2017 IEEE International Conference on Data Mining

AutoLearn - Automated Feature Generation and Selection

Ambika Kaul*, Saket Maheshwary*, Vikram Pudi
Data Sciences and Analytics Center, Kohli Center on Intelligent Systems
International Institute of Information Technology, Hyderabad, India
{ambika.kaul, saket.maheshwary}@research.iiit.ac.in, vikram@iiit.ac.in

***Abstract*—In recent years, the importance of feature engineering has been confirmed by the exceptional performance of deep learning techniques, that automate this task for some applications. For others, feature engineering requires substantial manual effort in designing and selecting features and is often tedious and non-scalable. We present AutoLearn, a regression-based feature learning algorithm. Being data-driven, it requires no domain knowledge and is hence generic. Such a representation is learnt by mining pairwise feature associations, identifying the linear or non-linear relationship between each pair, applying regression and selecting those relationships that are stable and improve the prediction performance. Our experimental evaluation on 18 UC Irvine and 7 Gene expression datasets, across different domains, provides evidence that the features learnt through our model can improve the overall prediction accuracy by 13.28%, compared to original feature space and 5.87% over other top performing models, across 8 different classifiers without using any domain knowledge.**

I. Introduction

Data scientists very often find that a central step in their work, is to implement an appropriate transformation restructuring the originally given data into a new and more revealing form. Although specific domain knowledge can be used to help design representations, data-driven learning with generic priors can also be used, and the quest for AI is motivating the design of more powerful representation-learning algorithms implementing such priors. There are two goals in analyzing the data as discussed in [1]:

- Inference - To develop stochastic models which fit the data, and then make inferences about the data-generating mechanism based on the structure of those models.
- Prediction - To be able to predict what the responses are going to be to future input variables.

'Predictive Modeling' prefers to discuss only accuracy of prediction made by different algorithms on various data sets, but the crucial methodology here, driving success is the predictive culture's secret sauce of *feature engineering* [2].

Over the last few years, the impressive success of deep learning techniques, in the context of image [3], text, audio [4] and video data has further confirmed the importance of feature learning. These techniques, however, are effective when extremely large amounts of training data and intensive computational resources are available. Generally, feature engineering requires substantial manual effort in designing and selecting features and is often tedious and non-scalable. Learning optimal feature representation is not a trivial task as it requires manually identifying complex patterns from data, and a common practice to deal with such a problem is to use all available attributes and entrust the learning model to select the relevant features. However, an approach like this does not always work well. Most of the time, dealing with such large feature space proves to be ineffective as it raises computational complexity when only a small number of features are actually useful. Many a times, construction algorithms employ construction methods, that are task-specific and serve well to their underlying representation. There are, however, several problems with this approach. For instance, given a new classification problem, it is not obvious which of the various representations and associated algorithms should be selected. It is possible that none of the existing schemes is the right one for the problem at hand and in many real-world classification problems, the target concept is best expressed by features constructed using domain-specific knowledge. The existing algorithms are stringent and thus, does not allow for easy altering of the representations.

In the case of supervised learning, this problem can be formulated as using training examples to find a function $f : X \rightarrow Y$ that maps objects $x \in X$ to labels $y \in Y$ with high accuracy. Conceptually, the problem is broken into 2 steps [5]:

- Learning a feature representation $\phi : X \rightarrow F$ which maps the input terms $x \in X$ to feature vectors $\phi \in F$.
- Defining a class C of functions $c \in C$ with $c : F \rightarrow Y$ and learning the optimal function.

The prediction task at hand, thus, strongly influences the choice of feature representation ϕ. This also influences the predictive power of the learning classifier. In predictive modeling, the input item x has traditionally been identified with its feature representation $\phi(x)$, and often the second of the two problems above has been considered in isolation.

In this paper, we propose a learning model to alleviate difficulties involved in feature engineering through automation. Our contributions are as following:

- We develop a learning model, based on regression between feature pairs, that discovers *underlying patterns* and its *variations* in the data, by the way features are related to each other and selects a very small number of new features to create a significant improvement in predictive performance.

* Authors contributed equally.

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.31

- We present a novel method for feature generation, that captures the prominent variations in feature pairs that result in highly discriminative information.
- We experimentally demonstrate the merits of our approach on a large group of datasets and multiple classifiers, achieving on an average 13.28% improvement in the prediction accuracy, compared to the original feature space.

The usefulness of our features is intuitive and demonstrated in our experiments. We do not however, claim that our features are complete and there may be scope for further improvements in future. The organization of the rest of the paper is as follows: Section II reviews related work; Section III introduces the formal definition of the problem statement; Section IV provides a detailed description of our proposed model along with the intuition and the reasons why we think our approach performs better than the others; Section VI presents the datasets and the evaluation procedure used to validate our method.

II. Related Work

The classification algorithms receive as input, a set of attributes of the classified objects. These are, however, in many cases, not sufficient for creating an accurate and comprehensible representation of the target concept. To overcome this problem, a number of feature learning models have been proposed, such as [6], [7], [8], that use two step batch feature construction to transform the original features. For instance, the proposed model in [6] uses prior domain information and explanation-based interaction of training examples to construct distinguishing features, that are task-relevant. The work in [7] presents an iterative feature generation algorithm, known as TFC framework, that exhausts all obtained features and then selects the best ones using information gain. However, exhaustive search leads to combinatorial explosion of feature space making this approach non scalable. To avoid exhaustive search, learning models based on decision trees, such as [9], [10], [11] have been proposed. The approach in [9] leverages the initial bias calculated on the original feature set to generate discriminative attributes. These methods, however, do not generalize well for other classifiers. The framework proposed in [12] improves over [9], but requires domain proficiency to select feature constructors. The work in [13] proposed FCTree, where new features were learned using decision trees as a number of several sequential transforms of the original feature space without any domain knowledge to improve classification performance. ExploreKit (EK) [14] has the similar goal of automatically generating and selecting new features to improve classification performance using machine learning.

The majority of above feature construction algorithms have been specifically designed to generate features of a rigidly predefined representation. Among the popular representations are simple Boolean expressions, M-of-N expressions, hyperplanes, logical rules and bit strings. Each of these representations was shown to be beneficial in specific classes of problems. For example, it was shown that M-of-N expressions are particularly useful for medical classification problems where expert systems make use of criteria tables that are essentially M-of-N concepts [15].

While some classification problems may require a combination of several representation schemes, this is difficult to automate with existing feature learning algorithms. In this paper, we use regularized regression models to automate feature construction and selection without any domain knowledge such that, its improved accuracy maintains over a large number of classification algorithms. Our proposed approach selects a very small number of new features to reach the desired performance, while models proposed in prior literature require thousands of new features. To the best of our knowledge, this is the first attempt to automatically generate and select features using regularized regression models.

III. Problem Definition

Let $x \in X$ be a member of the input space, where $x = \langle f_1, f_2, ..., f_d \rangle$ is a fixed length vector of original feature values. Let $y \in Y$ be a member of the output space. Let training data with m records be denoted as $T = \{(x_1, y_1), ..., (x_m, y_m)\}$. Likewise, T' represents a set of testing examples. The goal is to learn a predictor $h : X \rightarrow Y$ from T, with low error Err_h. For this purpose, we transform an original feature vector x into a new feature vector x' such that $x' = \phi(x)$ and $\phi(x) = \langle \phi_1(x), \phi_2(x), ..., \phi_N(x) \rangle$ where each $\phi_i \in F_1$, the transformed feature space, and $N = |F_1|$. Each transformed feature value $\phi_i(x)$ is obtained by applying regression between pairs of features in the original space F_0. We want to infer a learning model $h' : \phi(x) \rightarrow Y$ with the aim that its error $Err_{h'}$ is less than Err_h [16].

IV. AutoLearn

A. Rationale for AutoLearn's Design

In supervised learning, the objective is to accurately determine the class of each record in the input dataset. Each record usually contains an instantaneous snapshot of parameter values (features) that are recorded for the objects or people being studied. The overall behaviour of data is difficult to recognize by looking at isolated records. However, we can expect that each class of objects will have different behaviour, from which it is much easier to identify the class, than using just the original set of features. In some cases, the classes may be indistinguishable by their original features, and only apparent when more discriminative information not obvious in the original feature space is generated.

In prior literature, a set of domain dependent operators were identified to transform features, based on the understanding that highly informative features often result from manipulations of elementary ones. In this paper, instead of using operators, we use regression as a tool to discover underlying patterns by the way features pairs are related to each other. The intuitive and theoretical justification for this is that *the precise manner in which a feature, say f_1 influences a feature f_2, might vary for each class*. While we cannot claim that it will vary for every feature pair, it is very likely, that for at least some substantial number of feature pairs, this variation will be very *prominent* and would result in highly discriminative information. Our goal in this paper is not to achieve accurate regression, but simply to use regression.

- As a means to discriminate between how features influence each other across classes.

- To mine feature relationships, linear or non-linear and their precise coefficients and forecast.

B. The Mechanics of AutoLearn

In this paper, we use each feature to predict the values of other features by applying regression, and include those predicted values (regression forecast) to supplement the information in each record. Our proposed learning model, as shown in Figure 1, has following four major steps:

- **Preprocessing:** The task of evaluating all candidate features isn't feasible hence preprocessing based on Information Gain is carried out.
- **Mining correlated features:** Define and search pairwise correlated features in the original feature space using distance correlation [17].
- **Feature generation:** Transform the original (available) feature space to construct new feature space by learning a predictive relation between correlated features through regularized regression models.
- **Feature selection:** Constrain the newly constructed feature space by selecting a subset of features based on the *stability* and *information gain* of constructed features with the aim to improve prediction accuracy while preserving the semantics of the original features.

*1) **Preprocessing**:* Evaluating all the available candidate features in the original feature space is not feasible. Thus, we preprocess the original feature space where features were ranked based on Information Gain (IG). This step is depicted in Algorithm 1. This is particularly helpful for high dimensional real world datasets.

Algorithm 1: Preprocessing

Input : Original Features F_{orig}
Output: Features selected after preprocessing F_d

$i = 0, F_d = \phi$;
Calculate IG score $\forall F_{orig}$;
while $i < orig$ **do**
  if *IG_score*$(F_i) > \eta_1$ **then**
    $F_d \leftarrow F_d \cup F_i$;
  end
  i++;
end
return F_d

Setting the value of parameter η_1: For datasets with less number of original features (< 500), all the features with IG score greater than zero i.e $IG > 0$ were selected. An example of this is shown in Figure 2, which represents the feature-IG plot for Dermatology dataset. All 34 original features had an IG score greater than 0 and hence all were selected. Another example (Figure 3) shows the IG scores for Sonar dataset, where 45 features out of 60 were selected during preprocessing. In the case of gene expression datasets we selected top 2% of the features based on IG score, as it was often observed that in the domain of gene expressions only a small amount of features were relevant as discussed in [18].

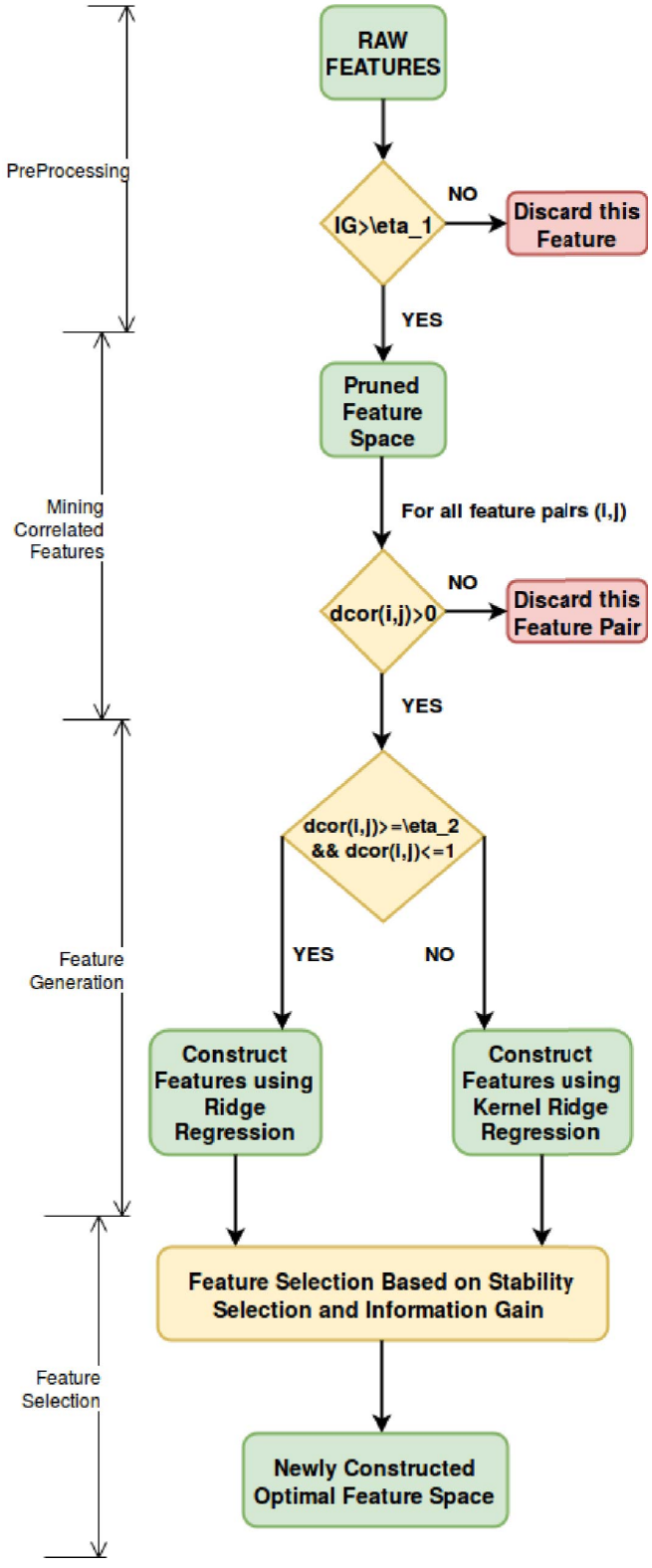

Fig. 1. Block diagram demonstrating the architecture of AutoLearn

*2) **Mining Correlated Features**:* The correlation or association between features is determined using distance cor-

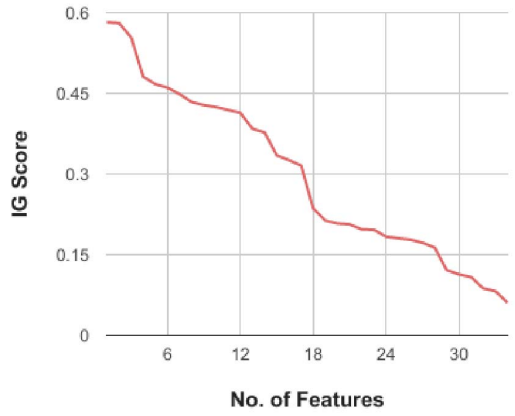

Fig. 2. Illustrated IG on Dermatology dataset for original features

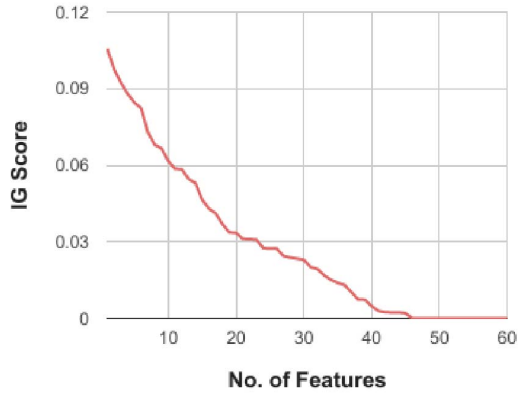

Fig. 3. Illustrated IG on Sonar dataset for original features

relation [17], a measure that decides whether there exists a predictive relationship of interest for a given feature pair. This measure is based on the Fourier transform, or characteristic function of sets of random variables and the related characterization of independence. This measure is useful in determining dependence between two random vectors of arbitrary dimensions. To be precise, let $\phi_u(t)$ and $\phi_v(s)$ be the respective characteristic functions of the random vectors u and v, and $\phi_{u,v}(t,s)$ be the joint characteristic function of u and v. They define the distance covariance between u and v with finite first moments to be the non negative number $dcov(u,v)$ given by:

$$dcov^2(u,v) = \int_{R^{d_u+d_v}} \|\phi_{u,v}(t,s) - \phi_u(t)\phi_v(s)\|^2 w(t,s)dtds \quad (1)$$

where d_u, d_v are the dimensions of u, v respectively, $\|\phi\|^2 = \phi\overline{\phi}$ for a complex-valued function ϕ, with $\overline{\phi}$ being the conjugate of ϕ and

$$w(t,s) = \{c_{d_u} c_{d_v} \|t\|_{d_u}^{1+d_u} \|s\|_{d_v}^{1+d_v}\}^{-1} \quad (2)$$

where the term c_d is given as:

$$c_d = \pi^{(1+d)/2}/\Gamma\{(1+d)/2\} \quad (3)$$

The distance correlation between u and v with finite first moments is defined as:

$$dcor(u,v) = \frac{dcov(u,v)}{\sqrt{dcov(u,u)dcov(v,v)}} \quad (4)$$

Following are some of the remarkable properties of the distance correlation that motivated us to use it for mining a wide range of interesting associations.

- Pearson's correlation [19] can only measure linear relationship whereas Spearmanś rho [20] and Kendallś tau [21] which both measure monotonic relationship, neither of them can capture all types of non-linear dependency between the two variables like distance correlation.
- Distance correlation is extremely general as it is applicable to random vectors of arbitrary and not necessarily equal dimension involving Euclidean pairwise distance only.
- $dcorr(u,v) = 0$ if and only if u and v are independent. Two univariate random variables U and V are independent if and only if U and $T(V)$, a strictly monotone transformation of V, are independent thus making distance correlation to be more effective in the presence of nonlinear relationship between U and V.
- $dcorr^2(u,v)$ satisfies the relation $0 \leq dcorr^2(u,v) \leq 1$.

We begin with a relatively small set of preprocessed features F_d and find correlated feature pairs useful for feature construction. This process is iteratively carried for all the pairs in the feature space. After this step, independent (non-correlated) feature pairs are filtered out (Line 4 Algorithm 2). The correlation between a given pair of feature vector can be linear (Line 9 Algorithm 2) or nonlinear (Line 5 Algorithm 2). We maintain a threshold parameter η_2 in order to segregate linear and nonlinear dependencies, respectively.

*3) **Feature Generation**:* Regression is a basic statistical process for estimating the relationship between independent and dependent variables. Here, given a training set of both of these variables, we use regularized regression algorithms to seek a connection between them for feature construction. They add additional constraints or penalty to a model, with the goal of preventing overfitting and improving generalization. Regularized regression models improve generalization capabilities especially when pair of features are highly correlated.

Specifically, we use ridge regression (RR) and kernel ridge regression (KRR) techniques to determine and model linear and non linear predictive relations respectively. Ridge regression is a highly popular approach that tends to determine a linear function that models the dependencies between covariates $\{x_i\}_{i=1}^n$ in $\mathbb{R}^p$ and response variable $\{y_i\}_{i=1}^n$ in $\mathbb{R}$.

Algorithm 2: Feature Generation

Input : Preprocessed Features F_d, η_1
Output: Newly constructed feature space F_N

1 $i = 0, j = 0, F_N = \phi$;
2 **while** $i < d$ **do**
3 **while** $j < d$ **do**
4 **if** *($i \neq j$) and ($dcor(F_i, F_j) \neq 0$)* **then**
5 **if** *($dcor(F_i, F_j) > 0$) and ($dcor(F_i, F_j) < \eta_2$)* **then**
6 $F_N \leftarrow F_N \cup KRR(F_{\mathrm{i}}, F_{\mathrm{j}})$;
7 $F_N \leftarrow F_N \cup (F_j - KRR(F_{\mathrm{i}}, F_{\mathrm{j}}))$;
8 **end**
9 **if** *($dcor(F_i, F_j) \geq \eta_2$) and ($dcor(F_i, F_j) \leq 1$)* **then**
10 $F_N \leftarrow F_N \cup RR(F_{\mathrm{i}}, F_{\mathrm{j}})$;
11 $F_N \leftarrow F_N \cup (F_j - RR(F_{\mathrm{i}}, F_{\mathrm{j}}))$;
12 **end**
13 **end**
14 j++;
15 **end**
16 i++ ;
17 **end**
18 return F_{N}

Ridge regression addresses some of the problems of Ordinary Least Squares (OLS) by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares,

$$\min_{w} \|Xw - y\|_2^2 + \alpha\|w\|_2^2 \tag{5}$$

Here, $\alpha \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity. This method has the same order of complexity than an Ordinary Least Squares.

Kernel ridge regression (KRR) combines Ridge Regression (linear least squares with l2-norm regularization) with the kernel trick. It thus learns a linear function in the space induced by the respective kernel and the data. For non-linear kernels, this corresponds to a non-linear function in the original space. The form of the model learned by KernelRidge is identical to support vector regression (SVR). However, different loss functions are used: KRR uses squared error loss while support vector regression uses ϵ-insensitive loss, both combined with l2 regularization. In contrast to SVR, fitting KernelRidge can be done in closed-form and is typically faster for medium-sized datasets.

Two types of features are generated for every correlated feature pair as follows:

- First type of feature is the forecast (prediction values) which we get by regressing F_i on F_j for every correlated feature pair (Line 6 and 10 in Algorithm 2) where F_i and F_j are independent and dependent variables respectively. Let this newly generated feature be denoted by $F_{forecast}$. Note that the feature constructed by regressing F_i on F_j is different from the feature built by regressing F_j on F_i. Intuitively, it discovers underlying patterns, both linear and non linear between feature pairs to generate informative features. So, in a perfect regression result, the new feature simply replicates another feature which is already present. We filter out such type of regression results on feature pairs in our feature generation step.
- Second feature is generated by taking the difference of the dependent variable F_j with the feature built by regressing F_i on F_j. It is expressed as $F_j - F_{forecast}$ (Line 7 and 11 in Algorithm 2). Intuitively, it depicts the variation between the actual and the predicted pattern for a given feature pair. So, here also in a perfect regression result, the difference will be zero between the actual and the predicted values. We filter out such type of regression results as well.

While the step of *mining correlated features* discovers the implicit patterns by the way features are related to each other, the *feature generation* step captures those prominent variations that result in highly discriminative information. The outcome here depicts that these two above categories of features try to learn the actual relationship between correlated feature pairs, thus improving the prediction accuracy. Other models [6], [7], [13] fail to capture this.

*4) **Feature Selection**:* All the newly constructed features F_N, generated by the feature generation step might not be of equal importance. For selecting the top performing features, we use a two step feature selection process, that employs stability based feature selection [22] followed by a final pruning based on Information Gain. The stability based feature selection plays three important roles in the context of classification: (1) it increases the classification accuracy by eliminating irrelevant features from the model; (2) prevents overfitting, and; (3) facilitates better interpretation by identifying sets of meaningful features that best discriminate the classes. Our two-step feature selection process is employed over only the constructed features as discussed in Algorithm 2 and is elaborated below:

Stability Based Selection: Stability based selection [23] is a relatively novel method for feature selection, based on subsampling in combination with selection algorithms (which could be regression, SVMs or other similar method). The high level idea is to apply a feature selection algorithm on different subsets of data and with different subsets of features. After repeating the process a number of times, the selection results can be aggregated, for example by checking how many times a feature ended up being selected as important when it was in an inspected feature subset. After a number of iterations, all features that were selected in a large fraction of the perturbations or subsamples are chosen. Finally, a cutoff threshold η_3 ($0 < \eta_3 < 1$) is applied in order to select the most stable features.

According to stability selection theory, any regression method which gives sparse representations can be used to select the features, as one is interested in the frequency of selected features and rather than its sparsity. In [22], the authors used lasso [24] formulation that includes an additional *l1* penalty bounding the absolute sum of all coefficients, thus

allowing weights of features to be set to zero. According to equation 6, $\hat{\beta}$ is the lasso estimate, p is the number of features and $\lambda \in \mathbb{R}^+$ is a regularization parameter that determines the model sparseness.

$$\hat{\beta} = \arg\min \beta \in \mathbb{R}^p \|X - Y\beta\|_2^2 + \lambda \sum_{k=1}^{p} |\beta_k| \qquad (6)$$

When there are highly correlated features, all of which are related to some extent to the response variable, lasso tends to select only one or a few of them and shrinks the rest to 0. This may not be a desirable property.

In order to overcome this problem we used Randomized lasso [25]. The Randomized lasso changes the penalty lambda to a randomly chosen value in a predefined range, according to the equation given below.

$$\beta^{\hat{\lambda},W} = \arg\min \beta \in \mathbb{R}^p \|X - Y\beta\|_2^2 + \lambda \sum_{k=1}^{p} \frac{|\beta_k|}{W_k} \qquad (7)$$

The re-weighting is not based on any previous estimate, but is simply chosen randomly. According to [25], applying this random re-weighting and looking for variables that are chosen often, turns out to be a very powerful procedure. The scores or the weights drop smoothly, but in general, the drop off is not sharp as is often the case with pure lasso. This means stability selection is useful for both pure feature selection to reduce overfitting, but also for data interpretation: in general, good features wont get 0 as coefficients, just because there are similar, correlated features in the dataset (as is the case with lasso). For feature selection, we have found it to be among the top performing methods for many different datasets and settings. In short, the features selected more often are good features and lead to generalization.

IG Based Selection: The most stable features are then selected based on the highest information gain scores. Information Gain (IG) looks at each feature in isolation and measures how important it is for the prediction of the correct class label. In cases like ours, where all features are not redundant with each other, IG proves to be useful.

V. Illustrative Example

For this example we use the Sonar dataset from the UC Irvine repository which has 60 features, 208 instances and class distribution of 50% for each of 2 class labels. During the first step of *preprocessing*, 45 features out of 60 original features were selected. While *mining correlated pairs* among these, 2037 dependent correlations emerged. Out of 2037 correlations, 62 were linearly correlated and 1975 were non-linearly related. The total number of newly constructed features were 4073 from these correlated feature pairs. However, all the features constructed using these correlated feature pairs might not be useful for prediction. Two-step feature selection helps in optimal subset selection of features constructed using Algorithm 2 thus reducing the number to 27 new features only. This is a small number in contrast to other models like FCT, TFC and EK. The relative importance of 60 original features and 27 newly constructed and selected features is illustrated in Figure 4. Here, the y axis shows the relative importance which is measured using a facility available in the forests of trees implementation in [26]. Figure 4 shows that some newly constructed features (indicated in blue) are relatively more important than the original features, allowing us to determine which features matter most to prediction. Note that this example and all the values in it are averaged over 5-folds.

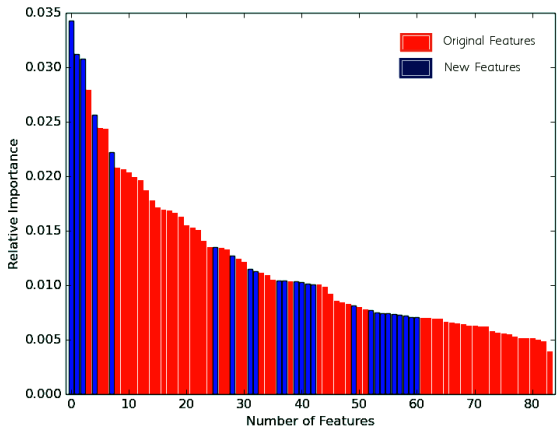

Fig. 4. Relative feature importance of Sonar dataset

VI. Evaluation

A. Datasets

We evaluated AutoLearn on 25 classification datasets with large variety in size, number of attributes and class imbalance. Different domains (life, images, computer, physical, animal, health and education, biology) were used to evaluate the proposed learning model. Out of 25 datasets, 7 of these are Gene Expression datasets (collected from different sources) and remaining are taken from the UC Irvine repository [27]. Their characteristics are discussed in Table 1. The Gene Expression datasets that belong to the bioinformatics and biomedical domain can be divided into two parts: microarray datasets (MA) ans mass spectrometry (MS) datasets. For each domain, datasets were included, typically consisting of several thousands of features and tens of instances in the case of microarray datasets (e.g. Colon, Leukaemia and Lymphoma), and up to 15,154 features and a few hundreds of instances in the case of mass spectrometry datasets (e.g. Ovarian Cancer, Prostate Cancer, Lung Cancer and Arcene). Due to their high dimensionality and low sample size, these datasets pose a great challenge for both classification and feature learning algorithms and were thus deliberately chosen to evaluate our models' efficacy.

B. Experimental Setup

We evaluate our model on 8 state-of-the-art classification algorithms (CLF) namely KNN, Logistic Regression (LR), SVM with linear kernel (SVM-L), SVM with polynomial kernel (SVM-P), Random Forest (RF), Adaboost (AB), Multi Layered Perceptron (NN) and Decison Tree (DT). For each classifier, we compared our model with 3 other top performing

TABLE I. CHARACTERISTICS OF 25 DATASETS

Dataset Name	Features	Instances	Labels
Abalone	7	4177	3
Arcene	10000	200	2
Bank Note	4	1372	2
Colon	2000	62	2
Dermatology	34	366	4
E.coli	7	336	8
FeatureFourier	76	2000	10
FeaturePixel	240	2000	10
Fertility	9	100	2
Haberman	3	306	2
Ionosphere	34	351	2
Letter Recognition	16	20000	26
Leukaemia	7129	72	2
Libras	90	360	15
Lung Cancer	12600	203	2
Lymphoma	4026	96	9
Optical Digits	64	5620	10
Ovarian Cancer	15154	253	2
Pima Diabetes	8	768	2
Poker	10	1025010	10
Prostate Cancer	5966	102	2
Shuttle	10	58000	7
Sonar	60	208	2
Waveform	21	5000	3
Wine	13	178	3

state-of-the-art models namely FCTree (FCT) [13], TFC [7] and ExploreKit (EK) [14]. The performances reported are measured in terms of accuracy on the actual feature space (ORIG) and then on the supplemented feature space (i.e original features combined with the features learned) which is also the interpretation in prior literature [7], [14]. Our model results for the supplemented feature space are mentioned under **AL*** column in the Tables II-V. The accuracy is reported after 5-fold cross-validation. Note that for both feature generation and feature selection only the training set in each fold of cross-validation was used.

C. Parameter Settings

The value of the first parameter η_1 (refer Figure 1) was selected, based on the rationale provided in *Section IV subsection B-1*. As explained, for datasets other than Gene Expressions, the value of η_1 was set to 0. For the latter case, the feature-IG plot was drawn for each dataset and the value was decided based on the IG scores of top 2% of the features as discussed in [18]. The parameter value for η_2 was set at 0.7 since this value best segregates linear and non linear correlations as discussed in [17]. The accuracy of the classifiers was computed using the scikit-learn's [26] default parameters. This is justified because we are more interested in the improvement in classifier accuracy due to different feature representations, than by tuning the classifiers themselves. The default scikit-learn parameters were also used for ridge and kernel ridge regression. For kernel ridge, we have used the *rbf* kernel as it is found to provide good generalization capabilities. The threshold for stability selection (η_3) was determined using grid search on values $\{0.01, 0.05, 0.1, ..., 0.7\}$. The value for information gain threshold was also determined using grid search on values of $\{0.1, 0.2, ..., 0.5\}$.

TABLE II. CLASSIFICATION ACCURACIES-I

Dataset	CLF	ORIG	TFC	FCT	EK	AL*
Abalone	KNN	**23.27**	21.64	22.60	23.09	22.71
	LR	24.61	23.69	23.60	24.79	**26.50**
	SVM-L	25.71	25.64	25.72	25.68	**26.07**
	SVM-P	19.46	17.64	22.12	21.38	**23.77**
	RF	22.91	18.78	23.02	**23.18**	22.21
	AB	20.61	19.10	19.97	**21.12**	20.61
	NN	27.53	26.32	26.41	27.09	**27.81**
	DT	19.27	19.00	19.13	19.29	**19.41**
Arcene	KNN	80.5	80.5	80.5	81.14	**82.50**
	LR	86.00	84.00	84.00	**86.48**	85.50
	SVM-L	88.50	86.50	87.50	**89.50**	86.50
	SVM-P	88.00	87.00	86.00	**89.00**	85.50
	RF	76.50	76.23	76.92	77.07	**78.50**
	AB	72.50	74.00	75.00	75.50	**77.50**
	NN	65.50	68.97	69.95	73.68	**88.00**
	DT	69.00	69.00	69.00	71.46	**74.50**
Bank	KNN	99.92	99.27	99.52	99.52	**99.70**
	LR	98.90	97.95	98.68	98.97	**99.70**
	SVM-L	98.76	98.97	99.27	99.27	**99.92**
	SVM-P	98.90	98.27	99.16	99.27	**99.63**
	RF	99.05	98.27	98.75	99.19	**99.56**
	AB	99.63	99.27	**99.78**	**99.78**	99.48
	NN	**100.00**	99.02	99.02	99.92	99.92
	DT	98.25	98.12	98.56	98.19	**99.19**
Colon	KNN	78.97	79.34	79.56	80.12	**80.38**
	LR	75.38	74.68	75.12	77.99	**80.51**
	SVM-L	75.38	74.07	75.02	**76.09**	74.10
	SVM-P	**73.97**	70.16	71.29	73.17	70.69
	RF	70.64	71.37	71.73	72.09	**72.30**
	AB	72.56	71.23	73.06	73.97	**75.76**
	NN	62.82	65.67	69.12	72.46	**78.84**
	DT	75.89	75.16	**76.27**	76.14	73.97
Dermat.	KNN	89.11	90.46	92.89	91.00	**96.09**
	LR	97.21	97.76	97.97	97.64	**98.61**
	SVM-L	**97.21**	96.02	96.27	96.27	96.92
	SVM-P	**94.41**	94.00	94.12	92.01	93.56
	RF	**96.92**	96.45	96.61	95.48	95.81
	AB	54.13	57.12	**61.00**	57.33	54.96
	NN	98.04	97.13	97.22	97.67	**98.22**
	DT	**95.24**	95.06	94.96	95.36	94.68
E.coli	KNN	86.59	**88.42**	87.56	**88.42**	84.82
	LR	75.88	78.23	79.24	82.83	**87.19**
	SVM-L	85.71	85.71	85.71	**86.30**	**86.30**
	SVM-P	56.54	59.32	62.14	72.31	**80.33**
	RF	82.73	83.46	83.76	85.12	**86.59**
	AB	62.47	63.54	64.37	**65.75**	**65.75**
	NN	78.28	80.37	81.97	83.67	**86.90**
	DT	79.74	76.32	77.67	**80.34**	76.48

D. Comparative Evaluation

Tables II-V show how Figure 1's *newly constructed optimal feature space*, when combined with original features perform against TFC, FCTree (FCT) and ExploreKit (EK) frameworks. This accuracy is denoted as AL* in these tables. We achieved **13.28%** and **5.87%** improvement in terms of accuracy over the original feature space and the supplemented feature space learned by other top performing models respectively, across all 25 datasets and 8 different classifiers. In general, we noticed that AL* tends to outperform other methods and learned features *significantly* improved the comprehensibility of the produced classifiers by capturing prominent implicit patterns of the underlying target concept. It can be observed that our

TABLE III. CLASSIFICATION ACCURACIES-II

Dataset	CLF	ORIG	TFC	FCT	EK	AL*
Fourier	KNN	83.85	82.17	83.82	**83.98**	83.55
	LR	79.45	79.97	80.00	82.24	**83.15**
	SVM-L	81.45	81.15	82.86	82.45	**83.05**
	SVM-P	8.70	42.25	57.97	66.65	**82.30**
	RF	79.9	78.90	79.16	**80.78**	79.31
	AB	48.65	46.66	49.29	49.95	**50.40**
	NN	81.90	82.34	83.12	83.38	**85.50**
	DT	74.00	74.00	74.00	**74.09**	74.35
Pixel	KNN	97.75	98.12	97.23	97.96	**97.95**
	LR	94.35	94.22	94.28	95.54	**95.75**
	SVM-L	92.9	92.57	93.26	**94.27**	**94.27**
	SVM-P	98.35	98.22	**98.66**	**98.66**	97.25
	RF	95.5	94.26	95.12	**96.54**	94.20
	AB	54.05	54.00	54.86	55.25	**55.60**
	NN	97.15	97.15	97.15	97.15	97.15
	DT	87.30	86.12	86.78	86.62	**87.65**
Fertility	KNN	85.00	86.00	86.00	**87.00**	**87.00**
	LR	88.00	88.00	**89.00**	88.00	87.00
	SVM-L	85.00	87.00	**88.00**	87.00	87.00
	SVM-P	**88.00**	87.00	87.00	**88.00**	**88.00**
	RF	82.00	87.00	87.00	**90.00**	84.00
	AB	79.00	83.00	**84.00**	83.00	79.00
	NN	**88.00**	**88.00**	**88.00**	**88.00**	85.00
	DT	80.00	84.00	84.00	**85.00**	**85.00**
Haberm	KNN	71.89	70.00	71.28	**72.27**	68.68
	LR	74.19	72.07	73.96	74.52	**76.16**
	SVM-L	74.18	73.97	74.18	75.37	**75.82**
	SVM-P	74.18	73.98	74.81	75.12	**75.52**
	RF	68.63	68.91	69.07	**69.98**	65.34
	AB	70.25	71.19	71.57	**72.17**	69.93
	NN	**73.19**	69.02	71.19	72.21	70.91
	DT	66.65	66.09	66.79	**67.23**	66.34
Ionosp	KNN	84.31	84.66	84.87	**85.97**	83.46
	LR	87.44	87.26	87.39	87.67	**87.95**
	SVM-L	87.44	86.71	87.78	**88.01**	84.30
	SVM-P	64.10	70.16	71.45	72.62	**74.63**
	RF	93.15	91.65	93.16	**93.97**	92.30
	AB	92.02	90.94	90.12	90.31	**92.43**
	NN	93.14	92.45	92.13	**93.64**	92.29
	DT	88.32	87.12	88.04	88.12	**88.59**
Letter	KNN	95.02	95.00	95.96	**96.14**	95.96
	LR	71.66	77.42	80.07	72.25	**86.57**
	SVM-L	54.96	57.98	58.81	60.00	**62.34**
	SVM-P	95.01	95.12	95.26	95.70	**96.52**
	RF	93.96	93.74	93.79	**94.66**	94.14
	AB	27.82	28.00	29.07	31.00	**31.38**
	NN	91.67	92.45	93.45	95.16	**96.02**
	DT	87.78	88.03	88.34	89.00	**90.12**

TABLE IV. CLASSIFICATION ACCURACIES-III

Dataset	CLF	ORIG	TFC	FCT	EK	AL*
Leukae.	KNN	82.09	85.31	87.12	90.07	**97.23**
	LR	84.76	87.64	91.21	92.17	**95.80**
	SVM-L	84.26	86.83	88.92	91.92	**95.64**
	SVM-P	65.23	69.31	74.15	79.12	**81.90**
	RF	**90.28**	89.48	89.86	90.17	90.19
	AB	92.95	92.11	92.49	93.02	**93.15**
	NN	86.09	90.11	91.37	92.46	**94.38**
	DT	83.33	83.95	84.13	**86.00**	**86.00**
Libras	KNN	70.00	71.00	71.18	**73.70**	69.44
	LR	60.27	64.68	67.12	**71.70**	70.00
	SVM-L	68.61	69.88	**70.83**	70.44	67.22
	SVM-P	2.22	36.68	47.97	47.75	**49.44**
	RF	71.94	72.12	73.07	**77.60**	70.22
	AB	8.05	10.12	13.11	16.88	**18.05**
	NN	71.66	72.35	74.24	75.69	**78.33**
	DT	62.5	62.64	63.12	63.74	**65.55**
Lung	KNN	88.88	88.96	89.88	90.14	**92.61**
	LR	91.93	93.14	91.96	92.19	**94.20**
	SVM-L	91.95	92.64	92.66	93.0	**93.79**
	SVM-P	90.09	88.46	90.32	**91.42**	85.81
	RF	87.03	86.47	88.62	89.18	**90.03**
	AB	82.72	83.01	83.17	**83.33**	**83.33**
	NN	74.69	76.88	79.43	86.97	**94.44**
	DT	88.30	89.17	88.75	**90.37**	85.22
Lymp.	KNN	87.42	87.67	88.12	**88.27**	84.26
	LR	**87.52**	86.07	86.31	86.93	86.42
	SVM-L	85.47	85.37	85.31	86.39	**88.52**
	SVM-P	58.26	60.37	62.49	64.97	**68.73**
	RF	76.05	77.34	76.46	77.97	**79.05**
	AB	52.15	51.46	52.71	**53.09**	50.84
	NN	83.36	84.65	85.12	**85.36**	**85.36**
	DT	63.42	64.34	64.89	65.77	**68.73**
Optical	KNN	**98.77**	97.20	98.02	98.02	97.03
	LR	96.49	96.40	96.40	**96.96**	95.83
	SVM-L	94.89	94.12	**95.17**	95.09	94.01
	SVM-P	**99.09**	99.03	99.03	99.07	96.20
	RF	96.38	96.36	96.91	**97.27**	96.57
	AB	68.65	67.62	68.35	69.68	**73.78**
	NN	**98.02**	95.62	95.37	96.46	96.93
	DT	89.90	88.00	88.46	**90.41**	**90.41**
Ovarian	KNN	93.29	95.64	95.97	96.21	**98.02**
	LR	**100.00**	99.00	99.00	**100.00**	**100.00**
	SVM-L	**100.00**	99.24	99.41	99.72	**100.00**
	SVM-P	64.01	67.23	69.21	79.97	**95.66**
	RF	97.23	96.27	95.46	97.17	**97.62**
	AB	98.80	97.45	98.12	98.97	**99.27**
	NN	54.50	59.62	63.12	77.34	**98.41**
	DT	94.47	95.37	96.12	96.37	**97.63**

model is effective even when the number of features are as small as 4 or as big as 15154.

Due to space constraints, we have summarized the accuracy results at every step of our learning model in Table VI. The results demonstrated in Table VI validate the effectiveness of all 4 major steps of our learning model as discussed in Section IV. All the improvements mentioned in Table VI are in comparison to the original feature space. We are confident that this improvement is mainly due to the essential feature generation step of our framework and not because of the feature selection step alone. To validate this point we have also calculated the accuracy in the scenario where only the two step feature selection was applied on the pruned feature space (refer Figure 1 and Table VI, row 4) and not much improvement in the results was observed compared to AL*. We also calculated accuracy for the following two cases (refer Figure 1): (1) *original features* combined with our features generated just after the *feature generation step* (Table VI, row 2) and (2) *Original features* combined with the *newly constructed optimal feature space* (Table VI, row 3). The % improvement for these are noteworthy.

VII. SCALABILITY ANALYSIS

Our scalability analysis experiments show the final number of new features after construction and selection. This ensures that classification algorithms that use these features as input are

TABLE V. CLASSIFICATION ACCURACIES-IV

Dataset	CLF	ORIG	TFC	FCT	EK	AL*
Pima	KNN	71.48	72.42	73.52	**73.58**	68.36
	LR	**76.55**	75.92	76.52	73.86	74.99
	SVM-L	65.23	62.71	72.52	73.71	**74.85**
	SVM-P	64.89	65.71	70.52	72.57	**76.32**
	RF	**75.37**	72.42	73.52	74.01	73.05
	AB	**74.34**	74.08	74.08	74.28	72.52
	NN	64.32	64.12	64.22	67.31	**72.39**
	DT	72.38	70.23	70.46	70.94	**71.05**
Poker	KNN	61.76	58.27	62.78	63.02	**66.14**
	LR	50.11	54.62	57.47	55.01	**59.46**
	SVM-L	49.74	46.54	47.44	47.69	**50.13**
	SVM-P	58.32	59.23	**61.17**	55.68	60.02
	RF	68.1	70.32	71.51	70.68	**72.26**
	AB	35.76	36.23	37.00	**39.95**	39.46
	NN	**99.72**	99.57	99.57	**99.72**	99.57
	DT	63.81	64.33	64.16	65.71	**66.17**
Prostate	KNN	87.23	88.25	89.32	90.19	**91.19**
	LR	90.19	90.89	91.46	92.00	**92.14**
	SVM-L	91.14	90.46	91.12	**91.97**	90.19
	SVM-P	91.19	88.12	89.97	**92.02**	88.47
	RF	86.28	87.16	88.30	90.19	**91.19**
	AB	88.19	88.30	89.12	**90.19**	**90.19**
	NN	50.90	54.67	59.12	82.46	**90.19**
	DT	77.61	77.21	78.37	78.19	**79.42**
Shuttle	KNN	99.78	99.02	99.57	**99.92**	**99.92**
	LR	92.90	93.31	93.76	94.52	**96.28**
	SVM-L	90.41	91.19	92.18	93.23	**96.57**
	SVM-P	**99.92**	99.81	99.81	99.88	99.88
	RF	**99.97**	99.72	99.72	99.92	99.81
	AB	87.50	89.17	90.42	91.56	**92.46**
	NN	99.71	99.52	99.71	99.92	**99.98**
	DT	**99.98**	99.18	99.52	99.67	99.67
Sonar	KNN	78.35	81.48	82.70	82.40	**83.19**
	LR	77.42	78.12	78.72	78.73	**79.00**
	SVM-L	73.54	74.54	75.75	76.11	**77.30**
	SVM-P	53.36	58.41	66.44	33.64	**81.71**
	RF	73.55	81.00	**82.54**	47.40	77.87
	AB	80.74	80.00	**81.04**	53.95	78.83
	NN	80.30	81.07	82.00	82.37	**84.09**
	DT	75.01	74.23	74.52	74.96	**75.02**
Waveform	KNN	**82.48**	81.28	82.00	82.09	81.14
	LR	86.58	86.72	**87.18**	86.92	85.12
	SVM-L	86.9	84.54	86.23	**86.92**	84.4
	SVM-P	81.7	81.62	82.54	80.40	**85.42**
	RF	82.1	81.45	82.04	**82.12**	81.12
	AB	83.62	82.54	82.84	83.04	**83.78**
	NN	**85.84**	82.31	3.10	84.67	83.72
	DT	**75.04**	72.46	73.00	73.16	73.06
Wine	KNN	67.93	74.89	79.93	83.40	**93.84**
	LR	95.52	96.89	97.24	95.12	**98.30**
	SVM-L	83.03	88.14	89.94	90.82	**98.31**
	SVM-P	96.65	**96.68**	96.65	92.12	92.68
	RF	96.07	96.68	97.12	90.00	**97.20**
	AB	85.82	**88.12**	91.23	62.80	84.71
	NN	42.73	46.23	49.56	64.63	**97.19**
	DT	91.57	91.79	92.01	92.45	**93.22**

not burdened with the curse of dimensionality, and are hence scalable. This is also the interpretation in prior literature [13]. As discussed earlier in related work, exhaustive searching using all possible set of operators is ineffective, due to the infinite feature space it may create. Even if the number of operators are limited, exhaustive searching can result in combinatorial explosion. The approach submitted by us selects a very small number of new features to reach the desired performance, thus making our model scalable. This is in contrast to the thousands of features considered by models suggested in earlier works. Figure 5 illustrates the number of features finally selected by our learning model against the number selected by the FCTree, TFC and EK on 25 datasets. Clearly, we can conclude that the number of features required by our model are significantly less. Our model on an average uses at least *ten* times lesser number of features than the best performing submitted techniques across all datasets.

TABLE VI. ACCURACY EVALUATION AFTER EVERY STEP ON AUTOLEAN IN COMPARION TO ORIGINAL FEATURES

Model	% Improvement
After Preprocessing	1.25
After Feature Generation Step	10.13
After 2-step Feature Selection (our complete model results as reported in table II-V)	13.28
With only 2-step Feature Selection on preprocessed features	4.12

VIII. CONCLUSIONS AND FUTURE WORK

In order to expand the scope and ease of applicability of machine learning, it would be highly desirable to make learning algorithms less dependent on feature engineering, so that novel applications could be constructed faster. In this work, we have presented how to efficiently and automatically generate and select highly predictive features that can be generalized over a large number of classifiers on datasets of different domains. We generate two types of features: (1) The first type discovers underlying patterns between feature pairs. (2) The second type is a measure of how much the forecast deviates with respect to the independent variable. These lead to better inference from raw data. Our analysis shows that: (1) the distance correlation can effectively mine important pairwise associations; (2) correlated features assist the regression model to construct highly predictive features without domain related heuristics; (3) a set of features can be selected, without running into exhaustive search via a two-step process of stability selection and information gain, that prevents overfitting and improves model generalization as well. These factors combined with our experimental results are very encouraging and subscribe to the reasons why we think our proposed approach works better. In our current work, we plan to combine our approach with deep learning models.

REFERENCES

[1] L. Breiman *et al.*, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical science*, vol. 16, no. 3, pp. 199–231, 2001.

[2] D. Donoho, "50 years of data science," in *Princeton NJ, Tukey Centennial Workshop*, 2015.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

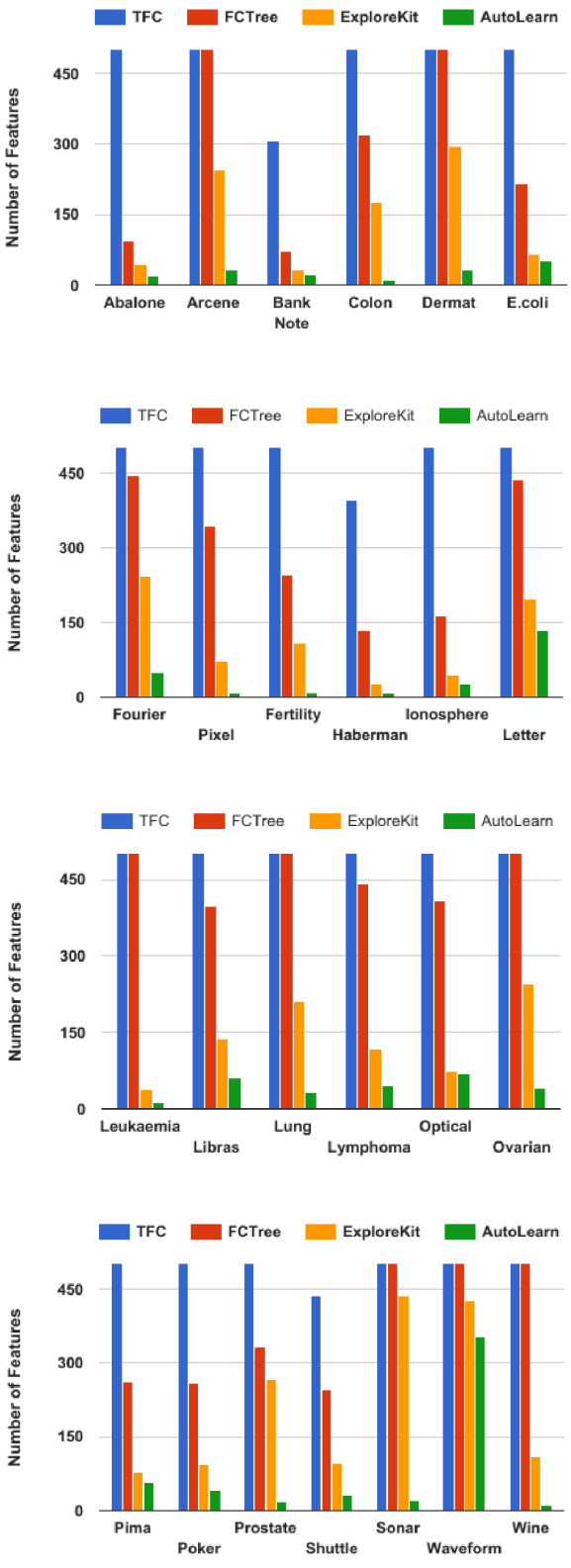

Fig. 5. Scalability Analysis of TFC, FCT, EK and AL* on 25 datasets

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] W. Cheng, G. Kasneci, T. Graepel, D. H. Stern, and R. Herbrich, "Automated feature generation from structured knowledge," in *CIKM*, 2011.

[6] S. H. Lim, L.-L. Wang, and G. DeJong, "Explanation-based feature construction," *IJCAI*, 2007.

[7] S. Piramuthu and R. T. Sikora, "Iterative feature construction for improving inductive learning algorithms," in *Expert Syst. Appl.*, 2009, p. 34013406.

[8] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," in *Genetic Programming and Evolvable Machines*, July 2005.

[9] G. Pagallo, "Learning dnf by decision trees," in *IJCAI*, 1989.

[10] J. C. Schlimmer and R. Granger, "Incremental learning from noisy data," *Machine Learning*, vol. 1, pp. 317–354, 1986.

[11] C. J. Matheus and L. A. Rendell, "Constructive induction on decision trees," in *IJCAI*, 1989.

[12] S. Markovitch and D. Rosenstein, "Feature generation using general constructor functions," *Machine Learning*, vol. 49, pp. 59–98, 2002.

[13] W. Fan, E. Zhong, J. Peng, O. Verscheure, K. Zhang, J. Ren, R. Yan, and Q. Yang, "Generalized and heuristic-free feature construction for improved accuracy," in *SDM*, 2010.

[14] G. Katz, E. C. R. Shin, and D. Song, "Explorekit: Automatic feature generation and selection," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 979–984.

[15] P. M. Murphy and M. J. Pazzani, "Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees," in *Proceedings of the eighth international workshop on machine learning*, 1991, pp. 183–187.

[16] P. Sondhi, "Feature construction methods: A survey," 2009.

[17] G. Szekely, M. Rizzo, and N. Bakirov, "Measuring and testing dependence by correlation of distances," in *Annals of Statistics*, 2007.

[18] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[19] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240–242, 1895.

[20] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[21] S. Yue, P. Pilon, and G. Cavadias, "Power of the mann–kendall and spearman's rho tests for detecting monotonic trends in hydrological series," *Journal of hydrology*, vol. 259, no. 1, pp. 254–271, 2002.

[22] N. Meinshausen and P. Bühlmann, "Stability selection," 2008.

[23] T. Lange, M. L. Braun, V. Roth, and J. M. Buhmann, "Stability-based model selection," in *Advances in Neural Information Processing Systems 15*, S. Thrun and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2002, pp. 617–624. [Online]. Available: http://books.nips.cc/papers/files/nips15/AA17.pdf

[24] R. Tibshirani, "Regression shrinkage and selection via the lasso," 1994.

[25] S. Wang, B. Nan, S. Rosset, and J. Zhu, "Random lasso," *Annals*, vol. 5, no. 1, pp. 468–485, 2011.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml