DATA MINERS
ASHHAR ZAMAN 22881 | MD FARHAD HUSSAIN 23113 | PRASHANT JADHAO 22432

1 Introduction

In the era of widespread social media usage, the accurate determination of user locations, specifically latitude and longitude, from tweets has become increasingly crucial. Various applications, including recommender systems, targeted marketing, population modeling, and addressing negative behaviors like cyberbullying and cyberstalking, heavily rely on precise geographical information. While platforms like Twitter offer the option to geotag tweets, many users do not utilize this feature. The precise prediction of latitude and longitude from tweet data holds immense potential with significant implications for targeted marketing, disaster response coordination, public health monitoring, and combating online harassment. Despite these advantages, a notable challenge arises due to a substantial portion of social media posts lacking precise geotagging or containing unreliable location information. This challenge not only hampers the extraction of meaningful insights from user-generated content but also underscores the urgency in developing innovative approaches for accurate latitude and longitude prediction to address the limitations of negative online behaviors.

2 Objective

The objective is to develop a precise machine learning model for predicting latitude and longitude coordinates using a Twitter dataset, encompassing user locations (latitude-longitude), tweet, tweet counts and tweet times (in UTC). The challenge lies in optimizing the model's accuracy to effectively capture the dynamic nature of user-generated content on Twitter.

3 Methodology

3.1 Data Visualisation

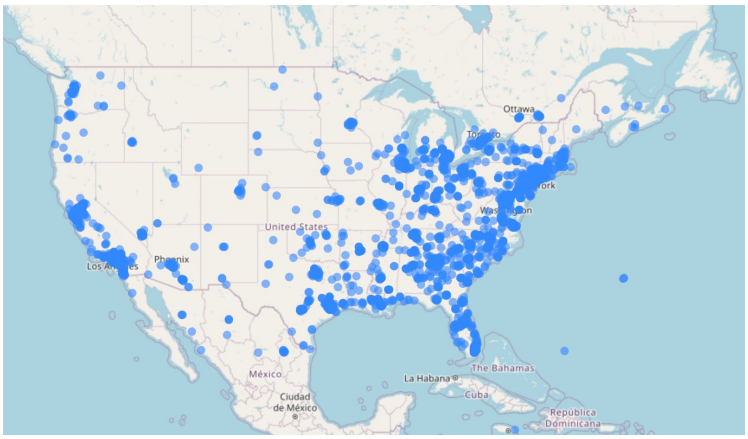
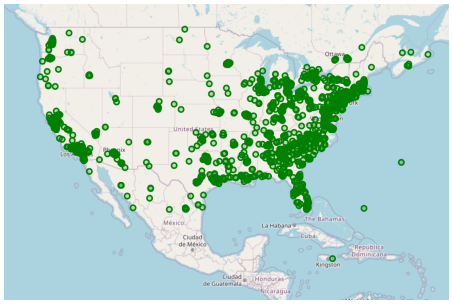


Figure 1: Latitude and Longitude of twitter users

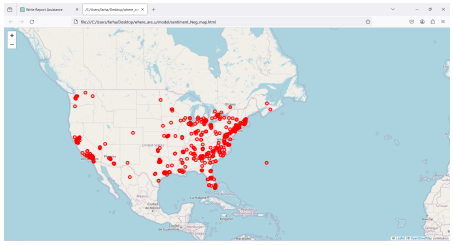
Almost all users in database located in united states.

3.2 Preprocessing

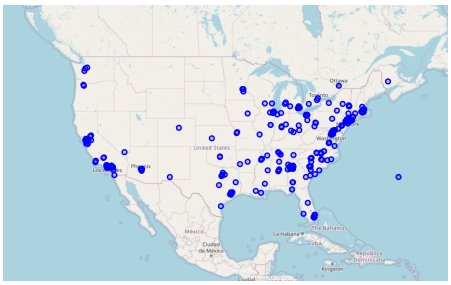
In twitter dataset as target is (latitude and longitude), and feature columns are tweet text, tweet count and tweet timestamps within a specific duration for each unique user, and there is no null entry. The tweet timestamps provide insights into temporal trends, revealing how people use the platform and their locations at different times, For instance, users in metro cities might be more active late at night and less frequent in the morning than user in rural or sub-town



(a) Positive tweets



(b) Negative tweets



(c) Neutral Tweets

Figure 2: Location of different sentiments tweets

location. We're leveraging this temporal information to enhance our model's ability to predict user locations. The tweet text also tells us about people's feelings. So, we're using both time and text info to make our model better at understanding where the user is.

3.3 Feature Engineering

Table 1: Timestamp-based Features

Feature	Description
mean_hour	Mean hour of tweet timestamps
mean_minute	Mean minute of tweet timestamps
std_hour	Standard deviation of hour in tweet timestamps
std_minute	Standard deviation of minute in tweet timestamps
avg_hour	Average time between consecutive tweets
midnight	Number of tweets in midnight
morning	Number of tweets in the morning
afternoon	Number of tweets in the afternoon
night	Number of tweets in the night

Table 2: Tweet Text-based Features

Feature	Description
polarity	Sentiment polarity of text
subjectivity	Subjectivity of text
polarity_subjectivity	Product of subjectivity and polarity of text
Compound	Compound sentiment information of text
Pos	Positive probability of text
Neu	Neutral probability of text
Neg	Negative probability of text

Initially, transformed the tweet timestamps, originally in string format, into a structured list. Following this, extracted the hour and minute components from each timestamp, generating separate lists for both. Subsequently, I computed the mean and standard deviation of the hour and minute lists, providing insightful metrics for our analysis.

Moreover, to gain a nuanced understanding of our users' posting habits, I categorized the tweet timestamps into distinct segments: night, midnight, afternoon, and morning. This segmentation facilitates a more detailed examination of temporal patterns in our data, contributing to the overall depth of our analysis.

In the preprocessing of text-related features, we began by eliminating elements such as user IDs, web links, special characters, and emoticons. Additionally, we addressed misspelled words using TextBlob, and subsequently determined the sentiment polarity and subjectivity of the text. To further assess sentiment, we employed a lexicon-based approach using VADER, calculating the probabilities for positive, negative, neutral, and compound sentiments.

3.4 Machine learning Models

In our approach, we adopted various machine learning models available in the scikit-learn (sklearn) library. These models included Linear Regression, Decision Tree regressor, and Random forest regressor for multiple outputs.

Models	Training MSE	Validation MSE	Training R-squared	Validation R-squared	Test MSE
Linear Regression	116.95	127.98	0.0387	0.021	130.73
Random Forest	89.58	107.22	—	—	124.44
Decision Tree	111.45	118.97	0.0624	0.0634	126.88

Table 3: Model Comparison

4 Model Comparisons

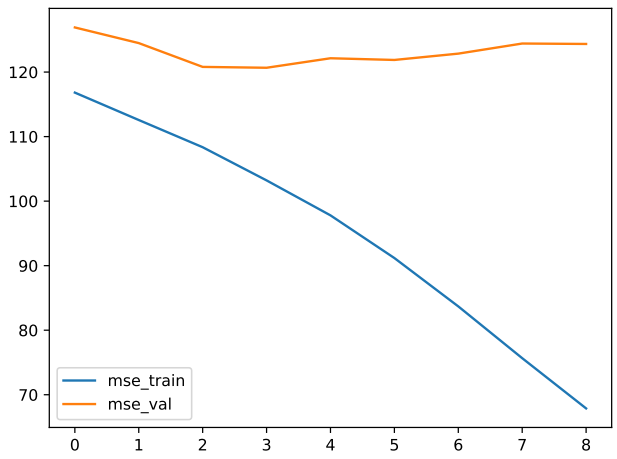
We have implemented Linear regression, Random Forest and Decision tree regressor models .

- Linear Regression: The MSE on both the training and validation sets is quite high, indicating that the linear model might not be capturing the underlying patterns in the data effectively. The R-squared values are also low, indicating poor fit.
- Random Forest: As the depth of the trees increases, the training MSE decreases, while initially, the validation MSE decreases then starts to increase again. This suggests that the model is overfitting the training data as the depth increases, leading to poorer performance on unseen data.
- Decision Tree: The MSE values for the decision tree are similar to those of the random forest, with the model showing signs of overfitting as the depth increases. The R-squared values are also low, indicating limited explanatory power.

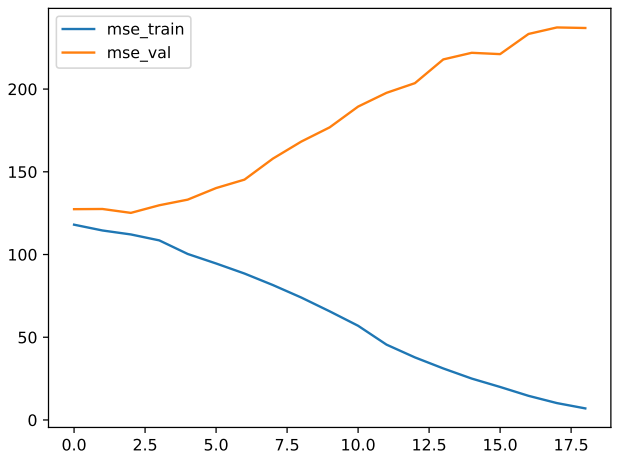
5 Results and Discussions

Performance Plots of MSE error for Training and Validation data: Data is also split into Validation set for Hyperparameter Tuning,Preventing Overfitting and Model selection.

- Random Forest: As we can observe with depth, the Validation error increasing it indicates that the model is likely overfitting to the training data. So we chose optimal depth 3 to train our model with minimum MSE for both Training and Validation data.
- Decision Tree: A similar trend was observed for the Decision Tree Model also. But an increase in Validation error is more prominent in Decision Tree. We chose optimal depth 2 to train the model.



(a) Random Forest



(b) Decision Tree

6 Conclusion

The project highlights the complexity of predicting geo-location from tweet data and the challenges associated with the lack of precise geo-tagging. While machine learning models were employed, achieving high accuracy remains a challenge. It was found out that more the number of features we were taking other than features related to time the accuracy was decreasing. In Random Forest, we observed a minimum MSE error of 10.004 (public score) and 10.140 (private score) for the test data.
Since as number of features are increasing model accuracy is going downhill (curse of dimensionality), applying the Bayesian approach can a solution by considering error term, this will lead to regularization of the model.

The project served as a rich learning experience for the team, encompassing technical aspects of machine learning, data preprocessing, and model evaluation, as well as fostering collaborative problem-solving and a deeper appreciation for the real-world implications of their work.

7 References

- Pre-HLSA: Predicting home location for Twitter users based on sentimental analysis
- Extracting Twitter Data: Pre-processing and Sentiment Analysis
- Sentiment Analysis using the SentiWordNet Lexicon
- What is TF-IDF?
- Text Preprocessing: Removal of Punctuations
- Correcting Misspelled Words in Twitter Text
- TF-IDF for Twitter Sentiment Analysis
- Spelling Correction in Python with TextBlob