for the final stage, with three members refining the dataset and finalizing ML models, and six members working on Flutter Design, Integration, and Deployment.
- The group showed good organization, planning, and commitment to excellence in project management.

DEMO

SCAN THE QR CODE TO SEE THE DEMONSTRATION OF HATERON

THANK YOU