

Data Augmentation with Scikit-LLM

EuroSciPy, Szczecin
28th August, 2024

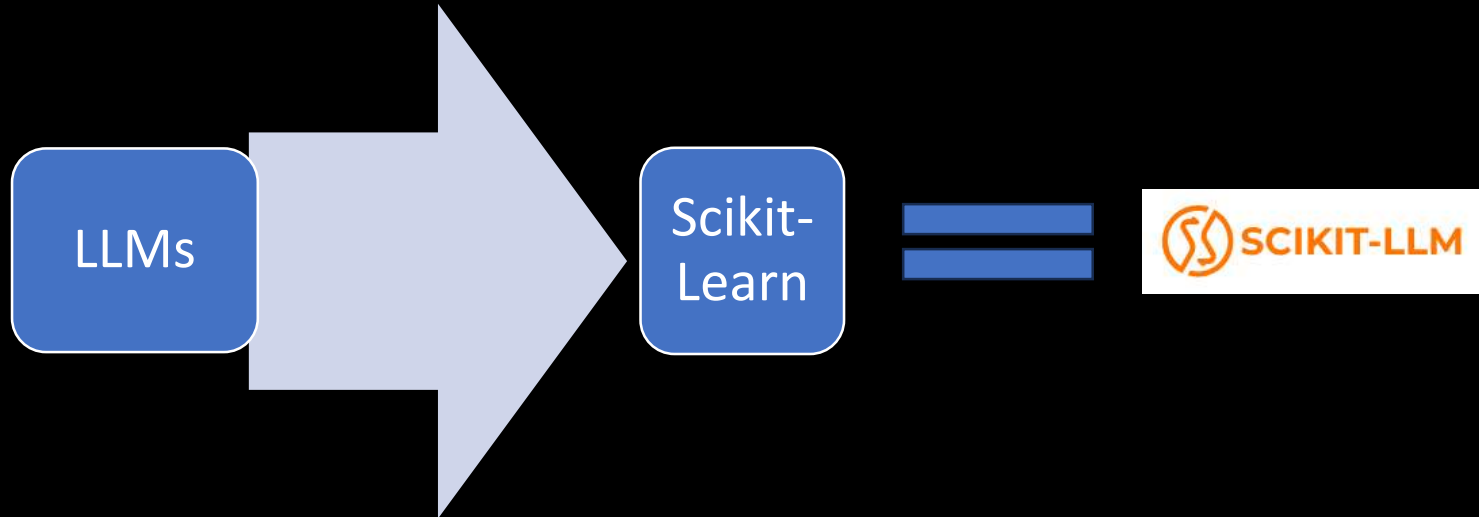
Claudio G. Giancaterino

Summary

- ❑ **Scikit-LLM overview**
- ❑ **Some examples**
- ❑ **Data Augmentation**
- ❑ **Use case**

What is Scikit-LLM ?

Scikit-LLM is an innovative Python library that seamlessly integrates Large Language Models (LLMs) into the Scikit-Learn framework, providing powerful tools for natural language processing (NLP) tasks within the Scikit-Learn pipeline.



Scikit-LLM backends

Scikit-LLM can be considered a game-changer in text analysis, combining the strengths of Large Language Models with machine learning functionalities of scikit-learn.

At high level, Scikit-LLM estimators are split on the language model backend family defined by the API format, and not by the language model architecture.


The two main backend families are GPT and Vertex.

Scikit-LLM key estimators

- **Zero-Shot Text Classification**
- **Few-Shot Text Classification**
- **Multi-Label Text Classification**
- **Dynamic Few-Shot Text Classification**
- **Chain-of-Thought Text Classification**
- **Tunable Text Classification**
- **Text Translation**
- **Text Summarization**
- **Tunable Text-to-Text**
- **Text Vectorization**
- **Named Entity Recognition**

Few-Shot Text Classification

It uses a small number of examples to learn how to classify new text data, leveraging the model's prior knowledge.

```
Python   
  
# Few-Shot Text Classification  
  
# Create an instance of FewShotGPTClassifier with the specified model  
clf = FewShotGPTClassifier(model="gpt-3.5-turbo")  
# Prepare the training data (X) and labels (y) for few-shot learning  
X_train = [  
    "I love the new feature of this app!",  
    "I had a terrible experience yesterday.",  
    "The product is okay, not too bad.",  
]  
y_train = ["positive", "negative", "neutral"]  
# Fit the model using few-shot learning  
clf.fit(X_train, y_train)  
# Define the input text to be classified  
X_test = [  
    "The support team was very helpful.",  
    "I'm not satisfied with the product quality.",  
    "It works as expected."  
]  
# Predict the class labels for the input text  
labels = clf.predict(X_test)  
# Print the predicted labels  
print(labels)
```

Chain-of-Thought Text Classification

It involves breaking down classification process into a series of intermediate steps or reasoning patterns, enabling more transparent and accurate decisions.

```
Python ▾  
  
# Chain-of-Thought Text Classification  
  
# Define the training data and labels for a sentiment analysis task  
X_train = [  
    "I love this new phone, its performance is outstanding and \\  
    the battery life is amazing.",  
    "This software update is terrible, it crashes all the time and is very slow.",  
    "The book was okay, some parts were interesting but others were boring.",  
]  
y_train = ["positive", "negative", "neutral"]  
# Initialize the CoTGPTClassifier with the specified model  
clf = CoTGPTClassifier(model="gpt-3.5-turbo")  
# Fit the classifier with the training data  
clf.fit(X_train, y_train)  
# Define a new test sentence (not seen during training)  
X_test = ["The movie had some good moments."]  
# Predict the class labels for the new test sentence  
predictions = clf.predict(X_test)  
# Extract labels and reasoning from predictions  
labels, reasoning = predictions[:, 0], predictions[:, 1]  
# Print the predicted label and corresponding reasoning for the new test  
for i, (label, reason) in enumerate(zip(labels, reasoning)):  
    print(f"Sentence: '{X_test[i]}'")  
    print(f"Predicted Label: {label}")  
    print(f"Reasoning: {reason}")  
    print("-" * 60)
```

Text Translation

It involves converting written text from one language into an equivalent text in another language, maintaining the original meaning and content.

Python ▾

```
# Text Translation

# Define the training data
X_train=["I love dancing salsa and bachata. It's a fun way to express myself."]
# Initialize the GPTTranslator with the specified model
t = GPTTranslator(model="gpt-3.5-turbo", output_language="Portuguese")
# Translate the sentence
translated_text = t.fit_transform(X_train)
translated_text
```


Data Augmentation

Data augmentation refers to a set of techniques used to artificially increase the size and diversity of a dataset by applying various transformations to the existing data.

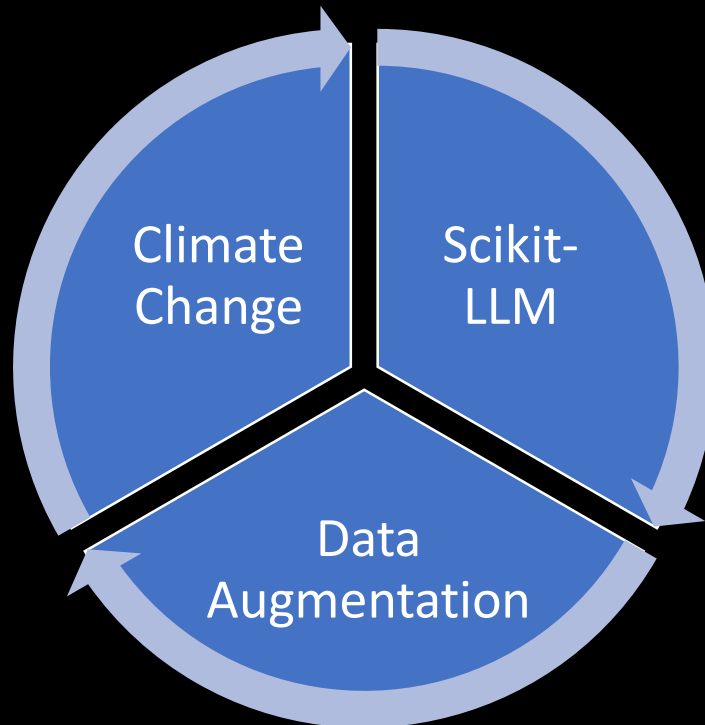
Why do we need Data Augmentation?

- ✓ **Fight Overfitting**
- ✓ **Enhance Model Robustness**
- ✓ **Address Imbalanced Datasets**
- ✓ **Reduce necessity of Large Datasets**

Data Augmentation for Different Data Types

- **Tabular data: synthetic data generation, feature perturbation, interpolation...**
- **Text data: synonym replacement, random insertion and deletion...**
- **Image data: geometric transformation, color jittering, random erasing...**

Use case



Data Collection + Data Cleaning = Work Dataset

[NCDC
Storm
Events
Database
\(noaa.gov\)](https://www.noaa.gov/data/stormevents)



Data collected from the NCDC Storm Events Database, covering 1950 to 2022 in the USA, includes statistics on personal injuries and damage estimates from various weather events such as hurricanes, tornadoes, and floods. From the website has been downloaded dataset for each year and then merged into one big dataset with 51 columns, included 2 text columns, and 1,794,914 rows. The focus was on "flood" events, for which relevant records were selected, and missing values, redundant columns, and unrelated data were removed. Finally, monetary values in the dataset were converted from strings to numbers.

The final dataset consists of 31 columns and 38,398 records.

Flood_Event_Type

```
dftot_flood = dftot[dftot['EVENT_TYPE'].isin(['Flood'])]
```

Data Augmentation with Scikit-LLM: Code

Python ▾

```
# Zero-Shot Text Classification

# select text column to fit zero-shot classification
# (sentiment and risk on episode_narrative)
X_sentiment=df.iloc[:,29]

# fit the model
clf1 = ZeroShotGPTClassifier(openai_model="gpt-3.5-turbo-0125")
clf1.fit(None, ["positive", "negative", "neutral"])
predicted_sentiment_gpt = pd.DataFrame(clf1.predict(X_sentiment),\
columns=['predicted_sentiment_gpt'])

# fit the model
clf2 = ZeroShotGPTClassifier(openai_model="gpt-3.5-turbo-0125")
clf2.fit(None, ["low risk", "medium risk", "high risk"])
predicted_risk_gpt = pd.DataFrame(clf2.predict(X_sentiment),\
columns=['predicted_risk_gpt'])
```

Python ▾

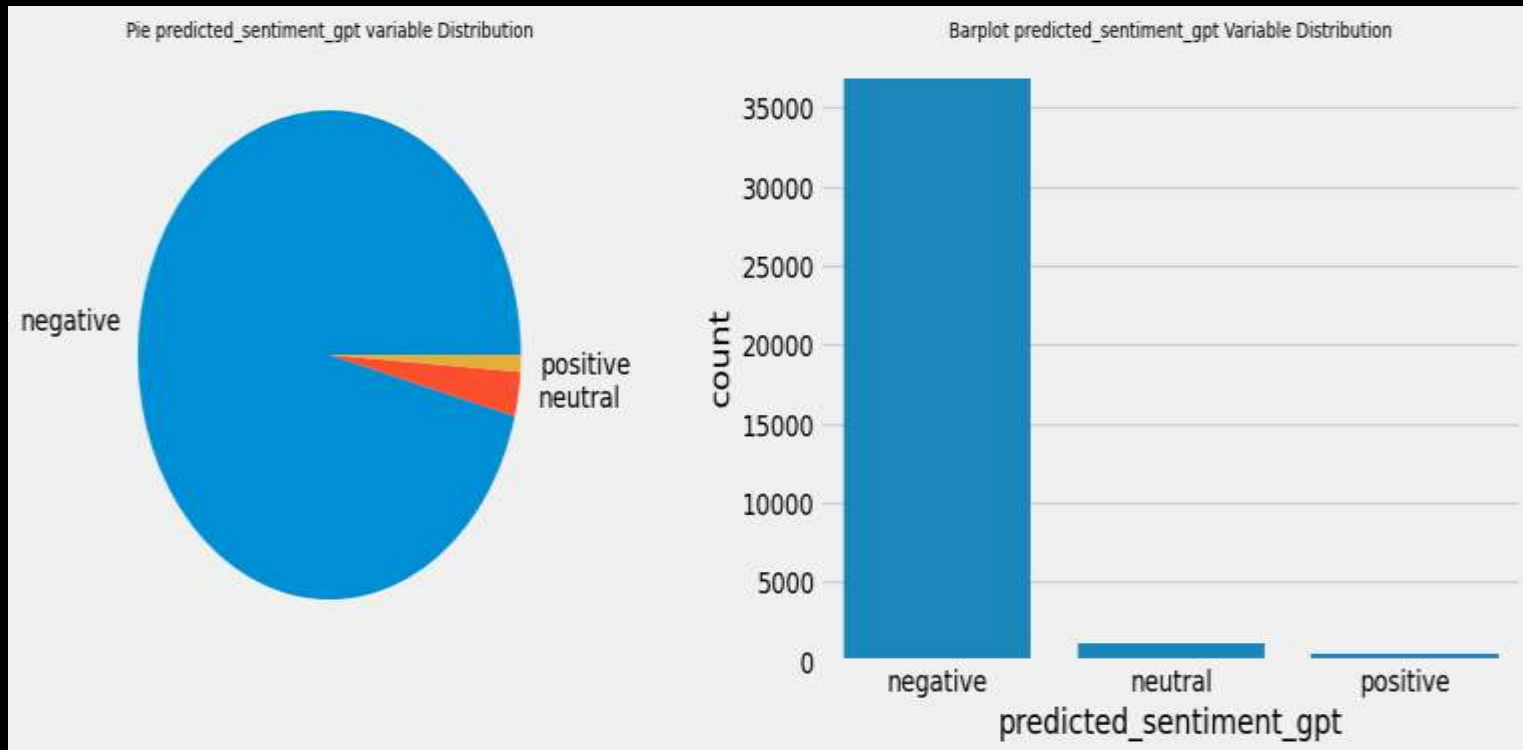
```
# Embedding

# select text columns to build embedding
X_sentiment=df.iloc[:,29]
X_event=df.iloc[:,30]

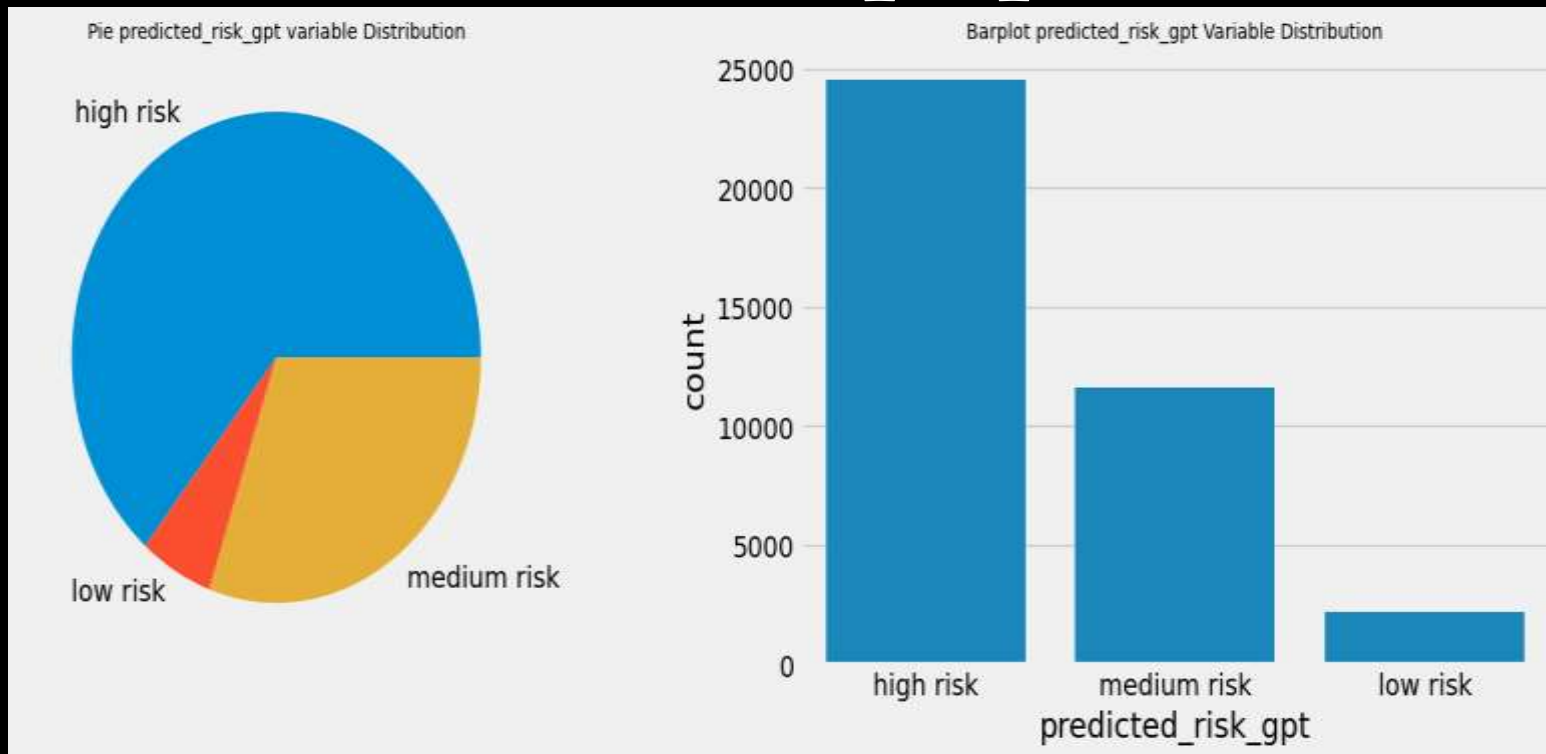
# build embedding on episode_narrative
model_episod = GPTVectorizer()
vectors_episod = model_episod.fit_transform(X_sentiment)[:,:10]

# build embedding on event_narrative
model_event = GPTVectorizer()
vectors_event = model_event.fit_transform(X_event)[:,:10]
```

Data Augmentation with Scikit-LLM: `predicted_sentiment_gpt`



Data Augmentation with Scikit-LLM: **predicted_risk_gpt**

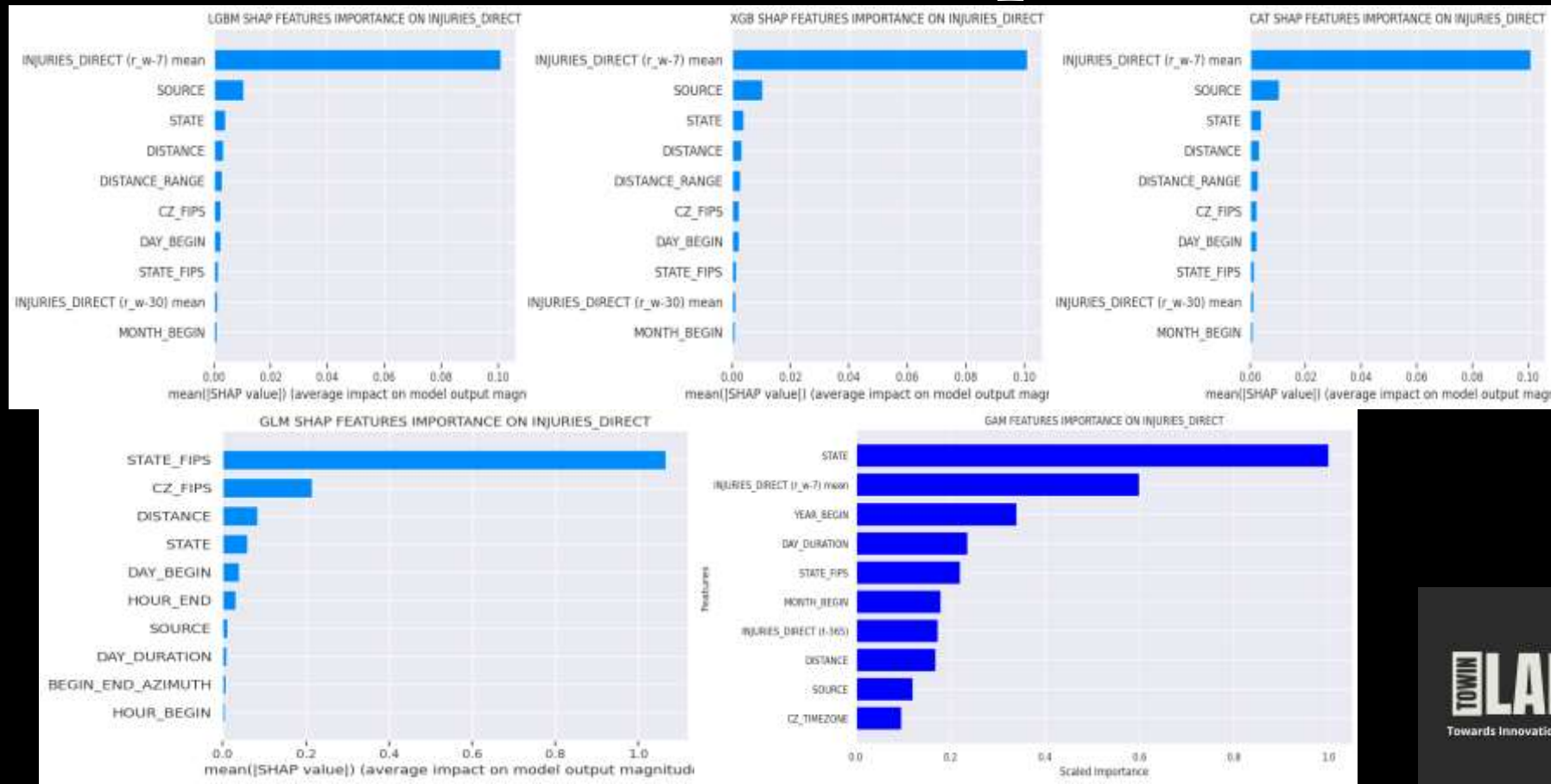


Data Augmentation with Scikit-LLM: embeddings

```
df.iloc[:,32:52].describe().T
```

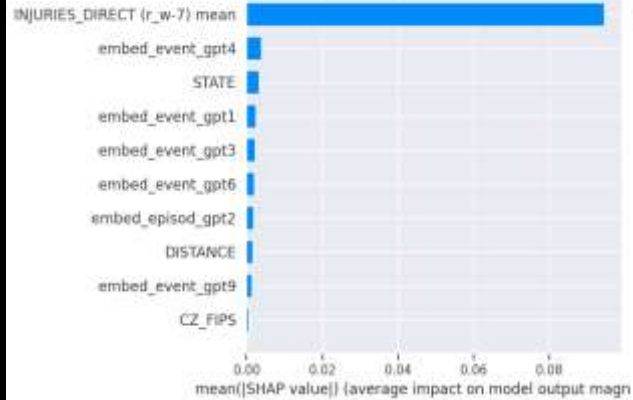
	count	mean	std	min	25%	50%	75%	max
embed_episod_gpt0	38398.0	-0.016808	0.009427	-0.052020	-0.023349	-0.016708	-0.010367	0.021528
embed_episod_gpt1	38398.0	-0.006413	0.010204	-0.045055	-0.013191	-0.005958	0.000404	0.030149
embed_episod_gpt2	38398.0	0.004548	0.009586	-0.034101	-0.001928	0.004899	0.011089	0.038639
embed_episod_gpt3	38398.0	-0.008144	0.010529	-0.047241	-0.015404	-0.008313	-0.001156	0.035343
embed_episod_gpt4	38398.0	-0.005666	0.011905	-0.055183	-0.013564	-0.005558	0.002819	0.037800
embed_episod_gpt5	38398.0	0.020970	0.009892	-0.018789	0.014344	0.021122	0.027610	0.059796
embed_episod_gpt6	38398.0	-0.005247	0.010494	-0.046455	-0.012238	-0.004995	0.001793	0.039692
embed_episod_gpt7	38398.0	-0.013841	0.010543	-0.050245	-0.020813	-0.014273	-0.007203	0.032356
embed_episod_gpt8	38398.0	-0.008262	0.011402	-0.054461	-0.015653	-0.008077	-0.000625	0.032176
embed_episod_gpt9	38398.0	-0.025418	0.009086	-0.062480	-0.031091	-0.025496	-0.019398	0.010344
embed_event_gpt0	38398.0	-0.005881	0.009103	-0.043164	-0.012045	-0.005965	0.000182	0.034685
embed_event_gpt1	38398.0	-0.005816	0.010580	-0.049058	-0.012815	-0.005692	0.001314	0.037376
embed_event_gpt2	38398.0	0.002403	0.010754	-0.047314	-0.004862	0.002595	0.009719	0.044456
embed_event_gpt3	38398.0	0.001375	0.010144	-0.043979	-0.005255	0.001394	0.008207	0.039966
embed_event_gpt4	38398.0	-0.016768	0.011458	-0.058653	-0.024500	-0.016564	-0.009045	0.032566
embed_event_gpt5	38398.0	0.015199	0.011297	-0.028740	0.007385	0.015186	0.022911	0.059868
embed_event_gpt6	38398.0	-0.011737	0.011669	-0.051402	-0.019752	-0.011663	-0.003751	0.036160
embed_event_gpt7	38398.0	-0.007407	0.010022	-0.048194	-0.014242	-0.007453	-0.000847	0.030899
embed_event_gpt8	38398.0	-0.005101	0.013574	-0.052318	-0.014769	-0.004464	0.004788	0.042354
embed_event_gpt9	38398.0	-0.022895	0.009566	-0.058607	-0.029435	-0.023082	-0.016529	0.018322

Feature Importance Injuries_Direct Prediction

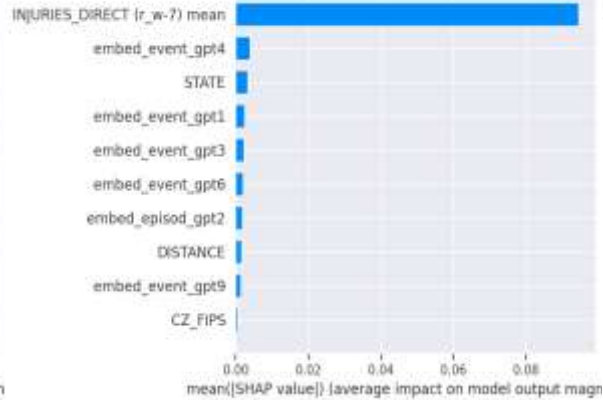


Feature Importance Injuries_Direct Prediction AUG

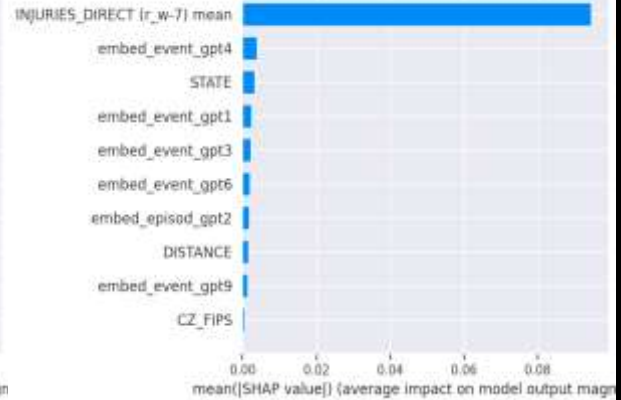
LGBM SHAP FEATURES IMPORTANCE ON INJURIES_DIRECT



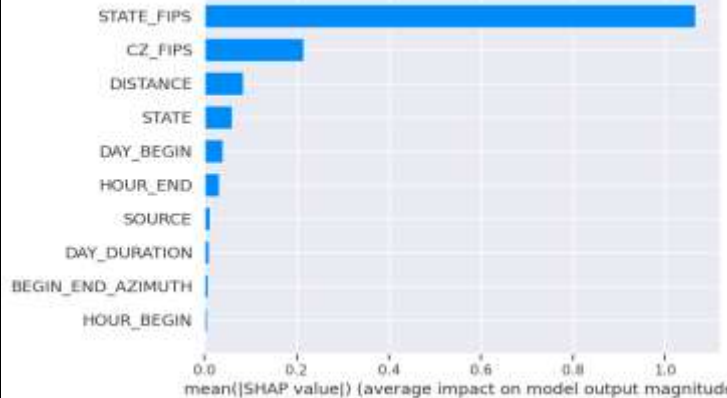
XGB SHAP FEATURES IMPORTANCE ON INJURIES_DIRECT



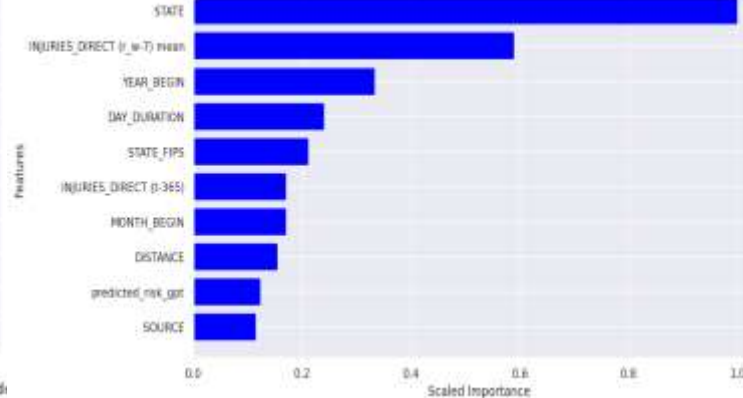
CAT SHAP FEATURES IMPORTANCE ON INJURIES_DIRECT



GLM SHAP FEATURES IMPORTANCE ON INJURIES_DIRECT



GAM FEATURES IMPORTANCE ON INJURIES_DIRECT



Conclusions

- ✓ **Scikit-LLM integrates large language models (LLMs) with the familiar Scikit-learn API, making it easy for data scientists to incorporate advanced language models into their existing workflows without needing to learn a new interface.**
- ✓ **By bringing the power of LLMs to Scikit-learn, Scikit-LLM significantly boosts the natural language processing capabilities of traditional machine learning pipelines, enabling more complex and nuanced text analysis tasks.**
- ✓ **Scikit-LLM offers modular components that can be easily customized and extended, allowing users to tailor the implementation to specific needs.**

References

- Scikit-LLM documentation: [Scikit-LLM](#)**
- Medium article: [Data Augmentation with Scikit-LLM](#)**
- [Injuries Flood Prediction notebooks](#)**
- [Scikit-LLM notebook examples](#)**
- [NCDC Storm Events Database \(noaa.gov\)](#)**

?

?

?

?

?

?

Thank you

**Keep in
touch:**

- [Linkedin](#)
- [Newsletter](#)
- [Medium](#)
- [Website](#)