**Universal Bank Data Set kNN analysis**

The Universal Bank dataset comprises 5,000 data points featuring information on customers with online accounts, personal loans, CCAVg, age, zipcode, family size, mortgage, and credit card details. The personal loan variable is categorical, denoted as 1 if the individual has a personal loan and 0 otherwise. The objective is to explore the feasibility of constructing a model to predict whether the bank should promote personal loan products based on the provided information.

To proceed with model development, certain assumptions about the dataset are necessary. We assume that when a new request is submitted to the bank, all relevant information about the individual, including account type, age, zip code, and other details, is available. Leveraging this information, we can input independent variables into the model to predict the likelihood of the individual having a personal loan (1) or not (0).

Before we could start with the kNN classification we did some analysis on the data and did a scatter plot for the Income vs CCAvg and see how many have personal loan or not. Below is the graph.

From the graph it looks like if you have low income and low CCAvg then there is almost no chance of bank distributing a personal loan, however as the income grows till 130 and CCAvg increases the personal loan as well seems to be related to both CCAvg and Income. However, once income is more than 135K then CCAvg seems irrelevant and most of the customer were given personal loan.

We are not saying this is causation nor correlated. We just highlighted as it seems interesting pattern and may need to dig in further for more concrete conclusions.

A graph of income and loan

Description automatically generated

1. **kNN Classification with k=3**

After normalizing all the data variables, we partitioned the data on the validation data and training data into the ration of 25% and 75% respectively. We build the data model with K =3 and got the accuracy of 96.08% which is very good, considering the depth.

We did not choose any explicit new values to generate the prediction and has used the validation data as our base to check the prediction values.

Below are the sample records 🡪

|  | **Age** | **Experience** | **Income** | **Family** | **CCAvg** | **Education** | **Mortgage** | **Securities\_Account** | **CD\_Account** | **Online** | **CreditCard** | **Personal\_Loan** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1501** | 30 | 4 | 35 | 2 | 0.30 | 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| **2586** | 47 | 23 | 149 | 4 | 6.10 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| **2653** | 30 | 5 | 121 | 2 | 3.10 | 1 | 408 | 0 | 0 | 1 | 0 | 0 |
| **1055** | 31 | 6 | 62 | 1 | 1.00 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| **705** | 62 | 36 | 30 | 3 | 0.70 | 2 | 0 | 0 | 0 | 1 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4141** | 43 | 19 | 63 | 3 | 2.10 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3168** | 51 | 25 | 180 | 1 | 1.70 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| **2478** | 30 | 5 | 178 | 2 | 6.70 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4214** | 46 | 22 | 89 | 1 | 2.70 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| **4180** | 36 | 6 | 11 | 1 | 0.67 | 3 | 0 | 1 | 1 | 1 | 1 | 0 |

We evaluated the model acuracy with kNN of 3 to be 96.08. We did more evaluations by changing kNN to range of 1-50 to see which model gives us the most consistent result and may potentially does not overfit the model. We did the plotting of accuracy vs k value and noticed that when k >=13 and <=23 we see the graph shows less volatility. Considering avoiding overfitting and more consistent result it looks like k =14 should be a good choice as number of neighbors are not very huge and we did consider good spectrum of neighbors as well as we are not making the calculations tedious as well.

Below is the chart for your reference.

A graph with a red line

Description automatically generated

1. Increasing the k reduces the accuracy as we are considering more further neighbors which tends to take away the fact of the having nearest neighbors to conclude. However, if we continue to use very low k value it means we are not considering enough data points to derive the nearest neighbor and predict/classify the variable of interest.

In our case it the graphs seems to isolate at k >= 13 and <=23. So, it be better to have more consistent accuracy at k =14 so we have more than enough data points but not too many.

1. A classification tree approach can also be used in the above scenario to decide on the personal loan.

Here the classification tree for your reference

A diagram of a family tree

Description automatically generated

1. After, building the classification tree, we did the accuracy testing of the model for different depts and at depth 4 it seems to provide more coherence between the training accuracy and validation predicted data.

The training data accuracy is.

Confusion Matrix (Accuracy 0.9886)

Prediction

Actual 0 1

0 3158 11

1 29 302

And prediction data accuracy is.

Confusion Matrix (Accuracy 0.9813)

Prediction

Actual 0 1

0 3158 11

1 29 302

As you notice both are close to each other and hence we will go with 4.

In our case it be better to go ahead with tree classification approach as it is easier to read, will work fine as data grows, as you need to derive the nearest 14 neighbors.

A tree graph would also assist the bank in quickly identifying if a customer can have a personal loan account or not based on the tree map.

Also, the accuracy in case of classification tree is higher.

**EBay Auction Data**

This data set has the data points if the bidding/auction was a competitive one or no.

**Build the Basis Classification Tree with max depth 7**

We built a classification tree using all the predictors however, we cannot use the variables, like End Day, categories, and currency as is; as they are text fields and model will not generate on text fields. Hence, we converted those values to the labeled values. And use those values in the model.

A diagram of a company's flowchart

Description automatically generated

As you can see the above tree has minimum 50 items on the terminal nodes and the tree depth is 7.

One rule we can build is for is - if the open price is 2 and closing price is 12 in this we have the first node condition openPrice <=3.82 so we go to the true node and we see the closePrice condition as <=3.685 so we go to the right as false and again our open price is 2 which is True for node hence we go to the right edge and notice that its 1. Hence for Open Price as 2 and Closing price as 12 we get the Competitive as 1.

**Does the Above Tree work?**

This tree does not seem to work in our case as in real life you would not know the closed price, you would not as a seller beforehand, so we should remove this from the training data. Also, currency is irrelevant as its same value and does not actually help in this scenario.

We should remove the variables that are not available at the time of auction start date, like closing Price, and we can remove fields like currency.

We can build a new model that captures more rules and have more structured approach in terms of determining if the auction was competitive or not. The enc\_endDay, ENC EndDay, Enc Category are the encoded values for as they cannot be put into the decision tree algorithm for classification hence, we labeled them and then did the scalar fit.

At 50% split we have an accuracy of 73% on training data and 71% on validation which is around similar and hence it could be a better decision tree then having a variable that is part of the decision tree whose values are not known at the beginning. This is incorrect and lead to have more inconsistent results.

Here is the decision tree-

A diagram of a number of items

Description automatically generated with medium confidence

The accuracy on validation data is as mentioned below is –

Confusion Matrix (Accuracy 0.7110)

Prediction

Actual 0 1

0 321 122

1 163 380

Again, there are more better ways in term so depth and sizing which we can explore, however in the current context we are stopping here.