Feature Reduction Method Comparisons with Regards to Statistical Performance, Memory Usage, and Interpretability

Eliot Shekhtman
Cornell University Ithaca, New York, USA ess239@cornell.edu

Pakin Wirojwatanakul
Cornell University Ithaca, New York, USA pw273@cornell.edu

Saharat Rodjanamongkol
Cornell University Ithaca, New York, USA sr865@cornell.edu

Abstract

In this paper, we evaluate the system performance and statistical performance of many of the common dimensionality reduction methods on multiple datasets with respect to interpretability, and determine the optimal memory and statistical performance tradeoff for each method and dataset. The results show that many data reduction methods can improve statistical performance or reduce the dimensionality without much reduction in performance, but no dimensionality method outperforms significantly across datasets.

1. Introduction

Dimensionality Reduction is an important component of the machine learning pipeline because using fewer features improves inference latency, which is important for time sensitive tasks like autonomous driving vehicles. Dimensionality reduction is also important because using fewer dimensions saves memory and is especially important for workflows that run on edge devices, which have low memory. In some cases, dimensionality reduction can even improve statistically performance by reducing noise.

This report is structured accordingly. Section 2 explains the dimensionality reduction methods used. Section 3 explains the metrics, datasets, and standardizations used to evaluate the methods. In section 4, we present the results of each of the dimension reduction methods on the four datasets. Finally, section 5 presents the final remarks and insights gained from this research.

2. Related Works

2.1 Random Forest Feature Selection

The Random Forest classifier is a set of decision trees with each tree trained on a subset of the data [1]. Each split of the decision tree occurs at the value of a single dimension that minimizes the impurity. The Random Forest classifier has a feature importance attribute that is calculated by averaging the decrease in impurity of the splits per feature. The Random Forest feature selection is done by selecting a subset of most important features based on the feature importance attribute. The Random Forest Feature Selection has a high degree of interpretability because it selects a subset of original features instead of constructing new ones.

2.2 Lasso Feature Selection

The Lasso Classification is a linear classifier with weights calculated by minimizing the squared loss and L1 regularization [8]. The alpha constant dictates the importance placed on the L1 regularizer. The larger the alpha the sparser the weight vector and the fewer the value of selected features as demonstrated in Figure 1. Lasso Feature Selection is done by selecting the entries of the weight vector with a non-zero value for a given alpha value. The Lasso Feature Selection has a high degree of interpretability because it selects a subset of original features instead of constructing new ones.

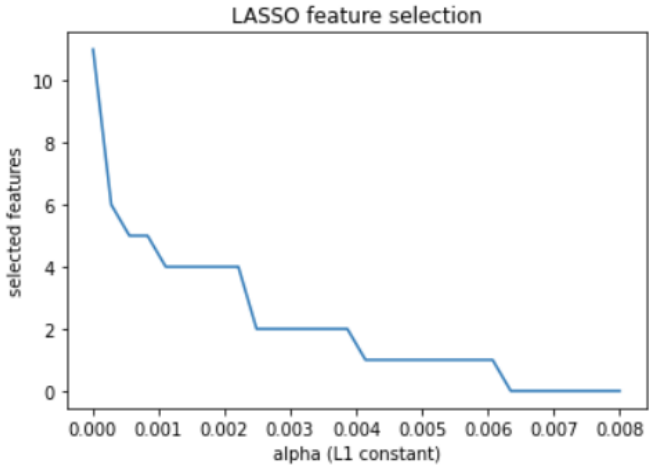


Figure 1: The number of features selected as a function of alpha for the Wine Dataset.

2.3 Control Burn Feature Selection

The Control Burn Classifier first creates an ensemble of trees using the Incremental Depth Bag Boosting Method. Then it uses LASSO to remove the trees resulting in a subset of trees that do not use all the features [4]. The Control Burn Feature Selection method is done by selecting a subset of the features based on the feature importance from the subset of features that the Control Burn Classifier uses. The Control Burn Feature Selection has a high degree of interpretability because it selects a subset of original features instead of constructing new ones.

2.4 Principal Component Analysis

Principal Component Analysis, or PCA, is an unsupervised method that calculates the principal components of a matrix and performs a change of basis on the data [2]. A common usage of PCA is in dimensionality reduction, where dimensions of the change of basis corresponding to small principal components may be zeroed such that output representations of data have a smaller feature space. While

PCA will minimize the loss of information while projecting down, it is unfortunately uninterpretable as each output feature corresponds to a linear combination of all input features. Some methods exist attempting to limit the number of input features per output feature, but results are still inconclusive.

2.4.1 SubsetPCA

From PCA, we developed SubsetPCA as a method to increase interpretability of PCA. A pairwise correlation matrix is calculated on all input features, and then features with the greatest correlation magnitude are grouped into subsets until there exists a partition of the feature space with 10 disjoint connected components. PCA is then run on each component/grouping of correlated features outputting only the dimension with the highest principal component, resulting in a new dataset where each feature represents a linear combination of a disjoint subset of the original feature space. This allows only the most related features to merge, giving interpretability to the output dataset.

We reason that this might give interesting results as it will be more interpretable than PCA, with each output feature only incorporating information from a disjoint subset of the feature space and the transformation required to create it will be less complex as a result. Furthermore, it has different performance than the supervised methods as instead of deleting features, it attempts to keep as much of the original variance as it can with respect to the original feature space, meaning that output features might retain useful information lost during feature deletion.

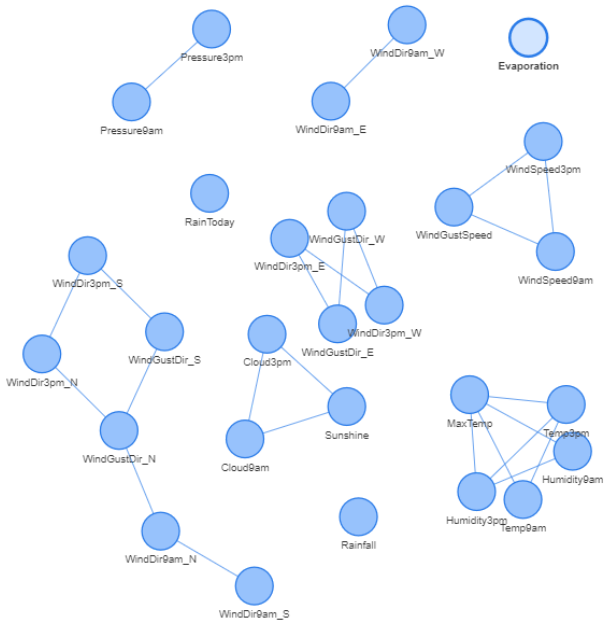


Figure 2: Partition of the WeatherAUS feature space to create 10 output components for SubsetPCA. Nodes represent features, edges represent connections made between highly correlated feature pairs.

First, we may see that the most natural metric for a similarity score between features would seem to be the first principal component of PCA conducted on two features divided by the second principal component; however, since the principal components are chosen to maximize the variance, this is effectively an equivalent metric to the absolute value of correlation [6]. This can be visually reasoned by considering the meaning of correlation and the meaning behind having a large ratio between the first and second principal components of PCA conducted on two features.

One issue with this method is that resulting feature subsets are created greedily on this metric; an example can be seen with the kite structure in Figure 2, where WindDir3pm_S has limited guarantees on similarity with WindDir9am_S. Unfortunately, a true optimal partition involves a solution to the clique problem which is NP-complete, and so would be intractable to solve for large feature spaces [3]. The greedy solution represents an approximation of the true solution.

2.5 Feature Hashing

The Feature Hasher implements the Hashing Trick [10] in which it takes in a dictionary of feature and value and hashes the features into a sparse matrix with the specified number of features buckets. The values will get added together when collision happens. Dimensionality reduction comes from the property that the number of hashed features buckets can be lower than the number of original features. It is used to create a space efficient feature vector especially for sparse high dimensional features such as text bags of words. Research has shown that using the hashed data would result in little loss of performance and also preserve the sparsity of the original data; however, since the data is hashed and can have collisions, the data is not interpretable and is uninvertible. Do note that the data hashed are still hashed even if the size of the feature bucket equals the original feature. This causes the prediction at 100% of the feature size to differ from the full dataset.

Dict Vectorizer was also explored. It is similar to Feature Hashing but it couldn't hash data to arbitrary numbers of feature buckets and can be inverted back to original data. From our experiments, its performance is almost identical to the full unmodified data but it doesn’t provide direct dimensionality reduction so it wasn’t pursued further. There might be some space benefit for sparse data because the output is a sparse matrix.

2.6 Nonlinear dimensionality reduction

We also explore nonlinear dimensionality reduction methods such as Kernel PCA and Manifold learning. Kernel PCA is an extension of PCA where it uses a kernel trick to map the features to a higher nonlinear feature space and perform dimensionality reduction in the new feature space. On the other hand, manifold learning tries to find non-linear structure in the data, for example data lying on a S curve structure. For manifolds, we explore Isometric Mapping (Isomap) [7],

Multidimensional Scaling (MDS) [5] and t-distributed Stochastic Neighbor Embedding (t-SNE) [9].

The result we found is that the nonlinear methods generally used too much memory and were too slow to run. t-SNE didn't have these issues; however, it doesn't support reducing to dimensions above 3. The higher dimensionality is likely the cause of the signification slow down. Manifold learning usually reduces the data for the purpose of visualization instead of performance so it might not be suitable for our purpose. Furthermore, the messy nature of the data might also be unsuitable for the algorithm because there's no clear structure behind it. Nonlinear dimensionality reduction wasn't explored further.

2.7 Lower Float Bit Precision

If precision is not a major concern, using a 16 bit float (fp16) instead of a 32 bit float (fp32) is a simple way to reduce the memory usage by half. However, if the data values are too large this might lead to overflow because fp16 uses 5 exponential bits instead of 8 exponential bits used by fp32.

3. Methodology

3.1 Dimensionality Reduction Method Comparison

The different dimensionality reduction methods were evaluated on four different datasets with different degrees of sparsity and two classes of classifiers, linear and tree based. For each dataset and classifier type, we varied the numbers of features selected to evaluate the tradeoff between memory usage and statistical performance, measured by the accuracy and the F1 score. For each dataset, eighty percent was used for training and twenty percent was used for testing.

3.2 Data Processing

We converted non-continuous features to numeric (boolean features converted to 0 and 1, directions converted to many-hot vectors, etc). Features with high missing value counts (for example, evaporation and sunshine in WeatherAUS) had missing values filled in with the means. StandardScaler was used to normalize the mean and standard deviation of every feature.

4. Experimental Results

4.1 Adult Dataset

The Adult dataset has 50 partially sparse demographic features and 20,000 data points. The goal is to predict if a person makes more than fifty thousand dollars per year.

4.1.1 Random Forest

The RF Feature selection method can find a subset of features that performs better than the complete set of features for both linear and tree based classifiers, but oscillates in performance.

4.1.2 Lasso

For tree based classifiers,the Lasso feature selection method can find a subset of features that performs better than the subset of features selected by random forest. The 25 features selected by LASSO outperformed the complete set of features whereas the 25 features selected by RF merely performed the same as the complete set of features.

4.1.3 Control Burn

The subset of features selected by control burn performed worse than the full set of features.

4.1.4 Principal Component Analysis

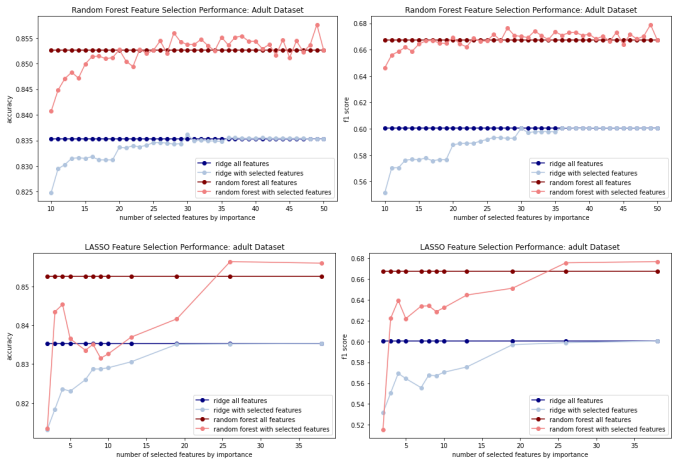
Overall PCA can be seen to have consistent results for the dataset, and achieves high accuracy and F1 scores with only 4 features, indicating that only 4 dimensions in the transformed space truly hold significant variance from the feature set. It outperforms many other methods such as Feature Hashing and Random Forest at this dimensionality; however, it fails to achieve the same or higher accuracy as the full feature set like supervised methods are able to due to its unsupervised nature.

4.1.5 Subset PCA

The subset version can be seen to perform poorly when less than half the original feature dimensionality is used, likely due to the greedy clustering algorithm creating suboptimal feature partitions. This is further exemplified by the step-function nature of its degradation as the number of features decreases, each jump likely due to two clusters with high intra-cluster but low inter-cluster correlation being connected. It may be noted that subset PCA outperforms PCA and achieves higher accuracy and F1 than the full feature set when over half the original feature dimensionality is used, likely due to the greedy algorithm producing near-optimal clusters at that stage.

4.1.6 Feature Hashing

Feature hashing performed worse than full dataset and already has a substantial drop in performance at around 80% of the original size. The ridge classifier has a constant drop in performance as the number of features decreases while the random forest classifier only has a minor performance drop past 80%. The feature selection with hashing performs slightly better than hashing only. The performance of hashed Lasso features varies a lot, with improvements at 40% of the data size and a dip at 60% of the data size.



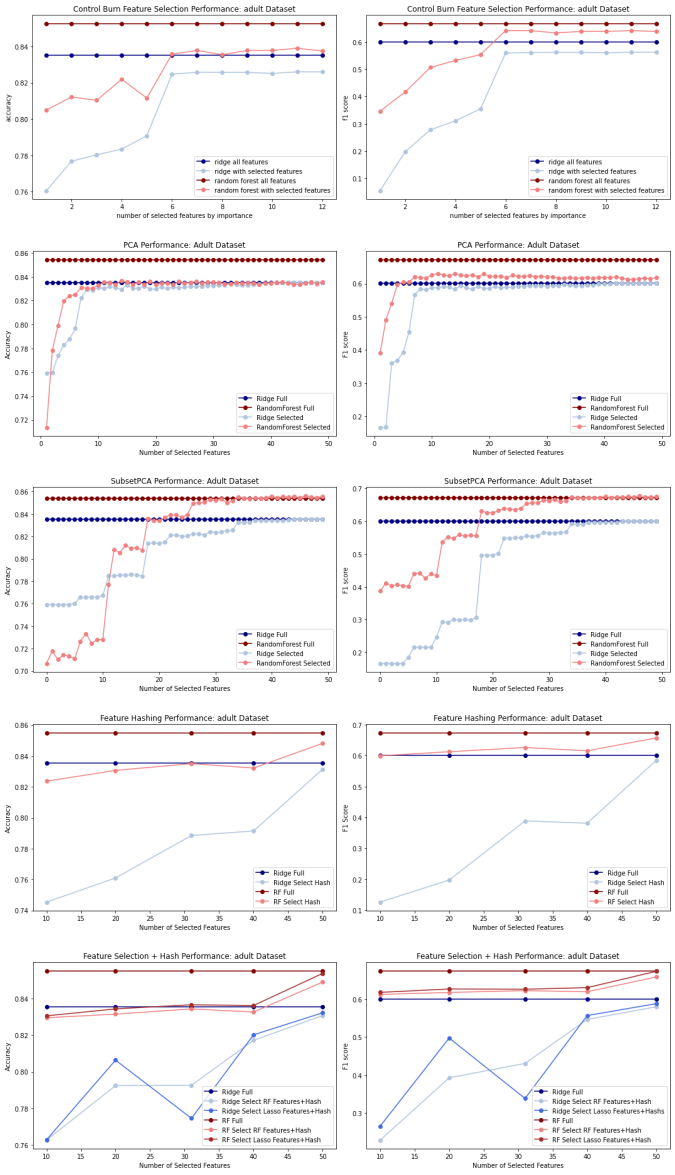


Figure 3: Performance comparison between different algorithms on the Adult Dataset

4.2 Titanic Dataset

The titanic dataset has 21 partially sparse features and 1309 data points. The goal is to predict if a person survives.

4.2.1 Random Forest

The RF Feature selection method can find a subset of features that performs better than the complete set of features for both linear and tree based classifiers. The two most important features are sex_male and sex_female. Performance steadily degrades for Random Forest classification as the number of features increases, indicating that many of the features add noise; however, performance for Ridge classification doesn't exhibit this behavior, instead increasing as the number of features increases until 15 features are reached, then decreasing to the baseline.

4.2.2 Lasso

The six features selected by Lasso outperformed the six features selected by RF for Random Forest classification. At six features, RF performed worse than using all the features, but Lasso performed better. A similar trend as RF is found for Ridge classification.

4.2.3 Control Burn

CB selected features that performed much worse than Lasso and RF for low feature counts. The top three selected features were CabinType_NaN, CabinType_C, and Pclass, which were not important as demonstrated by the low accuracy.

4.2.4 Principal Component Analysis

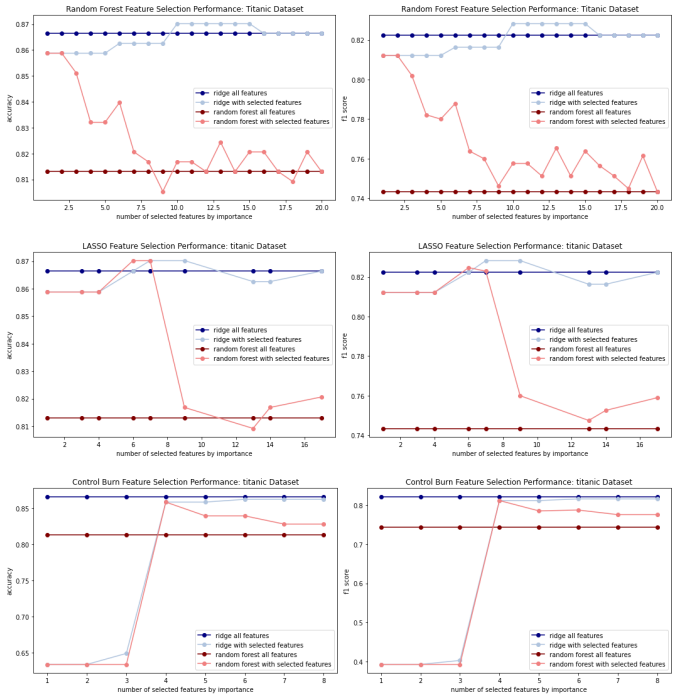
Once again we see that PCA exhibits consistent performance and effectively represents the most of the feature space in two transformed dimensions; however, downstream methods fail to achieve higher performance than on the full dataset as PCA isn't able to optimize its transformation with respect to the label space or a prediction task.

4.2.5 Subset PCA

Once again we can observe that while subset PCA performs worse than PCA for large levels of dimensionality reduction, it outperforms PCA and other unsupervised methods for lesser levels of reduction but still fails to optimize for the prediction task. The step-function performance degradation is also clearly visible.

4.2.6 Feature Hashing

The Titanic dataset's trend differs from other datasets. Usually random forest performs better but here ridge performs better. Also hashing the features to 60% the size of original and random forest selected features, ridge classifier could perform as well as full data. Hashing features selected by Lasso perform better than full data in random forest classifiers. This might be due to the low number of important features as seen by the other algorithm, so noises from the collision doesn't affect the performance as much.



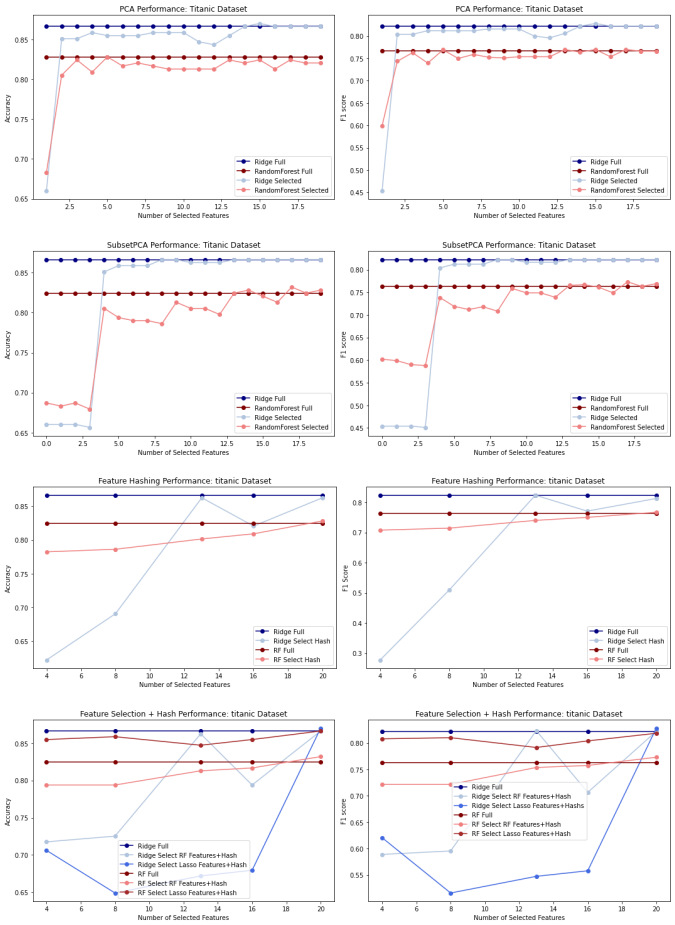


Figure 4: Performance comparison between different algorithms on the Titanic Dataset

4.3 WeatherAUS Dataset

The Weather AUS dataset contains 29 partially sparse features and 20,000 data points. The goal is to predict if it will rain the next day in Australia given a set of weather-related features collected the day before. The original dataset was imbalanced so subsampling was done to balance out the classes.

4.3.1 Random Forest

RF was able to select a subset of features that performed better than the whole set of features for tree based classifiers.

4.3.2 Lasso

Lasso was unable to find a subset of features that performed better than the complete set of features.

4.3.3 Control Burn

CB was unable to find a subset of features that performed better than the complete set of features.

4.3.4 Principal Component Analysis

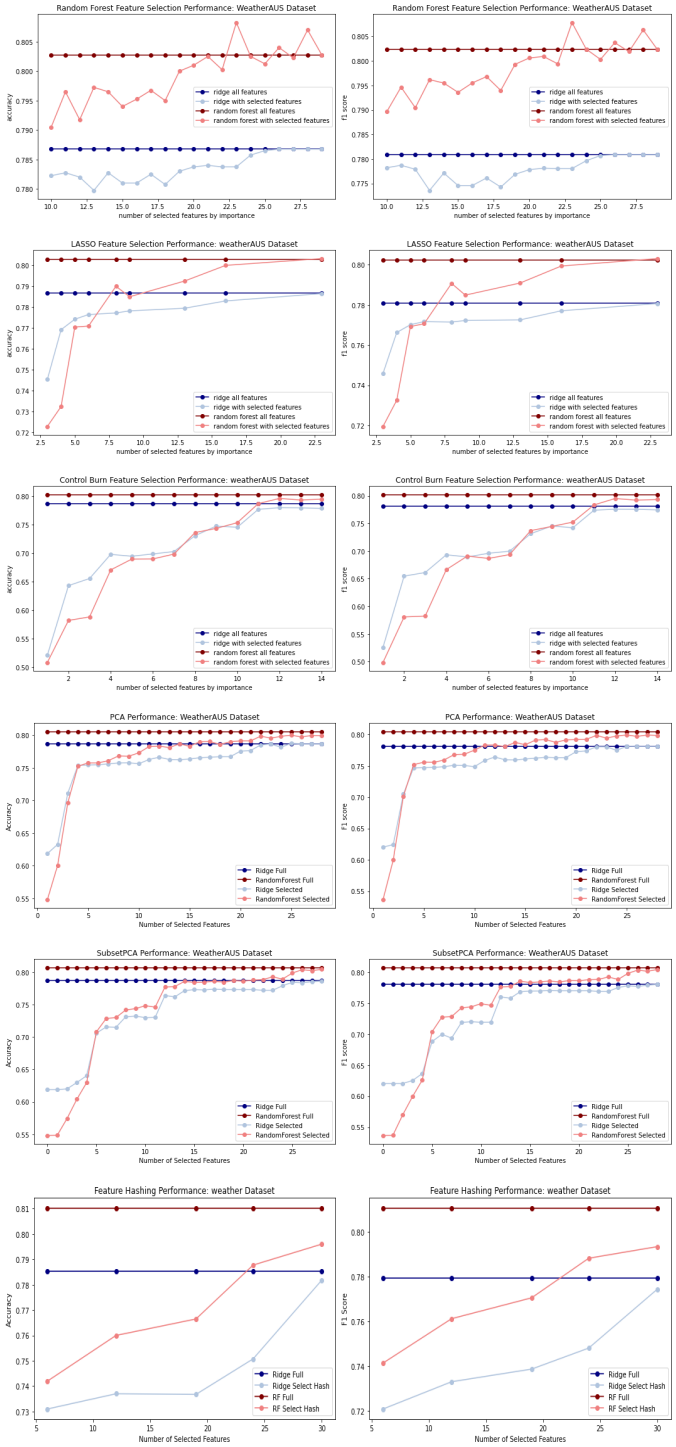
PCA exhibits similar behavior to previous trials here, representing most of the variance in the dataset with only 4 output features but failing to outperform supervised methods.

4.3.5 Subset PCA

For this dataset, subset PCA performs similarly to PCA, once again underperforming when the dimensionality reduces by a factor of two but with less severity than earlier datasets. This indicates the greedy algorithm performed more optimally than in other datasets.

4.3.6 Feature Hashing

Here, feature hashing follows the general trend. It performs worse as the number of features decreases. The performance of hashed Lasso features also varies a lot, with improvements at 40% of the data size and a dip at 60% of the data size.



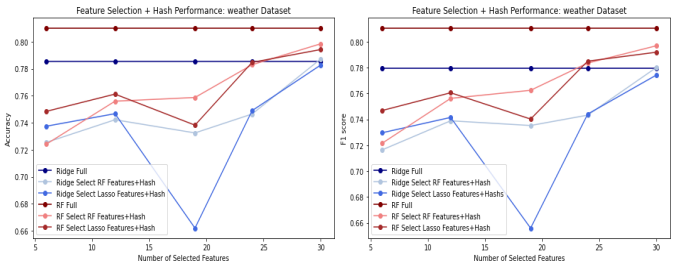


Figure 5: Performance comparison between different algorithms on the Australian Weather Dataset

4.4 Wine Dataset

The Wine dataset has 11 dense features relating to the chemical properties and 1279 data points. The objective is to predict wine quality.

4.4.1 Random Forest

RF was able to select a subset of features that performed better than the whole set of features for Ridge classification.

4.4.2 Lasso

Lasso was unable to find a subset of features that performed better than the complete set of features, but generally performed similarly to RF.

4.4.3 Control Burn

CB was unable to find a subset of features that performed better than the complete set of features. It performed worse than the other two supervised methods for small feature counts; however, it outperformed the others as the number of features reached full.

4.4.4 Principal Component Analysis

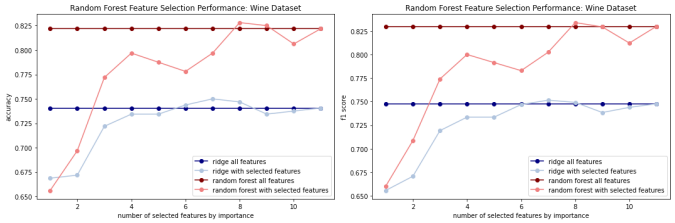
PCA performs similarly to previous datasets, performing worse than supervised methods but reducing the data efficiently.

4.4.5 Subset PCA

Subset PCA once again slightly outperforms PCA for large feature dimensionalities but performs poorly for small dimensionalities.

4.4.6 Feature Hashing

Feature hashing has a slight bump around 40% of the feature size and almost has the same F1 score as the full feature set. The combination of feature selection and hashing has a big dip at 80% of the feature size and a big performance boost at 60% of the feature size and performs almost as well as the full feature set on a ridge classifier.



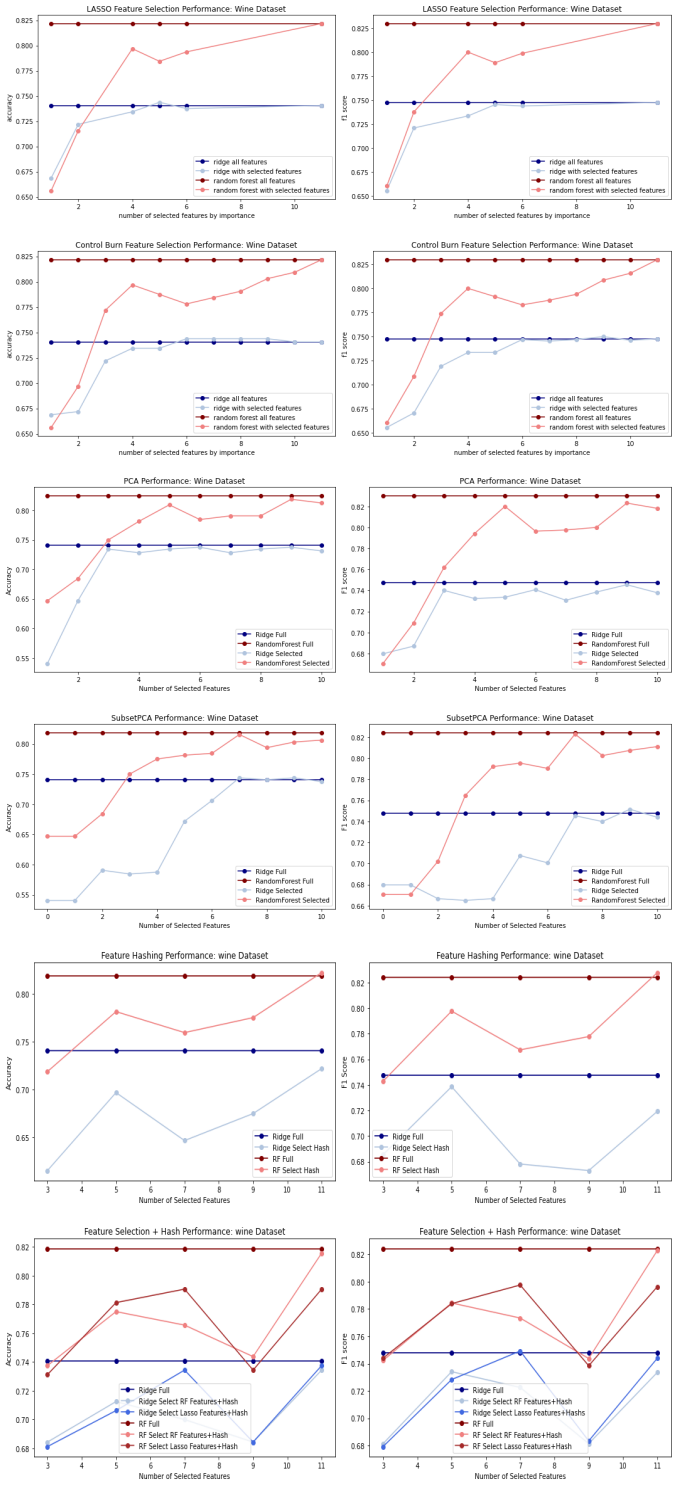


Figure 6: Performance comparison between different algorithms on the Wine Dataset

5. Conclusions

Random Forest and Lasso feature selection could consistently find a subset of features that improves the performance of both tree and linear based classifiers by removing noisy features. The features selected by Random Forest outperformed the features selected by Lasso on the Adult and Titanic dataset, but performed worse on the WeatherAUS and Wine datasets.

Control Burn feature selection could reduce some of the features and maintain good performance; however, it usually just performs as good or worse than the full data set and only sometimes finds a subset of data that performs slightly better than the full data. It generally performs worse than features selected by random forest.

PCA can reduce a significant number of features while maintaining a good performance; however, most of the time the reduction results in performance that matches the full data set and doesn't improve on it. Furthemore, the data becomes uninterpretable.

Feature hashing can reduce a significant number of features and maintain decent performance. On some datasets, it could match the performance of the full dataset. But in general it performs worse than full data and also performs worse than other algorithms. Combining feature selection and hashing slightly improves the performance compared to just feature hashing by itself. However, it still performs worse than full data and worse than other algorithms. The improvement might come from having less features and therefore less collision. The overall poor performance of feature hashing could be because the datasets we used are not suitable for hashing. While some of our datasets are sparse and have a lot of features, the dimension isn't as high and data isn't as sparse as the intended dataset of the algorithm such as text bag of words.

As mentioned earlier, nonlinear approaches are very slow to run and might not be suitable for data reduction with respect to performance.

We can see that No Free Lunch Theorem applies to our dimensionality reduction methods. While most of the time features selected by Random Forest perform best, sometimes features selected by Lasso could be smaller and maintain better performance. This indicates that there isn't necessarily a single true optimal algorithm, but rather that different algorithms may have different use cases on datasets.

We would currently recommend the company to implement feature dimensionality reduction using Random Forest and Lasso to select the most relevant features because of their ability to not only reduce the number of features, but also to improve on the performance of the classification while maintaining the interpretability of the data. Furthermore, with reduced number of features, the model could make prediction faster, save on cost of gathering and storing unnecessary data, and reduce memory requirement so that lower end edge devices with low memory can run our model.

Our recommended action item is to first use Random Forest and Lasso to get a number of features subsets to identify useful and necessary features. The second action item is to train the current models with different features and see which subset is the smallest while maintaining the same or better performance as the full dataset. The last action item is to deploy the new model with a smaller feature set and deprecate the gathering and processing of the unused features.

6. Future work

First, we would like to evaluate these methods on more datasets and possibly generated synthetic datasets to achieve greater guarantees about performance, and evaluate on more metrics than Random Forest and Ridge Regression classification to prevent possible bias in our metrics for algorithm selection towards solely these forms of classification problems. All of this would give more data collection opportunities, and we would like to find if there is statistical significance behind the differences between these algorithms at various levels of dimensionality reduction. Possible data collection on slower methods such as SubsetKernelPCA would also be beneficial to examine the performance of nonlinear methods.

Another objective is to determine a computational bound on the optimality of the greedy algorithm for feature subsetting in the SubsetPCA algorithm. This would help determine the utility of implementing the true optimal algorithm considering runtimes versus performance improvement. This bound would consist of first finding a bound on the correlation between two features each connected to a common highly-correlated feature (for a given minimum correlation to create an edge), and hopefully expanding this to a general case with respect to a given number of initial features and desired output dimensionality.

Finally, it is known that supervised methods for feature selection perform primarily by removing noisy features from the feature space, while unsupervised methods perform by transforming the feature space into a lower-dimensional space that attempts to capture as much information from the original distribution (the variance) as possible, collapsing correlated and redundant features. We would like to examine how the combination of these methods performs, in a joint algorithm that seeks to remove features not useful for prediction but merge correlated features to ensure possibly useful information isn't lost.

7. Works Cited

1. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001
2. K. Pearson, “On lines and planes of closest fit to systems of points in space”, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Sixth Series, 2, pp. 559-572 (1901)
3. Karp, R. (1972). Reducibility among combinatorial problems. In R. Miller & J. Thatcher (ed.), Complexity of Computer Computations (pp. 85--103). Plenum Press .
4. Liu, B., Xie, M., & Udell, M. (2021, August 1). *Controlburn: Feature selection by sparse forests*. ControlBurn | Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. Retrieved December 3, 2021, from https://dl.acm.org/doi/pdf/10.1145/3447548.3467387.
5. “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis” Kruskal, J. Psychometrika, 29, (1964)
6. Principal Component Analysis. UCLA: Statistical Consulting Group. from https://stats.idre.ucla.edu/spss/output/principal_components/ (accessed August 22, 2016).
7. Tenenbaum, J.B.; De Silva, V.; & Langford, J.C. A global geometric framework for nonlinear dimensionality reduction. Science 290 (5500)
8. Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the lasso". Journal of the Royal Statistical Society.
9. van der Maaten, L.J.P.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9:2579-2605, 2008.
10. Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009, June). Feature hashing for large scale multitask learning. In Proceedings of the 26th annual international conference on machine learning (pp. 1113-1120).