1. Tell us how you validate your model, which, and why you chose such evaluation technique(s).

   Used Stratified K-Fold cross-validation of test set AUC to be in line with Kaggle's target benchmark metric. Stratified sampling is necessary as we have a minority class and without it, it is too easy to accidentally create a set w/o the minority class. In addition, PR curve is also plotted to see how well the model performs when used to predict the naturally-imbalanced real-world delinquency distribution. See #2 for further discussion.

   In terms of model validation to choose a best model (e.g. after hyperparameter tuning, model experiments etc.), the average AUC over K-Fold test is used (not always shown in the notebooks, but some runs are blatantly obvious).

2. What is AUC? Why do you think AUC was used as the evaluation metric for such a problem? What are other metrics that you think would also be suitable for this competition?

   AUC generally refers to the area under the ROC curve (TPR vs. FPR). By varying the 0/1 classification threshold we can trace the model's distinguishing ability as a model with discerning power should have its recall increase (or decrease) faster than FPR as we lower the positive class threshold. ROCs must beat the 'random prediction' line where TPR is proportional to FPR so they must be >0.5 or <0.5 (the latter meaning sign flip of the positive class).

   AUC is robust against class imbalances and is a good indicator of discerning power. However in this scenario the natural distribution of cases is highly imbalanced (~6% positive), so we need to plot PR to see how well the model handles precision. Here, all our models quickly suffer from low precision simply because of the disproportionate number of true negative cases - though, comparing against the baseline (at constant 0.06 precision for all recall except 0) it does not perform too badly. We should ideally use PR AUC too in this case as model selection benchmark but it is skipped here simply because of time constraint and that the original requirement is simply ROC AUC. Other possible metrics are e.g. F1 score, but that requires threshold selection and is just a single point of an AUC or PR curve.

3. What insight(s) do you have from your model? What is your preliminary analysis of the given dataset

   Didn't do a deep dive due to time constraint, but referring to 1_logistic_simple.html we see that the 3 past dues/late payment count variables & 1 revolving utilization of unsecured line variable have very high coefficients (>1.0 vs others which are often ~0.01-0.1), and this is backed up by 2_logistic_woe.html when where we also see suspiciously high IV for these variables. It is almost saying, hand-wavingly, that "customer's past behavior of non-payment and high credit utilization is a strong predictor

of delinquency in 2 years". No attempt was made to create a model based solely on these 4 variables to see if they alone can achieve good AUC due to time constraint.

4. Can you get into the top 100 of the private leaderboard, or even higher

   Private AUCs for various models - final ensemble gave the best output of 0.86825 which is within top 100 of the private leaderboard (AFAIK the 100th position score is 0.86723) - as I didn't clamp the random seeds at any step, your mileage may vary but they should be within the ballpark (or, just load the saved models).

Model	***Output File***	***Private AUC***
sm_model_4 @ 1_logistic_simple.ipynb *Logistic with Logged Data*	*pred_out_4.csv*	*0.85517*
sm_model_woe_v2 @ 2_logistic_woe.ipynb *Logistic with WoE*	*pred_out_woe_v2.csv*	*0.85679*
lgbm_tune @ 3_trees.ipynb *LightGBM with crude hyperparam opt*	*pred_out_lgbm_tune.csv*	*0.86784*
xgb_tune @ 3_trees.ipynb *XGBoost with crude hyperparam opt*	*pred_out_xgb_tune.csv*	*0.86651*
final_ens @ 3_trees.ipynb *Final Ensemble with crude weight opt*	*final_ens.csv*	*0.86825*

   The final model is a combination of 0.4 weighting from xgb_tune and 0.6 from lgbm_tune, and we found that 0 weight for the logistic model gave better results. The fact that trees are giving better results hints at nonlinear partitioning, and that ensemble/boosted trees are giving better results also hints at ensembles being better - it is worth exploring with more advanced methods that can handle such nonlinear partition boundaries (e.g. SVM, or just jump straight to simple MLPs)