1QBit

Optimal feature selection in credit scoring and classification using a quantum annealer

White Paper

Optimal feature selection in credit scoring and classification using a quantum annealer

Andrew Milne, Maxwell Rounds, and Phil Goddard

Abstract

In credit scoring and classification, feature selection is used to reduce the number of variables input to a classifier. This can be done with a quadratic unconstrained binary optimization (QUBO) model, which attempts to select features that are both independent and influential. Quadratic optimization scales exponentially with the number of features, but a QUBO implementation on a quantum annealer has the potential to be faster than classical solvers. Tests were done using the German Credit Data from UC Irvine, and the results compared with those reported in the literature. In comparison with recursive feature elimination (RFE), a technique found in many software packages, QUBO Feature Selection yielded a smaller feature subset with no loss of accuracy. This opens up the possibility of using quantum annealers to programatically reduce the size of very large feature sets, especially as the size and availability of these devices increases.

Keywords: credit scoring, feature selection, computational finance, quadratic optimization, quantum annealing

1 Introduction

Credit scoring and classification is a significant problem in computational finance. Although the amount of money loaned globally is difficult to measure (OECD1 2016, World Bank 2017), in the United States, household debt alone is estimated to be around $14 trillion (US FED 2016). The Federal Reserve also reports that approximately 2% of these loans are non-performing. Superficially, this indicates that lenders are making good decisions 98% of the time. However, the U.S. Federal Deposit Insurance Corporation (FDIC1 2017) publishes yearly summaries of bank failures, and in the 15 years since 2001, there have been 547 failed institutions. Non-performing loans have been a major cause.

According to the 2016 Credit Access Survey by the U.S. Federal Reserve Bank of New York (FRBNY 2016), approximately 40% of U.S. credit applications are rejected. Moreover, between 20% and 40% of consumers *expect* their applications to be rejected (it depends on the type of credit), and many do not even apply. Yet, among these people, there may well be customers for the right kind of lender.

In a literature survey by Huang (Huang 2014), the academic approach to credit scoring is typically one of bigger data and bigger models. However, in a recent article in Forbes magazine, consumer lending veteran Matt Harris took a different view (Harris 2016):

> "Most start-up originators focus on the opportunity to innovate in credit decisioning. The headline appeal of new data sources and new data science seem too good to be true as a way to compete with stodgy old banks. And, in fact, they are indeed too good to be true."
>
> "The recipe for success here starts with picking a truly underserved segment. Then figure out some new methods for sifting the gold from what everyone else sees as sand; this will end up being a combination of data sources, data science and hard won credit observations."
>
> "...[P]rogressive and thoughtful traditional lenders like Capital One have mined most of the segments large enough to build a business around. The only way to build a durable competitive moat based on Customer Acquisition is to become *integrated* into a channel that is *proprietary*, *contextual* and *data-rich*." (original emphasis)

If Harris is correct, the key feature in new credit scoring and classification tools will not be their success rate in large-scale applications for which tools like FICO already exist (FICO 2017), but in their flexibility and ease of integration into specialized applications.

Feature selection has an important role. The operator of a credit scoring and classification system must sell it to the owners of proprietary channels, who might be skeptical of complex schemes aimed at different customers in different areas of business. In contrast, the idea of finding key features in a broad swath of data has a simple and natural appeal.

Ideally, the selected features should be both influential and independent. Yet here too one must be realistic. People lie. Data can mislead (Goel 2016). The "redundant" feature might actually be the *corroborative* feature. The correct balance of influence and independence will depend on the "hard-won credit observations" that Harris sees as crucial, and on the lender's confidence that the model genuinely includes them.

Later, we will define "independence" and "influence" in terms of correlation between data series, as has been done by other researchers in this field (Demirer 1998, Huang *op. cit.*). From this premise, the optimal balance between feature influence and feature independence can be expressed as a quadratic unconstrained binary optimization (QUBO) problem, with a parameter $\alpha\ (0 \leq \alpha \leq 1)$ representing the desired tradeoff. However, the classical solution time grows exponentially with the size of the feature set, and it was thought until quite recently that quadratic optimization could only be used in a hybrid fashion, where feature selection and classification are optimized separately.

In contrast, a high-speed solver makes it possible to use QUBO Feature Selection more widely in "wrapper models", where selection and classification are "wrapped" together and optimized holistically. This leads to better performance, and allows the analyst to focus on studying the data as opposed to managing the model, an obvious advantage for the operator of a *proprietary* and *data-rich* channel. For a fuller discussion of "filter models", "wrapper models", and "hybrid models", see (Huang *op. cit.*) or (Waad 2016).

In this paper, QUBO Feature Selection is applied to the original Hofmann data set in the Statlog (German Credit Data) Data Set, as published by the Machine Learning Repository at the University of California, Irvine (UCI) (Lichman 2013).

The results from QUBO Feature Selection are compared with the results from recursive feature elimination (RFE), a standard technique available in packages such as scikit-learn (Pedregosa 2011, scikit 2017). Two variants of the RFE method were studied:

- stand-alone RFE, where the desired number of features is set explicitly. The least influential features are eliminated one at a time until the desired number of features is left.
- RFE wrapped with cross-validation (RFECV), where the program evaluates the performance of its classifier after every feature removal, and terminates when the highest accuracy score has been found. Cross-validation is discussed in Section 7 (Evaluation Metrics). Broadly speaking, it involves using a part of the data set for training, and the remaining part for evaluation. The roles are then switched so that every data point is used an equal number of times in each role.

In both cases, logistic regression was used as the classifier. Without feature selection, logistic regression had a 75% success rate on the German Credit Data. Its performance is comparable to other methods reported in the literature, such as support vector machines (SVM), decision trees, neural networks, k-Nearest Neighbours (k-NN) classification schemes, and so forth (Waad *op. cit.*, Huang *op. cit.*). It is an easy-to-understand and widely used method that allows us to focus on the feature-selection step.

The feature subsets from the QUBO and RFE methods were compared with feature subsets discovered via random search. Since these give only a partial view of the problem landscape, perturbation searches were also made in the immediate neighbourhoods of the best QUBO subset and the best RFE subset. These showed that QUBO Feature Selection did a good job in finding the best local subset, but also that additional searching could be worthwhile.

We also give some examples of how QUBO Feature Selection can be implemented with the 1QBit quantum-ready SDK and used in an analysis, with about the same coding effort needed to use feature selection classes found in popular machine learning packages.

The main difference between QUBO Feature Selection and RFE lies in how aggressively each tries to reduce the number of feature variables in the feature subset. QUBO Feature Selection considers both the independence and the influence of the features under consideration. RFE focuses on eliminating features that are less influential. Both approaches yield good results on the German Credit Data, and both have a place in the analyst's toolkit.

2 Formulation of the credit scoring and classification problem

Assume that we have some data on past credit applicants that we believe will be useful in predicting the creditworthiness of new applicants. The data is composed of *features*, where each feature may be:

- an integer, where the order (lower to higher) has potential meaning, for example, bank balances, years of education, etc.
- a category, such as a geographic region code, where higher and lower values are arbitrary. It may also include value such as "missing" or "refused to answer".
- a decimal number, representing, for example, age, dollar amounts, interest rates, and so on, where the order has meaning. Data such as the latitude and longitude of the applicant's home would be better represented as a category.
- a Boolean (yes/no) value, which may be considered as an integer or a category.

A credit observation, or sample, consists of an observation for each feature, some means of identifying the applicant, and the outcome. The raw data may come from many sources.

In practice, we clean the raw data to create a vector of "feature variables" for each observation. For example, we may convert some or all of the categorial variables to binary indicators, a step described in a later section of the paper. We may also scale the data and (in some cases) replace missing values with inferences. In the description that follows, we will assume that these steps have been taken and the data is in a form suitable for input to our feature selector and classifier.

For convenience, we organize the clean data as a matrix of m rows and n columns. Each column represents a feature, and each row represents the specific data values for a specific past credit applicant.

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ u_{21} & v_{22} & u_{23} & \dots & u_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & u_{m3} & \dots & u_{mn} \end{bmatrix}$$

Our goal is to determine how the data on past applicants can inform us on the creditworthiness of new applicants. For this, we need a record of the decisions that were made. We represent these as the m-element vector

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}.$$

The v_i will be constrained to take on the values 0 and 1, where 0 represents the acceptance and 1 represents the rejection of credit application i. Conceptually, the classifier will be a "credit risk detector" that signals when an applicant should not be granted credit. This is consistent with the U.S. Fed data showing that rejections are less common than acceptances. Acceptance is the rule, and rejection is the exception. The credit risk detector has many analogues in other fields, and there is well-established terminology in the literature, for example, (Powers 2011).

3 Feature selection

Assume that from the original set of n features we want to select a subset of K features to use in making a credit decision. Our reasons for doing this may be manifold. Data costs money. Some data may be redundant. Other data may be so weakly correlated with the outcome that it acts more like noise than anything useful.

We want to search broadly, without prejudging the data. However, the number of possible subsets for each K is given by the combinatorial function $C(n, K)$. Even if we can eliminate some candidates at the outset, the search space will typically be very large. We want to focus our search on areas where a good subset is likely to be found.

Mathematically, our goal will be to find the columns of U that are correlated with V, but not correlated with each other. We have not yet defined what correlation means here, but we assume that such a calculation is possible and that the value of the correlation coefficient can take on values from -1 to 1. Note that we can interpret "correlation" quite liberally: a "hard-won credit observation" might appear as a "hard requirement" that an attribute be present. Taking this a step further, the conversion of categorical variables to binary indicators (see below) can be expanded to include corroborations that the lender sees as being necessary.

Let ρ_{ij} represent the correlation between column i and column j of the matrix U, and let ρ_{V_j} represent the correlation between column j of U and the single column of V.

To find the "best" subset, we introduce n binary variables x_j, which have the property

$$x_j = \begin{cases} 1, & \text{if feature } j \text{ is in the subset} \\ 0, & \text{otherwise} \end{cases}$$

We refer to these collectively as the vector $\mathbf{x}$, where

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}.$$

We will associate the best subset with the value of $\mathbf{x}$ that minimizes an objective function, which we construct from two components. The first component represents the influence that features have on the marked class, shown here in a form that increases as more terms are included:

$$\sum_{j=1}^{n} x_j |\rho_{Vj}|.$$

The second component represents the independence. The form shown below increases as more of the cross-correlated terms are included, which is the opposite of independence.

$$\sum_{j=1}^{n} \sum_{\substack{k=1, \\ k \neq j}}^{n} x_j x_k |\rho_{jk}|$$

To obtain an objective function that is maximized at the optimum, we will need to subtract it from the first term. We combine the terms using a parameter α $(0 \leq \alpha \leq 1)$ to represent the relative weighting of independence (greatest at $\alpha = 0$) and influence (greatest at $\alpha = 1$). We then negate the expression overall to optimize at the minimum to obtain

$$f(\mathbf{x}) = -\left[\alpha \sum_{j=1}^{n} x_j |\rho_{Vj}| - (1-\alpha) \sum_{j=1}^{n} \sum_{\substack{k=1, \\ k \neq j}}^{n} x_j x_k |\rho_{jk}| \right].$$

We can make use of the property that $x_j x_j = x_j$ for binary variables, which allows us to rewrite the summation as a vector product:

$$f(\mathbf{x}) = -\mathbf{x}^T Q \mathbf{x}\,.$$

From this, we can express the problem in terms of the `argmin` operator, which returns the vector $\mathbf{x}^*$ for which its function argument is minimized:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\text{argmin}} \left[-\mathbf{x}^T Q \mathbf{x} \right].$$

4 Classification

The classification problem may be stated as follows: given a row vector $\mathbf{u}$ of new observations from a new applicant, calculate whether the vector belongs to the creditworthy class. More specifically, find a function $f(\mathbf{u})$ that returns 0 for acceptance and 1 for rejection.

One of the premises of machine learning is that such a function can be derived from a programmatic analysis of existing data. The existing data is divided into a training set and a test set. A candidate function is derived from the training set, and its performance is measured on the test set. Much has been written on the best way to define such functions, the best way to divide the data, how to adapt to new data, and so on. For example, see the citation lists at the UCI Machine Learning Repository (UCI MLR 2016), or (Chen 2010). For a useful cautionary note on data patterns, see (Anscombe 1973).

For this paper, we used a simple classifier based on logistic regression (scikit *op. cit.*). Unlike with *linear* regression, where one can imagine two continuous variables and fitting a line to a scattered set of points, *logistic* regression assumes that the dependent variable is a category, for example, the zero and one of a binary classifier. The fitted line no longer predicts the value of the dependent variable, but rather the probability that the dependent variable will have a specific value. It is a well-established technique with a long pedigree (Cox 1958).

5 Binarizing, scaling, and correlating the German Credit Data

The German Credit Data under consideration was originally published in 1994 by Hans Hofmann at the Institute for Statistics and Econometrics the University of Hamburg. It has been studied extensively (UCI MLR *op. cit.*, Huang *op. cit.*).

The data consists of 20 features (7 numerical, 13 categorical) and a binary classification (good credit or bad credit). There are 1000 rows, of which 700 are "good" and 300 are "bad". The data is intended for use with a cost matrix, where giving credit to a bad applicant is five times as bad as not giving credit to a good applicant. In this study, however, we were concerned mainly with the relative "predictive power" of the feature subsets, so the cost matrix was not used.

The data was prepared as follows:

- The *german.data* file from UCI was imported into a Jupyter (iPython) notebook as a pandas `DataFrame` and given column headers with names from the accompanying *german.doc* file.
- The categorial variables were converted to "one-hot" binary indicators using the `DictVectorizer` class from scikit-learn.
- The first binary indicator in each group was removed (for k indicators, only $k-1$ are independent).
- All of the numerical features were scaled to mean zero and variance one.
- The classification variable was transformed to $0 =$ good, $1 =$ bad.

A variety of correlation methods were studied. These led to small differences the feature subsets, but no real difference in the accuracy of the classifications. It was noted, however, that methods with a "smooth" distribution of coefficients (Spearman, Pearson, etc.) worked better with the quadratic objective function than correlation methods with sharper jumps, such as "mutual information" scores (scikit *op. cit.*, Rosenberg 2007). In the end, the Spearman method was chosen as it is simple and easy to reproduce. However, this is an area where more research is needed.

The binarization and scaling procedure transformed the 20 features in the German Credit Data into 48 feature variables. For example, Attribute 1 ("Status of existing checking account") which had four possible values in the original data, was converted into three binary indicators. (In the resulting `DataFrame`, these appeared as columns "ChqStat=A12", "ChqStat=A13", and "ChqStat=A14".) These 48 feature variables are the input to the feature selection algorithm. The feature subset at the output of the feature selector then forms the input to the classifier.

6 Coding for the 1QBit SDK versus other machine learning packages

The 1QBit SDK provides a toolkit for solving quadratic unconstrained binary optimization (QUBO) problems. The details of the underlying quantum hardware are abstracted away, so that the code used by the analyst is no more complex than the code needed to use machine learning packages such as Weka (Bouckaert 2009), or scikit-learn (Pedregosa *op. cit.*).

The `RFECV` class from scikit-learn is a good example. Recursive feature elimination is "wrapped" with a classifier chosen by the user, for example,

```
featureMatrix = U.values
classVector   = V.values
estimator     = LogisticRegression()
selector      = RFECV(estimator, step = 1, cv = 3)
selector      = selector.fit(featureMatrix, classVector)
indexList     = selector.get_support()
featureList   = np.where(indexList)[0]
```

The 1QBit SDK allows `LogisticRegression()` and other classes from scikit-learn and other packages to be wrapped with its solution to the QUBO problem, along with a testing and scoring class such as scikit-learn's `ShuffleSplit` or `StratifiedShuffleSplit`. Examples of how to do this are provided at the 1QBit website (1QBit 2017).

Inside the wrapper, a search is made for the best value of α for the feature selections and the best values for the parameters used by the classifier.

At the core of the search loops, or wherever the QUBO problem must be solved to obtain a candidate feature set, we construct a Q matrix using the `QuadraticBinaryPolynomialBuilder` class, which returns a polynomial object representing the objective function seen earlier in the paper. The poly object is passed to a solver and the solution is returned as a list of ones and zeros, referred to as a configuration. We convert this to a list of integer indices that can be used to extract columns from a pandas `DataFrame` or NumPy array, which is typically how feature matrices and class vectors are passed to machine learning methods. A simplified example is shown below.

```
builder           = qdk.QuadraticBinaryPolynomialBuilder()
##                ... Some code to construct the Q matrix as in the equations above...
poly              = builder.build_polynomial()
solver            = qdk.MultiTabu1OptSolver()
solutionList      = solver.minimize(poly)
lowEnergySolution = solutionList.get_minimum_energy_solution()
config            = lowEnergySolution.configuration
featureList       = np.argwhere(config.values()).flatten().tolist()
featureMatrix     = U.iloc[:, featureList].values
classVector       = V.values
estimator         = LogisticRegression()     # and so on...
```

In this paper, we fixed the parameters for the classifier, and focused on studying how the value of α affected the feature subset returned by QUBO Feature Selection, and how that subset affected the accuracy scores returned by the logistic regression classifier. A full holistic optimization across all of the available parameters is a topic for future work.

7 Evaluation metrics

The evaluation of QUBO Feature Selection and recursive feature elimination was performed by wrapping them with the `LogisticRegression` model from scikit-learn. The evaluation metric was defined as the unweighted accuracy, that is, the number of correct classifications divided by the total number of classifications made. Other metrics from scikit-learn were attempted, but they always led to the same optimal alpha or RFE feature set. Unweighted accuracy was kept for compatibility with other work and ease of understanding, for example, (Huang *op. cit.*).

Testing and scoring were performed using the `StratifiedShuffleSplit` cross-validation class from scikit-learn. Given the feature matrix, this class returns sets of row indices that can be used to divide the matrix rows into a training set and a test set. The separation is done in folds, with the number of folds set by an argument. For example, a shuffle and split with 5 folds will take 80% of the matrix for the training set and 20% for the test set, and repeat this process until all 5 of the possible 20% folds have been used as test sets.

The accuracy score for each split is slightly different. Since these reflect a random selection of data for training and testing, it is conventional to report the mean of the individual accuracy scores. We follow the convention used by scikit-learn (sklearn cv 2017) and calculate the error bars for the 95% confidence level.

8 Experimental results

8.1 Establishing the zero-rule and other baseline properties

The German Credit Data has 700 class 0 samples ("good credit") and 300 class 1 samples ("bad credit"). A zero-rule classifier that assigns *all* of the samples to class 0 will therefore achieve a success rate of 70%.

We want our proposed feature selection and classification scheme to do better than the zero-rule. We want the feature selection component to choose subsets that are better than randomly selected subsets and (for that matter) better than no selection at all. These will form a baseline against which QUBO Feature Selection can be compared.

Feature selection is motivated by the intuitive concept that not all features are equally important. For example, in Figure 1 below, we see the feature variables from the binarized German Credit Data ranked by their Spearman correlation coefficient with the classification variable. There are four relatively important features at the left-hand end, followed by a gradual, almost linear decline. It turns out that the four features on the left are not enough to form a predictive subset on their own, so the problem for the feature selector is to find where on the line "enough is enough".

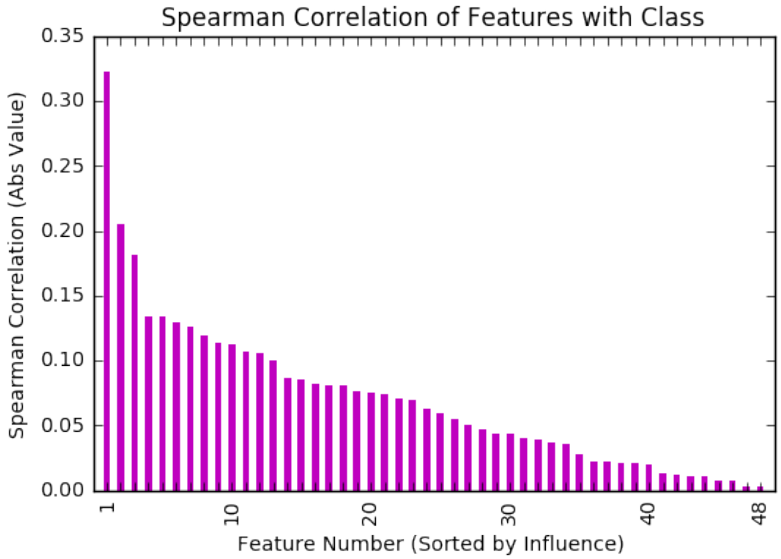

Figure 1: Spearman correlation. The most influential features involve the status of the applicant's chequing account, savings account, and loans at other institutions, all which are included in all practical feature subsets. As we move towards the right, however, the influence of each new feature declines.

It also turns out that the smooth decline of the coefficients works well with the quadratic objective function used in QUBO Feature Selection, as was seen in comparisons of Pearson, Spearman, and Kendall correlations (readily available in the pandas package). In contrast, the use of a mutual information score in place of a correlation coefficient was not as successful. The integer features were binned into ordered categories and generally behaved as expected. However, the more arbitrary binarized categories (e.g., loan purpose) had uniformly low mutual information scores, and the objective function tended to cycle amongst features. A plot of the ranked mutual information scores is shown in Figure 2.

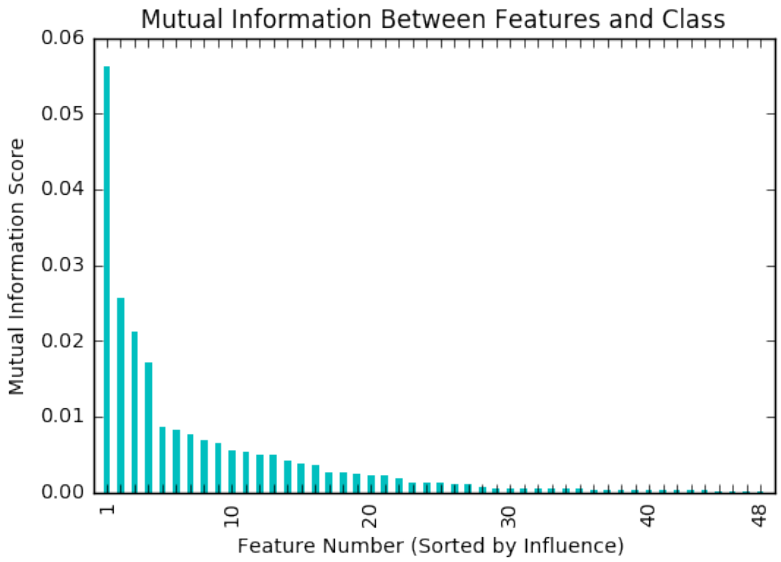

Figure 2: Mutual information scores. The age, term, and credit amount fields were binned into categories. However, the net effect was fluctuation in the accuracy scores and a lower mean accuracy at a "best" feature subset with 34 elements.

The results shown in this paper were all calculated using the Spearman correlation coefficient. As mentioned previously, this is an area where more research is needed, for example, using other data sets with a broader mix of feature variables.

Before we select any features, however, we first examine how well the logistic regression classifier performs on the full feature set, that is, all 48 feature variables. We do this using the `StratifiedShuffleSplit()` class from scikit-learn. Figure 3 shows that the mean accuracy depends on how many times the data is shuffled, and on how the data is split between the training set and the test set.

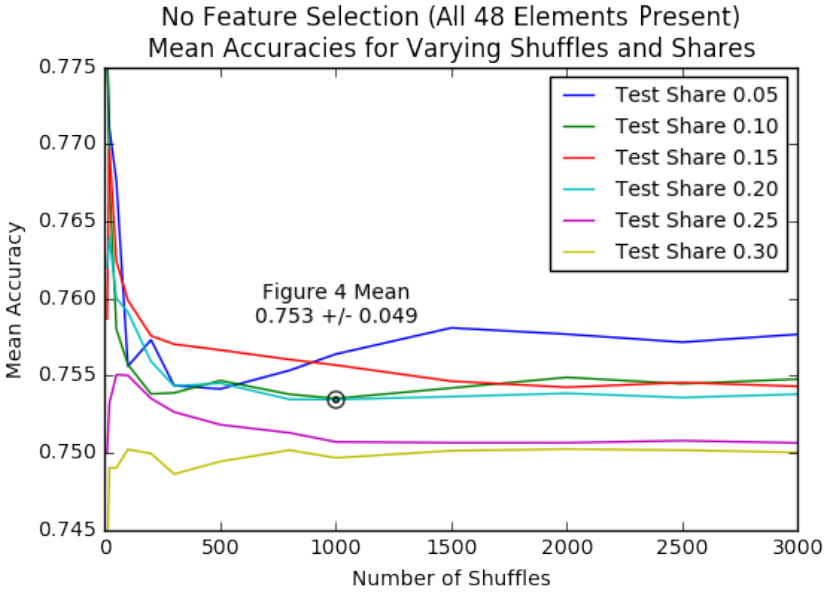

Figure 3: Mean accuracies measured for various numbers of shuffles (10 to 3000) and for different fractions of the data assigned to the test set.

The combination of 1000 shuffles and 20% test share was chosen arbitrarily as the standard for initial performance comparisons, being much more convenient than the larger numbers. It avoids the fluctuations found below 500 shuffles, and is close to the converged scores at 3000 samples and above. For the definitive score comparison, however, the full 3000 shuffles were used. The results for 10%, 15%, and 20% share were always very close and typically in the median position. The 20% test share was therefore used throughout.

It can be seen in Figure 3 that the mean accuracy increases as the test share is reduced, that is, a bigger training set yields a more accurate predictor. However, the difference is small in relation to the dispersion of scores from different shuffles, as can be seen in Figure 4.

Note that the 30% share curve does not fluctuate as much as the curves for smaller shares. It turns out that scikit-learn chose a 3-fold default for RFECV, and this may be one reason why it can attain a good (although not optimal) result in relatively little time. It is also important to keep track of absolute numbers as well as percentages: a 5% test share of 1000 samples consists of only 50 samples, which stratification on the German Credit Data will constrain to 35 good credit samples and 15 bad samples. It fluctuates widely at the outset, and converges slowly.

In Figure 4, we see the distribution of accuracy scores for all 48 features at a 20% test share (800/200 train/test split) counted over 1000 shuffles. Stratification forces the training set and the test set to have the same 70/30 distribution of good and bad credit samples, so that a 200-sample test set will contain 60 bad credit samples chosen from 300 in the set overall. In a large number of shuffles there will inevitably be some repetition (a point highlighted by scikit-learn in its documentation). Thus, although the data looks "Gaussian" and fits into the Gaussian overlay, in practice there are certain scores that occur more frequently, and extreme values are not observed above a certain limit. Note that in comparison to the spread of accuracy scores from 0.7 to 0.8 seen in Figure 4, the (converged) spread of mean scores from 0.75 to 0.76 in Figure 3 is relatively small. So long as we avoid small test shares and low numbers of shuffles, the error bars from the accuracy scores will dominate the uncertainty overall.

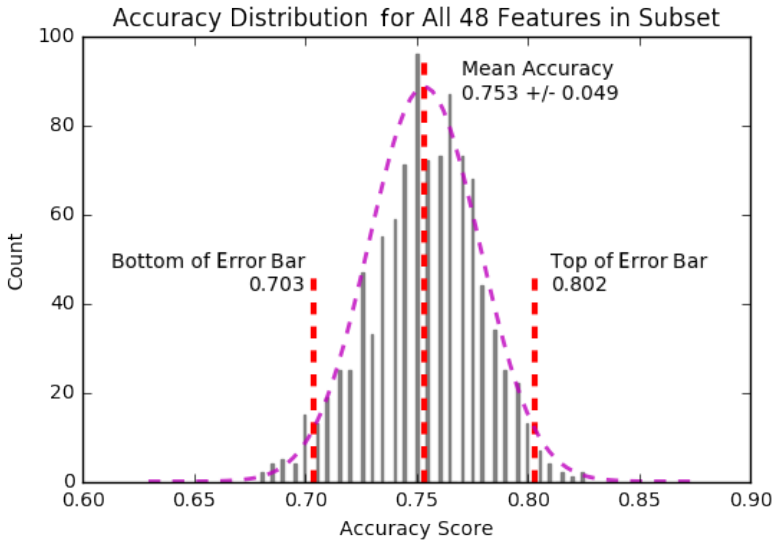

Figure 4: Accuracy scores for all 48 samples, using 1000 shuffles with 20% test share. This is a typical distribution for accuracy scores in the German Credit Data.

We now take a moment to examine the behaviour of logistic regression on feature subsets with fewer than 48 features.

The number of possible subsets is given by the combinatorial function $C(48, K)$, where K is the cardinality of the subset. The largest number of possible subsets occurs when 24 feature variables are selected, and is approximately 32 trillion. There are some 280 trillion subsets possible overall.

It is not possible to test these trillions of subsets systematically. However, we can gain an idea of how they behave from random sampling. For example, Figure 5 shows the accuracy of logistic regression for 10,000 randomly selected subsets at each of the 48 possible cardinalities, which examines 432,354 feature subsets out of 281,474,976,710,656 possibilities. If we record the best of the mean accuracies for each group of 10000 subsets, and plot them separately

(the red dots in figure 5), we can identify a "best detected" subset at cardinality 35, with an accuracy of 0.76 ± 0.05. We then examine how this mean was calculated from the accuracy scores for 1000 shuffles with a 20 % test share. In Figure 6, we see that the variance is large, and comparable to what we saw with all 48 features present. Feature selection looks to be a search for small improvements in collections of very noisy test results.

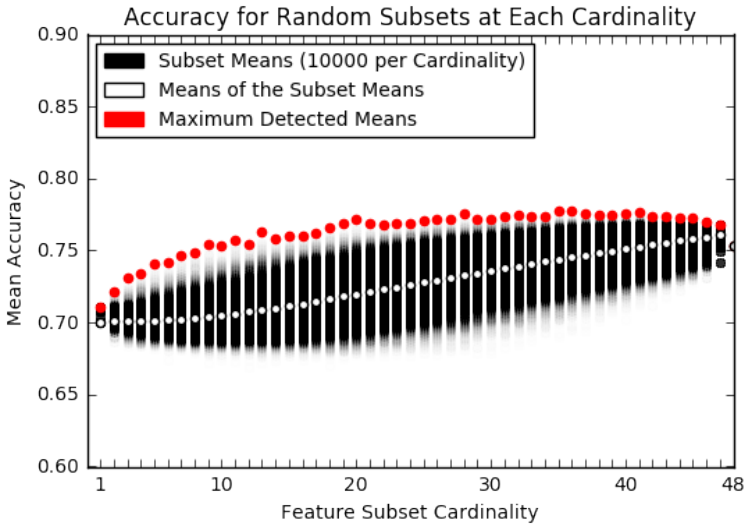

Figure 5: Sample means plotted as nearly transparent black circles. The largest mean found for each cardinality group is rendered in red.

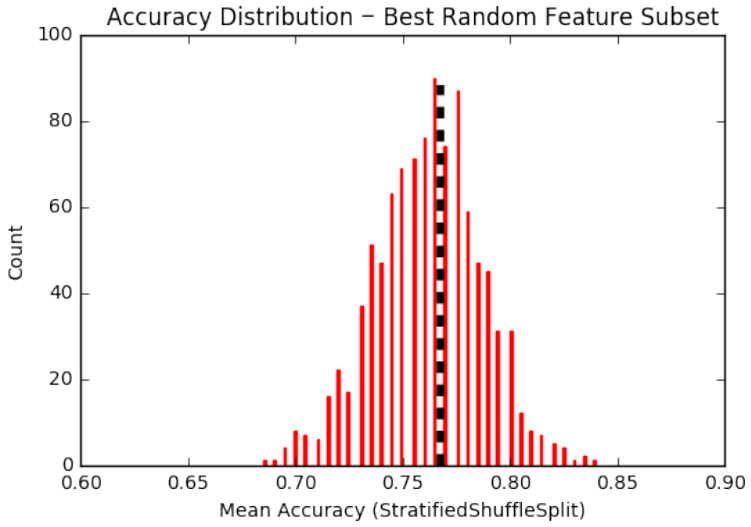

Figure 6: Distribution of accuracy scores for the best feature subset found through random search.

It must be stated at this point that accuracy scores reported by other researchers on the German Credit Data are typically in the 70% to 75% range, with standard deviations around 5%. For example, see (Chen *op. cit.*), (Rao 2016), or (Huang *op. cit.*). Our 0.76 ± 0.05 baseline accuracy puts QUBO Feature Selection squarely in the mainstream.

We can now summarize our baseline requirements as follows:

- Select a feature subset with 35 features or less;
- Deliver accuracy equal to or better than 0.76 ± 0.05
- Calculate the feature subset in an efficient way that can scale to larger initial feature sets.

8.2 QUBO Feature Selection with logistic regression

QUBO Feature Selection and a logistic regression classifier were "wrapped" together. Practically speaking, this means that the feature selector and the classifier were placed together at the interior of the loops used to optimize the selection and classification parameters. For simplicity, the only optimization parameter was α, which determines the relative weighting of *independence* (greatest at $\alpha = 0$) and *influence* (greatest at $\alpha = 1$).

It was found that the cardinality of the feature subset tended to increase with α, as had been noted by other researchers (Huang *op. cit.*, Demirer *op. cit.*). However, an advantage of the wrapper model is that optimization can be done "at the α level" without looking at the details of the subsets.

Figure 7 shows the full range of α from 0 to 1. On the left-hand side, where α is close to zero, the emphasis is on feature independence. This favours small subsets, and since their regression coefficients are often not large enough to "push" the classifier across the cutoff point of $p \geq 0.5$, the predicted class is 0. They classify almost all of the samples as "good credit" and achieve the zero-rule's 70% success rate.

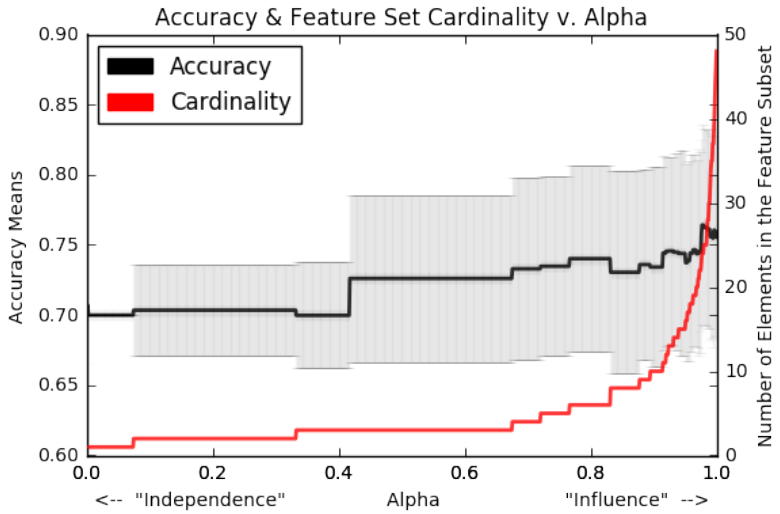

Figure 7: Increase in accuracy as influential features are chosen over independent ones.

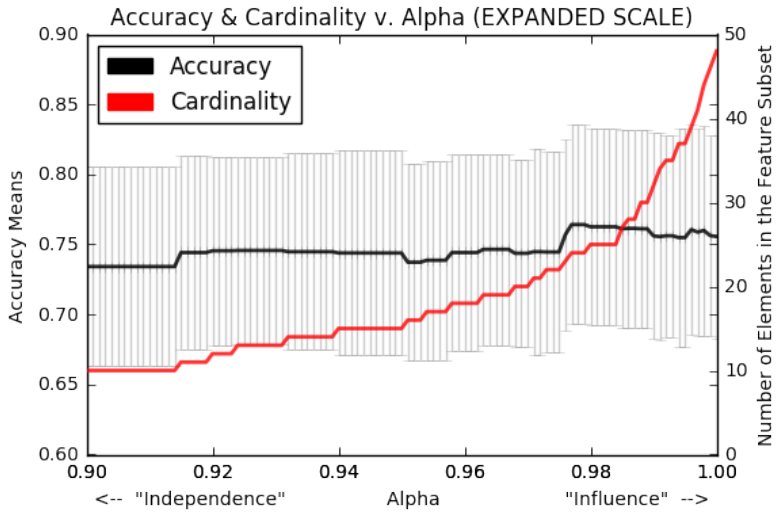

Figure 8: A closer look at QUBO Feature Selection in the $\alpha = 0.9$ to $\alpha = 1$ region.

In Figure 8, we look more closely at the region between $\alpha = 0.9$ and $\alpha = 1$. Here, the emphasis is on feature influence, and the subsets eventually grow to include all 48 available feature variables.

It is interesting that accuracy increases with the size of the subset, reaches a peak at $\alpha = 0.977$ with 24 elements, and then declines gradually as more features are added. The drop in accuracy to the left of $\alpha = 0.977$ is quite sharp, and although it is encouraging to see a global maximum so clearly defined, this may be due to the data, and should be further investigated.

8.3 Recursive feature elimination with logistic regression

Recursive feature elimination is a technique for pruning features from a feature list. The procedure begins with a set of available features. It fits the logistic regression model, and eliminates the feature with the lowest weight. The fitting and elimination is continued until the desired number of features is reached. In recursive feature elimination with cross-validation (RFECV), the fitting is accompanied by testing, using a training set and test set chosen according to a folding parameter.

Conceptually, RFE is like starting the QUBO objective function at $\alpha = 1$ and working downwards towards $\alpha = 0$, except that RFE is recursive and treats each iteration as a new feature set[1]. Unlike QUBO Feature Selection, RFE does not test explicitly for feature independence, nor does it allow a feature to "come back" after it has been eliminated.

We began with the direct version of RFE, where we specified the desired number of features and then measured the performance of the returned feature subset.

The accuracies were measured with the same 1000 shuffles and 20% test share that was used with the other methods. The results are shown in Figure 9.

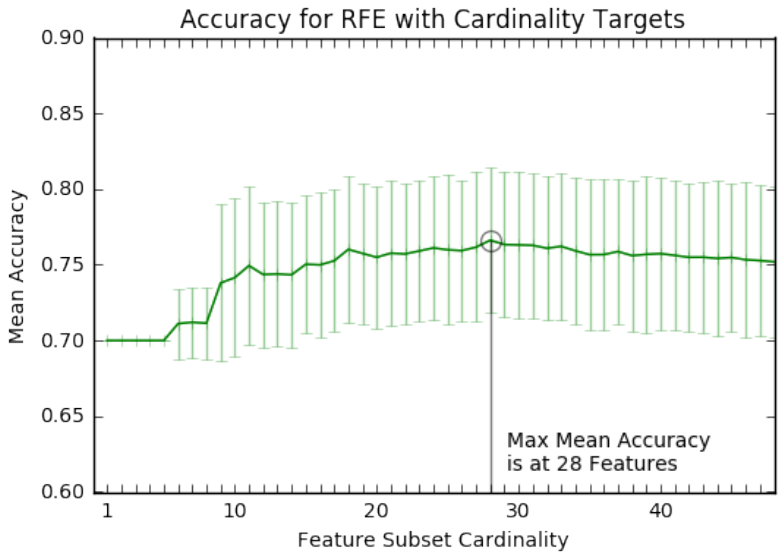

Figure 9: Recursive feature elimination for cardinality targets from 1 to 48, using 1000 shuffles with 20% test share. Error bars represent a confidence level of 95%. The maximum mean accuracy is 0.77 ± 0.05.

RFECV was very fast, although it converged to different feature subsets as the cross-validation settings and random seeds were varied. In practice, however, it was easy to search these (and faster than running RFE for thousands of shuffles). RFECV ultimately delivered a 31-element feature subset with an accuracy of 0.76 ± 0.05, comparable with the other methods.

[1] One could imagine the recursive elimination of features as α is iterated from 1 down to 0. A recursive version of QUBO Feature Selection would make an interesting topic for future work.

8.4 Comparison of QUBO Feature Selection and recursive feature elimination

Figure 10 shows the mean accuracies for QUBO Feature Selection and recursive feature elimination (and, for comparison, a random search of 10,000 sample subsets). The differences between the methods are smaller than the error bars.

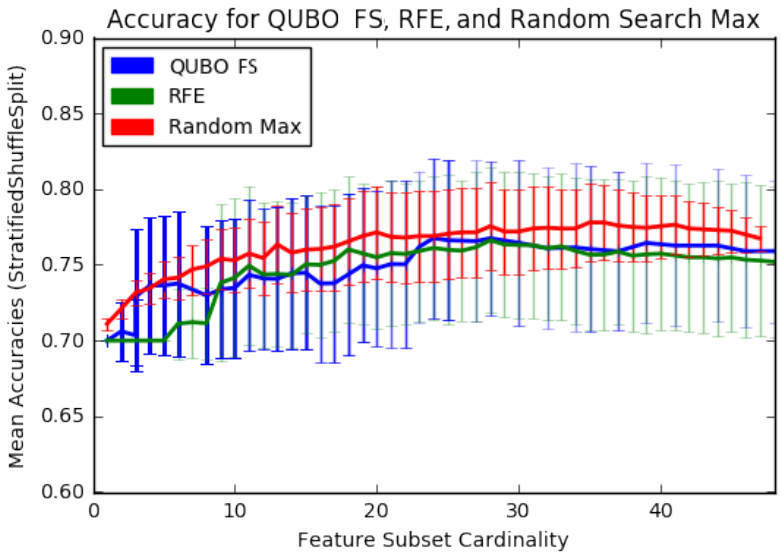

Figure 10: Comparison of QUBO Feature Selection, recursive feature elimination, and random search over all 48 cardinalities.

In Figure 11, a closeup view of the the region between 20 features and 49 features shows that the results never differ by more than the spread of mean accuracy scores in Figure 3.

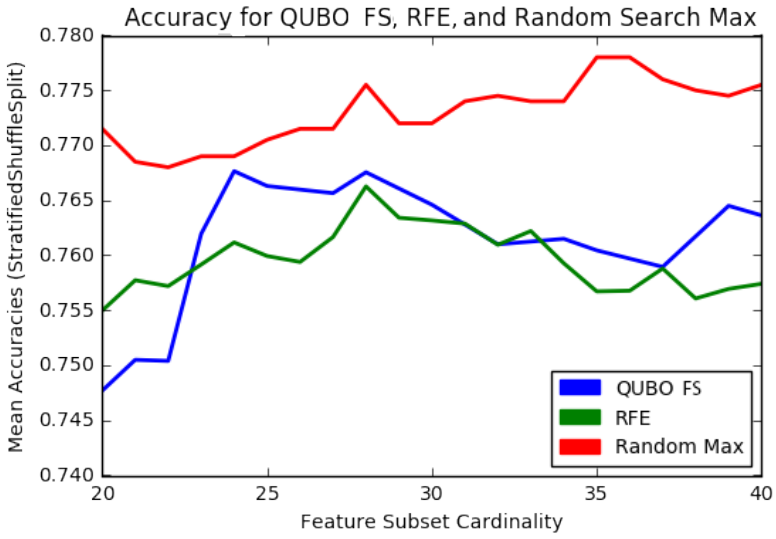

Figure 11: Comparison (using a different vertical scale) of QUBO Feature Selection, recursive feature elimination, and random search.

When the best subsets from each method are compared at 3000 shuffles, we obtain the following table, ranked by the number of features in the subset.

Method	Accuracy	F1-Score	Number of Features
QUBO Feature Selection	0.764 ± 0.049	0.54 ± 0.1	24
RFE28	0.767 ± 0.050	0.55 ± 0.1	28
RFECV	0.764 ± 0.050	0.54 ± 0.1	31
Rand10k	0.762 ± 0.050	0.54 ± 0.1	35

These are equivalent accuracies, with QUBO Feature Selection giving the smallest feature subset. However, random feature selection with 10,000 samples per cardinality had the lowest score of all, which shows that 10,000 samples (out of several trillion) cannot tell us much about extreme values. There may be other subsets "out there" that were missed by both QUBO Feature Selection and RFE. To assess this possibility, we examine some of the feature subsets that are "near" the QUBO Feature Selection and RFE results, in the sense that they differ by one or two features.

The F1-Score was computed using the `f1_score()` method of the `LogisticRegression()` class from scikit-learn. It is defined as the harmonic mean of the precision and the recall, where *precision* is the ratio of true positives to all predicted positives, and *recall* is the ratio of true positives to all actual positives (Powers 2011). The above values reflect the classification of the German Credit Data without the application of a cost matrix, and can be compared with results in the literature that were calculated in the same way.

8.5 Comparison with potentially missed subsets

In Figure 12, we see the mean accuracy results from feature subsets that have the same cardinality as the best subsets, but which differ in one feature, two features, and so on. Gaussian curves showing the distribution of the accuracy scores for the best subsets have been overlaid on the histograms.

The dashed vertical lines represent the means of the best subsets. The vertical bars represent the counted occurrences of mean accuracies for the perturbed feature subsets. QUBO Feature Selection does a better job than RFE in finding a subset that is better than its neighbours. This reflects the behaviour of the quadratic objective function, which considers all of the feature sets simultaneously (especially in the quantum annealer implementation). In contrast, once a feature has been eliminated by RFE, there is no possibility of bringing it back on a later iteration.

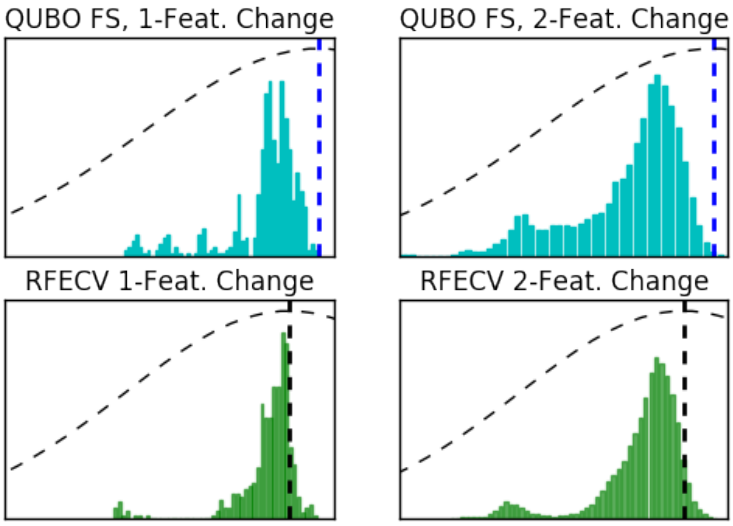

Figure 12: Comparison of mean accuracies from the "best" QUBO Feature Selection (blue) and "best" RFE (green) with mean accuracies from feature sets of the same cardinality but differing by one or two features. Gaussian distributions with the corresponding "best" standard deviations have been overlaid. The horizontal scale on all four graphs is the same, in both this figure and the next.

The same behaviour is observed when adding or subtracting a feature from the best subsets. In Figure 13, we see that the accuracy of the best QUBO subset is again better than the perturbed subsets.

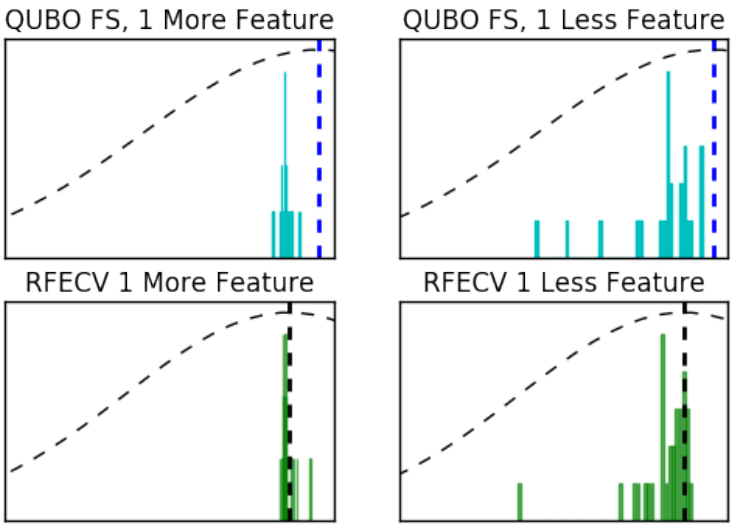

Figure 13: Comparison of mean accuracies from the "best" QUBO Feature Selection (blue) and "best" RFE (green) with mean accuracies from feature sets with one more feature and one less feature. Gaussian distributions with the corresponding "best" standard deviations have been overlaid.

9 Comparison with previously reported results

The German Credit Data was originally published in 1994 by Hans Hofmann at the Institute for Statistics and Econometrics at the University of Hamburg. Since that time, it has been studied extensively. At the time of writing, the most thorough survey of how quadratic optimization compares with other methods is given by Waad (*op. cit.*). Waad also used a publicly available machine learning package, Weka 3.7.0 (Bouckaert 2009), and created a "three-stage feature selection fusion" technique with QUBO Feature Selection as the first stage, which yielded very good accuracies. Waad's results are not directly comparable to the results in this paper, since they were given in terms of precision and recall, which are affected by whether the 5:1 cost ratio prescribed by Hoffmann has been applied in the training set. In this paper, we did not use the cost ratio and Waad does not mention it. Also, the Weka package does not have a function like scikit-learn's `StratifiedShuffleSplit()`, and in Waad's reported experimental procedure, the division of the samples into a training set and a test set was done at an early stage with 10 folds but no shuffling.

Taken in total, however, Waad provides strong motivation for studying how QUBO Feature Selection might be used as part of a larger procedure. For example, Figure 11 shows that, although QUBO Feature Selection found a very good subset with 24 elements, there were other subsets nearby that were slightly better. Additional searching could uncover more.

For feature selection overall, Chen and Li (Chen *op. cit.*) were able to achieve good results with 12 features from the original German Credit Data with its 20 categorical and integer (or fixed decimal) features. These were manually selected after comparing various correlation measures between the features and the classification. Chen and Li did not use the programmatic binarization of categorical variables that came into greater popularity between 1998 and the present day. Their results are primarily for support vector machines and do not deliver notable accuracy, especially since Chen and Li also report performing only a single 10-fold cross validation. The real lesson from this early work is that correlation makes a difference, and that automatic binarization might not always be a good idea.

The thrust of this paper has been to show how QUBO Feature Selection can be used with common techniques from publicly available software packages as a means of programmatically reducing the number of feature variables, including those created by the wholesale binarization of categorical variables. This was done successfully, but the literature shows that there is still some progress to be made.

10 Conclusions

On the binarized German Credit Data, QUBO Feature Selection delivered a smaller feature set (24 features) than either recursive feature elimination (28 features) or recursive feature elimination with wrapped cross-validation (31 features). All three methods showed comparable accuracy. A priority for future work is to study the behaviour of the QUBO method on different data sets, with different correlation methods, and with a multi-stage approach that could improve its performance. The authors hope that the availability of QUBO Feature Selection via 1QBit's quantum-ready SDK will make this method more accessible to researchers interested in studying its possibilities.

11 Acknowledgements

The authors would like to thank Majid Dadashi and the 1QBit Software Development Team for creating the SDK, and their assistance with its use. Anna Levit provided useful comments on the draft. Jaspreet Oberoi contributed the idea that led to *recursive* QUBO Feature Selection, a concept whose possibilities we have yet to explore.

12 References

1QBit (2017), qdk.1qbit.com/documentation/, accessed Jan. 23, 2017

Anscombe (1973), Anscombe, F. J., Graphs in Statistical Analysis, The American Statistician, Vol. 27, No. 1 (Feb., 1973), pp. 17–21. Stable URL: http://www.jstor.org/stable/2682899 (and the summary on the Wikipedia is excellent)

Bouckaert (2009), Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., and Scuse, D., Weka manual (3.7.1). This is the citation given by Waad (2016), and describes the Weka features available at the time. For additional details, see www.cs.waikato.ac.nz/ml/weka/

Chen (2010), Chen, Fei-Long , Li, Feng-Chia, Combination of feature selection approaches with SVM in credit scoring, Expert Systems with Applications 37 (2010) 4902–4909, www.elsevier.com/locate/eswa

Cox (1958). Cox, David, "The regression analysis of binary sequences (with discussion)". J Roy Stat Soc B. 20: 215–242. JSTOR 2983890

Demirer (1998), Demirer, Riza and Eksioglu, Burak, Subset Selection in Multiple Linear Regression: A New Mathematical Programming Approach (working paper), University of Kansas, School of Business. A later version of the paper appeared in Computers and Industrial Engineering, Volume 49 Issue 1, Aug. 2005

FDIC1 (2017), Bank Failures in Brief, https://www.fdic.gov/bank/historical/bank/, accessed Jan. 8, 2017

FDIC2 (2017), Statistics at a Glance, https://www.fdic.gov/bank/statistical/stats/, accessed Jan. 8, 2017

FICO (2017), FICO (formerly the Fair Isaac Company), www.fico.com, accessed Jan. 10, 2017

FRBNY (2016), SCE Credit Access Survey (October 2016), Center for Microeconomic Data, Federal Reserve Bank of New York,
https://www.newyorkfed.org/microeconomics/sceIndex/credit-access.html#indicators/overall-credit-rates-t1-a/g18, accessed Jan. 8, 2017

Goel (2016), Goel, Vindu, Russian Cyberforgers Steal Millions a Day With Fake Sites, New York Times Online, Dec. 20, 2016, http://www.nytimes.com/2016/12/20/technology/forgers-use-fake-web-users-to-steal-real-ad-revenue.html

Harris (2016), Harris, Matt, The Short History and Long Future of the Online Lending Industry, Forbes Valley Voices, www.forbes.com, accessed Jan. 8, 2017

Huang (2014), Huang, Jun, Feature Selection in Credit Scoring—A Quadratic Programming Approach Solving with

Bisection Method Based on Tabu Search, PhD Dissertation by Jun Huang at Texas A&M International University

Lichman (2013), Lichman, M., UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

OECD1 (2016), Household debt (indicator). doi: 10.1787/f03b6469-en, accessed Dec. 21, 2016)

OECD2 (2016), Household disposable income (indicator). doi: 10.1787/dd50eddd-en, accessed Dec. 21, 2016)

Pedregosa (2011), Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825–2830, 2011

Powers (2011), Powers, D.M.W., Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation, Journal of Machine Learning Technologies, ISSN: 2229-3981 & ISSN: 2229-399X, Volume 2, Issue 1, 2011, pp. 37–63, available online at http://www.bioinfo.in/contents.php?id=51, accessed Jan. 11, 2017

Rao (2016), Rao, Malleswara, How to Evaluate Bank Credit Risk Prediction Accuracy based on SVM and Decision Tree Models, Capgemini "Capping IT Off" Blog, Nov. 2, 2016, https://www.capgemini.com/blog/capping-it-off/2016/11/how-to-evaluate-bank-credit-risk-prediction-accuracy-based-on-svm-and/, accessed Jan. 18, 2017

Rosenberg (2007), Rosenberg, Andrew and Hirschberg, Julia, V-Measure: A conditional entropy-based external cluster evaluation measure, Department of Computer Science, Columbia University, New York, 2007, from http://www1.cs.columbia.edu/ amaxwell/pubs/v_measure-emnlp07.pdf, accessed Jan. 18, 2017

scikit (2017), http://scikit-learn.org/, accessed Jan 12, 2017. Also Scikit-learn: Machine Learning in Python, Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre et al., JMLR 12, pp. 2825–2830, 2011 (citation requested on publisher's website)

sklearn cv (2017), 3.1. Cross-validation: evaluating estimator performance, at http://scikit-learn.org/stable/modules/cross_validation.html, accessed Jan. 12, 2017

UCI MLR (2016) University of California, Irvine. Machine Learning Repository, http://archive.ics.uci.edu/ml/, accessed Jan. 8, 2017

US FED (2016), Board of Governors of the Federal Reserve System (US), Households and Nonprofit Organizations; Credit Market Instruments; Liability, Level [CMDEBT], retrieved from FRED, Federal Reserve Bank of St. Louis, https://fred.stlouisfed.org/series/CMDEBT, Dec. 21, 2016

Waad (2016), Waad Bouaguel Ben N'Cir, On Feature Selection Methods for Credit Scoring, PhD Dissertation, Université de Tunis, Institut Supérieur de Gestion, Ecole Doctorale Sciences de Gestion LARODEC, Jan. 2015

World Bank (2016), World Bank, Bank Non-Performing Loans to Gross Loans for United States [DDSI02USA156NWDB], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/DDSI02USA156NWDB, Dec. 21, 2016

World Bank (2017), International Debt Statistics 2017. Washington, DC: World Bank. doi: 10.1596/978-1-4648-0994-1. License: Creative Commons Attribution CC BY 3.0 IGO