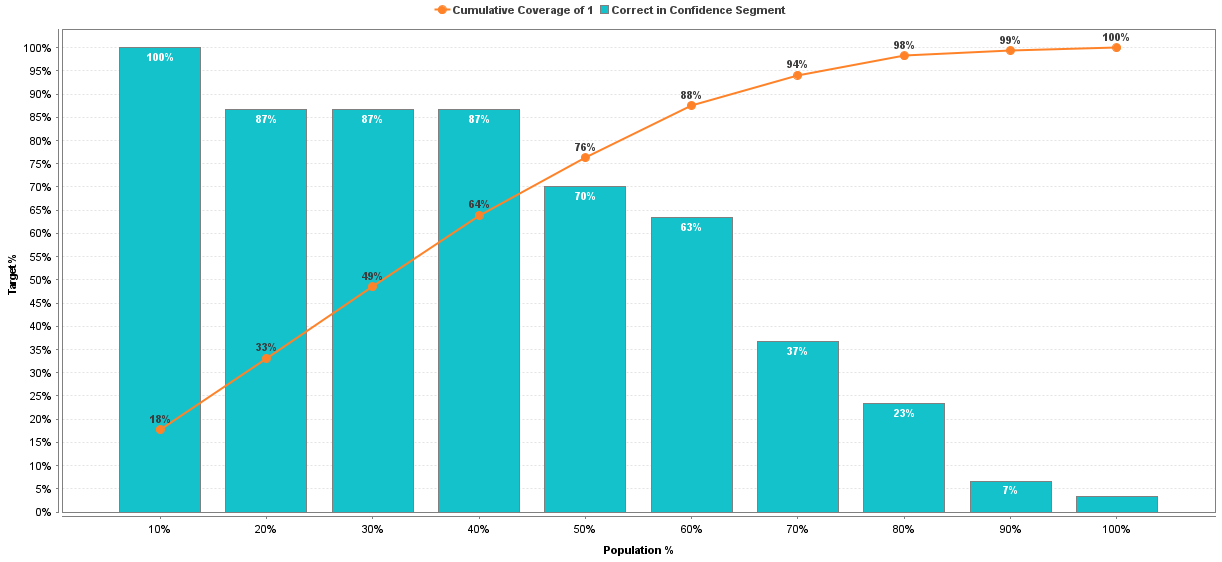
To prepare the data set, I remapped the frequency and spending attributes with the function f(x)=log(x+1), because they had long-tail distributions. I then performed z-normalization for all numeric attributes. Prior to building any predictive models, I split off 15% of the data set to test the overall performance of the best models found through parameter optimization. I investigated the accuracy of four types of model: decision trees, optimizing for splitting criterion; k-nearest-neighbors, optimizing for k; neural networks; and linear regression, optimizing for feature selection method.

|  |  |  |
| --- | --- | --- |
| Model | Best hyper-parameter(s) | Accuracy |
| Decision tree | Criterion = gain ratio | 0.785 |
| k-NN | k = 17 | 0.817 |
| Neural network |  | 0.879 |
| Linear regression | Feat. selection = greedy | 0.802 |

1. The highest accuracy belonged to the model from the neural network. Applying the model found this way to the test set, it has an accuracy of 0.810 and an f-measure of 0.824. In the plot below, the blue histogram shows the fraction of customers in each bin for whom the model correctly predicted made a purchase and the orange line shows the cumulative fraction of correct predictions out of the whole data set. The cumulative fraction line does not begin to flatten out until at least 50% of the population is included, at which point the model correctly predicts 76% of the customers who will make a purchase, which is significantly higher than random. I also applied the AdaBoost algorithm to combine multiple neural networks for better performance, but the cumulative curve in the lift plot did not significantly change.

|  |  |  |
| --- | --- | --- |
| Model | Best hyper-parameter(s) | rms error |
| Decision tree | Conf. = 0.30  Min. gain = 0.01 | 0.705 |
| k-NN | k = 16 | 0.682 |
| Neural network |  | 0.680 |
| Linear regression | Feat. selection = none | 0.703 |

1. I slightly modify the decision tree into a regression tree, and instead optimize for confidence and minimal gain when splitting, since the split criterion is fixed. I also look at each model’s rms error instead of accuracy, which is only defined for a binomial label attribute. The neural network again gives the best model. Applying the model found this way to the test set, it has an rms error of 0.745. I wasn’t able to try an ensemble method, because I couldn’t get either AdaBoost or Bayesian boosting to accept numeric labels.

|  |  |  |
| --- | --- | --- |
| Model | Best hyper-parameter(s) | rms error |
| Decision tree | Conf. = 0.00  Min. gain = 0.063 | 0.895 |
| k-NN | k = 16 | 0.870 |
| Neural network |  | 0.791 |
| Linear regression | Feat. selection = none | 0.865 |

1. Going back to the original data set, I restrict the data set by removing all records where Purchase is false. I do this prior to normalization, where it turns out that the Spending is an actual lognormal distribution when only considering records where spending happened. Otherwise, the analysis is the same as in Question 2B. The neural network again gives the best model. Applying the model found this way to the test set, it has an rms error of 0.876.
2. Unexpectedly, all of the models perform more poorly, which may be because the training set is half the size as before. The neural network is least degraded by this change, probably for the same reason is performed the best to begin with: it was already down-weighting less informative attributes, which were overrepresented in the half of the data set that had no purchases to begin with.