Two Approaches to Forecasting Fraud Applications

Yupei Shu, Miao Wang, Lingfei Zhang

Under Supervision of Prof. Ali Hirsa and Moffy Jiang from CraiditX

Introduction

Before the moment when Artificial Intelligence was applied to fraud detection, loan businesses usually hired employers to manually check customers' static information–citizenship, age, employment, and income. With the high development of the Internet, more and more online lending took place. The involvement of AI in this topic would help to collect dynamic information like web page sequences and make predictions in a more efficient way.

In our project, we will outline the use of sequential time series data and basic information data to determine unacceptable loan applications. An unacceptable application is classified in two ways: a fraudulent application, or an application where the applicant is not creditworthy. Sequential data is classified as data collected from users completing loan applications where their time spent on the application was tracked. The basic data tracked if the client was new, how many days overdue the application was, and the submission time of the application.

The methods used to determine these two outcomes were derived using various **graph embedding** and **sequential embedding** techniques. Here, we will provide a comprehensive analysis and comparison of how different graph embedding models can find potential group and individual risks, and automatically extract features from loan application data. Also, we will develop machine learning and deep learning models to extract features from sequential data and compared their effectiveness for fraud detection.

Future Works

Graph embedding part:

- For **GraphSAGE** and **FI-GRL** models, our current results are based on a subset of the data, excluding the 'call_info' component. However, we encountered a challenge with StellaGraph algorithm, which requires unique node IDs. In our case, each node is connected to a different set of nodes based on the timestamp, and duplicate IDs hold significance. Consequently, GraphSAGE and FI-GRL lack the capability to incorporate temporal relationships. To address this issue, we can consider modifying the structure of GraphSAGE and FI-GRL to accommodate temporal links, or update StellaGraph to allow for self-loops.
- Regarding **MIDAS** and **MIDAS-R**, it is essential to investigate why our current structure is unsuitable for our task.

Sequential embedding part:

- **Deep learning models**: These models can impose more intricate constraints on time-based features while applying techniques such as fake label generation. Additionally, attention-based models can be explored further.
- **Transformer**: We can attempt to utilize the preprocessed data from Word2Vec in a plain vanilla LSTM model. Further emphasis can be placed on mapping non-sequential features of the data to sequential features.
- **BERT**: Although we used PCA for dimensionality reduction, we encouraged to find the optimal number of categories for the continuous features. Despite the unconvincing results of the NLP models, we still saw potential in approaching this problem as a NLP problem. Trying a Stochastic Context Free Grammar (SCFG) model to see what other NLP methods could improve the accuracy and results. Lastly, creating more sequences that take into account interactions between different features to feed the SGT model.

Evaluation

- **AUC score of ROC curve.** AUC calculates the area under the ROC curve which plots the true positive rates and false positive rates at a range of thresholds from 0 to 1. The thresholds represent different prediction cutoffs for classification of the response variable to a 0 (non-fraud) or a 1 (fraud). The closer the AUC is to 1, the better the model is.
- **KS score.** KS score gives an assessment of a distance of two samples. Here our KS score = TPR - FPR. Therefore, the closer the KS score to 1, the better the model is.
- **New Evaluation Method.** It is first adopted by CraiditX. We rank the predicted logits (value of the logit function) which are the output of the second layer in descending order and look at the number of fraud labels detected among the top 1, 3, 5, and 10th percentile. An ideal model should make all fraud data have top 10 percent logits, since the percentage of all fraud data is less than 10 percent. From a business perspective, this method represents a trade-off between rejecting the top x percentile and reducing default probabilities of the accepted sample by a certain percent.
- The Precision-Recall AUC score(**PR score**) is a value between 0 and 1, and a score of 1 represents a model with perfect skill. We could obviously note that PR curves does not consider true negatives, which makes sense because in fraud detection we are usually concerned more about if our model is capable of identifying a rare fraud once it arises.

Graph Embedding

Data

- The provided raw dataset comes from CraiditX's internal client data and consists of 35,373 loan applications. Each line of the raw data text file contains the whole or partial data for an individual application in the format shown in the figure, and all the data are already encoded.
- For a given application, its related device may be shared or have contacts or calls connections with another application, thus creating the basis for network analysis.
- Our dataset is highly imbalanced, where only 3% of the observations were labeled as fraudulent vs. 97% were labeled as non-fraudulent.

1. [apply_info] — Base properties for an individual application
2. [device_info] — Base properties for a device related to this application
3. [contacts_info] — Contact book details on a device related to this application
4. [calls_info] — Call logs details on a device related to this application

Methods

Graph Convolutional Networks (GCN)

- The work is based on a model by Kipf and Welling. Such problems as classifying nodes in a graph can be framed as graph-based semi-supervised learning.
- We need two input matrices built from the raw data: the **feature matrix** and the **adjacent matrix**.
- As for the dataset we used for this model, we only keep the records that are not rejected, which are 14149 in total. Since if a record is rejected, there is no chance for it to be overdue.
- We generated two models, and we simply call them model 1 and model 2. Model 2 is an improved GCN model comparing with model 1. Here are the differences between the two:

feature name	model 1	model 2
num of applications	✓	✓
is new client	✓	✓
num of devices	✓	✓
num of phones	✓	✓
last 7d avg call in duration	✓	✓
last 7d avg call out duration	✓	✓
last 7d call counts	✗	✓
last 14d avg call in duration	✗	✓
last 14d avg call out duration	✗	✓
last 14d call counts	✗	✓
last 1m avg call in duration	✗	✓
last 1m avg call out duration	✗	✓
last 1m call counts	✗	✓
node degree	✗	✓
transitivity	✗	✓
cluster coef	✗	✓

The difference in the feature matrix.

connection type	model 1	model 2
common device id	increase by 1	increase by 0.3
common user id	increase by 1	increase by 0.1 for each common user id
common idfa id	increase by 1	increase by 0.1 for each
common idfv id	increase by 1	increase by 0.1 for each
common mac id	increase by 1	increase by 0.3 for each
common imsi id	increase by 1	increase by 0.1 for each
common phone id	increase by 1	increase to 1, then by 0.3 for each
common contact info	increase by 1	increase to 1, then by 0.1 for each
common call info	increase by 1	increase to 1, then by 0.3 for each

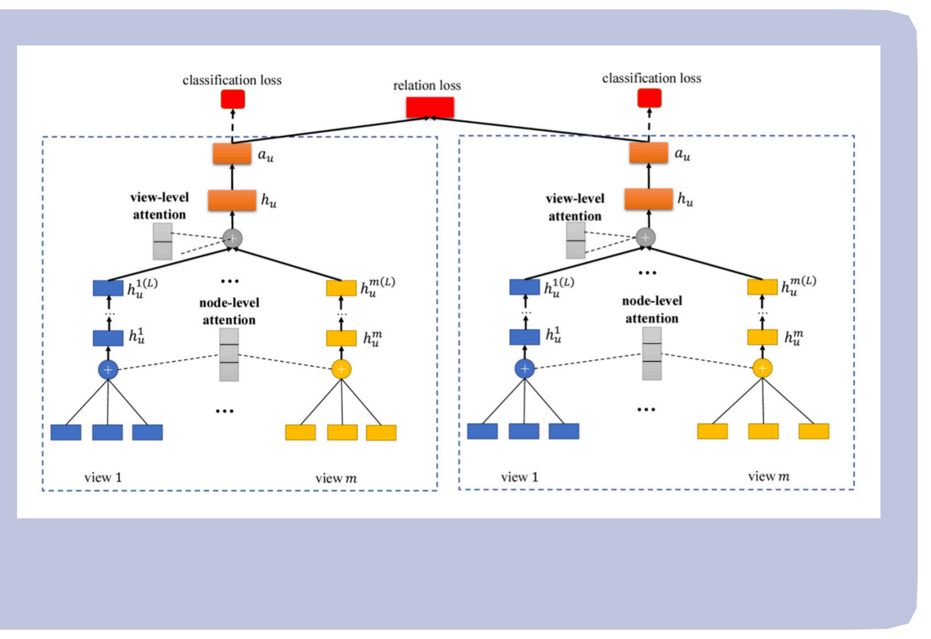
The difference in the adjacent matrix.

GCN with Extra Features

- One potential way to improve it is to try analyzing subgraphs. The main idea is to further analyze the characteristics inside a small community by applying the **Personalized Pagerank Algorithm** and **Dense Graph Community Detection Algorithm**, and import our results to GCN model as new features.

Semi-supervised Graph Attentive Network (SemiGNN)

- SemiGNN by Wang et al. learns weightings of each dataset dynamically instead of manually setting weights for different datasets in the GCN model.
- SemiGNN model is performant on imbalanced datasets since we found our dataset is highly imbalanced.
- SemiGNN aggregates across different datasets ("views"). Variables in each dataset are translated into adjacency matrices that form a multi-view graph.
- The GCN model and the SemiGNN model share the same feature matrix, but the their adjacent matrices are different. A very straightforward difference is that the adjacency matrix for GCN model is a single matrix, and for the SemiGNN model, we generate multiple views according to different variables.
- We need to apply the **sampling methods** to generate sample view matrices for evaluation purpose, because the original adjacent matrices(views) are too large and the computation is impossible. Here we used 3-hop Random Walk sampling method.



Four New Sampling Techniques

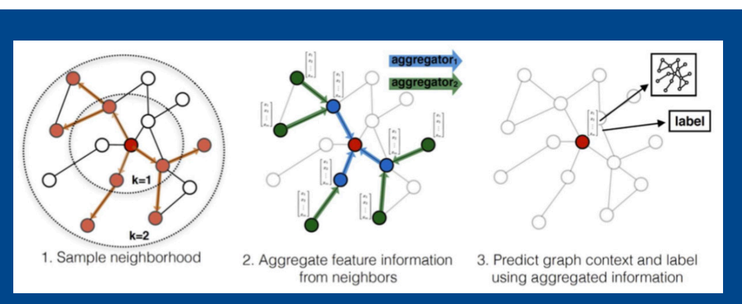
- Random Walk with Restart, Random Jump Sampling, PageRank Node Sampling, Random Degree Node Sampling.
- We also evaluate these different sampling methods on GCN model.

Temporal Graph Analysis, MIDAS & MIDAS-R

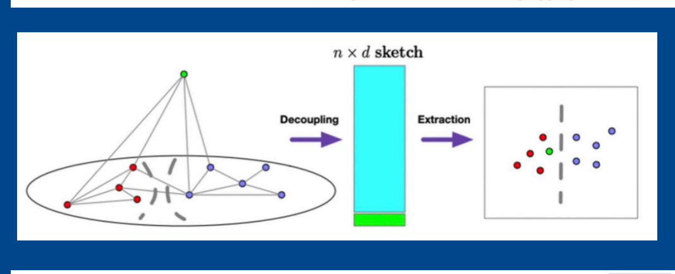
- Microcluster-Based Detector of Anomalies in Edge Streams (MIDAS) is an anomaly detection method that identifies anomalies based on microclusters or sudden groups of suspiciously similar edges using constant time and memory. The method is based on an assumption that fraudulent activities occur in microclusters of suspiciously similar edges.
- MIDAS-R adds time flexibility and slightly different anomalous node scores to its algorithm. MIDAS-R allows the past edges to be incorporated into fraud detection of the most current time by adjusting the weights of the edges.
- MIDAS and MIDAS-R were both based on homogeneous graph.

Inductive Graph Models: GraphSAGE & FI-GRL

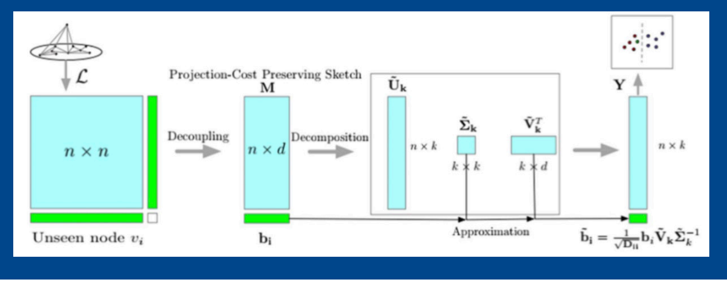
- Both models could be utilized to build a heterogeneous graph network.
- GraphSAGE learns aggregator functions, which would induce the embedding of a new node given its features and neighborhood. While algorithms like MIDAS, it needs to update the previous data structure for the new edge and retrieve updated counts.
- Fast Inductive Graph Representation Learning (FI-GRL), proposed by Jiang et al., is a fast and weightings of flexible framework that can preserve important graph topological information with provable theoretical guarantees and can be naturally generalized to unseen nodes. There are two stages in this framework: decoupling and feature extraction.
- Feature Embedding: to preserve the temporal feature of the graph, we deployed a fixed-length sliding window and pre-determined step size. We finally passed the embeddings into an XGBoost classifier to perform binary classification.


k=1
k=2
aggregator₁
aggregator₂
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

GraphSAGE sample and aggregate approach.



Intuition of FI-GRL.

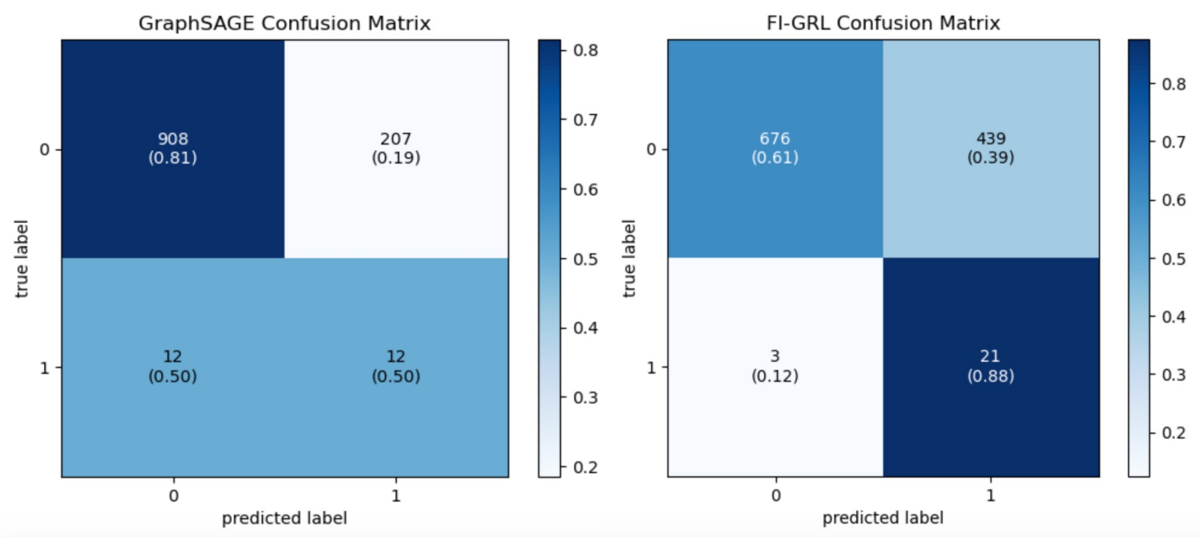


FI-GRL framework.

Results

Model	ROC-AUC	KS	top 1%	top 3%	top 5%	top 10%
GCN(model 1)	0.5	0.0	1.5	4.14	5.64	11.28
Random Forest Baseline Model	0.5081	0.0162	5.77	13.85	19.62	34.62
SemiGNN(3-hop RW)	0.4986	0.0459	1.24	3.23	4.84	10.43
GCN	-	-	1.19	3.97	4.76	8.33
GCN with focal loss	-	-	0.39	1.54	3.86	9.65
GCN with more layers	-	-	0.40	2.78	3.97	7.54
SemiGNN(RJ1)	0.5109	0.0528	1.49	3.60	5.47	9.81
SemiGNN(RJ4)	0.5013	0.0471	1.74	3.98	6.21	10.43
SemiGNN(RWWR1)	0.5307	0.0851	0.75	3.11	4.10	9.32
SemiGNN(RWWR4)	0.4936	0.0278	0.87	2.48	4.10	8.82
SemiGNN(RDN1)	0.4966	0.0385	0.62	2.11	4.47	8.82
SemiGNN(RPN1)	0.4941	0.0373	0.87	2.86	5.47	10.19
MIDAS	0.3333	0.0	-	-	-	-
GraphSAGE	0.6572	0.5507	4.17	4.17	8.33	25
FI-GRL	0.7406	0.5343	0.0	4.17	4.17	8.33

- **GraphSAGE** and **FI-GRL** model outperform the other models according to the AuROC score and KS score.



- GraphSAGE model receives a high True Negative rate, but receives a much less True Positive rate than FI-GRL model.
- The results from GraphSAGE and FI-GRL models show that node features provide critical information to the learning algorithm.
- MIDAS demonstrates obvious lower AuROC and KS scores compared to the baseline model and other models, suggesting that the utilized structure and features may not be a suitable fit for the task.

Sequential Embedding

Data

- The data consists of 230,000 records, 200,000 of which is used for training and 30,000 is used for testing.
- The features are limited. The sequential features are: the name of the current page (categorical), the start time a user visited the page, the end time we visited, process ID, and session ID. Non sequential features consist of number of days the loan was overdue, new applicant (or not), submission date time, and default (categorical).
- The training dataset is a labeled set with each entry denoting if the application was a default or not, while the testing set is unlabeled.
- The data is in json format with two elements for each entry: Order Info and Data. Order Info is a description about the application whereas Data is the information recorded in the application.
- The dataset is highly imbalanced.

Key	Value	Type
overdue	the amount of days the application was overdue that was a default	int
new_client	whether client submitted a successful application	int [1 - yes, 0 - no]
order_time	Time of application	Float (Unix time)
label	Application default	int [1 - yes, 0 - no]

Key, value pairs for the Order Info application data.

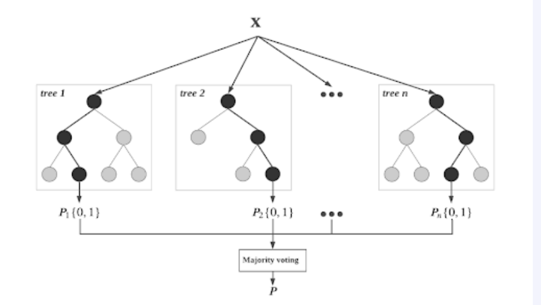
Key	Value	Type
pname	Page name	string
pstime	Page start view time	int (Unix time)
petime	Page end view time	int (Unix time)
pid	Page ID	string
sid	Session ID	string of int ID

Key, value pairs for the Data in the application data.

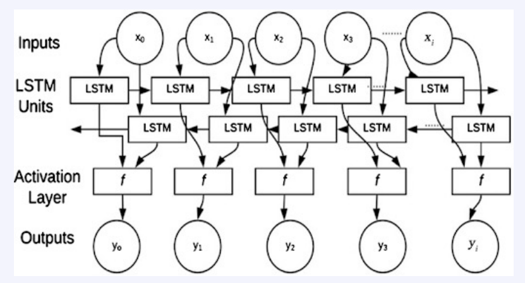
Methods

Baseline Models

- Unsupervised learning classification algorithms.
- **Weighted Random Forest**, is a decision tree model that incorporates tree-level weights to emphasize more accurate trees in prediction and calculation of variable importance.
- **KNN model**, which is a non-parametric supervised learning model where the predicted data point is classified according to the class most common to its k nearest neighbors. KNN can be useful in predicting transitions between different pages in our page sequence data. If the page sequence order is abnormal, then that may be a predictor of fraudulent application activity.
- **Bi-LSTM model** is composed of two independent RNNs, one with information flowing forward and one flowing backwards. The Bi-LSTM model used three bi-directional LSTM layers using the page sequences. Each layer had a different encoding technique; Markov Transition Field (MTF), Convolutional Neural Network (CNN), or plain vanilla LSTM.



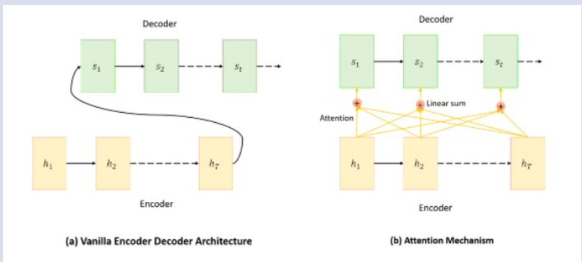
Random forest classifier uses majority voting of the predictions made by randomly created decision trees to make the final predictions.

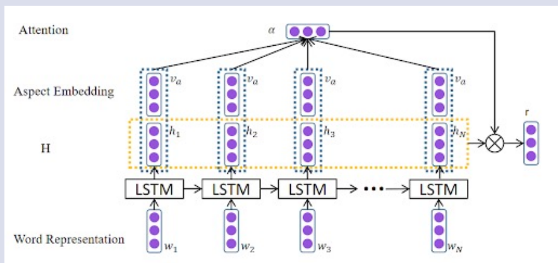


A Bi-Directional LSTM model with information flowing forward and backwards through activation layers.

Deep Learning Methods

- We focused on deep learning methods with LSTM, RNN, CNN, and autoencoders.
- For data preprocessing, **One Hot Encoding** helped turn the categorical data into a binary-like format to be processed by a machine learning algorithm more efficiently. We gravitated towards a Neural Network with LSTM layers, Bi-LSTM layers, and LSTM with attention layers.
- A **CNN model** was created with MTF and Gramian Angular Field (GAF), another type of time series to image transformation, which helped turn the sequence data into image data to feed the model.
- LSTM and CNN with autoencoders, another method of encoding data for more efficient model consumption.
- An important part of using image classification models is to structure the data into an image-like data structure.

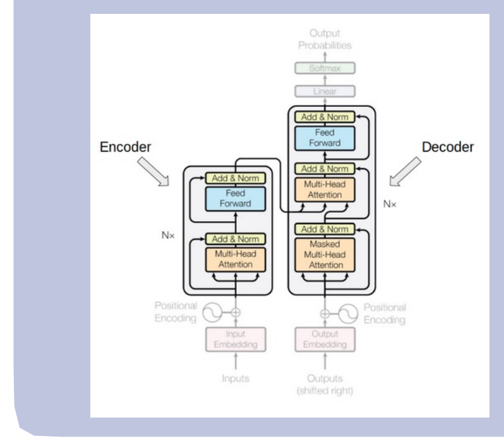




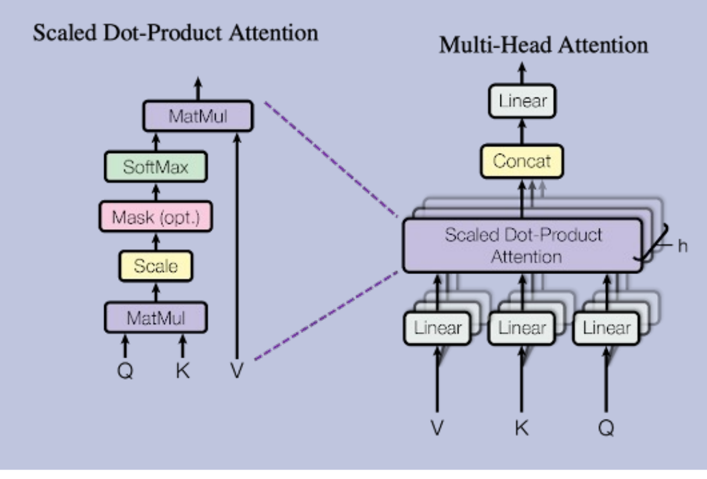
The Architecture of Attention-based LSTM.

Transformer Models

- We focused on how to improve performance from a feature extraction and model optimization perspective. The main focuses were **Word2Vec embedding** and **Transformer** models with **Multi-Head Attention** mechanisms. We also introduced a new dataset of 30,000 entries with higher fraud occurrences. We called this dataset "low income" and the original dataset was called "high income".
- Word2Vec can map words to a vector of real numbers which can then be processed by a deep learning algorithm.
- A Transformer model with multi-head attention was used to link the page type embeddings with page stay time. Both are time-series features crucial for research on the customer behaviors before submitting a loan application.
- The Transformer model works in two parts: an encoder and a decoder. The word2vec embedded data is passed through N encoder layers consisting of a feed forward network and a layer normalization. The data is then decoded with another feed forward network and multi-layer attention.



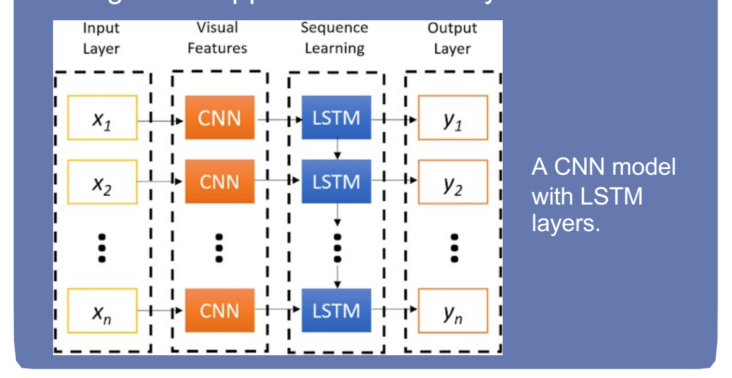
Transformer model with encoder and decoder with multi-head attention.



Multi-Head Attention consists of linear layers split into heads, followed by scaled dot-product attention, concatenation of heads and a final linear layer.

Feature Engineering

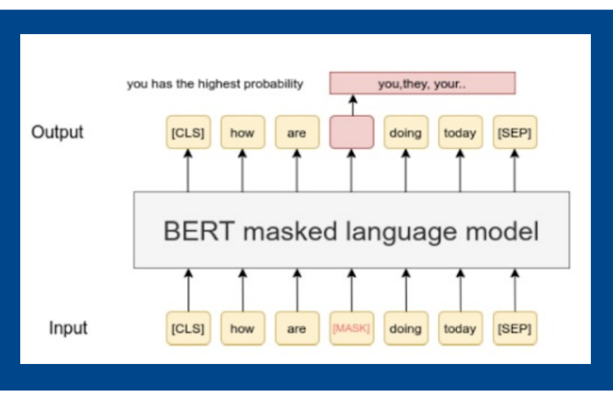
- To the CNN we added Conv2D, a layer where a filter is applied to the elements of a matrix and results in an output of tensors that can be better processed by the CNN, and LSTM layers and achieved better results.
- In order to **determine relationships between variables** and developed it, we also figured out how to optimize the number of categories by splitting continuous data variably with weights as opposed to uniformly.



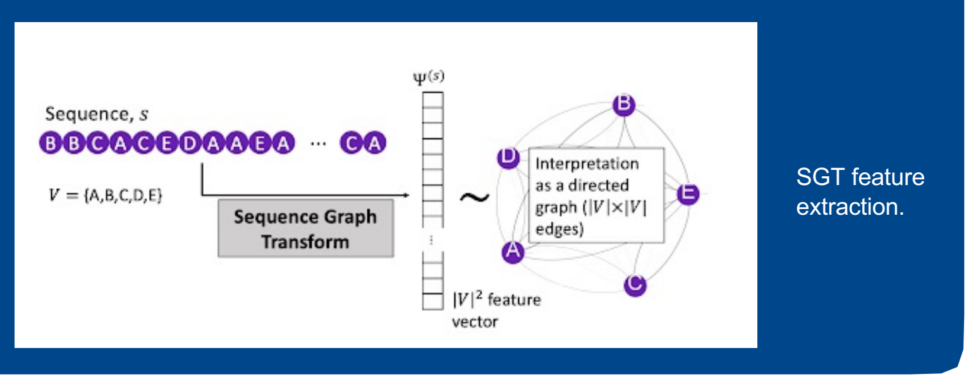
A CNN model with LSTM layers.

NLP

- **BERT** is a language modeling and sentence prediction tool mainly used in NLP, the continued approach of the problem as an NLP one makes sense because the sequential time data is analogous to words and sentences.
- Compared to traditional embedding models, which read texts either from left-to-right or right-to-left, BERT reads the entire sentence at once and is able to understand the full context of the words.
- We tried a new feature extraction technique, **Sequence Graph Transform (SGT)**. SGT can extract a varying amount of short-to long-term dependencies without increasing computations. SGT features can yield significantly superior results in sequence clustering and classification.
- To maximize the efficacy of SGT, we categorized the data into bins and buckets for better model performance. **PCA** was an effective way to explain most of the variance of the model while keeping only the most important features.
- We fine tuned the dictionary created by the SGT and ran the **CatBoost model**, which enables gradient boosting on decision trees.



A BERT masked language model.



SGT feature extraction.

Evaluation

- Accuracy Rate, AUC and KS score are used to evaluate the results of models.
- **AUC** and **KS score** weights more than Accuracy Rate, since we could always get a high value of Accuracy Rate when labeling all the samples in testing set as the majority class.

Results

- The top performing models use **Sequence Graph Transform (SGT)** with a variation of some sort of **Gradient Boosting Model (GBM)**. **LGBM** and **CatBoost** are both variations of GBMs.
- SGT is able to extract long term dependencies from the data and convert them into a sequence of events, usually denoted by an alphabetical mapping of events to letters.
- The benefit of LGBM is that it is more computationally efficient than traditional GBM models because it uses leaf-wise tree evolution instead of level-wise tree evolution.
- CatBoost then separates the SGT sequences into categories and penalizes a misclassified category on an iteration and implements an overfitting detector to ensure that the leaf-wise growth does not conform to the full entry perfectly.
- Overall the use of GBM together with SGT is a great combination for predicting sequences, especially ones like our page view and cross page movement because it converts them into shorter sequences with SGT and then lightly iterates over the best model until the optimal one is converged upon.

Model	KS Score	AUC
Weighted Random Forest	0.1746	0.5764
KNN (Best)	0.1884	0.5842
Bi-LSTM	0.0954	0.5381
LSTM (Plain Vanilla)	0.1027	0.5456
MTF + CNN + LSTM	0.1022	0.5446
[Dense, LSTM1]	0.145	0.585
[Dense, CNN1]	0.03	0.507
[Dense, CNN1, LSTM1]	0.022	0.508
[Dense, LSTM1, CNN3]	0.1487	0.59
[Dense, LSTM2, LSTM3, CNN3]	0.1284	0.59
[Dense, LSTM1, CNN2]	0.09	0.562
[LSTM1, CNN2]	0.001	0.5
CNN with LSTM and Autoencoder	0.014	0.05
Transformer with Attention (Original Dataset)	0.17	0.61
Transformer with Attention (Higher Income Dataset)	0.11	0.56
Transformer with Attention (Lower Income Dataset)	0.16	0.60
BERT	0.2057	0.615
SGT with Handcraft features	0.123	0.585
SGT + LGBM	0.1996	0.6316
SGT(CatBoost)	0.2041	0.634