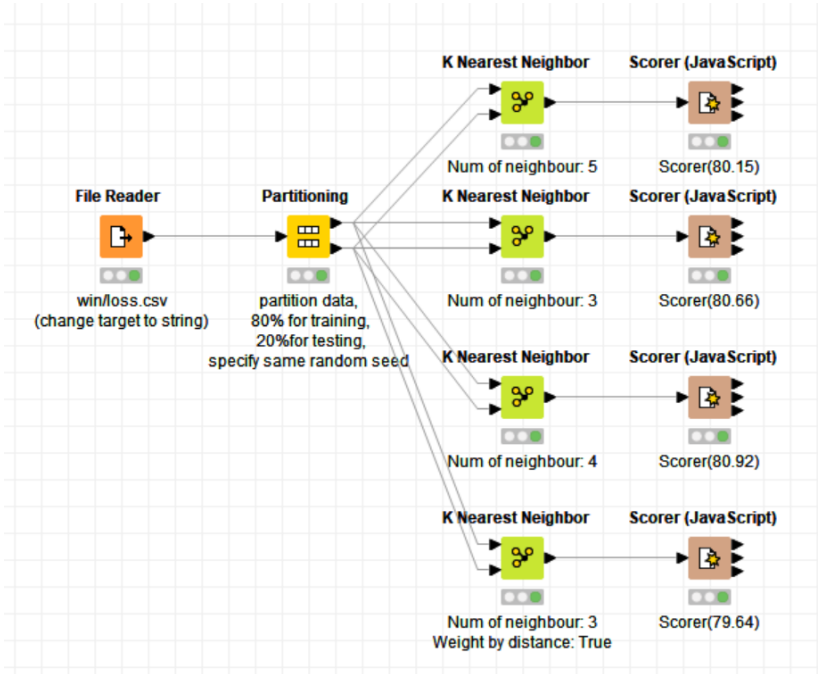
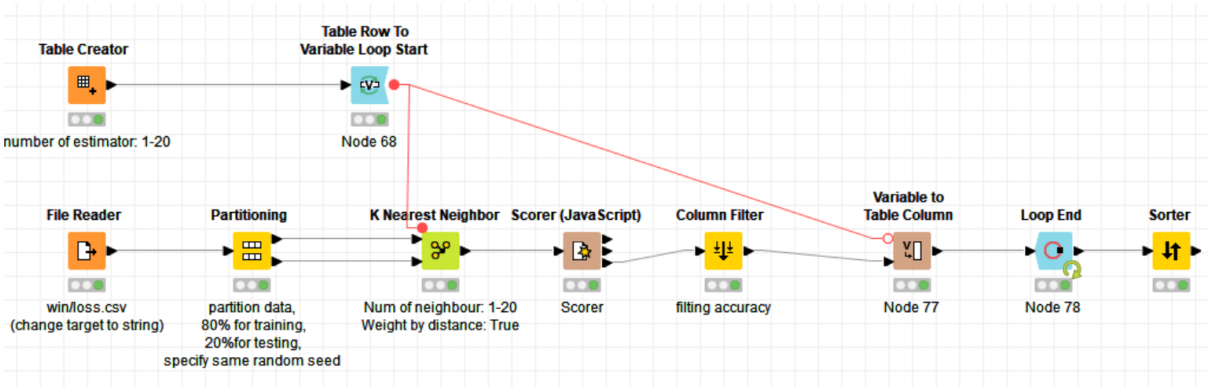
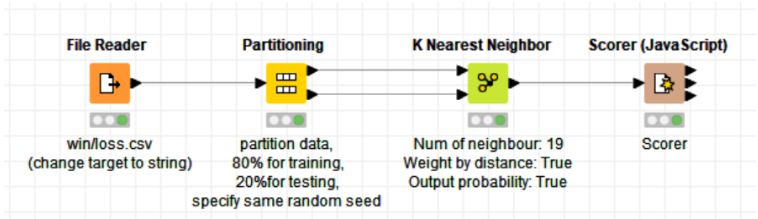
Overall workflow (with comment explanations):

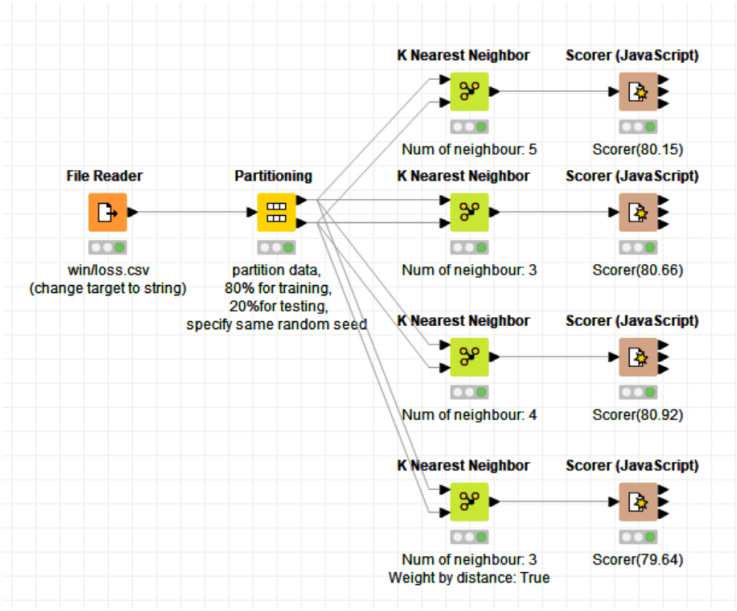




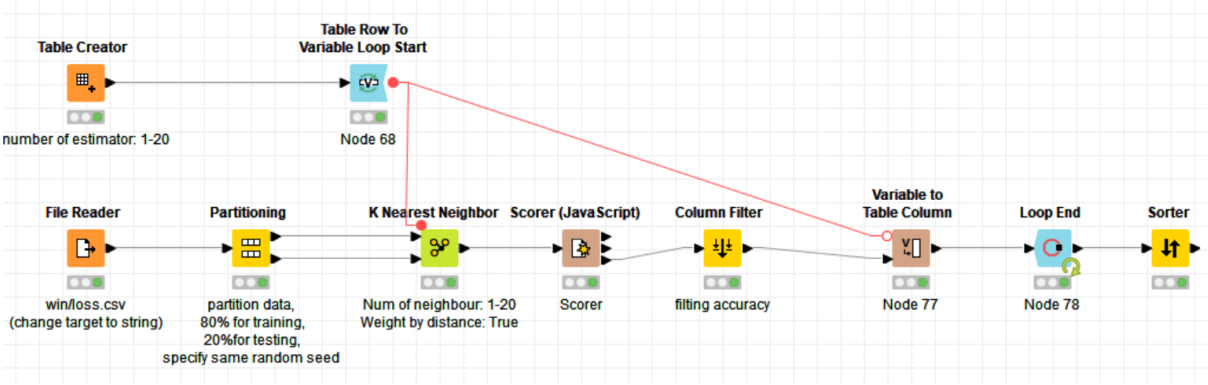


Question1:

In the first question, I created a workflow for k nearest neighbor, with win/loss dataset. I tried different parameters for the number of neighbours and whether closer neighbours should have greater influences. I compare them by total accuracy for each prediction.



But I think this is complex to find the best parameter. So, I created a prediction with loop.



By this workflow, I tried number of neighbours from 1-20, set Weight by distance is True. Then, the result for accuracy is 0.83, when number of neighbours equals 19.

Row ID	Overall ...	k	Iteration
Overall#18	0.83	19	18
Overall#19	0.824	20	19
Overall#16	0.819	17	16
Overall#3	0.814	4	3
Overall#8	0.814	9	8
Overall#14	0.814	15	14
Overall#11	0.812	12	11
Overall#17	0.812	18	17
Overall#12	0.809	13	12
Overall#13	0.809	14	13
Overall#10	0.807	11	10
Overall#15	0.807	16	15
Overall#9	0.804	10	9
Overall#4	0.802	5	4
Overall#5	0.802	6	5
Overall#2	0.796	3	2
Overall#7	0.796	8	7
Overall#6	0.791	7	6
Overall#0	0.786	1	0
Overall#1	0.786	2	1

When there is no weight by distance,

Row ID	Overall ...	k	Iteration
Overall#18	0.824	19	18
Overall#13	0.817	14	13
Overall#16	0.817	17	16
Overall#19	0.817	20	19
Overall#14	0.814	15	14
Overall#17	0.814	18	17
Overall#8	0.812	9	8
Overall#15	0.812	16	15
Overall#3	0.809	4	3
Overall#11	0.809	12	11
Overall#12	0.809	13	12
Overall#2	0.807	3	2
Overall#4	0.802	5	4
Overall#10	0.802	11	10
Overall#7	0.799	8	7
Overall#5	0.796	6	5
Overall#9	0.796	10	9
Overall#6	0.791	7	6
Overall#0	0.784	1	0
Overall#1	0.768	2	1

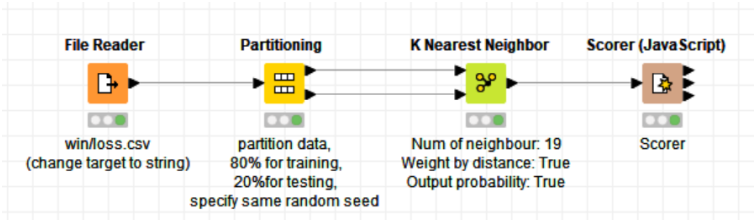
The best accuracy is 0.824, less than above (with weight by distance). The number of neighbours equals 19, which is the same.

So, the best parameter for this dataset and partitioning is:

Accuracy: 0.83, number of neighbours: 19, weight by distance: True.

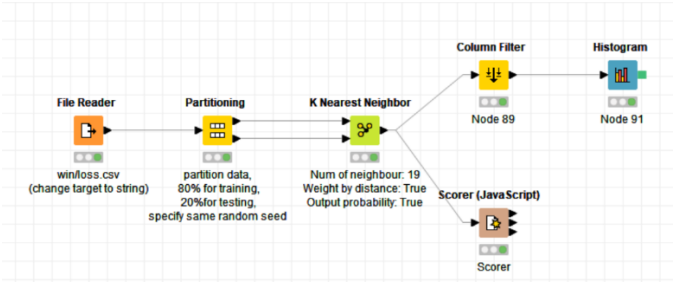
Question 2:

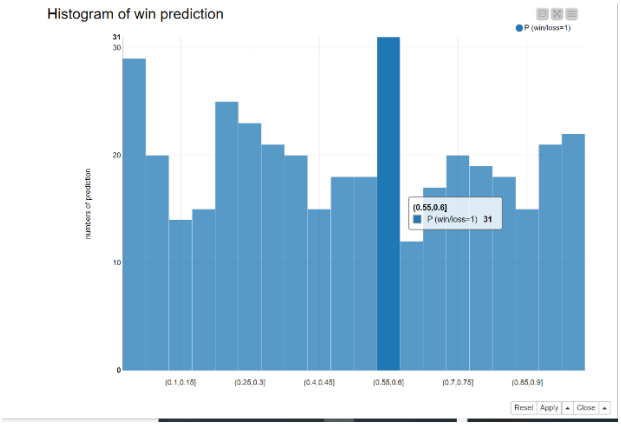
Setting probabilities in predictions. I'm using the best parameter from above.



S Class [k...	D P (win/l...	D P (win/l...
1	0.426	0.574
1	0.477	0.523
0	0.69	0.31
1	0.336	0.664
1	0.05	0.95
1	0.407	0.593
1	0.167	0.833
1	0.154	0.846
1	0.294	0.706
1	0.218	0.782
0	0.64	0.36
1	0.352	0.648
1	0.458	0.542
1	0.164	0.836
1	0.369	0.631
1	0.475	0.525
1	0.097	0.903
1	0.096	0.904
1	0.198	0.802
0	0.615	0.385
1	0	1

There are extra two columns for the probabilities associates with each row's prediction. These two columns show the probability of predict 1s and 0s. Since these two columns shows similar information, the sum of them is 1. So, I'm using one of the columns, probability for 1, to show its distribution.





It is evenly distributed. So, most classifications are clear out (in the interval [0,0.45], [0.55,1]).

However, by this interactive histogram, there is still 18 prediction between 0.45 and 0.5, 18 prediction between 0.5 and 0.55. These are hard to predict.

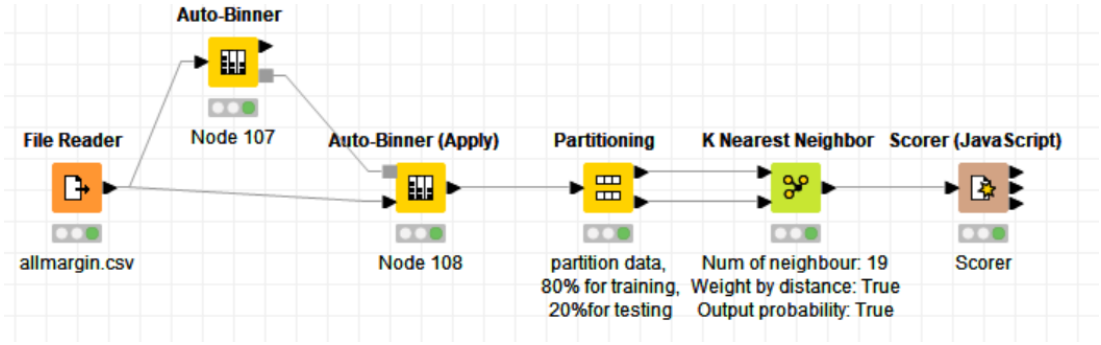
Question 3

Since I does not find KNN regressor in KNIME, I will not use the regression method.

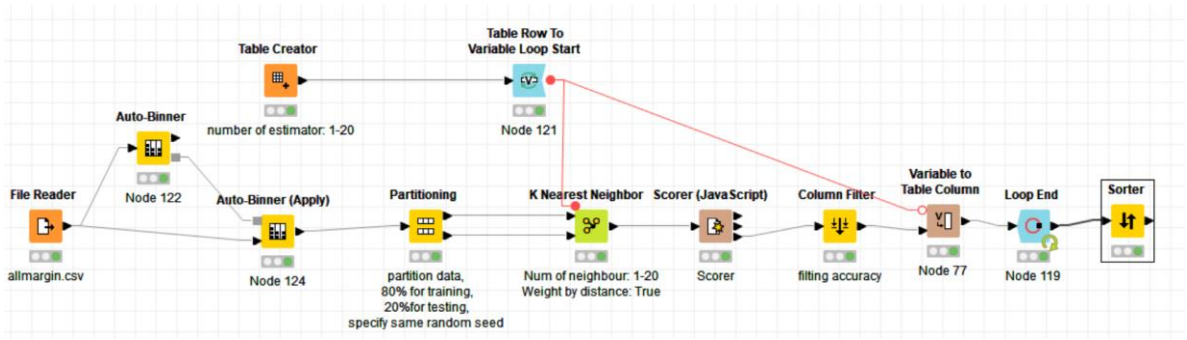
(such like in python, there is KNeighboursRegressor)

So, I will use discretization and classification for margin.csv. (like homework 2)

I used auto-binner for binning the target column into 2 bins. Then I set the binning by frequency (or widths, we try discretize by frequency first), naming them by border. (suppose larger margin group is win, lower is loss). Then apply auto-binner into the dataset. After that, apply K nearest neighbor.

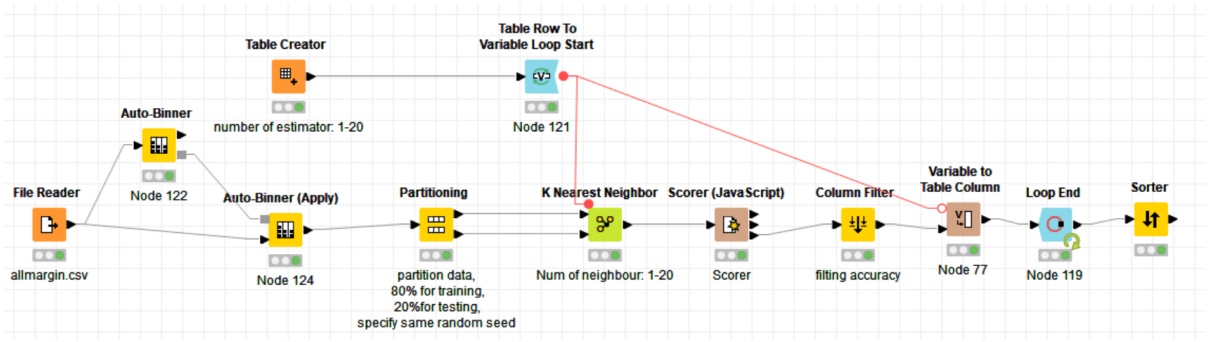


I use loop to try different parameters.



Row ID	D Overall ...	I k	I Iteration
Overall#18	0.83	19	18
Overall#19	0.824	20	19
Overall#16	0.819	17	16
Overall#3	0.814	4	3
Overall#8	0.814	9	8
Overall#14	0.814	15	14
Overall#11	0.812	12	11
Overall#17	0.812	18	17
Overall#12	0.809	13	12
Overall#13	0.809	14	13
Overall#10	0.807	11	10
Overall#15	0.807	16	15
Overall#9	0.804	10	9
Overall#4	0.802	5	4
Overall#5	0.802	6	5
Overall#2	0.796	3	2
Overall#7	0.796	8	7
Overall#6	0.791	7	6
Overall#0	0.786	1	0
Overall#1	0.786	2	1

By this workflow, I tried number of neighbours from 1-20, set Weight by distance is True. Then, the result for accuracy is 0.83, when number of neighbours equals 19.



Row ID	Overall ...	k	Iteration
Overall#18	0.824	19	18
Overall#13	0.817	14	13
Overall#16	0.817	17	16
Overall#19	0.817	20	19
Overall#14	0.814	15	14
Overall#17	0.814	18	17
Overall#8	0.812	9	8
Overall#15	0.812	16	15
Overall#3	0.809	4	3
Overall#11	0.809	12	11
Overall#12	0.809	13	12
Overall#2	0.807	3	2
Overall#4	0.802	5	4
Overall#10	0.802	11	10
Overall#7	0.799	8	7
Overall#5	0.796	6	5
Overall#9	0.796	10	9
Overall#6	0.791	7	6
Overall#0	0.784	1	0
Overall#1	0.768	2	1

The best accuracy is 0.824, less than above (with weight by distance). The number of neighbours equals 19, which is the same.

Then, try to discretize the data by width, with the same workflow. The result is the same as discretize data by frequency.

So, the best parameter for this dataset and partitioning is:

Accuracy: 0.83, number of neighbours: 19, weight by distance: True.

For both question 1 and 3, the best parameters are the same.

4. For each prediction, look at the true positive, true negative, false positive, false negatives.

We need to know the accuracy for predicting winner and loser of the game.

For example, if we use the best parameter for this dataset above, by the classification workflow in question 1:

Scorer View

Confusion Matrix

Rows Number : 393	0 (Predicted)	1 (Predicted)	
0 (Actual)	165	32	83.76%
1 (Actual)	35	161	82.14%
	82.50%	83.42%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
0	165	35	161	32	83.76%	82.50%	83.76%	82.14%	83.12%
1	161	32	165	35	82.14%	83.42%	82.14%	83.76%	82.78%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
82.95%	17.05%	0.659	326	67

By the score overview, the accuracy for predict 0 (loss) is 83.76%, the accuracy for predict 1(win) is 82.14%. So, by the best parameter, it is more difficult to predict the winner.

It might be different when have different partition and parameters, we can use this method for each prediction, to decide which side is more difficult.