Chapter 4

Case Studies

In this chapter we examine two case studies to demonstrate the techniques and methodologies described in the previous chapter. The first case study is an example of a classification problem, and involves learning to classify loan applicants as good or bad credit risks. The second case study is an example of a prediction problem, and involves learning to predict the daily exchange rate of the Australian dollar against the US dollar. Both case studies are constructed from the initial raw data sets, showing the required preprocessing and decisions relating to architecture and parameters. The final results are presented and evaluated.

4.1 CLASSIFICATION: LOAN APPLICANT EVALUATION

Neural networks have been used successfully in the banking sector for almost a decade. In this case study we consider a common application of neural networks: learning to distinguish loan applicants who are likely to repay their debt from those who are likely to default on their loan.

4.1.1 The Problem

Suppose we have 1000 previous applications for credit at a bank, together with those applicants' subsequent classification as 'good' or 'bad' customers. Can we use this information to learn to successfully classify new applicants as 'good' or 'bad' based only on their applications? In other words, can we learn which combinations of responses indicate a 'good' credit risk and which indicate a 'bad' credit risk? Neural networks are an ideal tool for solving this problem, since they have the ability to learn the features of the data, while ensuring the relationships discovered will generalise to new data. Several studies have reported neural network applications within the credit industry including prediction of corporate bankruptcy (Altman et al., 1994), and credit card approvals (Carter and Catlett, 1987).

4.1.2 The Data

The data has been obtained from Professor Bob Henery, Department of Statistics and Modelling Science, University of Strathclyde, Glasgow, U.K., and was originally from a German bank. The questions used in the application form for a loan at the bank can be found in Appendix C. The application form consists of 20 questions relating to personal and financial details of the applicant. Some of the questions require a written answer, while others require selection from a set of predefined categories. For example, question 1 requires the applicant to provide the status of their checking account by selecting one of four options: i) in debt, ii) between zero and two hundred Deutchmarks, iii) more than two hundred Deutchmarks, or iv) no checking account. Question 2, however, asks the applicant to specify the number of months that their account has existed.

The original data file contains the responses to these questions, together with the observed (long term) classification of each applicant as a good or bad credit risk. There are 1000 rows in the data file, comprising 700 known good applicants and 300 known bad risk applicants. This data has been preprocessed to replace categorical data (such as the responses to multiple choice questions) with numerical data. Neural networks need to receive numerical data as inputs, and the data needs to contain a sense of scale and order. By this is meant that a neural network will treat inputs of 1 or 2 as being similar to each other but different from say 9 or 10. It will also interpret an input of 2 as being closer to an input of 3 than an input of 1 is. For most responses to the questionnaire the required ordering will naturally emerge as a result of the multiple choice options or the nature of the question (eg. age), but for some data we will need to manipulate the data so that it can be used by the neural network.

The original data contains 20 columns or variables: 7 numerical data columns (eg. age or credit amount) and 13 categorical data columns (eg. loan purpose or job status). The categorical data needs to be converted to numerical data which is usually achieved by translating a single column categorical data into multiple columns with binary variables. Suppose we have a variable which has 4 possible responses, for example colour which could be red, blue, yellow or green. There is no natural order to the responses when we consider the relevance of the chosen input to our decision. It would be inappropriate to assign the numbers 1, 2, 3 and 4 to the colour respectively, since colour is not an ordered quantity like age or income. Instead we could replace the single variable colour with 4 new variables so that red becomes represented as (1 0 0 0), blue as (0 1 0 0), yellow as (0 0 1 0) and green as (0 0 0 1). Thus if there are N possible values for a variable which does not possess an inherent order, the variable can be replaced with N variables each containing either a zero or a one.

Using this technique (called *1-out-of-N encoding*):

- question 4 (credit purpose) becomes
  - 1 0 (new car), 0 1 (old car) and 0 0 (other)

- question 10 (other debtors/guarantors) becomes
  - 1 0 (none), 0 1 (coapplicant), 0 0 (guarantor);
- question 15 (Housing type) becomes
  - 1 0 (renting), 0 1 (owner), 0 0 (free housing); and
- question 17 (employment type) becomes
  - 1 0 0 (unemployed), 0 1 0 (unskilled), 0 0 1 (skilled), 0 0 0 (highly skilled).

Thus the original 20 variables or columns are replaced with 24 columns, and all of the data can be treated as numerical inputs to the neural network. The known classification of each customer in the data set as 'good' or 'bad' can be coded as either: credit class 1 or 2 (using a single column) or using two columns to encode good (1 0) or bad (0 1). An example showing the transformation of the first 5 rows of the data after preprocessing is shown in Table 4.1.

Table 4.1 Original data and preprocessed data showing 1-out-of-N encoding

SAMPLE RESPONSES TO ORIGINAL QUESTIONS

A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1 A192 A201 1
A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1 A191 A201 2
A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2 A191 A201 1
A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2 A191 A201 1
A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2 A191 A201 2

SAMPLE RESPONSES AFTER PRE-PROCESSING OF DATA

Q1	Q2	Q3	Q5	Q6	Q7	Q9	Q11	Q12	Q13	Q14	Q16	Q18	Q19	Q20	Q4	Q10	Q15	Q17	Class
1	6	4	12	5	5	3	4	1	67	3	2	1	2	1	0 0	1 0	0 1	0 0 1	1
2	48	2	60	1	3	2	2	1	22	3	1	1	1	1	0 0	1 0	0 1	0 0 1	2
4	12	4	21	1	4	3	3	1	49	3	1	2	1	1	0 0	1 0	0 1	0 1 0	1
1	42	2	79	1	4	3	4	2	45	3	1	2	1	1	0 0	0 0	0 0	0 0 1	1
1	24	3	49	1	3	3	4	4	53	3	2	2	1	1	1 0	1 0	0 0	0 0 1	2

4.1.3 Results

We now have a choice of trying to classify a loan applicant according to their class number (1 for good or 2 for bad) using a single output neuron, or using two output neurons and training the neural network to classify good applicants as 1 0 and bad applicants as 0 1. Generally, classification problems are better suited to neural network architectures with as many output neurons as there are classes, although we will try both approaches here.

As an initial experiment, we choose all 24 variables from the preprocessed data file as inputs, and construct a MFNN architecture consisting of 24 inputs, 30 hidden neurons and 2 output neurons (plus extra inputs for the thresholding of the neurons). 20% of the data was randomly extracted to form the test set. The learning rate and momentum rate were fixed at $c = 0.1$ and $\alpha = 0.1$ while the initial weights were

small random numbers around 0.3. The order in which input patterns were presented to the MFNN was random, and the network performance was measured on the test set every 200 epochs. If the test set error had not improved within 20,000 epochs, then training was terminated to prevent memorisation of the training data. Using this approach, and evaluating the performance across the entire data set, the neural network was able to correctly classify 160 of the 300 known bad applicants, and 636 of the 700 known good applicants. These decisions have been made by choosing the maximum firing neuron, ie. if neuron 1 fires more than neuron 2 for a given input pattern, then the input pattern (applicant) is determined to belong to class 1 (good applicants), and vice versa. The results are summarised in Table 4.2 showing the accuracy levels of the decisions.

Table 4.2 Classification accuracy using MFNN with 24-30-2 architecture

	classified good	classified bad	row accuracy
actually good	636	64	90.9%
actually bad	160	140	46.7%
column accuracy	79.9%	68.6%	

Thus the trained neural network can be expected to accurately classify good applicants 90.9% of the time, but only recognise a bad applicant 46.7% of the time. Obviously it is much more difficult for the neural network to learn to recognise the characteristics of bad applicants, especially since it is not exposed to as many examples of bad applicants compared to good ones. The neural network is relunctant then to classify an applicant as bad unless it is very confident. When the neural network classifies an applicant as bad, it is correct 68.6% of the time, and correct 79.9% of the time when it classifies an applicant as good. For the bank to minimise its exposure to risk, it is more important to correctly detect bad credit risks than good ones. If the neural network rejects a good applicant, the bank loses that customer's business, but if a bad applicant is accepted, then the bank stands to lose much more. Therefore, we need to experiment with the neural network architecture and parameters to try to make the number of bad applicants classified as good (160 in Table 4.2) as small as possible. This will likely mean a trade off in accuracy for classifying good applicants. It is easy to produce a small error in classifying bad applicants if we reject all applicants!

Looking at the contribution each variable is making to the decision making process (software like NeuroShell2 enables this to be seen in graphical form) reveals that some variables are extremely significant to the output of the network, while other have little effect. Variable numbers 1, 2, 3, 5, 7, 10, 15, 17, 18 and 19 are found to have the greatest effect. In a second experiment, we will use only these 10 variables for each input pattern rather than the set of 24 variables. All other parameters are set as before. The results for this experiment are shown in Table 4.3.

Table 4.3 Classification accuracy using MFNN with 10-30-2 architecture

	classified good	classified bad	row accuracy
actually good	603	97	86.1%
actually bad	148	152	50.7%
column accuracy	80.3%	61.0%	

The effect of removing unnecessary variables from the network has been to slightly improve the percentage of bad applicants that are detected (50.7% rather than 46.7% as before). But now that the network is less reluctant to classify applicants as bad, more good applicants are being misclassified and the accuracy level when the neural network classifies an applicant as bad has been reduced (61% rather than 68.6% as before). The answer to this trade-off is found through trial and error: experimenting with architectures, parameters, etc until an acceptable level of accuracy is found. This accuracy depends to a large extent upon the requirements of the business and the goals of the organisation.

As a third experiment, we have varied the number of hidden neurons from 30 to try 20 and 40 neurons, still with the reduced 10 inputs. The results are shown in Tables 4.4 and 4.5 for 20 and 40 hidden neurons respectively. Neither architecture appears to improve the performance of the network, and so we will revert to the original 30 hidden neurons for subsequent experiments.

Table 4.4 Classification accuracy using MFNN with 10-20-2 architecture

	classified good	classified bad	row accuracy
actually good	615	85	87.9%
actually bad	156	144	48.0%
column accuracy	79.8%	62.9%	

Table 4.5 Classification accuracy using MFNN with 10-40-2 architecture

	classified good	classified bad	row accuracy
actually good	625	75	89.3%
actually bad	164	136	45.3%
column accuracy	79.2%	64.5%	

Further experimentation with learning rates and momentum rates does not significantly improve the results, and the average error for each output neuron seems unable to reduce below around 0.2. We now alter the architecture to contain only a single output neuron which is attempting to classify each applicant as either a '0' or a '1' for good and bad respectively. Using this approach, similar results are again obtained, as shown in Table 4.6.

At this stage it is natural to wonder if any changes in architecture, inputs or parameters will improve the classification accuracies. Perhaps the performance is limited by how we have decided to interpret the neural network output? For the single output neuron architecture we have decided the applicant is likely to be

Table 4.6 Classification accuracy using MFNN with 10-30-1 architecture

	classified good	classified bad	row accuracy
actually good	630	70	90.0%
actually bad	150	150	50.0%
column accuracy	80.8%	68.2%	

bad if the output of the network is greater than 0.5 and good if the output is less than 0.5 (ie. we have rounded the output up or down to the nearest integer 0 or 1). What happens if we experiment with this decision threshold. If we state that we will classify applicants as bad if the output of the network is greater than 0.25, then the network is much more willing to classify applicants as bad, as shown in Table 4.7. Reducing the decision threshold even further to 0.1, as shown in Table 4.8 shows that only 19 of the 300 known bad applicants are incorrectly classified as good, although many of the good applicants are now incorrectly classified as bad.

Table 4.7 Classification accuracy using MFNN with 10-30-1 architecture and decision threshold of 0.25

	classified good	classified bad	row accuracy
actually good	460	240	65.7%
actually bad	54	246	82.0%
column accuracy	89.5%	50.6%	

Table 4.8 Classification accuracy using MFNN with 10-30-1 architecture and decision threshold of 0.1

	classified good	classified bad	row accuracy
actually good	233	467	33.3%
actually bad	19	281	93.7%
column accuracy	92.5%	37.6%	

Thus the interpretation of the continuous neural network output into a discrete decision of accept or reject also needs careful consideration, and can be selected to achieve the desired level of accuracy as required.

4.2 PREDICTION: FOREIGN EXCHANGE RATE PREDICTION

The second case study we examine relates to one of the largest markets in the world: the foreign exchange market. Here we use a neural network to model the movement of the exchange rate between the Australian dollar and the US Dollar. Many researchers have used neural networks and statistical approaches to model exchange rates (Flower et al., 1996; Hian, 1996; Green & Pearson, 1994; Steurer, 1995; Yao et al., 1997), and shown that neural networks frequently outperform more traditional approaches.

4.2.1 The Problem

In this case study we are interested in predicting the daily spot rate of the Australian dollar exchange rate against the US Dollar (AUD/USD). We will attempt to predict tomorrow's spot rate based upon today's and previous days' spot rates of the AUD/USD, together with some moving averages and commodity prices. Additional to predicting the spot rate, we will also use neural networks to learn to predict the movement of the exchange rate, ie. predicting only if the AUD/USD will increase or decrease, rather than attempting to predict the magnitude of the movement. These results can be used to assist in buying and selling strategies.

4.2.2 The Data

The data used in this study covers the period 1st January 1993 to 24th April 1998. The tail end of the data set thus includes the effect of the 1998 Asian economic crisis and explains the noticeable decrease in the exchange rate shown in Figure 4.1. There are 1377 daily data points available, and the maximum value is 81.79 while the minimum value is 63.26. Both the maximum and minimum values occur during the 1997-1998 portion of the data, thus we need to consider the test set extraction method carefully. In order that we can ensure good generalisation of the network performance, it is important that the test set and the training set be extracted from data which is statistically similar.

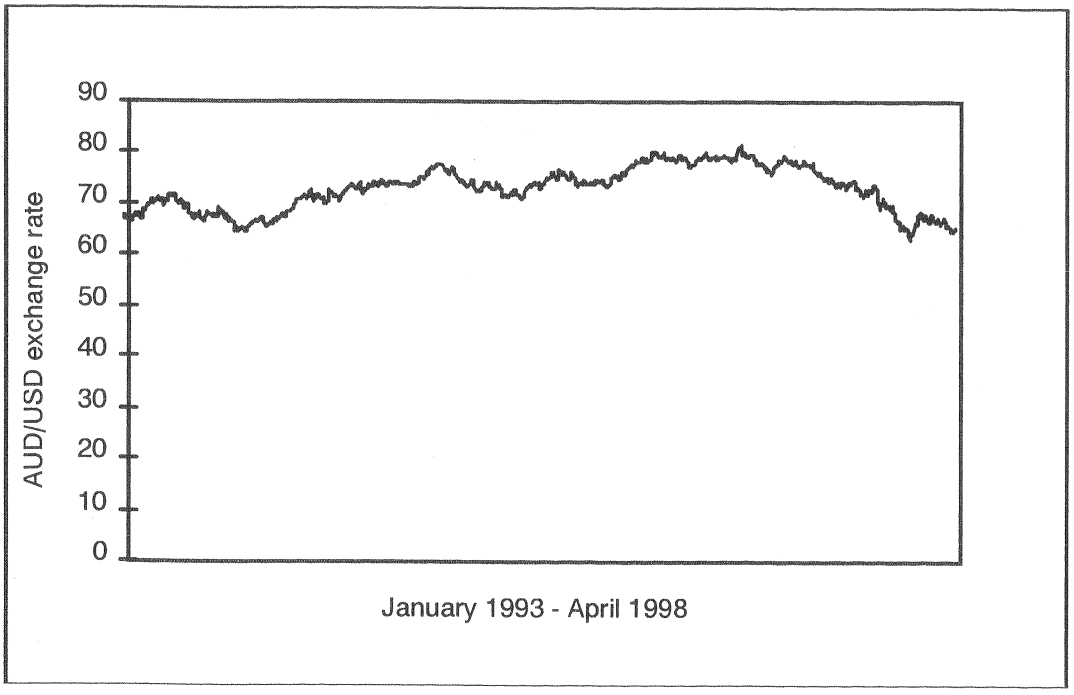

Figure 4.1 Daily spot rate of AUD/USD exchange rates

When considering what information we should give the neural network as inputs, it is important to think about what a human expert would require to make an informed decision, and to give the network that same information where possible. While it is difficult to provide the neural network with information about political issues and

other factors which we know have an impact on the world markets, we can certainly make the most of the data we have by preprocessing it. In order to predict tomorrow's spot rate, we should most certainly provide today's spot rate and the lagged spot rates from previous days. How far back we go depends upon the volatility of the market, and is determined mostly by experimentation. We should also provide some summarised information about recent behaviour of the exchange rate through the use of moving averages. Additional information such as the All Ordinaries Index, significant commodity prices, or the exchange rates of other nations, could also be provided to increase the amount of information to which the neural network is exposed.

In this case study we have chosen to use up to nine inputs to the neural network to predict tomorrow's AUD/USD spot rate:

1. Today's AUD/USD spot rate
2. 1 day lag of AUD/USD spot rate
3. 2 day lag of AUD/USD spot rate
4. 3 day lag of AUD/USD spot rate
5. 4 day lag of AUD/USD spot rate
6. 5 day moving average of daily AUD/USD spot rate
7. 10 day moving average of daily AUD/USD spot rate
8. Today's All Ordinaries Index
9. Today's Gold spot rate

4.2.3 Results

Several experiments were conducted to determine the architectures and parameters which achieved the best results. These experiments are summarised in Table 4.9 and show variations in inputs, test set extraction method, and the number of hidden neurons. The learning and momentum rates were fixed at 0.05 and 0.5 respectively. The number of hidden neurons was varied according to the formula used by NeuroShell2:

$$J = \tfrac{1}{2}(N+K)+\sqrt{P}$$

where P is the number of patterns in the training set, N is the number of input variables used, and K is the number of output neurons.

Each experiment was trained for 100 epochs. The results are shown in Table 4.10 where the performance measure used is the coefficient of determination R^2. This statistical indicator provides a measure of how accurate the predicted values are compared to the known values of tomorrow's AUD/USD spot rate across all of the examples used in the data sets. An R^2 of close to 1 indicates a near perfect degree of accuracy, while an R^2 around zero indicates that the results are no better than if we had used the average value as the basis for our prediction. In Table 4.10 we have measured the performance of the network on both the training and test data sets to provide an indication of the degree to which the network has successfully generalised its learning.

Table 4.9 Architecture and test set extraction method for AUD/USD prediction

	Input Variables	Test Set Extraction	Number of Hidden Neurons
experiment 1	1-3	20% at random	35
experiment 2	1-3	final 20%	35
experiment 3	1-3	1998 data only	37
experiment 4	1-7	20% at random	37
experiment 5	1-7	final 20%	37
experiment 6	1-7	1998 data only	39
experiment 7	1-9	20% at random	38
experiment 8	1-9	final 20%	38
experiment 9	1-9	1998 data only	40

Table 4.10 Performance of experiments on training and test sets

	R^2 for training set	R^2 for test set
experiment 1	0.9606	0.9614
experiment 2	0.9776	0.9830
experiment 3	0.7254	0.4531
experiment 4	0.9752	0.9762
experiment 5	0.9788	0.9836
experiment 6	0.7682	0.7126
experiment 7	0.9689	0.9730
experiment 8	0.9684	0.9810
experiment 9	0.8870	0.8523

The first observation from Table 4.10 is that experiments 3, 6, and 9 are not successful since the network performs much better on the training data than it does on the test data, indicating that the network has found features in the data to memorise rather than learning the underlying trends. The more information the network is exposed to the better the performance becomes, but the results are still very limiting. The test sets for these experiments only contain 1998 data. Since the network is not trained on any 1998 data, it has difficulty in accurately predicting the results for the test set. Better results are obtained from the remaining experiments, where the test set is larger and comprises data that is more statistically similar to the remaining training data. The experiments show that some improvement in performance is obtained by including more input variables, although the additional information about the All Ordinaries Index and the Gold spot price does not appear to have much impact.

Moving away from predicting the future spot rate now, perhaps a more useful resource for speculators are the movements of the future exchange rates. Post-processing the results of the spot rate prediction, so that the results of the neural

network were only interpreted as a method for predicting whether or not the spot rate will move up or down, a high level of accuracy can be obtained. Table 4.11 shows that the accuracy when predicting upwards or downwards movements alone is quite high regardless of the R^2 value obtained for the original spot rate prediction.

Table 4.11 Classification accuracy for movement prediction

	Number Incorrect	Percentage Correct
experiment 1	25	98.18%
experiment 2	24	98.25%
experiment 3	34	97.53%
experiment 4	20	98.54%
experiment 5	21	98.47%
experiment 6	30	97.82%
experiment 7	19	98.62%
experiment 8	18	98.69%
experiment 9	25	98.18%

Of course, an alternative (and perhaps better) approach would be to train a neural network to learn to classify the movement of the daily spot rate as up or down. This converts the problem into a classification rather than prediction problem, and it becomes similar to the first case study considered in this chapter.

CHAPTER SUMMARY

In this chapter we have examined two case studies: one using neural networks to solve a classification problem, and the other using neural networks for prediction. The first case study considered the task of classifying loan applicants as good or bad credit risks, so that a decision can be made as to whether they should be granted a loan. The data has come from responses to a series of questions on the application form. The neural network has the task of learning which types of applicants end up defaulting on their loans based upon 1000 examples of existing bank customers. The second case study has examined the popular prediction of exchange rates. We have used data from January 1993 until April 1998 to learn to predict the Australian dollar versus US Dollar daily spot rate. We have also shown how these results are also highly accurate when used to provide an indication of market movement. Both of these case studies have been used to demonstrate the methodology presented in Chapter 3.

Most neural network business applications will fall under the general categories of either classification or prediction, and hence MFNNs with backpropagation learning are an appropriate technique as demonstrated in this chapter. There are, however, situations where we do not have a set of training data, complete with known outputs. In such situations, supervised learning is not an option, and we need a more exploratory unsupervised method for learning characteristics of the data. Such techniques will be presented in the following chapters.

Appendix C

QUESTIONS FOR CREDIT APPLICATION EXAMPLE (GERMAN DATA)

QUESTION 1: (qualitative)

Status of existing checking account
A11 : ... < 0 DM
A12 : 0 <= ... < 200 DM
A13 : ... >= 200 DM
A14 : no checking account

QUESTION 2: (numerical)

Duration of checking account in months

QUESTION 3: (qualitative)

Credit history
A30 : no credits taken / all credits paid back duly
A31 : all credits at this bank paid back duly
A32 : existing credits paid back duly till now
A33 : delay in paying off in the past
A34 : critical account / other credits existing (not at this bank)

QUESTION 4: (qualitative)

Purpose
A40 : car (new)
A41 : car (used)
A42 : furniture/equipment
A43 : radio/television
A44 : domestic appliances
A45 : repairs
A46 : education
A47 : (vacation - does not exist?)
A48 : retraining
A49 : business
A410 : others

QUESTION 5: (numerical)

Credit amount

QUESTION 6: (qualitative)

Savings account/bonds
A61 : ... < 100 DM
A62 : 100 <= ... < 500 DM
A63 : 500 <= ... < 1000 DM
A64 : .. >= 1000 DM
A65 : unknown / no savings account

QUESTION 7: (qualitative)

Present employment since
A71 : unemployed
A72 : ... < 1 year
A73 : 1 <= ... < 4 years
A74 : 4 <= ... < 7 years
A75 : .. >= 7 years

QUESTION 8: (numerical)

Installment rate in percentage of disposable income

QUESTION 9: (qualitative)

Personal status and sex
A91 : male : divorced/separated
A92 : female : divorced/separated/married
A93 : male : single
A94 : male : married/widowed
A95 : female : single

QUESTION 10: (qualitative)

Other debtors / guarantors
A101 : none
A102 : co-applicant
A103 : guarantor

QUESTION 11: (numerical)

Number of years in current residence

QUESTION 12: (qualitative)

Property
A121 : real estate
A122 : if not A121 : building society savings agreement/life insurance
A123 : if not A121/A122 : car or other, not in QUESTION 6
A124 : unknown / no property

QUESTION 13: (numerical)

Age in years

QUESTION 14: (qualitative)

Other installment plans
A141 : bank
A142 : stores
A143 : none

QUESTION 15: (qualitative)

Housing
A151 : rent
A152 : own
A153 : for free

QUESTION 16: (numerical)

Number of existing credits at this bank

QUESTION 17: (qualitative)

Employment Status
A171 : unemployed/ unskilled - non-resident
A172 : unskilled - resident
A173 : skilled employee / official
A174 : management/ self-employed/
highly qualified employee/ officer

QUESTION 18: (numerical)

Number of people being liable to provide maintenance for

QUESTION 19: (qualitative)

Telephone
A191 : none
A192 : yes, registered under the customers name

QUESTION 20: (qualitative)

foreign worker
A201 : yes
A202 : no