Capstone Report 2

1. Background

Anomaly detection is broadly used for credit card repayment behaviors. Generally, customers have three kinds of behaviors in terms of repayment:

(1) a user is paying off their credit card balance each month, which is referred to as ‘good’ state
(2) a user is falling behind on payments and only paying the interest on their balance each month, but sometimes making payments or missing payments, which is referred to as ‘struggling’ state
(3) a user is missing payments consistently, but occasionally paying interest, which is called ‘delinquent’ state.

We would like to propose a model to generate data resembling the credit card repayment behavior data in the real world and apply both classical machine learning algorithms, more specifically classification algorithms, and deep learning algorithms on the generated data for anomaly detection. In this case, the anomaly is an alternative word to delinquent state. We regard both the good state and struggling state as normal behaviors.

2. Data Modeling And Generation

Our data model is based on the Hidden Markov Model, which contains two state matrices Z and Y, both N * T in size. N represents the number of customers and T represents the timestamp. In our examples, we define N = 10000 and T = 60.

Z is the matrix of hidden states of customers: good (0), struggling (1) and delinquent (2), while Y is the matrix of observable states of customers: paying off the balance in full (0), paying only the interest (1) and missing payment (2). Z_{it} represents the hidden state (which is the label) of customer i at time point t and Y_{it} represents the observable state of customer i at time point t.

Apart from the two-state matrices Y and Z, we include another two transition matrices to define the probability of transition from Z_{it-1} to Z_{it} and from Z_{it} to Y_{it}. In our model, to maximize the

proximity to reality, we generate two matrices unique to every customer at random (we use uniform distribution) to mimic different behaviors of different customers.
Our transition matrices $P(Z_{it}|Z_{it-1})$ and $P(Y_{it}|Z_{it})$ are defined as follows:

Table 1: Transition Matrix of $P(Z_{it}|Z_{it-1})$

$P(Z_{it}\|Z_{it-1})$	$Z_{it}=0$	$Z_{it}=1$	$Z_{it}=2$
$Z_{it-1}=0$	U(0.79, 0.99)	U(0.05, 0.15)	U(0.01, 0.02)
$Z_{it-1}=1$	U(0.05, 0.15)	U(0.65, 0.75)	U(0.15, 0.25)
$Z_{it-1}=2$	0	0	1

Table 2: Transition Matrix of $P(Y_{it}|Z_{it})$

$P(Y_{it}\|Z_{it})$	Z=0	Z=1	Z=2
Z=0	U(0.95, 0.99)	U(0.01, 0.02)	0
Z=1	U(0.45, 0.55)	U(0.45, 0.55)	0
Z=2	0	U(0.15, 0.25)	U(0.75, 0.85)

3. Creating training data and testing data

We introduce a sliding window method to intercept each data sequence of each customer for creating the training and testing dataset. We suppose the defined window size is w. Y_{it-w}, Y_{it-w+1}, …, Y_{it-2}, Y_{it-1} ($w <= t < T$) are extracted from the sequence as features and Z_{it} is the label. To be more specific, If Z_{it} is equivalent to 2 (delinquent state), we regard it as an anomaly and label it as 1. Otherwise, we label it as 0.

Due to the assumption of our model that once a customer becomes delinquent (hidden state 2), the hidden (actual) state of his or hers will never change. So if we encounter an anomaly (label 2), we will discard the following sequence of the customer.

4. Methods and Experiments

We choose to use some classic machine learning methods, such as logistic regression, naive bayes, SVM, random forest, xgboost and the neural network method like LSTM upon generated dataset. Precision and Recall are both chosen as the metrics of our model. We plot the ROC curve and P-R curve for each model to consider the tradeoff between precision and recall. Window size and future steps are also hyperparameters we will consider for our model, and we only show the results of the best model here for each machine learning method. The window size is the length that we sliced raw data. The future steps denote the steps after the end of the window we need to take into consideration to determine the ground truth of labels. We use 20% of the customers as testing.

4.1 Classic Machine Learning Methods

We applied five traditional machine learning models (Logistic Regression, SVM, Naive Bayes, Random Forest, XGBoost) and used GridSearch to tune their hyperparameters. We tried to create the training and testing data with different values of future step and window size and tune the parameters for each model. The best one for each kind of machine learning model is displayed.

4.1.1 Logistic Regression

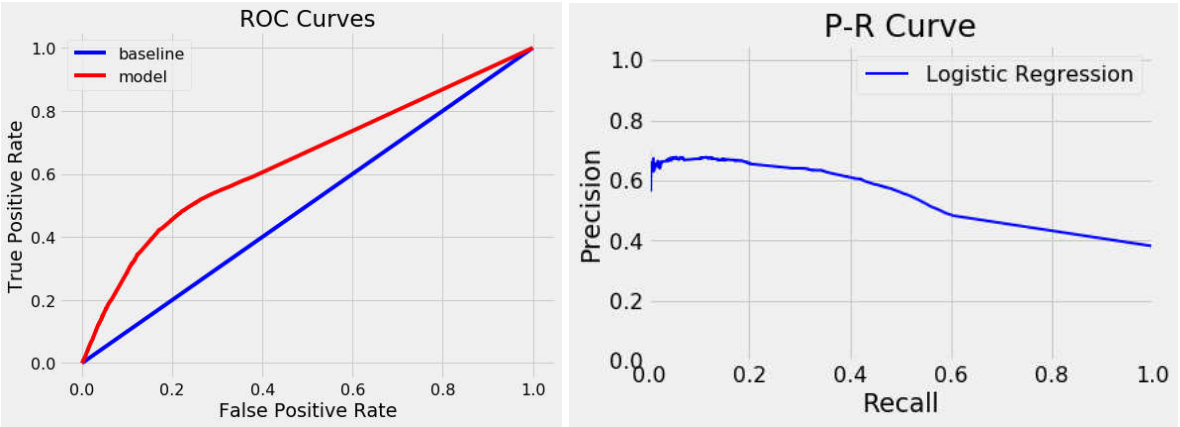

Figure 1: ROC curve and P-R curve of the best Logistic Regression model

The hyperparameters of generating training data and testing data for logistic regression model are: future step variable equivalent to 5 and window size variable equivalent to 10. The parameters for the best logistic regression model are: penalty = 'l2' and solver = 'sag' and other others remain default.
For the best logistic regression model: precision is 0.56, and recall is 0.44.

4.1.2 SVM

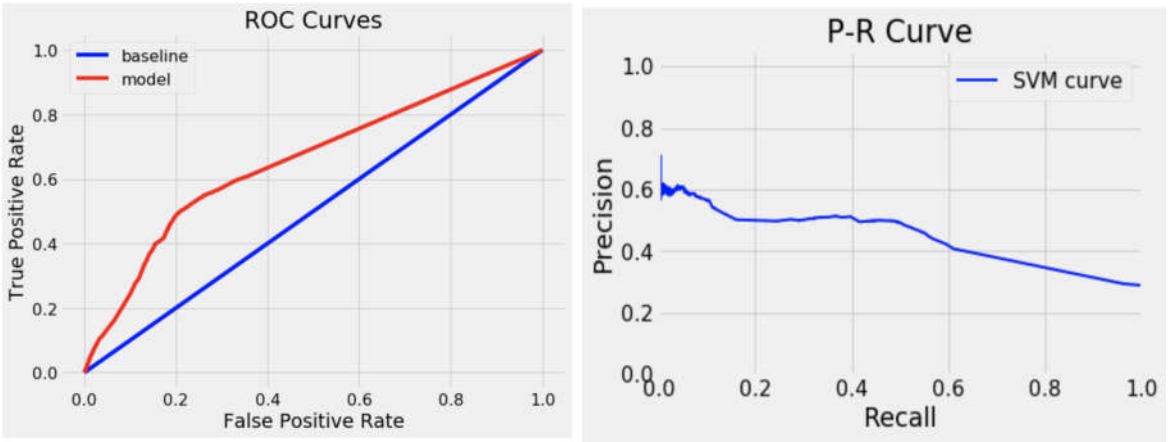

Figure 2: ROC curve and P-R curve of the best SVM model

The hyperparameters of generating training data and testing data for SVM model are: future steps variable equivalent to 5 and window size variable equivalent to 10. The parameters for the best SVM model are: gamma = 'auto', class_weight = 'balanced', kernel = 'rbf', degree = 3, and others remain default.
For the best SVM model: precision is 0.49, and recall is 0.50.

4.1.3 Naive Bayes

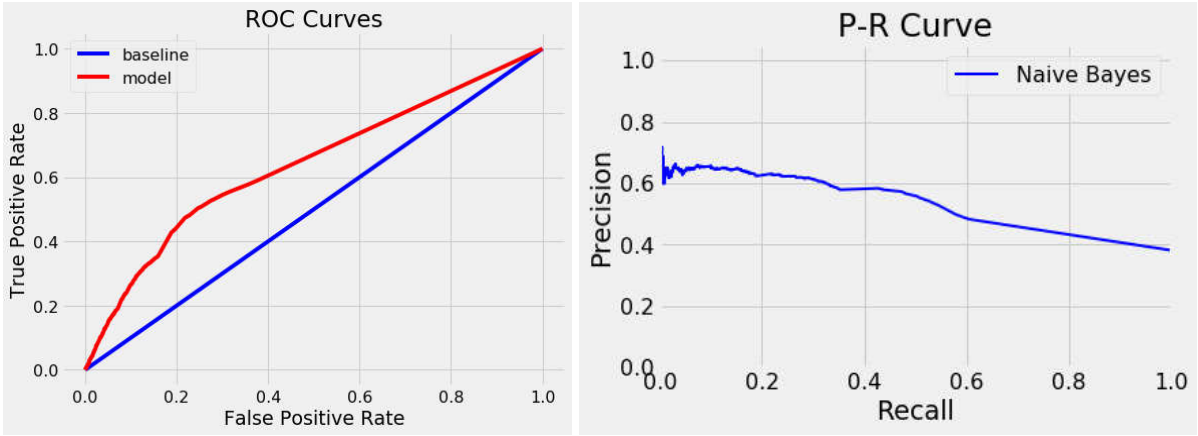

Figure 3: ROC curve and P-R curve of the best naive bayes model

The hyperparameters of generating training data and testing data for Naive Bayes model are window size equivalent to 10 and future steps equivalent to 7. The parameter of the best Naive Bayes model are all in default .
For the best Naive Bayes model, the precision is 0.59, and the recall is 0.45.

4.1.4 Random Forest

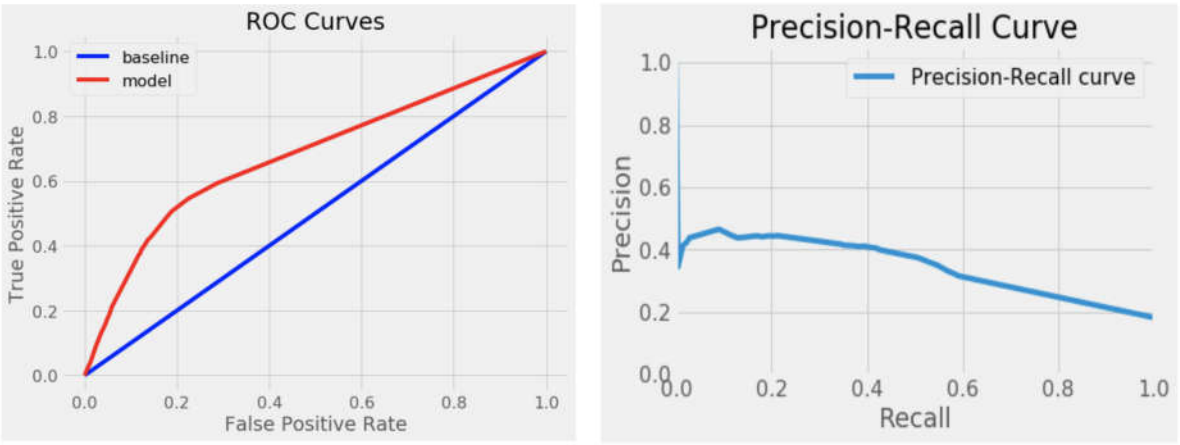

Figure 4: ROC curve and P-R curve of the best random forest model

The hyperparameters of generating training data and testing data for the best random forest model are future step variable equivalent to 5 and window size variable equivalent to 5. The parameters of the best random forest model are: max_depth = 20, min_samples_leaf = 2, n_estimators = 20 and class_weight = 'balanced'. Other parameters not mentioned are default parameters of random forest model. The best result shows that precision is 0.35 and recall is 0.55.

4.1.5 XGBoost

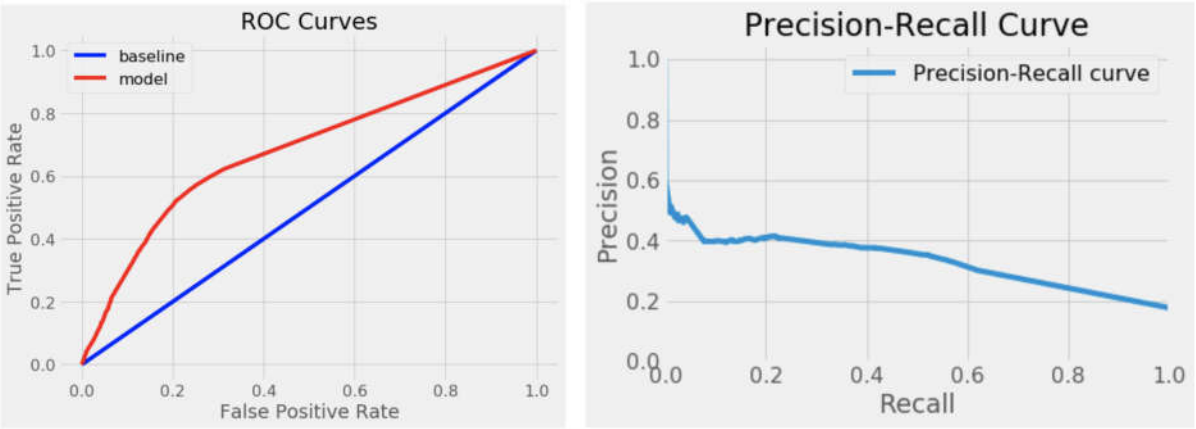

Figure 5: ROC curve and P-R curve of the best XGBoost model

The hyperparameters of generating training data and testing data for the best xgboost model are future step variable equivalent to 5 and window size variable equivalent to 10. The parameters of the best xgboost model are: colsample_bytree = 0.7, learning_rate = 0.05, max_depth = 2, min_child_weight = 1, scale_pos_weight = number of negative samples in the training set / number of positive samples in the training set. Other parameters not mentioned are default parameters of xgboost model. The best result shows that precision is 0.34 and recall is 0.56.

4.2 Deep Learning Methods

We implemented four deep learning models: CNN, LSTM, DNN, and LSTM with the attention layer. For each model, we set window_size = 20 with future step = 4. 20% of all generated customer is held out as test dataset.

4.2.1 ANN

There are 2 layers in the ANN model, only consisted of 2 dense layers and an activated layer. Evaluation results as follows.

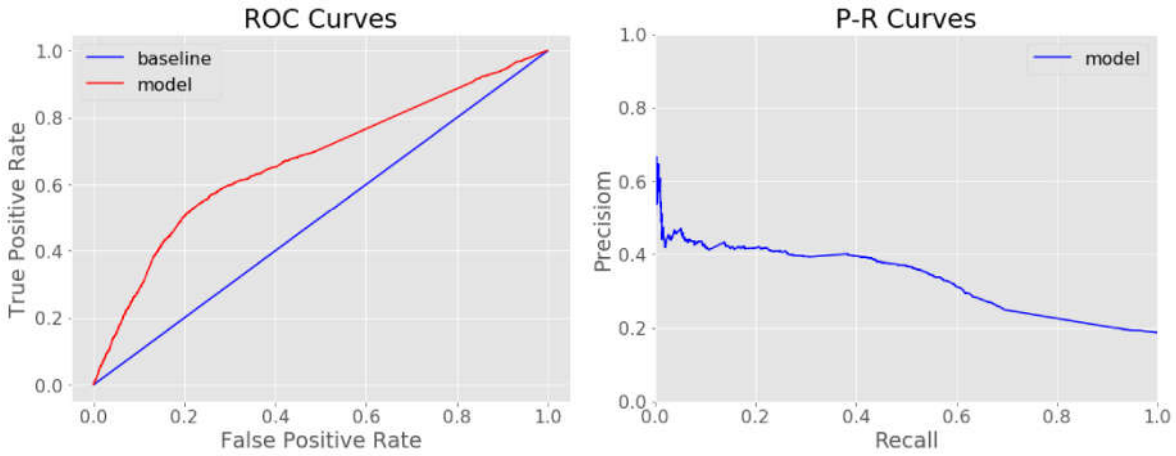

Figure 6: ROC curve and P-R curve of the best ANN model

4.2.2 CNN

CNN-based model is more suitable for image-related task because of the nature of CNN is extracting neighbor feature of an element with the shared weight. We are here to try to explore the possibility of applying a CNN-based model to our problem. By using this kind of model, we are better at extracting neighbor feature stage by stage and appropriately merge all information into the final element which we can use to output our final prediction. We use a very simple CNN-based model consisted of 2 convolution layer followed by ReLU layer and then 2 dense layers. Each convolution layer is a convolution kernel with a size of 3 and output 3 channels. We evaluate our model on the test dataset by drawing the P-R curve and ROC curve as follows.

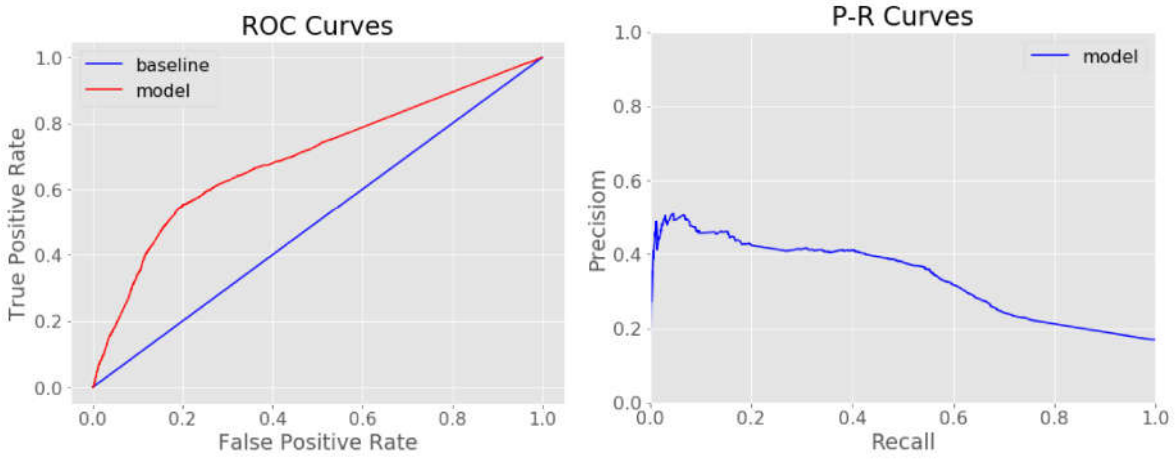

Figure 7: ROC curve and P-R curve of the best CNN model

4.2.3 LSTM

RNN-based model is intuitively a good model to deal with time series problems. We apply a typical RNN-based model LSTM to the data. For RNN-based models, there is an output for each time step and in our context, we only take the final step output as the prediction and calculate loss function and metrics function based on the final step output. Specifically, we apply a single direction LSTM with a hidden neural of size 3, and an output neural of size 1 for each time step and the window size is 20. We apply Adam optimizer as the learner and binary cross-entropy as the loss function. To deal with the highly unbalanced label, we add weigh (5) for positive samples in the dataset. We evaluate our model on the test dataset by drawing the P-R curve and ROC curve as follows.

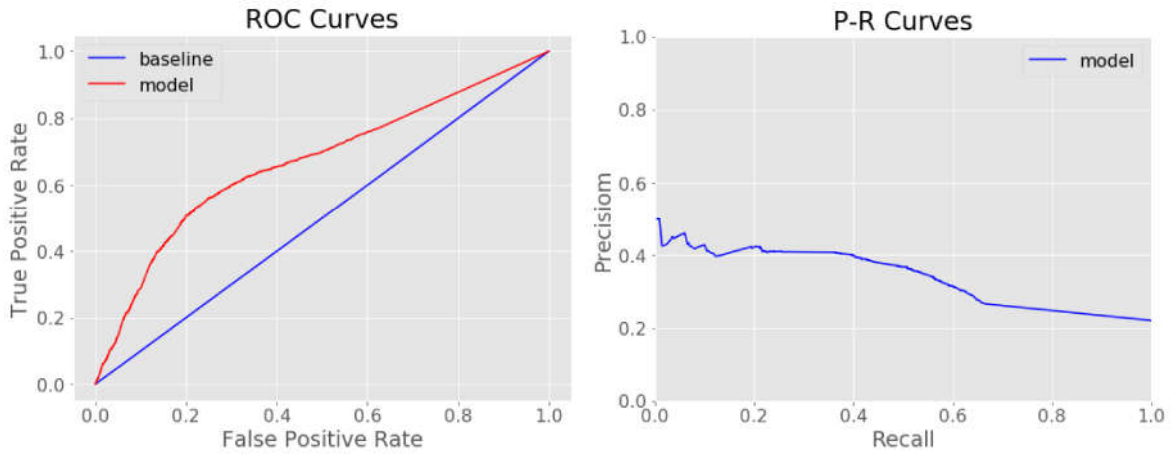

Figure 8: ROC curve and P-R curve of the best LSTM model

4.2.4 LSTM with Attention

By adding attention layer into the LSTM model, we achieve better precision scores within the low precision area and barely the same ROC curve. To focus more on the last steps, we only combine the final 5 steps output into the attention layer. We evaluate our model on the test dataset by drawing the P-R curve and ROC curve as follows.

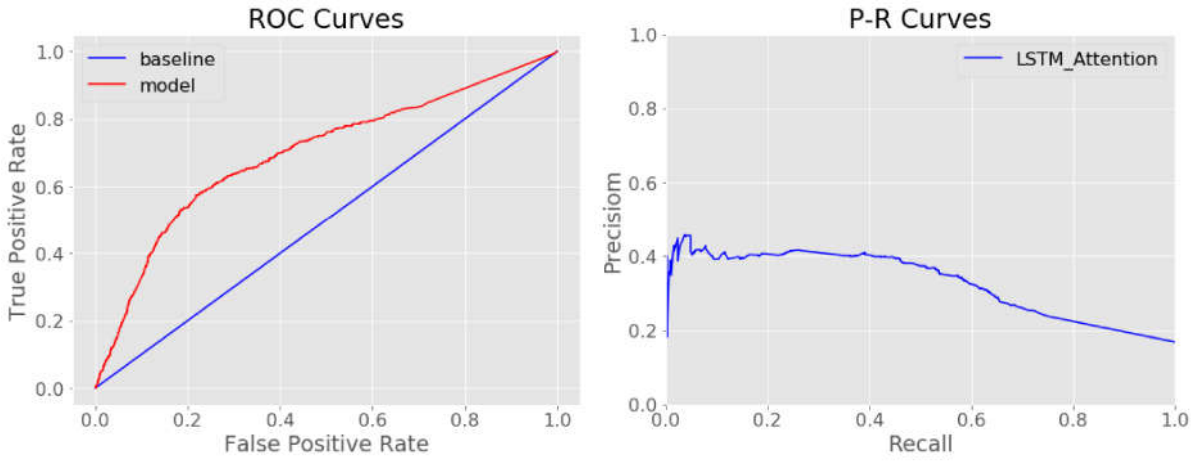

Figure 9: ROC curve and P-R curve of the best LSTM Attention model

4.2.5 Comparison among all NN-based models

To compare all NN-based models, we simply draw the performance of all models together as follows.

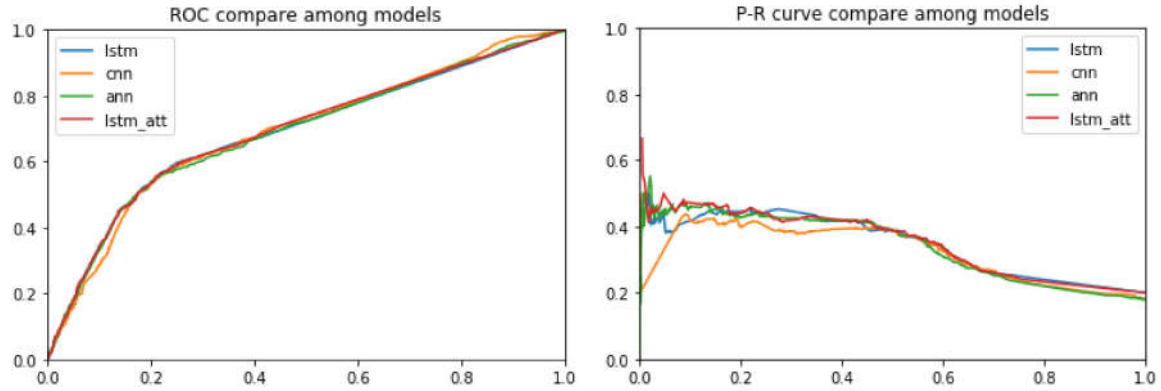

Figure 10: Comparison among all NN-based models

5. Conclusion

For classic machine learning models, we use some non-tree-based models like logistic regression, SVM and naive bayes, as well as some tree-based models like random forest and xgboost. Both models have their own advantages. As indicated by the precision and recall, non-tree-based models perform better in terms of precision while tree-based models are good at recall. Generally, non-tree-based models get precision up to 0.59 and recall around 0.45. The precision of tree-based models is around 0.35 and recall could reach up to 0.55. For neural network models, from the perspective of ROC, the performances of each model are similar. However, from the P-R curve, CNN has the worst performance while LSTM-Attention has the best one.

6. Future Work

For the next step, we need to change our Y_{ij} from binary to a continuous variable. Also, we can add some hidden variables such as consumption habits to make our data generation model more suitable for realistic problems. For example, in November, especially around Thanksgiving Day, people tend to buy more leading to more payment. Thus, they may be more likely to only fulfill the minimum payment instead of the whole credit. For now, the feature only indicating whether a customer paying all the balance, paying minimum payment amount or just paying nothing. However, in real-world cases, the feature value should be more specific paying amount value which is continuous. Converting this categorical value into a continuous amount value making the data more complicated and lower the overfitting risk when using small window size slicing the origin sequence.

Also, we may consider altering the architecture of the neural network and the training schedule such as include dropout layer to prevent overfitting and include focal loss to achieve better classification on hard examples. And finetune the model with a smaller learning rate.

7. Contributions

Data Generalization: Zilin Zhu
Classic Machine Learning Models: Yunbai Zhang and Zichen Pan
Deep Learning Models: Feihong Liu, Han Ding, and Zilin Zhu