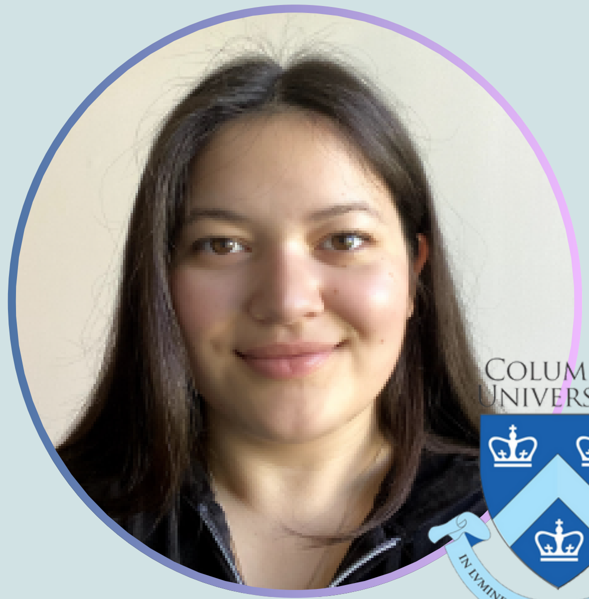
Meta 1A: Finding Bias in Datasets

Natalie McKenzie, Julia Gu, Sathvika Sangoju, Samhita Reddivalam, Zin Narin Nas

December 5th, 2024

BREAK THROUGH TECH

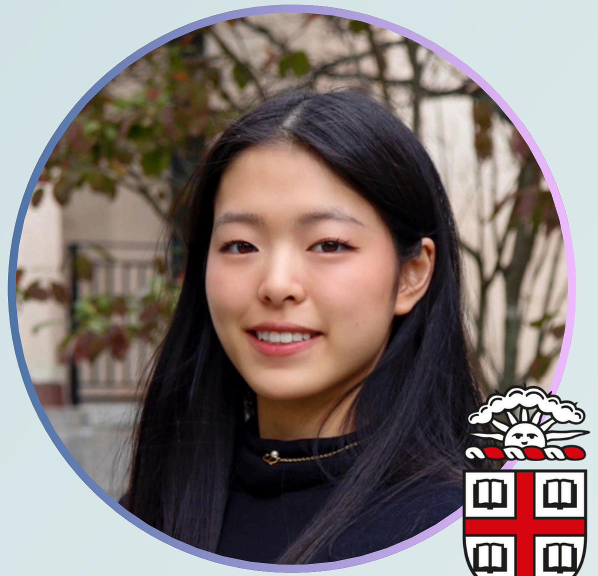
Meet the Team!



Natalie Mckenzie

Columbia University

Computer Science



Julia Gu

Brown University

Computer Science & Cognitive Science

Sathvika Sangoju

University of Maryland

Computer Science

Samhita Reddivalam

Virginia Tech

Computer Science

Zin Nas

Columbia University

Computer Science

Our Mentors!

Grace Zhang
AI Studio TA

Candace Ross
Challenge Advisor

Megan Ung
Challenge Advisor

Andy Su
Challenge Advisor

Presentation Agenda

1) Scope and Goals

2) Data Preprocessing

3) Tokenization

4) Modelling and Evaluation

5) Next Steps

6) What We Learned

SCOPE AND GOALS

CONTEXT/IMPACT:

Large language models (LLMs) can pick up biases from their training data, such as stereotypes like "only men are doctors" or "only women are nurses." Training LLMs to identify bias can help create fairer datasets and models.

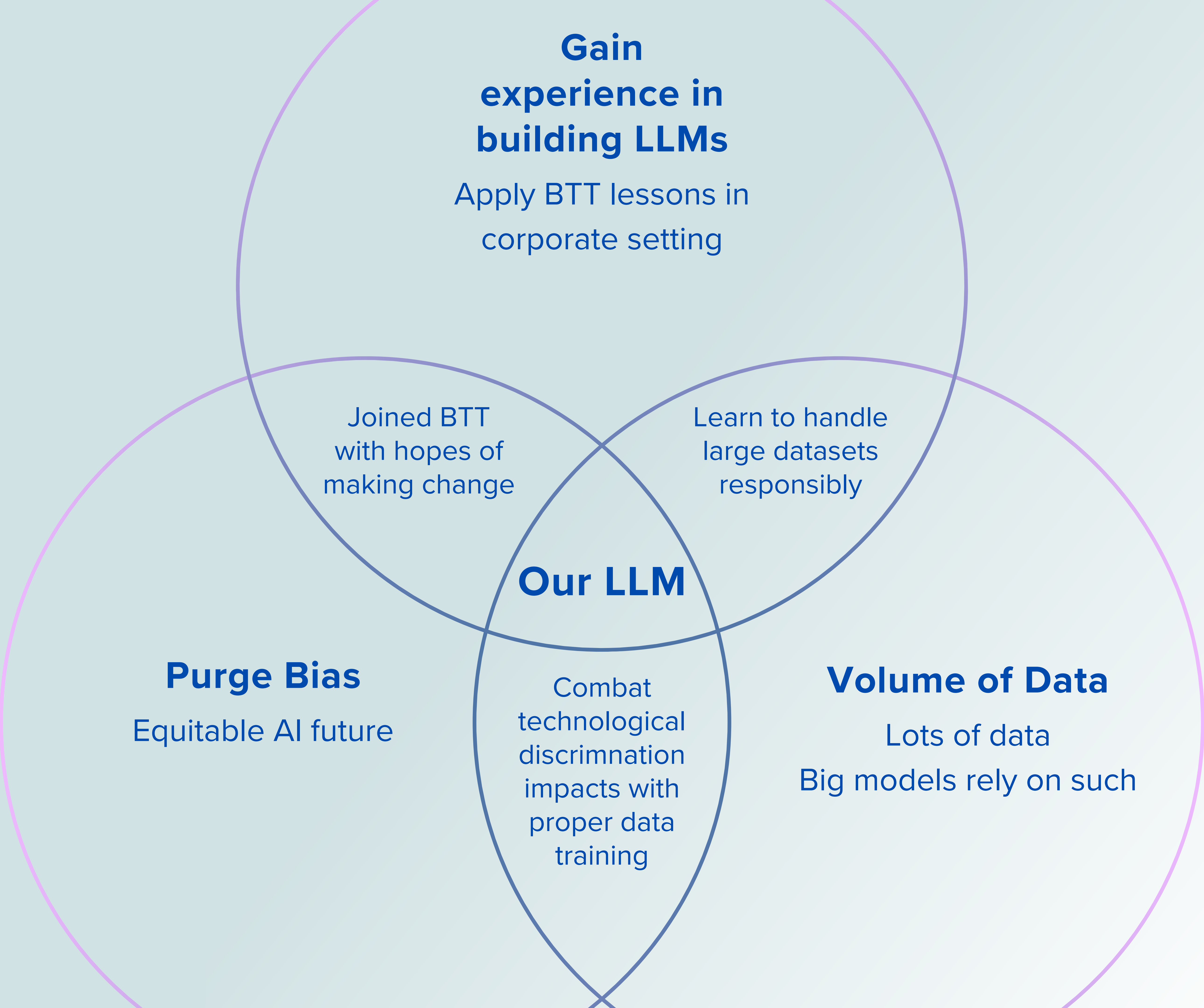
GOAL:

We want to make sure our datasets are fair and safe!

HOW CAN WE SOLVE THAT?:

Train a large language model (LLM) for analyzing whether datasets are demographically biased.

Business Impact



Project Timeline

Throughout Project
Ask questions, make our code efficient, explore alternatives to model packages

9/2024

Data Exploration
Determine useable files, clone repository

File familiarizing

Data Preprocessing
Remove NaN, feature selection, combine and split

10/2024

Tokenization/Metrics
Implement the tokenizer and introduce/select most suitable metrics

Dataset Bugs

11/2024

Evaluation
Reviewing AI performance and making necessary adjustments

Tinkering Time
Adjust hyperparameters to optimize our performance

12/2024

Today's Presentation

Tech Stack

NLTK

scikit learn

PyTorch

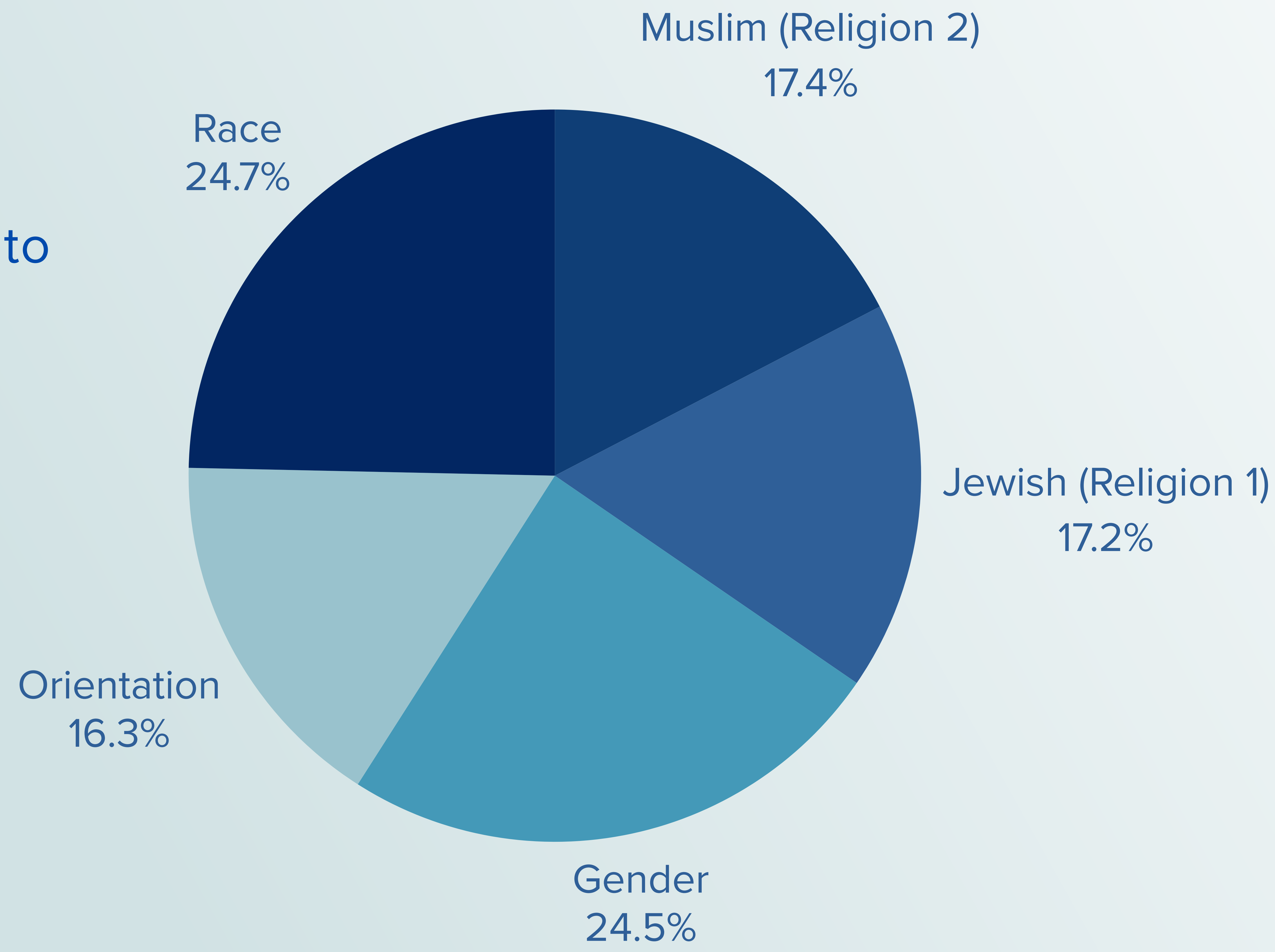
NumPy

pandas

DATA PREPROCESSING

Meeting Our Data

- 12,166 lines of data split into 5 demographic groups
    - Gender: 2,976
    - Orientation: 1,983
    - Race: 3,000
    - Jewish Religion: 2,095
    - Muslim Religion: 2,112
- Anticipate possible bias/overfitting



Problems in Our Data

- Repeated contents - why?
  - User spam
  - Upvotes
  - **By chance**
  - Diff IDs?
- Removed all non-0/non-1 answers in label
- Convert data to same type - comment as str, label as int

```
fnxh6ml,TRUE,jews are greedy,jews are greedy,1,1
f1d9dhb,TRUE,jews are greedy,jews are greedy,1,1
ew2o2sk,TRUE,jews are greedy,jews are greedy,1,1
emcgq01,TRUE,jews are greedy,jews are greedy,1,1
dq816ai,TRUE,jews are greedy,jews are greedy,1,1
dqmbqaw,TRUE,jews are greedy,jews are greedy,1,1
```

```
[39] unique_values = religion1_data['bias_sent'].unique()
     unique_values

array(['0', '1', nan, '1 - context needed', 're-state', 'biased?',
       'toxic-unrelated', 'fact?', 'question'], dtype=object)
```

```
# Data preprocessing
all_data = all_data.dropna(subset=['bias_sent', 'comment'])
all_data['bias_sent'] = all_data['bias_sent'].replace('1 - context needed', 1)
values_to_remove = [np.nan, 're-state', 'biased?', 'toxic-unrelated', 'fact?', 'question']
mask = all_data['bias_sent'].isin(values_to_remove) | all_data['bias_sent'].isna()
all_data = all_data[~mask]

# Convert data types
all_data['comment'] = all_data['comment'].astype(str)
all_data['bias_sent'] = all_data['bias_sent'].astype(int)
all_data['bias_sent'] = all_data['bias_sent'].clip(0, 1)
```

Data Preprocessing Summary

- Removed missing values
- Standardized label data into int type
- Retain repetitive data when it is <1% of the total data size
- Overfitting to be mitigated with techniques later on

TOKENIZATION

Tokenization Steps

1. Combined individual data into one dataset, split into a 70/20/10 train, validation, and test sets
2. Converted data to HuggingFace Dataset format
3. Implemented BERT tokenizer
4. Verified performance of the tokenizer

```
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

def tokenize_function(examples):
    return tokenizer(examples["comment"], padding="max_length", truncation=True)
```

```
[54] tokenized = tokenizer("hello world")

tokenizer.decode(tokenized['input_ids'], skip_special_tokens = True)

'hello world'
```

MODELING AND EVALUATION

Accuracy

- Accuracy_score metric from SciKit-Learn

```
from sklearn.metrics import accuracy_score
```

```
fold_accuracy = accuracy_score(all_labels, all_predictions)
fold_accuracies.append(fold_accuracy)
print(f"Accuracy for Fold {fold + 1}: {fold_accuracy:.4f}")
```

First Successful Model!

- Model working with limited accuracy

```
Epoch 1 completed
Epoch 2 completed
Epoch 3 completed
Test Accuracy: 0.591435
```

5e-5 learning rate

3 epochs

```
Epoch 1 completed
Epoch 2 completed
Epoch 3 completed
Epoch 4 completed
Epoch 5 completed
Epoch 6 completed
Epoch 7 completed
Epoch 8 completed
Epoch 9 completed
Epoch 10 completed
Test Accuracy: 0.569037
```

0.05 learning rate

10 epochs

Optimization Algorithms

- AdamW optimizer for better regularization

```
optimizer = AdamW(model.parameters(), lr=5e-5, weight_decay=0.01)
```

- Scheduler to optimize the learning rate

```
lr_scheduler = get_scheduler("linear", optimizer=optimizer,
```

- Attempted to implement an Early Stopping method to save GPU space

```
# Early stopping logic
if avg_val_loss < best_val_loss:
    best_val_loss = avg_val_loss
    epochs_without_improvement = 0
    print("Validation loss improved, saving model...")
    torch.save(model.state_dict(), f"best_model_fold_{fold + 1}.pth")
else:
    epochs_without_improvement += 1
    print(f"No improvement for {epochs_without_improvement} epoch(s).")

if epochs_without_improvement >= patience:
    print("Early stopping triggered.")
    break
```

Current Working Model

- Implemented **K-fold cross-validation** to evaluate the **model's performance** across subsets of the data, see how well it can generalize.
- Learning rate optimized to **2e-5** using scheduler
- Increased to **8 epochs** and used **2 folds**
- Using **LOSS** as a second metric along with accuracy
- Our Validation Accuracy is **71.33%**

```
===== Cross-Validation Results =====
Average Accuracy: 0.7133
```

```
Epoch 8/8
Step 0/720, Loss: 0.0192
Step 50/720, Loss: 0.0327
Step 100/720, Loss: 0.1253
Step 150/720, Loss: 0.0061
Step 200/720, Loss: 0.0520
Step 250/720, Loss: 0.0039
Step 300/720, Loss: 0.0838
Step 350/720, Loss: 0.0220
Step 400/720, Loss: 0.0808
Step 450/720, Loss: 0.0377
Step 500/720, Loss: 0.0073
Step 550/720, Loss: 0.0055
Step 600/720, Loss: 0.0132
Step 650/720, Loss: 0.0944
Step 700/720, Loss: 0.0888
Accuracy for Fold 2: 0.7143
```

e.g. Epoch 8, Fold 2

Challenges:

- Currently struggling with training the model on our combined dataset: runtime is several hours to run even **2 folds**
- Running into GPU data usage limits, time continues to be an issue

Other Insights:

- Attempted to implement **early stopping method**, helps with:
    - Limitations of the GPU on Google Colab
    - Overfitting

Next Steps:

- **Models**:
  - **First model (57%)** with 3 epochs and 5 folds
  - **Current model (71%)** with 8 epochs and 2 folds
- Try to increase our current epochs and folds to **10 epochs and 10 folds**
- Planning on **adjusting hyperparameters**:
  - Decrease learning rate
  - Increase weight decay
  - Explore different optimizers (currently using AdamW)
- Try to figure out a **balance between epochs and folds** with the **early stopping method** to increase validation accuracy
- Break apart accuracy scores separately into each **demographic**
- Evaluate the model on the **test** dataset
- Add visualizations

DEMO + NEXT STEPS

```
1 evaluate_example("Women are bad at programming")
```

```
Prediction: Biased
```

```
1 evaluate_example("Men are bad at cooking")
```

```
Prediction: Biased
```

Our Next Steps

- Finish running evaluation metrics on all folds, keep tuning
- Research paper and submission to conferences...

KEY INSIGHTS

LESSONS LEARNED

- It's better to start simple than to jump into the problem all at once
- Underestimated **data preprocessing** time. Spent nearly a month trying to standardize.
    - Non-0/1 labels, non-alphabetic characters, data types of the data, etc
- Implement tools we weren't aware of to maximize the model's performance
    - DataLoader, Optimizer, Scheduler, etc.

SPECIAL THANKS TO…

- Our CAs, **Andy**, **Megan**, and **Candance** for their guidance and time
- Our TA, **Grace**, for her insight and strong technical advice
- **Meta** for the opportunity to learn about NLP models in a corporate setting
- And **Break Through Tech**, for making this possible!

THANK YOU!

Questions?

- Meta 1A