# Enhancing TabLLM for Tabular Data Handling

## Introduction:

TabLLM introduces an innovative approach for classifying tabular data using large language models (LLMs). Traditional methods typically rely on machine learning models specifically designed for structured data. However, TabLLM leverages the power of LLMs, which are generally used for natural language processing tasks, by transforming tabular data into a natural text format that LLMs can interpret and process.

Eg:

| TransactionID | Amount | Location | Type | Fraud |
|---|---|---|---|---|
| 1 | 100 | New York | Online | Yes |
| 2 | 50 | Los Angeles | In-Store | No |
| 3 | 300 | New York | Online | Yes |
| 4 | 20 | Chicago | In-Store | No |

## Inspiration:

Inspired from the original paper on TabLLM for insights and techniques related to serialization and large language models for tabular data. This resource offered key ideas and methodologies that shaped the approach used in this experiment.
**Link**: https://arxiv.org/abs/2210.10723

***Future Ideas:*** If given more time, I would delve deeper into experimenting with TabLLM's approach. This would include testing various advanced serialization techniques and exploring the integration of multiple LLMs to potentially improve the model's performance and accuracy in classifying tabular data.

## Core Process of TabLLM

### Serialization of Features

**Description**: Convert tabular data into a natural language string format that the language model can interpret. For example, each row in a dataset is transformed into a descriptive sentence.

**Transaction 1:** The transaction with ID 1 involves an amount of 100 dollars. It took place in New York and was an Online transaction. The transaction was flagged as fraudulent.

**Transaction 2:** The transaction with ID 2 involves an amount of 50 dollars. It took place in Los Angeles and was an In-Store transaction. The transaction was not flagged as fraudulent.

**Transaction 3:** The transaction with ID 3 involves an amount of 300 dollars. It took place in New York and was an Online transaction. The transaction was flagged as fraudulent.

**Transaction 4:** The transaction with ID 4 involves an amount of 20 dollars. It took place in Chicago and was an In-Store transaction. The transaction was not flagged as fraudulent.

**Use in LLMs:** The serialized data is then fed into the LLM, which can be used for various tasks such as classification, prediction, or even generating insights from the data.

## Why This Approach is Interesting:

**Flexibility:** TabLLM is agnostic to the specific LLM used, making it versatile and adaptable to various scenarios.

**Innovative Use of LLMs:** This approach creatively repurposes language models, which are traditionally used for unstructured data, to handle structured tabular data.

**Improved Performance:** By converting tabular data into a form closer to what LLMs are trained on (natural language), the model's performance, especially in zero-shot and few-shot contexts, can be significantly enhanced.

## Why TabLLM over traditional approaches?

TabLLM, which involves using large language models (LLMs) for tabular data classification, can reduce or even avoid some traditional feature engineering and data preprocessing steps. Here's how it addresses these aspects:

1. **Reduction in Feature Engineering**:

   **Natural Language Representation**: TabLLM transforms tabular data into natural language text, which can simplify feature engineering. Instead of manually creating interaction features or applying domain-specific transformations, you provide the model with a textual representation of the data, which allows the LLM to understand and process the features in context.

   **Serialization Flexibility**: By using various serialization methods (e.g., list templates, text templates), TabLLM can capture feature relationships and interactions in a way that might not require extensive manual feature engineering.

2. **Minimized Data Preprocessing**:

   **Direct Serialization**: The primary preprocessing step involves converting tabular data into a text format. This is often simpler than traditional data preprocessing methods, which may include normalization, encoding categorical variables, or handling missing values.

   **Adaptability to Raw Data**: Since LLMs are pretrained on diverse text data, they can often handle raw tabular data in textual format without needing detailed preprocessing. This is particularly true if the serialization method effectively conveys the necessary information.

3. **Contextual Understanding**:

   **Implicit Feature Interaction**: LLMs can implicitly learn feature interactions and relationships from the textual representation, potentially reducing the need for explicit feature engineering. This can be advantageous in scenarios where feature interactions are complex and not easily captured through traditional methods.

However, it's important to note that some preprocessing might still be necessary depending on the specific implementation and the nature of the data. For example:

**Data Cleaning**: Ensuring that the data is clean and well-formatted before serialization is still important

**Serialization Design**: Choosing the right serialization method and format can impact model performance, so some level of experimentation and tuning is required.

---

## Steps for Experiment

1. **Design Serialization Methods**:
   - **Select Serialization Techniques**:
     - *List Template*: List column names and feature values.
     - *Text Template:* Textual description of features (e.g., "The age is 30").
     - *Table-to-Text:* Use a pre-trained model fine-tuned for table-to-text generation.
     - *Text T0*: Use T0 model to serialize pairs of column-value tuples.
     - *Text GPT-3*: Use GPT-3 to serialize all features into natural text.

2. **Implement Serialization**:
   - **Serialization Function**: Develop a function to convert tabular rows into textual format based on chosen serialization method.

- **Prompt Design**: Create task-specific prompts that will be combined with serialized data.

3. **Choose and Fine-Tune LLM**:
   - **Select LLM**: Choose an LLM suitable for the task (e.g., GPT-4).
   - **Fine-Tuning**:
     - *Few-Shot Learning:* Fine-tune the model with a subset of the data.
     - ***Zero-Shot Testing:*** Evaluate performance with zero additional training
4. **Generate Embeddings**:
   - **Textual Embeddings**: Use the fine-tuned LLM to generate embeddings from the serialized text representations of the data.
   - **Embedding Extraction**: Extract embeddings that will be used for downstream tasks.
5. **Downstream Classification Task:**
   - **Feature Extraction**: Use the generated embeddings as features for the classification task.
   - **Train Classifier**: Apply a classification algorithm (e.g., logistic regression, CatBoost) on the embeddings to build a classification model.
   - **Evaluation**: Assess the performance of the classifier using metrics like accuracy, precision, recall, F1-score.
6. **Analysis and Comparison**:
   - **Performance Comparison**: Compare the performance of the TabLLM-based approach with traditional models like CatBoost.
   - **Error Analysis**: Perform error analysis to understand where the model may be underperforming or making errors.