university of groningen

faculty of mathematics and natural sciences

Comparison of feature selection techniques in real and synthetic data

Bachelor's thesis

June 2015

Student: Folmer Heikamp

Primary supervisor: Prof. dr. Alexandru C. Telea

Secondary supervisor: Paulo Eduardo Rauber

Abstract

Feature selection is a process used for selecting a subset of features from the feature space of a dataset, according to some criteria. The main goals of feature selection are creating simpler and/or better models and getting insights about the data. The problem is that the accuracy of different feature selection techniques might be dependent on the classifier or the dataset used. In this thesis different feature selection techniques are compared with each other, each feature selection technique is tested with a number of different classifiers and datasets, which either may be real or synthetic data. By providing these insights, we support practitioners in machine learning and classification with a better understanding of the relative advantages and challenges of several feature selection methods, and thereby arguably help them in the process of classifier design.

Contents

List of Figures

Nomenclature

Observation An observation is a data instance represented by an n-dimensional vector.

n Number of observations

m Number of features

X Matrix of dimension n x m containing all known observations. Each column represents a feature and each row represents an observation

X_i n-dimensional vector representing the i'th observation of X

X^f m-dimensional vector representing all the values for feature f in X

y n-dimensional vector which defines the class for each observation in X

C Set of all classes

1 Introduction

In machine learning classification is the problem of assigning a class to an observation on the basis of a model which is described by some parameters. These parameters are optimized by a training set which contains data for which the class is known. A typical example of classification is classifying a new image of skin where the classes are "skin cancer" and "no skin cancer". The image is assigned to one of the classes based on the model which in turn uses the training set. This example also demonstrates the importance of classification. If a system could make accurate predictions about whether or not the skin in the image contains skin cancer it could save lives.

Data is often represented by an n-dimensional vector $\mathbf{x}$ where each element x_i of $\mathbf{x}$ represents a feature. A feature describes a property of the data and is usually represented by a number. For example a color is usually described by a vector with three integers between 0 and 255, where the integers describe the red, green and blue coefficient respectively. Each coefficient is a feature, and the three features together describe the data. In this example each feature is relevant, because all features are needed to correctly specify the color, but in some cases there might be features which are not relevant or are redundant. Feature selection aims to eliminate such features to make the model simpler and maybe better. More formally: Feature selection is a process which selects a subset k from the features available where k contains the best features according to some criteria, usually relevance or usability.

The goal of this thesis is to compare several feature selection techniques with each other, to see how well each feature selection technique performs in comparison with the other techniques and how robust the results of the features selection techniques are. The feature selection techniques will consist of a baseline method, fast filter methods, computationally more expensive wrapper methods and experimental methods. The baseline method should be the worst method and is used for comparison. The experimental methods are usually not used for the task of feature selection but contain a rationale which makes sense for feature selection. So they might be good feature selection techniques. More interestingly are the results of the filter and wrappers methods. A method with slightly less accuracy but with a faster execution time might be preferable to a slower more accurate method.

The comparison is done between several different classifiers and datasets, to make the results more robust/meaningful.

Section 2 describes machine learning, supervised learning and classification. In section 3 feature selection will be explained in more detail. Section 4 describes the experiments performed for this project. In section 5 the implementation will be explained. In section 6 the project will be discussed and in section 7 this report will be concluded.

2 Classification

Classification is a machine learning problem, so before explaining classification, machine learning is introduced briefly.

According to Alpaydin [10] the definition for machine learning is: "Machine learning is programming computers to optimize a performance criterion using example data or past experience". Another famous definition of machine learning by Mitchell [24] is: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". The main problems machine learning tries to solve is predicting future data and gaining knowledge about the data [10]. The basic algorithm for machine learning is selecting a model with some parameters and optimize the parameters with respect to the training data. The outcome associated to the observations can be known or unknown. This is called supervised and unsupervised learning respectively. A typical example of machine learning is spam detection.

Classification is a supervised learning problem which has to do with assigning a class to a new observation. An observation is a representation of a single data entry. In this case an observation is an n-dimensional vector where each element in the vector represents a feature. Let C be the set of all classes and let $X = \{(x^t, y^t)|x \in \mathbb{R}^n, y \in C\}_{t=0}^n$ be the training set which contains all known data. The training set contains observations x^t with their corresponding outcome y^t. Classification is the process of assigning a class $c \in C$ to a new observation o, where the observation is evaluated by a model $g(x|\theta)$. The model $g(.)$ is chosen from a hypothesis class H which is a set of models where each model has different parameter values. Because the parameters θ of $g(.)$ are unknown at the beginning the best model from H has to be selected. Finding the model which describes the data best three main choices have to be made [10].

1. Choosing a hypothesis class H, which limits the number of possible classifiers.

2. A loss function L to measure the difference between the known output y and the predicted outcome $g(x|\theta)$.

3. An optimization procedure to find the optimal parameters for a model, e.g. $\theta^* = \arg\min_{\theta} \sum_t L(y^t, g(x^t|\theta))$.

Different classifiers have either a different hypothesis class or differ on one of these functions. In classification a model is also called a discriminant function $d(x) = y, y \in C$, because it discriminates between classes. Classification is important in a lot of cases, skin cancer detection is one of the many examples.

2.1 Classifiers

As our focus is on feature selection, rather than classifier evaluation, we wanted to base our work on a set of well-proven classifiers, as studied and described in the literature. For this purpose, we selected a number of well-known classifiers in the machine learning community. These are described next.

2.1.1 K-nearest Neighbours

A k-nearest neighbours classifier, abbreviated knn, classifies a new observation o according to the k closest neighbours of o. Let N_o be the set of k-nearest neighbours for o, then the probability of o belonging to class C_i is set to $P(C_i|o) = \frac{count(C_i|N_o)}{k}$, which is the ratio of the number of observations in N against k. The discriminant function $d = C_i \ such \ that \ P(C_i|N) = \max(P(C_j|N)), \ j \in [0..p]$, where p is the number of classes, assigns the most frequent class in N to the new observation o. The nearest neighbours are determined by a distance function. The most common function is the Euclidian distance function $d(x, y) = \sqrt{\sum_{i=0}^{m}(x_i - y_i)^2}$, but other distance functions can be used. The rationale behind this method is that observations of the same class should be close to each other. Knn is included because it is a widely-used, simple and intuitive way of classifying new observations.

2.1.2 Random Forest

A decision tree is a classifier which makes use of the tree data structure. Each node in the tree represents a rule and each leaf represents a class. A rule splits up the data into multiple directions and may invoke several features. For example if a feature f for a new observation o has the value 1 and the rule for a node is $f > 2$, then o travels to the next node b via the edge with the label "false". Methods for creating decision trees from training data are for example C4.5 [25] and CART [28]. Decision trees are a popular classifier because they don't make assumptions about the distribution of the data it tries to predict. A big disadvantage of a decision tree is the high variance, which means that small changes in the data might change the model significantly which makes the model less robust and prone to overfitting. A possible solution to this is tree bagging, which introduces a random factor by re-sampling the data k times, selecting only a certain percentage of the training-data each time, and creating a decision tree for each sample [11]. The final class outcome is calculated by taking a majority vote over all created decision trees. Research shows that bagging keeps the same bias, but decreases variance [11].

The random forest uses the same process as bagging, but it differs in one aspect. When the tree has to be split, most algorithms try to select a rule which minimizes the error taking into account all features. With a random forest at each split a random subset of features is chosen and from that set the best rule is chosen. This is done to minimize correlation between individual decision trees [12] and thus reduces overfitting. Research has shown that random forest outperform bagging and decision trees in general [18]. The random forest classifier is available in scikit-learn [7] and is specified by several parameters such as the number of trees to use and the percentage of observations to be selected in each sample. The working of the random forest classifier is explained in detail by Leo Breiman [12]. It is included as a classifier because random forests is a well-known and robust classifier.

2.1.3 Support Vector Machines

Support Vector Machines (usually abbreviated as SVM) are well known in classification because they have computational advantages over other methods [21]. An SVM specifies a linear decision boundary, dividing the space into two half-spaces. Observations are

classified according to their position of the decision boundary. An SVM is trained on all the data but the resulting model only uses so-called support vectors for classification [21]. A support vector is a known observation which is close to the decision boundary. An SVM trains the weights of a decision function, which, in the end, determines the class of the data. The data is not always linearly seperable, that's why SVMs make use of kernel functions, a kernel function transforms the data into another space which might be better linearly seperable. Thus SVMs operate on the transformed data. There are several kernel functions possible, we use the linear kernel and a radial based kernel.

2.1.4 Dummy

As a sanity check for our experiments, we introduce a dummy classifier. Dummy assigns new data to the most frequent class, so the probability of a new observation o belonging to class C_i is $P(C_i \mid X) = \frac{count(C_i \mid X)}{n}$. Let k be the number of classes then the decision function $d(o) = C_i \ if \ P(C_i \mid X) = max(P(C_j \mid X)), \ j \in [0..k]$ assigns o to the most frequent class. The dummy classifier is used as a baseline classifier, the other classifiers should never be worse then the dummy. It can also be used for verification of the implementation since this method has the property that it always has a ROC-AUC score of 0.5. The working of the ROC-AUC score will be explained in section 4.1 .

3 Feature selection

As mentioned in the introduction, data is often represented by features. The number of features used is typically large, think of a few hundred/thousand features [20]. A feature selection technique selects a subset with the best features according to some criteria. It is possible that two or more features are related so that they partially measure the same quantity. If the goal is to select the best features regardless whether the best features are related to each other, then the criteria is relevance. If the goal is to select the features which optimize a model, then the criteria is usefulness. Although other criteria are possible these two are the most used [20]. Note that the most relevant features are not always optimal for a model, overlapping features may cancel each other or even decrease the accuracy of the model [20].

All the feature selection techniques used in this project give each feature a score, which is why they are reffered to as scorers. By doing so we can iterate over the best features by changing one parameter i where i defines how many of the best features are to be selected. This makes the experiments easier since there is only one variable which has to be changed each iteration. It is also used in the work of Isabelle Guyon [21]. Note that this is not the only possible solution.

3.1 Importance

The most obvious reason why feature selection is important is the dimensionality reduction which leads to a reduction in storage needed, processing time and simplifies the model by making it dependent on less features. A second reason is that a model with less features might describe the data equally well or even better then with all features, since a simpler model reduces overfitting [21]. For example the article by Rauber et al [26] shows that for a specific dataset the number of features can be reduced to five percent of the total number of features whilst keeping the same prediction accuracy. The third reason is that feature selection might give relevant insights about the data [20, 23, 10]. For example in the skin cancer case it is useful to know if the color of the skin is good indication for skin cancer. These insights can be used to create potentially better classifiers.

3.2 Feature selection categories

The techniques used in this thesis are divided into two main categories.

Filters A filter scores the features according to a scoring function $S(f)$, where f is a feature, based on the input X and output y. Filters are a preprocessing step and are completely independent of the classifier. The advantage of filters is that they are usually faster than wrappers and are more robust against overfitting because it is independent of a classifier [20]. The filters used in the scope of our project are chi-squared [30], one-way ANOVA [30], variance based coherence, random and silhouette coefficients [9], all of them will be explained in more detail later on.

Wrappers Wrappers use a classifier as a black box to score the features [20]. So they optimize the features according to their predictive power for the selected classifier [23]. The wrappers used in this project are recursive feature elimination [21], randomized decision trees [18] and randomized logistic regression [8].

Hybrid Hybrids combine filters and wrappers, it uses a filter for selecting potential models and the wrapper to verify it [23]. For this project no hybrid methods were used. The reason for this is that we wanted to study the methods individually. It is up to the reader to combine methods, if it appears interesting.

3.3 Scorers

The scorers are split up into three groups based on their category and their purpose.

The first group contains a random scorer, which is used as a baseline, and two experimental methods, which are not often used for feature selection.

Random The random scorer gives each feature a score between zero and one randomly. It is used as a baseline method. The quality of other scorers can be compared with respect to the random scorer. If a scorer scores worse than the random scorer it is very probable that scorer selects bad features instead of good ones.

Silhouette Coefficient The silhouette coefficient makes use of clusters. A cluster is a subset of observations which are similar to each other in feature space. The silhouette coefficient assumes cluster compactness is a good way to discriminate between classes. It is calculated by the formula $\frac{b-a}{max(a,b)}$, where a is the mean intra-cluster distance and b the mean nearest-cluster distance [9]. The Euclidian distance is used by default but this can be changed. The score for a feature f is calculated by the formula $\frac{\sum_{i=1}^{n} \frac{b_i - a_i}{max(a_i, b_i)}}{n}$, where a_i is the mean distance to observations of the same class and b_i is the smallest mean distance to observations of another class. The score is calculated for each feature f using all observations X^f for f. The division by the maximum normalizes the score to a value in the range $[-1..1]$, 1 being the best score and -1 the worst. Positive values indicate that the inner-cluster distance is smaller than the nearest-cluster distance, negative values indicate the opposite. The rationale behind this method is that the features with the best score are the most isolated from the rest so if a new observation is close to this cluster it is very probable it has the same class. In the code the silhouette coefficient is calculated for each feature and the end result is normalized so it contains values between zero and one. This method is not often used as a feature selection technique. It has been proposed for instance selection and is used for evaluating clustering algorithms [13]. It is expected that this method is not very good in selecting good features because the data for a given class don't have to be in a compact cluster. It is included because it is a simple method and had some success in instance selection.

Variance Based Coherence The variance based scorer scores the features based on their variance by calculating the variance for a feature for all observations $var(p+n)$ and calculating the variance for only observations of the positive class $var(p)$ and then taking of the ratio of the two: $\frac{var(p)}{var(p+n)}$, the highest score meaning the worst feature. In the case of divison by zero the score for that feature is the same as the worst score. The rationale behind this is that without variance you have no way of knowing to which class a new observation belongs so the feature cannot be worse. After this step the values are normalized and negated, so the best feature has the highest score. The idea behind this scorer is that if the variance in the positive class is significantly smaller than the total variance for the feature, the feature can

be used to discriminate between the classes. If the variance of the positive class is higher or almost equal it gets harder to distinguish between classes. The scorer is included because of its simple and intuitive way of selecting features, although it is expected that it does not work well all the times since there are examples where this scorer selects bad features. For example, if there is a feature f which should have zero variance but due to an outlier the variance in the positive class is zero and the total variance is 0.00001 than the score for this feature is very good because $1 - \frac{0}{0.00001} = 1$. This of course is not true and this feature should not even be considered as one of the best.

The second group contains filter scorers. This group contains widely-used feature selection techniques which do not use a classifier to score the features.

Chi-squared Chi-squared is a filter method that uses the chi-squared statistic. Specifically the Pearson chi-squared statistic is used. The statistic measures goodness of fit, the likelihood of the variable belonging to some distribution and the independence with respect to another variable. The Pearson chi-squared test is defined as $\sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$ [30], where O_i is the observed value, E_i is the expected value and n the number of observations. This statistic is used for feature selection in the following way [1]:

1. y is converted to a binary class problem if it wasn't already, 1 being the positive class and 0 the negative.
2. The observed value O for a feature f is calculated by taking the sum over all feature values of f for the postive class.
3. The class probability is calculated by $P = \mu(y)$. This is correct because y is binarized.
4. The expected value E is calculated by the formula $E = P \sum f$, where $\sum f$ is the sum over all feature values for feature f.

In the scope of our project the statistic is used to calculate the dependence between the class variable and the selected feature. High scores indicate dependencies so features with a high score are probably more important.

Now the expected and observed values are calculated the chi squared metric can be calculated by simply applying O and E to the formula. Note that the summation in the formula is implemented with dot products, so to calculate the final score the summations in the formula have to be ignored. Chi-squared is chosen as a scorer because it is a well known method for measuring dependence between variables and it is also a fast filter method, so it would be nice to compare this with the more advanced methods. Chi-squared scores features independently, any dependence between features will not be detected by this scorer.

One-way ANOVA The one-way ANOVA statistic tests the null hypothesis that the means of all groups are equal. One-way ANOVA can only work with numeric features and makes several assumptions about the data, e.g. it is normally distributed, the samples are independent and all the groups have the same standard deviation [30, 5]. Let $\mathbf{z}^p, p \in \mathbb{N}$ be the vector containing all feature values of feature f for class p. The hypothesis which one-way ANOVA tests then becomes $\mu(\mathbf{z}^1) = \mu(\mathbf{z}^2) = ... =$

$\mu(\mathbf{z}^p)$. Note that we only talk about one feature in this section. In our experiments these calculations have to be executed for each feature. The inter class variance is calculated by the formula $\sum_{j=1}^{p}\sum_{i=1}^{n_j}(z_i^j - \bar{\mathbf{z}^j})^2$, where n_j denotes the number of values for class j, p is the number of classes, z_i^j is the i'th element of $\mathbf{z}^j$ and $\bar{\mathbf{z}^j}$ is the mean of feature f for class j. The treatment is calculated by the formula $\sum_{j=1}^{p} n_j(\bar{z^j} - \bar{\mathbf{z}})^2$, where $\bar{\mathbf{z}}$ is the mean value of feature f regardless the class. The score for feature f is calculated by summing the outcomes of the two formulas together. In our experiments j will always be 2 because we only work with binary classification problems. The features with the highest scores are the best scores according to the one-way ANOVA scorer. The rationale behind this is that a big difference in means between classes for the same feature can be used to distinguish between classes. The higher the score the bigger the difference in means so the easier it becomes to distinguish between classes. Like the chi-squared scorer it does not take into account dependencies between features. This scorer is included because it makes the filter methods equally represented.

The last group contains wrapper scorers. Wrapper scorers score features with help of a classifier and are widely used for feature selection.

Recursive Feature Elimination Recursive feature elimination is a wrapper method which makes use of a classifier. It consists of three basic steps [21].

1. Train the classifier.
2. Compute the ranking for each feature, based on the classifier.
3. Remove feature with the highest rank.

This process can be repeated and more than one feature may be selected each iteration. In our case we use the combination of Results for the Random scorer with the linear SVM as suggested in the article by Guyon [21]. We are only interested in the ranking. The ranking of features is based on the weights they receive when training the SVM [21]. A higher weight means that the feature is more relevant. Recursive feature elimination is added because it seems a intuitive way to select features, it is also different from the other wrapper methods which make use of randomization.

Randomized Decision Trees Randomized decision trees use extremely randomized trees as classifier to find the best features. Extremely randomized trees is an extension of the random forest classifier with the difference that extremely randomized trees use the entire training set multiple times and it chooses a random rule at each split. It has a fast execution time and delivers good results [18]. The randomized decision trees classifier works by generating k (partially) random decision trees [18]. New observations are classified based on a majority vote over all k random decision trees. The main rationale behind the random element is that it tries to decrease bias and variance. Randomized decision trees can be used for ranking features, although research shows that it does not belong to the top scorers [19]. The features are scored using the gini impurity measure according to [3]. In the random forest classifier randomization leads to less variance while keeping the same bias. It seems

to apply in general that randomization makes the results more robust. That is why this method is included, because it is a randomized method and because it is the only scorer to use a tree classifier.

Randomized Logistic Regression In this method a logistic regression classifier is used to train a model. The logistic regression classifier is defined as $g(x) = \frac{1}{1+e^{-t}}$ where $t = \sum_{i=1}^{n} a_i x_i + a_0$. The weights $\mathbf{a}$ are optimized with respect to the data X [22]. This is done by selecting the weights which minimize the error for the training set. The function returns for an observation o a value between zero and one. o Is classified as "positive" if $g(o) > 0.5$ and classified as "negative" otherwise. The training of the classifier can be done in several ways and is outside of the scope of this project. Randomized logistic regression works by re-sampling the input data k times, each time selecting only a fraction r of the training set, and train the logistic function only using r [8]. The weights for each feature are summed up for each iteration and the features are ranked according to their summed weight, a high weight indicates a good feature [8]. Note that this is not the only way to score a feature, squaring the weights or taking the absolute value of the weights before summing up are also possibilities. It is added to this project because it uses a completely different classifier.

4 Experiments

The goal of the experiments is to compare the performance of feature selection techniques with each other and the baseline scoring technique. The feature selection techniques are used in combination with multiple datasets and classifiers to make the results more robust and meaningful. The prediction accuracy is used as a metric to score the feature selection techniques. Particularly interesting is the difference between faster filter and slower wrapper methods. It is also interesting to see how well the experimental scorers perform.

4.1 Scoring

A scoring method is used for measuring the performance or accuracy of a classifier. There are multiple ways of scoring a classifier. Intuitive and simple ones are the error and success ratio. But in some cases these are not good enough. For example when there are two classes and the prior probability of class 1 is 0.99 then a classifier using a simple majority vote classifies the data with a success rate of 0.99 or equivalently an error rate of 0.01. The scores indicate a very good classifier whilst this is not the case. To prevent this problem a different and more robust scoring method is used. The scoring method used in the experiments is called the "receiver operating characteristic area under the curve", abbreviated "ROC-AUC". The following definitions are used for calculating the ROC-AUC score [10].

True positive, TP: A true positive occurs when the classifier predicted correctly that the observation belonged to the positive class.

True negative, TN: A true negative occurs when the classifier predicted correctly that the observation belonged to the negative class.

False positive, FP: A false positive occurs when the classifier predicted incorrectly that the observation belonged to the positive class. This type of error is known as a type I error.

False negative, FN: A false negative occurs when the classifier predicted incorrectly that the observation belonged to the negative class. This type of error is known as a type II error.

Total positive, ToP: $ToP = TP + FN$.

Total negative, ToN: $ToN = TF + FP$.

True positive rate, TPR: Ratio of correctly predicted positives against ToP, $TPR = TP/ToP$. Best score is 1, everything is predicted correctly, note that this is also the case when you always predict positive. Worst score is 0 everything is predicted incorrectly.

False postive rate, FPR: Ratio of incorrectly predicted positives against ToN, $FPR = FP/ToN$.

Threshold, T: The threshold determines the class the data belongs to if $data > T$ it belongs to positive otherwise it belongs to negative.

For different values of T the TPR and FPR are calculated, these values are than placed inside the plot. In the end the plot contains several data-points and the resulting curve is obtained by drawing a line through all these data-points. The final score is the area under the curve and because the axes both have the range [0..1] so the score is also between zero and one. One denoting the best score and zero the worst. It is an elaborate scoring method but it is generally more robust. More information about the ROC-AUC score can be found in [10].

4.2 Protocol

The protocol has two main parts, the first part is about obtaining the relevant data, the second part is about converting that data into useful plots and statistics.

Each feature selection technique in this project works by scoring all features, so each technique has a ranking for the features. This makes it possible to select the k most important features according to a feature selection technique or scorer. Let I be a set with integers where $\forall_{i \in I}(i \in [1, 2, .., m-1, m])$. Each i defines with how many of the best features the experiment has to be evaluated. In the best case I contains all integers in the range $[1, 2, .., m-1, m]$ but this means that for each i all the classifiers have to be retrained which is time-consuming. To overcome this problem the number of features selected at each iteration is increased exponentially so if a dataset has 500 features, the experiment is evaluated with 1, 2, 4, 8, 16, 32, 64, 128, 256 and 500 of the best features respectively according to scorer f. Note that the experiment always includes one iteration using all the features because it can be used as a baseline score. The evaluation of the feature selection technique is based on the score of the underlying classifier because the aim of feature selection is to select the best features with as criteria to optimize the performance of the classifiers. So if for a classifier c and feature selection techniques a and b the resulting scores for c are 0.5 and 0.8 when selecting the two best features according to a and b respectively, than b is considered the better feature selection technique. This of course might change when selecting another number of features. To make results more robust several different classifiers are used. The classifiers are scored using the ROC-AUC score because it is more robust than other scoring methods. A classifier is defined by several parameters. A classifier with specific parameter settings is called an estimator. The experiment is executed multiple times for each classifier each time using a different estimator, the best score from those estimators will be the resulting score for the classifier. Let F be the set containing all scorers, then the end-result is that for each triple $(f, c, i), f \in F \; c \in C \; i \in I$ there is a corresponding score s. A ten-fold cross-validation is used when calculating the results, making the outcome more robust. The following pseudo-code will explain the experiments more clearly. Store is a command which writes data to disk. F is the set containing all scorers or feature selection techniques, C is the set containing all classifiers, I is the set containg the number of best features for which the experiment has to be executed and E is the set containing all estimators for c where $c \in C$.

Algorithm 1 Algorithm for obtaining relevant information

```
procedure EXECUTE(X, y)                                   ▷ executes experiment for a dataset
    for all f ∈ F do
        scores ← sorted list containing scores for each feature in X using f
        STORE(scores)                ▷ stores the feature scores with corresponding names
        for all i ∈ I do
            Xt ← training-set containing the data for the i best features for f
            yt ← classes corresponding to Xt
            for all c ∈ C do
                best_score ← 0
                for all e ∈ E do          ▷ E is the set of all estimators for the classifier c
                    TRAIN(e, Xt, yt)      ▷ Train model with training data and estimator
                    score ← ROC-AUC score for e using Xt and yt
                    if score > best_score then
                        best_score ← score
                    end if
                end for
                STORE(best_score, f, c, i)                ▷ store best score for (f, c, i)
            end for
        end for
    end for
end procedure
```

Note that the cross-validation is not included in the pseudo-code but implicitly assumed when returning the score for an estimator. To make the results more useful and robust this experiment is executed for multiple datasets, the datasets are described and evaluated in section 4.3 .

The second part of the experiment is about the generation of relevant statistics. There are three different types of statistics generated.

- A plot for each classifier. Shows the curves for each feature selection technique in the same figure. Each curve displays the number of features plotted against the corresponding ROC-AUC score. All known data-points are marked by colored dots or squares. The data-points are linearly interpolated to make the plots more readable. This plot can be used to see which feature selection technique worked the best for the classifier of the figure.

- A plot for each feature selection technique. Displays curves for each classifier in the same figure. Like the previous type of plot it plots the number of features against the ROC-AUC score, only in this case the feature selection technique does not change. This plot can be used to see if a feature selection technique yields good results for all classifiers.

- A table for each feature selection technique which contains all features with their corresponding ranking-score. The table is sorted from best to worst. The tables can be used to validate if different feature selection techniques select the same features.

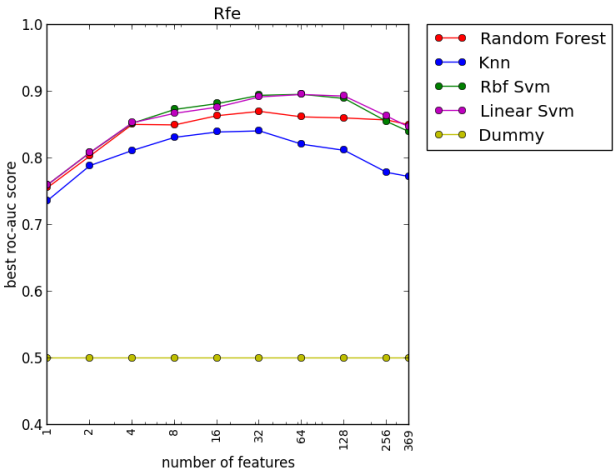
A subset of the plots and statistics are used in section 4.3 . A complete overview of all plots and statistics can be found in appendices A to G.

4.3 Analysis

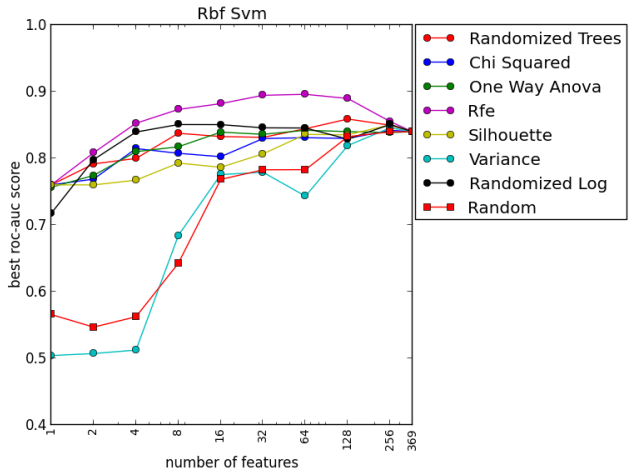
The experiments were executed on several datasets to make the results more meaningful and robust. Each dataset will be analyzed individually and this section will be concluded by a general conclusion which takes all datasets into account.

4.3.1 Melanoma

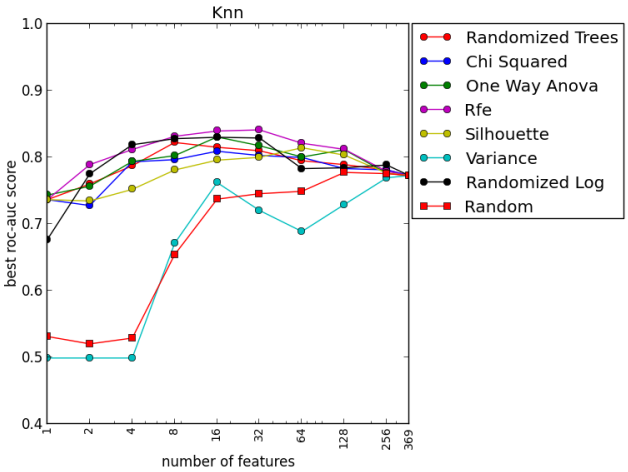
The melanoma dataset contains 753 images of skin. Each observation is specified by 369 features. Two of the most important types of features used in this dataset are features which describe color properties and features which describe boundaries in the image. A complete overview of all the features can be found in the master thesis by Feringa [17]. There are two classes "malignant skin lesion" and "benign skin lesion" which are represented by 268 and 485 observations respectively, so the dataset is in favor of the second class. The images are a part of the EDRA atlas of dermoscopy [14] and are provided by M. Emre Celebi.



(a) Results for the Recursive Feature Elimination scorer



(b) Results for the Rbf SVM classifier



(c) Results for the Knn classifier

Figure 1: Most relevant plots for the Melanoma dataset

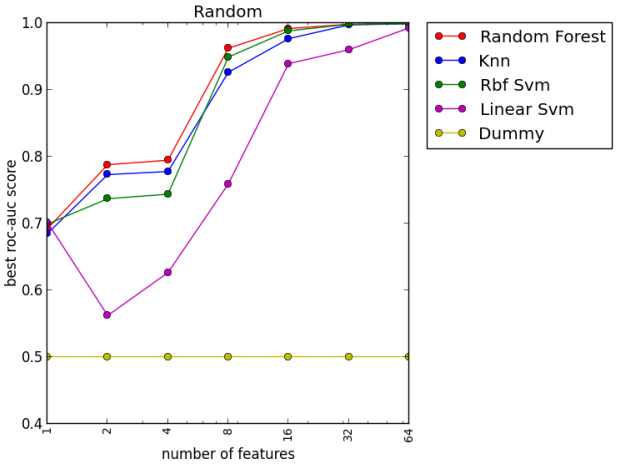
The variance based coherence scorer scored less than the random scorer, which functions as baseline, as can be seen in figure 1b and 1c. This indicates that the variance scorer selects bad features instead of good ones. There are examples where the variance scorer makes wrong assumptions about the data, for example if a feature has always the value 0 except in one case where due to an outlier the value is 0.01, this will lead to a local variance of 0 and a global variance of almost 0 this leads to the equation $\frac{0}{0.01} = 0$, so this feature is selected as the best feature possible, this of course is not true since there is no good way to distinguish the classes from each other. This or a related problem might explain why the results for the variance scorer are bad. The silhouette scorer, which was expected not to work very well, performed reasonably well. It was not one of the top scorers but it scored significantly better then the random baseline scorer, see figure 1b and 1c. The filters chi-squared and one-way anova performed as well as the wrapper methods with the exception of recursive feature elimination. It could create a model with less features while keeping the same accuracy. Figure 1a shows the recursive feature elimination scorer which has the best overall performance for all classifiers especially with the SVM classifiers it performs well. With an rbf SVM it scores about 0.05 better than the second-best scorer when selecting the 32 most relevant features. These observations strengthen the claim that a wrapper optimized with a classifier performs better when classified by the same classifier, because the recursive feature elimination scorer was trained using an SVM. Recursive feature elimination is also the only scorer which decreases significantly if all the features are used. Recursive feature elimination has some features in the top ten which no other scorer has in the top ten, namely "variance_g" and "inverseDi_om0_r" which might expain its superiority over other scorers. The randomized logistic regression scorer performed well, only with one feature were the results less, about 0.05, then the other methods, strangely the feature selected was "lab_std_b" and is also one of the top ranked features in the recursive feature elimination scorer, so it is likely that this feature only works well when used with another feature.

Of the classifiers knn had the worst performance, although the difference was only about 0.02. Knn was also the only classifier which increased in performance when selecting less features for other scorers then the recursive feature elimination scorer, as can be seen in figure 1c. The other classifiers performed about equal except when recursive feature elimination is used in combination with an SVM.

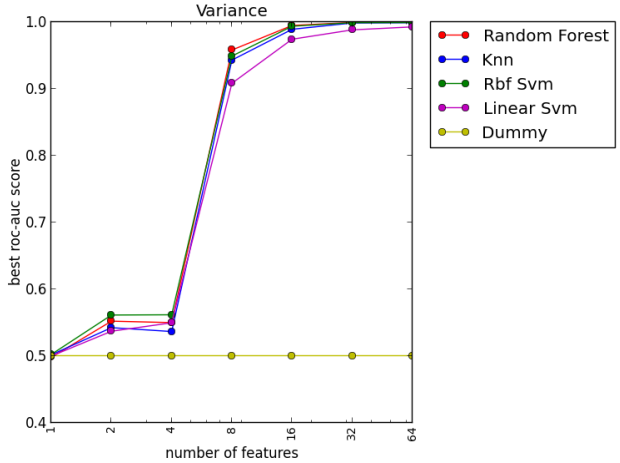
The results for this dataset validate the fact that more features do not always lead to a better model and that the performance can even increase when selecting less features. When selecting only one feature most scorers already achieve a reasonable score between 0.7 - 0.8 and with eight features most scorers receive a ROC-AUC score which is almost equal to the score when selecting all the features. This means that the number of features in those cases can be reduced by a factor $\frac{369}{8} \approx 46$, which leads to a much simpler and faster model. This dataset also indicates that wrappers optimized with a classifier tend to perform better when evaluated by the same classifier c. There are also some signs that some features only work well if combined with other features, for example in the case of randomized logistic regression. The results, using all the features, we obtained are slightly better the results in [26], this is probably due to the fact we use ten-fold cross-validation and they are using five-fold cross-validation. The results also match when selecting only the top five percent features only recursive feature elimination out-performs the scorer mentioned in [26].

4.3.2 Digits

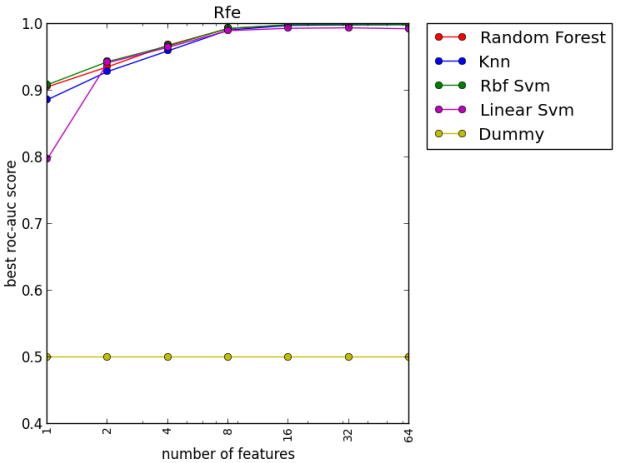
The digits dataset contains 1797 images of handwritten digits represented as a vector with 64 features. The data has 10 classes, one for each digit. In this dataset each pixel is a feature. Each class is almost equally represented, so in our case we used the default positive class which is the digit "1" in this case. The original data can be found in the machine learning repository [6]. This dataset is obtained from scikit-learn, which is a package for python. They transform this dataset into a dataset containing 1797 observations and 64 features, where each observation is a 8 by 8 image [2]. The detection of character is a very practical problem, that is why this dataset is included.



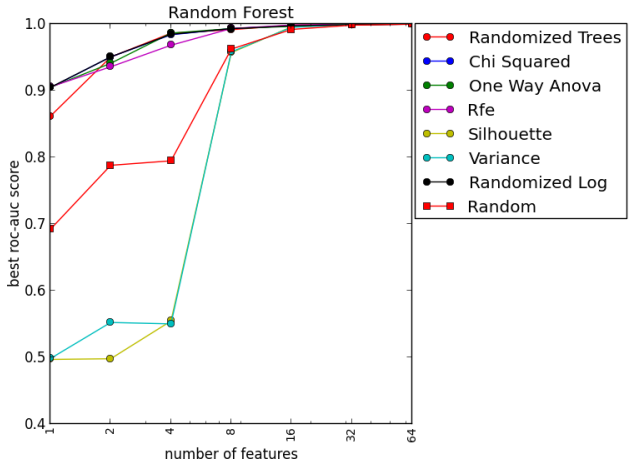
(a) Results for the random scorer



(b) Results for the variance scorer



(c) Results for the recursive feature elimination scorer



(d) Results for the Random Forest classifier

Figure 2: Most relevant plots for the Digits dataset

The variance and silhouette scorer perform very bad for this dataset, only when selecting eight or more features they come close to the random scorer, for less then eight features their score is approximately 0.2 lower, as can be seen in figure 2d and the comparison between figure 2a and 2b for the variance scorer. It is noteworthy that the top four of these scorers consist of the same features "f16", "f63", "f24" and "f55" and that these features are not in contained in the top ten for other scorers, indicating that the variance and

silhouette scorer make the same wrong assumption about the data. The experimental methods make a big leap going from four to eight features. This is probably due to the fact that the features with rank 5-8 for these scorers contains features which are in the top three for better scorers, for example the features "f19" and "f10". Only in the linear SVM the difference between random and the variance and silhouette scorer is lower although the random scorer is still better. The other scorers have more or less the same performance. Only the randomized trees scorer performs significantly less when selecting only one feature, as can be seen in figure 2d. The filter methods do not have any problems with competing against the other methods. The recursive feature elimination scorer reaches a score of almost 1 if eight or more features are selected, just like several other methods, but in the case of recursive feature elimination there is almost no variance between classifiers.

The random forest classifier performs very well (figure 2d), after eight features it has an accuracy of almost one, for all relevant scorers. The Knn classifier is almost as good as the Random Forest, it only has more variance and some scorers perform a bit worse. The SVMs are worse, especially the linear SVM, which indicates that the data is not entirely linearly seperable. The rbf SVM performs better but only if more than eight features are selected.

The results indicate that feature selection and classification can be independent, because the filter methods perform very well, as well as most wrappers. It also shows again that selecting less features does not lead to a decrease of accuracy, although in this case the accuracy does not increase but rather stays the same. It also shows that with only one feature the accuracy is about 0.9 for most scorers, so if a very high accuracy is not necessary, one can argue for only selecting one feature which will drastically increase the time performance. This dataset shows that the experimental scorers are not robust feature selection techniques.

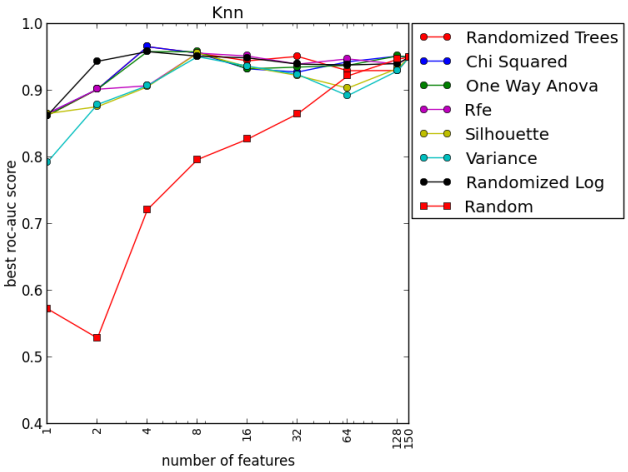
4.3.3 Corel

Corel is a dataset of 1000 images where each image is represented by 150 SIFT features according to [16]. It has ten equally represented classes with the labels: "Africa", "beach", "buildings", "buses", "dinosaurs", "elephants", "flowers", "horses", "mountains" and "food". No class has a majority so the default positive class "beach" is used.

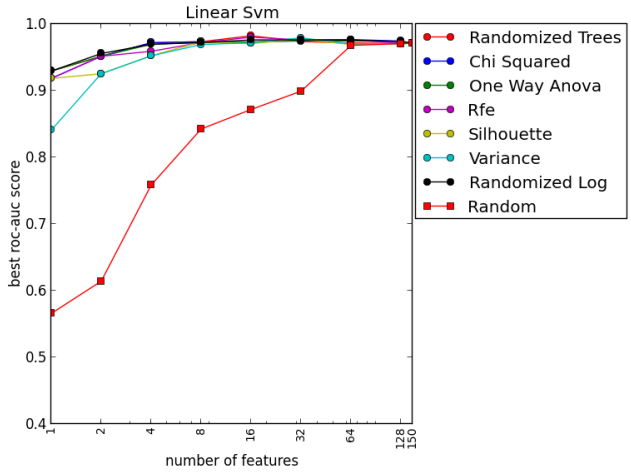
The experimental scorers are performing well when compared with the baseline scorer, although still worse than more serious scorers which can be seen in figure 3a and 3b and the comparison between figure 3c and 3d for the silhouette scorer. The filter and wrapper methods are almost indistinguishable which also can be seen in figures 3a and 3b, this is because they select almost the same features, especially the features "attr123", "attr143" and "attr145" are very significant for the accuracy because they are in the top three of four of the scorers and are also in the top-ten of the other scorers except the random scorer. It goes even so far that some methods have the same top ten features but in different order. For example, "chi-squared", "randomized-trees", "one-way-anova", "silhouette" and "randomized logistic regression" have the same top ten features.

The Knn classifier has a lot of variance compared to the other methods, whilst the linear SVM seems to predict the data better and is more robust.

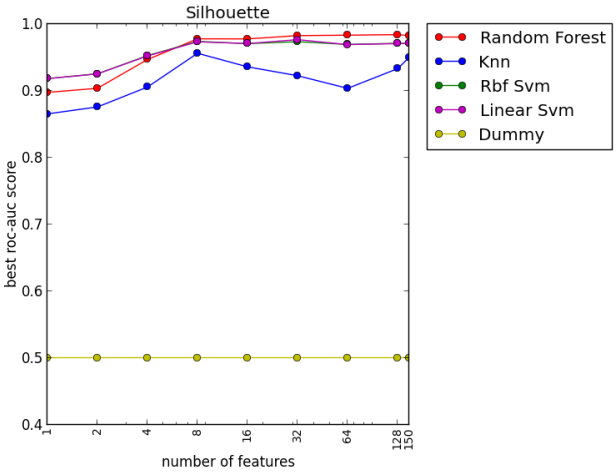
In this dataset we observe that many features do not add accuracy to a classifier. For most classifiers the accuracy is constant after four features, which is only $\frac{2}{75}$ of the total number of features. It also shows as in the previous datasets that filters are able



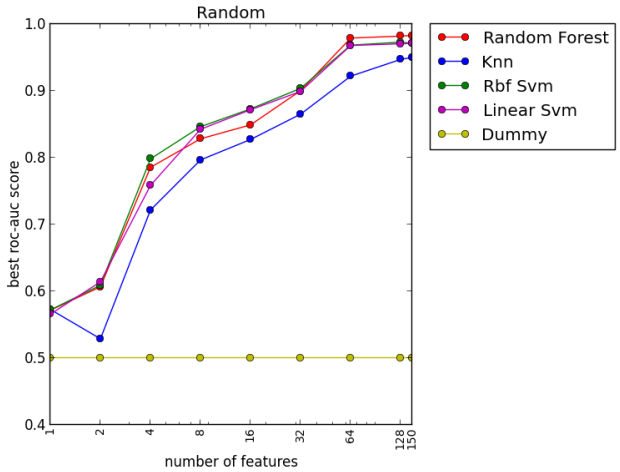
(a) Results for the Knn classifier



(b) Results for the Linear SVM classifier



(c) Results for the silhouette scorer



(d) Results for the random scorer

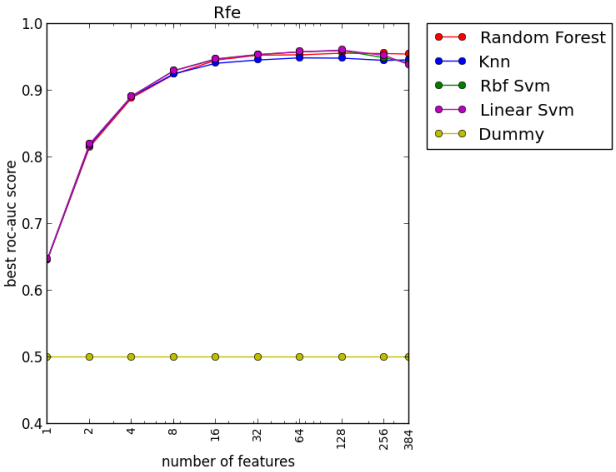
Figure 3: Most relevant plots for the Corel dataset

to perform as well as wrappers. Another fact obtained from these results is that the experimental scorers can be reasonable, but they are not at all robust, so practitioners should be careful when using them for feature selection.

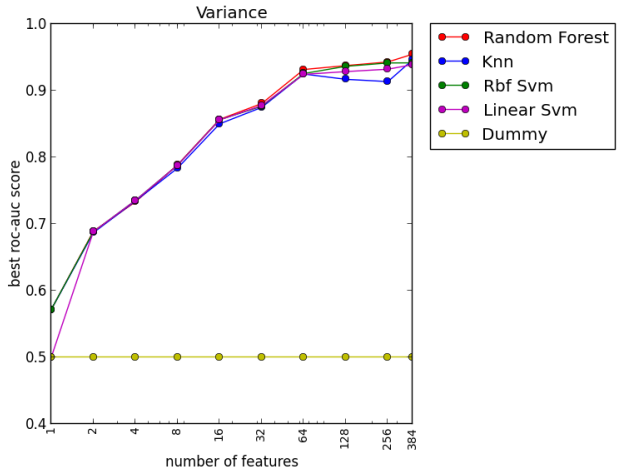
4.3.4 Rome

The dataset represents a satellite image of Rome. Each observation represents a superpixel. Superpixels are an alternative way of representing images in comparison with normal pixels. The image is represented by several groups, where each group contains neighbouring pixels which look like each other, such a group is called a superpixel. Rome has seven classes with labels: “road”, “tree”, “shadow”, “water”, “building”, “grass” and “soil”. Class five “building” is the most frequent, so it is chosen as the positive class [29].

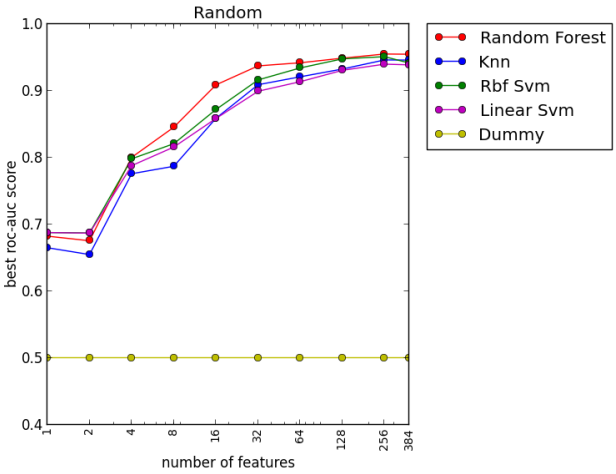
For this dataset there is a lot of variance between scorers, especially when few features are selected. The variance scorer scores bad again, as can be seen in figure 4d and the comparison between figure 4b and 4c. The other experimental method, the silhouette scorer, does better and can be considered the average scorer. The scorers “chi-squared”,



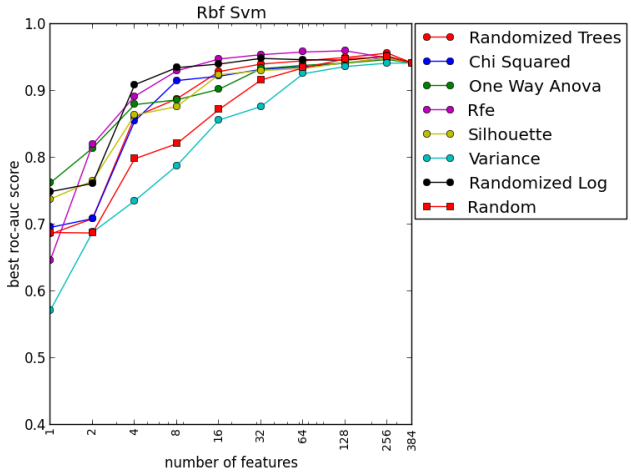
(a) Results for the rfe scorer



(b) Results for the variance scorer



(c) Results for the random scorer



(d) Results for the rbf SVM classifier

Figure 4: Most relevant plots for the Rome dataset

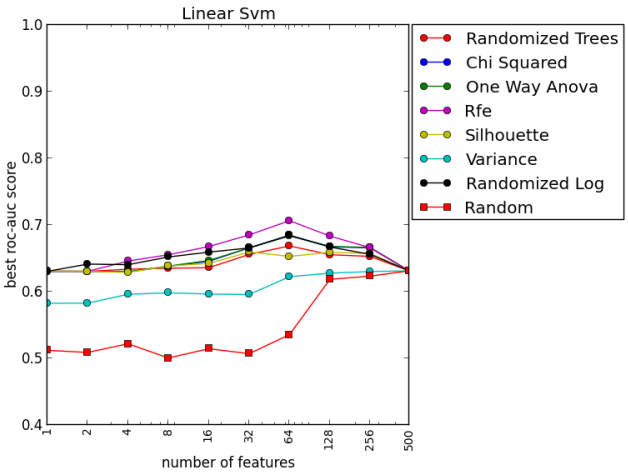
"randomized logistic regression", "variance scorer", "randomized trees" have a slow start, which means that for some reason the ranking order is wrong. The recursive feature elimination scorer is clearly the best scorer for this dataset it shows a nice almost logistic grow(figure 4a) and has almost no variance between classifiers, only when selecting one feature is the recursive feature elimination bad since it has a lower score than the random scorer. Strangely enough a lot of scorers with about equal performance do not always share the same top ten features. For example the top four features of recursive feature elimination "56", "213", "253" and "35" are not found in the top ten of the one-way anova scorer and they don't differ very much performance wise.

Figure 4a shows that recursive feature elimination increases the model a bit when selecting between 16 and 128 features, which validates the hypothesis that less features describe the data better or equally well. This dataset also shows that feature selection techniques are not perfect because a scorer might score well for a certain number of features but score relatively bad for another number of features, a intuitive solution is to use several scorers and use the sum of the ranks to rank the features, the feature with the lowest rank being the best feature. For example if features "f1" and "f2" have ranks 4 and 7

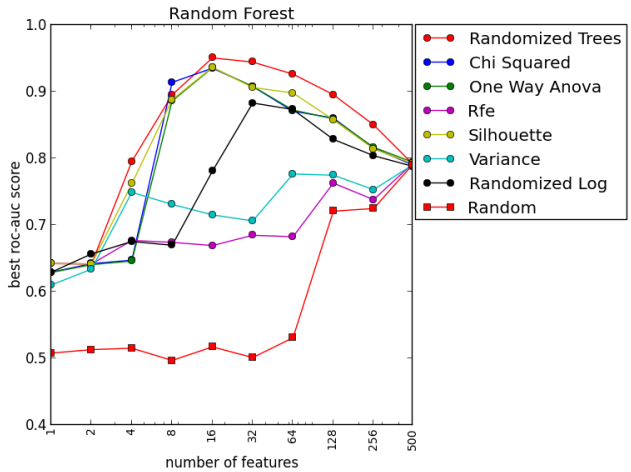
for scorer "a" and ranks 10 and 3 for scorer "b" than the final scores are $f1 = 4 + 10 = 14$ and $f2 = 7 + 3 = 10$ so "f2" is the best feature. This dataset also displays filters as reasonable scorers and the reasonable score for the silhouette scorer indicates that it can be used in some cases.

4.3.5 Madelon

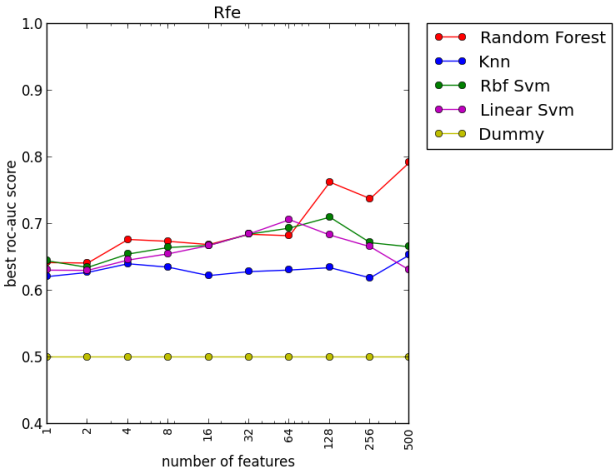
Madelon is an artificial dataset. The data consists of 32 clusters drawn in a 5-dimesional hypercube. Each data-entry is randomly assigned to a class, 1 or -1. In total, it contains 500 features where only 20 are useful the other are just noise. Of the 20 relevant features 15 are linear combinations of the others The dataset was created by Isabella Guyon and is available on the machine learning repository [4]. This dataset has two classes.



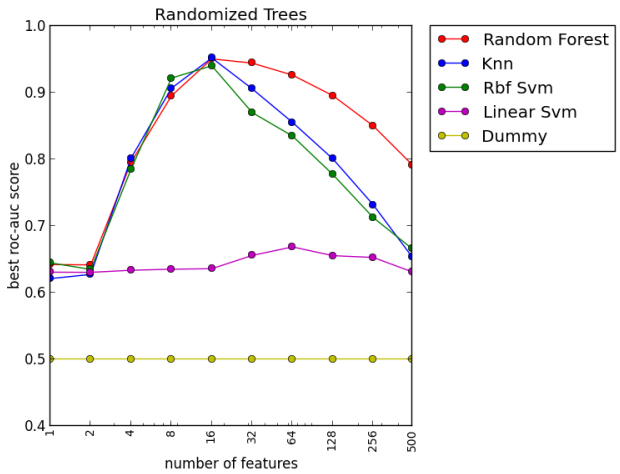
(a) Results for the Linear SVM classifier



(b) Results for the Random Forest classifier



(c) Results for the Recursive Feature Elimination scorer



(d) Results for the Randomized Trees scorer

Figure 5: Most relevant plots for the Madelon dataset

This dataset is very interesting because it is clear on first sight(see figure 6) that the data cannot be separated well linearly. Because of this, several methods will fail with respect to classification. For example the linear SVM has a terrible performance as can

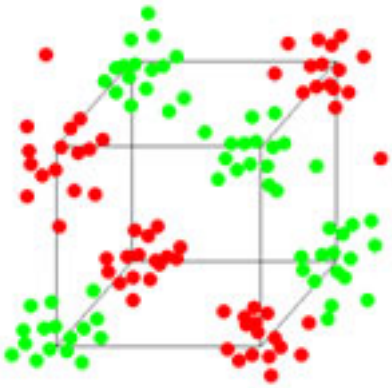


Figure 6: Overview of the madelon dataset [4]

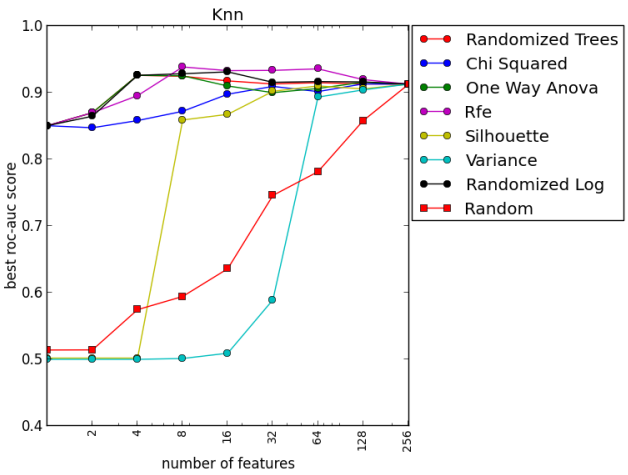
be seen in figure 5a. The recursive feature elimination scorer also obtains very bad results because it uses a linear SVM to select the best features. The variance scorer is better then the random scorer most of the time and has a peak when four features are selected, which indicate that it found some relevant features, however when selecting more then four features it doesn't find any more relevant features where other methods do so it is not a good scorer for this dataset. It also does not select the features "475" and "241" which are selected by other methods. The randomized trees scorer (figure 5d) does very well because it makes use of the Extremely Randomized Trees classifier, which is not linear. It increases drastically when selecting four or more features and decreases when selecting more then 16 features, this matches with the number of relevant features. The filter methods have also peaks between four and 16 features, although the graphs are not as smooth as the randomized trees scorer. The silhouette scorer does reasonably well again when compared to the baseline random scorer.

The linear SVM performs badly because the data cannot be separated linearly in a good way. The other classifiers have a better performance because non of them are linear.

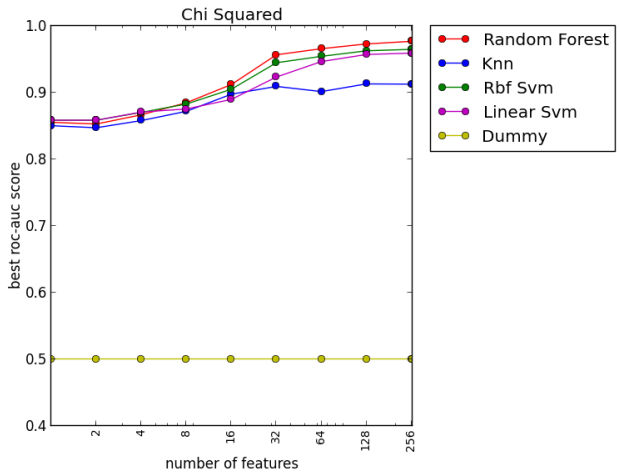
The results show that usually good classifiers can be very bad in certain cases that is why a platform helping people to select the correct classifier and feature selection technique is a good idea [15]. It also shows the big difference between different kernel functions for an SVM, the rbf SVM performs way better then his relative the linear SVM. This dataset also gives a big disadvantage for the use of wrappers. Because of the dependence between the classifier and the scorer, the scorer might get wrong results because the classifier does not work well, which can clearly be seen in the case of recursive feature elimination. It also strengthens the claim that filter methods are more robust. This dataset gives a very good example that a model can significantly increase when selecting less features, the difference being almost 0.2 for randomized trees in the random forest classifier when selecting only 16 features.

4.3.6 Parasites proto

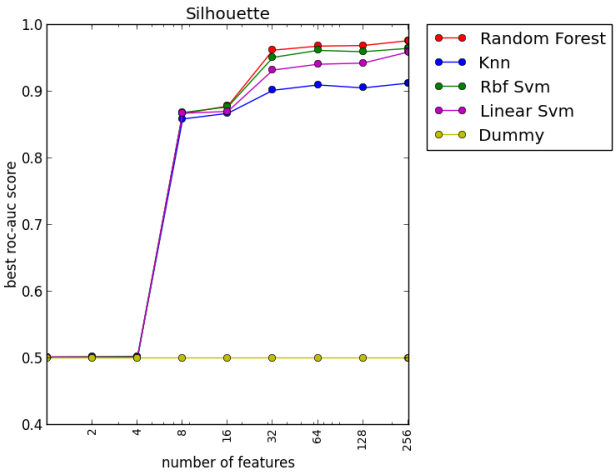
This dataset contains seven classes. Each class represents a type of protozoa parasite, except for class seven which are not protozoa parasites but impurities. Class seven contains the most examples and is also the class you want to distinguish from the rest, since the goal is most likely to recognize protozoa parasites, therefore class seven is set as the positive class and the other classes represent the negative class. More information about this dataset can be found in [27].



(a) Results for the Knn classifier



(b) Results for the chi squared scorer



(c) Results for the silhouette scorer

Figure 7: Most relevant plots for the Parasites dataset

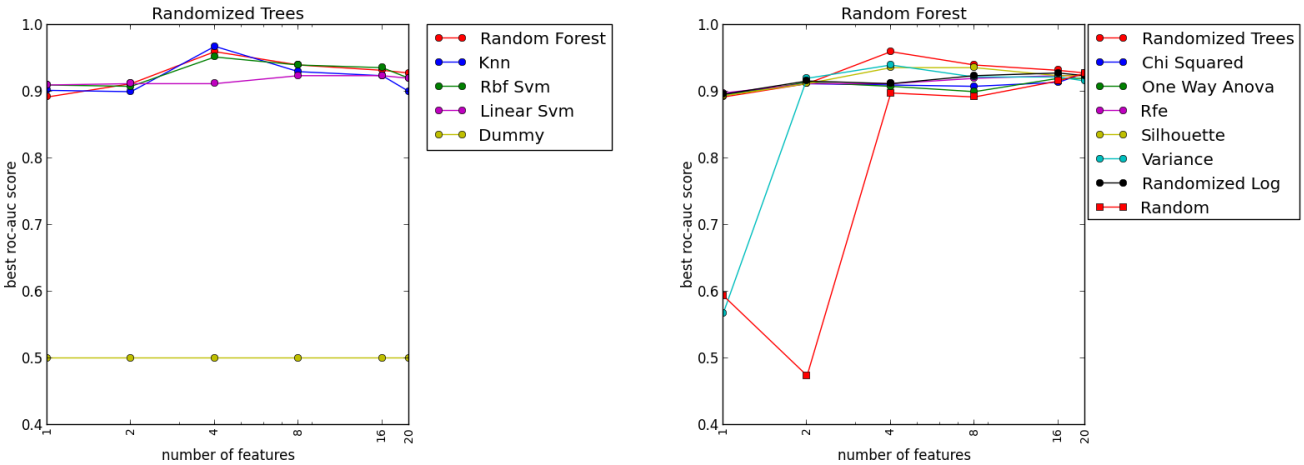
The experimental scorers do not select the best features right from the beginning. The silhouette scorer only performs well when eight or more features are selected (figure 7c) possibly because the feature "2" is selected, this feature is in the top-scoring methods the best scoring feature. It is very probable that the variance scorer also selects this feature when going from 32 to 64 features, this shows that it is not a good scorer. The chi-squared scorer performs worse than top-scoring methods between 1 and 32 features, although it is still way better then the random scorer, see figure 7b. The other filter method, the one-way anova scorer, performs better and is almost indistinguishable from the other wrapper scorers. The wrapper methods are close to each other as can be seen in figure 7a, which shows the classifier with the most variance for this dataset.

This dataset shows that feature selection techniques with almost the same performance also have a lot of features in common. It also verifies that filters can compete with other methods and once again indicates that a lot of features can be ommited from the model without losing accuracy and when only selecting one feature with a good scorer the accuracy decreases only with approximately 0.1 whilst the number of features decreased

by a factor 260 which should lead to a very fast and simple model.

4.3.7 Sksynth

Sksynth consists for the most part of synthetic data, only three features are meaningful. This data can be used for verification. There are two classes. This dataset was generated using a procedure very like the procedure used for the Madelon dataset.



(a) Results for the Randomized Trees

(b) Results for the Random Forest classifier

Figure 8: Most relevant plots for the Sksynth dataset

Most scorers have an accuracy of 0.9 or higher right from the beginning, only the random and variance scorer start slow. This means that most methods found a relevant feature immediatly and the variance and random scorer did not. The variance scorer selected feature "f9" as the most relevant, this does not comply with the other scorers which all selected "f6", "f3" and "f4" as the most relevant features, these features are the most relevant after "f9" for the variance scorer, which explains why the accuracy increases when selecting two or more features. Because the top three is the same for a majority of the scorers and because the scores differ significantly from the other scores, it is assumed "f6", "f3" and "f4" are the relevant features. The randomized trees scorer is the only one with a peak around four features, see figure 8a, the most likely explenation for this is that the fourth feature in this case "f19" does not influence the outcome and the fourth feature of other scorers do decrease the accuracy of the model. The random scorer performs when selecting four or more features, this can all be explained by probability $1-(17/20)^4 = 0.48$ which makes it very likely to happen.

This dataset shows once again that the variance scorer makes wrong assumptions about the data because the first ranked feature is just noise. It also indicates that noise does not necessarily decrease the accuracy, as can be seen in plot b.

4.3.8 Discussion of the results

Based on the observations of the different datasets we can analyze the overall results.

The dummy classifier always has a score of 0.5. This is a property of the ROC-AUC score, it is included to verify the plots, since score other than 0.5 would indicate that

something has gone wrong with the experiment. All the plots with de dummy classifier show a score of 0.5 which indicates that the experiment succeeded.

The variance scorer is usually not a good scorer, in some cases the results are really bad and when it performs better then the baseline score it is still worse than the other methods. Because of this it is not recommended to used this method for feature selection.

The other experimental method, **the silhouette scorer** has better results, but still should be used with uttermost care.

In the scope of our datasets **the randomized trees scorer** was the most robust scorer, it was among the best scorers for all instances.

The recursive features elimination scorer did also well, but like some other scorers it failed on the madelon dataset. This could probably be prevented by taking another kernel for the SVM than the one we used. The recursive features elimination scorer showed that in some datasets it was better than other scorers, so if a good classifier is chosen for the recursive features elimination scorer it is likely it performs very well or even better than other methods.

The general observation is that for each dataset there is at least one classifier which works well with the dataset. Filters describe the data generally very well. At least for the datasets used in this experiment this makes the choice for a filter as feature selection technique more logical because it is way faster and does not decrease the quality drastically. The other randomization method, randomized logistic regression performed well only with the madelon dataset was the performance not very good. Thus randomized methods seem to give more robust results. It is difficult to select the best feature selection technique since different methods make different assumptions and these assumptions might not always hold for different datasets. If the ultimate goal is to select a robust method then it is probably best to select a randomized wrapper method or a filter method. If the goal is to get the best performance it is best to select the recursive feature elimination scorer with an appropriate classifier.

5 Implementation

In this section we will explain how the experiments of the previous section are implemented and which important choices were made.

5.1 Requirements

The only functional requirement is that the program must be able to execute the experiment described in the previous section. The execution of the program may take a while so it should be robust. Reliability is also an important non-functional requirement for this program, because wrong results make the whole experiment invalid. The last important non-functional requirement is efficiency, because it is an intensive task in itself the program should not waste time when not necessary. Another requirement is that the program must be able to load and store data and/or plots.

5.2 Decisions

Throughout the project several important decision were made.

- The programming language used is python because there was already code availabe written in python. Python has some packages for scientific and machine learning purposes, which are really useful.
- The programming paradigm is a combination of imperative and object-orientated, since this goes nicely with python. Object-orientated is useful for encapsulating data and splitting the code up in multiple cohesive parts. But in some cases it is simpler to use the imperative paradigm.
- The project is decoupled in two programs. The first program generates data, whilst the second one analyzes the data. It is possible to couple them but then the user cannot run the analysis part without running the generation part. In software engineering it is considered a good principle to split software up into multiple cohesive parts.
- The generation part makes use of settings files. The settings could be hardcoded but this limits the use-ability of the program. With settings files the user can define his own experiment to a certain degree of freedom and since we had to execute it for seven datasets it was easier to create seven settings files. The settings are written in json because it is human readable, easy parseble and reasonably efficient.
- Instead of a normal profiler, which only gives execution time per function call, we use a line profiler which gives execution time per line. This is done because we want to have a detailed overview of the time each line of code takes.
- Pickle was used for storing the generated data. Pickle is availabe in python, widely-used and can store and load data easily.
- For generating the plots matplotlib is used. This is a package for python which can make plots in more or less the same way as the programming language matlab.

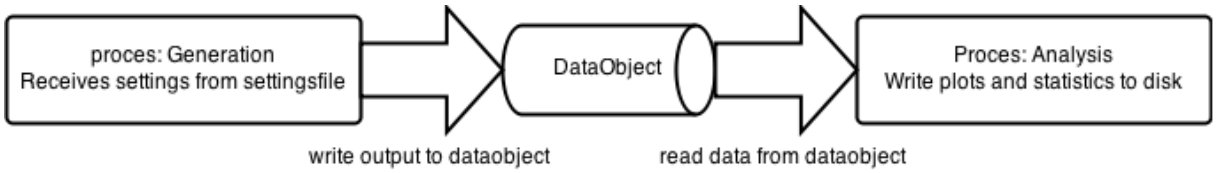


Figure 9: Architecture

5.3 Architecture

Figure 9 shows the architecture of the system. The process generation generates data according to the loop specified in the experiment settings and stores the data in a DataObject which is stored to disk using pickle. The analysis process, reads the DataObject with *pickle* and generates plots for each classifier and scorer. It also writes all the features with their corresponding scores to a file for each scorer.

5.4 Design

Figure 10 displays the class diagram. It gives an overview of the classes and methods available.

Figure 11 shows the sequence diagram for the generation part of the system.

The generation part, handles the retrieval of data. It is specified by a settings file, which contains information like:

1. Input location.
2. Output location.
3. Search method to search in the set of best features.
4. Classifiers to use, and parameter ranges for the classifiers.
5. Scorers to use.

The settings files are written in json to make it parseble for the computer and readable for humans. The settings files are static, which makes them easier to use, it also enhances the flexibility of the program because now you can run several experiments with different settings without changing the source-code. After the setup work is done the generation part generates all data according to the loop specified in the previous section. The generation part contains the following files.

helper_functions.py Contains functions which can be useful in the main code but do not have anything todo with the real purpose of the project.

settings_controller.py Handles the json settings files, checks if they are valid and other parts use it for indirect access to the settings file. By making a controller it hides the details of the other parts.

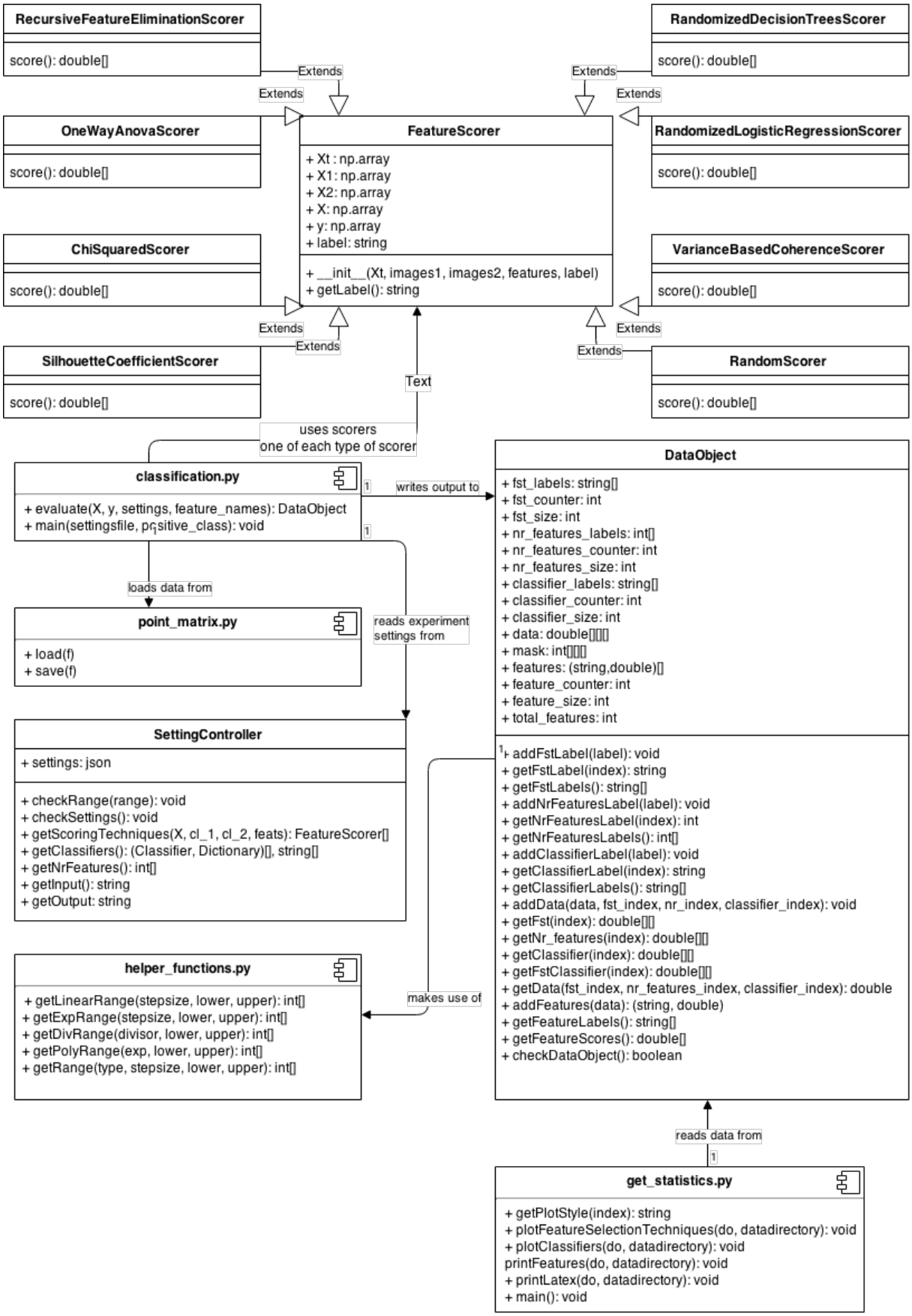


Figure 10: Class diagram

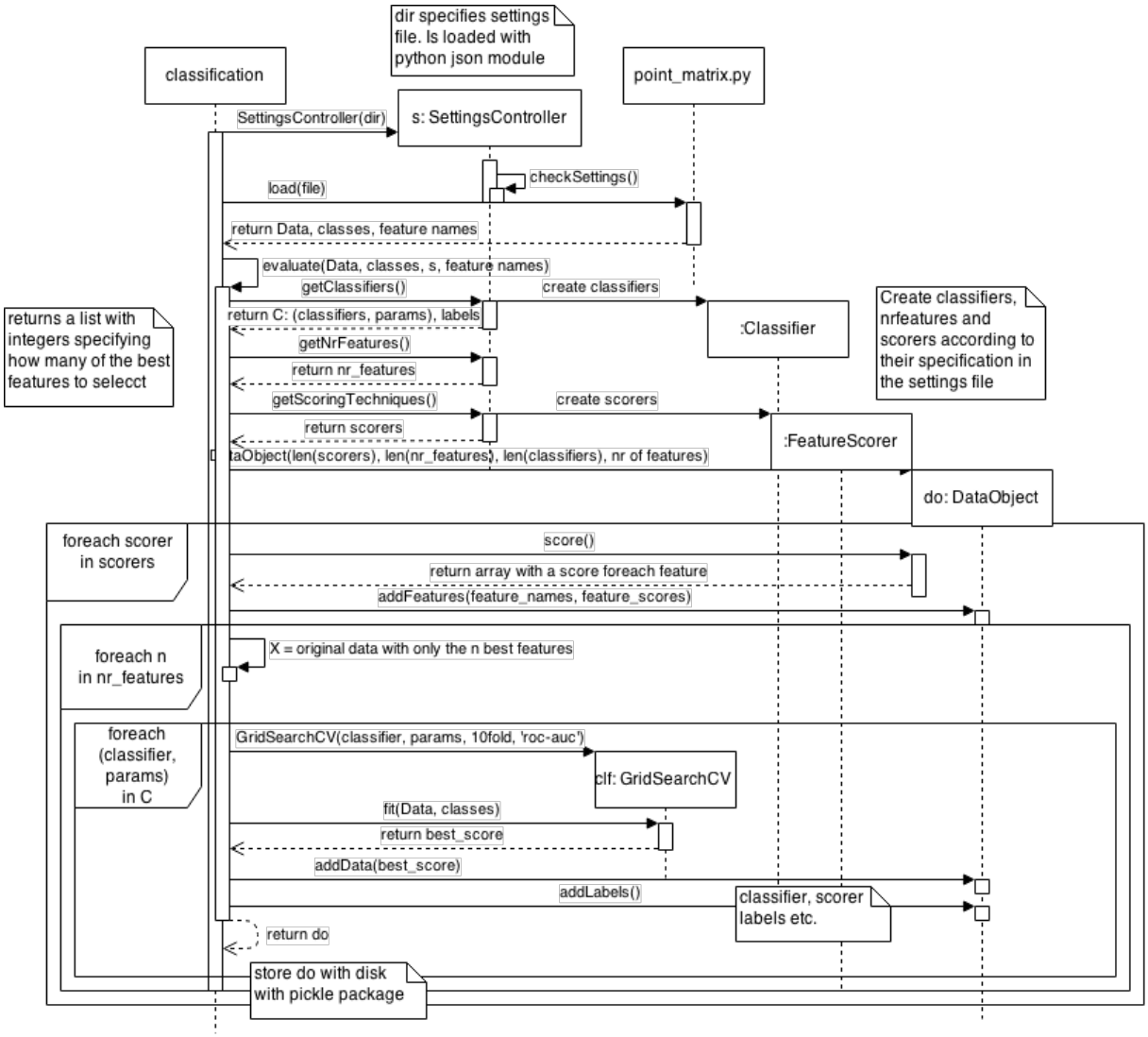


Figure 11: Generation

feature_scoring.py Consists of all the feature scoring techniques. Each technique is a class which inherits from a base class called FeatureScorer. Each scorer has to define a score function, which scores each feature according to the input data and corresponding classes.

classification.py Main part of the program, generates the data and stores it inside a DataObject. The DataObject is stored to disk with pickle.

The DataObject is a class which is in between generation and analysis, by using a DataObject the generation and analysis part are split from each other. So you can do multiple analysis on the same data, or do the analysis much later than the generation, which might be useful.

The diagram in figure 12 displays the analysis part of the system. The analysis part loads the DataObject using pickle, after that it generates the plots using matplotlib and the tables by simply writing the features plus their score to disk, in plain text. The plots are stored in png format.

The analysis part consist of one file generating all plots and other statistics.

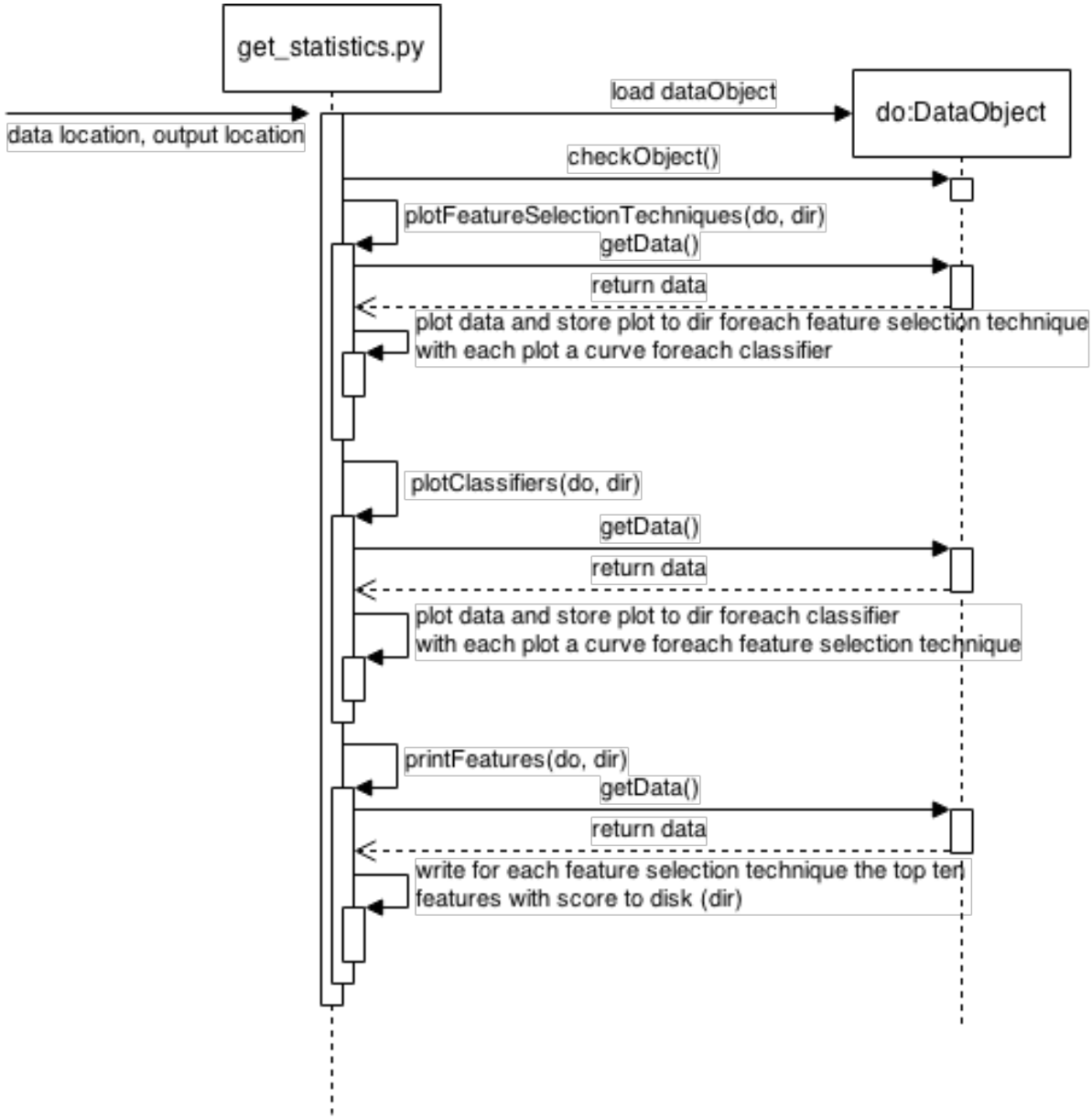


Figure 12: Analysis

5.5 Testing

To ensure that the source-code is correct unit tests were written for each file to test the implementation. With use of a line profiler the performance of the evaluation function was tested on the digits dataset with the same settings as in the experiments. The most important result was that 99.5 percent of the time was spent to training classifiers and the other 0.5 percent was due to scoring the features. This corresponds to about 7510 and 35 seconds, respectively, so, an average of about 4.5 seconds per scorer. As can be seen GridSearchCV takes the most time. This is implemented by the scikit-learn package, which contains state of the art code. Therefore we can be pretty sure GridSearchCV is implemented efficiently. So it can be concluded that our implementation works efficiently. The entire log of the line profiler can be found in Appendix H.

6 Discussion

Although the results are valid and can be used for comparing feature scorers there are a few limitations to the way the experiments were executed. First of all the experiments are not very scalable. Using the scorers, datasets and classifiers mentioned in this thesis it took about 24 hours to run the experiments. Adding a new dataset will increase the execution time significantly because it has to be evaluated with each combination of classifiers and scorers. The same holds for adding scorers or classifiers. As shown in the previous section the most time is spend on training the models. So the best solution is to limit the number of models which have to be trained or to make the training phase of models more efficient.

Right now we use cross-validation in combination with classifier selection. So the classifier is trained using all the data [10]. The classifier is scored by averaging the ROC-AUC score over all folds. The problem with this is that it does generalize well because the score for the classifier is calculated with data it has already seen. For example take a random scorer with as parameter a seed, there is a seed for which the classifier has a perfect score on the training data. It may take a long time but it sure is possible. This would lead to the false conclusion that it is a good classifier. A better way for evaluating the classifier is to seperate the data into a test and training set. The training set is used to train the classifier, possibly using cross-validation. The resulting classifier is evaluated with the test set, which contains data it has not seen before. This way the results generalize better. This approach can be extended by making using of double k-fold cross-validation. This works by splitting up the data into test and training sets k times. The resulting k scores are then averaged to get the final score. Note that this kind of cross-validation does not influence the classifier parameters.

So a better type of experiment using this approach would have been:

1. Split data up into training and testing sets.
2. Obtain the best parameters of the classifier using the training set and cross-validation.
3. Train the classifier with the trainings set using the parameters obtained in the previous step.
4. Evaluate the classifier using the test set.
5. Repeat steps 1 to 4 multiple times and use the average score as final score.

More information about model assessment can be found in chapter seven of the book by Hastie et al [22]. The addition of a validation step is also explained in the book by Alpaydin [10].

7 Conclusions

The goal of this thesis was to compare several feature seclection techniques with each other. Some feature selection techniques make use of a classifier to select features. Other methods are completely independent of a classifier. Two of these feature selection techniques were experimental.

The main research questions were:

- Can experimental methods be used for feature selection?
- What is the difference between filters and wrappers in terms of accuracy and robustness?
- How many features can be removed without influencing the accuracy?
- Which method has the best performance?

Our research shows that the experimental methods are not well suited for feature selection, although there are datasets for which they obtain reasonable results. It also shows that the difference in accuracy between filters and wrappers is minimal in our data. Only if a wrapper is carefully choosen it outperforms the filter methods significantly. This can be seen in the RFE scorer in combination with the Melanoma dataset. Filters tend also te be more robust, posssibly because they are not dependent on a classifier. Wrappers on the other hand are in some cases less robust because it is dependent of a classifier. A fine example of this is the result obtained with the RFE scorer in combination with the Madelon dataset. Wrappers which make use of randomization are also more robust than other methods. Especially randomized decision trees perform well. It is in general among the top scorers and never has really bad results. The number of features usually can be reduced by a factor larger than 10 using a good feature selection technique.

There are feature selection techniques which perform really well. However each technique makes assumptions so there are cases where a feature selection technique does not work because it makes the wrong assumptions about the data. Because of this it is a good idea that people can select good feature selection technique and classifier according to their dataset. A solution for this problem is described by Li [15].

Extensions of this project could be: Generate metrics for measuring the differences between scorers. For example how many features are both in the top ten of scorer A and scorer B, and how many of them are in the correct place. It could be extended to even more datasets, for this an automatic inspector has to be written because for a lot of datasets it is hard to inspect them all manually. Another direction of research is to combine several scorers to get a ranking to possibly increase model efficacy.

References

[1] Chi squared. `https://github.com/scikit-learn/scikit-learn/blob/bb39b49/sklearn/feature_selection/univariate_selection.py#L166`. Accessed: 2015-05-19.

[2] The digit dataset. `http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html`. Accessed: 2015-05-04.

[3] How are feature importances in randomforestclassifier determined. http://stackoverflow.com/questions/15810339/how-are-feature-importances-in-randomforestclassifier-determined. Accessed: 2015-05-27.

[4] Madelon data set. https://archive.ics.uci.edu/ml/datasets/Madelon. Accessed: 2015-05-04.

[5] One-way anova implementation scikit-learn. https://github.com/scikit-learn/scikit-learn/blob/bb39b49/sklearn/feature_selection/univariate_selection.py#L38. Accessed: 2015-06-16.

[6] Pen-based recognition of handwritten digits data set. http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits. Accessed: 2015-05-04.

[7] Random forest classifier. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed: 2015-06-09.

[8] Randomized logistic regression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RandomizedLogisticRegression.html. Accessed: 2015-05-11.

[9] Silhouette score. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html. Accessed: 2015-05-18.

[10] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, third edition, 2014.

[11] L. Breiman. Bagging predictors. *Machine Learning*, 1996.

[12] L. Breiman. Random forests. *Machine Learning*, 2001.

[13] D. Dey, T. Solorio, M. M. y Gomez, and H. J. Escalante. Instance selection in text classification using the silhouette coefficient measure. *Advances in Artificial Intelligence*, 2011.

[14] Argenziano et al. Interactive atlas of dermoscopy. *EDRA Medical Publishing and New Media*, 2002.

[15] Li et al. Automatic linguistic indexing of pictures by a statistical modelin approach. *TPAMI*, 2003.

[16] Paiva et al. Improved similarity trees and their application to visual data classification. *TVCG*, 2011.

[17] S. Feringa. Comparison of features used in automatic skin lesion classification. Master's thesis, University of Groningen, 2015.

[18] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 2006.

[19] P. Geurts and L. Gilles. Learning to rank with extremely randomized trees. *JMLR: Workshop and Conference Proceedings*, 2011.

[20] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 2003.

[21] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 2002.

[22] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, second edition, 2009.

[23] L. Huan and Y. Lei. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[24] T. Mitchell. *Machine Learning.* McGraw Hill, first edition, 1997.

[25] J. Quinlan. *C4.5.* Morgan Kaufmann, first edition, 1992.

[26] P. E. Rauber, R. O. da Silva, S. Feringa, M. E. Celebi, A. X. Falcao, and A. C. Telea. Interactive image feature selection aided by dimensionality reduction. *EuroVis Workshop on Visual Analytics*, 2015.

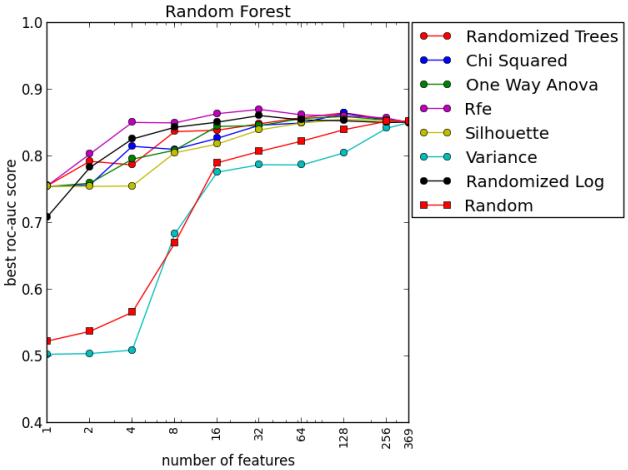
[27] C. T. N. Suzuki, J. F. Gomes, A. X. Falcao, J. P. Papa, and S. Hoshino-Shimizu. Automatic segmentation and classification of human intestinal parasites from microscopy images. *IEEE TBE 2013*, 2013.

[28] R. Timofeev. Classification and regression trees(cart) theory and applications. Master's thesis, Humboldt University, 2004.

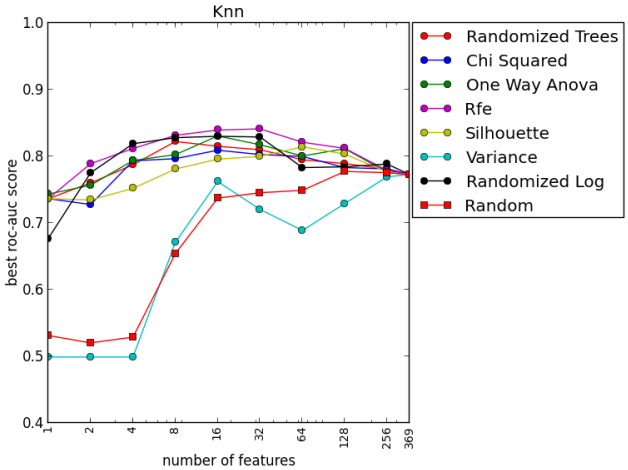
[29] J.E. Vargas, A.X. Falcão, J.A. dos Santos, J. C. D. M. Esquerdo, A.C. Coutinho, and J. F. G. Antunes. Contextual superpixel description for remote sensing image classification. *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015.

[30] J. Verzani. *Using R for Introductory Statistics.* Chapman and Hall/CRC, first edition, 2002.

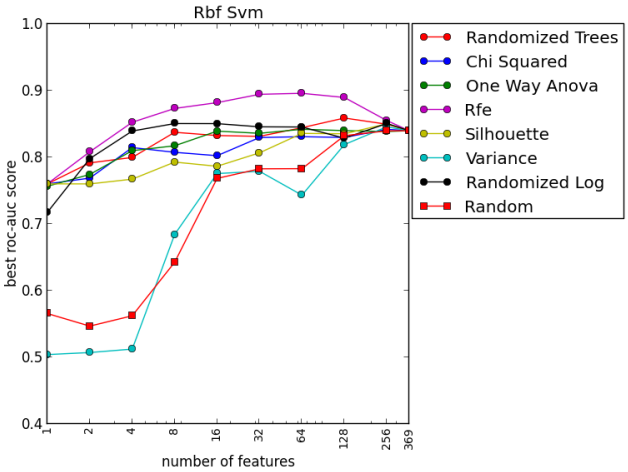
A Additional results for the Melanoma dataset



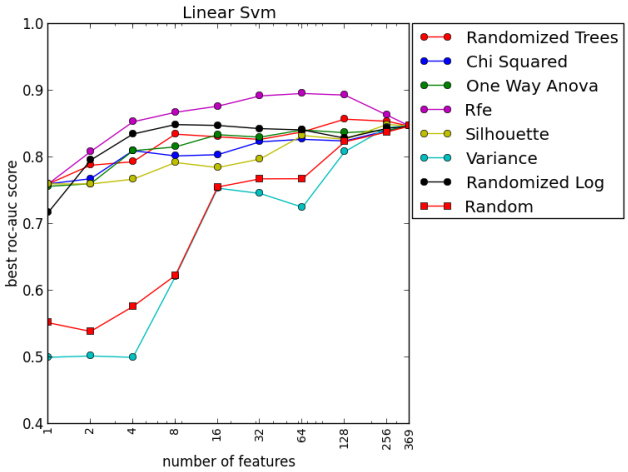
(a) Results for the Random forest classifier



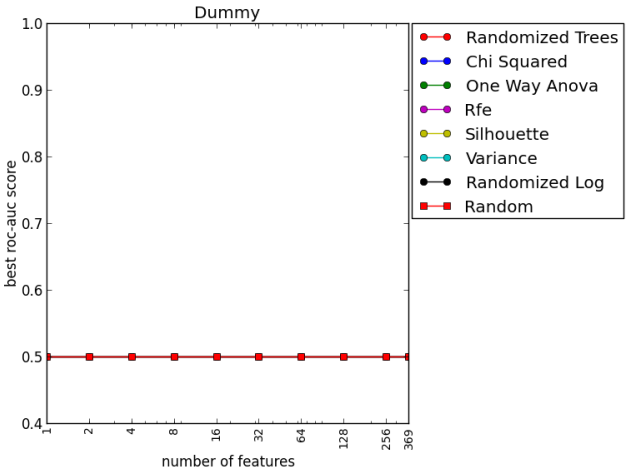
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

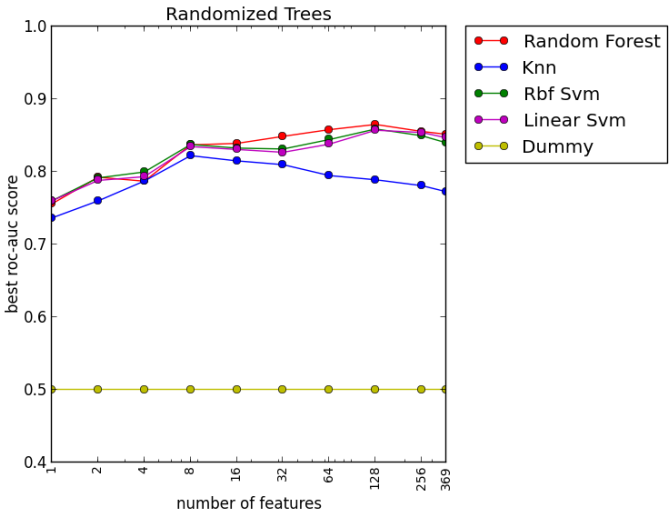


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

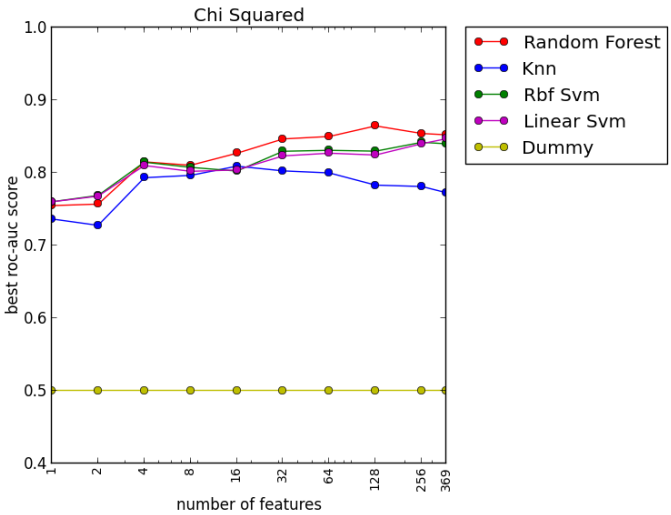
Figure 13: Plot per classifier for the Melanoma dataset



(a) Results for the randomized trees scorer

Name	Score
lab_std_b	0.0148338892955
hist_r_3	0.0142589015212
hist_r_4	0.0138597755617
lab_variance_b	0.0130158747192
hist_r_5	0.0123145163531
lab_hist_b_8	0.0120252048574
lab_hist_b_9	0.0109351894358
hist_r_6	0.0101349858022
lab_hist_l_4	0.00968096165561
hist_r_2	0.00913302535397

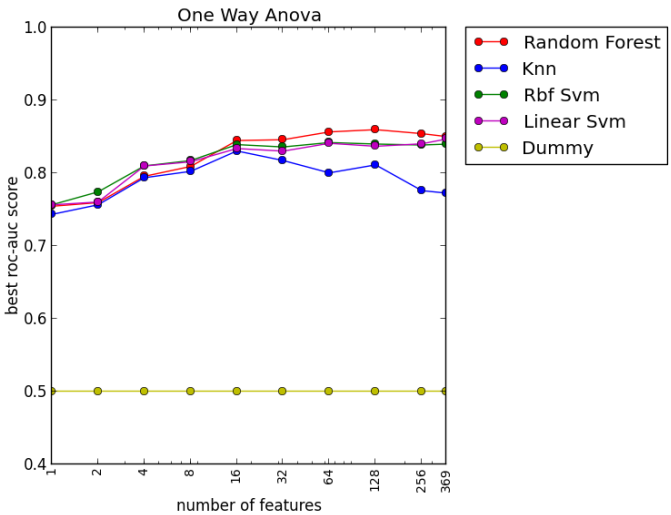
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
hist_r_3	0.0359568506748
hist_r_4	0.0304006606698
hist_r_5	0.0290082755382
lab_hist_b_8	0.0279183112921
hist_r_6	0.0260343476115
lab_hist_l_2	0.0248916772959
lab_hist_l_1	0.0236718452136
lab_hist_l_4	0.0220431877404
lab_hist_l_0	0.0218224521993
hist_g_0	0.0217374110083

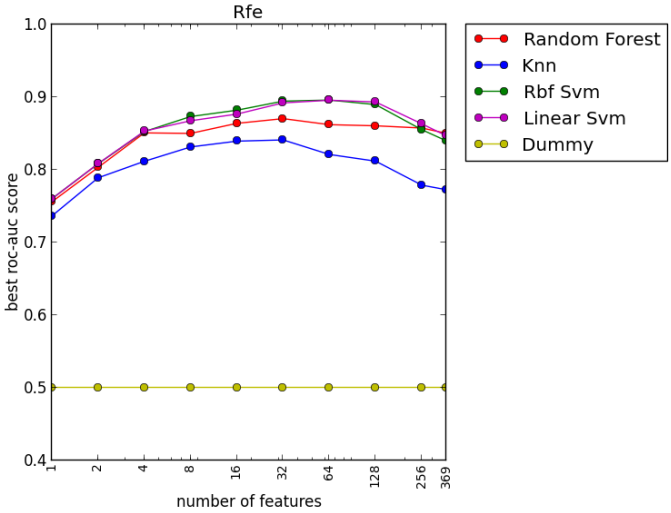
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
hist_r_5	0.020644787813
hist_r_4	0.0200175261969
hist_r_6	0.0197561940106
lab_std_b	0.0191422136593
lab_variance_b	0.0182906100394
hist_r_3	0.0175687216118
lab_hist_l_5	0.0167886194297
lab_hist_l_4	0.0164500746041
lab_hist_b_8	0.0161123089701
lab_hist_b_9	0.0150488639745

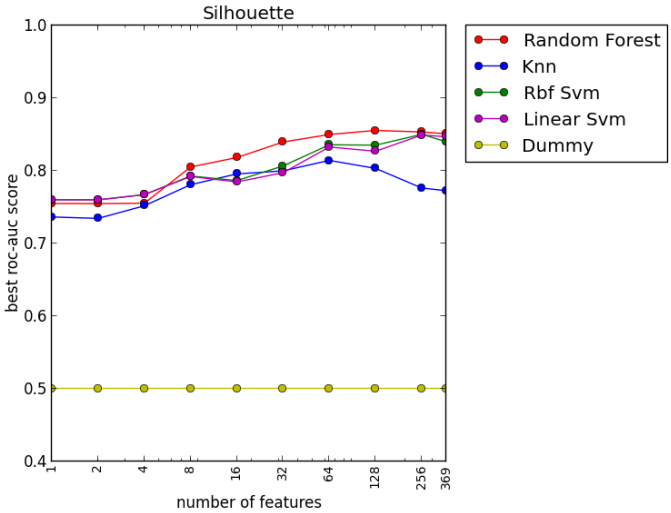
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
hist_r_3	368.0
lab_std_b	367.0
variance_g	366.0
std_r	365.0
inverseDiff_om0_r	364.0
entropy_om0_b	363.0
lab_hist_l_2	362.0
hist_g_5	361.0
sobel_hist_r_15	360.0
homogeneity_om0_r	359.0

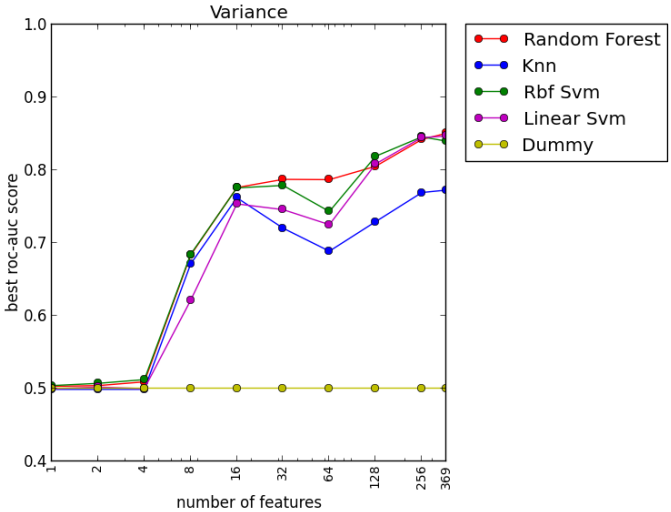
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
hist_r_3	1.0
hist_r_2	0.99199822364
hist_r_4	0.963150536308
lab_hist_l_1	0.958592569175
hist_r_1	0.951259499909
lab_hist_l_0	0.950707661522
lab_hist_b_7	0.944554779482
lab_hist_b_5	0.941243318487
lab_hist_b_6	0.939229341784
lab_hist_a_5	0.933254688319

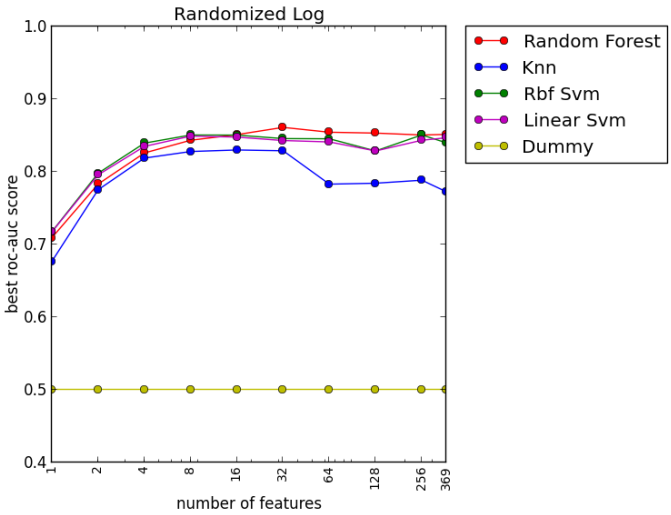
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
lab_lbp_a_5	1.0
lbp_b_0	1.0
border_path_hist_13	1.0
lbp_g_0	1.0
border_path_hist_14	1.0
lab_hist_a_4	1.0
hog_1	0.996757541528
correlation_om1_g	0.986532789084
correlation_om0_g	0.98620703887
kurtosis_r	0.976948040637

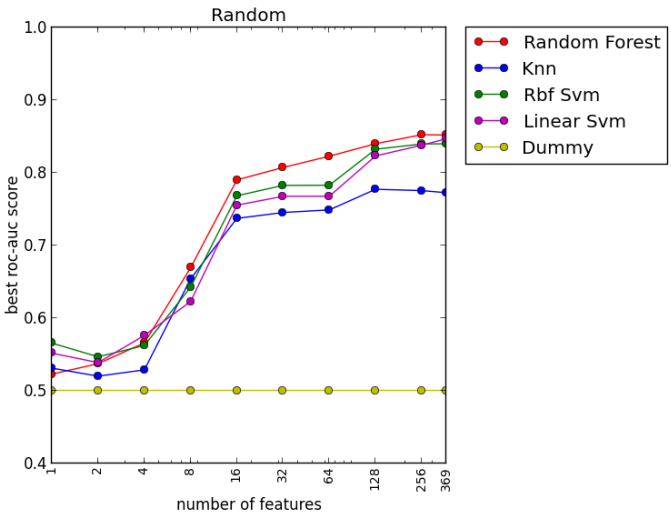
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
lab_std_b	0.8
std_r	0.535
hist_r_3	0.505
lab_col_var_a	0.505
lab_hist_b_9	0.5
col_var_g	0.495
lbp_g_13	0.485
hist_r_6	0.47
hist_g_5	0.465
lab_hist_l_5	0.455

(b) Top 10 scores for randomized log



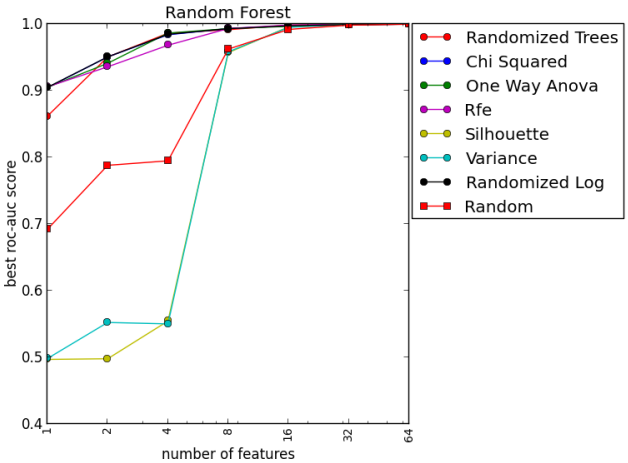
(c) Results for the random scorer

Name	Score
lab_std_l	0.998362836963
lab_lbp_a_10	0.993958065297
lbp_g_6	0.988120215588
lab_hist_a_4	0.986382779525
lab_lbp_b_0	0.983617055697
hist_r_6	0.982508485252
hist_r_9	0.977734782086
hog_3	0.973555694366
hist_r_3	0.965127362623
absolute_value_om0_b	0.963289191914

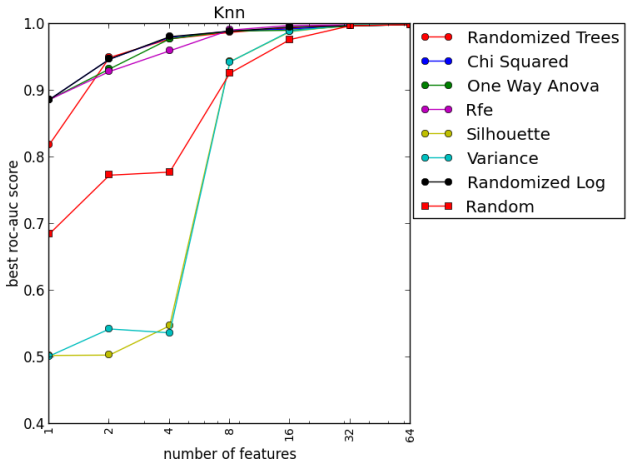
(d) Top 10 scores for random

Figure 16: Plots for each scorer for the Melanoma dataset

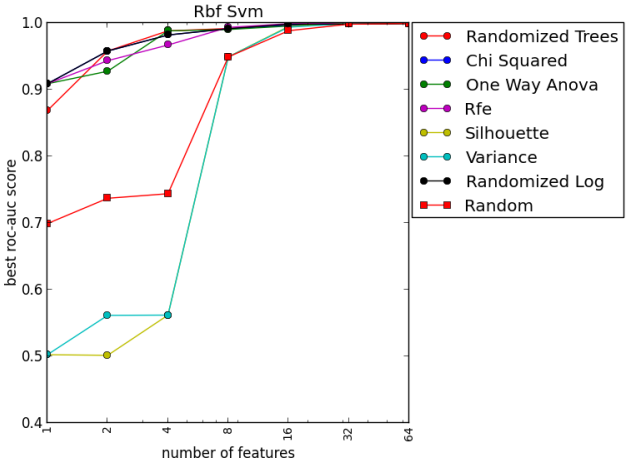
B Additional results for the Digits dataset



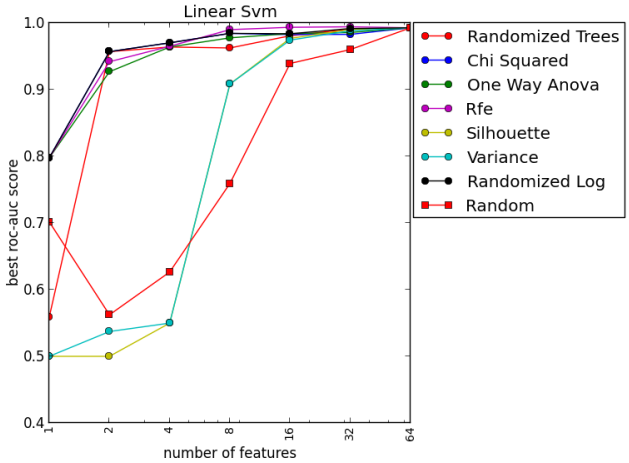
(a) Results for the Random forest classifier



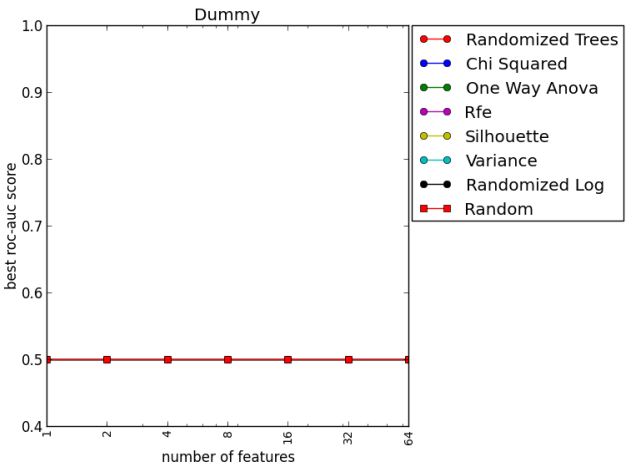
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

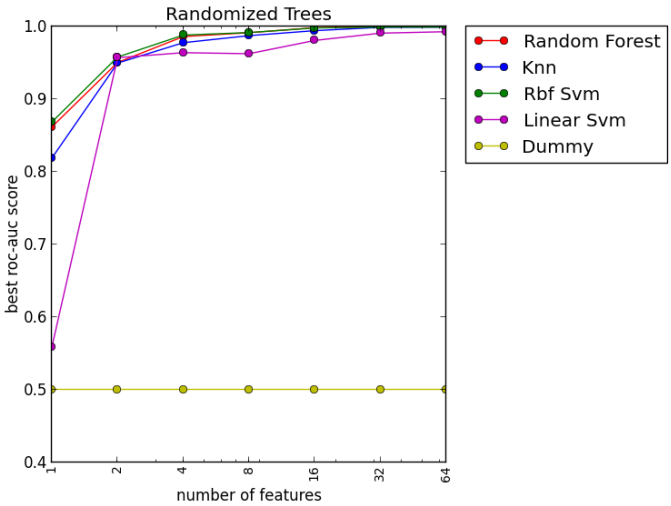


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

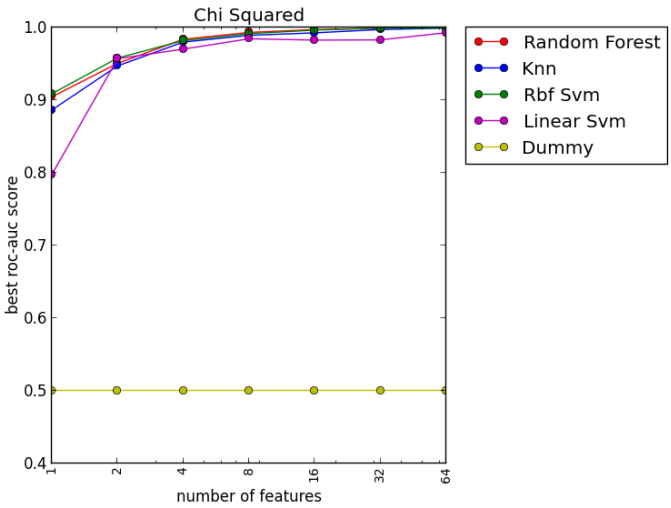
Figure 17: Plot per classifier for the Digits dataset



(a) Results for the randomized trees scorer

Name	Score
f19	0.10567942271
f20	0.0875587647514
f10	0.0660622074265
f27	0.0503572316224
f12	0.0478233578682
f44	0.0357803118701
f28	0.0267335541157
f38	0.0232645208855
f58	0.0223366340933
f52	0.0221080334224

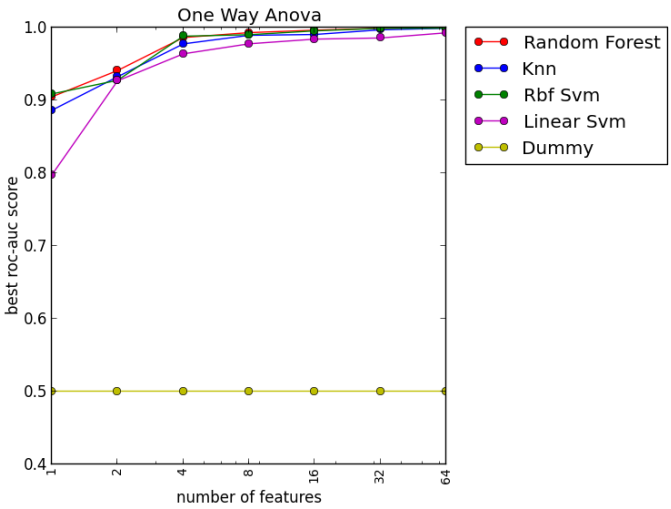
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
f19	0.132954096066
f20	0.105766181021
f44	0.0645509137885
f10	0.0577648160746
f63	0.0526614367775
f27	0.0508817835288
f46	0.041921695524
f38	0.0382971995856
f58	0.0297386744623
f52	0.0276357603666

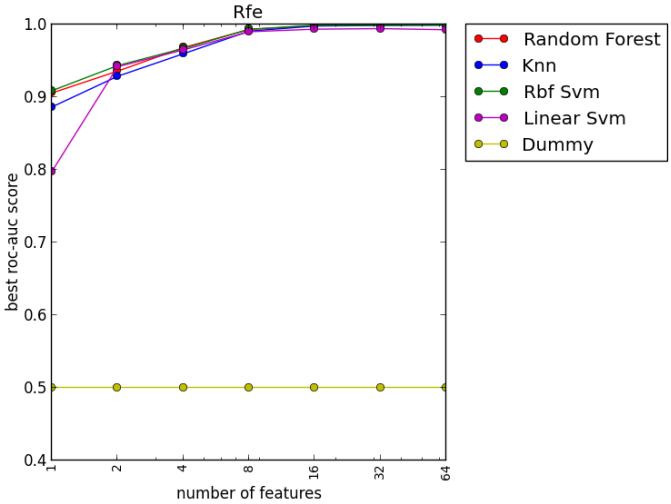
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
f19	0.139670766736
f10	0.0966246948478
f20	0.0926192901094
f27	0.0576203846917
f44	0.0560045042786
f52	0.0398964367228
f12	0.0379294067782
f38	0.0331891143006
f46	0.0328872389791
f59	0.0289312871171

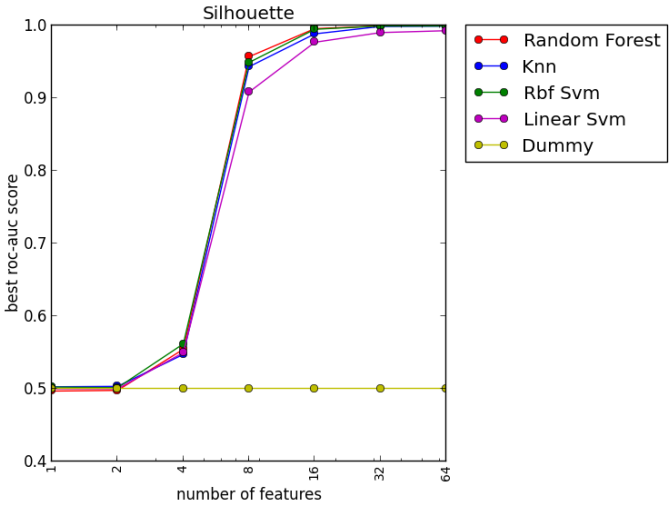
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
f19	63.0
f38	62.0
f9	61.0
f46	60.0
f20	59.0
f41	58.0
f10	57.0
f4	56.0
f14	55.0
f5	54.0

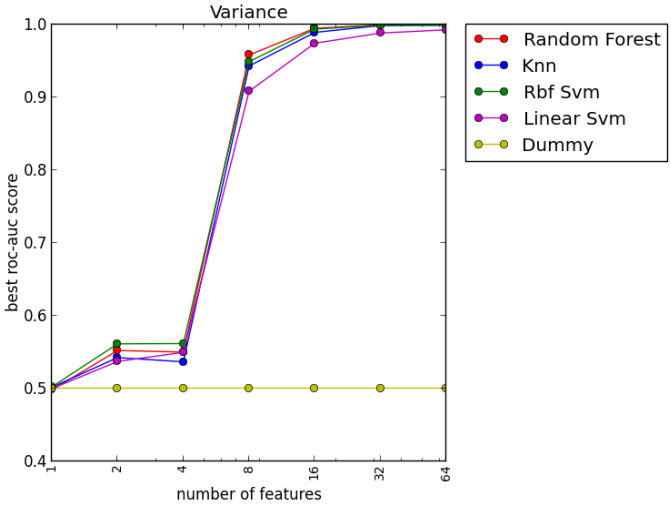
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
f24	1.0
f16	0.969477781255
f63	0.96438208973
f55	0.914063118897
f59	0.716250798882
f10	0.71237743338
f3	0.706067413655
f19	0.693655421404
f62	0.680003153229
f20	0.641870912662

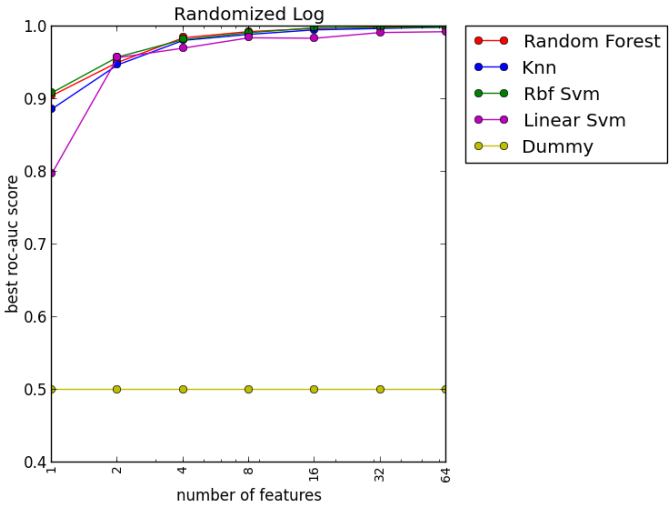
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
f16	1.0
f63	0.89505793292
f24	0.874639054774
f55	0.746490720317
f3	0.416967416185
f59	0.409420223374
f10	0.388392765023
f19	0.388304011616
f20	0.32248959444
f62	0.316956588531

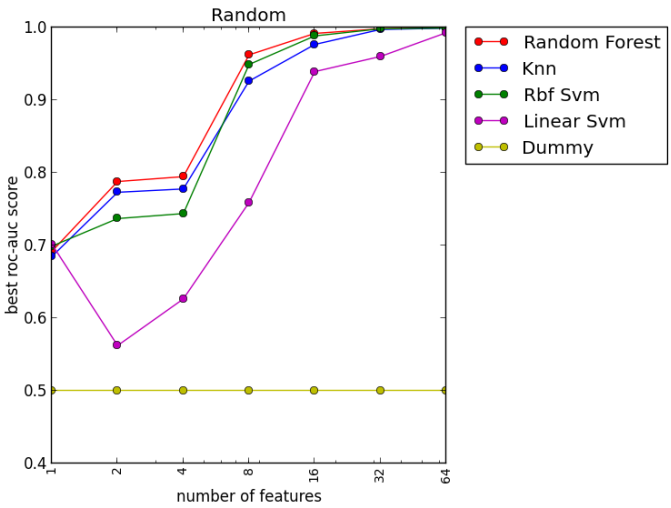
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
f19	0.945
f20	0.94
f10	0.635
f44	0.555
f63	0.535
f38	0.485
f46	0.455
f41	0.415
f27	0.36
f58	0.345

(b) Top 10 scores for randomized log



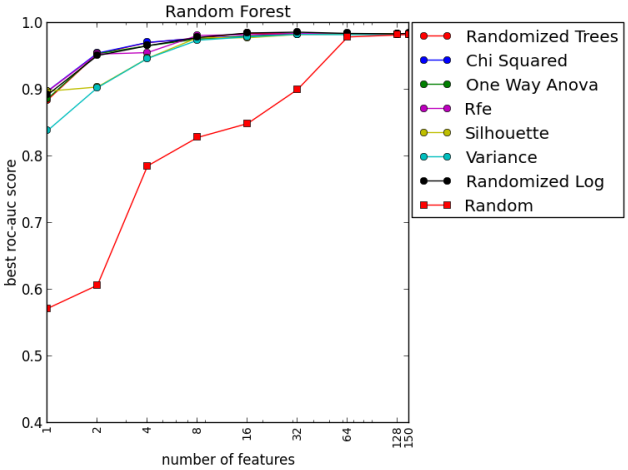
(c) Results for the random scorer

Name	Score
f41	0.995941666291
f3	0.994443449662
f19	0.978092868557
f59	0.962949496107
f20	0.941403199487
f6	0.923812653839
f34	0.921031795717
f27	0.897539469812
f4	0.891458172657
f45	0.866221274319

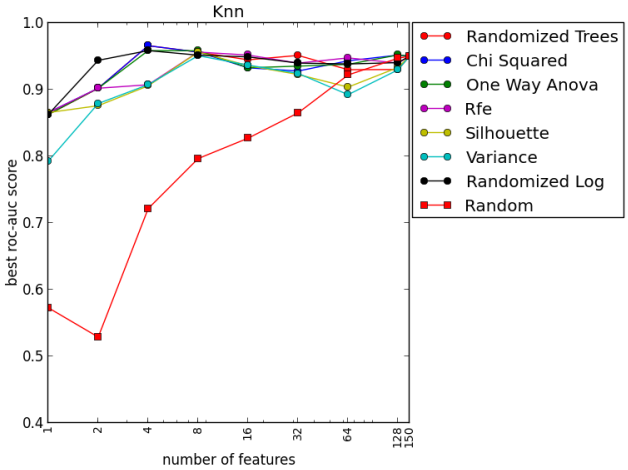
(d) Top 10 scores for random

Figure 20: Plots for each scorer for the Digits dataset

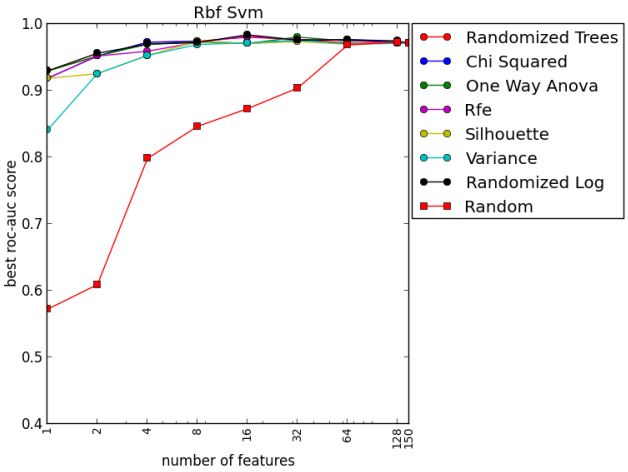
C Additional results for the Corel dataset



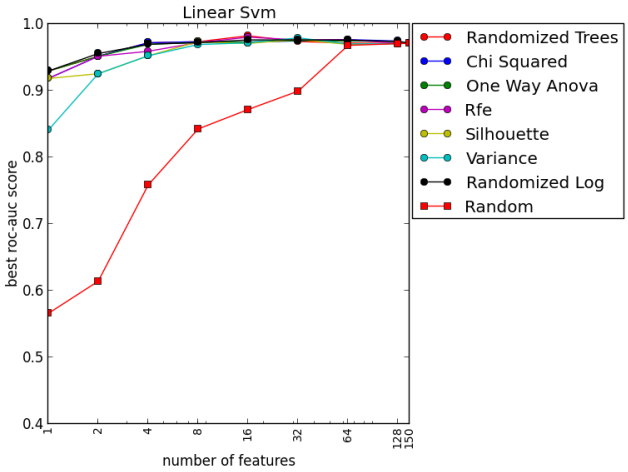
(a) Results for the Random forest classifier



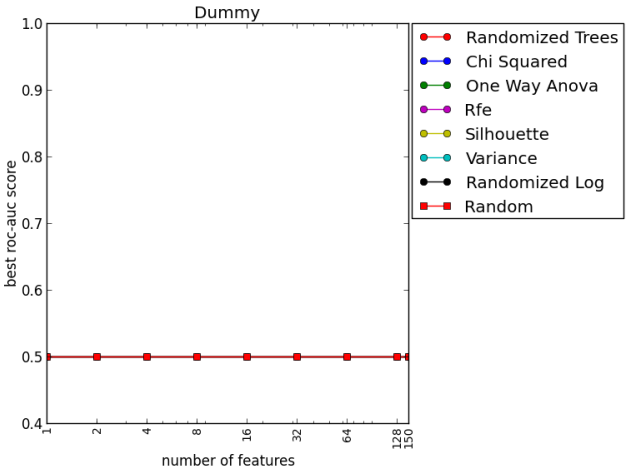
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

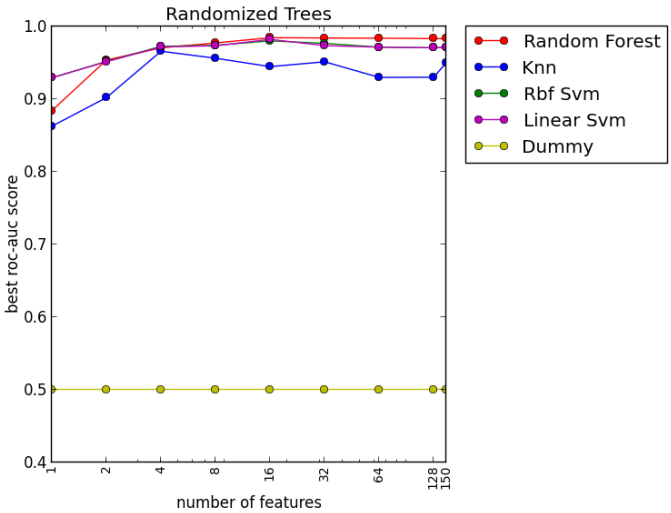


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

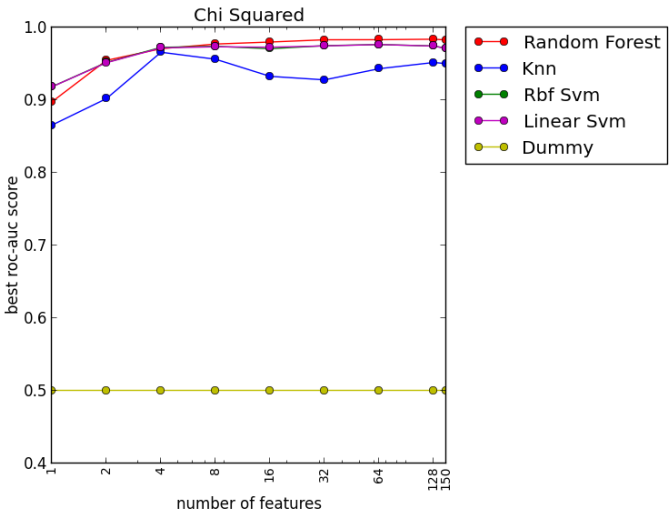
Figure 21: Plot per classifier for the Corel dataset



(a) Results for the randomized trees scorer

Name	Score
attr123	0.0950427231352
attr145	0.0853061342341
attr143	0.0592886559544
attr81	0.0540110322172
attr110	0.0422034270036
attr0	0.0415073514927
attr66	0.0399078610317
attr141	0.0362986142991
attr148	0.0324108187772
attr72	0.0235734596777

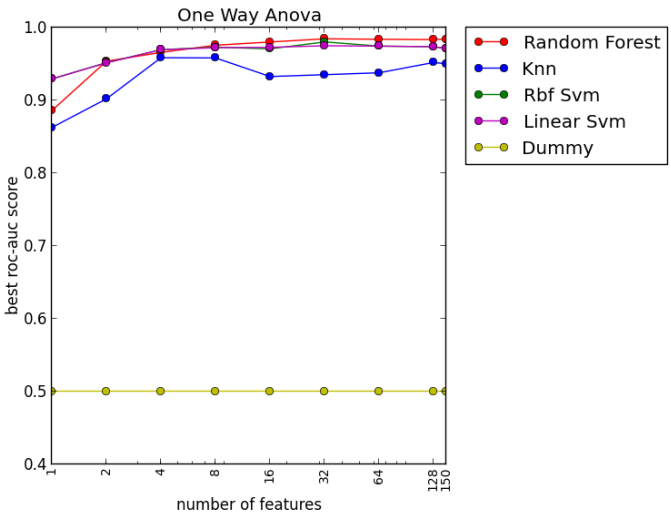
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
attr145	0.132612663268
attr123	0.111216866361
attr143	0.0851230005166
attr81	0.0654659967753
attr0	0.0602401893638
attr110	0.0577038057427
attr66	0.0575556508298
attr141	0.0488615937185
attr148	0.0390524001778
attr72	0.0308416781475

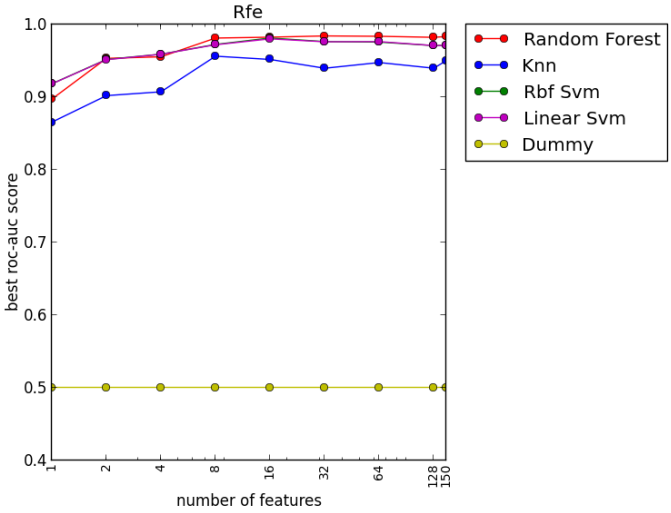
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
attr123	0.154673148991
attr145	0.138508684039
attr143	0.0995180522793
attr0	0.0695152955519
attr110	0.0686587149349
attr148	0.0554994612826
attr81	0.0524750644841
attr141	0.0474136614442
attr66	0.0445349356824
attr72	0.0228885159958

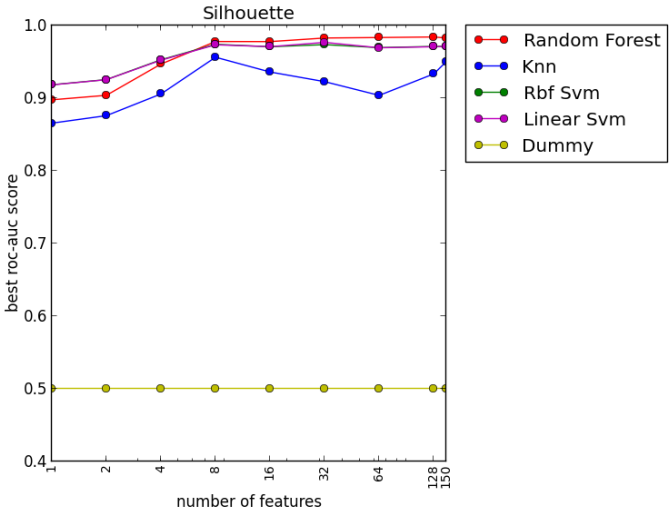
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
attr145	149.0
attr123	148.0
attr81	147.0
attr66	146.0
attr110	145.0
attr72	144.0
attr101	143.0
attr143	142.0
attr70	141.0
attr0	140.0

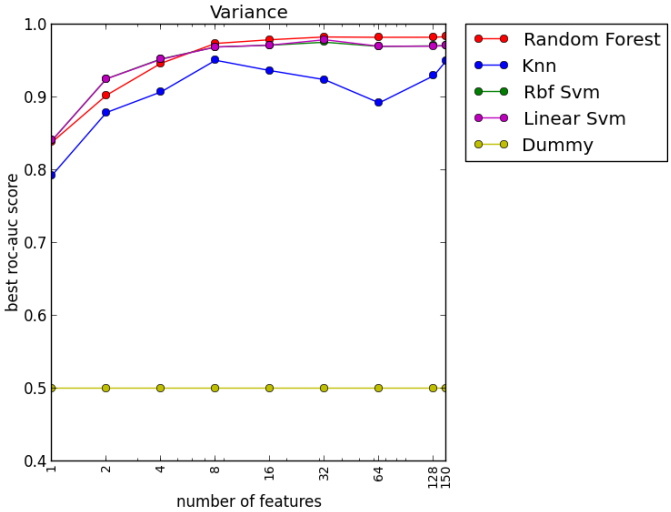
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
attr145	1.0
attr81	0.999421898531
attr141	0.981740168549
attr66	0.966666635467
attr123	0.961284963898
attr0	0.95384394716
attr110	0.932241975489
attr143	0.887792899226
attr72	0.878182473215
attr148	0.859124936718

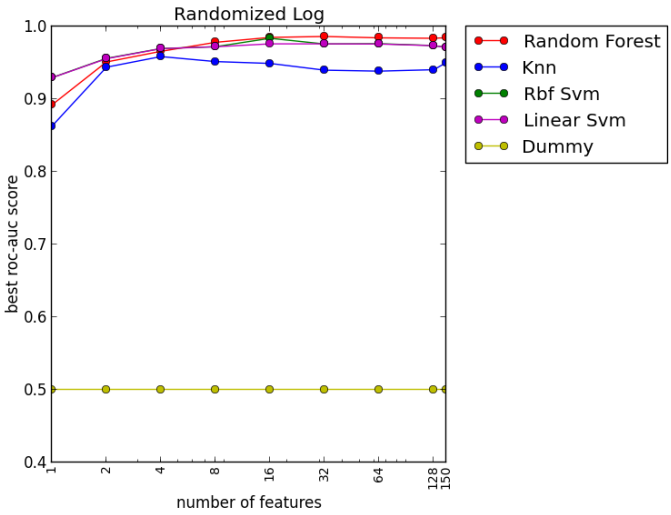
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
attr81	1.0
attr145	0.967216037752
attr66	0.942512155036
attr141	0.928577858512
attr123	0.865745611263
attr0	0.827107414611
attr110	0.688862054653
attr148	0.612969876077
attr143	0.582230992666
attr129	0.544364110488

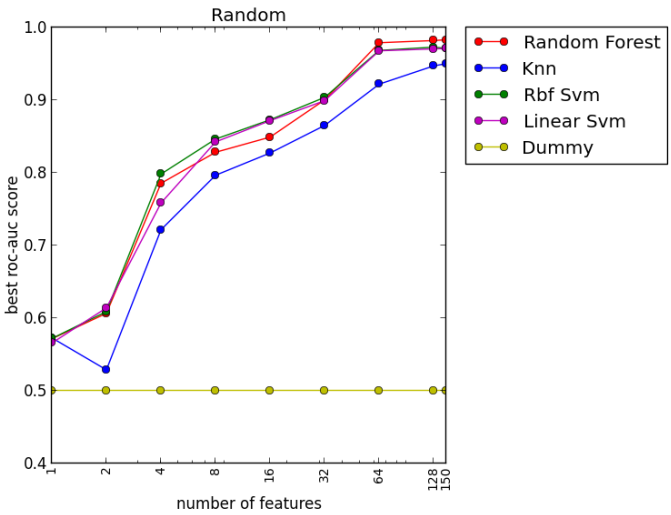
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
attr123	0.92
attr143	0.775
attr145	0.61
attr0	0.54
attr110	0.53
attr66	0.485
attr148	0.48
attr72	0.39
attr141	0.33
attr81	0.305

(b) Top 10 scores for randomized log



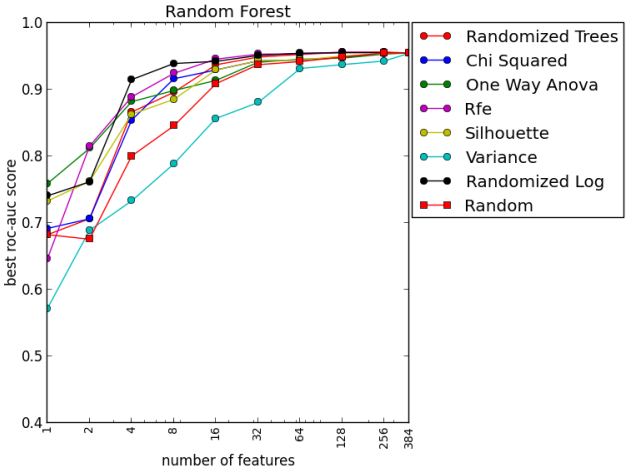
(c) Results for the random scorer

Name	Score
attr86	0.999121338192
attr8	0.994959860373
attr42	0.991992517339
attr75	0.988966828792
attr107	0.98344854582
attr31	0.982535991119
attr118	0.967058609346
attr108	0.964581433912
attr1	0.959594800063
attr49	0.956350860838

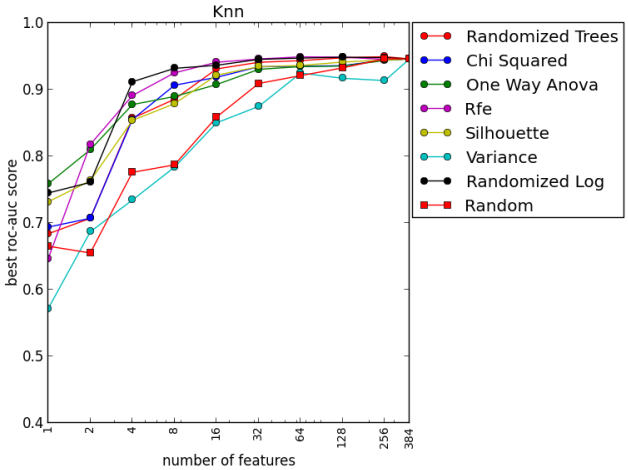
(d) Top 10 scores for random

Figure 24: Plots for each scorer for the Corel dataset

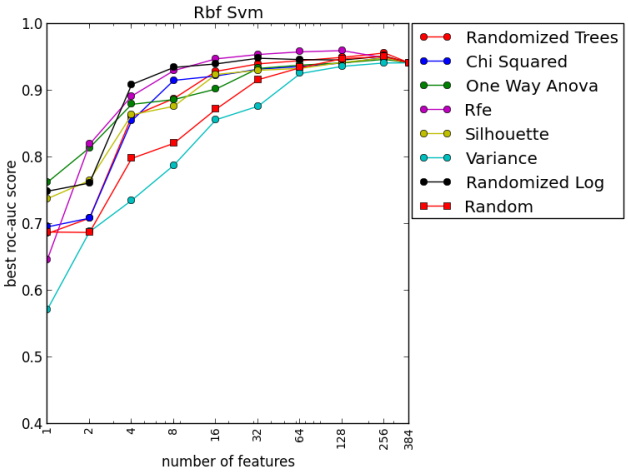
D Additional results for the Rome dataset



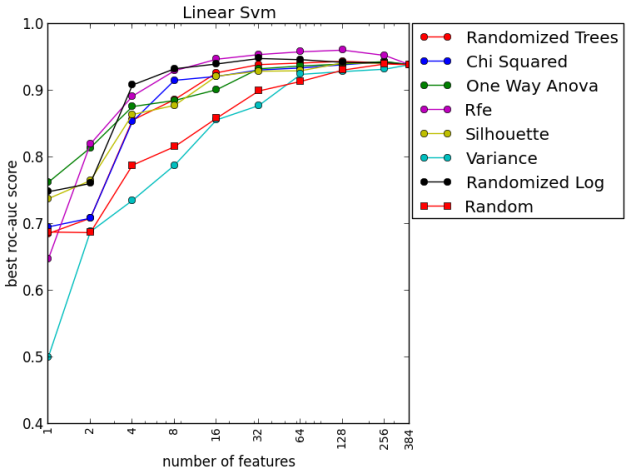
(a) Results for the Random forest classifier



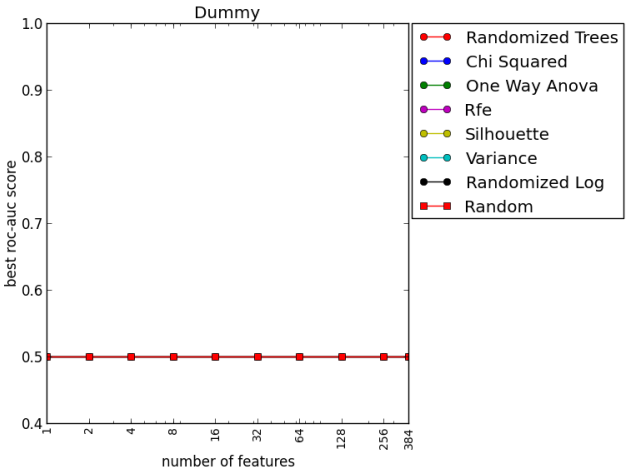
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

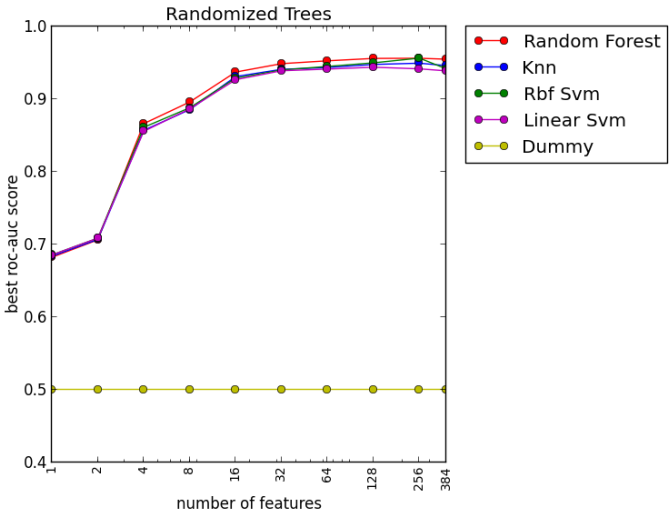


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

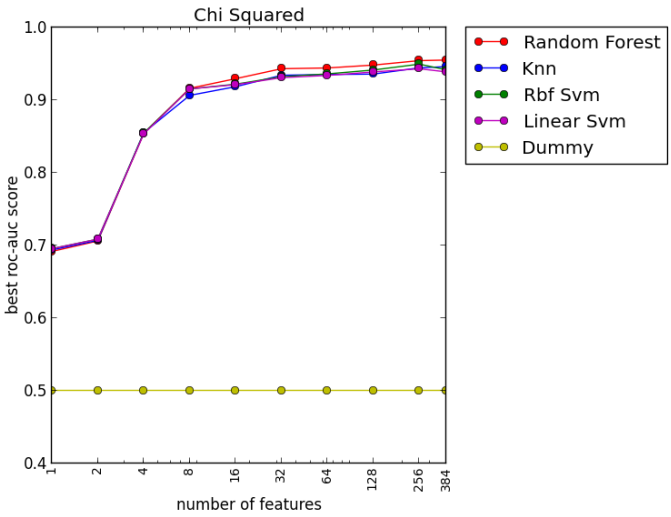
Figure 25: Plot per classifier for the Rome dataset



(a) Results for the randomized trees scorer

Name	Score
61	0.0201869013719
253	0.0172949140379
201	0.0129710542644
255	0.011872208636
213	0.011854524397
218	0.0118346110912
47	0.0116841457018
245	0.0109208410979
199	0.0104825083523
33	0.0104763997192

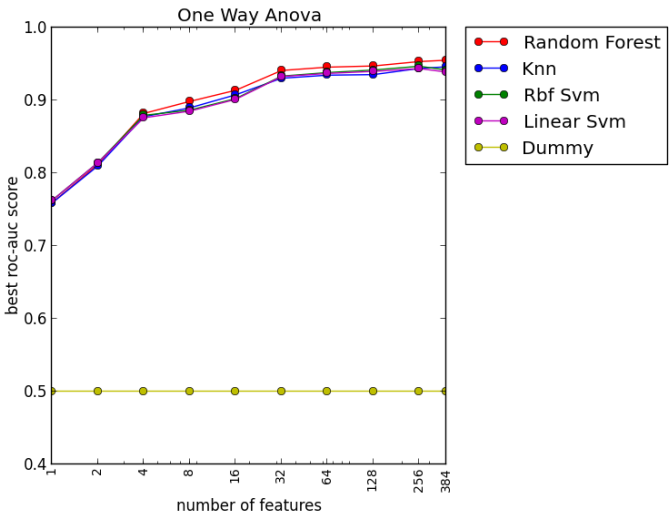
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
61	0.0149587118146
253	0.0139575441209
207	0.0135545754851
255	0.0134361486243
213	0.012833039011
47	0.0120380922142
224	0.0119017860626
245	0.0116580111835
33	0.0115593685903
201	0.0102477201808

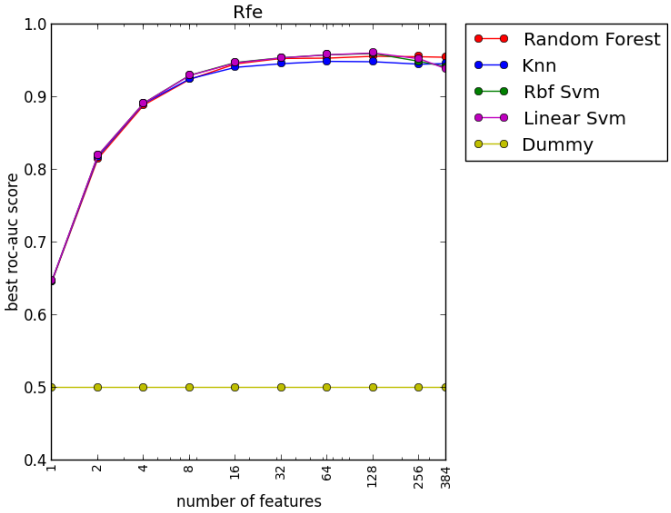
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
255	0.0122926243166
47	0.0121837490046
201	0.0118608519783
218	0.010459678321
33	0.010275102957
17	0.0098019143204
207	0.00973228989228
226	0.00944227908781
61	0.00912692600461
245	0.00885838196647

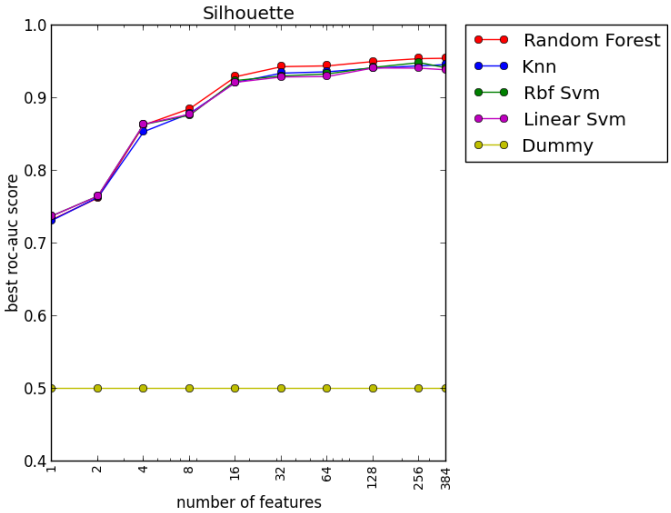
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
56	383.0
213	382.0
253	381.0
35	380.0
217	379.0
59	378.0
241	377.0
61	376.0
49	375.0
130	374.0

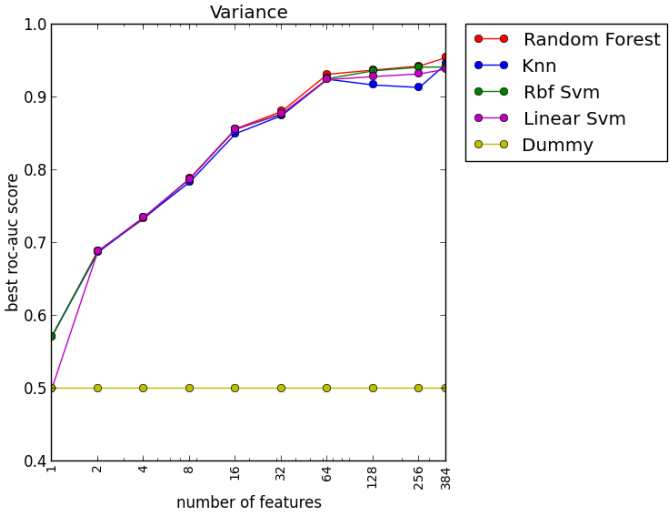
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
33	1.0
58	0.907900221671
47	0.904640821972
14	0.879093500973
61	0.858091204425
255	0.84289231251
253	0.815565180503
251	0.809517086893
218	0.793860479763
227	0.778944943931

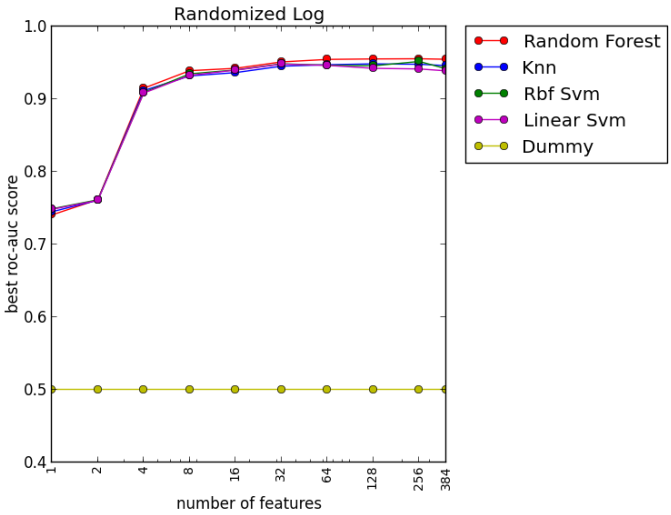
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
32	1.0
39	0.993218376773
15	0.990877453084
62	0.989888752467
211	0.986338642228
19	0.985943601898
36	0.984809663401
9	0.982918429456
240	0.982352790124
28	0.979333390518

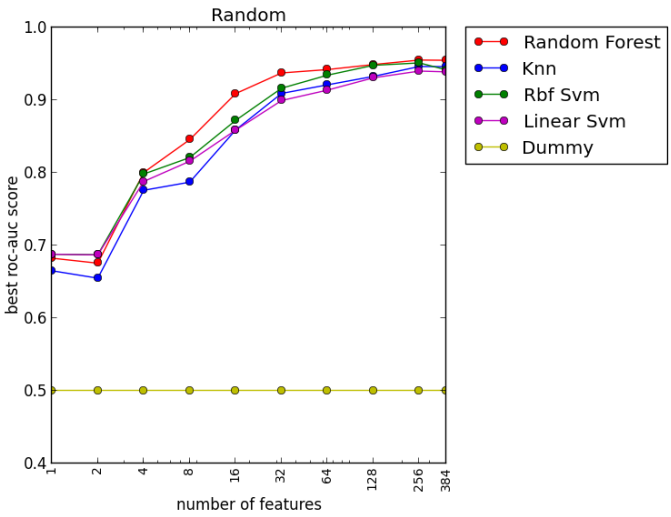
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
218	0.955
217	0.78
213	0.645
61	0.595
245	0.585
253	0.585
39	0.525
310	0.525
215	0.525
43	0.52

(b) Top 10 scores for randomized log



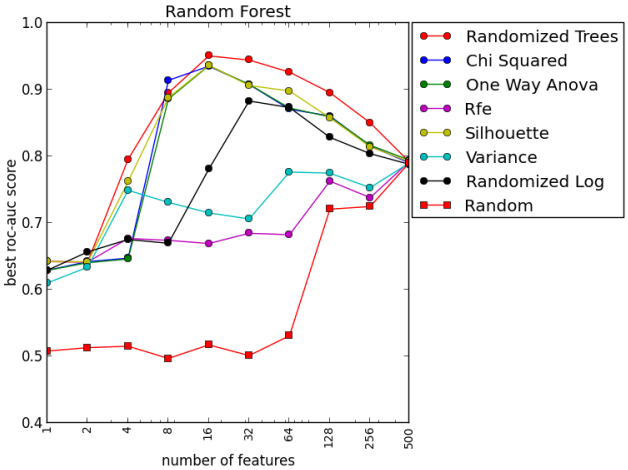
(c) Results for the random scorer

Name	Score
38	0.997550469219
10	0.99570975089
48	0.995565685396
364	0.984220461139
296	0.984055423903
88	0.981049247893
376	0.979443910035
368	0.978712106731
314	0.974609625752
247	0.969026539289

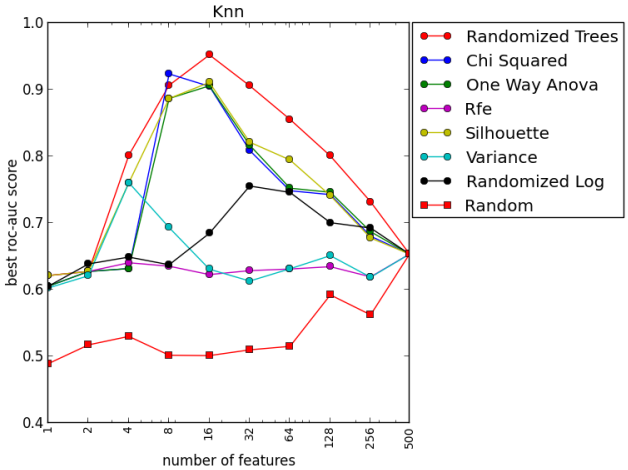
(d) Top 10 scores for random

Figure 28: Plots for each scorer for the Rome dataset

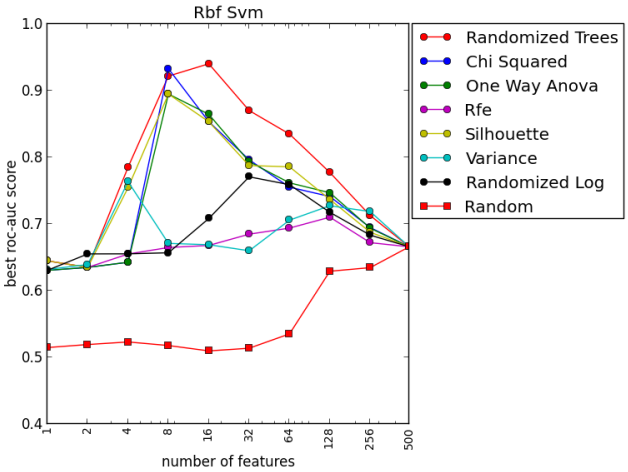
E Additional results for the Madelon dataset



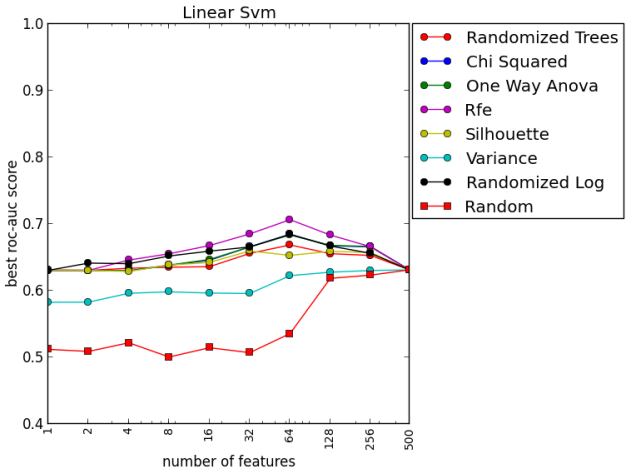
(a) Results for the Random forest classifier



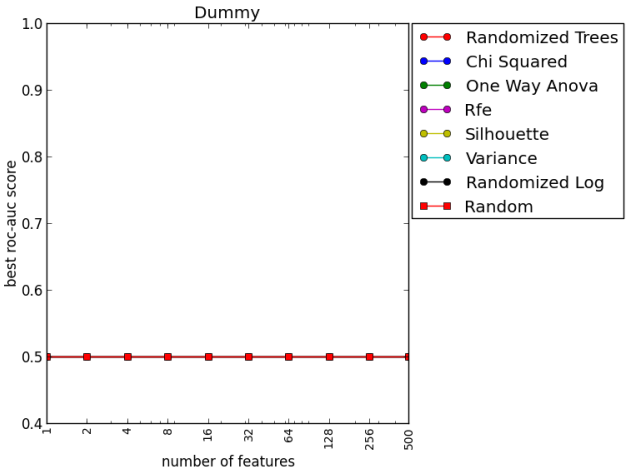
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

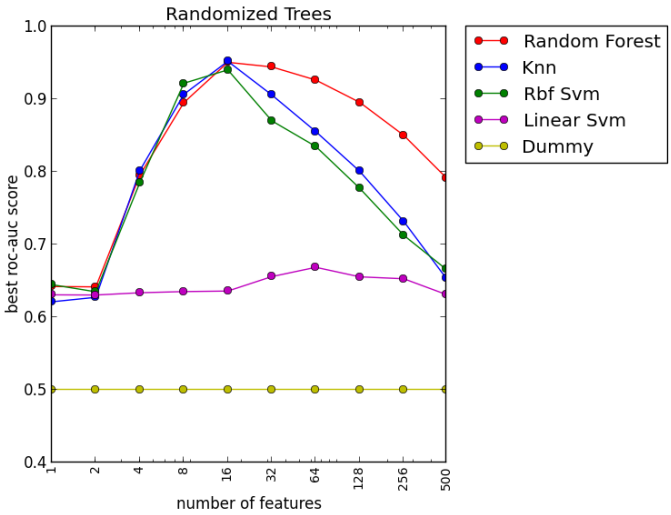


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

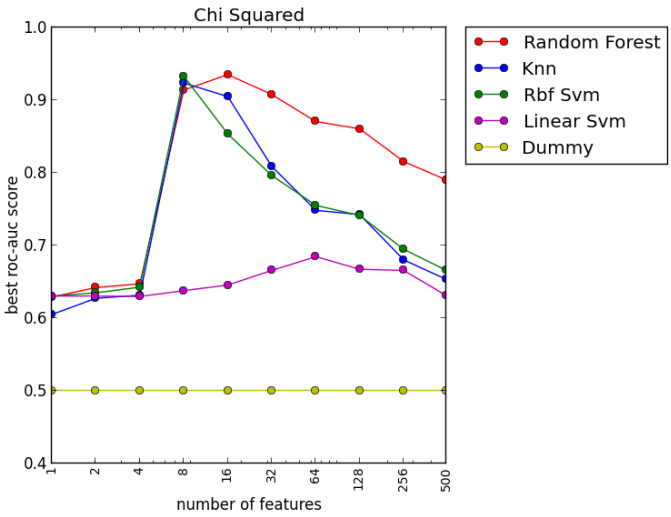
Figure 29: Plot per classifier for the Madelon dataset



(a) Results for the randomized trees scorer

Name	Score
475	0.0109540831656
241	0.0102285203101
338	0.00738356257387
442	0.00653542364078
105	0.00624560265883
128	0.00611196678276
336	0.00602427332174
472	0.00587298463172
064	0.00563062242935
378	0.00560338944075

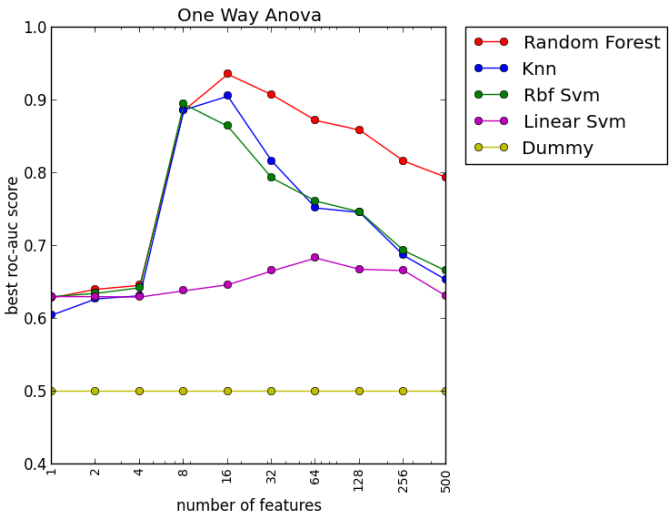
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
475	0.12632866166
241	0.102972125632
336	0.0515447813676
064	0.0458818428259
378	0.0385958064493
048	0.035612181048
338	0.0305317126688
442	0.0291851257586
472	0.0271462248829
128	0.0237881639042

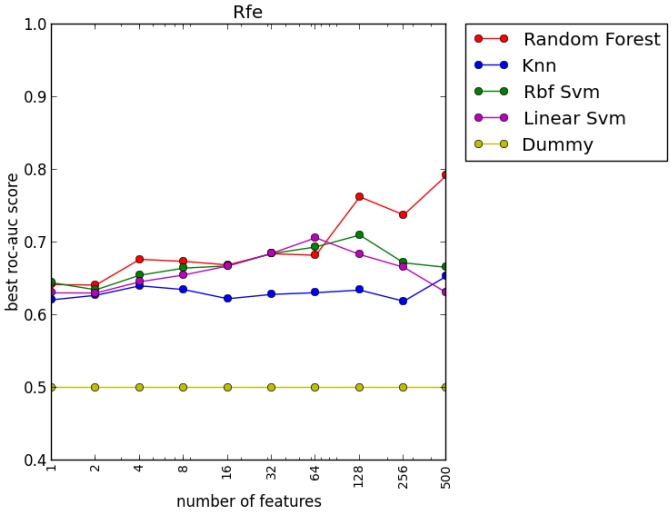
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
475	0.0963528692938
241	0.0944244985621
336	0.0446344074369
064	0.0438278024389
128	0.0309551786835
105	0.0307101582963
338	0.0285398129648
048	0.0269872239391
378	0.0258381827772
442	0.0251180485261

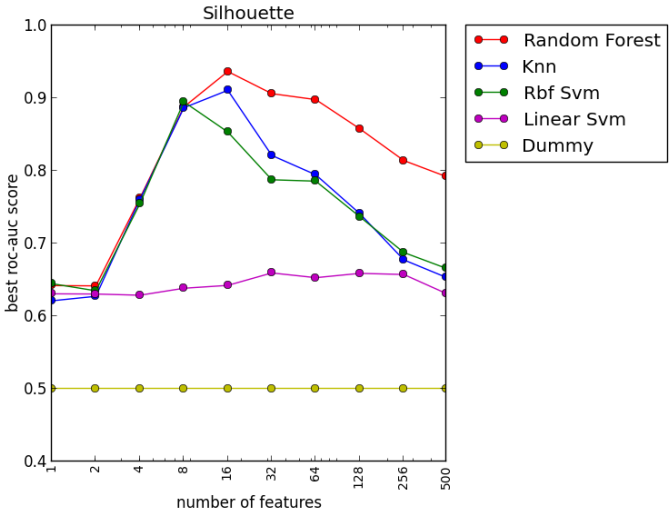
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
241	499.0
475	498.0
048	497.0
424	496.0
205	495.0
204	494.0
496	493.0
323	492.0
026	491.0
347	490.0

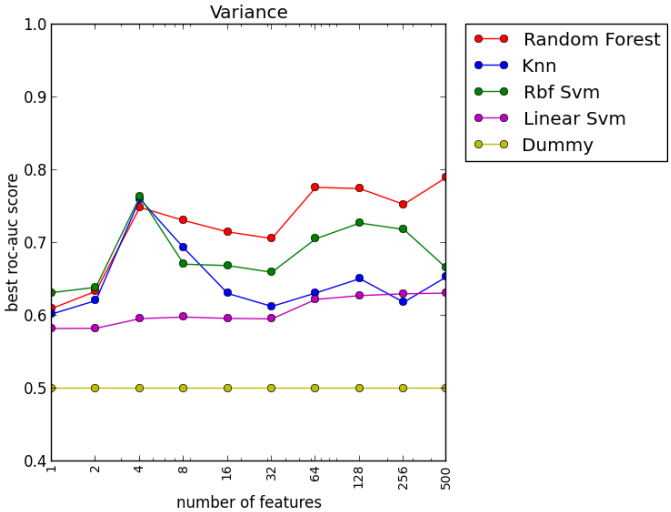
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
241	1.0
475	0.993689163308
338	0.562949113223
336	0.452753086656
064	0.428577408269
128	0.28994237544
105	0.289529779365
048	0.235534042417
378	0.231834769291
442	0.215304495521

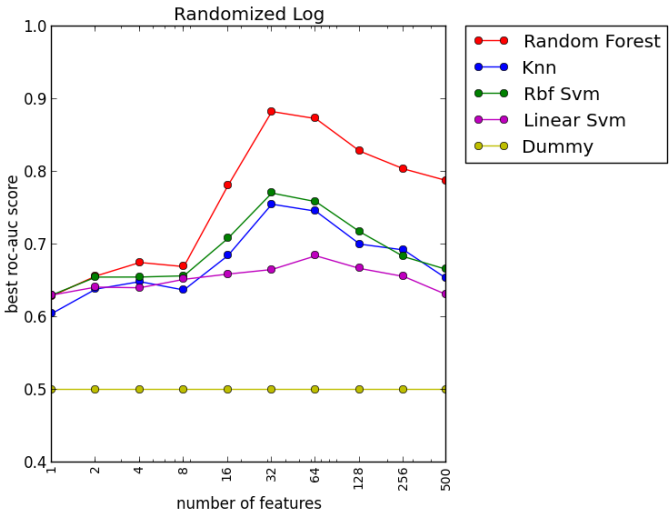
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
338	1.0
442	0.799515281732
105	0.779853750638
472	0.776870302889
128	0.752101254204
056	0.560237549865
348	0.532661362016
282	0.51506831874
302	0.501784292524
102	0.499515901258

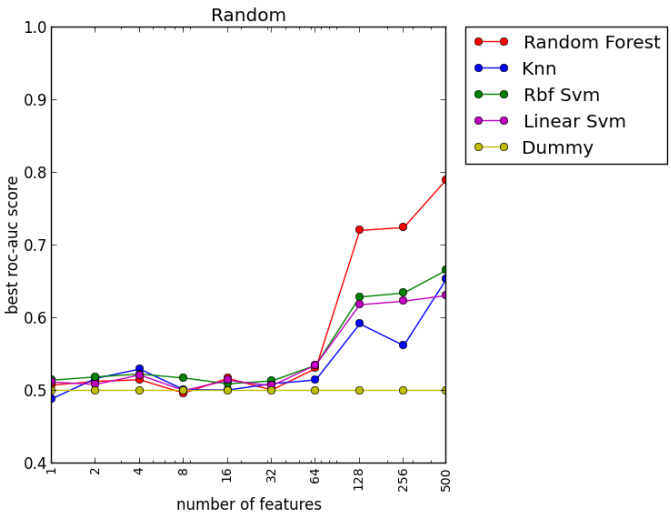
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
475	0.675
048	0.54
241	0.405
323	0.335
424	0.295
378	0.24
282	0.215
496	0.19
205	0.18
204	0.17

(b) Top 10 scores for randomized log



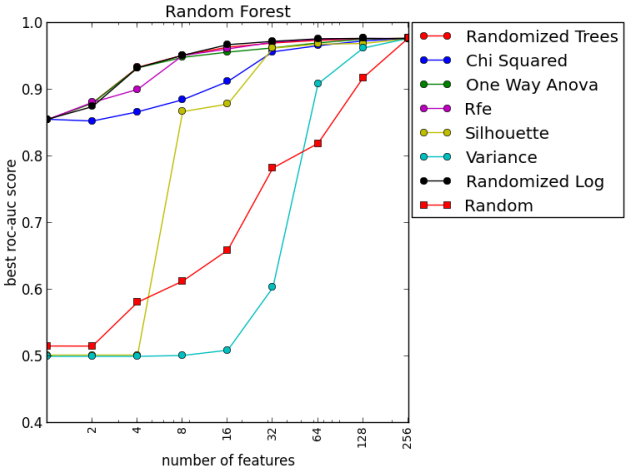
(c) Results for the random scorer

Name	Score
391	0.998655320078
081	0.994937023127
482	0.992722618283
375	0.990655863785
150	0.990496598159
168	0.988793214309
141	0.987825988485
461	0.987282818016
290	0.986734835822
009	0.985665468088

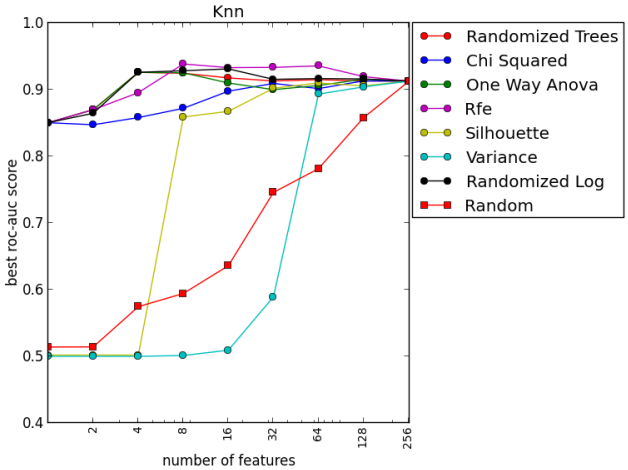
(d) Top 10 scores for random

Figure 32: Plots for each scorer for the Madelon dataset

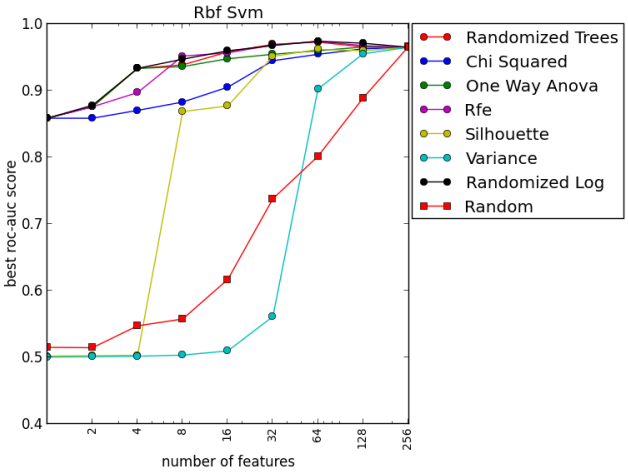
F Additional results for the Parasites dataset



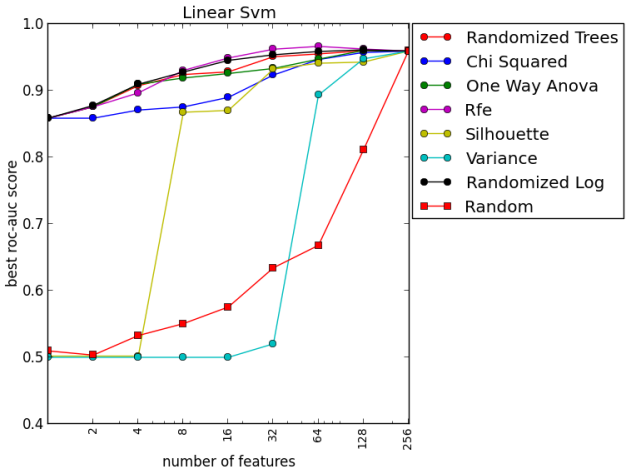
(a) Results for the Random forest classifier



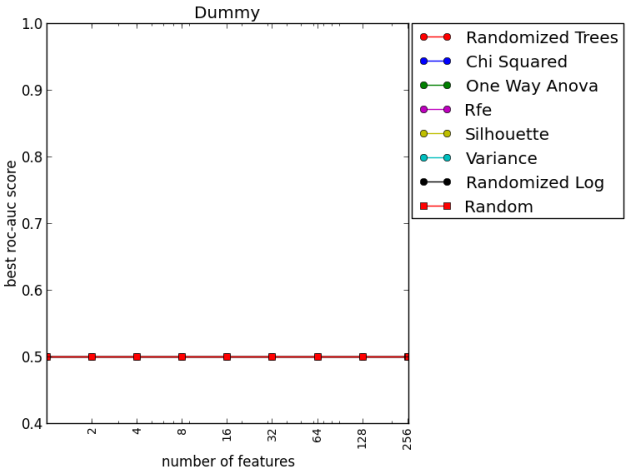
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

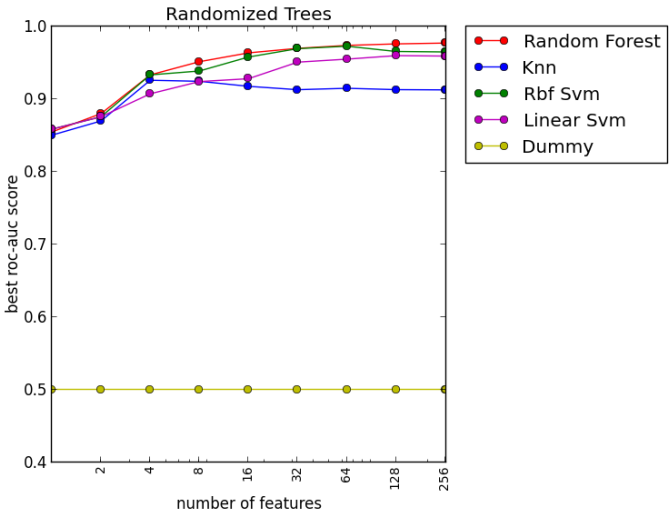


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

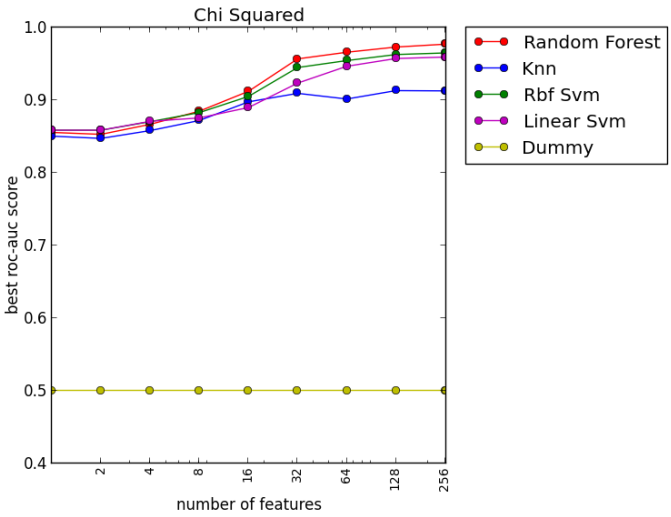
Figure 33: Plot per classifier for the Parasites dataset



(a) Results for the randomized trees scorer

Name	Score
2	0.092933138756
5	0.0519248403642
4	0.0360136039488
171	0.031738583283
135	0.0217043163677
41	0.0211925745191
0	0.0210051611428
1	0.0207453594445
16	0.0198591900865
147	0.0187997076424

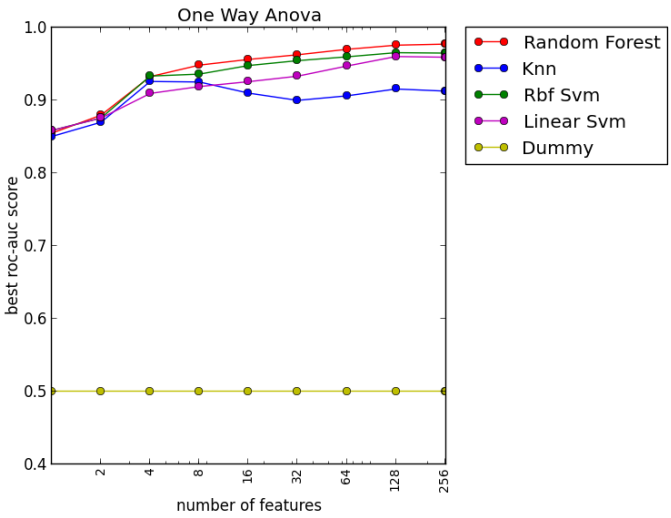
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
2	0.0906002297746
179	0.0301953852307
184	0.0297636000662
10	0.0289129759991
22	0.0272626329801
135	0.0267423012443
72	0.023376705017
154	0.0232911614741
5	0.0230838454747
73	0.0211607565219

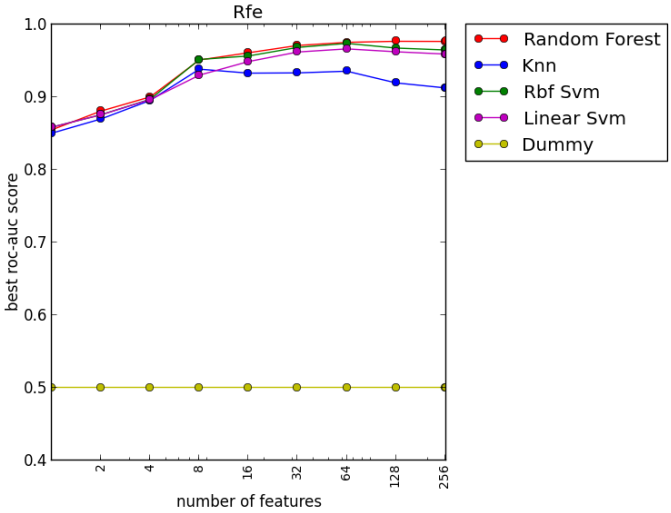
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
2	0.218894056319
5	0.0733822535761
171	0.0689257994257
4	0.0448052736124
22	0.022967608426
72	0.0221302423627
135	0.0209016044557
10	0.020783210577
147	0.017584631104
179	0.0144047087783

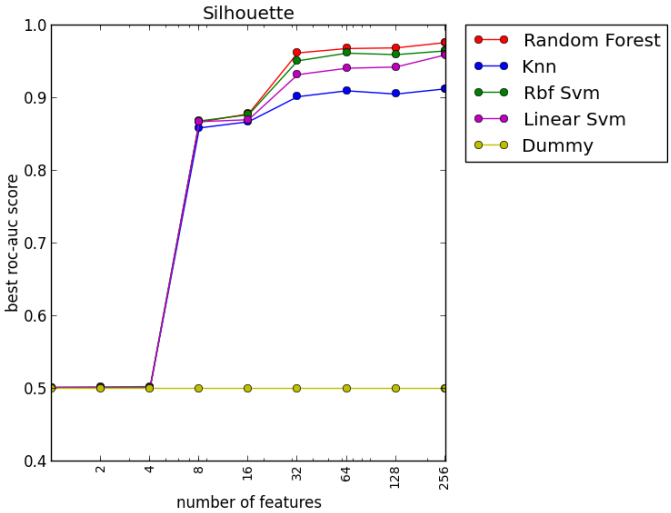
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
2	259.0
5	258.0
7	257.0
10	256.0
134	255.0
71	254.0
4	253.0
171	252.0
217	251.0
72	250.0

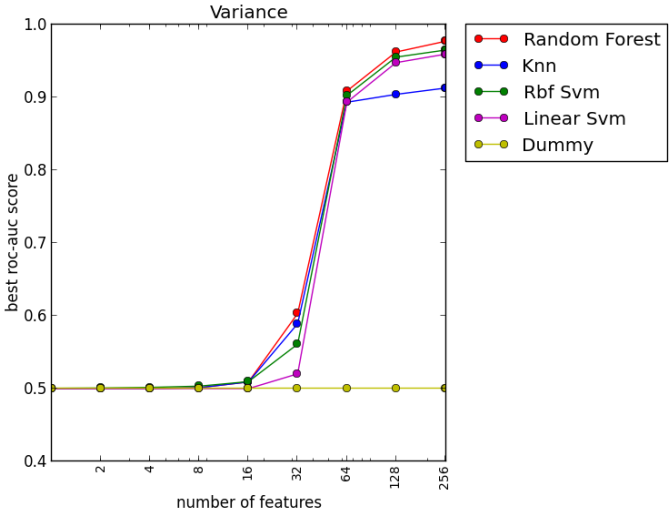
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
129	1.0
37	0.997432936663
57	0.997432936663
65	0.997432936663
181	0.997432936663
2	0.966005629678
134	0.865601205805
43	0.770851007093
109	0.74176462881
256	0.726007410968

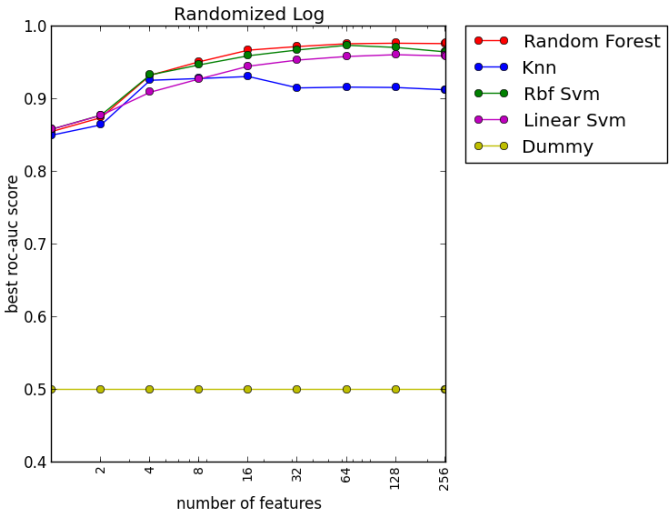
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
4	1.0
0	0.864009641119
129	0.835846682981
1	0.691339806498
22	0.680661053908
190	0.67891716729
185	0.633872230294
195	0.537945921745
3	0.52070357781
66	0.480472915019

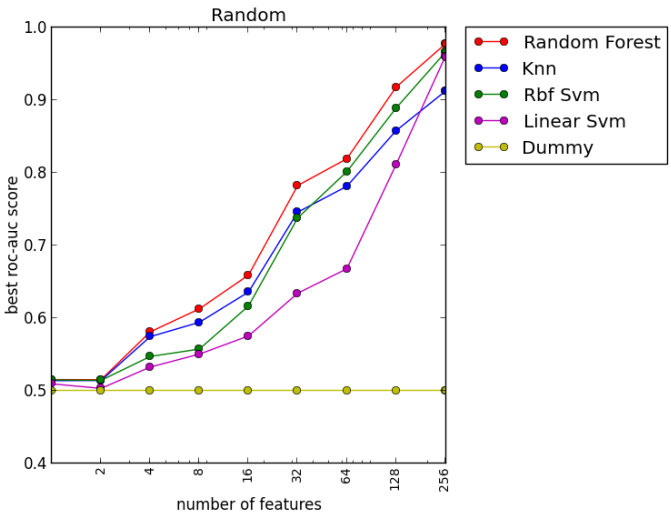
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
171	1.0
2	1.0
5	0.815
4	0.685
41	0.605
165	0.56
72	0.54
135	0.525
134	0.49
7	0.485

(b) Top 10 scores for randomized log



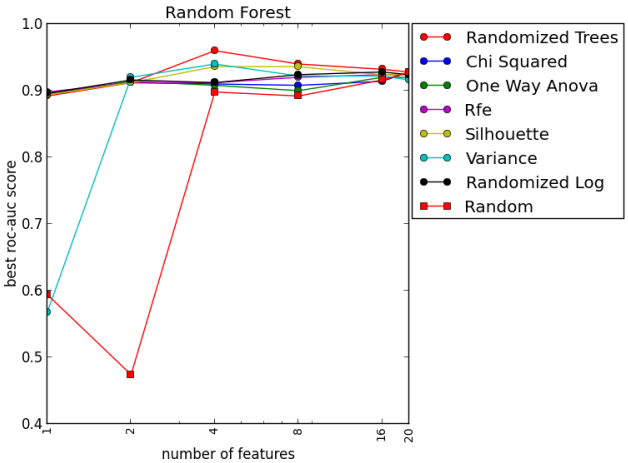
(c) Results for the random scorer

Name	Score
171	0.9936896345
154	0.993298824308
99	0.988808850919
61	0.98687016443
223	0.983048066936
33	0.980428988186
141	0.974389473964
103	0.974048980261
41	0.973403247005
219	0.958794452823

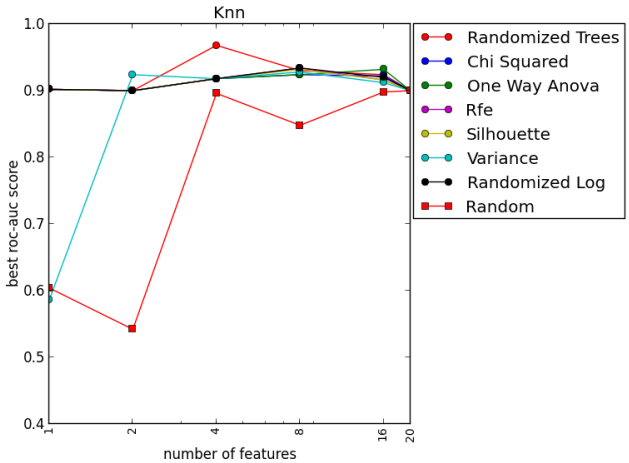
(d) Top 10 scores for random

Figure 36: Plots for each scorer for the Parasites dataset

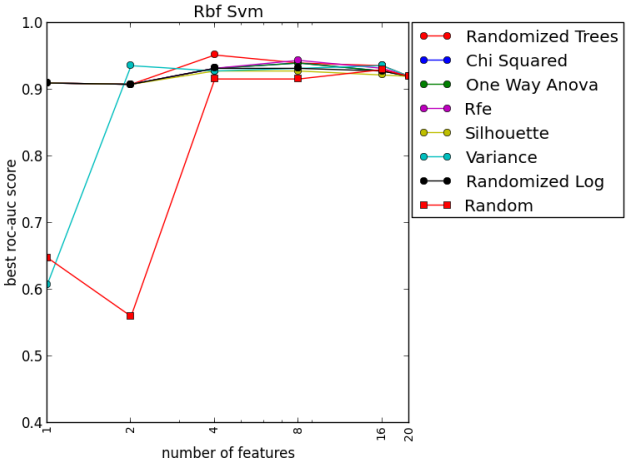
G Additional results for the Sksynth dataset



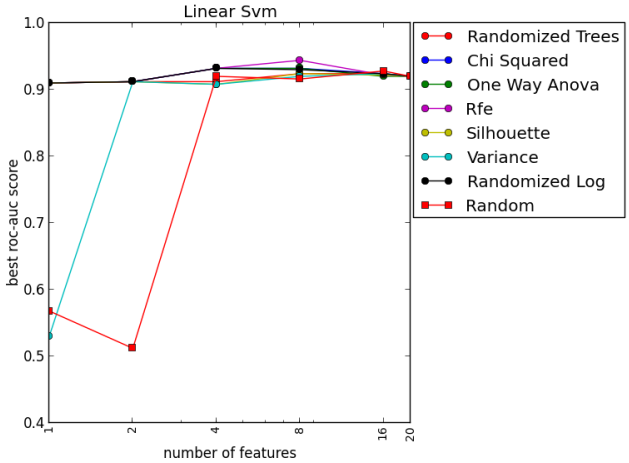
(a) Results for the Random forest classifier



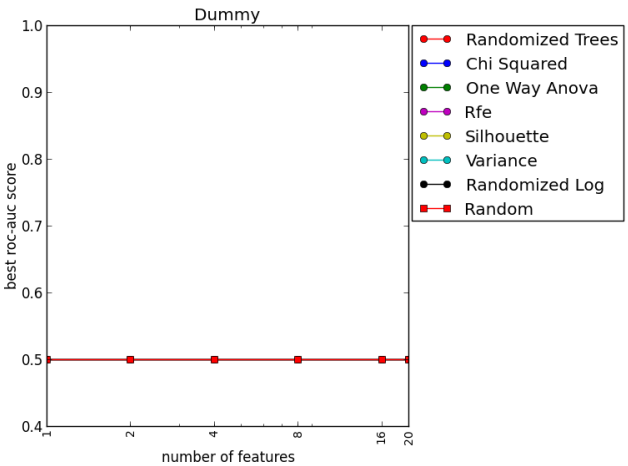
(b) Results for the Knn classifier



(c) Results for the Rbf SVM classifier

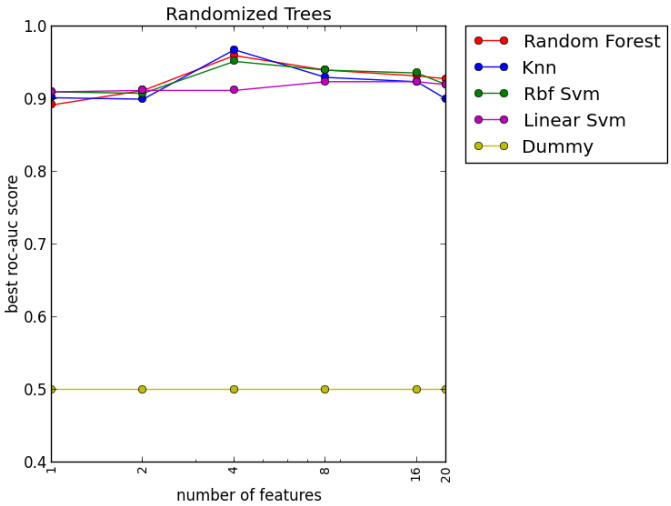


(d) Results for the Linear SVM classifier



(e) Results for the Dummy classifier

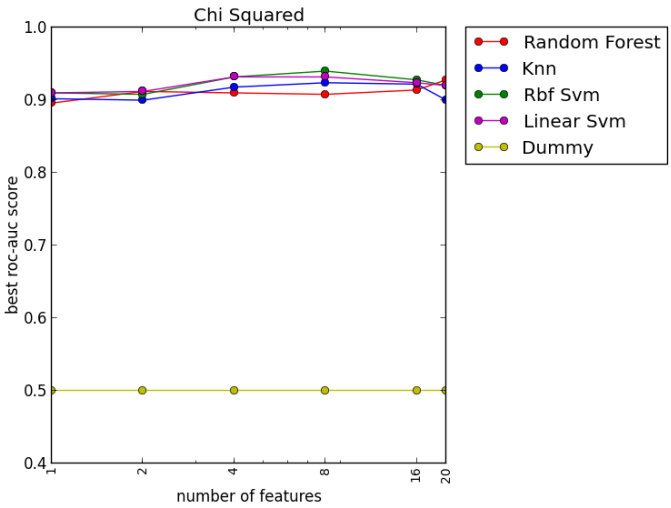
Figure 37: Plot per classifier for the Sksynth dataset



(a) Results for the randomized trees scorer

Name	Score
f6	0.167490262366
f4	0.159821609341
f3	0.141972769964
f19	0.0553401003623
f9	0.044002837317
f10	0.0346166614365
f8	0.0328801337595
f12	0.0318136354101
f13	0.0310454492882
f15	0.0304890403211

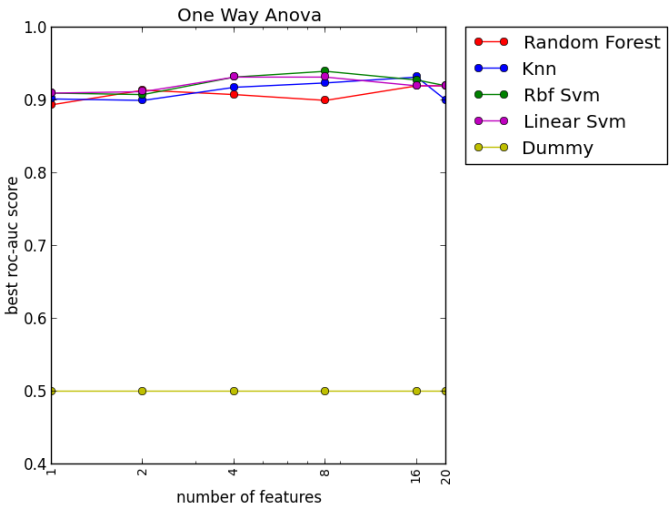
(b) Top 10 scores for randomized trees



(c) Results for the chi squared scorer

Name	Score
f6	0.335584411341
f4	0.330232223176
f3	0.260939633072
f10	0.0259015309361
f8	0.0119229583282
f1	0.00940787121124
f18	0.00801405315001
f13	0.00676639918151
f0	0.00339976737998
f7	0.0014506093438

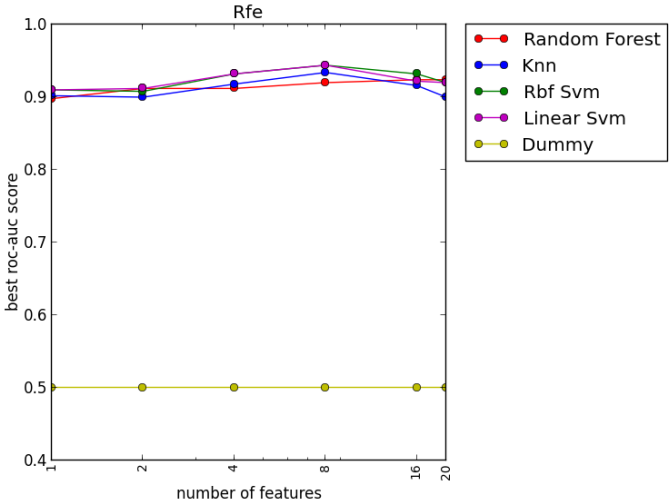
(d) Top 10 scores for chi squared



(e) Results for the One way anova scorer

Name	Score
f6	0.342706582815
f4	0.328880103192
f3	0.2839953226
f10	0.0121423825723
f8	0.0102026358022
f13	0.00553012152792
f18	0.00399759350745
f1	0.00363896641771
f0	0.00279733285744
f7	0.00143601340405

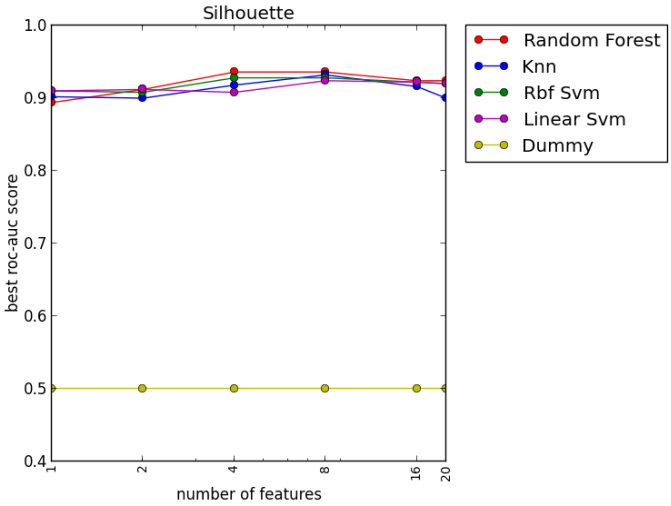
(f) Top 10 scores for One way anova



(a) Results for the rfe scorer

Name	Score
f6	19.0
f4	18.0
f3	17.0
f10	16.0
f8	15.0
f16	14.0
f15	13.0
f14	12.0
f7	11.0
f19	10.0

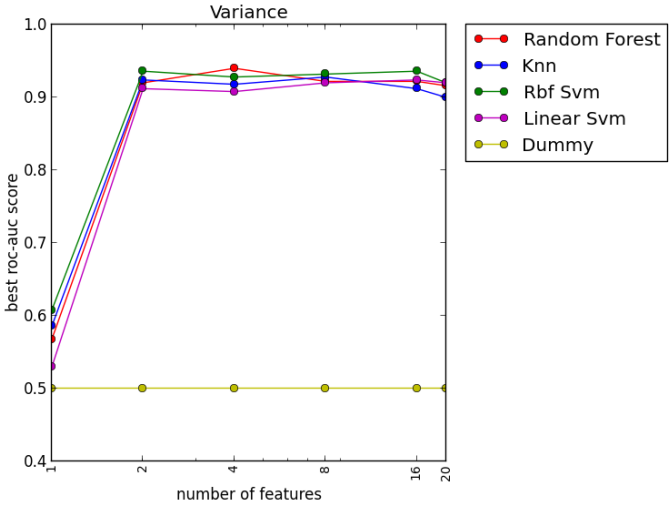
(b) Top 10 scores for rfe



(c) Results for the silhouette scorer

Name	Score
f6	1.0
f4	0.955653472322
f3	0.900586894446
f9	0.10354453289
f18	0.0806105293092
f19	0.0787184551866
f10	0.0784042740004
f8	0.0709801705659
f16	0.0496751441226
f11	0.0400356245532

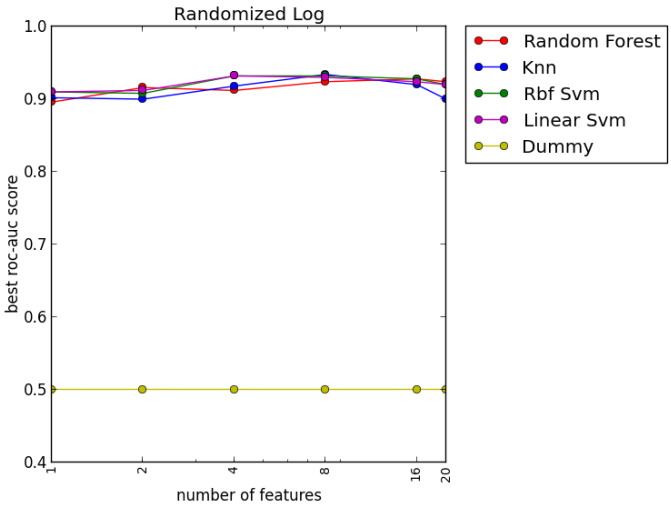
(d) Top 10 scores for silhouette



(e) Results for the variance scorer

Name	Score
f9	1.0
f6	0.990591833613
f3	0.968579509456
f4	0.930857665661
f1	0.5586981562
f16	0.553516444863
f11	0.553330266291
f12	0.539940329393
f10	0.497132845155
f13	0.485911809085

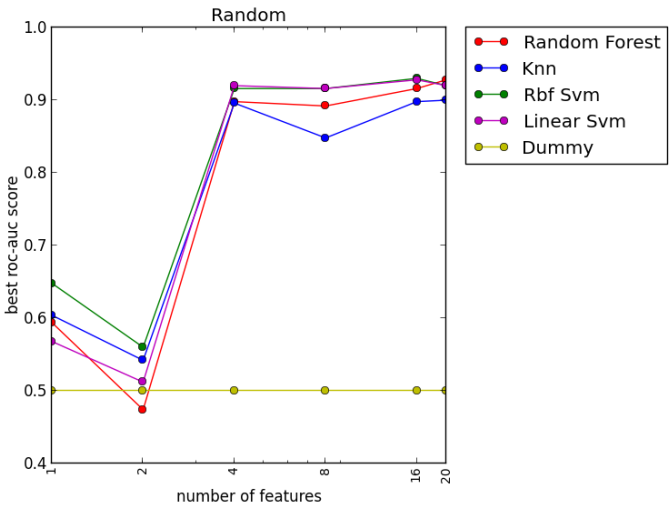
(f) Top 10 scores for variance



(a) Results for the randomized log scorer

Name	Score
f6	0.605
f4	0.37
f3	0.26
f10	0.005
f19	0.0
f8	0.0
f1	0.0
f2	0.0
f5	0.0
f7	0.0

(b) Top 10 scores for randomized log



(c) Results for the random scorer

Name	Score
f8	0.951885813117
f6	0.944003896319
f12	0.932801362359
f7	0.922374266388
f18	0.904031787533
f14	0.901298769978
f3	0.884545022676
f0	0.872187095895
f11	0.819584690988
f16	0.671799982283

(d) Top 10 scores for random

Figure 40: Plots for each scorer for the Sksynth dataset

H Performance results

```
Timer unit: 1e-06 s

Total time: 7547.29 s
File: classification.py
Function: evaluate at line 33

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    33                                           @profile
    34                                           def evaluate(X, y, settings, feature_names):
    35                                               # determine the indices of X belonging to class 1 and 2 repectively
    36         1           21     21.0      0.0      class_1 = np.nonzero(y == 0)[0]
    37         1           12     12.0      0.0      class_2 = np.nonzero(y == 1)[0]
    38         1           10     10.0      0.0      scaler = preprocessing.MinMaxScaler()
    39         1         2161   2161.0      0.0      X = scaler.fit_transform(X)
    40                                               # select all features
    41         1            5      5.0      0.0      feats = range(X.shape[1])
    44         1         1342   1342.0      0.0      cv = StratifiedKFold(y, n_folds = 10)
    46                                               # get all scorers and classifiers with corresponding labels and parameters
    47         1        13191  13191.0      0.0      scoring_techniques = settings.getScoringTechniques(X, class_1, class_2, feats)
    48         1           38     38.0      0.0      nr_features = settings.getNrFeatures()
    49         1          206    206.0      0.0      classifiers, classifier_labels = settings.getClassifiers()
    50                                               # calculate lengths
    51         1            2      2.0      0.0      scoring_techniques_length = len(scoring_techniques)
    52         1            2      2.0      0.0      nr_features_length = len(nr_features)
    53         1            2      2.0      0.0      classifiers_length = len(classifiers)
    55                                               # initialize dataObject
    56         1           84     84.0      0.0      data = do.DataObject(...)
    60         9           52      5.8      0.0      for i in range(0, scoring_techniques_length):
    61         8           53      6.6      0.0          scoring_technique = scoring_techniques[i]
    63         8     35685842 4460730.2      0.5          score = scoring_technique.score()
    64         8          208     26.0      0.0          sorted_indices = score.argsort()[::-1]
    66         8          368     46.0      0.0          features = zip(feature_names, score)
    68         8         1027    128.4      0.0          data.addFeatures(map(lambda i: features[i], sorted_indices))
    69        64          615      9.6      0.0          for j in range(0, nr_features_length):
    70        56          368      6.6      0.0              nr_feature = nr_features[j]
    71                                                       # select i best features
    72        56          753     13.4      0.0              columns = sorted_indices[0 : nr_feature]
    73        56        35521    634.3      0.0              X2 = X[ : , columns]
    74       336         3341      9.9      0.0              for k in range(0, classifiers_length):
    75                                                           # deep copy data so the classifiers array is not changed
    76       280       391442   1398.0      0.0                  (model, params) = copy.deepcopy(classifiers[k])
    77       280        78568    280.6      0.0                  clf = GridSearchCV(model, params, ...)
    78       280   7511058901 26825210.4    99.5                  clf.fit(X2, y)
    79                                                           # add data to data object
    80       280        12312     44.0      0.0                  data.addData(clf.best_score_, i, j, k)
    81
    82                                               # add labels to the datastructure
    83         1          128    128.0      0.0      map(lambda x: data.addFstLabel(x.getLabel()), scoring_techniques)
    84         1           57     57.0      0.0      map(lambda x: data.addNrFeaturesLabel(x), nr_features)
    85         1           45     45.0      0.0      map(lambda x: data.addClassifierLabel(x), classifier_labels)
    86         1            5      5.0      0.0      return data
```