Part 1: Classification

1. Run the following classifiers, with the default parameters, on this data: ZeroR, OneR, J48, IBK and construct a table of training and cross-validation errors. You can get the training error by selecting "Use training set" as the test option. what do you conclude from the results?

Run No	Classifier	Parameters	Training Error	Cross-valid Error	Over-Fitting
1	ZeroR	None	50.0%	50.0095%	(0.0095%)None
2	OneR	B=6	26.1297%	29.9017%	(3.772%)None
3	J48	C=0.25 & M=2	10.6825%	14.6436%	(3.9611%)None
4	IBK	K=1 & W=0	0%	28.1244%	(28.1244%)Yes

Result Analysis: From above table, we can see that ZeroR Error is 50.0095%, OneR is 29.9017%, J48 is 14.6436% and IBK is 28.1244%. Compared with Training Error for Classifier ZeroR, OneR and J48, just slightly higher than their training Error, so we deem there is no overfitting for these three Classifiers. However, for the Classifier IBK, the Cross-valid Error is great than training Error, so it's overfitting. J48 get the highest performance compared with others, ZeroR get the lowest.

2. Using the J48 classifier, can you find a combination of the C and M parameter values that minimizes the amount of overfitting? Include the results of your best five runs, including the parameter values, in your table of results

Parameters values		Training Error	Cross-valid Error	Over-Fitting ("Cross-Valid Error"- "Training Error")
C	M			
0.25	20	13.5753%	15.277%	1.7017%
	50	14.6341%	15.8631%	1.229%
0.5	50	14.6341%	15.8253%	1.1912%
0.75	50	14.6058%	15.8442%	1.2384%
1	50	14.6058%	15.8442%	1.2384%

Result Analysis: I run 24 times [C (=0.25/0.5/0.75/1) * M (=2/3/5/10/20/50)] and found out when C=0.25 and M=50 the amount of overfitting is lowest. And other best five runs as above table.

3. Reset J48 parameters to their default values. What is the effect of lowering the number of examples in the training set? Include your runs in your table of result

Classifier J48	Percentage (training set)	Accuracy
C=0.25 M=2	99%	78.3019%
	95%	85.2552%
	92%	85.8156%
	90%	85.5388%
	80%	84.8771%
	70%	85.2506%
	66%	85.1821%
	60%	84.6136%
	50%	83.8722%
	40%	83.7088%
	30%	83.6192%
	20%	83.2782%
	10%	81.3025%
	1%	74.3029%

Result Analysis: From above table, we know that when the percentage of training set is 99% the accuracy is 78.3019% which is lowest. Then the percentage decreases from 99% to 90% the accuracy increase to 85.5388% which is the highest. Then the percentage reduces from 90% to 1% the accuracy is decreasing.

4. Using the IBK classifier, can you find the value of k that minimizes the amount of overfitting? Include your runs in your table of results.

Run No	Parameters (K)	Training Error	Cross-valid Error	Over-Fitting ("Cross-valid Error" – "Training Error")
1	K=1	0%	28.1244%	28.1244%
2	K=2	12.1573%	28.5309%	16.3736%
3	K=4	15.3526%	25.9406%	10.588%
4	K=8	18.6803%	25.052%	6.3717%
5	K=16	20.9113%	24.8629%	3.9516%
6	K=32	23.114%	25.0709%	1.9569%
7	K=64	24.8818%	26.2715%	1.3897%
8	K=128	26.2999%	27.0089%	0.709%
9	K=256	27.5383%	28.1433%	0.605%
10	K=512	28.616%	29.0509%	0.4349%
11	K=1024	29.1548%	29.4952%	0.3404%

Result Analysis: From above table, I just picked up 11 values for K [1/2/4/8/16/32/64/128/256/512/1024]. we know that when k equal to 1024 the amount of overfitting is lowest. However, I am not sure in which specific value of k the amount of overfitting is lowest as k is increasing.

5. Try a number of other classifiers. Aside from ZeroR, which classifiers are best and worst in terms of predictive accuracy? Include 5 runs in your table of results.

Run No	Classifier	Accuracy (Cross-validation)	Accuracy (Percentage-split)
1	OneR(B=6)	70.0983%	70.9202%
2	OneR(B=1)	57.8748 %	56.6861 %
3	OneR(B=2)	69.3609 %	55.9355 %
4	OneR(B=3)	69.654 %	70.4476 %
5	OneR(B=4)	69.8809 %	70.6144 %
6	J48(M=2)	89.3175 %	85.1821%
7	J48(M=4)	85.3375 %	84.5705 %
8	J48(M=6)	85.0633 %	85.0431 %
9	J48(M=8)	85.1673 %	84.9319 %
10	J48(M=10)	85.1106 %	84.7929 %
11	IBK	71.8756%	70.9758%

Result Analysis: From above table, we can see that J48 is best and OneR is worst in terms of predictive accuracy.

6. Compare the accuracy of ZeroR, OneR and J48.What do you conclude?

Run No	Classifier	Accuracy (Cross-validation)	Accuracy(Percentage-split 66%)
1	ZeroR	49.9905%	49.6247%
2	OneR	70.0983%	70.9202%
3	J48	85.3564%	85.1821%
4	IBK	71.8756%	70.9758%

Result Analysis: From above table, we know that the highest accuracy is J48 and lowest accuracy is ZeroR. The model is more complex; the accuracy is more high.

7. What golden nuggets did you find, if any?

Compared with other classifiers J48 is better to predict accuracy. Different classifiers will have different accuracy as the different parameters.

Part 2: Numeric Prediction

Numeric Prediction of the *balance* attribute in the bank data of part 1. The main goal is to achieve the lowest mean absolute error with the lowest amount of over-fitting.

1. Run the following classifiers, with default parameters, on this data: ZeroR, MP5, IBk and construct a table of the training and cross-validation errors. You may want to turn on "Output Predictions" to get a better sense of the magnitude of the error on each example. What do you conclude from these results?

Run No	Classifier	Parameters	Training Error (Relative absolute error)	Cross-valid Error(Relative absolute error)	OverFitting ("Cross-Valid Error" – "Training Error")
1	ZeroR	None	100%	100%	(0) None
2	MP5	M=4	92.5365%	97.4332%	(4.8967%) Yes
3	IBk	K=1	0%	123.8191%	(123.8191%) Yes

Result Analysis: From above table, we know that the accuracy is not good. Which means that attribute is not significant or not dominant.

2. Explore different parameter settings for M5P and IBK. Which values give the best performance in terms of predictive accuracy and overfitting. Include the results of the best runs in your table of results.

Run No	Classifier	Parameters	Correlation coefficient (Training/Cross-valid)	Training Error (Relative absolute error)	Cross-valid Error(Relative absolute error)	OverFitting ("Cross-Valid Error" – "Training Error")
1	M5P	256	0.2069/0.1905	95.5104%	95.9139%	0.1073%
2		512	0.2015/0.1893	95.8085%	95.9158%	0.1073%
3	IBK	200	0.2234/0.1928	95.5173 %	96.1013 %	
4		2000	0.1807/0.1751	97.3096 %	97.4537 %	0.1441%

Result Analysis: I run 40 times for the classifier M5P [(batchsize=50/100/200/500) * M (2/4/8/16/32/64/128/256/512/1024)] and found that there is no impact for the performance when the value of batchsize is increasing. When M=256 I got the best performance which the correlation coefficient is 0.1905. When M=256 or 512 I obtained the lowest amount of overfitting which is 0.1073%. however, we do not know in which specific value of M between 128 and 512 we can get the best performance. And get the lowest amount of overfitting.

For the classifier IBK, I run 12 times (K=1/5/10/20/50/100/200/500/1000/1500/2000/2500). When K=200 I got the best performance (Correlation coefficient = 0.1928 for the testing set) for the classifier IBK and when K=2000 the lowest amount of overfitting. However, we have no idea in which specific value of K we can get the best performance.

The whole performance is not good, so we think that attribute is not important.

3. Investigate three other classifiers for numeric prediction and their associated parameters. Include your best five runs in your table of results. Which classifier gives the best performance in terms of predictive accuracy and overfitting?

Run No	Classifier	Parameters	Correlation coefficient (Training/Cross-valid)	Training Error (Relative absolute error)	Cross-valid Error(Relative absolute error)	OverFitting ("Cross-Valid Error" – "Training Error")
1	Decision Table	Batchsize (20/50/100 /200/300)	0.1847/0.1637	96.2565 %	96.7988 %	0.5423%
2	Ramdom tree		1/0.044	0.7221 %	130.8572 %	130.8572 %
3	REPtree		0.2827/0.0857	94.2446 %	99.4384 %	5.1938%

Result Analysis: Decisiontable gives the best performance compared with other two classifiers. And also the amount of overfitting is lowest.

4. What golden nuggets did you find, if any?

For the same dataset to use different classifiers, the result may have huge difference.

Part 3: Clustering

Clustering of the bank data of part 1. For this part use only the attributes age, marital, education and balance.

The aim is determining the number of clusters in the data and assess whether any of the clusters are meaningful.

1) Run the Kmeans clustering algorithm on this data for the following values of K: 1,2,3,4,5,10,20. Analyze the resulting clusters. What do you conclude?

Result Analysis: From above table, we can easily know that if the value of K is too small (for example K=1) all the examples will be closer together, if the value of K is too big the clustered instance will be more similar relatively.

2) Choose a value of K and run the algorithm with different seeds. What is the effect of changing the seed?

The value of K	seeds	Cluster	age	martial	education	balance	Clustered Instance
2	1	0	33.2655	single	secondary	1456.5727	3698 (35%)
		1	45.4382	married	secondary	1633.3785	6880 (65%)

	5	0	43.5073	married	tertiary	1934.026	4656 (44%)
		1	39.3551	married	secondary	1286.5966	5922 (56%)
	10	0	33.2655	single	secondary	1456.5727	3698 (35%)
		1	45.4382	married	secondary	1633.3785	6880 (65%)
	20	0	46.3819	married	secondary	1545.3981	6295 (60%)
		1	33.5412	single	tertiary	1610.0325	4283 (40%)
	50	0	54.0048	married	secondary	1844.7555	3942 (37%)
		1	33.566	married	secondary	1409.2863	6636 (63%)
	100	0	38.4522	married	tertiary	1875.0765	4065 (38%)
		1	42.887	married	secondary	1382.1379	6513 (62%)
	200	0	33.2655	single	secondary	1456.5727	3698 (35%)
		1	45.4382	married	secondary	1633.3785	6880 (65%)
	300	0	33.6538	single	tertiary	1604.9213	4359 (41%)
		1	46.4599	married	secondary	1548.1907	6219 (59%)
	500	0	45.4382	married	secondary	1633.3785	6880 (65%)
		1	33.2655	single	secondary	1456.5727	3698 (35%)
	600	0	45.4382	married	secondary	1633.3785	6880 (65%)
		1	33.2655	single	secondary	1456.5727	3698 (35%)
	700	0	45.4382	married	secondary	1633.3785	6880 (65%)
		1	33.2655	single	secondary	1456.5727	3698 (35%)
	800	0	45.4382	married	secondary	1633.3785	6880 (65%)
		1	33.2655	single	secondary	1456.5727	3698 (35%)

Result Analysis: I selected the values [1/5/10/20/50/100/200/300/500/600/700/800] of seeds as the testing. When the value of seeds is decreasing, the ratio of cluster 0 is increasing to 44% and then decreased to 35% when the value of seeds is 1. While the value of seeds is increasing, the ratio of cluster 0 is increasing then decreasing then increasing again until the value of seeds is 500 and the ratio of cluster 0 reached the highest which is 65%. When the value of seeds exceeded 500 the ratio of cluster 0 keep unchanged.

3) Run the EM algorithm on this data with the default parameter and describe the output.

Attribute	Cluster									
age		0	1	2	3	4	5	6	7	8
	Mean	50.9025	59.9904	30.7283	28.5446	51.5947	35.5395	42.2766	46.1046	34.6122
	Std.dev.	11.015	13.8276	4.6754	3.6502	12.914	4.389	11.1094	8.0763	5.5819
marital	Single	136.0826	3.3313	1173.2365	783.5311	41.4513	390.6714	162.7493	278.4626	379.4839
	Married	1533.4332	248.9625	594.8768	9.6991	238.243	943.5874	445.5546	1724.7073	269.936
	divorc	312.01	71.47	70.020	2.3385	43.761	107.31	118.9	500.73	20.427

	ed	37	79	2		6	24	1	85	2
	Total	1981.5295	323.7717	1838.1336	795.5686	323.4559	1441.5713	727.2139	2503.9085	669.8471
education	tertiary	565.8276	43.1004	594.1249	360.2252	121.9741	603.9194	239.8241	551.0881	373.9164
	secondary	881.7798	62.3121	1047.9471	367.947	119.171	770.5083	372.158	1387.3431	242.8335
	unknown	151.2905	40.464	66.1897	44.4963	15.587	9.6944	22.9106	119.6727	23.6947
	primary	383.6316	178.8953	130.8719	23.9	67.7238	58.4492	93.3211	446.8045	30.4026
	total	1982.5295	324.7717	1839.1336	796.5686	324.4559	1442.5713	728.2139	2504.9085	670.8471
balance	mean	2511.757	454.0952	226.9217	1433.8054	12101.8974	1008.9087	0.009	350.1323	5624.7015
	Std. dev.	1587.4325	368.0193	210.2198	1098.7911	10158.5329	764.1577	0.1034	389.7713	389.7713

Clustered Instances								
0	1	2	3	4	5	6	7	8
1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)

Result description: For the numeric attributes, EM will output the mean and standard deviation value for every cluster, for the nominal attributes EM will display the number of every value for the attribute and the total number of that attribute. EM will also output all the clusters and the ratio of every cluster.

4) The EM algorithm can be quite sensitive to whether the data is normalized or not. Use the weka normalize filter (Preprocess→Filter→unsupervised→normalize) to normalize the numeric attributes. What difference does this make to the clustering runs?

Before normalization:

Attribute	Cluster 0 (0.19)	1 (0.03)	2 (0.17)	3 (0.07)	4 (0.03)	5 (0.14)	6 (0.07)	7 (0.24)	8 (0.06)
age									
mean	50.9025	59.9904	30.7283	28.5446	51.5947	35.5395	42.2766	46.1046	34.6122
std. dev.	11.015	13.8276	4.6754	3.6502	12.914	4.389	11.1094	8.0763	5.5819

After normalization:

Attribute	Cluster 0 (0.19)	1 (0.03)	2 (0.17)	3 (0.07)	4 (0.03)	5 (0.14)	6 (0.07)	7 (0.24)	8 (0.06)
age									
mean	0.4273	0.5453	0.1653	0.1369	0.4363	0.2278	0.3153	0.365	0.2157
std. dev.	0.1431	0.1796	0.0607	0.0474	0.1677	0.057	0.1443	0.1049	0.0725

From above two screenshots, we can see that the values of mean and std. dev. for the attribute “age” that after normalization has been changed into between 0 and 1.

5) The algorithm can be quite sensitive to the values of minLogLikelhoodImprovementCV minStdDev and minLogLikelihoodImprovementIterating, Explore the effect of changing these values. What do you conclude?

Keep other parameters’ value default, just change the value of minStdDev

	Clustered Instances											
minStdDev (Time taken)	0	1	2	3	4	5	6	7	8	9	10	11
1.0E-6 (258.94s)	1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)	-	-	-
0.001 (350.05)	1617 (15%)	518 (5%)	1045 (10%)	1037 (10%)	505 (5%)	2217 (21%)	431 (4%)	1879 (18%)	518 (5%)	721 (7%)	90 (1%)	-
0.01 (404.24s)	112 (1%)	761 (7%)	1451 (14%)	568 (5%)	1592 (15%)	1231 (12%)	721 (7%)	1321 (12%)	572 (5%)	770 (7%)	1273 (12%)	20 (2%
0.1 (269.42s)	1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)	-	-	-
1 (260.93s)	1769 (17%)	180 (2%)	2164 (20%)	766 (7%)	246 (2%)	1275 (12%)	826 (8%)	2805 (27%)	547 (5%)	-	-	-
10 (70.33s)	2177 (21%)	1021 (10%)	965 (9%)	3011 (28%)	1684 (16%)	928 (9%)	792 (7%)	-	-	-	-	-
100 (129.84)	2665 (25%)	1515 (14%)	850 (8%)	3081 (29%)	1848 (17%)	276 (3%)	343 (3%)	-	-	-	-	-

Result Analysis: I selected [1.0E-6/0.001/0.01/0.1/1/10/100] for the parameter “minStdDev” as testing. When the value is increasing from 1.0E-6 to 0.01 the number of “clustered instances” is extended to 12 which is the biggest, the “time taken” is increasing correspondingly. However, when the value continues to increase the number of “clustered instances” decreased to 9. For the cluster that have same number of instances the ratio is also not same. But the “time taken” is decreasing.

Keep other parameters default, just change the value of MinLogLikelhoodImprovementCV

	Clustered instance									
MinLog Likelhood Improveme nt CV(time taken)	0	1	2	3	4	5	6	7	8	9
1.0E-6 (259.15s)	1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)	-
0.001 (259.42s)	1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)	-
0.01 (259.36s)	1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)	-
0.1 (70.03 s)	4329 (41%)	927 (9%)	1814 (17%)	3508 (33%)	-	-	-	-	-	-
1(5.24s)	10578 (100%)	-	-	-	-	-	-	-	-	-
10(5.3s)	10578 (100%)	-	-	-	-	-	-	-	-	-
100(5.14s)	10578 (100%)	-	-	-	-	-	-	-	-	-

Result Analysis: I selected [1.0E-6/0.001/0.01/0.1/1/10/100] for the parameter "MinLogLikelhoodImprovementCV". When its value is increasing, the number of "Clustered instance" is decreasing to 1 and the "time taken" is improving.

Keep other parameters default, just change the value of minLogLikelihoodImprovementIterating

	Clustered Instances									
minLogLikel ihoodImpro vementIter ating (time taken)	0	1	2	3	4	5	6	7	8	9
1.0E-6 (259.15s)	1804 (17%)	161 (2%)	2197 (21%)	774 (7%)	249 (2%)	1286 (12%)	721 (7%)	2830 (27%)	556 (5%)	-

0.001 148.99(s)	1462 (14%)	445 (4%)	2085 (20%)	832 (8%)	236 (2%)	998 (9%)	2241 (21%)	1677 (16%)	602 (6%)	-
0.01 (60.36s)	884 (8%)	1315 (12%)	1026 (10%)	2000 (19%)	2606 (25%)	509 (5%)	1431 (14%)	807 (8%)	-	-
0.1 (7.76s)	4686 (44%)	2983 (28%)	2909 (28%)	-	-	-	-	-	-	-
1 (38.8 s)	549 (5%)	1276 (12%)	1663 (16%)	1384 (13%)	493 (5%)	1008 (10%)	1040 (10%)	2968 (28%)	197 (2%)	-
10 (39.6s)	549 (5%)	1276 (12%)	1663 (16%)	1384 (13%)	493 (5%)	1008 (10%)	1040 (10%)	2968 (28%)	197 (2%)	-
100 (38.6s)	549 (5%)	1276 (12%)	1663 (16%)	1384 (13%)	493 (5%)	1008 (10%)	1040 (10%)	2968 (28%)	197 (2%)	-

Result Analysis: I selected [1.0E-6/0.001/0.01/0.1/1/10/100] for the parameter "minLogLikelihoodImprovementIterating". When its value increases from 1.0E-6 to 0.1 the number of "clustered instances" decreased from 9 to 3, the corresponding "time taken" decreased to 7.76 seconds. When its value continues to increase from 0.1 to 100, the number of "clustered instances" extended to 9. But the "time taken" has a little bit increase

6) How many clusters do you think are in the data? Give an English language description of one of them.

It's hard to say how many, different parameters will generate different number of clusters. But with above testing, I think there are 9 clusters.

7) Compare the use of Kmeans and EM for these clustering tasks. Which do you think is best? Why?

I think Kmeans is better. Because there are too many parameters to effect the number of "cluster" for EM and too much information is outputted. Easy is perfect.

8) What golden nuggets did you find, if any?

Different parameters will generate different results (different clustered instances).

Part 4: Association Finding

The main aim is to determine whether there are any significant associations in the data.

These files contain the same details of shopping transactions represented in two different ways. You can use a text viewer to look at the files.

1) What is the difference in representations?

In the file supermarket1.arff, the value for the attribute is in "{f, t}", however in the file supermarket2.arff, it's "{ t}".

2) Load the file supermarket1.arff into weka and run the Apriori algorithm on this data. You might need to restrict the number of attributes and/or the number of examples. What significant associations can you find?

I only selected baby needs, bread and cake, baking needs, coupons, juice-sat-cord-ms, tea,biscuits,canned fish-meat, canned fruit, canned vegetables, breakfast food as testing.

Best rules:

People purchases baby needs or tea or canned fish-meat or canned fruit or "break and cake" they will use coupons.

People purchases baby needs and tea or canned fish-meat they will use coupons.

People purchases tea and fish-meat they will use coupons.

People purchases canned vegetables they will use coupons.

People purchases baby needs and canned fruit they will use coupons.

3) Explore different possibilities of the metric type and associated parameters, what do you find?

MetricType	Generated sets of large itemsets:
Confidence (MinMetirc=0.9)	Size of set of large itemsets L(1): 8 Size of set of large itemsets L(2): 12 Size of set of large itemsets L(3): 5
Lift (MinMetirc=1.1)	Size of set of large itemsets L(1): 12 Size of set of large itemsets L(2): 36 Size of set of large itemsets L(3): 37 Size of set of large itemsets L(4): 12
Leverage (MinMetric=0.1)	Size of set of large itemsets L(1): 21 Size of set of large itemsets L(2): 163 Size of set of large itemsets L(3): 582 Size of set of large itemsets L(4): 1148 Size of set of large itemsets L(5): 1272 Size of set of large itemsets L(6): 766 Size of set of large itemsets L(7): 234 Size of set of large itemsets L(8): 32 Size of set of large itemsets L(9): 1
Conviction (MinMetric=1.1)	Size of set of large itemsets L(1): 10 Size of set of large itemsets L(2): 21 Size of set of large itemsets L(3): 13 Size of set of large itemsets L(4): 1

Different metricType will generate different rules(itemsets). Even for the minMetric that have same value, the generated sets are different.

4) Load the file supermarket22.arff into weka and run the Apriori algorithm on this data. What do you find?

There are no rules under “Best rules found”

5) Explore different possibilities of the metric type and associated parameters. What do you find?

MetricType	Generated sets of large itemsets:
Confidence (MinMetirc=0.9)	Size of set of large itemsets L(1): 10 Size of set of large itemsets L(2): 32 Size of set of large itemsets L(3): 38 Size of set of large itemsets L(4): 15 Size of set of large itemsets L(5): 2
Lift (MinMetirc=1.1)	Size of set of large itemsets L(1): 6 Size of set of large itemsets L(2): 7 Size of set of large itemsets L(3): 1
Leverage (MinMetric=0.1)	Size of set of large itemsets L(1): 10 Size of set of large itemsets L(2): 32 Size of set of large itemsets L(3): 38 Size of set of large itemsets L(4): 15 Size of set of large itemsets L(5): 2
Conviction (MinMetric=1.1)	Size of set of large itemsets L(1): 6 Size of set of large itemsets L(2): 7 Size of set of large itemsets L(3): 1

When the MetricType is “Confidence” or “Leverage”, they get the same results. When MetricType is “Lift” or “Conviction” both of them have same outcomes as well.

6) Try the other associators. What are the difference to Apriori?

For the “FilterredAssociator”, in default, which generated a lot of “sets of large itemsets”. For the “Best rules found” it’s diverse.
For example,
if people purchase “bread and cake” they will purchase “baby needs”.
if people purchase “baby needs” they will purchase “bread and cake”.

For the “FPGrowth”, no rules found.

7) What golden nuggets did you find, if any?

Different associator will generate different result, we need to find one that is best suitable for our case.

8) [Optional] Can you find any meaningful associations in the bank data?