PART 1: CLASSIFICATION WITH NEURAL NETWORKS

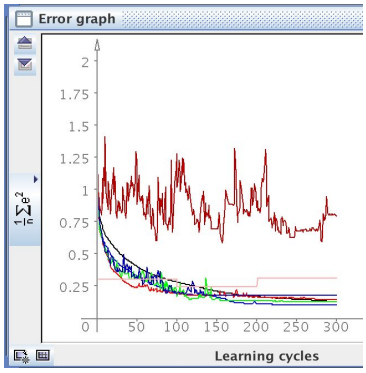
1.) There will be 22 input nodes - for each possible value of the attributes one node, except for binary attributes like 'sex' which can be used in one node using 1 = male and 0 = female, opposed to using 2 nodes. This shall keep the amount of nodes lower as a high amount of nodes might affect overall performance. Nodes are as following: 'age'= 1, 'sex' = 1, 'cp' = 4, 'trestbps' = 1, 'chol' = 1, 'fbs' =1, 'restecg' = 3, 'thalach' = 1, 'exang' = 1, 'oldpeak = 1, 'slope = 3, 'ca' = 1 'thal' = 3: that makes 1 + 1 + 4 + 1 + 1 + 1 + 3 + 1 + 1 + 1 + 3 + 1 + 3 = 22 Nodes. There will be five output nodes - for each possible value of the "predicted" attribute 'num' one node: '<50', '>50_1', '>50_2', '>50_3', '>50_4' = 5

2.) To prepare the pat files I first went into the heart-v.arff and tried to understand how the data structure is build up. Then I normalised the data under Weka.. As a third task I used the provided javanns-env.sh as basis for my heart_jnns.sh script. Following Task 1 understanding of the amount of input and output notes I modified the script to incorporated this. There are 605 data sets/patterns but 14 of them were incomplete which I decided to deleted from the arff file thus reducing to 591 data sets. Considering the amount of patterns given it shouldn't have any significant effect on results - a reduction of 2.3% of the total amount of patterns. I amend the script to divide the 591 sets into 67% (394 sets) training data, 20 % validation data (118 sets) and 13 % test data (77 sets) together with the mapping of nominal attributes into N-width binary strings. Running the script gave me three necessary pat files (for test, validation and training).

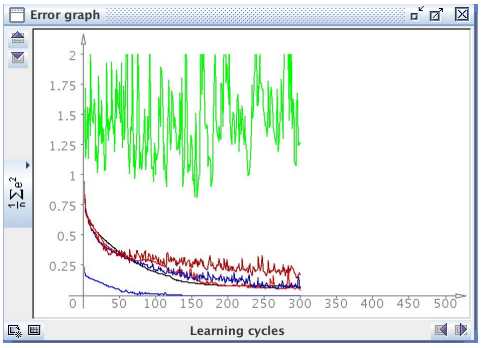
3.) For comparison reason I left the error method at WTA as it defines between error and correct classified patterns unlike 402040 which has a certain amount of unknown. I choose to gradually increase the Epochs to understand the test error behaviors with increasing cycles. The expectation is that with increasing epochs the error reduces to a certain point. My observation is that the more epochs the lower the training error until a certain error rate (at around 500-600 Epochs). On the other hand the Test error rose a bit between 50 to 100 Epochs and then also dropped a bit to the error rate it had at 50 epochs. The overfitting is significant, especially at 200 epochs where it is at it's highest.

Run No	Architecture	Epochs	Parameters	Train MSE	Train Error	Test MSE	Test Error	Difference	Overfitting
1	22-5-5	50	lr = 2.0	0.126982	6.09%	0.242951	14.29%	8.20%	significant
2	22-5-5	100	lr = 2.0	0.100134	4.57%	0.269946	18.18%	13.61%	significant
3	22-5-5	200	lr = 2.0	0.056393	2.28%	0.236002	16.88%	14.60%	significant
4	22-5-5	400	lr = 2.0	0.043364	1.78%	0.246199	15.58%	13.80%	significant
5	22-5-5	600	lr = 2.0	0.035254	1.27%	0.254208	14.29%	13.02%	significant

4) In this task I choose to do 10 runs with two different epochs (50 & 600) on 5 different architectures consisting of each of 5, 10, 15, 20, 30 hidden notes but always 22 input and 5 output nodes. Using WTA I captured the train and test error. Generally spoken it seems 15 hidden nodes give the best result at 600 epochs. Train (1.27 %) and test (3.90 %) was smallest as was the overfitting (2.63 %). At 50 Epochs it seems 20 hidden nodes are better as train (5.84 %) and test error (10.39 %) and subsequently the overfitting was lowest (4.55%). Another observation I made was, the more hidden nodes were chosen the slower the processing of results.



5). Using a WTA error method on a 22-10-5 Architure tests were made with learning rate (lr) = 0.2, 0.5, 1.0, 2.0, 3.0, 6.0. For comparison reason I left the epochs in all 6 runs at 300. The results shows that the train error with each lr increase rose (from 1.78 (at lr 0.2) to 12.94 (at lr 6.0). On the test error it was a bit of an up and down (14.29 (at lr 0.2), 12.99 (at lr 0.5), 11.69 % at 2.0/3.0. While at all times the the overfitting was significant the tendency was that each step up of learning rate the overfitting would reduce (starting at 12.51 and in the last run end up with 5.24). For test error it seems in general the higher you go with the learning rate the higher the error gets. Another observation is that the MSE rises and falls significantly per epoch the higher the learning rate gets. See left picture.



6. For this tasks I sticked with WTA and an amount of 300 Epochs. I only changed the learning rate and momentum values. I executed 6 runs, one set with learning rate 0.2 (default) and momentum = 0.5, 0.75 and 1.0; the same again with learning rate = 1.0. The observation I made was that the higher the learning rate and the higher the momentum the higher the error but also the closer train and test error get (lr = 0.2/m=0.5 result in 5.58 % train error and test 19.48 %) and the lesser the overfitting (lr = 1/m=1 result in 24.37 % train error and test 24.68 %). Similar to task 5 the MSE rises and drops significantly between epochs - resulting in the right picture.

7. Overfitting is similar to run number 5 of Task 1. The overfitting is significant with 13% difference between train (1.27%) and test (14.29%) error using WTA error method. I took around 800 epochs to get to the point where the MSE didn't seem changing anymore.

8. In comparison JNNS seems more accurate in absolute values as train and test accuracy are both higher than J48 and the Multilayer-Perceptron, but while all algorithms have a significant overfitting of more than 10% JNNS has the highest overfitting value (14.6).

Run No	Architecture	Parameters	Epochs/CV-Folds	Train Accuracy	Test Accuracy	Difference	Overfitting
JNNS	22-10-5	lr = 0.2	200	97.72 %	83.12 %	14.6 %	significant
Weka 1	J48	C = 0.25, M= 4	10	82.03 %	70.51 %	11.52 %	significant
Weka 2	MultilayerPerceptron	default	10	90.63 %	79.49 %	11.14 %	significant

PART 2: NUMERIC PREDICTION WITH NEURAL NETWORKS

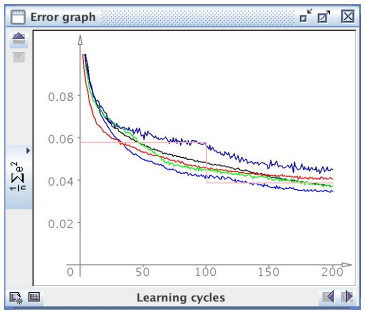
1.) There will be 26 input nodes - for each possible value of the attributes one node (except for binary attributes like “sex” which can be used in one node using 1 = male and 0 = female, opposed to using 2 nodes to keep the amount of nodes lower performance reasons. Nodes are as following: 'sex' = 1, 'cp' = 4, 'trestbps' = 1, 'chol' = 1, 'fbs' =1, 'restecg' = 3, 'thalach' = 1, 'exang' = 1, 'oldpeak = 1, 'slope = 3, 'ca' = 1 'thal' = 3, 'num' = 5 that makes 1 + 4 + 1 + 1 + 1 + 3 + 1+ 1+ 1 + 3 +1 +3 + 5 = 26 Nodes. There will be only 1 output node for attribute/node ‘age’.

2. Please see Appendix - Script for PART 2. I used the normalised arff data from part 1 and exported it to csv. I used the csv to shift the first row with the age data to the end of the data sets, since it’s the searched value/predicted value for this tests. Then I copied the data back into the arff file. As a third task I used the previous script heart_jnns.sh as basis and modified it for the requirements of this task and saved it as heart_np.sh. Like in part one I let it create 3 files from 591 data sets. 67% (394 sets) training data, 20 % validation data (118 sets) and 13 % test data (77 sets). Since it bases on the previous script the mapping of nominal attributes into the N-width binary string where taken over from there and the arff file didn’t contain incomplete data sets.

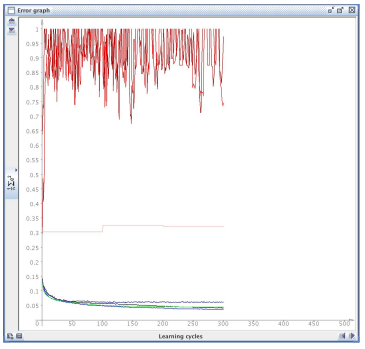
3. For this task I used a 26-5-1 architecture and a learning rate of 0.2 and gradually increasing epochs from 50 to 500. I tried using WTA and 402040 error methods but the amount of unknown error was very high. I used error method ‘Band’ in the end, which is said be useful for networks with only one output node. But I will base my analysis on the network error (TSS) and MSE as it seems more reliable (it’s always the same between the different methods). Comparing the values it seems like in Part 1 the more epochs are done the lower the error. The difference between train and test data seems less significant in terms of overfitting than part 1 task 1 but is still there.

Run No	Architecture	Parameters	Epochs	Train TSS	Train MSE	Train Error	Test TSS	Test MSE	Test Error
1 (Band)	26-5-1	lr=0.2	50	8.034501	0.020392	53.05 %	2.260705	0.029360	61.04 %
2 (Band)	26-5-1	lr=0.2	100	7.254376	0.018412	51.52 %	2.149065	0.027910	61.04 %
3 (Band)	26-5-1	lr=0.2	150	6.869882	0.017436	51.02 %	1.990913	0.025856	59.74 %
4 (Band)	26-5-1	lr=0.2	200	6.486104	0.016462	50.00 %	1.848997	0.024013	58.44 %
5 (Band)	26-5-1	lr=0.2	500	5.388981	0.013678	42.89 %	1.533979	0.019922	46.75 %

4) Like in part 2 I choose to do 10 runs with two different epochs (50 & 500) on 6 different architectures consisting of each of 3, 5, 10, 15, 20, 30 hidden notes but always 26 input and 1 output node. Using ‘Band’ I captured the train and test error. Generally it shows again that the more epochs the lesser the error. Like Part 1 the more hidden nodes are added the lower the processing speed. At 50 epochs the lowest train TSS/MSE is with 10 (TSS=7.834, MSE=0.0199) and 20 (TSS=7.294, MSE=0.0185) hidden nodes. While for test 10 hidden nodes seem the best as TSS is the lowest there (TSS=1.960662). Converting it to MSE (MSE=0.0255) and making a difference between train and test shows that 10 hidden nodes have the lowest difference/overfitting (0.0056). At 500 Epochs it seems that for training data between 20 (TSS=3.520408, MSE=0.0089) and 30 (TSS=3.6483/MSE=0.0093) hidden nodes is a optimum while for test data at 10 hidden nodes seem be the lowest point (TSS =1.4243, MSE =0.0185) . Curiously when it comes to the lowest difference between train and test MSE with 500 epochs the lowest difference of 0.0062 is with 5 hidden nodes but 10 hidden nodes compute a acceptable value of 0.0073 which points to a lower degree of overfitting.



5) Using ‘Band’ Error method on a 26-5-1 architure tests were made with lr = 0.2, 0.5, 1.0, 2.0, 3.0, 6.0. For comparison reason I left the epochs in all 6 runs at 200. The results shows that the train and test network error is until lr = 3.0 relative stable and doesn’t fluctuate as much as it did in Part A. It seems to have an optimum at lr = 3.0 where the error for train (TSS = 5.525268, MSE = 0.0140) and test is lowest (TSS = 1.642898 MSE = 0.0213). Once the lr is over 3.0 the MSE fluctuates more even though at lr = 6.0 (see left picture) is has the lowest difference/overfitting between test and train MSE of all the runs with 0.0050.



6. For this tasks I sticked with ‘Band’ and a amount of 300 Epochs. Like Part A - 6 runs, one set with learning rate 0.2 (default) and momentum = 0.5, 0.75 and 1.0; the same again with learning rate = 1.0. The observation I made was once the momentum passes 0.75 the error increases and fluctuates per each epoch significantly (see right picture). Otherwise with increasing learning rate increases the TSS and MSE (e.g lr = 0.2, m=0.5: TSS 5.473 and lr = 1.0, m=0.5: TSS = 6.065).

7. For this run using the same 26-5-1 Architecture and lr = 0.2 it took 1400 epochs to reach the point in which the MSE didn’t seem changing anymore. The observation is that despite both train and test MSE (0.0123) and TSS (0.0295) being lower than in task 1 the difference between those two is significantly higher (0.0172) - more than twice much as in run 5 in task 1 (MSE Train/Difference = 0.0062).

8. In comparison JNNS seems more accurate in absolute values as train and test accuracy are both higher than M5P and the Multilayer-Perceptron. Also overfitting the NNS is better as the difference between train and test accuracy very low with only 0.0090.

Run No	Architecture	Parameters	Epochs/ CV-Folds	Train Accuracy	Test Accuracy	Difference
NNS 1	26-5-1	lr = 0.2	50	0.9796	0.9706	0.0090
Weka 1	M5P	M= 4	10	0.8516	0.8144	0.0372
Weka 2	MultilayerPerceptron	default	10	0.8148	0.6251	0.1897

PART 3: DATA MINING

I decided to analyse the data of the Portuguese banking institution.
First step was to transform the data into an arff format. For this joined the input variables of file bank-names.txt with the data of the bank-full.csv. Then I transformed the input variables to be attributes in arff style and prepared the csv data using a modified version of the script of part 2 to clear the data to make it usable for arff. It also including shuffling the data.

Looking at the bank-names.txt the aim seems to be to get people to subscribe to term deposits. For that reason several telemarketing campaigns were done which collected several meta data information. The question seems to be to find out any patterns within the data collected to more effectively target specific groups of the general population to improve the success of the marketing campaign.

To get to this point I will try to understand if there are any meaningful classifications of groups and cross check them with some clustering algorithms. I hope that Visualisations will assist in understanding those groups. I expect with this to see if there any specific patterns on subscribing. If these steps don't give already ideas about data using attribute selection will be applied to narrow down the attributes and hopefully gaining better insights to the data.

Thinking about the design of the data, association finding might be less useful to generate insightful knowledge, because the data given are not several products that can be associated to a group by certain transactions. This is a marketing campaign with one outcome the product term deposit got successfully sold or not. Association finding are more useful for data like from shopping bags or people watching movies as it can show associations between similar behaviour types and generate recommendation for people with similar purchasing behaviours.

Step 1. ZeroR

At first ZeroR is applied onto the data to establish a basic measure on overfitting and classification error and accuracy as all other chosen classifiers will have to beat results which are generated by Zero R.

Run No	Classifier	Parameters	Training Error	X-valid Error	Overfitting
1	ZeroR	default	11.6985 %	11.6985 %	None
2	IBk	default	0%	13.1649 %	Significant
3	J48	default	5.8769 %	9.9532 %	Some

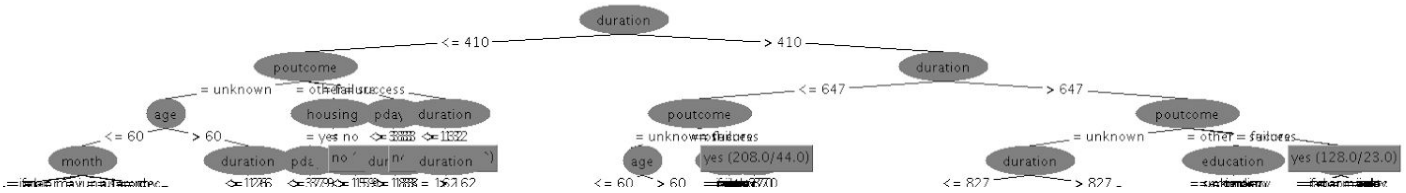
Step 2. IBK

Trying a nearest neighbor classifiers in this case IBk shows a significant amount of cross validation error and thus overfitting. Thats means this type of classifier will be neglected.

Step 3. J48

Trying a decision tree classifier such as J48 shows better results in accuracy for both training and validation data. There is still some degree of overfitting but around 4% which should be tolerable. This means I will continue using this classifier and analyze the data with it.

After letting the system do another cross validation test with a minimum number of objects of 3, which resulted in a even lower error results, a visualised decision tree should give a first clue about which data is important.

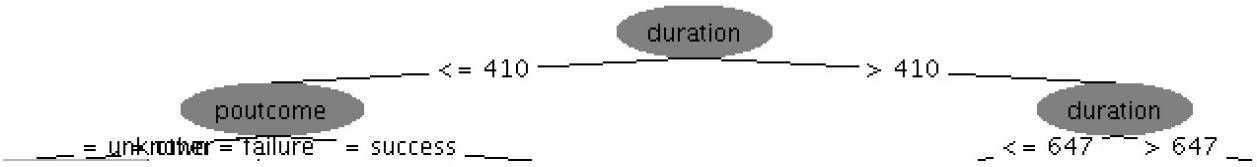


The first conclusion that I draw from this visualised decision tree is that there too much attributes - an attribute selection might be necessary later on. Apart from this looking at the top three levels of tree the duration of the last call, the outcome of the previous marketing campaign, age, housing and the days that passed since last contact seem to have most weight on the outcomes.

Concluding from this I state two hypophysis. 1. The longer the marketing call lasts the higher the likelihood that it will be successful as it most probably indicates that a person is interested.
Second hypophysis - if the outcome has been successful before the likelihood is high that it will happen again as the person might have encountered a positive experience.

Step 4. Manual Attribute Selection

Based the conclusions from on the analysis from Step 3 I will reduce the attributes down to only the attributes: duration, poutcome, age, housing, pdays, y to validate if my hypophysis can be confirmed. I will again use the J48 decision tree to visualise the outcome.

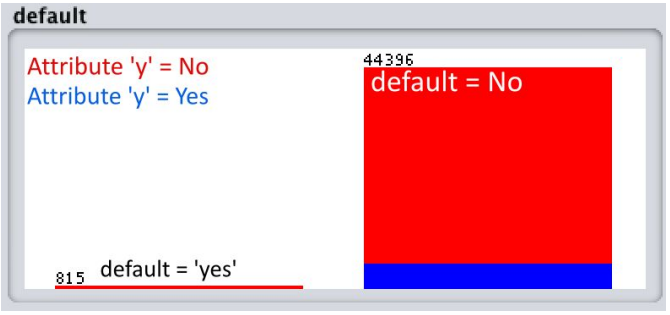


Looking through the data I can confirm my first hypophysis - the tendency is that the longer the duration of the call last the more likely a positive outcome. While the "magical" number seems to be 410 seconds or 6.8 minutes - 69% of successful calls are longer than 6 minutes. The problem is we don't know what has been talked about in this calls. But still the golden nugget I see here is the longer someone can hold the conversation the higher the chances of success. For a next marketing campaign I would recommend based on this to capture the content of the calls (there might be hints to find on what exactly convinces people e.g. nice chat, humorful approaches, serious professional talk).

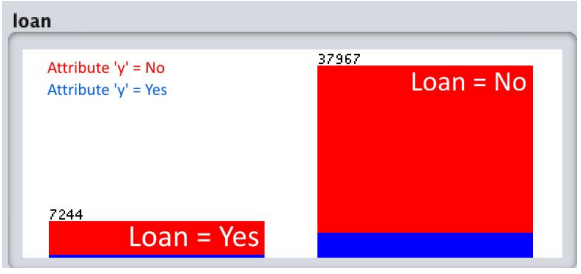
My second hypothesis I couldn't confirm yet as the majority of the cases it is not know if a previous campaign was successful or not, thus making difficult to make an statement about it. To solve this I will create a arff file with a subset of data excluding the 'unknown' cases of attribute poutcome because we can't use that information. The result of the analysis confirms the second hypophysis. The majority of previous unsuccessful calls turn mostly turn into failures again (92. 4% fail rate) and only 7.6% turn rate. While on the other hand almost 80% previous successfully called people could be convinced to subscribe to term deposit again.
The next golden nugget I see here is that it doesn't seem useful to pursue people that haven't been convinced in previous campaigns. The effort and cost is not worth it with a turn rate of less than 8 % and for that matter also pervious outcomes that lead in "others" there the turn rate is even less with 1.8%. I would recommend to better drop the contacts from the contact list with the next campaign which turned either into a failure or "others" and save the cost/effort on those call.

Step 5. Visualisation

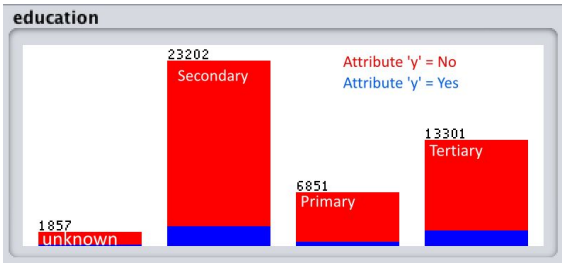
Checking on the visualisation of the distribution of the attribute 'y' which stands for if the call was successfull or failed to convince the person to subscribe to a term deposit. Going through the various visualization I will check if there are more useful information to be found or even golden nuggets.



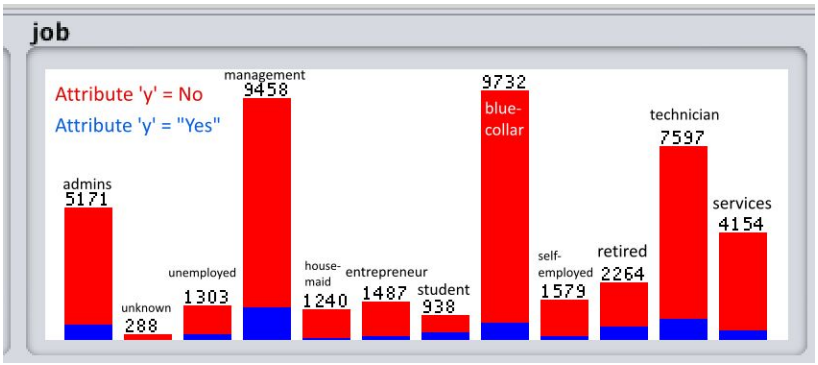
Three visualisation stick out in their distribution.
First is the default attribute - it means if the person has credit in default. Even though it's just a few. All persons that are having such a credit in default are not being convinced to subscribe. So the nugget here is again to exclude people with credits on default from the contact list as none of them want term deposit.



The next is obvious visualisation are loans, the majority of people that are loan free take the risk of investing in a term deposit. That makes sense as people with loan might not have the necessary money available to make such a deposit. A future marketing campaign should focus more on loan free people.



The last visualisation I think could help for designing a future marketing campaign is the education, because it shows that both higher educated people and people with secondary level education are more likely to subscribe to the term deposits in comparison to primary educated people. So the knowledge here is use this information to direct the marketing efforts maybe to people in positions where people need higher education or buy data sets from Linkedin or SEEK where usually the education is stated. This is confirmed with another visual where the jobs that need higher education (manager, technicians, admins and students) are subscribing more often than with supposedly lower education such as service personnel, housemaids or unemployed people. But this statement is to be seen carefully as often personal financial situation plays a role e.g. Entrepreneurs or self employed people can be higher educated but seem to be less interested in term deposits probably due to their financial situation and needing the money for their business.



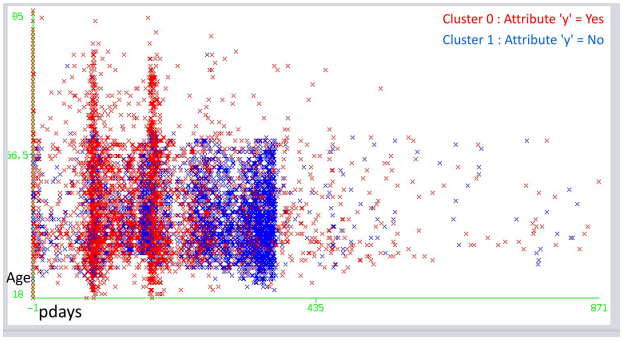
To confirm that I would need to analyse the relation between deposit, job and outcome (attribute 'y'). For this I will make two subsets one with balance in negative and one in positive. It shows here as well that people with negative balance are less likely to suscribe than people with positive balance. 7% of the people with a negative balance would go for the term deposit while 13% people with positive balance would do it too. Comparing the amount the jobs against negative balance or positive balance doesn't show any significance in difference to previous visual. In all jobs it seem having a negative balance makes it less likely to agree to a term deposit except the entrepreneurs there actually the 9% of the entrepreneurs agreed with a negative balance to a term deposit while 8% with positive balance did it as well.

Step 6. Algorithmic Attribute Selection

After analysing and digging deeper into the data and selecting data/attributes by myself using decision trees and following the leads mixing a bit of domain knowledge as a next step will use the system supported attribute selection to see if more golden nuggets can be found.
For this I used Attribute Evaluator: CfsSubsetEval and BestFirst Search Method on default settings. The result is to reduce the attributes down to marital, housing, loan, duration, poutcome when using attribute 'y' as class.

Run No	Classifier	Parameters	Training Error	X-valid Error	Overfitting
Without attribute selection	J48	default	5.8769 %	9.9532 %	Some
With attribute selection	J48	default	9.7277 %	10.0595 %	none

Using the attributes did actually lower the amount of overfitting (which was relatively low before) even more but on the expense of some accuracy when running J48 again. Building the tree shows a similar results as without the attribute selection. Duration and poutcome are like before the most upper defining attributes. Looking at housing and marital status shows that people without housing loan and single or married people are more likely go for the term deposit than divorced people or people with a housing loan. Since the difference isn't significant I won't see it as a nugget, especially since in step 4 & 5 I figured already out that people with loans, or negative balance are more hesitant on term deposits. So this finding supports the underlying theory of people in "difficult" or "indebted" financial situations are less likely to subscribe to the term deposit probably because they don't have the means to do it.



Step 7. Clustering

After having done some classification I wanted to see if classification algorithms can bring some more insights into the data as well. I am aware it is better to use classifications on unlabelled data in which groups are not known and the algorithm shall try finding groups.
I did one round with all attributes using Kmeans on default and going through the visualizations pdays and age show an interesting pattern. According to this plot there are two clusters for yes on attribute 'y' that lay around the area of 90 days and 180 days since last contact. While another cluster with attribute 'y' = no starts at around 250 days and goes until 300 days after then the data gets less condens and more scattered. The nuggets I see here is that across all ages it seems it's highly likely to be successful if the person contacted is after either around 3 or around 6 month, after that time it seems too much time passes and people are less likely to subscribe to the term deposit. I would recommend the marketing personnel to call again every 3 month to either check on the person if they an have interest on a term deposit (maybe again).

Conclusion/Golding Nuggets

Following my journey of data mining bit of a knowledge discovery I found a few golden nuggets of "different size" so to say that can be applied to the next marketing campaign to either cut cost/improve effectiveness, further shape the campaign towards a profitable target market or generally help the marketing efforts. Here a list of nuggets I found:

1. People in financial "difficult" or "indebted" situations like having a negative balance, a credit on default or a loan are much less likely to subscribe to a term deposit. Advice: People in that situation can be neglected and shouldn't be the focus of the marketing campaign.
2. Marketing calls longer than 6.8 minutes are more likely to succeed. Advice: Try figuring out what is being discussed over the course of those, successful 6 minutes and more, calls and try to figure out if those topics/strategies can be applied generally.
3. Higher educated people and people in jobs that likely require higher education (secondary and tertiary) are more likely to be convinced into a term deposit than lower levels of education and jobs. Advice: Focus the marketing efforts on people with higher education.
4. If a previous campaign had a negative or 'other' outcome with a person, the person is much less likely to subscribe. Advice: Drop the people from the list where it's known that they had a negative/other outcome from a previous campaign.
5. Follow up calls are more successful after 3 or 6 month and much less successful after 250 days. Advice here is to check on the customer every 3 or 6 month.

Consideration to be taken into account is that this nuggets reflect state of the information at that time and for the future might need to be adjusted to trends and events. E.g. this data is from 2012 from Portugal - at this time portugal was suffering from harsh economic austerity politics imposed by European Union due to the financial / european crisis. This is an 'unforeseen event' that certainly lowered confidence in financial institution. With recovering economical situation in Portugal confidence in both banks and their future might return and shift the results of the findings extracted from this data. Therefore at least a yearly revision should be done to verify the validity of the data/nuggets..

APPENDIX - SCRIPT FOR PART 2

```
#!/bin/bash
TRAIN=394
VALID=118
TEST=77

tail -n +19 ./heart-np.arff |\
  fgrep -v "%" | gshuf |
  sed -e"s/,/ /g" > temp2.txt
# The training file
/bin/echo "SNNS pattern definition file V3.2"  >heart-train_np.pat
/bin/echo "generated at Sun Oct 01 15:00:00 2017"  >>heart-train_np.pat
/bin/echo ""  >>heart-train_np.pat
/bin/echo ""  >>heart-train_np.pat
/bin/echo "No. of patterns : $TRAIN" >>heart-train_np.pat
/bin/echo "No. of input units : 26" >>heart-train_np.pat
/bin/echo "No. of output units : 1" >>heart-train_np.pat

head -$TRAIN temp2.txt |
#attribute cp
  sed -e"s/atyp_angina/1 0 0 0/g" |
  sed -e"s/asympt/0 1 0 0/g" |
  sed -e"s/non_anginal/0 0 1 0/g" |
  sed -e"s/typ_angina/0 0 0 1/g" |
#attribute thal
  sed -e"s/fixed_defect/1 0 0/g" |
#sed -e"s/normal/0 1 0/g" |
  sed -e"s/reversable_defect/0 0 1/g" |
#attribute restecg
  sed -e"s/left_vent_hyper/1 0 0/g" |
  sed -e"s/st_t_wave_abnormality/0 0 1/g" |
  sed -e"s/normal/0 1 0/g" |
#attribute num
  sed -e"s/<50/1 0 0 0 0/g" |
  sed -e"s/>50_1/0 1 0 0 0/g" |
  sed -e"s/>50_2/0 0 1 0 0/g" |
  sed -e"s/>50_3/0 0 0 1 0/g" |
  sed -e"s/>50_3/0 0 0 0 1/g" |
#attribute slope
  sed -e"s/up/1 0 0/g" |
  sed -e"s/flat/0 1 0/g" |
  sed -e"s/down/0 0 1/g" |
#Attribute age
  sed -e"s/female/0/g" |
  sed -e"s/male/1/g"  |
#exang {no,yes}
  sed -e"s/no/0/g" |
  sed -e"s/yes/1/g" |
#attribute fbs
  sed -e"s/t/0/g" |
  sed -e"s/f/1/g" >>heart-train_np.pat

# The validation file
/bin/echo "SNNS pattern definition file V3.2"  >heart-valid_np.pat
/bin/echo "generated at Sun Oct 01 15:00:00 2017"  >>heart-valid_np.pat
/bin/echo ""  >>heart-valid_np.pat
/bin/echo ""  >>heart-valid_np.pat
/bin/echo "No. of patterns : $VALID" >>heart-valid_np.pat
/bin/echo "No. of input units : 26"  >>heart-valid_np.pat
/bin/echo "No. of output units : 1"  >>heart-valid_np.pat
FROM=`expr $TRAIN + 1`
tail -n +$FROM temp2.txt | head -$VALID |
#attribute cp
sed -e"s/atyp_angina/1 0 0 0/g" |
sed -e"s/asympt/0 1 0 0/g" |
sed -e"s/non_anginal/0 0 1 0/g" |
sed -e"s/typ_angina/0 0 0 1/g" |
#attribute thal
sed -e"s/fixed_defect/1 0 0/g" |
#sed -e"s/normal/0 1 0/g" |
sed -e"s/reversable_defect/0 0 1/g" |
#attribute restecg
```

```
sed -e"s/left_vent_hyper/1 0 0/g" |
sed -e"s/st_t_wave_abnormality/0 0 1/g" |
sed -e"s/normal/0 1 0/g" |
#attribute num
sed -e"s/<50/1 0 0 0 0/g" |
sed -e"s/>50_1/0 1 0 0 0/g" |
sed -e"s/>50_2/0 0 1 0 0/g" |
sed -e"s/>50_3/0 0 0 1 0/g" |
sed -e"s/>50_3/0 0 0 0 1/g" |
#attribute slope
sed -e"s/up/1 0 0/g" |
sed -e"s/flat/0 1 0/g" |
sed -e"s/down/0 0 1/g" |
#Attribute age
sed -e"s/female/0/g" |
sed -e"s/male/1/g"  |
#exang {no,yes}
sed -e"s/no/0/g" |
sed -e"s/yes/1/g" |
#attribute fbs
sed -e"s/t/0/g" |
sed -e"s/f/1/g" >>heart-valid_np.pat

# The test file
/bin/echo "SNNS pattern definition file V3.2"  >heart-test_np.pat
/bin/echo "generated at Sun Oct 01 15:00:00 2017"  >>heart-test_np.pat
/bin/echo ""  >>heart-test_np.pat
/bin/echo ""  >>heart-test_np.pat
/bin/echo "No. of patterns : $TEST"  >>heart-test_np.pat
/bin/echo "No. of input units : 26"  >>heart-test_np.pat
/bin/echo "No. of output units : 1"  >>heart-test_np.pat
FROM=`expr $FROM + $VALID`
tail -n +$FROM temp2.txt | head -$TEST |
#attribute cp
sed -e"s/atyp_angina/1 0 0 0/g" |
sed -e"s/asympt/0 1 0 0/g" |
sed -e"s/non_anginal/0 0 1 0/g" |
sed -e"s/typ_angina/0 0 0 1/g" |
#attribute thal
sed -e"s/fixed_defect/1 0 0/g" |
#sed -e"s/normal/0 1 0/g" |
sed -e"s/reversable_defect/0 0 1/g" |
#attribute restecg
sed -e"s/left_vent_hyper/1 0 0/g" |
sed -e"s/st_t_wave_abnormality/0 0 1/g" |
sed -e"s/normal/0 1 0/g" |
#attribute num
sed -e"s/<50/1 0 0 0 0/g" |
sed -e"s/>50_1/0 1 0 0 0/g" |
sed -e"s/>50_2/0 0 1 0 0/g" |
sed -e"s/>50_3/0 0 0 1 0/g" |
sed -e"s/>50_3/0 0 0 0 1/g" |
#attribute slope
sed -e"s/up/1 0 0/g" |
sed -e"s/flat/0 1 0/g" |
sed -e"s/down/0 0 1/g" |
#Attribute age
sed -e"s/female/0/g" |
sed -e"s/male/1/g"  |
#exang {no,yes}
sed -e"s/no/0/g" |
sed -e"s/yes/1/g" |
#attribute fbs
sed -e"s/t/0/g" |
sed -e"s/f/1/g" >>heart-test_np.pat
```