*Journal of Advances and Scholarly Researches in Allied Education*
*Vol. 16, Issue No. 9, June-2019, ISSN 2230-7540*

# A Modified Framework for Automated Feature Extraction using Deep Feature Synthesis

**Vimla Jethani[1]* Rohit Singhal[2]**

[1] Research Scholar, Sunrise University, Alwar, Rajasthan, India

[2] Research Guide, Sunrise University, Alwar, Rajasthan, India

***Abstract – In recent years, advances in machine learning have led to various innovations in to automating various iterative and time-consuming tasks. One such tasks that data scientists carries out is feature extraction. Feature Extraction is time consuming and requires domain knowledge and chances are few features can be missed. As a result, automating this process will provide ease for data scientists as all possible features can be generated. Automated Feature Engineering majorly focuses on reducing the time require to generate features which can be used to train the models. As a result, a framework is provided to reduce the time required for feature extraction by considering only candidate set of features as an input that also helps to generate only useful features.***

----------X----------

## I. INTRODUCTION

One of the elements for a successful machine learning technique is a properly organized data set. The information needs to be cleaned and organized for the algorithm to induce an excellent model. Preparing a data set efficiently, usually takes a big amount of time, and it calls for strong understanding about the domain. One specifically crucial task is feature engineering, which includes developing new variables that make it less difficult for the algorithm to reap a very good version. Recent years have visible development in automating model choice and hyperparameter tuning, however the most vital issue of the machine learning, feature engineering, has in large part been omitted. Automated feature engineering can bring significant assistance to individuals and organizations. This will help data scientists to focus more on the other steps of machine learning pipeline and iterate successfully.

Feature is a measurable attribute of the object in an attempt to analyze datasets. Features are also termed as “variables” or “attributes”. For instance, customer datasets often comprise of customer id, income and date of joining and so on. Features are the basic building blocks of datasets. In any datasets, features appear as columns. The quality of features there in the dataset have first-rate effect on the quality of insights derived when used for machine learning.

Feature engineering is the process of taking a dataset and building explanatory variables — features — that may be used to educate a machine learning model for a prediction hassle. Often, fact is spread across multiple tables and need to be gathered into a single table with rows comprising the observations and features in the columns. Feature engineering if carried out properly then it increases the predictive power of machine learning algorithms.

Feature engineering is an art to create features from raw data which in turn creates the big difference between good model and bad model. It is the important parameter for any successful model.

The data used to generate predictive model encompasses an outcome variable that holds data which requires prediction. It also holds series of predictor variable that holds data supposed to be predictive of the outcome variable. To illustrate consider an application to predict the property prices. The data which shows actual prices of property is termed as an outcome variable and the data such as size of houses, the location of house, the number of bedrooms in a house is considered as predictor variables and are believed to determine the property value. A "feature" in the framework of predictive modeling is simply another name for a predictor variable.

## II. OVERVIEW OF BACKGROUND AND PREVIOUS WORK

Nowadays, it is becoming quite common to work with datasets which consists of hundreds (or even thousands) of features. However, in machine learning the dimensionality of a dataset is equal to number of variables that are used to represent it.

The aim of feature extraction is to reduce the number of features in the dataset by creating new features from the existing ones or by applying transformations. However, the original features are then discarded. The reduced new set of features consists summarized information about original version. The various manual feature extraction techniques are [2]:

1. **Principal Component Analysis (PCA):** This technique is also termed as linear dimensionality reduction technique and is one of most commonly used techniques. In this technique, the original data is taken as input and trial is carried out to find a combination of the input features which can best summarize the original data distribution so that the original dimensions can be reduced. This is achieved by exploiting variances and minimalizing the reconstruction error by looking at pair wised distances. In this the original data is projected into a set of orthogonal axes and the axes gets ranked based on the order of importance.

2. **Independent Component Analysis (ICA):** This technique is also as a linear dimensionality reduction method that takes as input data a combination of independent components and purposes to appropriately identify each of them (All unnecessary noise is deleted here). The two input features will be considered independent only if both their linear and not linear dependency is equal to zero.

3. **Linear Discriminant Analysis (LDA):** This technique aims to maximize the distance calculated between the mean of each class and minimalize the spreading within the class itself. This is considered as an optimal choice because maximizing the distance between the means of each instance when bulging the data in a lower-dimensional space can yield to better classification results. It is assumed that input data in LDA follows a Gaussian Distribution, consequently applying LDA to not Gaussian data can conceivably lead to poor classification results.

4. **Autoencoders:** Autoencoders are also type dimensionality reduction technique. The difference between Autoencoders and other dimensionality reduction approaches is that Autoencoders use non-linear transformations to create data from a high dimension to a lower one. The different types of Autoencoders such as: Denoising Autoencoder, Variational Autoencoder, Convolutional Autoencoder and Sparse Autoencoder. An Autoencoder basically can be broken down into two main components:

Encoder: It takes the input data and compress it, to remove all the probable noise and obstructive information. The output of the Encoder stage is typically called bottleneck or latent-space.

Decoder: It takes the input i.e. encoded latent space and attempts to reproduce the original Autoencoder input by using compressed form.

If all the given input features are independent of each other, then it will be difficult for autoencoder to encode and decode to input data into a lower-dimensional space.

Feature engineering is one of the most significant steps and also very time-consuming steps. It is very significant because the efficacy of many learning algorithms depends on heavily on input features; therefore, the performances of the same machine learning model with the identical configuration with a set of even slightly different features can vary expressively [3]. Similarly, it is very time-consuming because it requires perception and domain-specific knowledge of the data to create useful features. For example, the champions of the Grupo Bimbo Inventory Prediction competition quantified that 95% of their time was consumed on feature engineering and only 5% on modeling [4].

The area of automated feature engineering is relatively unexplored, which leaves plenty of opportunities for researchers for developing new methods. In this section, related research work of the study is presented.

Kanter, J. M., and Veeramachaneni, K. (2015, October). Deep feature synthesis: Towards automating data science endeavors [1], which applies transformations to all presented features in dataset and selects the most capable among them.

G. Katz, E. C. R. Shin, and D. Song., ExploreKit [5] is the most closely related work that attempts to automate feature engineering. It operates on relational datasets with a single table and exhaustively enumerates all possible features using a predefined set of operators. Then, it employs a machine learning based strategy for feature selection that considers characteristics of the dataset that may affect the likelihood of features being effective and sorts them based on their likelihood. ExploreKit relies on statistical tests to gather characteristics of the dataset and the candidate features and uses them to select features that seem most likely to be effective. While the approach is novel, its effectiveness is unclear. Also, it is unknown whether dataset characteristics can actually help estimate feature

importance. Lastly, using a classification algorithm to select features to improve the performance of another classification algorithm adds unnecessary complexity and uncertainty to the problem. ExploreKit, operates on relational datasets with a sole table for reasons described earlier. Similarly, AutoFE shares a similar collection of operators used to produce candidate features, which contains basic operators such as normalization, discretization, multiplication, and group-by-then-mean that are most frequently used in feature engineering.

Hyunjoon Song, AutoFE: Efficient and Robust Automated Feature Engineering [6] generates a large set of new interpretable features by combining information in the original features. Given an augmented dataset, it discovers a set of features that significantly improves the performance of any traditional classification using an evolutionary algorithm. The architecture of AutoFE is composed of a feature generator, a splitter, a distributed system of feature selectors, and an evaluator. Given a dataset, AutoFE uses a feature generator to generate a large set of new interpretable features and augment the dataset. Then, it uses a splitter to split the augmented dataset into training data, validation data, and test data. With training data and validation data, we run feature selection in a distributed manner. After the best set of features obtained, the model uses an evaluator that trains a classification algorithm with the feature set on training data and evaluates its predictive performance on test data. Here, the author demonstrated the effectiveness and robustness of our approach by conducting an extensive evaluation on 8 datasets and 5 different classification algorithms. It showed that AutoFE can achieve an average improvement in predictive performance for all classification algorithms over their baseline performance obtained with the original features.

DIFER: Differentiable Automated Feature Engineering [7], by Guanghui Zhu, Zhuoer Xu, Xu Guo, Chunfeng Yuan and Yihua Huang, the authors here proposed an effective gradient-based method called DIFER to perform differentiable automated feature engineering in a continuous vector space. Feature optimizer based on the encoder predictor-decoder framework is introduced, which maps features into the continuous vector space via the encoder, optimizes the embedding along the gradient direction induced by the predictor, and recovers better features from the optimized embedding by the decoder. Based on the feature optimizer, it further proposed an evolutionary method to search for better features iteratively. Extensive experiments on classification and regression datasets carried by authors demonstrated that DIFER can significantly outperform the state of-the-art AutoFE methods in terms of both model performance and computational efficiency.

## III. PROPOSED METHODOLOGY

To achieve high performance in any machine learning model it requires engineering good and relevant features. However, when any problem with dataset is given then it is often not clear which attributes are good for achieving better results. The result is that all available system variables / attributes (large numbers of common features) are used as attributes and the problem of identifying most important useful features is the leftover as the task of learning model. Applying such a simple approach is not always good. If we manually select features then it will take considerate amount of time which becomes tedious and also problem specific. So, the work is proposed to consider only candidate features and then carry out the automated feature engineering. For the automated feature engineering Deep Feature Synthesis [1] framework is used.

Deep Feature Synthesis considers input as a set of interconnected entities in which each entity has a primary key that acts as a unique identifier for each instance of an entity on which that the table is based. An entity optionally may also have a foreign key, which is uniquely referred to an instance of a related entity. An instance of any entity has fields which has one of the following data types: numeric data, categorical data, timestamps data and free text data.

In deep feature synthesis mathematical function such as simple, direct, cumulative distribution features, relational aggregation features can be applied at two level i.e., Entity Level and Relational Level. Consider an entity $E^k$ for which features are synthesized:

**Entity level features (EFEAT):** Features calculated here are by considering the fields values in the table related to the entity $k$ alone.

**Relational level:** The features at this level are derived by combinedly analyzing entity $E^i$ related to the entity $E^k$. There are two possible categories of relationships between these two entities: forward and backward.

1. **Forward**: A forward relationship is represented between an instance *m* of entity $E^i$, with a single instance of other entity *i* of entity $E^k$. This is called as the forward relationship because *i* is explicitly dependent on *m*.

**Direct Features (DFEAT)**: Direct features can be applied over the forward relationships. In these, features in a related entity $i \in E^k$ will be transferred directly as features for the $m \in E^i$.

2. **Backward:** A backward relation is represented between an instance *i* of

entity $E^k$, to all instances $m = \{1..M\}$ in $\varepsilon^i$ that further has forward relationship to $k$.

**Relational Features (RFEAT):** Relational Features can be applied over backward relationships. In relationship with backward case r-agg and r-dist operations can be applied. The relational functions here can be applied because every target entity has a set of values related with it in the associated entity.

The deep feature synthesis technique is an automatic feature synthesis algorithm which generates features that are also based on human intuition. The algorithm recursively scans along the relationships in the data then applies mathematical functions over the features in this scanning and further the results are appended in the base table. The final output achieved is an expanded base table which consists of features.

In DFS complete dataset is given as input as a result features generated are not useful with respect to the target variable used for prediction. As a result, a modified approach is presented here which first generates candidate set of features and further automated features are generated on candidate set as result the time required to generate features is also reduced.

**Modified Architecture Overview**

The architecture presented in figure 1, is composed of candidate feature generator, automated feature extraction, data splitter. Given a dataset, a candidate feature generator is used which selects or rejects the attributes based on its usefulness measured with respect to target predictor variable. This can be achieved by calculating the ratio of missing column information and if this is beyond the threshold value (threshold value is defined by the data scientist) i.e., missing information is high then that column is not added to the candidate set. With this technique the candidate set of attributes are generated and then the deep feature synthesis is applied.

Assuming here $y^j$ represents the number of useful candidate-features generated and $z^j$ represents actual features available in the dataset. As candidate feature generator now generates only useful features as a result, $y^j$ is less than $z^j$ which in turn will also lead to a smaller number of recursive calls made by deep feature synthesis algorithm. This approach will lead to reduction in time and also useful features will be generated.

Once this carried out the candidate features are forwarded to AutoML Feature Tools [1], which uses deep feature synthesis to generate large set of features but here it generates features on candidate set provided. Once the useful features are generated then data splitter is used to split the data as training and testing data for training model using machine learning algorithms (here in study decision tree classification is used) with feature set on training data.

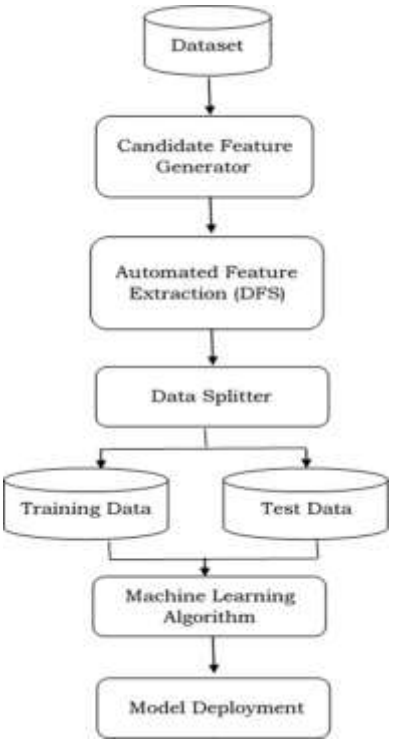


**Figure 1: Proposed framework for generating features**

Once the features are generated, the next step is to build a predictive model framework for prediction problems.

Further various continuous and categorical models can be used for prediction. The model used here is Decision Tree algorithm for predictive modelling. The dataset considered in this work on loan predictions so decision tree is selected for the measuring the performance. Decision trees are extremely useful and its being used in lots of financial industry. However, the approach can be used with any of the machine learning algorithm.

**Decision Tree Classification Algorithm**

Decision Tree [8], is a type of supervised learning algorithm that are used for classification as well as regression problems, but generally it is preferred for solving classification problems. This is a type of tree-structured classifier, where the internal nodes signify the features of a dataset, branches represent the decision rules and each leaf node represents the outcome of the decision made.

To divide the dataset, various splitting rules may be chosen. The most common approach is to use an entropy measure for calculating the information gain (IG) of the split part of dataset, such that,

$$IG(parent, children) = entropy(parent) - [p(c1) \times entropy(c1) + p(c2) \times entropy(c2) + \cdots]$$

It begins at the root node and then recursively divides the dataset in such a way that the IG is highest for each split. This approach is a type of greedy algorithm, as a local optimum is resolved at each split in a try to find the global optimum. It can also be observed that IG measures the difference between entropy of the parent and the weighted sum of the entropy of the children, where each child is denoted $(c_k)$. Entropy for each node is calculated as:

$$entropy = -p_1 \, log(p_1) - p_2 log(p_2)$$

While using Decision Trees, some hyperparameters are also specified. For instance, entropy is not the only way to measure, the Gini index is a common alternative. The maximum number of recursive partitions that are allowed is specified by maximum depth of the tree. The maximum branch specifies the maximum number of the branches that may be split in each node.

The algorithm selected is executed to identify patterns, drawing inferences, and building, and training model by using 80% dataset and remaining 20% of dataset is or validating model. The entire process is carried out while the model training is in progress. Once a model is trained, it provides various statistics that were computed during the training process in the model's evaluation report. The evaluation in machine learning pipeline provides percentage values for the following attributes of a classification model in the form of a confusion matrix:

- True Positive (TP): A true positive result is achieved where the model correctly predicts the positive class.
- True Negative (TN): A true negative result is achieved where the model correctly predicts the negative class.
- False Positive (FP): A false positive result is encountered where the model incorrectly predicts the positive class.
- False Negative (FN): A false negative result is encountered where the model incorrectly predicts the negative class.

The evaluation parameters that are used as a metrics for generating evaluation report are:

**1. Accuracy**

The accuracy is the fraction of total predictions made by the model on the test data that were correct, as a percentage value.

**2. Precision**

The precision is given as the fraction of total positive predictions made by the model that were correct on test data. This indicates how much correct is model's prediction.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ Total\ Predictions}$$

**3. Recall**

The recall is given by the fraction of the true positive predictions made by the model, considering out of all true positives and false negatives results. This parameter helps to select best model when there are highly associated false negative results. The recall is also termed as True Positive Rate.

$$Precision = \frac{TP}{(TP + FP)}$$

**4. F1 Score**

The F1 score is a useful metric for finding balance between precision and recall. It is given as harmonic mean of the precision and recall.

$$F1\ score = \frac{2\ x\ (Precision * Recall)}{(Precision + Recall)}$$

**PERFORMANCE EVALUATION**

The efficacy of the work is demonstrated in this section by performing the tasks as discussed in figure 1. The automated feature extraction is carried out on using Deep Feature Synthesis as well as using modified approach to reduce the time required for generating relevant features is carried out. The work was compiled with system configuration of 16GB RAM and i7 Processor 2.60GHz.

The Home Credit Risk dataset which is publicly available on Kaggle platform [9] is considered, processed and features are generated. A prediction model is built with feedback which indicates improvement in accuracy when data with new trends are recorded and model is retrained. The objective of Home Credit Default Risk Prediction is to predict if a customer will default on loan based on complete data on past loan payments details. The credit default risk is given as the probability that loan is given and there is a chance that money will not be paid back on time. Also, such system helps to ensure that clients who are capable of paying loans are rejected'.

Automated Feature Engineering is carried out using featuretools library which uses deep feature synthesis to extract features. There are seven tables and here information is staggered across multiple tables these tables are referred as entity.

To carry out DFS understanding of relationship is important. Example: application entity has one row for each client and bureau has multiple previous loans for each parent in application data. This represents one is-to-many relationship, as represented in figure 2. Further bureau is represented as a parent of bureau_balance because in this each loan of bureau is represented as multiple monthly records of bureau_balance.

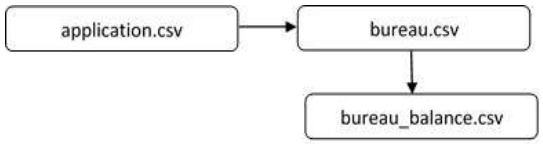


**Figure 2: Relationship between entities**

For each relationship parent and child variable are selected. After applying DFS 7583 features are generated using selected feature primitives. If manually we try to extract features then the domain knowledge is also required to understand the dataset and generate best suitable features. Also, the time required for generating these features will be increased with limited features. Total 121 features were generated using manual feature extraction techniques whereas modified approach using deep feature synthesis algorithm generated 5438 useful features as depicted in figure 3.

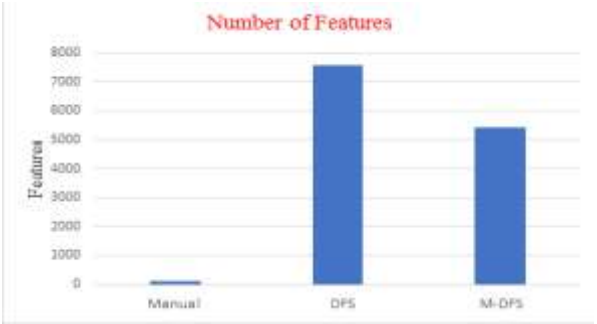


**Figure 3: Features Generated by Manual, DFS and M-DFS approach**

The work was compiled on system configuration with 16GB RAM and it took 44.99 seconds for generating features using DFS and 39.91 seconds using modified approach as shown in in figure 4. The time here represents only the time require generating features.


**Figure 4: Feature Generation Time by DFS and M-DFS**

**Predictions using Decision Tree Classification Results**

The automatic features generated are further check with the target and important features are extracted. These important extracted features are further used to the train the model using training dataset. After training the performance is measured using Decision Tree classification. The classification accuracy of Decision Tree Classification is measured as a percentage value of correct predictions made by the model by the total number of predictions made. Performance of algorithm was evaluated at different stages of training set. The algorithm was trained with records sets containing 140000 records, 160000 records, 180000 records and 200000 records.

**Table 1: Decision Tree Classification Results**

| Number of Records | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| 140000 | 76.22% | 0.73 | 0.77 | 0.75 |
| 160000 | 77.48% | 0.74 | 0.80 | 0.77 |
| 180000 | 79.28% | 0.76 | 0.82 | 0.79 |
| 200000 | 80.65% | 0.78 | 0.82 | 0.80 |

The accuracy, precision, recall rate, F1 score at all stages are measured and represented in table 1 based on the confusion matrix generated during execution. The measured accuracy is depicted in figure 5.


**Figure 5: Decision Tree Classification Accuracy Chart**

Automated feature engineering of AutoML helps to automate the time-consuming tasks of data scientists to generate features automatically. However, using the candidate features more useful features are generated in less time.

## V. CONCLUSION

In today's world machine learning have deeply rooted in our lives. To achieve good performance there is the need of human in every aspect of machine learning pipeline to complete the process. However, this becomes tedious and so there comes a need of automating these tasks. As a part of fulfilling aims, an automated feature engineering approach is presented that provided insights in the key architectural design of the system build which optimizes the time required for automatically extracting useful features. The work demonstrated the implementation of automated feature extraction and selection in detail. The number of useful features generated were 5438 as compared to 7583 features using DFS which generates all possible features that also adds on the time required to extract the features. However, the technique of generating candidate features first helped to generate features less in count but more useful features. But the result achieved are considerably better. Further the model was trained using decision tree classification algorithm on the training data set and the accuracy measured is 80.65%.

## REFERENCES

[1] Kanter, J. M., and Veeramachaneni, K. (2015, October). Deep feature synthesis: Towards automating data science endeavors. In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (pp. 1-10). IEEE.

[2] Diving Deeper into Dimension Reduction with Independent Components Analysis (ICA), Paperspace. Accessed at https:/blog.paperspace.com

[3] Pedro Domingos (2012). A few useful things to know about machine learning. Communications of the ACM, 55(10): pp. 78–87.

[4] Kaggle Team (2016). Grupo Bimbo Inventory Demand, Winners' Interview: Clustifier & Alex & Andrey. http://blog.kaggle.com/2016/09/27/grupo-bimbo-inventorydemand-winners-interviewclustifier-alex-andrey.

[5] G. Katz, E. C. R. Shin, and D. Song (2016). Explorekit: Automatic feature generation and selection. IEEE International Conference on Data Mining,.

[6] Hyunjoon Song (2018). Auto FE: Efficient and Robust Automated Feature Engineering, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

[7] Guanghui Zhu, Zhuoer Xu, Xu Guo, Chunfeng Yuan and Yihua Huang (2020). DIFER: Differentiable Automated Feature Engineering, arXiv - CS - Machine Learning, 2020-10-17, DOI: arxiv-2010.08784

[8] Provost, F., & Fawcett, T. (2015). Data Science for Business. United States: O'Reilly Media

[9] Home Credit Default Risk Data Set: https://www.kaggle.com/

**Corresponding Author**

**Vimla Jethani***

Research Scholar, Sunrise University, Alwar, Rajasthan, India