1. **Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.**

Artem Volgin is currently a PhD candidate in Social Statistics at the University of Manchester, UK. His main research interests revolve around applications of Network Analysis and Natural Language Processing methods. Artem has versatile experience in working with real-life data from different domains and was involved in several data science projects at the World Bank and the University of Oxford. He actively participates in data science competitions consistently achieving top places.

Ekaterina Melianova is currently a PhD candidate in Advanced Quantitative Methods at the University of Bristol, UK. She focuses on applying innovative statistical techniques to tackle real-world social issues, with a particular emphasis on social epidemiology and education. She has contributed to projects at the World Bank and various market research organizations. Ekaterina actively participates in data analytics competitions continuously seeking opportunities to apply her skills in novel substantive and methodological domains.

2. **What motivated you to compete in this challenge?**

There were many factors in play: the desire to try the latest development in Large Language Models in practice, the flexibility of the challenge format allowing us to experiment with ChatGPT-based solutions beyond prediction tasks, an interesting dataset with a lot of hidden value to unpack, and an opportunity to contribute to a socially impactful task.

3. **High level summary of your approach: what did you do and why?**

Using a downsampling method with ChatGPT and ML techniques, we obtained a full NEISS dataset across all accidents and age groups from 2013-2022 with six new variables: fall/not fall, prior activity, cause, body position, home location, and facility. We showed how these new indicators vary by age and sex for the fall-related cases and how they impact the likelihood of falls and post-fall hospitalization of older people. We also revealed seasonal and yearly trends in falls and provided a comparative age-based perspective, underscoring the value of the full ED narratives dataset in studying falls of older adults.

4. **Do you have any useful charts, graphs, or visualizations from the process?**

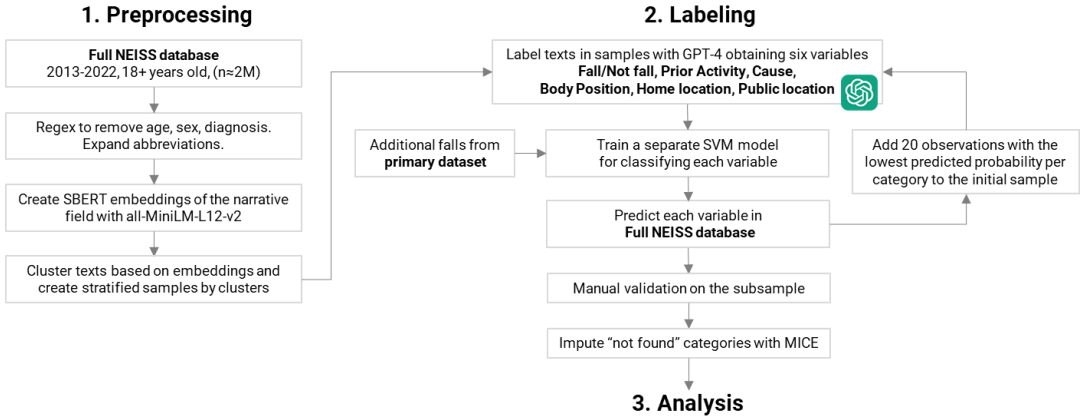

Figure 1. Data Processing Pipeline.

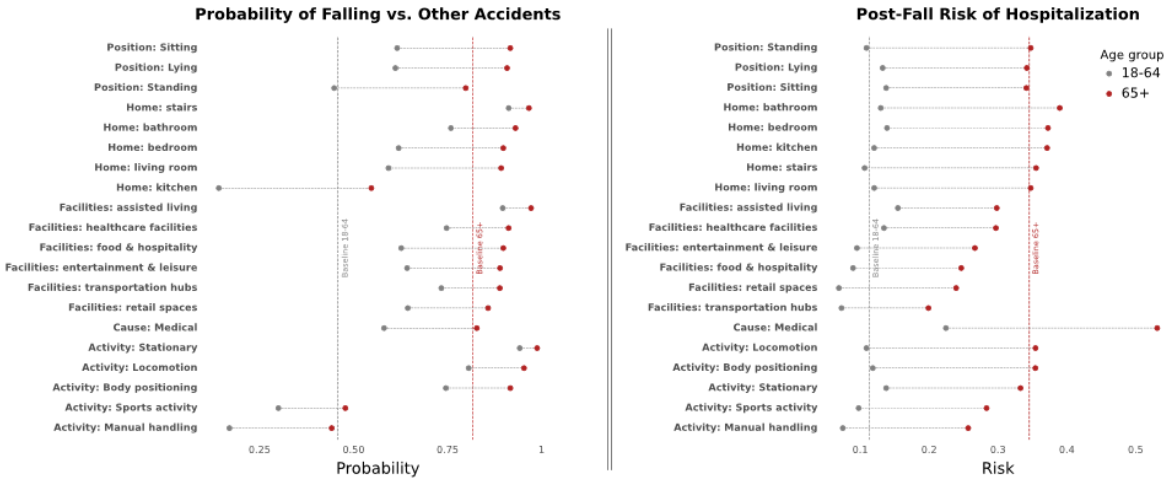

Figure 2. Probability of Falls vs. Other Accidents and Post-Fall Risk of Hospitalization for Older Adults

5. **Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.**

```
def queryGpt(text_sample, prompt, model):
    task = f'''
    You are a text classification model.
    You are given a text desciption of an accident.
    Do the following:
    {prompt}
    '''

    input_text = f'''
    Text: "{text_sample}"
    Your response:
    '''

    url = "https://api.openai.com/v1/chat/completions"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {openai.api_key}"
    }
    params = {
        "model": model,
        "messages": [{"role": "system", "content": task},
                     {"role": "user", "content": input_text}],
        "temperature": 0,
        'n':1
    }
    retries = 5
    timeout = 20

    while retries > 0:
        try:
            response = requests.post(url, headers=headers, data=json.dumps(params), timeout=timeout)
            response.raise_for_status()
            gpt_output = response.json()['choices'][0]['message']['content']
            break
        except (requests.exceptions.RequestException, json.JSONDecodeError) as e:
            retries -= 1
            print(f"Request failed with error: {e}")
            time.sleep(20)

    if retries == 0:
        gpt_output = '===FAIL==='
        print('===FAIL===')

    return gpt_output
```

"queryGpt" provides a way to query ChatGPT using API with a predefined general "task" of the model, "prompt" with a coding scheme and examples for specific variables, and "text_sample" with a selected narrative. An argument "model" allows to query a particular OpenAI model (gpt-3.5, gpt-4, etc.). Overall, this function provides an efficient way to label a sample of text narratives.

```python
def fitSVM(df, target, validate=False, sampling=False):

    print(target)

    # Data
    y = df[target]
    X = df[[col for col in df.columns if col.startswith('emb')]]

    if sampling:
        # SMOTETomek sampling
        smt  = SMOTETomek(random_state=RANDOM_SEED)
        X, y = smt.fit_resample(X, y)

    # Model
    model_svm = svm.SVC(random_state=RANDOM_SEED, probability=True, class_weight='balanced') #

    if validate:

        # Define custom scorer
        f1_scorer = make_scorer(f1_score, average='weighted')

        # Perform 5-fold cross-validation
        scores = cross_val_score(model_svm, X, y, cv=5, scoring=f1_scorer)
        print(np.mean(scores))

    # Retrain model on the full dataset
    model_svm.fit(X, y)

    return model_svm
```

The function “fitSVM” fits a separate SVM model using a sample dataset with a target variable labelled by ChatGPT and columns with text embeddings encoding information about text narratives in this sample. One might try to use SMOTETomek sampling activating with parameter “sampling” to boost the model”s performance for the imbalanced targets. The 5-fold cross-validation of the model performance is available using the argument “validate”. It is worth noticing that this validation will only show the agreement between ChatGPT labelling and the SVM classifier; the best way to get the correspondence between the SVM classifier and the “ground truth” will be to code a sample of narratives manually.

```python
def runMICE(df):

    start_time = time.time()

    # Create kernel.
    kds = mf.ImputationKernel(
      df,
      save_all_iterations=True,
      random_state=RANDOM_SEED
    )

    # Run the MICE algorithm
    kds.mice(5)

    # Return the completed dataset.
    df_imputed = kds.complete_data()

    print("--- %s seconds ---" % round((time.time() - start_time), 2))

    return df_imputed
```

In our classification, we addressed the lack of details in some narratives with the category “not defined …”. While such “not defined” categories can still be utilized for the regression analysis, they might introduce bias since certain hospitals or patient demographics have a higher probability of incomplete narratives. One viable approach is imputing these categories using both textual and non-textual data. In our methodology, we considered “not defined” categories as NAs and employed the miceforest algorithm for imputation utilizing non-textual variables such as “product,” “diagnosis,” “body part”, etc., as predictors for the missing categories. This method helped us partially mitigate biases stemming from disparities in the narrative descriptions. The function “runMICE” run imputation for the given dataset containing NAs instead of “not defined” categories.

6. **Please provide the machine specs and time you used to run your model.**
    - CPU (model): Intel Core i7-8750h @ 2.20ghz
    - GPU (model or N/A): NVIDIA GeForce RTX 2070
    - Memory (GB): 32 GB
    - OS: Windows
    - Train duration: 4 hours for ChatGPT labelling, 1 min for SVM training
    - Inference duration: 10 min for the inference on the full dataset

7. **Anything we should watch out for or be aware of in using your notebook (e.g. code quirks, memory requirements, numerical stability issues, etc.)?**

To label a sample with ChatGPT, you will need to specify your OpenAI API key at the start of your script.

8. **Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?**

No

9. **How did you evaluate the quality of your analysis and insights, if at all?**

We manually coded a stratified sub-sample from the full dataset, selecting 20 observations from each category in each variable. Two independent coders marked the model-generated categories as correct or incorrect. The overall agreement between human coders and the ChatGPT + ML pipeline was high, with accuracies ranging between 80% and 97%, depending on the variable and coder. Moreover, we cross-checked the classification of falls/non-falls on the segment of the primary_dataset that was not part of the training. Notably, 99.5% of falls in the primary_dataset were correctly classified by the model.

10. **What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?**

We found that open-source models like Llama were not as effective as GPT-3.5 and GPT-4. Specifically, GPT-4 showed superior performance over GPT-3.5, especially when it came to labeling multi-categorical variables. Binary variables were generally more accurately labeled than the multi-categorical ones, and categories such as "Other" and "Not defined" were particularly challenging for our pipeline to be classified correctly. Drawing multiple answers from the GPT API as opposed to a single response did not enhance the accuracy. Furthermore, using advanced prompt engineering techniques, like chain-of-thoughts, had minimal impact on our results. As for the classification phase, we observed that intricate ML models did not offer a significant advantage, i.e., a simple SVM delivered the best results.

11. **If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?**

Future work may include carrying out a sensitivity analysis, possibly by changing the sequence of examples in the prompt or by gauging the consistency of answers when making slight adjustments to the prompt. Additionally, there is potential in finetuning the transformer model for the classification stage instead of relying on embeddings paired with SVM. Testing and validation stages can also be expanded to encompass other types of accidents.

12. **What simplifications could be made to run your solution faster without sacrificing performance?**

Switching entirely to binary variables or reducing the number of categories in nominal variables might reduce the required sample size for initial labelling with ChatGPT. Additionally, using TF-IDF features

instead of SBERT embeddings can speed up the pipeline without a significant drop in model performance.