Data-Driven Representation Learning with Applications in Gene Expression and Fake News Detection

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS in Computer Science and Engineering
by Research

by

Ambika Kaul
201450855
ambika.kaul@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500032, INDIA
November, 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Data-Driven Representation Learning with Applications in Gene Expression and Fake News Detection" by Ambika Kaul, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Dr. Vikram Pudi

To my dearest family, for their
endless love and support.

Acknowledgements

I would like to express my gratitude to my advisor, Dr.Vikram Pudi, who gave me the freedom to explore and to succeed. I would also like to thank him for his gentle help, encouraging me and instilling in me the confidence when I needed it the most.

I am greatful to IIIT Hyderabad, for giving me an inspiring platform, where I could nurture myself. I'm also thankful to Dr. Choppella, for giving me the opportunity to work with him and an exposure, that helped me a great deal in this journey.

I extend my thanks to my labmates - Pratibha, Soumyajit, Rahul, Mihir and Raghavendra, for taking out their time and giving me encouragement to progress.

The support of my close friends - Saket, Garima, Sanchita and Ashish, is greatly acknowledged. Lastly, I would like to express my deepest gratitude to my family, for their endless love and patience.

Abstract

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. In order to make it possible to apply machine learning to different domains, it is very important to make learning algorithms less dependent on manual feature engineering, so that novel AI based applications could be developed faster.

The work in this thesis focuses not only on the mapping from representation to output but also the representation itself by mining hidden patterns from available data. Keeping this in mind, we propose a learning model to alleviate difficulties involved in feature engineering through automation. 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. Because this model takes into account the inherent feature structures, we took our next motivating example from bioinformatics. The features in this domain have some natural spatial order (for instance, in the medical domain, genes are often associated with different types of clinical features) and thus incorporating such structure can help select more important features and achieve more accurate classification accuracy. It also provides better readability and interpretability to the models.

The final component of the work focuses on the problem of fake news detection. The low cost, easy access and rapid circulation of information over the internet has encouraged more people than ever, to seek out and consume news from online sources or social media rather than traditional news organizations. While the traditional (count-based) feature engineering strategies for textual data are effective methods for extracting features from text, due to the inherent nature of the model being just a bag of unstructured words, we lose additional information like the semantics, structure, sequence and context around nearby words in each text document. This formed as a motivation for us to explore more sophisticated models which can capture this information and give us features which are vector representation of words. Keeping in mind that fake news is intentionally written to mislead readers to believe false information, it can be difficult to detect it, solely based on news content. Therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a

determination. We therefore design, a novel attention based hybrid network which integrates the article information along with the meta data and claims of the news articles.

Contents

List of Figures

List of Tables

Chapter 1

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 *representation learning* [2].

The first component in this thesis focuses not only on the mapping from representation to output but also the representation itself by mining hidden patterns from available data. Learned representations often result in much better performance than what can be obtained with hand-designed representations [1]. A representation learning algorithm can thus discover a good set of features for a simple task in minutes, or a complex task in hours to months. In contrast, feature engineering requires substantial manual effort in designing and selecting features and is often tedious and non scalable. Keeping this in mind, we propose a learning model to alleviate difficulties involved in feature engineering through automation.

Because this model takes into account the inherent feature structures, we took our next motivating example from bioinformatics. The features (like gene expressions) in this domain have some natural spatial order and thus incorporating such spatial structure can help select more important features and achieve more accurate classification accuracy. On one hand, for many applications, where the raw input data do not contain any features understandable to a given learning algorithm, feature engineering is preferred. On the other hand, as feature engineering creates a set of new features, further analysis is problematic as we cannot retain the physical meanings of these features. In contrast, by keeping some of the original features, feature selection maintains physical meanings of the original features and gives

models better readability and interpretability. As a result, feature selection is often preferred in many applications such as text mining and genetic analysis.

The final and third component here, focuses on the problem of fake news detection. The low cost, easy access and rapid circulation of information over the internet has encouraged more people than ever, to seek out and consume news from online sources or social media rather than traditional news organizations. It has been proven recently that false rumors and fake news reach more people than the accurate stories [3]. It also penetrates deeper into the social network in a sense that it is more likely to go viral than a real story. Up until now, efforts to automate response detection typically modeled the spread of fake news as an epidemic on a social graph, or used hand-crafted features that were social-network dependent, such as the number of Facebook likes, combined with a traditional classifier [4]. As a result, one of the future research directions in fake news detection is *feature-oriented* [5].

Traditional (count-based) feature engineering strategies for textual data involve models belonging to a family of models popularly known as the Bag of Words model. This includes term frequencies, TF-IDF (term frequency-inverse document frequency), N-grams and so on. While they are effective methods for extracting features from text, due to the inherent nature of the model being just a bag of unstructured words, we lose additional information like the semantics, structure, sequence and context around nearby words in each text document. This forms enough motivation for us to explore more sophisticated models which can capture this information and give us features which are vector representation of words, popularly known as embeddings.

With regard to speech or image recognition systems, all the information is already present in the form of rich dense feature vectors embedded in high-dimensional datasets like audio spectrograms and image pixel intensities. However, when it comes to raw text data, especially count based models like Bag of Words, we are dealing with individual words which may have their own identifiers and do not capture the semantic relationship amongst words. This leads to huge sparse word vectors for textual data and thus if we do not have enough data, we may end up getting poor models or even overfitting the data due to the curse of dimensionality.

The fake news articles are produced online for a variety of purposes, such as financial and political gain. Keeping in mind that fake news is intentionally written to mislead readers to believe false information, it can be difficult to detect it, solely based on news content. Therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination [5]. Exploiting such information can be challenging but useful. This section of the work builds in the direction of automating fake news detection. The basis of our work is LIAR dataset [6], which is the largest public fake news dataset. We use this dataset and in our model, incorporate complete news articles as well, that will benefit us in capturing the underlying context of the originally available short statements. We design a novel attention based hybrid network which integrates the article information along with the meta data and claims of the news articles.

1.1 Related Work

A number of feature learning models have been proposed, such as [7, 8, 9], that use two step batch feature construction to transform the original features. For instance, the proposed model in [7] uses prior domain information and explanation-based interaction of training examples to construct distinguishing features, that are task-relevant. The work in [8] 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 [10, 11, 12] have been proposed. The approach in [10] 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 [13] improves over [10], but requires domain proficiency to select feature constructors. The work in [14] 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) [15] 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 [16].

A number of feature selection techniques have emerged in the field of bioinformatics as well due to the high dimensional nature of modelling tasks involved. Contrary to dimensionality reduction, these techniques do not alter the original representation of the variables, but merely select a subset of them. This preserves the original semantics of the variables, hence, offering the advantage of interpretability by a domain expert [17].

As [18] points out, several solutions have also been proposed for the problem of cancer detection using gene expression data, most of them performing feature space reduction by selecting features manually or in supervised ways. In [19], recursive feature elimination and univariate association filtering approaches are used to select a small subset of the gene expressions. Similarly, [20] applied recursive feature elimination using SVM to find a small number of gene expressions to be used as the feature space for the classification. Then [17] reports efforts to use single-layer, nonlinear dimensionality reduction techniques to classify samples based on gene expression data. The meaningful part of the data after extraction, enables identification of specific subsets of genes that are useful for biologists and physicians.

In addition, since specific cancer data are usually rare and most of the mentioned methods can not efficiently take advantage of data from other cancers than the one to be detected or classified, these meth-

ods have to operate with very small data sets, limiting the effectiveness of the automatic feature learning approaches used. Additional problem with these approaches is that these methods are not scalable and can not be generalized to new cancer types. The majority of these methods use manually designed feature selectors to reduce the dimensionality of gene expression. Researchers have also applied PCA to a set of combined genes of datasets to obtain the linear representation of the gene expression and then apply an autoencoder to capture nonlinear relationships [18]. We compare our results with this latest work, along with another baseline model of [8]. The usefulness of our features is demonstrated in experiments and shows the potential for effective feature learning in the presence of very limited data sets.

Most of the existing work in fake news detection has focused on either the text, the response an article receives, user behaviors that are data-dependent, or identifying the source of a false claim. There are however, lot of ambiguities that challenge the successful automation of fake news detection. Most of the time, linguistic characteristics are not fully understood, hand-crafted features are data-specific and laborious, and merely source identification does not lead to fake news detection [4].

References [21, 22] report that previous techniques have also focused on extracting various features, incorporating these features into supervised classification models, such as naive Bayes, decision tree, logistic regression, k nearest neighbor, and support vector machines, and then selecting the classifier that performs the best.

Broadly, fake news detection is organized in the literature based on three criteria: knowledge(fact checking), context and style. Some of the knowledge-based detection approaches [23, 24] presume that web resources or the frequency by which a fact is mentioned can be used as an indication of truth. As a result, methods have been proposed to determine the truthfulness of a given web resource. Reference [25] gives a comprehensive overview of challenges with this kind of approach. Reference [26] tries to assess the authenticity of a given fact by forming queries and then analyzing results to find out if the fact is either strongly or weakly supported. Context-based approaches like [27, 28, 29, 30] model how fake news is spread in social networks and propose algorithms to limit their spread. Results from [30] also report that exceeding a certain threshold in spreading a statement to be wrong or false is sufficient to remove the misinformation from the network. Also, [31] combines analysis from social network and linguistic features to identify rumors as they spread. Based on the hypothesis that deception has a style of its own, style-based approaches like [22, 32, 21] have also been proposed to differentiate between fake and bona fide news articles. Some of the stylometric features, among others, that formed the basis of such investigations were character and stop word n-grams, readability indices, and the average number of words per paragraph.

Recently, embedding techniques such as word embedding and deep neural networks are also attracting attention, as they can learn better representations needed for fake news detection [6]. However, the lack of manually labeled fake news dataset is still a bottleneck for advancing computational-intensive, broad coverage models in this direction. The very intent of false news is misleading and determining its correctness requires interpretation by people with domain expertise. Reference [5] reports gathering

news data with annotations in the following ways: Expert journalists, Fact-checking websites, Industry detectors, and Crowd-sourced workers. Some publicly available datasets include BuzzFeedNews [33], LIAR [6], BS Detector [34], CREDBANK [35] and Emergent [36].

1.2 Contributions

In order to tackle the above challenges involved, we make the following contributions in this thesis:

- In data-driven feature learning, we propose a novel approach of constructing new features without using any domain knowledge.
- In this, we also present an unsupervised feature construction procedure to solve the problem of manual feature engineering and demonstrate qualitative and quantitative results.
- We experimentally demonstrate the effectiveness of our approach on datasets from diverse domains across multiple classifiers.
- We propose a learning model, AutoLearn, that *automates feature engineering* by mining pairwise feature associations without any domain knowledge.
- The design of AutoLearn is simple, based on regression between feature pairs, and captures *behavioral trends* in the dataset.
- AutoLearn limits the new feature space by selecting a very small number of new features to reach the desired performance thus making it *scalable* in contrast to the infinite feature space considered by models proposed in prior literature.
- AutoLearn uses *stability selection*, a known robust technique for feature selection that removes unnecessary features and prevents overfitting.
- AutoLearn is *generic* as our experiments show it to provide more accurate estimates for various classifiers on datasets belonging to various domains.
- AutoLearn works effectively even when number of records and features are as small as 62 and 4 respectively.
- Finally, we design a novel attention based hybrid network for the problem of fake news detection. This approach integrates the article information along with the meta data and claims of the news articles. Our experimental results on the largest existing dataset demonstrates, that attention based hybrid network works better than the current state-of-the-art approaches.

1.3 Thesis Organization

The thesis is organized as follows. In Chapter 2, we introduce a novel data-driven feature learning approach. This is the basis of our first segment's work. In Chapter 3, we present our proposed AutoLearn, a feature learning model which is a robust and improved version of our data-driven feature learning approach, performing thorough experiments. Chapter 4 demonstrates how these techniques perform for problems specific to Gene Expression datasets. In Chapter 5, we introduce the problem of fake-news detection and develop an attention based hybrid network that works better than the current state-of-the-art approaches. Chapter 6 gives the conclusion along with a brief summary.

Chapter 2

Data Driven Feature Learning

2.1 Overview

In recent past, the impressive success of deep learning techniques [37, 38] has substantiated 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. A common practice, is to use all available features and leave the problem of identifying the useful feature sets to the learning model. 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. Furthermore, most construction algorithms employ special-purpose construction methods and heuristics that are especially suited to their underlying representation. There are, however, several problems with this scheme. For instance, given a new classification problem, it is not obvious which of the various representations and associated algorithms should be selected. In many real-world classification problems, the target concept is best expressed by features constructed using domain-specific knowledge.

In this work, we propose a model to alleviate difficulties involved in feature engineering through automation. We develop a learning model that automates discovery of *underlying patterns* 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. We present a novel method for feature generation, that captures the prominent variations in feature pairs via regression, by picking only those data points that are relevant and leads to highly discriminative information. We experimentally demonstrate the effectiveness of our approach on datasets from diverse domains across multiple classifiers.

2.2 Proposed Approach

Rationale for proposed design. 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 pattern, from

which it is much easier to identify the class. 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 earlier works, 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 to discover underlying patterns by the way feature pairs are related to each other. *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 feature pairs, this variation will be very *prominent* and would result in highly discriminative information. Our goal is to use regression as a means to discriminate between how features influence each other across classes and to mine feature relationships, linear or non-linear and their forecast. Our model has following 3 steps.

(1) Mining Correlated Features: Distance correlation [39] is used to determine if there exists a predictive relationship of interest for a given feature pair. We begin with original feature space 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 in Algorithm 1) and hence only the correlated data points (feature pairs) are picked up in this step for feature generation. The correlation between a given pair of feature vector can be linear (Line 8 in Algorithm 1) or nonlinear (Line 5 in Algorithm 1). We maintain a threshold parameter η_1 in order to segregate linear and nonlinear dependencies, respectively.

(2) Feature Generation: While *mining correlated features* detects the implicit patterns by the way features are related to each other, the *feature generation step* discovers and captures those prominent patterns and its variations that result in highly discriminative information. Here, given a training set of independent and dependent variables, we use regression algorithms to seek a connection between them for feature construction. Specifically, we use linear regression (LR) and support vector regression (SVR) techniques to determine linear and non linear relations respectively. We are using each feature to predict the values of other features by applying regression, and including those predicted values (regression forecast) to supplement the original feature space. For every correlated feature pair, the feature constructed is the forecast (prediction values) which we get by regressing F_i on F_j for every correlated feature pair (Line 6 and 9 in Algorithm 1) where F_i and F_j are independent and dependent variables respectively. 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. The features generated try to learn the underlying patterns between correlated feature pairs which models proposed in prior literature fail to capture, thus improving prediction accuracy.

A new feature is essentially the regression of one feature (independent variable) on another (dependent variable). This regression can be linear or non-linear. 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.

Algorithm 1 Feature Generation

Input : Input Features F_d, η_1

Output: Newly constructed feature space F_N

$i = 0, j = 0, F_N = \phi$ **while** $i < d$ **do**

  while $j < d$ **do**

    if *(*$i \neq j$*) and (*$dcor(F_i, F_j) \neq 0$*)* **then**

      if *(*$dcor(F_i, F_j) > 0$*) and (*$dcor(F_i, F_j) < \eta_1$*)* **then**

        $F_N \leftarrow F_N \cup SVR(F_\mathrm{i}, F_\mathrm{j})$

      end

      if *(*$dcor(F_i, F_j) \geq \eta_1$*) and (*$dcor(F_i, F_j) \leq 1$*)* **then**

        $F_N \leftarrow F_N \cup LR(F_\mathrm{i}, F_\mathrm{j})$

      end

    end

    j++

  end

  i++

end

return F_N

(3) Feature Selection: All the newly constructed features F_{N} might not be of equal importance. We use stability based selection [40] only on the newly constructed features to choose a subset of these features as it helps in increasing the classification accuracy by eliminating irrelevant features and prevents overfitting. In stability selection, data is perturbed several times (by iteratively sub-sampling the examples). For each perturbation, any learning model like regression, SVM etc. that produces sparse coefficients is applied to a sub-sample of the data. After a number of iterations, all features that were selected in a large fraction of the perturbations are chosen. Finally, a cutoff threshold η_2 ($0 < \eta_2 < 1$) is applied in order to select the most stable features. In this paper, instead of proposed lasso [41] we use Randomized lasso [42] for stability selection.

2.3 Experimental Setup and Results

We compare our model with 2 other top performing models, namely, FCTree (FCT) and TFC on 20 datasets (shown in Table 2.1) from diverse domains. 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 [8, 14]. Our model results for the supplemented feature space are mentioned under **AL*** column in Table 2.2-2.6. The accuracy is reported after 5-fold cross-validation on 8 state-of-the-art classification (CLF) algorithms 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). Note that for both feature generation and feature selection only the training set in each fold of cross-validation was used. The accuracy of the classifiers is computed using the scikit-learn's [43] default parameters. The default scikit-learn parameters were used for LR and SVR as well. For SVR, we use rbf kernel as it provides good generalization capabilities. The parameter value for η_1 was set at 0.7 since this value best segregates linear and non linear correlations as discussed in [39]. The value of parameter η_2 was determined by doing grid search on values $\{0.05, 0.1, 0.15, 0.2, ..., 0.7\}$. It is apparent from the results, demonstrated in Table 2.2-2.6 that for most of the scenarios, the classifiers can achieve much higher accuracy using our proposed model as compared to the original features, TFC and FCTree methods. We achieved an improvement of **12.17%** in accuracy over the original feature space across 20 datasets and 8 different classifiers. In order to verify that this improvement is mainly due to feature construction step rather than feature selection, we applied feature selection alone on the original feature space which resulted in an overall improvement of **3.93%**. This difference in overall improvement proves that the major improvement is due to our feature construction step.

2.4 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 not burdened

Table 2.1: Characteristics of 20 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
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
Ovarian Cancer	15154	253	2
Poker	10	1025010	10
Prostate Cancer	5966	102	2
Shuttle	10	58000	7
Sonar	60	208	2
Wine	13	178	3

Table 2.2: Accuracy comparison across 8 classifiers

Dataset	CLF	ORIG	TFC	FCT	AL*
Abalone	KNN	**23.27**	21.64	22.60	22.71
	LR	24.61	23.69	23.90	**25.56**
	SVM-L	25.71	25.64	25.72	**26.07**
	SVM-P	19.46	17.64	22.12	**22.77**
	RF	22.91	18.78	**23.02**	22.11
	AB	**20.61**	19.10	19.97	20.11
	NN	27.53	26.32	26.41	**27.81**
	DT	19.27	19.00	19.13	**19.41**
Arcene	KNN	80.5	80.5	80.5	**82.50**
	LR	**86.00**	84.00	84.00	85.50
	SVM-L	**88.50**	86.50	87.50	86.50
	SVM-P	**88.00**	87.00	86.00	85.50
	RF	76.50	76.23	76.92	**77.90**
	AB	72.50	74.00	75.00	**77.12**
	NN	65.50	68.97	69.95	**82.00**
	DT	69.00	69.00	69.00	**72.58**
Bank	KNN	99.92	99.27	99.52	**99.70**
	LR	98.90	97.95	98.68	**99.70**
	SVM-L	98.76	98.97	99.27	**99.92**
	SVM-P	98.90	98.27	99.16	**99.63**
	RF	99.05	98.27	98.75	**99.56**
	AB	99.63	99.27	**99.78**	99.48
	NN	**100.00**	99.02	99.02	99.92
	DT	98.25	98.12	98.56	**99.19**
Colon	KNN	78.97	79.34	79.56	**80.38**
	LR	75.38	74.68	75.12	**78.18**
	SVM-L	75.38	74.07	**76.02**	74.10
	SVM-P	**73.97**	70.16	71.29	70.69
	RF	70.64	71.37	71.73	**72.30**
	AB	72.56	71.23	73.06	**75.76**
	NN	62.82	65.67	69.12	**78.84**
	DT	75.89	75.16	**76.27**	73.97

Table 2.3: Accuracy comparison across 8 classifiers

Dermat.	KNN	89.11	90.46	92.89	**96.09**
	LR	97.21	97.76	97.97	**98.61**
	SVM-L	**97.21**	96.02	96.27	96.92
	SVM-P	**94.41**	94.00	94.12	93.56
	RF	**96.92**	96.45	96.61	95.81
	AB	54.13	57.12	**61.00**	54.96
	NN	98.04	97.13	97.22	**98.22**
	DT	**95.24**	95.06	94.96	94.68
E.coli	KNN	86.59	**88.42**	87.56	84.82
	LR	75.88	78.23	79.24	**87.19**
	SVM-L	85.71	85.71	85.71	**86.30**
	SVM-P	56.54	59.32	62.14	**80.33**
	RF	82.73	83.46	83.76	**86.59**
	AB	62.47	63.54	64.37	**65.75**
	NN	78.28	80.37	81.97	**86.90**
	DT	**79.74**	76.32	77.67	76.48
Fourier	KNN	83.85	82.17	**83.82**	83.55
	LR	79.45	79.97	80.00	**82.20**
	SVM-L	81.45	81.15	82.86	**83.05**
	SVM-P	8.70	42.25	57.97	**79.30**
	RF	**79.9**	78.90	79.16	79.31
	AB	48.65	46.66	49.29	**50.40**
	NN	81.90	82.34	83.12	**84.50**
	DT	74.00	74.00	**74.35**	**74.35**
Pixel	KNN	97.75	98.12	97.23	**97.95**
	LR	94.35	94.22	94.28	**95.75**
	SVM-L	92.9	92.57	93.26	**94.27**
	SVM-P	98.35	98.22	**98.66**	97.25
	RF	95.5	94.26	**95.12**	94.20
	AB	54.05	54.00	54.86	**55.60**
	NN	97.15	97.15	**97.75**	**97.75**
	DT	87.30	86.12	86.78	**86.95**

Table 2.4: Accuracy comparison across 8 classifiers

Ionosp	KNN	84.31	84.66	**84.87**	83.46
	LR	87.44	87.26	87.39	**87.95**
	SVM-L	87.44	86.71	**87.78**	84.30
	SVM-P	64.10	70.16	71.45	**74.63**
	RF	93.15	91.65	**93.16**	92.30
	AB	92.02	90.94	90.12	**92.43**
	NN	**93.14**	92.45	92.13	92.29
	DT	88.32	87.12	88.04	**88.59**
Letter	KNN	95.02	95.00	**95.96**	**95.96**
	LR	71.66	77.42	80.07	**83.71**
	SVM-L	54.96	57.98	58.81	**61.23**
	SVM-P	95.01	95.12	95.26	**96.52**
	RF	93.96	93.74	93.79	**94.14**
	AB	27.82	28.00	29.07	**30.38**
	NN	91.67	92.45	93.45	**95.22**
	DT	87.78	88.03	88.34	**90.12**
Leukae.	KNN	82.09	85.31	87.12	**94.27**
	LR	84.76	87.64	91.21	**93.75**
	SVM-L	84.26	86.83	88.92	**95.64**
	SVM-P	65.23	69.31	74.15	**81.90**
	RF	**90.28**	89.48	89.86	90.19
	AB	92.95	92.11	92.49	**93.15**
	NN	86.09	90.11	91.37	**93.81**
	DT	83.33	83.95	84.13	**86.00**
Libras	KNN	70.00	71.00	**71.18**	69.44
	LR	60.27	64.68	67.12	**70.00**
	SVM-L	68.61	69.88	**70.83**	67.22
	SVM-P	2.22	36.68	47.97	**49.44**
	RF	71.94	72.12	**73.07**	70.22
	AB	8.05	10.12	13.11	**18.05**
	NN	71.66	72.35	74.24	**76.33**
	DT	62.5	62.64	63.12	**65.55**

Table 2.5: Accuracy comparison across 8 classifiers

Dataset	CLF	ORIG	TFC	FCT	AL*
Lung	KNN	88.88	88.96	89.88	**92.61**
	LR	91.93	93.14	91.96	**93.92**
	SVM-L	91.95	92.64	92.66	**93.79**
	SVM-P	90.09	88.46	**90.32**	88.81
	RF	87.03	86.47	88.62	**90.03**
	AB	82.72	83.01	83.17	**83.33**
	NN	74.69	76.88	79.43	**90.44**
	DT	88.30	**89.17**	88.75	85.22
Lymp.	KNN	87.42	87.67	**88.12**	84.26
	LR	**87.52**	86.07	86.31	86.42
	SVM-L	85.47	85.37	85.31	**88.52**
	SVM-P	58.26	60.37	62.49	**68.73**
	RF	76.05	77.34	76.46	**79.05**
	AB	52.15	51.46	**52.71**	50.84
	NN	83.36	84.65	85.12	**85.36**
	DT	63.42	64.34	64.89	**68.73**
Ovarian	KNN	93.29	95.64	95.97	**97.12**
	LR	**100.00**	99.00	99.00	**100.00**
	SVM-L	**100.00**	99.24	99.41	**100.00**
	SVM-P	64.01	67.23	69.21	**91.66**
	RF	97.23	96.27	95.46	**97.62**
	AB	98.80	97.45	98.12	**99.27**
	NN	54.50	59.62	63.12	**88.41**
	DT	94.47	95.37	96.12	**97.63**
Poker	KNN	61.76	58.27	62.78	**64.14**
	LR	50.11	54.62	57.47	**59.46**
	SVM-L	49.74	46.54	47.44	**50.13**
	SVM-P	58.32	59.23	**61.17**	60.02
	RF	68.1	70.32	71.51	**72.26**
	AB	35.76	36.23	37.00	**39.46**
	NN	**99.72**	99.57	99.57	99.57
	DT	63.81	64.33	64.16	**66.17**

Table 2.6: Accuracy comparison across 8 classifiers

Dataset	CLF	ORIG	TFC	FCT	AL*
Prostate	KNN	87.23	88.25	89.32	**91.19**
	LR	90.19	90.89	91.46	**92.14**
	SVM-L	91.14	90.46	**91.19**	90.19
	SVM-P	**91.19**	88.12	89.97	88.47
	RF	86.28	87.16	88.30	**91.19**
	AB	88.19	88.30	89.12	**90.19**
	NN	50.90	54.67	59.12	**86.69**
	DT	77.61	77.21	78.37	**79.42**
Shuttle	KNN	99.78	99.02	99.57	**99.92**
	LR	92.90	93.31	93.76	**96.28**
	SVM-L	90.41	91.19	92.18	**96.57**
	SVM-P	**99.92**	99.81	99.81	99.88
	RF	**99.97**	99.72	99.72	99.81
	AB	87.50	89.17	90.42	**92.46**
	NN	99.71	99.52	99.71	**99.98**
	DT	**99.98**	99.18	99.52	99.67
Sonar	KNN	78.35	81.48	82.70	**83.19**
	LR	77.42	78.12	78.72	**79.00**
	SVM-L	73.54	74.54	75.75	**77.30**
	SVM-P	53.36	58.41	66.44	**81.71**
	RF	73.55	81.00	**82.54**	77.87
	AB	80.74	80.00	**81.04**	78.83
	NN	80.30	81.07	82.00	**84.09**
	DT	75.01	74.23	74.52	**75.02**
Wine	KNN	67.93	74.89	79.93	**90.12**
	LR	95.52	96.89	97.24	**98.30**
	SVM-L	83.03	88.14	89.94	**91.31**
	SVM-P	96.65	**96.68**	96.65	94.68
	RF	96.07	96.68	**97.12**	**97.12**
	AB	85.82	**88.12**	91.23	85.71
	NN	42.73	46.23	49.56	**87.19**
	DT	91.57	91.79	92.01	**93.02**

with the curse of dimensionality, and are hence scalable. 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 prior literature. Figure 2.1 illustrates the number of features finally selected by our learning model against the number selected by FCTree and TFC on 20 datasets. Clearly, we can conclude that the number of features required by our model are significantly less. Our model[1] on an average, uses at least *six times* lesser number of features than the best performing proposed techniques across all datasets.

[1] Saket Maheshwary, Ambika Kaul, Vikram Pudi, **Data Driven Feature Learning**, In Workshop Proceedings of International Conference on Machine Learning (ICML) 2017, Sydney, Australia.

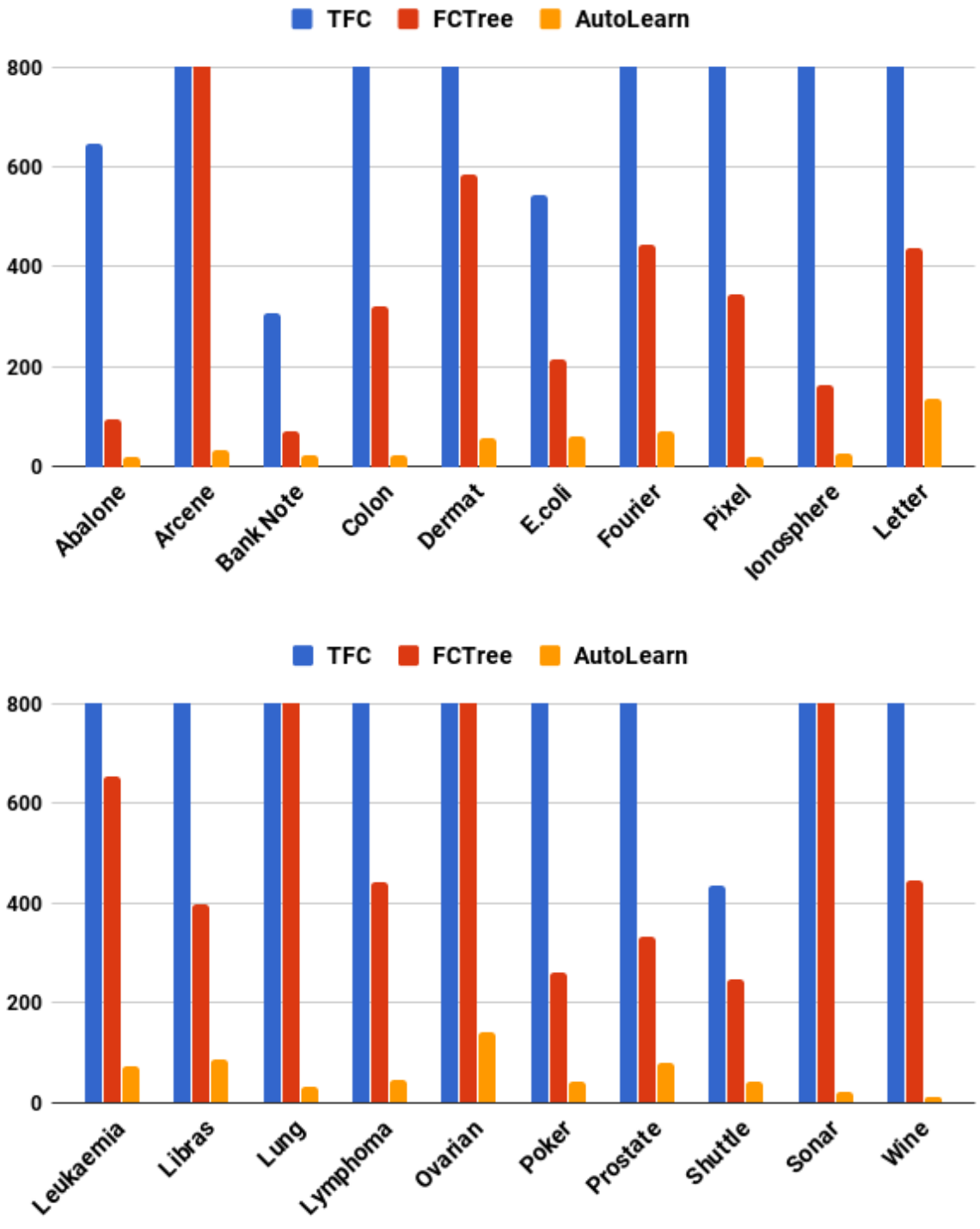

Figure 2.1: Scalability Analysis on 20 datasets

Chapter 3

AutoLearn: Automated Feature Generation and Selection

3.1 Overview

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 [44]:

- 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.

- 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.

3.2 AutoLearn

3.2.1 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.

3.2.2 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 3.1, 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 [39].

- **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.

3.2.2.1 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 2. This is particularly helpful for high dimensional real world datasets.

Algorithm 2 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 3.2, 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 3.3) 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 [45].

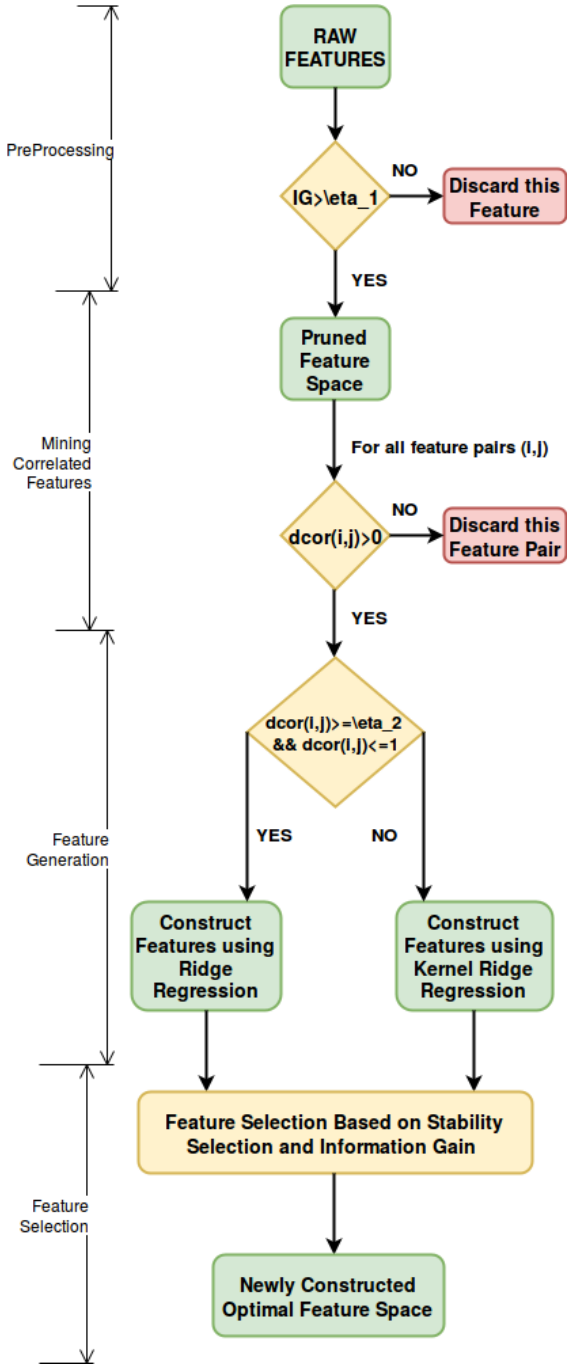

Figure 3.1: Block diagram demonstrating the architecture of AutoLearn

3.2.2.2 Mining Correlated Features

The correlation or association between features is determined using distance correlation [39], a measure that decides whether there exists a predictive relationship of interest for a given feature pair. This

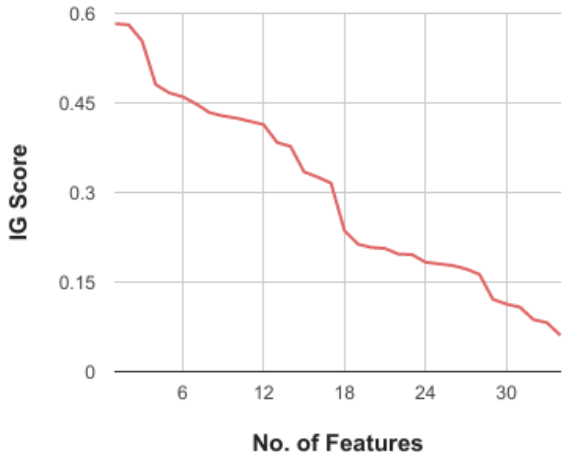

Figure 3.2: Illustrated IG on Dermatology dataset for original features

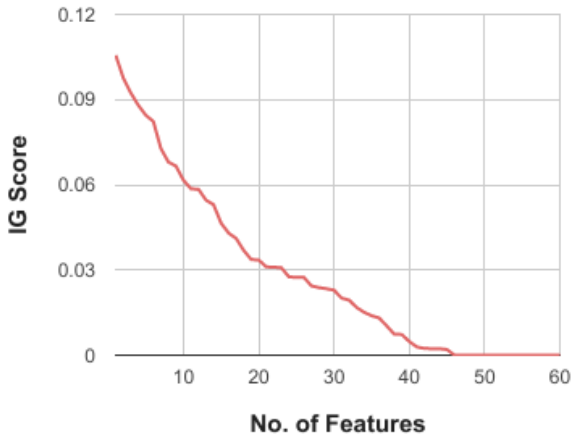

Figure 3.3: Illustrated IG on Sonar dataset for original features

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:

$$dcov^2(u,v) = \int_{R^{d_u+d_v}} \|\phi_{u,v}(t,s) - \phi_u(t)\phi_v(s)\|^2 w(t,s)dtds \tag{3.1}$$

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

$$w(t,s) = \{c_{d_u}c_{d_v}\|t\|_{d_u}^{1+d_u}\|s\|_{d_v}^{1+d_v}\}^{-1} \tag{3.2}$$

where the term c_d is given as:

$$c_d = \pi^{(1+d)/2}/\Gamma\{(1+d)/2\} \tag{3.3}$$

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

$$dcor(u,v) = \frac{dcov(u,v)}{\sqrt{dcov(u,u)dcov(v,v)}} \tag{3.4}$$

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 [46] can only measure linear relationship whereas Spearmanś rho [47] and Kendallś tau [48] 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 3). The correlation between a given pair of feature vector can be linear (Line 9 Algorithm 3) or nonlinear (Line 5 Algorithm 3). We maintain a threshold parameter η_2 in order to segregate linear and nonlinear dependencies, respectively.

3.2.2.3 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.

Algorithm 3 Feature Generation

Input : Preprocessed Features F_d, η_1

Output: Newly constructed feature space F_N

$i = 0, j = 0, F_N = \phi$;

while $i < d$ **do**

  while $j < d$ **do**

    if *(*$i \neq j$*) and (*$dcor(F_i, F_j) \neq 0$*)* **then**

      if *(*$dcor(F_i, F_j) > 0$*) and*

      ($dcor(F_i, F_j) < \eta_2$*)* **then**

        $F_N \leftarrow F_N \cup KRR(F_{\mathrm{i}}, F_{\mathrm{j}})$;

        $F_N \leftarrow F_N \cup (F_j - KRR(F_{\mathrm{i}}, F_{\mathrm{j}}))$;

      end

      if *(*$dcor(F_i, F_j) \geq \eta_2$*) and*

      ($dcor(F_i, F_j) \leq 1$*)* **then**

        $F_N \leftarrow F_N \cup RR(F_{\mathrm{i}}, F_{\mathrm{j}})$;

        $F_N \leftarrow F_N \cup (F_j - RR(F_{\mathrm{i}}, F_{\mathrm{j}}))$;

      end

    end

    j++ ;

  end

  i++ ;

end

return F_{N}

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 $\mathbb{R}^{\mathrm{p}}$ and response variable $\{y_i\}_{i=1}^n$ in $\mathbb{R}$. 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,

$$\min_{w} \|Xw - y\|_2^2 + \alpha\|w\|_2^2 \tag{3.5}$$

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 3) 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 3). 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 [7, 8, 14] fail to capture this.

3.2.2.4 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 [40] 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 3 and is elaborated below:

Stability Based Selection: Stability based selection [49] 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 [40], the authors used lasso [41] 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.

$$\hat{\beta} = \arg\min \beta \in \mathbb{R}^{\mathrm{p}} \|X - Y\beta\|_2^2 + \lambda \sum_{k=1}^{p} |\beta_k| \tag{3.6}$$

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 [42]. The Randomized lasso changes the penalty lambda to a randomly chosen value in a predefined range, according to the equation given below.

$$\beta^{\hat{\lambda},W} = \arg\min \beta \in \mathbb{R}^{\mathrm{p}} \|X - Y\beta\|_2^2 + \lambda \sum_{k=1}^{p} \frac{|\beta_k|}{W_k} \tag{3.7}$$

The re-weighting is not based on any previous estimate, but is simply chosen randomly. According to [42], 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.

3.3 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 3 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 3.4. Here, the y axis shows the relative importance which is measured using a facility available in the forests of trees implementation in [43]. Figure 4.4 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.

3.4 Evaluation

3.4.1 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 [50]. Their characteristics are discussed in Table 3.1. 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).

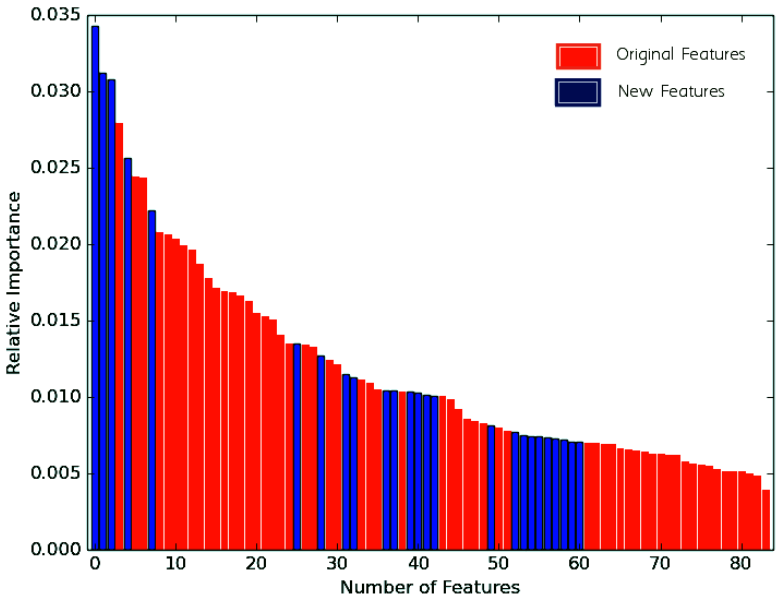

Figure 3.4: Relative feature importance of Sonar dataset

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.

3.4.2 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 state-of-the-art models namely FCTree (FCT) [14], TFC [8] and ExploreKit (EK) [15]. 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 [8, 15]. Our model results for the supplemented feature space are mentioned under **AL*** column in the Tables 3.2-3.8. 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.

3.4.3 Parameter Settings

The value of the first parameter η_1 (refer Figure 3.1) 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 [45]. The parameter value for η_2 was set at 0.7 since this value best segregates linear and non linear correlations as discussed in [39]. The accuracy of the classifiers was computed using the scikit-learn's [43] 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 $\{0.1, 0.2, ..., 0.5\}$.

3.4.4 Comparative Evaluation

Tables 3.2-3.8 show how Figure 3.1'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 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 3.9. The results demonstrated in Table 3.9 validate the effectiveness of all 4 major steps of our learning model as discussed in previous section. All the improvements mentioned in Table 3.9 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 3.1 and Table 3.9, 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 3.1): (1) *original features* combined with our features generated just after the *feature generation step* (Table 3.9, row 2) and (2) *Original features* combined with the *newly constructed optimal feature space* (Table 3.9, row 3). The $\%$ improvement for these are noteworthy.

3.5 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 not burdened with the curse of dimensionality, and are hence scalable. This is also the interpretation in prior litera-

ture [14]. 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 3.5 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 [51] on an average uses at least *ten* times lesser number of features than the best performing submitted techniques across all datasets.

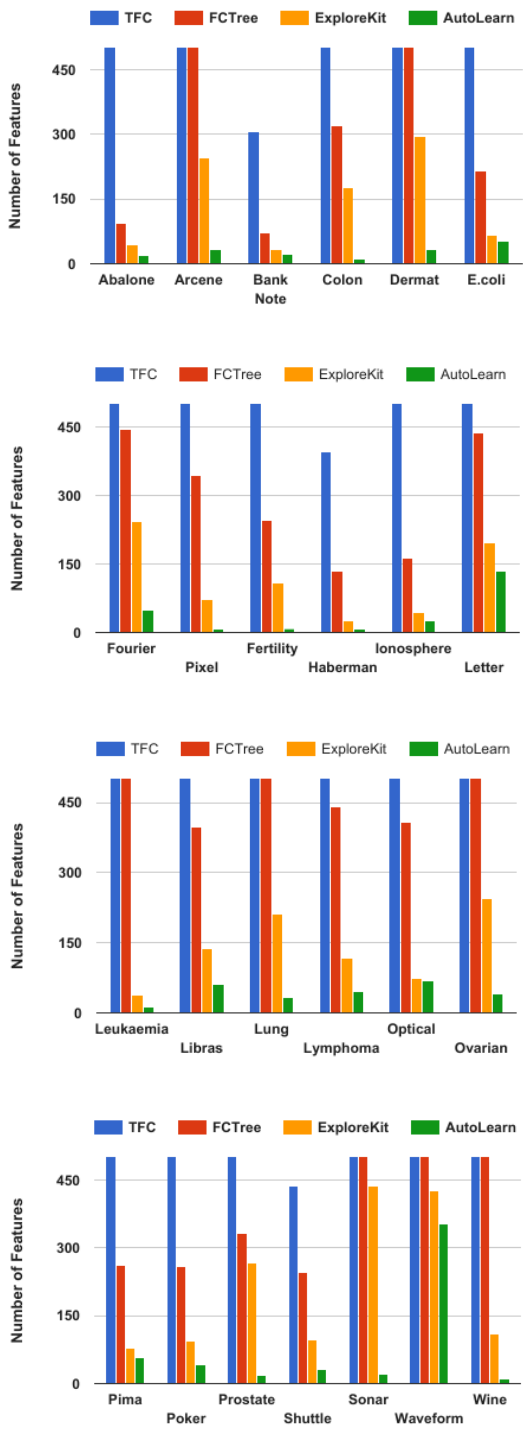

Figure 3.5: Scalability Analysis of TFC, FCT, EK and AL* on 25 datasets

Table 3.1: 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

Table 3.2: Classification Accuracies

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

Table 3.3: Classification Accuracies

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
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**

Table 3.4: Classification Accuracies

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 3.5: Classification Accuracies

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**

Table 3.6: Classification Accuracies

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**
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**

Table 3.7: Classification Accuracies

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

Table 3.8: Classification Accuracies

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**

Table 3.9: 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 2.2-2.8)	13.28
With only 2-step Feature Selection on preprocessed features	4.12

Chapter 4

Automated Representation Learning for Gene Expression Microarrays

4.1 Overview

This work introduces and validates a method of automated analysis of gene expression microarray data, that offers an opportunity for knowledge discovery in the fields of biology and medicine. Gene expression microarrays are the comprehensive snapshots of biological activity in a cell or tissue. Their correct analysis is important and can help in improving the diagnosis and prognosis, in identifying novel targets for drug design, and in treatment planning for those suffering from a disease. We show how regression-based feature learning can be used for cancer-type prediction from gene expression data. Our approach uses this microarray data to mine pairwise feature associations, identify the linear or non-linear relationship between each pair, apply regression on each class and select those relationships using autoencoders, that are stable and improve the prediction performance.

Any new development in biology, for instance the technique to measure the activity of thousands of genes at once, poses interesting challenges to researchers in the field of data mining. Gene expression microarrays measure the level of activity of genes within a given tissue and thus provide information regarding the complex activities within the corresponding cells [52]. Analyzing this data correctly, can contribute towards the discovery of gene regulatory targets, disease diagnosis and drug development [17].

While this data can hold probes for tens of thousands of genes, limited resources like time and money often lead to smaller number of experiments [18]. As a result, gene expression microarray data is characterized by high dimensionality and low sample size (very few records). For efficient classification, engineering a good feature representation is an essential step that aims at implementing an appropriate transformation restructuring the original data into a new and more revealing form.

For gene expression data, feature engineering generally requires substantial manual effort in designing and selecting features and is tedious and non scalable. Moreover, a predictive model cannot simply be defined in terms of reduced variables due to lot of inter-correlations and inter-dependencies between variables. Recent efforts, such as [17, 18] automate the feature extraction process for microarray data. These demonstrate the potential of the approach, and show the need for further exploration and refine-

ment. Moreover, due to the high dimensionality of the datasets, these approaches are not scalable. In this work, we propose a learning model that automates feature engineering for high-dimensional, low sample size gene expression microarray data. Based on regression between feature pairs, our model discovers underlying patterns and their variations in the data, by the way features are related to each other. A transformed subset is then selected using an autoencoder based deep learning approach to create a significant improvement in predictive performance. As such, our approach can be used on any dataset from different domains which have similar challenges of limited or unbalanced data, but in this study, we specifically aim to further improve the quality of cancer diagnosis and classification. We experimentally demonstrate the merits of our approach on a large group of datasets and multiple classifiers, achieving on an average 10.27% improvement in the prediction accuracy, compared to original feature space and 4.96% over other top performing models.

4.2 Proposed Approach

4.2.1 Preprocessing

For high-dimensional gene expression datasets, utilizing all the available features for feature construction is time consuming and ineffective. Hence we begin with an initial step of preprocessing, that is divided in two phases. In first phase, Information Gain (IG) is used for feature selection and in second phase, we use KPCA [18] to get the principal components of original features. Also, [45] motivated us to choose approximately top 4-5% of features overall, jointly given by both the phases. During Phase-I, approximately 2-3% features are selected based on the parameter tuning of IG score (η_1) for each dataset. Note that IG method can choose both linear and non-linear features given it qualifies for an IG score above the threshold (Line 4 Algorithm 4). However, ranking features based on *only* information gain to filter out irrelevant features may miss important feature interactions. This is a common problem in machine learning where a feature assessed alone may look as a poor predictor, but when looked in combination with other features, the assessment changes drastically. The preprocessing step may be discarding important features and thus a second step of KPCA was introduced. The number of components chosen depends on the parameter value C. The features F_d which we get after both the steps are then combined (Line 10 Algorithm 4) and sent as input for feature generation step.

4.2.2 Mining Correlated Features

The correlation between features is determined using distance correlation [39], a measure that decides whether there exists a predictive relationship of interest for a given feature pair. In this step, independent (non-correlated) feature pairs are filtered out (Line 4 Algorithm 5). The correlation between a given feature pair can be linear (Line 9 Algorithm 5) or nonlinear (Line 5 Algorithm 5). We maintain a threshold parameter η_2 in order to segregate linear and nonlinear dependencies. This measure is based on the Fourier transform, or characteristic function of sets of random variables and the related

Algorithm 4 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

$F_{KPCA} = KPCA(F_{orig}, C)$;

$F_d \leftarrow F_d \cup F_{KPCA}$;

return F_d

characterization of independence. It determines 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:

$$dcov^2(u, v) = \int_{R^{d_u+d_v}} \|\phi_{u,v}(t, s) - \phi_u(t)\phi_v(s)\|^2 w(t, s)dtds \tag{4.1}$$

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

$$w(t, s) = \{c_{d_u} c_{d_v} \|t\|_{d_u}^{1+d_u} \|s\|_{d_v}^{1+d_v}\}^{-1} \tag{4.2}$$

where the term c_d is given as:

$$c_d = \pi^{(1+d)/2} / \Gamma\{(1 + d)/2\} \tag{4.3}$$

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

$$dcor(u, v) = \frac{dcov(u, v)}{\sqrt{dcov(u, u)dcov(v, v)}} \tag{4.4}$$

4.2.3 Feature Generation

In this section, we discuss our regression-based approach for feature construction in detail. The goal is to generate new features that capture feature interactions. These feature interactions help in determining underlying relations which results in better representation. We use regularized regression models as

a basic tool for modelling relations between dependent and independent variables. Regularized ridge regression models add additional constraints or penalty to a model, with the goal of preventing overfitting and improving generalization. Such models improve generalization capabilities especially when pair of features are highly correlated.

Algorithm 5 Feature Generation

Input : Preprocessed Features F_d, η_1

Output: Newly constructed feature space F_N

$i = 0, j = 0, F_N = \phi$;

while $i < d$ **do**

  while $j < d$ **do**

    if *($i \neq j$) and ($dcor(F_i, F_j) \neq 0$)* **then**

      if *($dcor(F_i, F_j) > 0$) and ($dcor(F_i, F_j) < \eta_2$)* **then**

        $F_N \leftarrow F_N \cup KRR(F_\mathrm{i}, F_\mathrm{j})$;

        $F_N \leftarrow F_N \cup (F_j - KRR(F_\mathrm{i}, F_\mathrm{j}))$;

      end

      if *($dcor(F_i, F_j) \geq \eta_2$) and ($dcor(F_i, F_j) \leq 1$)* **then**

        $F_N \leftarrow F_N \cup RR(F_\mathrm{i}, F_\mathrm{j})$;

        $F_N \leftarrow F_N \cup (F_j - RR(F_\mathrm{i}, F_\mathrm{j}))$

      end

    end

    j++ ;

  end

  i++ ;

end

return F_N

Given a training set, our algorithm groups the available set of attributes by class and uses regularized regression algorithms to seek an underlying pattern between a feature pair for generation a new feature for each class. Specifically, we use ridge regression (RR) and kernel ridge regression (KRR) techniques to determine and model linear and non linear predictive relations respectively. Following are the two types of features that are generated per class. This procedure is carried out for all the classes in the dataset.

- 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 5) where F_i and F_j are independent and dependent variables respectively. Let this newly generated feature be denoted by $F_{forecast}$. Note

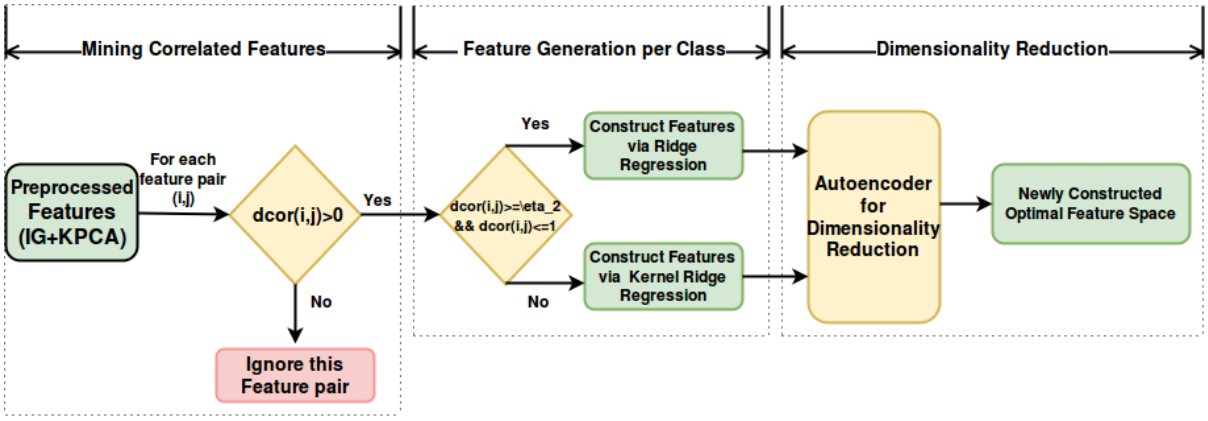

Figure 4.1: Illustration of the proposed workflow

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 5). 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.

The feature generation algorithm models the data on a per class basis. Note that since we are grouping attributes by class and then iteratively applying Algorithm 5 for all available classes, hence the feature learning model is supervised. While the step of *mining correlated features* discovers implicit patterns in the way features are related to each other, the *feature generation* step captures those prominent variations that result in highly discriminative information. These two categories of features try to learn the actual relationship between correlated feature pairs, thus improving the prediction accuracy. Other models proposed in prior literature fail to capture this. Figure 4.1 below demonstrates the overall workflow of the proposed approach.

4.2.4 Autoencoder for Dimensionality Reduction

An autoencoder is an artificial neural network used for unsupervised learning of efficient codings. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. If linear activations are used, or only a single sigmoid hidden layer,

then the optimal solution to an autoencoder is strongly related to principal component analysis (PCA). With appropriate dimensionality and sparsity constraints, autoencoders can learn data projections that are more interesting than PCA or other basic techniques. It turns out that PCA only allows linear transformation of data vectors. Autoencoders, on other hand, are non-linear by the nature, and thus, they can learn more complicated relations between visible and hidden units. Moreover, they can be stacked, which makes them even more powerful. We use the following settings.

1. **Input to Autoencoder:** The features generated by Algorithm 5 are given as input to the autoencoder. We used single hidden layer to encode the input.

2. **ReLU:** Non-linear activation function, Rectified linear unit (ReLU) is preferred by us to sigmoid or hyperbolic-tan because it simplifies backpropagation, makes learning faster while also avoiding saturation. ReLU is defined as $f(z) = max(0, x)$.

3. **Adam:** In recent times, several algorithms (with implemented software tools) are available for training a deep neural network. While stochastic gradient descent (SGD) for quite some time have been the top choice, there has been study which indicate some of the obvious flaws [53] in the vanilla implementation. There have been some attempts to automatically tune its learning rate thus resulting in much faster convergence. For our model we used Adam [54] instead of SGD which required a lot of fine-tuning with the learning rate and over 500 epochs to converge.

Using an autoencoder allows us to learn an efficient, compressed and distributed representation (encoding) for the set of features generated by our learning algorithm.

4.3 Experimental Setup

We evaluate our approach on 20 classification datasets. These are the gene expression microarray datasets with large variety in size, number of attributes and variation in number of classes. Their characteristics are discussed in Table 4.1. We evaluate our model on 6 state-of-the-art classification algorithms, namely, Logistic Regression (LR), SVM with linear kernel (SVM), Random Forest (RF), Adaboost (AB), Multi Layered Perceptron (NN) and Decision Tree (DT). For each classifier, the comparisons are made among (1) Original features, (2) Features constructed using our approach ((KP+IG) + FG + FS), (3) Original and our features together (O + (KP+IG) + FG + FS)), (4) Approach proposed by [18], referred to as ICML 2013 in the results, and (5) Approach proposed by [8], referred to as Binary operators. We have appended the original features in approaches 4 and 5 as well. 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.

Table 4.1: Characteristics of 20 datasets

Dataset Name	Features	Instances	Labels
Alon et al	2,000	62	2
Borovecki et al	22,283	31	2
Burczynski et al	22,283	127	3
Chiaretti et al	12,625	111	2
Chin et al	22,215	118	2
Chowdary et al	22,283	104	2
Christensen et al	1,413	217	3
Gashaw et al	12,625	20	2
Golub et al	7,129	72	3
Gordon et al	12,533	181	2
Gravier et al	2,905	168	2
Khan et al	2,308	63	4
Nakayama et al	22,283	105	10
Petricoin et al	15154	153	2
Shipp et al	6,817	58	2
Singh et al	12,600	102	2
Sorlie et al	456	85	5
Su et al	5,565	102	4
West et al	7,129	49	2
Yeoh et al	12,625	248	6

4.4 Results

4.4.1 Comparative Evaluation

Tables 4.2-4.3 show how our proposed approach performed in comparison to other baseline models discussed above. We achieved **10.27%** and **4.96%** improvement in terms of accuracy over the original feature space and the features learnt by other top performing models respectively, across all 20 datasets and 6 different classifiers.

Table 4.2: Classification Accuracies-I

Dataset	Approach	LR	SVM	RF	AB	NN	DT
Alon	Original features	**94.28**	**97.14**	70.47	77.61	58.57	77.61
	Our approach	91.29	92.18	**80.95**	81.94	87.14	84.28
	Original + our approach	91.29	90.95	80.47	**87.61**	**93.10**	**87.61**
	ICML 2013	90.25	91.42	77.61	85.21	90.95	74.76
	Binary operators	90.25	88.68	74.76	78.09	84.76	77.61
Borovecki	Original features	**97.14**	**97.14**	**97.14**	**97.14**	58.09	84.76
	Our approach	**97.14**	**97.14**	94.28	94.50	**93.81**	93.81
	Original + our approach	93.80	93.80	**97.14**	93.80	**93.81**	**97.14**
	ICML 2013	93.80	93.80	93.80	90.00	84.28	90.48
	Binary operators	93.80	90.47	93.80	80.00	81.00	90.48
Burczynski	Original features	**94.46**	**96.06**	73.10	54.24	44.00	61.38
	Our approach	91.26	90.52	**79.45**	**69.23**	82.58	70.12
	Original + our approach	92.46	88.91	74.76	68.52	**85.04**	**73.96**
	ICML 2013	78.71	76.35	73.13	66.8	74.67	69.23
	Binary operators	70.84	71.63	72.4	65.23	70.00	68.52
Chiaretti	Original features	**84.30**	**84.33**	71.01	73.41	39.66	67.97
	Our approach	76.52	78.12	**78.15**	75.04	**78.06**	62.64
	Original + our approach	71.96	77.29	77.32	77.26	72.70	**71.16**
	ICML 2013	66.49	66.24	75.69	**80.46**	72.61	70.43
	Binary operators	64.15	59.47	70.24	70.30	70.33	60.21
Chin	Original features	85.61	84.78	79.81	84.81	62.57	73.84
	Our approach	82.25	78.87	79.67	81.37	81.34	76.44
	Original + our approach	**85.75**	**85.91**	84.02	**88.11**	84.02	77.12
	ICML 2013	79.71	80.54	**87.12**	84.18	**86.52**	**78.94**
	Binary operators	77.06	77.75	81.44	80.61	69.27	72.10
Chowdary	Original features	96.09	**98.04**	95.19	95.19	63.38	90.42
	Our approach	**98.04**	**98.04**	95.19	90.42	**95.23**	87.52
	Original + our approach	**98.04**	**98.04**	**96.14**	92.23	94.23	89.28
	ICML 2013	96.14	**98.04**	93.28	**96.14**	89.47	**92.33**
	Binary operators	90.42	97.09	93.23	90.42	86.52	83.76
Christensen	Original features	**100.00**	**100.00**	**100.00**	99.08	**99.53**	97.68
	Our approach	99.07	99.07	99.54	98.60	**99.53**	97.20
	Original + our approach	99.53	99.53	**100.00**	99.08	**99.53**	98.16
	ICML 2013	99.53	99.53	99.53	**99.54**	**99.53**	**99.53**
	Binary operators	98.16	98.16	98.61	98.89	99.06	93.97
Gashaw	Original features	40.00	40.00	35.00	40.00	45.00	55.00
	Our approach	60.00	**70.00**	**60.00**	**65.00**	**65.00**	**60.00**
	Original + our approach	65.00	**70.00**	**60.00**	**65.00**	**65.00**	**60.00**
	ICML 2013	**70.00**	**70.00**	55.00	55.00	**65.00**	**60.00**
	Binary operators	**70.00**	**70.00**	55.00	55.00	**65.00**	**60.00**
Golub	Original features	**95.80**	**95.80**	83.23	92.95	41.71	84.66
	Our approach	**95.80**	94.47	**97.14**	93.14	95.80	**94.38**
	Original + our approach	**95.80**	94.47	94.57	93.04	**95.90**	91.61
	ICML 2013	95.71	94.47	94.38	**95.80**	94.57	90.38
	Binary operators	93.04	93.04	93.14	91.71	91.52	90.00
Gordon	Original features	**100.00**	**100.00**	95.57	**99.44**	85.10	92.25
	Our approach	99.44	99.44	**98.88**	97.79	**99.44**	96.66
	Original + our approach	99.44	99.44	98.34	96.11	98.33	**97.23**
	ICML 2013	99.44	99.44	95.61	**99.44**	96.12	95.01
	Binary operators	97.77	98.33	94.47	92.85	95.58	94.50

4.4.2 Parameter Settings

The value of the first parameter η_1 (threshold for IG) was determined by grid search on values $\{0.01,0.05,0.10,0.15,...,1.0\}$. The parameter value for η_2 was set at 0.7 since this value best segregates

Table 4.3: Classification Accuracies-II

Dataset	Approach	LR	SVM	RF	AB	NN	DT
Gravier	Original features	75.52	76.72	64.90	**73.19**	74.34	72.56
	Our approach	76.13	**77.34**	68.37	**73.19**	76.73	69.03
	Original + our approach	**77.34**	75.52	70.14	**73.19**	**78.52**	70.19
	ICML 2013	76.72	77.01	**71.39**	**73.19**	77.36	**74.91**
	Binary operators	66.70	70.80	70.00	71.94	69.57	68.88
Khan	Original features	98.46	98.46	83.97	77.43	87.43	79.35
	Our approach	**100.00**	**100.00**	**98.33**	**93.84**	**100.00**	84.10
	Original + our approach	98.46	**100.00**	95.12	91.79	98.46	85.76
	ICML 2013	98.46	98.46	91.79	92.05	98.46	**87.30**
	Binary operators	96.92	96.92	88.97	90.38	96.92	80.89
Nakayama	Original features	**71.42**	**70.47**	55.23	42.85	27.61	52.38
	Our approach	68.86	66.02	**57.14**	**44.76**	56.19	52.38
	Original + our approach	70.02	60.00	56.19	42.85	**66.66**	**53.33**
	ICML 2013	70.02	60.00	54.28	42.85	61.90	**53.33**
	Binary operators	58.09	58.09	48.57	40.00	57.14	49.52
Petricoin	Original features	**100.00**	**100.00**	91.50	98.02	44.08	96.04
	Our approach	98.00	97.33	**99.33**	97.33	96.70	**98.04**
	Original + our approach	**100.00**	**100.00**	97.36	**98.04**	98.02	**98.04**
	ICML 2013	**100.00**	**100.00**	98.04	**98.04**	**99.33**	94.75
	Binary operators	97.33	94.06	96.10	96.08	93.44	92.68
Shipp	Original features	**96.08**	**96.08**	85.83	**89.75**	74.00	**85.83**
	Our approach	84.41	85.75	85.91	81.83	83.08	83.08
	Original + our approach	84.41	85.75	**87.16**	**89.75**	**93.33**	82.08
	ICML 2013	84.41	85.75	85.91	**89.75**	89.41	81.91
	Binary operators	84.41	85.75	83.00	72.66	77.91	77.91
Singh	Original features	**94.09**	**93.09**	75.47	**91.14**	49.00	79.47
	Our approach	89.23	**93.09**	85.38	88.68	87.23	74.52
	Original + our approach	92.14	**93.09**	**87.28**	85.28	89.19	79.47
	ICML 2013	93.09	**93.09**	83.38	90.19	**90.14**	**81.28**
	Binary operators	89.23	**93.09**	84.28	89.28	80.33	80.00
Sorlie	Original features	**90.58**	**90.58**	72.94	41.17	90.58	51.76
	Our approach	**90.58**	**90.58**	78.82	52.94	**91.76**	65.88
	Original + our approach	**90.58**	**90.58**	**81.87**	**61.17**	**91.76**	**68.23**
	ICML 2013	88.23	88.23	80.00	47.05	88.23	63.52
	Binary operators	85.88	85.88	75.29	44.70	82.35	61.17
Su	Original features	**99.00**	**99.00**	89.14	67.52	96.04	91.09
	Our approach	**99.00**	**99.00**	90.14	68.52	**98.00**	93.09
	Original + our approach	**99.00**	**99.00**	**95.09**	71.42	97.00	**95.04**
	ICML 2013	**99.00**	**99.00**	89.23	**74.19**	96.04	**95.04**
	Binary operators	96.00	96.00	91.00	66.47	95.04	92.14
West	Original features	59.33	57.33	47.55	61.33	53.33	65.33
	Our approach	63.55	**67.33**	**68.00**	**71.77**	67.77	**69.55**
	Original + our approach	63.55	63.55	57.33	67.11	63.33	65.55
	ICML 2013	**67.55**	63.55	65.55	67.11	**71.55**	67.55
	Binary operators	61.55	63.55	53.33	57.33	55.33	61.33
Yeoh	Original features	96.77	**97.58**	87.08	72.97	14.48	87.48
	Our approach	95.96	96.78	**90.74**	73.05	**92.74**	84.36
	Original + our approach	**97.18**	96.78	89.93	**79.06**	90.69	84.36
	ICML 2013	96.35	96.37	88.73	76.58	89.23	**88.70**
	Binary operators	91.34	95.97	80.66	74.53	87.86	85.87

linear and non linear correlations as discussed in [39]. The accuracy of the classifiers was computed using the scikit-learn's [43] 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.

4.4.3 Scalability Analysis

Our experiments for scalability analysis show the final number of new features after construction and selection. This ensures that classification algorithms that use these features as input are not burdened with the curse of dimensionality, and are hence scalable. 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. We can conclude that the number of features required by our model are significantly less. Our model on an average uses at least *six* times lesser number of features than the best performing submitted techniques across all datasets.

Chapter 5

Hybrid Attention based network for Fake News Detection

5.1 Introduction

The low cost, easy access and rapid circulation of information over the internet has encouraged more people than ever, to seek out and consume news from online sources or social media rather than traditional news organizations. Equally true is the fact, that never in history has more *false* information been available to more people. The vast spread of fake news can lead to negative impacts on individuals and society, and once it is online, it is *news* forever.

The fake news articles are produced online for a variety of purposes, such as financial and political gain. It was shown recently, that the most popular fake news was more widely spread on Facebook than the most popular authentic mainstream news [3]. Therefore, in order to debunk not only the authenticity but also the intent of the fake news, growing research on fake news detection has recently become a center of tremendous attention, thereby raising the level of urgency.

For some, it may be a matter of curiosity, their interest aroused by an alarming headline or a sensational photo. But some people believe the information they find on false news sites, even when its not backed by established facts or scientific evidence. Although the term "fake"news has been circulating for several years, the new elements of social media and the global networks of friends who use it have proved to be a much more powerful force for its circulation. A plausible reason for this can be that people are less skeptical of information they encounter on platforms they have personalized, to reflect their interests and identity[1]. They let their guards down where friends, family members, and coworkers share a wide variety of information.

There are some emerging patterns that can be utilized for fake news detection. Most of the online news is collected from different sources, such as news agency homepages, search engines, and social media websites. While the problem of fake news detection is not new to the natural language community, the computational approaches have mostly relied on fact-checking websites such as PolitiFact.com[2] and

[1]https://hbr.org/2018/07/what-do-we-know-about-false-news

[2]http://www.politifact.com

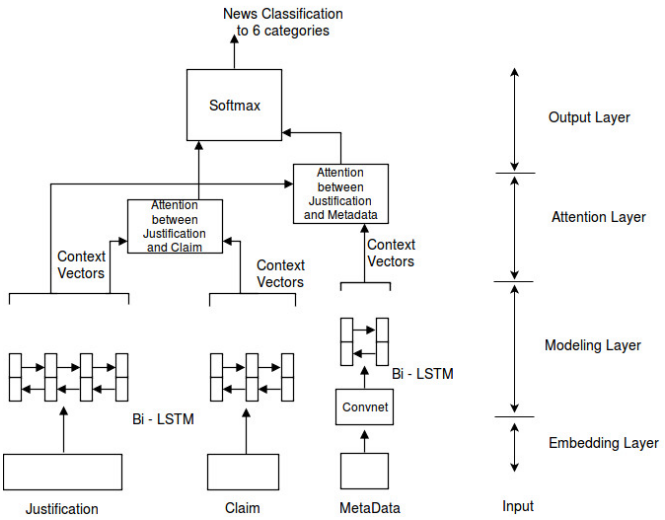

Figure 5.1: Architecture Diagram of Hybrid Attention based network

Snopes.com[3] among others. These require human expertise, thus making it difficult to obtain datasets that generalize over several domains [55].

Keeping in mind that fake news is intentionally written to mislead readers to believe false information, it can be difficult to detect it, solely based on news content. Therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination [5]. Exploiting such information can be challenging but useful.

This work builds in the direction of automating fake news detection. To help mitigate this problem, it is important to develop techniques to automatically detect fake news. The basis of our work is the LIAR dataset [6], which is the largest public fake news dataset. With less than a thousand samples in other datasets, it is impractical to use them as benchmarks for developing and evaluating machine learning algorithms for fake news detection.

The remainder of the work is organized as follows. In next section, we first explain the dataset used and then describe in detail, our hybrid attention based network, that performs seemingly better than other state-of-the-art approaches. We also give a detailed experimental setup along with the evaluation results.

5.2 Proposed Approach

In this section, we will introduce our proposed hybrid attention mechanism. The overall architecture of our model is illustrated in Figure 5.1. As demonstrated, our model follows a multi-level hierarchy and has the following *five* levels.

[3]https://www.snopes.com

- **Input Layer:** This layer takes three different kinds of inputs namely the Statement (claim), Justification and the Metadata (speaker, context, etc.). These inputs vary largely in terms of the average number of tokens they possess.

- **Embedding Layer:** This layer contains character level as well as the word level embeddings. (1) Character embedding layer maps each word to a vector space using character-level Convolutional neural netwoks (CNNs). (2) Word embedding layer maps each word to a vector space using a pre-trained word embedding model.

- **Contextual Layer:** It utilizes contextual cues from surrounding words to refine the embedding of the words.

- **Attention Layer:** In this layer, we use two attention based models namely word level based attention and sentence level based attention.
  (1) Word level based attention couples each metadata type and the justification context to a vector space that is metadata keyword aware. (2) Sentence level based attention couples each statement with the justification context vector.

- **Output Layer:** This layer predicts one of the six labels (provided in the dataset) namely pants-fire, false, barely-true, half-true, mostly-true, and true, hence classifying the news taking into account the statement, justification as well as the metadata.

We will now explain each of these components of our model below in detail.

5.2.1 Input Layer

The input layer contains three different modalities of the text i.e document (Justification), sentence (Statement/claim) and words (Metadata). We model and tackle each of them via our proposed model in an attempt to solve the problem of automated fake news detection.

5.2.2 Character level Embeddings

This layer is responsible for mapping each word to a high-dimensional vector space. Let $\{w_1, w_2, ...w_N\}$ represent words in the input. In order to obtain the character level embedding of each word we used CNNs as proposed recently [56] for modelling characters in the word. Characters are embedded into vectors. These vectors are then accountable as 1-dimensional inputs to the CNN, and whose size is the input channel size of the CNN. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word. Note that we used character level embeddings to model the input for metadata only.

5.2.3 Word level Embeddings

Word embedding layer also maps each word to a high-dimensional vector space. We use pre-trained GloVe [57] vectors of 100 dimension to obtain the fixed word embedding of each word. The words not found in the GloVe pre-trained embeddings were initialized randomly. Both, *Justification* as well as the *Statement* inputs are modelled using pre-trained word embeddings at the first step. Metadata does not use any kind of word level embeddings.

5.2.4 Contextual Layer

We use a Bidirectional Long Short-Term Memory Network (Bi-LSTM) on top of the embeddings provided by the embedding layers to model the interactions between words. Bi-LSTM places a Long Short-Term Memory Network (LSTM) in both directions, and concatenates the outputs of the two LSTMs.

5.2.5 Attention Layer

The deep learning attention mechanism is based on the idea of selecting the *most relevant* piece of information, rather than using *all* available information. While the exact idea of involving attention in neural processes has been largely studied in computational neuroscience [58, 59], a similar idea of focusing on specific parts of the input has been applied in deep learning [60], for speech recognition, translation, reasoning, and visual identification of objects. It is effectively used in natural language processing (NLP) as well [61, 62]. In text, attention mechanism allows fluent information flow by making each target side word directly and dynamically affected by subparts of the source sentence. It firstly leverages source hidden states *H* and the target hidden state to obtain the attention weights, and then outputs an attentive context vector that is further used to generate the target word. Concretely, in generating the *i*-th target word, the attention mechanism weights each source hidden state h_j according to:

$$\alpha_{ij} = \frac{\exp\left(e_{ij}^{\alpha}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}^{\alpha}\right)}, \tag{5.1}$$

where the energy function

$$e_{ij}^{\alpha} = A_{\alpha}\left(s_{i-1}, h_j\right) \tag{5.2}$$

is the key alignment model, typically in the form of feedforward neural network. In our work, we set it to be the same as [61]:

$$A_{\alpha}\left(s_{i-1}, h_j\right) = v_a^T \tanh\left(W_a s_{i-1} + U_a h_j\right) \tag{5.3}$$

Then the attentive context vector c_i^{α} is calculated by:

$$c_i^{\alpha} = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{5.4}$$

Apparently, such context vector is based on hidden states *H*. We therefore in the rest of the paper, name such context vector c_i^{α} as *hidden context*.

In our problem setting as well, with three modalities involved, all the words (sentences) do not contribute equally to represent the meaning of a sentence (document). Hence, we introduce attention mechanism to extract such words (sentences) that are important to the meaning of the sentence (document) and aggregate the representation of those informative words (sentences) to form a vector that is more informative using word as well as sentence level attention mechanism.

Word Attention This idea is motivated by the approach suggested by [63]. Here attention mechanism extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector.

$$u_{it} = \tanh\left(W_w h_{it} + b_w\right) \tag{5.5}$$

$$\alpha_{it} = \frac{\exp\left(u_{it}^T u_w\right)}{\sum_t \exp\left(u_{it}^T u_w\right)} \tag{5.6}$$

$$s_i = \sum_t \alpha_{it} h_{it} \tag{5.7}$$

We first feed the word annotation h_{it} through a one-layer MLP to get u_{it} as a hidden representation of h_{it} , then we measure the importance of the word as the similarity of u_{it} with a word level context vector u_w and get a normalized importance weight α_{it} through a softmax function. After that, we compute the sentence vector as a weighted sum of the word annotations based on the weights. The word context vector is randomly initialized and jointly learned during the training process.

Sentence Attention In order to reward sentences that are clues to correctly classify fake articles, we use attention mechanism and introduce a sentence level context vector u_s and use the vector to measure the importance of the sentences. This yields

$$u_i = \tanh\left(W_s h_i + b_s\right) \tag{5.8}$$

$$\alpha_i = \frac{\exp\left(u_i^T u_s\right)}{\sum_i \exp\left(u_i^T u_s\right)} \tag{5.9}$$

$$s_i = \sum_i \alpha_i h_i \tag{5.10}$$

where *v* is the document (Justification) vector that summarizes all the information of sentences in a document. Similarly, the sentence level context vector is randomly initialized and jointly learned during the training process.

5.2.6 Output Layer

The feature vector generated after concatenation of the output of attention layer gives a high level representation of the Justification, Metadata and the Statement and are used as features for classifying fake news:

$$p = softmax\left(W_c v + b_c\right) \tag{5.11}$$

We use the negative log likelihood of the correct labels as training loss:

$$L = -\sum_{d} \log p_{dj}, \tag{5.12}$$

where *j* is one of the 6 labels of news articles *d*.

5.3 Experiments

5.3.1 Dataset

We use the LIAR dataset [6] which includes 12,836 short statements labeled for truthfulness, subject, context/venue, speaker, state, party, and prior history [6]. The order of magnitude is larger than any currently available resources [64, 36] of similar type. These statements are crawled using POLITIFACT.COM's API and each statement is evaluated by POLITIFACT.COM editor for its truthfulness. There are six fine-grained labels for the truthfulness ratings: *pants-fire. false, barely-true, half-true, mostly-true, and true*. The distribution of labels in this dataset is also fairly well-balanced: except for 1,050 pants-fire cases, the instances for all other labels range from 2,063 to 2,638 [6].

For experimental purposes, we use this dataset and include (apart from statements), meta-data for each speaker - in addition to party affiliations, the current job, home state, and credit history are also included. We also include the justification supporting the label of each statement. The statistics taken from the LIAR dataset are shown in Table 5.1. Figure 5.2 shows what this dataset looks like i.e some random snippets from the LIAR dataset.

5.3.2 Experimental Setup

We perform an extensive experimentation with the state-of-the art baseline techniques. We also propose a Hybrid Attention based model that provides the best results among all other models. The dataset is divided for training and validation. We try different combinations of basic models, namely: *Bag-of-Words + Logistic Regression - (BOW + LR), Term frequency-inverse document frequency + Logistic Regression - Tfidf + LR, BOW-bigram + LR, Tfidf-bigram + LR, BOW + Tfidf+ LR, BOW +*

Statement: *"Facebook removes, censors Declaration of Independence as hate speech."*

Speaker: headline from Geller Report

Context: a website run by Pamela Geller

Label: Half True

Justification: The Geller Report story had a point that a newspapers post of the Declaration of Independence was temporarily removed from Facebook for appearing to violate "hate speech" standards. But the website did not mention that Facebook said it was a mistake and restored the post. It did not censor the entire speech.

Statement: *"NASA will pay you $100,000 to stay in bed for 60 days!"*

Speaker: Blog post

Context: social media posting

Label: False

Justification: NASA does have a program that pays volunteers to participate in bed rest studies, where one lies down for a long period of time. The point is to allow scientists to understand how the body might be affected by space travel. But the blog post grossly exaggerates the amount of money one would receive for such a task. In the past, rather than being paid $100,000 for 60 days of lying down, volunteers have been paid $18,000 for 70 days of their time.

Statement: "Just in: John Kerry facing prison."

Speaker: Facebook story

Context: social media posting

Label: Pants on Fire

Justification: A headline said "Just in: John Kerry facing prison." The story relies on a tweet that showed Kerry was in Paris – that part is true. But the allegations that Kerry was meeting with Iranian officials in an attempt to salvage the international nuclear agreement are unproven. So saying he violated any law is nothing more than guesswork. We rate this claim Pants on Fire.

Statement: "I wake up every morning in a house (the White House) that was built by slaves."

Speaker: Michelle Obama

Context: speech at the Democratic National Convention

Label: True

Justification: Obama said the White House "was built by slaves." Strictly speaking, the White House was not exclusively built by slaves; it was built by a combination of slaves, free blacks and whites. But slaves were significantly involved in the construction of the White House, so we have no quarrel with the way Obama worded her claim. We rate it True.

Figure 5.2: Some random snippets from the LIAR dataset.

Table 5.1: Statistics of LIAR dataset

Dataset	Statistics
Training set size	10,269
Validation set size	1,284
Testing set size	1,283
Average statement length	17.9
Average article length	xxxx
Top-2 Speaker Affiliations	
Democrats	4,150
Republicans	5,687
None (e.g., FB posts)	2,185

SVM, Tfidf + SVM, BOW-bigram + SVM, Tfidf-bigram + SVM, BOW + Tfidf + SVM, Word2vec + SVM, Word2vec-Pretrained + LR, Doc2vec + LR, Doc2vec + SVM.

The following models: *CNN, RNN, RNN + Attention*, and our *Hybrid Attention based* model use pretrained Glove embeddings of 100 dimensions.

5.3.3 Parameter Settings

The parameters of our proposed model were set as follows:

- Embedding Layer: We used pre-trained GloVe embeddings of 100 dimension. For words whose GloVe vectors were not available, the embedding weights are randomly initialized with uniform distribution in the interval [-0.5,0.5]. For regularization purpose, we set l_2-regularization to 0.0001 and dropout rate to 0.3 [65].

- Optimization: We adopted ADAM [66] optimizer for weight updating, with an initial learning rate of 0.001. We used batched size of 64 samples.

For Bi-LSTM, the number of units were set to 200. All language model features were trained on the training proportion of the dataset. The best filter sizes for the CNN model to model metadata was (2,3,4). In all cases, each size has 128 filters. Implementation is done with Theano [67] backend on Keras.

We used grid search to tune the hyperparameters for LR and SVM models. We chose accuracy as the evaluation metric, since we found that the accuracy results from various models were equivalent to f-measures on this balanced dataset.

Table 5.2: The evaluation results from the LIAR paper. Top section: text-only models. Bottom section: text + meta-data hybrid models

Models	Test
Majority	0.208
SVMs	0.255
Logistic Regression	0.247
Bi-LSTMs	0.233
CNNs	0.270
Hybrid CNNs	
Text + Subject	0.235
Text + Speaker	0.248
Text + Job	0.258
Text + State	0.256
Text + Party	0.248
Text + Context	0.243
Text + History	0.241
Text + All	**0.274**

5.3.4 Evaluation Results

In this section, we report the results of the evaluation of our method with several baseline methods and state-of-the-art approaches. The evaluation results on this dataset, that includes: statement (claim) + meta-data + text (justification), are as shown in Table 5.3. These include using logistic regression, SVMs, word2vec embeddings, doc2vec embeddings, CNN, LSTM and LSTM with attention. Some of these baseline methods and results are reported in [68, 69]. Table 5.3 shows that our method performs the best and Table 5.2 confirms that the reason in improvement in its accuracy is not just because of the hybrid nature of our dataset.

Table 5.3: The evaluation results

Models	Test
BOW + LR	56.11
Tfidf + LR	42.47
BOW-bigram + LR	71.78
Tfidf-bigram + LR	56.11
BOW + Tfidf + LR	56.04
BOW + SVM	53.85
Tfidf + SVM	50.74
BOW-bigram + SVM	71.55
Tfidf-bigram + SVM	65.08
BOW + Tfidf + SVM	53.70
Word2vec + LR	25.79
Word2vec-Pretrained + LR	25.09
Word2vec + SVM	27.66
Word2vec-Pretrained + SVM	26.73
Doc2vec + LR	30.55
Doc2vec + SVM	30.47
CNN	86.79
LSTM	95.87
LSTM + Attention	96.24
Our Model	**98.48**

Chapter 6

Conclusion

6.1 Conclusions

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 for numeric dataset as well as for gene expression dataset. While we do not claim that our features are complete, the above factors combined with our experimental results are very encouraging, and there may be scope for further improvement.

We also work on an important task of automatic fake news detection. Due to its tremendous impact on politics, economy, as well as general society, it poses unique set of challenges. Several models have been proposed to tackle this problem, but lack of sufficient labelled data has greatly hindered the effort to combat fake news. In this particular work, we use liar dataset: an existing publicly available dataset for fake news detection. This corpus comprises of a decade-long, 12.8K manually labeled short statements in various contexts from PolitiFact.com. We use this dataset and in our model, incorporate the complete news articles that will benefit us in capturing the underlying context of the originally available short statements. We design a novel attention based hybrid network which integrates this article information along with the meta data and claims of the news articles. Our experimental results

on the LIAR dataset demonstrate that attention based hybrid network works better than the current state-of-the-art approaches.

Related Publications

- Ambika Kaul, Saket Maheshwary, Vikram Pudi, **AutoLearn: Automated Feature Generation and Selection**, In IEEE Internatioanl Conference on Data Mining (ICDM) 2017, New Orleans, USA.
- Saket Maheshwary, Ambika Kaul, Vikram Pudi, **Data Driven Feature Learning**, In Workshop Proceedings of International Conference on Machine Learning (ICML) 2017, Sydney, Australia.
- Ambika Kaul, Saket Maheshwary, Vikram Pudi, **Hybrid Attention based network for Fake News Detection**, In IEEE Internatioanl Conference on Data Mining (ICDM) 2018 (under review), Singapore.

Bibliography

[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] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[4] N. Ruchansky, S. Seo, and Y. Liu, "Csi: A hybrid deep model for fake news detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 797–806, ACM, 2017.

[5] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[6] W. Y. Wang, "" liar, liar pants on fire": A new benchmark dataset for fake news detection," *arXiv preprint arXiv:1705.00648*, 2017.

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

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

[9] 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.

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

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

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

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

[14] 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.

[15] 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*, pp. 979–984, IEEE, 2016.

[16] 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*, pp. 183–187, 1991.

[17] P. Danaee, R. Ghaeini, and D. A. Hendrix, "A deep learning approach for cancer detection and relevant gene identification," in *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 22, p. 219, NIH Public Access, 2016.

[18] R. Fakoor, F. Ladhak, A. Nazi, and M. Huber, "Using deep learning to enhance cancer diagnosis and classification," in *Proceedings of the International Conference on Machine Learning*, 2013.

[19] C. F. Aliferis, I. Tsamardinos, P. P. Massion, A. R. Statnikov, N. Fananapazir, and D. P. Hardin, "Machine learning models for classification of lung cancer and selection of genomic markers using array gene expression data.," in *FLAIRS Conference*, pp. 67–71, 2003.

[20] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, *et al.*, "Multiclass cancer diagnosis using tumor gene expression signatures," *Proceedings of the National Academy of Sciences*, vol. 98, no. 26, pp. 15149–15154, 2001.

[21] S. Afroz, M. Brennan, and R. Greenstadt, "Detecting hoaxes, frauds, and deception in writing style online," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 461–475, IEEE, 2012.

[22] M. Potthast, J. Kiesel, K. Reinartz, J. Bevendorff, and B. Stein, "A stylometric inquiry into hyperpartisan and fake news," *arXiv preprint arXiv:1702.05638*, 2017.

[23] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[24] A. Magdy and N. Wanas, "Web-based statistical fact checking of textual documents," in *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pp. 103–110, ACM, 2010.

[25] A. L. Ginsca, A. Popescu, M. Lupu, *et al.*, "Credibility in information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 9, no. 5, pp. 355–475, 2015.

[26] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu, "Toward computational fact-checking," *Proceedings of the VLDB Endowment*, vol. 7, no. 7, pp. 589–600, 2014.

[27] D. Acemoglu, A. Ozdaglar, and A. ParandehGheibi, "Spread of (mis) information in social networks," *Games and Economic Behavior*, vol. 70, no. 2, pp. 194–227, 2010.

[28] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," in *Proceedings of the 20th international conference on World wide web*, pp. 665–674, ACM, 2011.

[29] N. P. Nguyen, G. Yan, M. T. Thai, and S. Eidenbenz, "Containment of misinformation spread in online social networks," in *Proceedings of the 4th Annual ACM Web Science Conference*, pp. 213–222, ACM, 2012.

[30] M. Tambuscio, G. Ruffo, A. Flammini, and F. Menczer, "Fact-checking effect on viral hoaxes: A model of misinformation spread in social networks," in *Proceedings of the 24th international conference on World Wide Web*, pp. 977–982, ACM, 2015.

[31] S. Kwon, M. Cha, K. Jung, W. Chen, *et al.*, "Prominent features of rumor propagation in online social media," in *International Conference on Data Mining*, IEEE, 2013.

[32] S. Badaskar, S. Agarwal, and S. Arora, "Identifying real or fake articles: Towards better language modeling," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[33] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of twitter accounts: Are you a human, bot, or cyborg?," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 811–824, 2012.

[34] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[35] M. Del Vicario, G. Vivaldo, A. Bessi, F. Zollo, A. Scala, G. Caldarelli, and W. Quattrociocchi, "Echo chambers: Emotional contagion and group polarization on facebook," *Scientific reports*, vol. 6, p. 37825, 2016.

[36] W. Ferreira and A. Vlachos, "Emergent: a novel data-set for stance classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pp. 1163–1168, 2016.

[37] 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.

[38] 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.

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

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

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

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

[43] 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.

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

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

[46] 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.

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

[48] 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.

[49] 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.), pp. 617–624, Cambridge, MA: MIT Press, 2002.

[50] M. Lichman, "UCI machine learning repository," 2013.

[51] A. Kaul, S. Maheshwary, and V. Pudi, "Autolearnautomated feature generation and selection," in *Data Mining (ICDM), 2017 IEEE International Conference on*, pp. 217–226, IEEE, 2017.

[52] S. Aluru, *Handbook of computational molecular biology*. CRC Press, 2005.

[53] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *ICML*, 2011.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[55] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," *arXiv preprint arXiv:1708.07104*, 2017.

[56] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[57] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[58] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[59] R. Desimone and J. Duncan, "Neural mechanisms of selective visual attention," *Annual review of neuroscience*, vol. 18, no. 1, pp. 193–222, 1995.

[60] S. Maheshwary and H. Misra, "Matching resumes to jobs via deep siamese network," in *Companion of the The Web Conference 2018 on The Web Conference 2018*, pp. 87–88, International World Wide Web Conferences Steering Committee, 2018.

[61] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[62] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[63] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[64] A. Vlachos and S. Riedel, "Fact checking: Task definition and dataset construction," in *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pp. 18–22, 2014.

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[67] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[68] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014.

[69] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1422–1432, 2015.