II. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

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.**

Ans. We are a team of four members (Seahawk).

1. Md Umarfaruque Bio

Umar is a senior Data Science Associate Consultant at ZS in New Delhi, specializing in the R&D AI domain. With over four years of experience, he has accumulated extensive expertise in leveraging various technologies such as Data Science and Machine Learning to design, implement, and monitor end-to-end pipelines for clinical trial management within the pharmaceutical sector. Collaborating with multiple pharma clients, Umar provides tailored solutions to enhance their processes.
His technical proficiency spans a wide range of fields, including Statistical Modeling, Probabilistic Programming, Operations Research, Natural Language Processing (NLP), Recommender Systems, and Advanced Machine Learning. Proficient in advanced programming languages such as Python, Umar also possesses a solid understanding of R, SQL, and PySpark. Well-versed in version control tools like Git and Bitbucket, he ensures code quality and facilitates collaboration. Additionally, Umar has a deep understanding of probabilistic programming, statistical modeling, operations research (including Linear Programming and Mixed Integer Linear Programming), Bayesian Inference, Bayesian Graphical Networks, Monte Carlo Markov Chain, and large language models.

In his role, Umar strives to harness these technical capabilities to deliver innovative data-driven solutions, ultimately helping pharmaceutical clients optimize their clinical trial processes and make informed decisions.

2. Aayush Khurana Bio

Aayush is a Senior Data Scientist based at the ZS New Delhi office, specializing in the R&D AI domain. Within the AI RDE life sciences space at ZS, he plays a key role in developing advanced data science solutions aimed at accelerating clinical trials, drug discovery, and building NLP-based pipelines. With over 2+ years of experience, Aayush has successfully built machine learning and deep learning solutions across various domains, including Clinical Data Transformation based on CDISC standards (SDTM, ADaMs, TLFs), Clinical Trial data extraction and standardization, and NLP-based pipelines within the pharmaceutical sector. His expertise encompasses different Recommender Systems, Survival analysis (Cox Models), Mixed effect models (GLMMs, non-linear mixed effects models), and handling high-dimensional longitudinal patient-level data.

Proficient in Python, Aayush brings extensive expertise in developing advanced data science solutions. His technical proficiency spans a wide range of fields, including Natural Language Processing, Transfer Learning, Ensemble Learning, and a deep understanding of Deep Learning libraries such as TensorFlow, Keras, and Pytorch. Aayush is adept at creating insightful

visualizations using libraries like Autoviz, Matplotlib, Seaborn, and Plotly, enhancing the interpretability of data-driven insights. His proficiency extends to utilizing Large Language Models toolkits such as HuggingFace, LangChain, OpenAI and LlamaIndex, showcasing his ability to leverage cutting-edge natural language processing capabilities. Well-versed in version control tools like Git and Bitbucket, he ensures code quality and facilitates collaboration.

In his role, Aayush is dedicated to harnessing his technical capabilities to drive innovation in data-driven solution. Aayush's passion for leveraging cutting-edge technologies reflects his ongoing pursuit of delivering impactful solutions in the evolving landscape of data science, thereby helping the pharmaceutical clients, and contributing to the healthcare industry.

3. Gaurav Yadav Bio

Gaurav Yadav holds the position of Senior Data Scientist at ZS and has contributed to the Research and Development unit for over two years. He specializes in crafting and implementing data science solutions within the healthcare sector. Gaurav's expertise spans a wide spectrum, encompassing the optimization of processes for drug discoveries, devising market strategies for clients, and refining study designs for clinical trials using AI.
His proficiency in Data Science revolves predominantly around text data and natural language processing with a focus on various research initiatives, notably exploring the efficacy of Large Language Models in resolving intricate business challenges. Proficient in Python, SQL, and PySpark, Gaurav excels in employing these technical skills to help the clients in making better business decisions. Moreover, Gaurav possesses valuable experience in developing AI-powered products in collaboration with cross-functional teams. His adeptness extends to utilizing code versioning tools such as git, bitbucket, and SVN, facilitating streamlined project management and collaboration.

4. Yash Aggarwal Bio

Yash Aggarwal, a seasoned Senior Data Scientist at ZS Associates, boasts over four years of dedicated service in the Research and Development domain. Specializing in healthcare, Yash has played a pivotal role in optimizing processes for drug discoveries, formulating market strategies, and refining study designs for clinical trials through the application of AI.
Yash's proficiency in Data Science is particularly pronounced in the realm of Natural Language Processing (NLP) and text data analysis. His focus on various research initiatives, including the exploration of Large Language Models' efficacy in resolving complex business challenges, showcases his commitment to innovative problem-solving.
With advanced skills in Python, SQL, and PySpark, Yash excels in implementing technical solutions to enhance decision-making for clients. Beyond his technical prowess, Yash brings valuable experience in collaborating with cross-functional teams to develop AI-powered products. His adept utilization of code versioning tools such as git, bitbucket, and SVN underscores his commitment to streamlined project management and effective collaboration in the dynamic field of Healthcare R&D.

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

Ans. The team was deeply motivated to participate in this challenge due to the profound societal impact of falls among adults aged 65 and older. Recognizing falls as a leading cause of injury-related deaths and the potential for serious injuries, we were driven by a collective sense of responsibility to contribute to the understanding and prevention of such incidents. The challenge's focus on utilizing unsupervised machine learning to extract insights from emergency department narratives aligned perfectly with our team's commitment to leveraging advanced analytics for meaningful social impact.
Our motivation was rooted in the realization that **medical record narratives, though rich in potential insights, are often under-explored due to the challenges associated with manual coding procedures.** The prospect of employing modern machine learning approaches to efficiently extract valuable information from these narratives at scale was both intellectually stimulating and aligned with our team's dedication to leveraging technology for societal benefit.
The team's curiosity extended beyond merely acknowledging the inherent frailty associated with old age; we sought to delve deeper into the nuances of the data to identify additional risk factors and contributors to falls. The goal was not only to conduct exploratory analysis but to contribute substantive insights that could inform targeted interventions for reducing falls among the elderly.
In essence, our motivation was fueled by a genuine desire to make a positive impact on public health by unraveling the complexities surrounding older adult falls. The commitment to understanding and addressing this critical issue not only propelled us through the challenge but also underscored our team's broader mission of applying data science for the betterment of society.

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

Ans. Our approach consists of a two-step methodology to effectively work with the primary dataset. In the initial phase/step, we leverage Large Language Models (LLMs) to create a gold standard dataset, which serves as training data for fine-tuning a classification model. This involves a unique clustering approach, where representative samples are extracted from the dataset utilizing PCA for dimension reduction and KMeans for cluster creation. Subsequently, LlaMA2 and OpenAI's GPT-3.5 are employed for zero-shot classification to extract important information such as the severity of fall, action performed just before fall, and risk factors associated/reason for the fall from the ED visit narrative texts. The reliability of this gold dataset is confirmed through manual validation and extensive Exploratory Data Analysis (EDA).

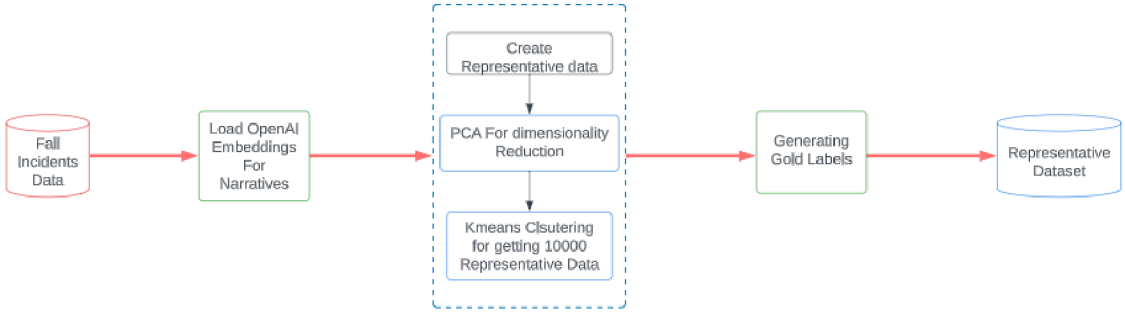
Then in the second step, we fine-tune a DistilBERT model using the golden dataset to classify severity, action before fall, and reason for fall. This fine-tuned model is then applied to predict classification labels for the complete primary dataset thereby enabling us to generate insights.

A distinctive feature of our approach is a novel data sampling strategy, wherein the data is segmented into 10,000 diverse clusters. To ensure maximum diversity in the sample dataset, we select the data point closest to each cluster's centroid. This approach is designed to capture the essence of the original dataset. It was observed that the sampled data followed similar distribution as that of the actual primary data. This is supported by the data analysis performed on the sample data as well as the actual data. Thorough data analysis validates the reliability of our sampling strategy, affirming the consistency between insights derived from the sample and the actual dataset.

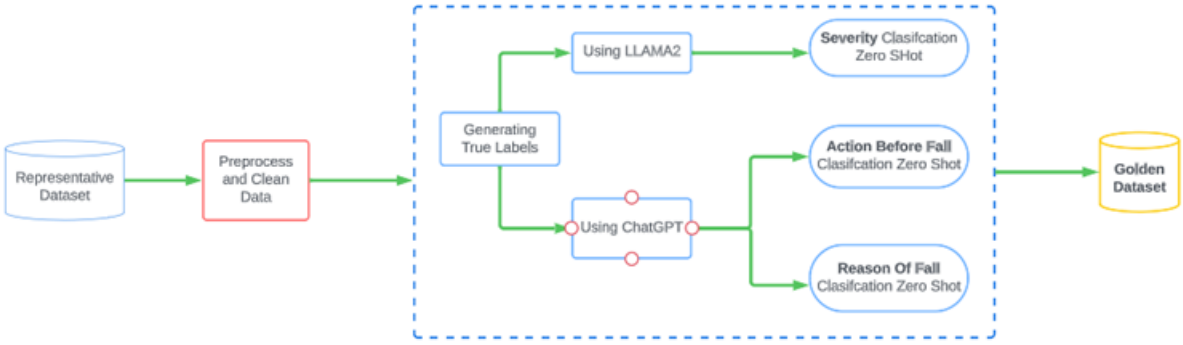
4. **Do you have any useful charts, graphs, or visualizations from the process?**

Ans. Important Graphs and Visualizations from the process are as follows –

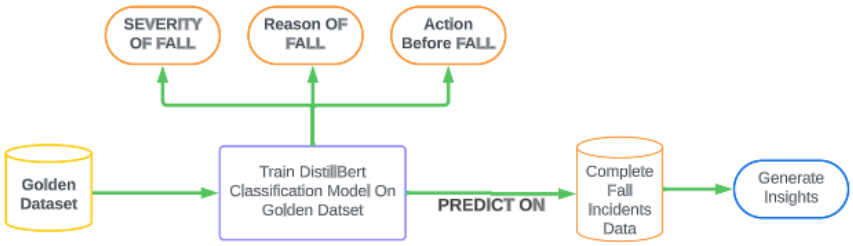
a. Representative Data Creation using Cluster Segmentation: The main idea behind generating clusters was to maximize the diversity between the falls incident data such that the sample data is representative of the primary data. The data was partitioned in 10,000 clusters which is equal to the sample size of the representative dataset. In order to increase the diversity, the data point closest to the centroid of each cluster was chosen and added to the sample dataset hence making the sample data as diverse as possible.



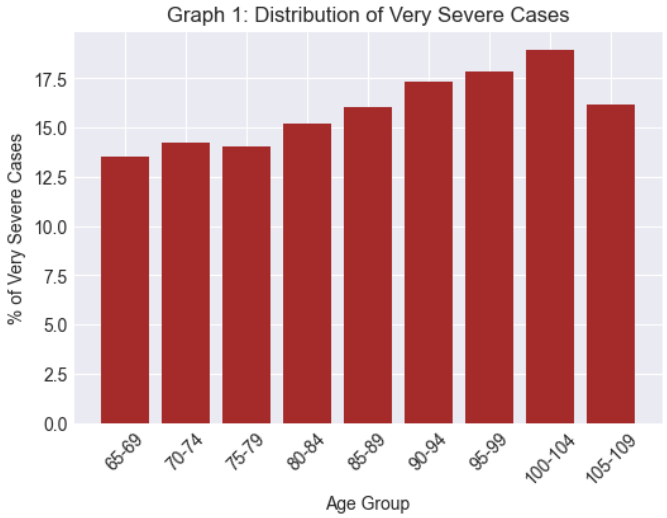
b. Creation of Gold Standard Data using representative data: Various Large language models (LLMs) were employed to generate gold standard data (training data) for fine-tuning the classification model.



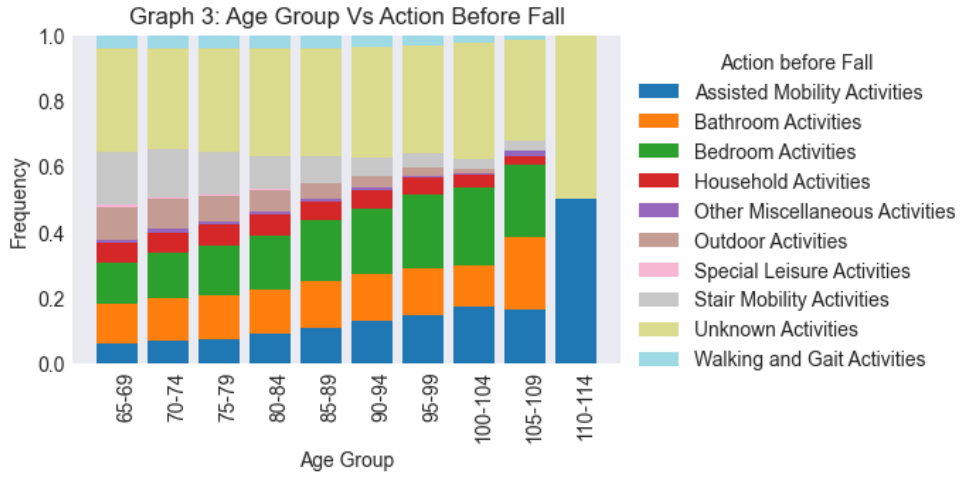
c. Predictions of Severity, Reason of fall & Action before fall on the Complete Dataset: The gold data created above was used to fine-tune a DistilBERT model (for classification), which was then used to generate predictions on the complete primary data.



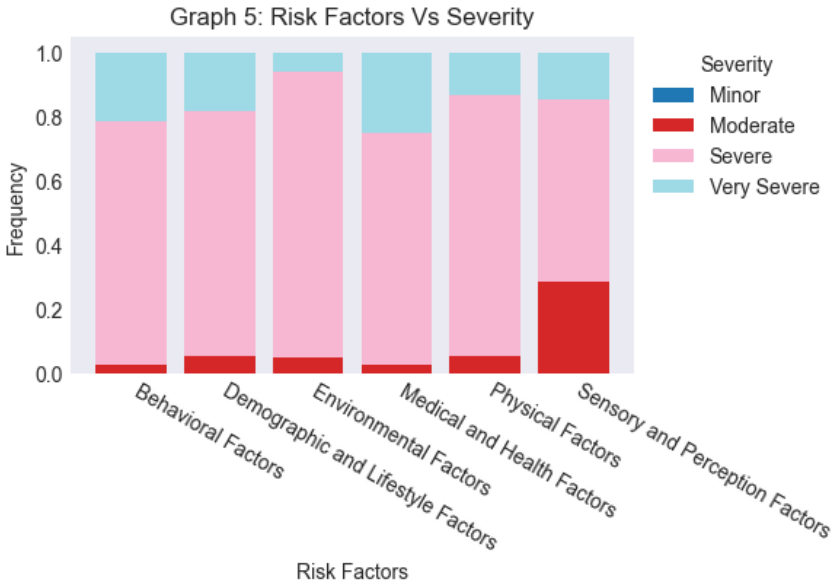
d. Severity Insight based on the predictions: It was observed that the percentage of "very severe" cases tends to increase with the age of the patients. Specifically, individuals aged 80 or above are at a significantly higher risk of experiencing very severe falls.



e. Action before fall insight based on the predictions: There is a clear correlation between age and the utilization of assisted mobility aids or assistance, such as walkers or wheelchairs, which in turn leads to an increase in assisted mobility-related falls.



f. Reason of fall/ Risk Factors associated with the fall Insight based on predictions: The data reveals that individuals with pre-existing medical/health conditions are at a higher risk of experiencing "very severe" falls. This emphasizes the importance of managing underlying health issues in elderly patients and not just focusing on fall prevention.



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

Ans. The 3 most impactful parts of our code and their explanations are as follows –

1. Data Sampling using Cluster Segmentation

```
file_path = 'data/openai_embeddings_primary_narratives.parquet.gzip'
# Open the Parquet file using pyarrow
table = pq.read_table(file_path)

# Convert the table to a Pandas DataFrame
embeddings_df = table.to_pandas()
cleaned_narrative_with_embeddings =
pd.merge(cleaned_narrative_primary_data,embeddings_df,on='cpsc_case_number',
how='inner')

embedding_data =
np.array(cleaned_narrative_with_embeddings['embedding'].values.tolist())

n_components = 200

pca = PCA(n_components=n_components)
```

```
pca_result = pca.fit_transform(embedding_data)

explained_variances = pca.explained_variance_ratio_

cumulative_explained_variance = np.cumsum(explained_variances)

kmeans = KMeans(n_clusters=10000)
kmeans.fit(pca_result)
cleaned_narrative_with_embeddings['cluster_label'] = kmeans.labels_
cleaned_narrative_with_embeddings.to_csv("clustering_results/k_means_cleaned_narrat
ive_10k_clusters_200_dim.csv", index=False)

clusters_data =
pd.read_csv("clustering_results/k_means_cleaned_narrative_10k_clusters_200_dim.csv"
)
embedding_clusters =
pd.merge(cleaned_narrative_with_embeddings[['cpsc_case_number', 'embedding']],
clusters_data[['cluster_label', 'cpsc_case_number']] , on = 'cpsc_case_number')

def get_closest_sample(df_filtered):
    if len(df_filtered)==1:
        return df_filtered['cpsc_case_number'].values[0]
    df_filtered = df_filtered.reset_index(drop=True)

    mean_array = np.mean(df_filtered['embedding'].values, axis=0)

    cosine_similarities =
np.dot(np.array(df_filtered['embedding'].values.tolist()), mean_array)

    # Find the index of the vector with the highest cosine similarity
    closest_index = np.argmax(cosine_similarities)

    return df_filtered['cpsc_case_number'].values[closest_index]

cpsc_sample_list = []
cluster_sample_list = []
for k,v in embedding_clusters.groupby('cluster_label'):
    cpsc = get_closest_sample(v)
    cpsc_sample_list.append(cpsc)
    cluster_sample_list.append(k)

df_sample = pd.DataFrame({'cpsc_case_number':cpsc_sample_list,
'cluster_label':cluster_sample_list})

sample_data_10k = pd.merge(df_sample, cleaned_narrative_with_embeddings,
on='cpsc_case_number', how='left')

sample_data_10k.to_csv('final_sample_data/sample_data_10k.csv', index=Fa
```

The initial part of the code is dimensionality reduction, a critical step in handling the (1536,) OpenAI embedding dimensions. To efficiently cluster these embeddings, Principal Component Analysis (PCA) is employed. The decision to retain 200 principle components is strategic, as it allows us to capture approximately 85% of the total variance. For >=95% explained variance, we had to keep the

principle component count greater than or equal to 300 which would have been a computational overhead.

Subsequently, the KMeans clustering algorithm is applied, generating 10,000 clusters—roughly 10% of the original data size. The main idea behind generating clusters was to **maximize the diversity between the falls incident data** such that the sample data is representative of the actual data. For each cluster, the mean embedding array is calculated, serving as a representative centroid.

A crucial aspect of the code involves identifying the embedding within each cluster that is the closest to the representative centroid We have used cosine distance as the similarity metric. To increase the diversity, the data point closest to the representative centroid of each cluster was chosen and added to the sample dataset hence making the sample data as diverse as possible.
In essence, this approach distills the complexity of the original embeddings into a manageable set of representative clusters, facilitating meaningful analysis and insight extraction in subsequent stages of model development.

It was observed that the representative data followed similar distribution as that of the actual Fall Incidents data. This is supported by the data analysis performed on the sample data as well as the actual data.

2. Creating Gold Standard Data for Severity of Fall using Llama 2 LLM

```
# GPU llama-cpp-python
!CMAKE_ARGS="-DLLAMA_CUBLAS=on" FORCE_CMAKE=1 pip install llama-cpp-python==0.1.78
--force-reinstall --upgrade --no-cache-dir --verbose
# For download the models
!pip install huggingface_hub


model_name_or_path = "TheBloke/Llama-2-7B-chat-GGML"
model_basename = "llama-2-7b-chat.ggmlv3.q4_0.bin" # the model is in bin format

model_path = hf_hub_download(repo_id=model_name_or_path, filename=model_basename)

# Loading on GPU
lcpp_llm = None
lcpp_llm = Llama(
    model_path=model_path,
  n_threads=2, # CPU cores
    n_batch=516, # Should be between 1 and n_ctx, consider the amount of VRAM in
your GPU.
    n_gpu_layers=10000, # Change this value based on your model and your GPU VRAM
pool.
    n_ctx=2048
    )

def create_prompt(prompt,incident):
```

```
    prompt_template=f'''SYSTEM: Classify the sverity of falls based on narratives.
User: Please classify the severity of the reported injury for an elderly patient in
the incident report into one of the following categories:
- "Very Severe": Life-threatening with long-term effects
- "Severe": Not life-threatening but requiring multiple days of hospitalization
- "Moderate": Requires a doctor's visit but no hospitalization
- "Minor": No hospitalization or doctor's visit needed
Severity Classification: [Very Severe/Severe/Moderate/Minor]

Fall Narrative: {incident}
Assistant:'''

    return prompt_template


results=[]
for i in range(data.shape[0]):
    try:
        prompt=create_prompt('',data.narrative.values[i]+'.')

        response=lcpp_llm(prompt=prompt, max_tokens=800, temperature=0, top_p=0.9,
                      repeat_penalty=1.2, top_k=1,
                      echo=False)
        result=response["choices"][0]["text"]

        results.append(result)

    except Exception as e:
        print(f"Error {e} occurred at count = {i}")
        results.append(None)



final_data=data.copy()
final_data['severity']=results


final_list=[]
for i in final_data.severity:
    try:
        pattern = r'\b(Very Severe|Severe|Moderate|Minor)\b'
        matches = re.findall(pattern, i)
        final_list.append(matches[0])
    except:
        final_list.append('Not Found')


final_data['severity_class']=final_list

final_data.to_csv('/content/drive/My
Drive/Severity_Results/severity_data.csv',index=False)
```

The initial part of the code entails loading the Llama 2 model onto the GPU and configuring specific hyperparameters for optimal performance. Subsequently, the prompt, a key component in text

generation using the Llama 2 model, is defined. The objective here is to categorize the severity of falls into distinct levels:

"Very Severe": Poses a life-threatening risk with enduring effects
"Severe": Not life-threatening but demands multiple days of hospitalization
"Moderate": Necessitates a doctor's visit but doesn't require hospitalization
"Minor": No need for hospitalization or a doctor's visit
Following the prompt definition, the Llama 2 model is employed to make predictions on the representative dataset. This approach streamlines the categorization of fall severity, providing a valuable foundation for further analysis and decision-making in the broader context of model development.

NOTE: Similarly we have used other LLMs to follow the same process of creating gold standard training data for finding Reason of Fall/ Risk Factors associated with the fall and also Actions performed just before the fall.

3. Building a Classifier – Training DistilBERT on Golden/Representative Data

```
from sklearn.metrics import f1_score
from transformers import AutoModelForSequenceClassification,
TrainingArguments, Trainer, pipeline, DataCollatorWithPadding

def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True,padding=True)

tokenized_data = dataset.map(preprocess_function, batched=True)
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

def compute_f1(pred):
    # Extract predicted labels and true labels from the prediction tuple
    predictions, labels = pred.predictions, pred.label_ids

    # Calculate F1 score
    f1 = f1_score(y_true=labels, y_pred=predictions.argmax(axis=1),
average='macro')

    return {"f1": f1}

#Number of labels
n_labels = len(list(label2id.keys()))
model =
AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased
", num_labels=n_labels, id2label=id2label, label2id=label2id).to("cuda")
training_args = TrainingArguments(
    output_dir="/content/drive/MyDrive/models",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
```

```
    num_train_epochs=50,
    weight_decay=0.01,
    evaluation_strategy="steps",
    eval_steps=2000,
    save_strategy="steps",
    save_steps=4000,
    load_best_model_at_end=True,
    push_to_hub=False,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_data["train"],
    eval_dataset=tokenized_data["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_f1,
)
trainer.train()

classifier = pipeline("sentiment-analysis",
model="/content/drive/MyDrive/models/checkpoint-20000/", device=0) #Setting
device=0 to enable GPU
predict_data=pd.read_csv('/content/drive/MyDrive/data/cleaned_narrative_pri
mary_data.csv')

predicted_outputs = []
batch_size = 1000

# Iterate over the dataframe in batches
for i in range(0, predict_data.shape[0], batch_size):
    print(i)
    batch_texts =
predict_data['cleaned_narrative'].iloc[i:i+batch_size].tolist()
    results=classifier(batch_texts)
    results=[i['label'] for i in results]
    predicted_outputs.extend(results)

predict_data['severity']=predicted_outputs
predict_data.to_csv('/content/drive/MyDrive/Model_Results_Distilbert_reason
_of_fall.csv', index=False)
```

The initial part of the code involves defining a preprocessing function crucial for tokenizing text narratives and implementing padding and truncation. Following this, the DistilBERT model is imported from the transformers package, with the configuration of hyperparameters for model training. The model is trained for nearly 50 epochs, with the validation F1-score reaching upto 0.7.

After that, the trained DistilBERT model is applied to make predictions across the entire dataset. This comprehensive approach, from preprocessing to model training and utilization, underscores the code's role in optimizing text analysis and prediction accuracy.

NOTE: We have 3 different DistilBERT models trained for Reason of Fall, Action just before the fall and severity of the fall classification.

6. **Please provide the machine specs and time you used to run your model.**
    - CPU (model): Intel(R) Xeon(R), 2.30GHz, 2 Core
    - GPU (model or N/A): 16GB Tesla T4 GPU
    - Memory (GB): 78.2 GB
    - OS: Windows and Linux
    - Train duration: ~4.5 hours
    - Inference duration: ~25 hours (inference from LLMs for different attributes, Eg: severity, reason of fall, etc)

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.)?**

Ans. An OpenAI API key would be needed to get inference from GPT3.5-Turbo model (OpenAI). A free version of Google Colab notebook would be sufficient to run the rest of the code files.

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

Ans. No, we did not use any external tools for data preparation or EDA. Everything else has been mentioned in the code submission.

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

Ans. Ensuring the quality of our analysis and insights has been a paramount consideration throughout our implementation. We undertook a multifaceted approach to assess and validate the robustness of our findings –

1. Manual Validation: Human validation played a crucial role in assessing the accuracy and relevance of our results. We conducted rigorous manual validation processes, involving domain experts and stakeholders, to cross-verify the outcomes of our models. We experimented with different prompts while using Large Language Models and chose the one which gave reproducible results ensuring confidence while taking inference from LLMs (LLMs are stochastic in nature and often generate different output on multiple runs). This approach helped us confirm the practical applicability of our insights in real-world scenarios.

2. Exploratory Data Analysis (EDA): Extensive Exploratory Data Analysis (EDA) was performed at various stages of our project. This involved scrutinizing key statistical metrics, distributions, and

visualizations to gain a deeper understanding of the dataset. EDA allowed us to identify anomalies, patterns, and outliers, ensuring the integrity of our analysis.

3. Consistency Checks: Consistency checks were applied to compare insights derived from different models or methodologies. Whether using traditional machine learning algorithms or advanced deep learning approaches, ensuring that conclusions aligned across methodologies provided a layer of validation for our analysis.

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

Ans. We experimented with several things that were not included in the final workflow. Following are some of the noteworthy things we tried –

1. BERTopic with Hyperparameter Tuning: We explored utilizing BERTopic with extensive hyperparameter tuning to uncover underlying patterns and themes within the data. However, despite the efforts, the results were not as anticipated. The model tended to cluster a significant portion of the data as noise, limiting the insights that could be extracted. Also, the computational demands of BERTopic was significant. While BERTopic provides an unsupervised method for topic modeling, interpreting the topics and clusters it generated was also challenging.

2. KMeans Clustering: We considered employing KMeans clustering to categorize the data. But, determining the optimal number of clusters (K) proved challenging, and even with different K values, the clusters obtained were not sufficiently homogeneous, making it less effective for the specific problem at hand. KMeans assumes that clusters are spherical and equally sized, which may not align with the true underlying structure of the data. In scenarios where clusters have non-spherical or uneven shapes, KMeans may struggle to accurately capture the inherent patterns. Also, the algorithm is sensitive to the presence of outliers, and their effect on the clustering outcome would have lead to suboptimal results.

3. Flan-T5-XL Large Language Model for finding Severity of fall, Reason/Risk Factors of fall, and Action Performed just before fall: To automate the extraction of severity of fall, reason of fall, and action performed just before the fall, we experimented with Flan-T5-XL LLM with zero-shot classification. The accuracy obtained (manually validation) from this approach was notably low, indicating that the model struggled to generalize well to the unique characteristics of the dataset.

4. XGBoost Model trained using Gold Standard Data: A different method explored involved training an XGBoost model on a subset of gold-labeled data, with the aim of achieving better predictive performance. However, after manual validation, it became apparent that the accuracy of this model was considerably lower than expected, leading to the decision to exclude it from the final workflow. XGBoost has a multitude of hyperparameters, and finding the optimal combination was complex. It required a systematic approach, involving grid search or randomized search, which was computationally expensive.

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?**

Ans. The team primarily explored around finding the causes and effects of the falls using state-of-the-art techniques. Doing this practice involved using LLMs for finding hidden answers, some data analysis and some bit of clustering techniques as well. However, the focus of the team was on finding the hidden insights/answers in the narratives. If the team were to continue working on this problem, they would focus more on applying various unsupervised algorithms to evaluate similar groups in the data which might lead to better insights. Also, due to lack of time and compute, the gold data was prepared on 10% of the data. Had there been more time, the sampled data coverage might have been more than 10%. Experimenting with other open-source Large Language models (like Llama 2) could also have been a field of exploration.

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

Ans. There are no such simplifications using which the solution could’ve run faster as the team tried their best on developing a pipeline that runs on minimum compute. However, there would’ve been a slight decrease in performance if the quantized version of large language models would’ve been used.