HW 5 – Ethical Ai, Privacy

Function used and how I approached them:

1. So first I imported the file and then I was able to to see how the data looks like so then it was pretty similar to what we had for HW4/EC1, so based on that I copy pasted the function I used in the previous homework. Here is the function below.

def workingWithThePartOne(df, model, trainOrTest='train', tfidf=None):

1. This function takes in the df which is dataframe of either the train data or the test data for the part one. The second it takes the model which is LR model. I am passing the model as an argument here so that I can pass the model if the model was trained earlier and return the trained model if it wasn't trained earlier. Then trainortest is used to define if the function will work for the training of the model or testing of the model.
2. It also passes the tf-idf embeddings which are used then we're testing the model, by passing the embeddings which we had trained the model earlier on.
3. This function was very basic, which I've been doing for a while now. In this function, where I'm transforming and fitting the tf-idf vectorizer model in the same function.
4. The function trains the model with the embedding when we're training the model and passing the trainOrTest as train. The function then calculates and prints the accuracy, precision, recall and f1 score for the model for the train data in the same function.
5. For train the function then returns the LR model and the embeddings.
6. For the test part of the function, we then pass the LR trained model and embeddings which have been fit earlier.
7. The model then calculates the accuracy, prevision, recall, f1 score and confusion matrix based on the test data we sent through the function argument.
   (FYI – the training data frame and the test data frame I had already declared before the function call, so that is what I'm passing right now).

With the help of this function, I then create a new dataframe with the name leak_inputs, where I store some of the input which helped me prove that the model is leaking the privacy of the users. Now what they were and how I proved them, I'll explain all of that in the result and analysis section.

Now to correct the above issue, I created the def anonymize_text(text): function.

def anonymize_text(text):

1. This function takes in the text as an input which is nothing but the text we want to anonymize or the column of the dataset text which we've.
2. This function just replaces the @name of the user with @user for each one the places where @{name} have occurred. Now here name is dynamic so that where @

occurred and the word attached to it, I replaced with the @user. To achieve this, I used the regex expression.

With the help of anonymize_text function I change the text for each one of the example in the trainset and the testset and then save them onto the file with the name train_politics_anonymized.csv and test_politics_anonymized.csv. After this, I then import these files and calcaute the result on that. After that I then again use the similar dataframe like the leak_inputs I defined earlier, to drive home the point that the leak has been fixed.

Part 2

Now the person requests us to remove all the words with the capital letters to think that this would remove all the privacy concern. So to achieve this, this is what I did. I first created the function:

def remove_names(text):

1. This function takes in the example of the test/train dataset to remove each one of the capitalized letters.
2. After splitting each word, I iterate though each one of the word and remove the one which has an uppercase letter as the first character in the word.

After this I then saved the text for both the train file and the test file into the train_politics_removed.csv and test_politics_removed.csv respectively.

I the train the model on the train file by using the function workingwithpartone and then use the train model and the tfidf embeddings to test the model with the test data by again utilizing the workingwithpartone function.

After getting the result I was able to prove, that accuracy decreased (As to why that decreased and what the decrease with, we're going talk about that in the result and analysis section).

To balance privacy and accuracy, I improved the anonymization from Part 1(b). Instead of replacing all usernames with @user, I now assign each one a unique placeholder like @user1, @user2, etc. I apply this only to actual usernames (words starting with @), not capitalized words like India or America, since those are important for classification. This avoids the loss of meaningful context while still anonymizing user identities. The function maps each unique username in a sentence to a different placeholder, keeping the structure intact and the model effective.

def better_anonymized_text(text):

1. This function takes in the text i.e. example from the train/test dataset and then checks all the usernames in the text and stores wach of the user unique value in the map.
2. As map is the key value pair, so value for each one of them I store as user1, user2, user 3 etc.
3. I then replace these values in the text example. This helps me achieve anonymity for each one of the text and holds the conversational contextual value in the text as well.

With the help of above function, applied to each one of the example in the dataset, I then save the dataset for both the test and train data. I then train the model using the flexible ability of the workingwiththepart one and then test the train models using the same function to get the result and prove that my method of preserving privacy while maintaining accuracy, precision recall worked.

Result and Analysis:

Part 1(a)

To begin with, I trained a logistic regression model using a TF-IDF vectorizer without applying any preprocessing to the text, as instructed. The model was trained and tested on the provided dataset. Below are the scores obtained for both the training and the testing sets.

Training and Testing Scores

Dataset	Accuracy	Precision	Recall	F1 Score
Train	0.94	0.95	0.93	0.94
Test	0.88	0.89	0.88	0.88

The results show that the model performs quite well on both training and testing data. The scores indicate that the model is able to learn the patterns effectively from the training data and generalize well to the test set.

After training, I moved to the part where I had to check for privacy leaks using known usernames. The model was probed using input texts containing usernames: @WilliamD, @TinaK, and @DonaldW, with knowledge about each of their political inclinations. William is associated more with Democrats, Tina with Republicans, and Donald is politically neutral.

I created small example inputs with combinations of usernames and political keywords and ran them through the trained model using .predict() and .predict_proba().

Probing Inputs and Model Predictions

Input Text	Predicted Label	Probability [Rep, Dem]
@WilliamD loves change	1 (Democrat)	[0.000, 1.000]
@TinaK loves america	0 (Republican)	[0.998, 0.002]
@DonaldW loves change and america	1 (Democrat)	[0.366, 0.634]
@WilliamD	1 (Democrat)	[0.000, 1.000]
@TinaK	0 (Republican)	[1.000, 0.000]

From the results, it's clear that the model is storing and relying on the usernames themselves for prediction. For example, the model confidently predicts @WilliamD as Democrat with a probability of 1.0 even when there is no political content in the text. The same behavior is seen with @TinaK, which is classified as Republican with near 100% confidence. This shows that the model has memorized associations between certain usernames and their political alignments, which is a clear privacy leak.

This confirms that removing just the UserID column is not enough, as the text still contains usernames that the model can latch onto. So even though the data seems anonymized on the surface, identity-based information is still being leaked through the model.

Part 1(b)

After confirming that the model was leaking private information through usernames, I moved forward to fix the issue by anonymizing all usernames in the dataset. For this, I assumed any word that contains the @ symbol is a username. I created a function called anonymize_text() which goes through each text and replaces any username with a common placeholder @user.

The reasoning behind this was to ensure that the model no longer sees unique identifiers like @WilliamD or @TinaK, which were being memorized earlier. Instead, all usernames were masked in the same way. This protects user identity and prevents the model from directly associating users with specific political leaning.

Once the anonymization was done, I created two new datasets (train_politics_anonymized.csv and test_politics_anonymized.csv) and retrained the logistic regression model on the new anonymized data.

Training and Testing Scores After Anonymization

Dataset	Accuracy	Precision	Recall	F1 Score
Train (Original)	0.94	0.95	0.93	0.94

Dataset	Accuracy	Precision	Recall	F1 Score
Test (Original)	0.88	0.89	0.88	0.88
Train (Anon)	0.93	0.94	0.92	0.93
Test (Anon)	0.86	0.87	0.86	0.87

As we can see from the table, the scores dropped slightly, but overall the model still performs well, and the accuracy and F1 score remained high. The small drop is expected because we removed specific name identifiers which the model previously relied on. But this is the trade-off between privacy and performance, and the current model still generalizes well.

To confirm that the privacy leak was fixed, I repeated the same test inputs used in Part 1(a) with the new model trained on anonymized data. This time, the model was not confident about the political alignment based only on usernames.

Input Text	Predicted Label	Probability [Rep, Dem]
@user loves change	1 (Democrat)	[0.306, 0.694]
@user loves america	0 (Republican)	[0.649, 0.351]
@user loves change and america	1 (Democrat)	[0.424, 0.576]
@WilliamD	0 (Republican)	[0.62, 0.38]
@TinaK	0 (Republican)	[0.62, 0.38]

From the predictions above, we can clearly see that the model no longer gives extreme confidence values based on just the usernames. Earlier the model predicted @WilliamD and @TinaK with full confidence (1.000 and 0.000), but now it gives moderate probabilities which shows it is not relying on those names anymore. This means the privacy leak has been resolved.

The slight drop in accuracy can be explained by the fact that the model was previously overfitting on certain names, and now it must rely more on actual political words and patterns in the text. This is a more reliable and privacy-respecting model.

<u>Part 2</u>

After fixing the username-related privacy leak in the previous part, the next step was to further explore how increased anonymization affects model performance. The idea proposed by coworkers was to remove all names from the text, where a "name" is defined as any word starting with a capital letter. I had concerns about this approach since capitalized words can include important nouns such as locations, organizations, or keywords like America, India, or Congress, which could influence the model's decision-making.

So to test the impact, I first removed all capitalized words from the already anonymized files (train_politics_anonymized.csv and test_politics_anonymized.csv) and trained a new logistic regression model on this “removed” dataset.

Training and Testing Scores After Removing Capitalized Words

Dataset	Accuracy	Precision	Recall	F1 Score
Train (Original)	0.94	0.95	0.93	0.94
Test (Original)	0.88	0.89	0.88	0.88
Train (Anon)	0.93	0.94	0.92	0.93
Test (Anon)	0.86	0.87	0.86	0.87
Train (Removed)	0.91	0.93	0.89	0.91
Test (Removed)	0.81	0.82	0.81	0.82

From the scores, it’s clear that this method caused a noticeable drop in both the training and test metrics. The F1 Score dropped from 0.87 to 0.82 on the test data. This drop makes sense because we are removing meaningful words that carry political or contextual signals. Removing all capitalized words takes out named entities and key topic indicators, which makes it harder for the model to correctly classify the text.

To fix this and still maintain privacy, I created a better anonymization method. In this version, I preserved the actual content and only improved the username anonymization. Instead of replacing all usernames with just @user, I made each username unique within the same example. So if a sentence had @WilliamD and @TinaK, they would become @user1 and @user2. This helps retain the conversation flow and makes it clear who is referring to whom, while still anonymizing the identity.

No other words in the text were changed. I left all capitalized words intact because of the earlier drop in performance.

Training and Testing Scores After Better Anonymization

Dataset	Accuracy	Precision	Recall	F1 Score
Train (Better Anon)	0.94	0.95	0.93	0.94
Test (Better Anon)	0.86	0.87	0.86	0.86

The model trained on the better anonymized data performed almost identically to the previous anonymized version but without any loss of context. This confirms that we can preserve privacy by anonymizing usernames in a smarter way, while not hurting the model's ability to learn and predict accurately.

So overall, removing capitalized words turned out to be a bad anonymization method that hurts performance, while our better method maintains both privacy and accuracy.

Conclusion:

Through this experiment, I first showed that the model was leaking private information by relying heavily on usernames to make predictions. I then fixed this by anonymizing all usernames, which successfully removed the privacy leak while maintaining good model performance. In Part 2, I tested a more aggressive anonymization method proposed by coworkers, which involved removing all capitalized words. This led to a noticeable drop in accuracy, confirming it as a poor choice. Finally, I implemented a better anonymization strategy that uniquely replaced usernames while keeping the rest of the text intact. This method balanced privacy and performance effectively, showing that thoughtful anonymization can protect identities without compromising model quality.

Difficulties faced:

The overall coding and understanding of the task was straightforward. The main difficulty was in Part 2 while designing a better anonymization method.

- It was challenging to think of a way that improves on simple masking without hurting context.
- After discussing the issue with Professor Russet, he suggested a direction that helped me finalize the idea of assigning unique usernames like @user1, @user2, which led to a better anonymization approach that preserves both privacy and context.