**Zodiac Killers**

**Link to Submission Video:**
**https://youtu.be/RmOvOX1zJ6k OR**
**https://drive.google.com/file/d/1q6xE7wZh1BPH7Fa3ePRkayiv0mpyU1aW/view?usp=share_link**

**Overview**

The goal of our project is to implement a binary text classification model for islamophobia, hate against Muslims. In addition to creating the model, we also designed a web interface for users to input text to see if it is Islamophobic or not. To allow for ongoing data collection, users can add additional islamophobic comments to the dataset through the user interface. Lastly, the prevalence of islamophobia from different data sources is visualized in a dashboard.

**Data Collection**

We have three data sources: Twitter, Reddit, and YouTube. Twitter data was collected programmatically through Twitter's API (link to code to get tweets), Reddit's API (link to code to get comments) and YouTube comments were collected manually. Hashtags and @'s were removed from tweets. Similarly, tweets that clearly came from bots were deleted (daily Quran recitations). The code to clean the text and to merge the data is here. Table 1 shows how much data in the preliminary data set came from each data source. This preliminary data set was used to evaluate the models – the newest data set has 1000 comments total.

**Table 1. Distribution of text by data source in preliminary data set**

| **Source** | **Count** |
|---|---|
| Reddit | 410 |
| Twitter | 338 |
| YouTube | 42 |

We encountered two problems while collecting data. First, Twitter and YouTube have decent hate-speech detection and censoring algorithms, so the data from those sources are heavily non-islamophobic. Only 17% of the data is islamophobic. Second, the manual labeling was very challenging because reading the hateful comments is mentally draining. Coder fatigue is especially high when the task is focused on something so discouraging.

**Picking a Model**

We estimated six different text classification models without weights: logistic regression, linear SVC, multinomial naive bayes, Bernoulli naive bayes, BERT, and Keras (as shown in Table 2). The models have a binary output (1 if Islamophobic; 0 otherwise. Each model uses tf-idf for vectorization. To evaluate model performance across the preliminary data set with a 75/25 train-test split, we primarily used macro f1-scores, which are preferable to micro f1-scores when the data set is as skewed as ours. However, four of the unweighted models (logistic regression, multinomial naive bayes, bernoulli naive bayes, and BERT) had 0 recall for the Islamophobic class, which means these models classify everything as not Islamophobic. When the data set is so heavily skewed, models learn to classify everything as the class that appears most frequently. To mitigate this, we decided to estimate two weighted models with logistic regression and linear SVC – none of the other models have built in functionality to add weights. The weights were designed to be proportional to the class distribution, such that Islamophobic labels held over four times the weight of non-Islamophobic labels. After adding the weights, the logistic regression no longer classified all text as not Islamophobic, but linear SVC still performs better when comparing both the macro- and micro- f1-scores; therefore, we decided to use the linear SVC model as our final model. Note that the f1-score is still lower than ideal, but given that hate speech detection is a naturally challenging research area, this is decent as the outcome of a class project that relies on a small, novel data set.

**Table 2. Model Metrics for preliminary data set with 75/25 train-test split**

| | **Unweighted** | | **Weighted** | |
|---|---|---|---|---|
| **F1-score** | **Macro** | **Micro** | **Macro** | **Micro** |
| Logistic Regression | 0.453 | 0.828 | 0.681 | 0.792 |
| Linear SVC | 0.644 | 0.848 | 0.685 | 0.823 |
| Multinomial Naive Bayes | 0.453 | 0.828 | N/A | N/A |
| Bernoulli Naive Bayes | 0.453 | 0.828 | N/A | N/A |
| BERT | 0.453 | 0.828 | N/A | N/A |
| Keras | 0.578 | 0.843 | N/A | N/A |

**Designing a UI**

The UI (link to UI code) was designed using flask since the python framework allows for easy integration of our models. To start the UI on the localhost, the steps listed in the UI folder

(link to instructions to start UI) must be followed. The UI has six endpoints, the home page which is the detector, dashboard page which connects to grafana to show data visualizations. There is a description page, add data page, and two post endpoints that take in user-entered data from the forms. The predict endpoint takes in the user-entered text and runs the model's predict function to display the classification output. The app.py has an instance of the prediction model and when the user enters text, it is vectorized using the TfidfVectorizer method. The predict method is called on the processed data, and the classification result is displayed on the UI. The label 1, means the model predicts the entered text as islamophobic and 0 as not. The addDataForm endpoint allows the user to enter labeled data if they choose to contribute to the data collection. The user enters the text, source, and labels to submit, and the UI redirects them to the grafana dashboard to see data visualizations on the entered data. The views rendered for each endpoint can be found under the templates folder, the base.html holds the structure seen on all pages and is extended to all other templates for efficiency and organization. Bootstrap and CSS files are used for styling the UI, bootstrap stylesheets are linked in the base.html and the CSS file is located in the static\css folder**.**

**Data Visualization**

For Data visualization, we decided to go with Prometheus and Grafana. Prometheus is a time series database that stores counts and gauges and keeps their values over time. We added Prometheus as a data source on Grafana to create neat visualizations of the data collected. Not only are we using the data we collected, we also allow the user to enter new data which gets scraped by Prometheus and added to the Grafana visualizations. Such an approach allows the user Islamophobic trends over time.

All of the graphs we included display the total counts of islamophobic and non-islamophobic responses collected. Doing so, allows us to get a better sense of the data. The stack and bar charts are very similar. Their intention is to display the differences between the classifications across different sources. The pie graph shows the same information on a single pie graph to give the user a clear and concise view about the weight of different classification results. Lastly, the bottom graph is a time series graph showing how the total counts are increasing overtime. If needed certain sources can be specified using the Source filter on the top left section of the dashboards page.

We consciously made the decision to run Prometheus and Grafana on docker to isolate them and not mess up the Grafana nor Prometheus states on our machines. Also, the UI site hosted locally via flask Also exposes localhost:9100 which grafana scrapes every second to get the most up to date data.

**Members:**

| | | | |
|---|---|---|---|
| Nelofer | Arjumand | narjum2Illinois.edu | leader |
| Emily | Sallenback | emilygs2@illinois.edu | |
| Mohammed | Alabdullatif | maa27@illinois.edu | |