Xingchen Xiao

Question 1：

Q1_code.py : the code file.
Q1_script.sh : the scripting file training with 10 cores.
Q1_script_20.sh : the scripting file training with 20 cores.

1. Finding best parameters for each model.

Random Forest	**Best model parameter**			
metric	**accuracy metric**		**auc metric**	
Execution Core#	**10-core**	**20-core**	**10-core**	**20-core**
numTrees	10	10	10	10
maxDepth	10	10	10	10
maxBins	15	15	15	15
Score	0.7081	0.7081	0.7821	0.7821

GBT	**Best model parameter**			
metric	**accuracy metric**		**auc metric**	
Execution Core#	**10-core**	**20-core**	**10-core**	**20-core**
maxIter	15	15	15	15
maxDepth	10	10	10	10
maxBins	15	15	15	15
Score	0.7200	0.7200	0.7977	0.7977

2.Training with best parameter

Model	Random Forest		GBT	
Execution Core#	**10-core**	**20-core**	**10-core**	**20-core**
accuracy score	0.7083	0.7083	0.7253	0.7253
AUC score	0.7827	0.7827	0.8051	0.8051
Execution Time: (In seconds)	548s	626s	582s	4376s
The most relevant top 3 Features	m_bb 0.3104 m_wbb 0.1506 m_wwbb 0.1326	m_bb 0.310474 m_wbb 0.150659 m_wwbb 0.132654	m_bb 0.1309 m_jjj 0.0947 m_jlv 0.0908	m_bb 0.130930 m_jjj 0.094743 m_jlv 0.090815

Observation:

The best model parameters choosing by ‘accuracy’ and ‘auc’ metrics are the same. Which means they can be both used for determining the model performance. However, the auc metric does produce higher score which is more precise for the training result. I believe it is because it is a binary classification problem. The performance between 10 cores and 20 cores are the same as well. Training under 20 cores take much more time which is strange. The score from the prediction is around 0.78~0.80 and it is very close to the score in the paper so that our model is very good on this dataset.

Question 2:

Q2_code.py: The code for question 2.2
Q2_code_2.py: The code for question 2.3
Q2_scirpt.sh : The question 2.2 scripting file training with 10 cores.
Q2_script_20.sh : The question 2.2 scripting file training with 20 cores.
Q2_scirpt_2.sh : The question 2.3 scripting file training with 10 cores.
Q2_script_2_20.sh : The question 2.3 scripting file training with 20 cores.

2.1:
For data preprocessing:
I have used one-hot encoding for categorical value in the dataset such as: Cat1, Cat2,Cat3,etc…

Some feature like Household_ID, Blind_Make, Blind_Model are not include the final training set because I don't think they are not directly relevant to our prediction.

In Addition, the data distribution in the dataset is very imbalanced. I tried using sample the non-zero data and make it equal to the numbers of zeros data. However, it does not perform very well.

2.2:

Model	**Linear Regression**	
Execution Core#	**10-core**	**20-core**
RMSE	40.98	36.09
MAE	2.66	2.60
Execution Time: (In seconds)	56.168s	156.93s

2.3:
I Believe logistic regression is a good model for binary classification especially for linear data so I choose logistic regression to classify the zero and non-zero data. And I also using the weightCol attribute to let the model determine which kinds of data has more importance. I have assigned 0.98 weight to the non-zero data and 0.02 weight to the zero data.

Model	**Logistic Regression**	
Execution Core#	**10-core**	**20-core**
AUC score	0.5837	0.599
Execution Time: (In seconds)	280.08s	68.46s

Model	**GLM**	
Execution Core#	**10-core**	**20-core**
RMSE	39.81	45.59
MAE	1.57	1.521
Execution Time: (In seconds)	13.66s	3.898s

Observation:
The 20-core training result does outperform the 10-core training result a little bit in training time. However, the score result is the same. It is obvious that the logistic regression can barely predict whether the data is zero or not and GLM performance is not good as well. I think both data need more tricks for data processing. Like I said before, sampling the non-zero data to make it as equal as the numbers of zero data does not improve the performance at all. We might need to change models or doing more sophisticated data processing.