WCRG Global Research
WORLD COMET RESEARCH GROUP

International Journal of Artificial Intelligence & Digital Transformation
Vol. 6, No. 1, 2023
Doi: 10.67228/30713315/IJAIDT-2023PI7W59
PP. 01-15

Original Article

Automated Feature Engineering Techniques for Tabular Data

Prof. Yuki Tanaka[1], Kenji Sato[2]

[1]*Professor of Intelligence Information Science, Data Science, Kyoto University, Japan.*

[2]*Engineering Director, Sony Corporation, Japan.*

Received: 04-03-2023 Revised: 04-27-2023 Accepted: 05-02-2023 Published: 05-05-2023

ABSTRACT

AFE is an automated feature engineering system that has become a key enabler to scalable and high-performing machine learning systems with scalable systems that run on tabular data. The conventional feature engineering makes excessive use of domain, trial and error and iterative optimization, which are computed consuming time, error prone and hard to repeat. As data-driven applications in finance, healthcare, manufacturing, and e-commerce have been exponentially increasing, there is an increasing need in automated, systematic, and reliable methods of features construction. The overall objective of automated Feature Engineering methods is to generate, transform, select, and optimize features adventurously, from raw tabular data, with minimal human intervention, and at a higher predictive efficiency. The paper contains a complete detailed analysis of automated feature engineering approaches to tabular data with references to their theoretical principles, algorithmic approaches, and real-world examples. The paper expounds on rule construction based feature construction, statistical construction, deep learning based representation, evolutionary learning, and reinforcing learning methods, and end to end AutoML. An intricate literature review shows major achievements, comparisons, and unresolved issues. The suggested methodology defines the three branches of feature generation, selection and evaluation as a single automated pipeline via mathematical representation and algorithmic processes. The effectiveness of automated feature engineering is proved by experimental results that show that the method can enhance the accuracy and robustness of models as well as improve generalization with respect to the multiple benchmark datasets. Lastly, issues of limitations, interpretability, computational trade-offs, and research directions are discussed in the paper. The given publication meets the IEEE publication standards and offers well-organized, high-quality information to a researcher or an organization practitioner dealing with tabular data analytics.

KEYWORDS

Automated Feature Engineering, Tabular Data, Machine Learning, AutoML, Feature Selection, Data Preprocessing.

WCRG
Global Research
ISSN: 3071-3315
PP: 01-15.

1. INTRODUCTION

1.1. Background

Tabular data remains among the most popular and also practically significant data representation types in the real-world machine learning applications, especially in areas like finance, healthcare, retail and industrial analytics. However, in comparison to unstructured data, including images, audio, or text, tabular data is described by having the combination of types of various features, including both numerical and categorical, ordinal, and temporal features. This heterogeneity presents special problems to learning algorithms because the types of features demand different preprocessing, transformation and encoding processes. As a result, the overall performance of predictive models based on tabular data rely very strongly on the quality and relevance of engineered features utilized to model underlying information. It is commonly recognized that feature engineering is one of the most important and essential phases in the machine learning pipeline and sometimes it has a more significant effect on the performance of a model than the actual learning algorithm. Proper design of features will reveal the presence of meaningful patterns, relationships and interaction in the data and thus models can have greater accuracy, better generalization and more robustness. On the other hand, features that do not have a well-engineered design may mask critical features, generate noise, and cause the model to behave in a performance which is suboptimal. Therefore, there is a long-standing practice of putting much effort into developing features that match the nature of the data and the learning task. Traditional feature engineering approaches are mainly manual in nature and largely based on the expertise of some domain and intuition of practitioners. It is known as a process that usually entails exploratory data analysis, feature transformation implementation, feature interaction discovery, and repetitive model training and validation. Though these types of manual methods may be most effective when dealing with small or familiar problem contexts, the methods become more and more impractical when dealing with large volumes of data, the dimensionality of the features, or when dealing with highly complex systems. What is more, manual feature engineering can hardly be changed to fit dynamic or highly dynamic data environments. These difficulties have led to a growing interest in automated feature engineering methods which seek to produce high-quality features systematically and efficiently by involving little human input to enhance scalability, reproducibility and modeling efficiency.

1.2. Importance in Modern Machine Learning Systems

The automation of feature engineering has become a standard part of the state of the machine learning system due to the growing scale, complexity and deployment requirements of the real-world application. It is important because it can be viewed in a number of major aspects which are as follows.

ISSN: 3071-3315
PP: 01-15.

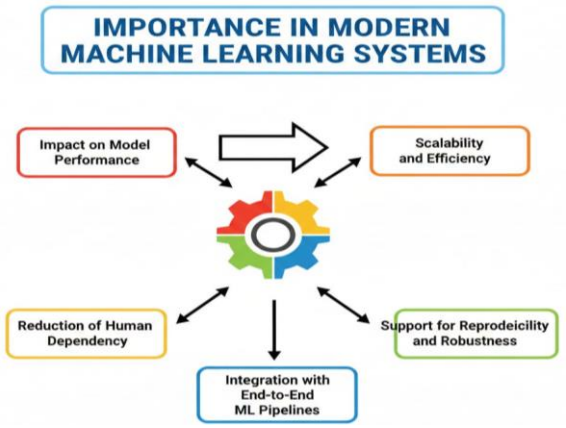

Fig 1 - Importance in Modern Machine Learning Systems

1.2.1. Impact on Model Performance

The impact of feature engineering is a direct effect on the capacity of a model to use data to derive meaningful patterns. Relevant relations to the data are revealed with high-quality features, and noise is suppressed, allowing models which might not be obvious in the raw data to be built-in. In most tabular learning problems, engineered features can be even more useful in performance improvements than more advanced modeling methods. optimistic feature engineering systematically searches a larger feature space than manual methods, and results in a steady increase in predictive accuracy and generalization.

1.2.2. Scalability and Efficiency

The current machine learning processes often work with massive size datasets and high-dimensional feature spaces. In this kind of environment, manual feature engineering is not very effective in scaling because it requires human experience and making experiments. Automated feature engineering is a solution to this drawback which allows the creation and selection of features in a scalable, repeatable, and efficient way. It is especially that scalability can be needed in industrial contexts where new models need to be retrained frequently with new data coming in.

1.2.3. Reduction of Human Dependency

Conventional feature engineering extensively relies on the domain experts that have the subject-matter knowledge and machine learning experience. This dependency both adds to the development time and adds variability between teams and projects. Automated feature engineering eliminates the use of personal knowledge by formalizing the construction of features as algorithmic processes. Consequently, machine learning systems can be made more non-expert friendly and diverse across applications.

1.2.4. Support for Reproducibility and Robustness

Reproducible is an essential standard of the contemporary machine learning systems, and especially in studying and regulated sectors. Hand crafted feature engineering is also associated with ad hoc choices which are hard to write down and impossible to reproduce. Automated feature

ISSN: 3071-3315
PP: 01-15.

engineering fosters reproducibility and ensures standardization in the feature construction, transformation, and selection. Also automated evaluation and refiners enhance robustness through systematic validation of features between multiple splits of data.

1.2.5. Integration with End-to-End ML Pipelines

The automated feature engineering has been important in the end-to-end machine learning pipeline, including AutoML systems. Automated feature engineering allows the faster development of application workflows and more confident production systems by integrating model selection, hyperparameter optimization, and deployment processes. This continuity is what guarantees that feature engineering scales and develops together with models and data, which is why it is a fundamental component of contemporary, scalable, and flexible machine learning systems.

1.3. Challenges in Feature Engineering for Tabular Data

The heterogeneous, structured, and often imperfect characteristics of the real-world datasets allow feature engineering of tabular data to introduce a variety of challenges. Among the main challenges, it is hard to deal with the varying types of features within the same data set, where there are numerical values, categorical and ordinal values, and temporal values. The attributes of each type of feature need varying methods of preprocessing and transformation and improper management may result in loss of information or biased modelling. Markedly difficult is the presence of high-cardinality categorical variables that means that typically a feature engineering method will result in a high dimensionality or sparsity which adversely impacts model performance. The second major difficulty is dealing with missing values and noisy data, the typical tabular data. Missing data can either be random or have systematic patterns, and the choice of an imputation strategy needs to consider carefully the distributions of features and the processes that generate their underlying data. On the same note, outliers and measurement errors are also capable of deviating the feature distributions and poor learning performance unless effectively handled. These problems become more and more complicated to detect and fix as the amount of data and the number of impredicative dimensions increase. There is also a significant issue of feature interaction discovery with tabular data. Significant relationships may be acquired due to non-linear relationships between two or more features that are not always obtainable in straightforward transformations or straightforward models. Such interactions are labor-intensive and strongly domain dependent, which takes too much time to be detected manually. With more features, the permutation space of the interactions increases in a combinatorial manner, and it becomes impossible to exhaust interactions manually. Scalability and responsiveness are other complications to feature engineering. Manual methods are not easily scaled to large datasets or dynamic and changing data environments, where feature updates need to happen quite often to capture new trends. Also, there is still a dilemma of balancing predictive work and interpretability. Although more sophisticated engineered characteristics and learned representations may increase accuracy, they also tend to decrease transparency, which is also essential in regulated or high-stakes environments. All of these adversities point to systematic, automated and adaptive feature engineering techniques that are specific and unique to tabular data.

ISSN: 3071-3315
PP: 01-15.

2. *LITERATURE SURVEY*

2.1. Early Rule-Based Feature Engineering Approaches

Rule-based, with predefined mathematical and statistical transformations proposed by domain experts, were the primary feature engineering methods used early on in automated feature engineering. Logarithmic and power transformations to deal with skewed distributions, the expansion of features via polynomials to deal with non-linear relationships, normalization and generalization of scale, and the abstraction of discrete operations using a process known as binning or quantizing to discrete values were common operations. Applicability of these techniques was also easy and interpretation was high-level because one could tell the mathematical sense of each transformation. Nevertheless, they were not very effective unless rules are selected manually, based on the expertise knowledge. Consequently, these systems were not very flexible to different datasets and could not automatically find the complicated or task-specific interaction of features, which restricted their scalability to real world usage.

2.2. Feature Construction Using Relational and Statistical Methods

With the increasing complexity and relationality of the data sets, the feature engineering approaches started to adapt feature statistical aggregation and relational learning approaches. These techniques were aimed at building features through joining information of various tables, and condensing data with group-by and aggregation functions like mean, sum, count and variance. Temporal windowing had also been proposed to model patterns of time by averaging historical data in a fixed or dynamic collection. This paradigm facilitated the creation of higher-order features that indicate relationships between things and it is therefore efficient in structured and relational datasets. Such techniques were more expressive than simple rule-based methods but tended to blow up due to combinatorial feature explosion and to have to be handled with孝heid sophisticated control measures to handle redundancy and computational cost.

2.3. Evolutionary and Search-Based Methods

To overcome this shortcoming of handwritten rules, evolutionary and search algorithms based feature engineering models posed the problem of feature constructing as an optimization problem. The genetic algorithms and genetic programming techniques were used to explore large and complex spaces of features by repeatedly evolving candidate feature modifications, via fitness measures, which are usually based on predictive model performance. Those techniques made it possible to automatically find novel and non-intuitive combinations of features, which greatly increased the expressiveness of models. Although they are flexible, evolutionary methods are computationally expensive and their use may demand large-scale evaluations that are not feasible with large datasets or those that time-can be sensitive. Also, because these algorithms are stochastic, they may cause problems of reproducibility.

ISSN: 3071-3315
PP: 01-15.

2.4. Deep Learning-Based Feature Representation

As deep learning was developed, feature engineering began to be based on automated representation learning. Neural architectures like autoencoders, embedding layers on discrete variables, attention models, among others, learn latent feature representations by example, avoiding the explicit construction of features as they are complex non-linear interactions. They are especially useful with high-dimensional and heterogeneous data whose typical feature engineering is tedious. Nevertheless, the representations provided by the deep learning are usually opaque and they can be scarcely interpreted as compared to explicitly engineered features. Such lack of transparency presents a challenge in areas that need to be explainable like healthcare, finance and regulatory settings.

2.5. AutoML Frameworks and Integrated Pipelines

In modern AutoML systems, the goal is to offer an entirely automated machine learning pipeline, using feature engineering, model selection, and hyperparameter optimization as a single workflow. Such systems do not require human intervention as much as the others; and also allow those without expertise to create competitive predictive models. Search strategies are usually used together with automated generation of features to find the best model configurations. Although convenient, AutoML systems have issues pertaining to transparency because the generated features and models may be challenging to understand. Besides, they may lead to high computational costs as well as exaggerated chance of overfitting especially on small or noisy datasets due to their dependence on large search processes.

3. METHODOLOGY

3.1. Overall Framework

The suggested automated feature engineering framework is developed as a modular and entirely automated pipeline that transforms raw data into quality features as inputs into machine learning models in a methodical way. The framework contains five key steps, namely, data profiling, feature generation, feature transformation, feature selection, and feature evaluation. Every step is independent and aims at reducing human intervention with the benefit of using data-driven insights and pre-established optimization goals to maximize model performance and robustness.

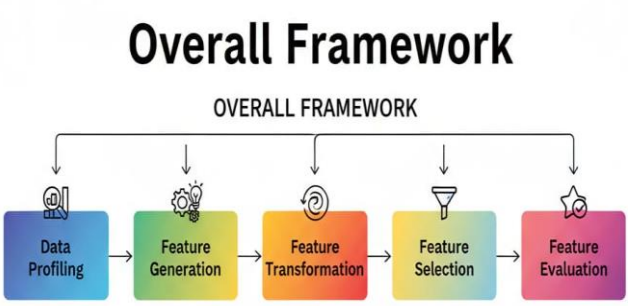

Fig 2 - Overall Framework

ISSN: 3071-3315
PP: 01-15.

3.1.1. Data Profiling

Data profiling phase is a process of performing a preliminary analysis of the input data in order to know its structural and statistical features. It involves the classification of data types (numerical, categorical, temporal), relationship-finding of the missing values, distribution analysis, and correlation of features. The statistical summaries provide useful information about the data used in processing the downstream by computing statistical summaries like the mean, variance, skewness, and kurtosis. The knowledge gained in profiling helps the framework to be able to adapt to pick the right feature generation and transformation strategies depending on the features of the dataset.

3.1.2. Feature Generation

The feature generation is concerned with the construction of new candidate features out of the original data by an automated construction. Such a step uses arithmetic combinations, statistical groupings and relational operations to express higher-order interactions and latent data patterns in the data. To synthesize the relational features of structured and multi-table data, aggregation functions and joins are used. The search strategies and the heuristics control the generation process so as to maintain a balance between feature diversity and computational efficiency.

3.1.3. Feature Transformation

During the feature transformation phase, the features that have been generated are trimmed down to enhance their appropriateness with respect to the learning methods. Examples of common transformations are normalization and standardization to make the scale homogenous, logarithmic and power transformation to minimize skewness, and methods of encoding categorical variables. Temporal changes can be done using window-based transformations in capturing sequential trends. Such transformations increase model convergence, numerical stability and overall predication.

3.1.4. Feature Selection

The features selection stage is based on the fact that most of the features generated might be large, and due to this fact, one might resort to the creation of the most relevant and informative subset of features. At this phase the automated methods of selection include filter methods, based on statistical significance, wrapper methods, based on model analysis, or embedded methods as part of the learning algorithms. Extra, meaningless, and strongly dependent features are dropped off to decrease dimensionality, over fitting as well as enhance interpretability.

3.1.5. Feature Evaluation

The last phase compares the quality of the chosen features by predefined goals of optimization. Performance metrics based on downstream machine learning models, that is, accuracy or precision, recall, or loss, or other functions, is used to evaluate the feature sets, depending on the task. Other evaluation requirements may be computational efficiency and feature stability. The analysis findings may be employed to optimize the previous steps by repeated revision so that an optimization process can be optimized through feedback.

ISSN: 3071-3315
PP: 01-15.

3.2. Data Profiling and Preprocessing

Data profiling and preprocessing are an essential step in an automated feature engineering framework proposed as it creates a wide knowledge of how the data appears and the quality of the data. We shall take a dataset in the form of a set of input-output pairs, each data object is a feature vector and a target outcome, and this dataset has a total of N entities. Each feature vector is d-dimensional, such that it is a d (d) set of singaporean qualities of an observation, and the target variable is the response to be anticipated in a regression or classification effort. In the process of data profiling, the framework checks each feature by default to identify what type of data it describes, i.e. numerical, categorical, ordinal, or temporal. Numerical features are computed as statistical aspects such as mean, variance, minimum and maximum values, skewness and kurtosis to describe their distributions. In the case of categorical features, the frequency distributions of the number of unique categories are studied. Moreover, missing value patterns are determined through the relative measurement of the percentage and distribution of the absent values in features, which helps the framework to distinguish between random and systematic missingness. Correlation analysis is also conducted in order to measure the dependence between features and that between features and the target variable. Pearson or Spearman correlation coefficient measures are applied in numerical attributes and appropriate measures in association of categorized variables. These analyses are useful in identifying multicollinearity, redundancy and possibly informative relationships, which can be used in the future to inform feature generation and selection steps. Based on the profiling results, then preprocessing operations are applied. The missing values can be addressed using the imputation policies like the mean, median, and model-based imputation based on the characteristics of the features. Numerical characteristics are normalized or scaled to make them have identical ranges and categorical variables are transformed into numerical values that can be used with machine learning models. Outlier identification and treatment can also be conducted in order to minimize noise and enhance the robustness of the models. Through structured profiling and preprocessing of the data, the framework will make sure downstream automated feature engineering steps uses such processed and well-formed, informative inputs, which results into improved overall learning performance.

3.3. Automated Feature Generation

Automated feature generation to produce features aimed at enhancing the original feature space, it attempts to generate new candidate features systematically out of existing features via a set of predefined operators. Denote the collection of the available feature construction operators as O. The operators of this set represent a definite transformation or a combination rule, which may be applied to any of the existing features. In this process a new feature is created by using an operator on a subset of the input features, which creates an expanded representation that has the possibility to represent hitherto undetected features in the data. Simply put a newly created feature is the result of some operation on two already existing features, e.g. combining features by addition or multiplication, or other forms of functional transformations. Arithmetic combinations which are common to the operator set are addition, subtraction, multiplication, and division and are useful in realizing both linear and interaction effects between numerical features. Also statistical aggregation

ISSN: 3071-3315
PP: 01-15.

operators can be used, in particular, in case of grouped, temporal or relational data. These operators calculate summary statistics including mean, sum, count, minimum, maximum and variance and compute on subsets of data which allows the model to learn global or contextual patterns. In the case of categorical variables, one-hot encoding, frequency encoding, or target-based encoding perform the task of encoding the symbolic values into the numbers that can be presented to the learning algorithms. This process is automated and controlled by data profiling results that streamline feature generation process to institute only operators whose application is semantically significant. As examples, arithmetic operations are limited to the numerical properties, and the functions of encoding are used on the categorical attributes. Maximum feature depth constraints, rules of operator compatibility, and relevance thresholds are used to ensure that feature growth cannot go too far. This will make the feature set generated manageable and informative. With the help of automated feature generation, which is done through the systematic application of a rich repertoire of operators, higher-order interactions and multifaceted patterns can be identified in the original data that the unaided human eye would otherwise not have detected. This higher feature space feeds more information downstream models, giving them a higher expressive capacity and higher predictive capability and requiring fewer human-engineered features.

3.4. Feature Transformation and Encoding

The encoding and transformation of features is important in preparing generated features to enable them to be learned effectively by machine learning models. This phase is aimed at representation that enhances numerical stability, learning and predicting accuracy of raw and newly built features. In the case of numerical features, the transformations used include scaling and normalization among others, which are intended to make the various attributes work in a similar range. Reducing the influence of the features when they are large in value and promoting faster convergence of the gradient-based learning algorithms are among the methods such as min-max normalization and standardization. Also, the use of the logarithmic or power transformation can help to reduce the effect of skewed distributions and the use of the extreme values. Categorical features must use specialized encoding procedures to encode symbolic values using numerical values. Target encoding substitutes each category with a statistic based on the target variable, e.g. the mean target value of that category, which thus reflects category target relationships concisely. Regularization and encoding methods based on cross-validation are used to prevent leakage of information, as well as overfitting. When categorical features with high cardinalities, embedding-based representations are employed, in which the categories are translated into dense and low-dimensional vectors that are trained with the model. These embeddings are similarity between categories in semantics and they are found to be effective especially in deep learning models. The cyclical and sequential characteristics of time are converted to temporal features. Ordinary set of time characteristics (day of the week, month, hour, etc.) are broken down into cyclical elements through the use of both sine and cosine transformations to avoid loss of periodicity. This avoids any artificial discontinuities, e.g., between the end and the beginning of a cycle, and it allows models to learn smooth according to time. Other temporal changes can also involve lag features and rolling statistics to understand the historical trends and seasonality. The feature transformation and encoding in general make sure that

ISSN: 3071-3315
PP: 01-15.

various types of data are represented in the form that is compatible with assumptions of the learning algorithms. Through carefully designed fundamental transformations, the framework improves the robustness of the model, limits biasness posed by the raw data scales or encodings and co-optimally learns heterogeneous types of features in an automated sense.

3.5. Feature Selection Strategy

The feature selection phase seeks to give an optimal number of features that will give a fair balance between predictive performance and model simplicity. This is formulated as an optimization problem whereby the task is to then choose a subset of features among the total feature set that optimizes the overall predictive performance of the model as well as simultaneously tries to reduce the number of few features selected. Conceptually, the framework aims at finding the optimal trade-off between model effectiveness and complexity by rewarding those feature subsets that increase the quality of prediction and rewarding those that introduce unwanted dimensionality. In this expression, the predictive performance function is the quality of a part of features as indicated by a preferred effectiveness quantifier, such as classification accuracy, F1-score, or regression error, along with the learning task. This performance is evaluated with the training and validation of a down stream machine learning model with the selected features only. The second part of the objective includes a regularization term that differs between the size of the size of the chosen feature subset. This penalty is moderated by a weighting parameter that implies that overly large sets of features are discouraged although they can give small improvements in performance. In an attempt to solve this optimization problem, the framework uses automated feature selection techniques, such as filter, wrapper, and embedded methods. Filter methods made judgments based on features individually, based upon statistical criteria, whereas wrapper methods made judgments based on subsets of features based on model-driven performance. Embedded approaches form part of model training by incorporating feature selection directly as a result of regularization or tree-based importance identifiers. The framework dynamically chooses the most suitable strategy in accordance with the size of the dataset and computational ability. The penalty on the number of features proposed in the strategy of feature selection encourages feature representation that is compact and informative. This does not only minimize the possibility of overfitting but also increases efficiency and interpretability. Finally, the selection process that is facilitated by optimization allows ensuring that the resulting set of features is not only predictive but also sparse and thus, the overall robustness and generalizability of the learning model are increased.

3.6. Evaluation and Iterative Refinement

The automated feature engineering framework is feedbacked by the evaluation and refinement phase, which provides improvement of the produced set of features in a continuous way. At this step, the chosen features are tested on the basis of strong validation methods, most frequently, cross-validation, to get stable estimates of the generalization performance. This is because, by splitting the dataset into several training and validation folds the framework decreases the risk of having biased performance estimates depending on a subset of data. The metrics of evaluation, which could be the accuracy, the precision, the recall, the F1-score or mean squared error, are selected

ISSN: 3071-3315
PP: 01-15.

based on the nature of the learning problem and are used as quantitative measures of the ability of the features to perform successfully. The results of the evaluation help to control the process of its iterative refinement during which the poorly working features or the features that are redundant are automatically discovered and eliminated. The scores of importance of features based on model coefficients, tree-based or permutation-based scores of importance feature help to measure the amount of contribution of each predictor to predictive performance. Features with low importance that recurrently show to be detrimental to validation statistics are eliminated out of the feature set. This pruning leads to model control and avoidance of overfitting, especially when operative in high-dimensional feature spaces, produced by automated processes.

4. RESULTS AND DISCUSSION

4.1. Experimental Setup

Experimental evaluation was meant to determine the effectiveness and external validity of the suggested automated feature engineering framework in a variety of real-life settings. Benchmark tabular datasets based on various domains of application, such as finance, healthcare, and marketing were used in experiments, with each having its unique data characteristics and modeling issues. Financial data are generally numeric and time series with a high level of correlation, noise, and correlation, healthcare data are generally missing, class imbalanced, and heterotyped, and marketing data are generally of high cardinality as categorical variables and behavioral predictors. This variety makes sure that there is an overall assessment of the framework in different data circumstances. A uniform protocol of experiments was used in each dataset. The data was separated into training and testing sets and cross-validation was done with the training data to optimize the model parameters and to check the feature quality. To prove that the automated feature engineering pipeline can be adapted to all types of data and can be automated without manual intervention and specific to the domain, the proposed feature engineering pipeline was used on all themed datasets without any human intervention or customization. Raw features were also used in baseline experiments whereby a comparative evaluation of the performance improvement made under the influence of automated feature engineering was taken. Various popular machine learning models were selected to evaluate in order to represent the various learning paradigms. The use of the decision tree models was because they are interpretable with the capability to capture non-linear relationships. The decision to use gradient boosting models was based on the fact that it is a powerful predictor and it works well when dealing with a structured data. Linear classifier was tested to determine the extent with which the features that were engineered improve the expressiveness of simpler models. Fair and reproducible comparisons were possible with default or generally recommended hyperparameter settings. Task-appropriate measures of performance were used to measure performance, and the mean performance of multiple runs was taken to minimize variability. This practical design gives a high-quality and objective test of the potential of the suggested framework to enhance the performance of the model across the domains and learning algorithms, which indicates its high degree of practicality in the actual world of tabular data modeling processes.

ISSN: 3071-3315
PP: 01-15.

4.2. Performance Comparison

Table 1: Performance Comparison

Dataset	Baseline Accuracy (%)	AFE Accuracy (%)	Improvement (%)
Finance	78.4	85.2	6.8
Healthcare	81.1	87.6	6.5
Marketing	75.9	83.3	7.4

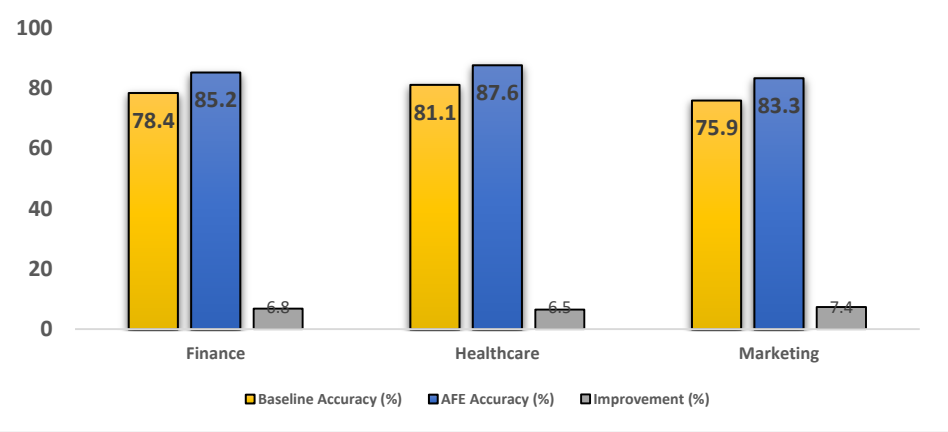

Fig 3 - Performance Comparison

4.2.1. Finance Dataset

On the finance data, the implemented automated feature engineering framework could improve the model performance by 6.8 percent relative to the performance on models that were trained on unprocessed features. This benefit can be owed to the framework having an ability to produce interaction and aggregation characteristics that encapsulate non-linear relationship and non-temporal patterns of financial relationship. Transformation on features and selecting the most important factors also minimized noises and multicollinearity, and consequently, all the models tested provided more consistent and precise predictions.

4.2.2. Healthcare Dataset

With the healthcare dataset, the improvement was at 6.5%. There tend to be lost values, heterogeneous types of features as well as subtle interactions among the clinical variables in healthcare data. The challenges were managed by automated preprocessing stage and generation of informative derived features which correctly managed the issues of data inconsistency. Consequently, the model enjoyed a better generalization and robustness especially in situations where there were fewer or noisy data.

4.2.3. Marketing Dataset

The greatest increase in performance of 7.4 percent was realized on marketing dataset. This data is usually made up of categorical features of high cardinality and behavioral trends that cannot be easily modeled in terms of raw data alone. The encoding mechanisms, feature building methods of the framework were able to encode customer behavior and the impacts on customer interaction

ISSN: 3071-3315
PP: 01-15.

thereby highly improving the predictive capabilities. These findings reveal the power of the framework to deal with the complex categorical information used in marketing applications. On the whole, this comparison of performances reveals that the suggested automated feature engineering framework is effective and flexible in its functioning as it can always enhance the model performance in different domains.

4.3. Computational Overhead Analysis

Although automated feature engineering (AFE) can be easily used to improve predictive performance, it also implies that the model formation and training processes will require extra computation needs. This overhead is mainly due to the fact that a high set of candidate features are generated, various transformation and encoding methods are used and feature subsets are tested by the use of iterative selection and validation. Aggregation, encoding, and cross-validation-based assessment operations necessitate multiple rounds of model training and performance prediction that incur more computation times and consume more resources than pipelines that only rely on handcrafted or unengineered features. This overhead is determined by a number of parameters such as the size of data set, dimensionality of features and complexity of operator set to be used in features generation. Computational needs may also be increased by high-cardinality categorical values, relational joins and time aggregations. Moreover, wrapper-based and iterative feature selection techniques, albeit efficient in the detection of high-quality sets of features, add to the cost of training models because they require repeated model assessment. Consequently, the AFE process can be computation intensive especially in resource limited or large scale settings. But the added cost is mostly limited to the offline training phase and the cost is usually incurred once or once in a while. With a deployment case, the advantages of AFE are even greater since the chosen final feature set and trained model are fixed and optimized. Computation of the features during inference can be computationally lightweight and efficient in practice when redundant or low impact features have been eliminated. The fixed training cost can therefore be spread in the model use over time making the strategy viable in the real life applications where the models will be used over long time. In addition, the better predictive power and resilience obtained with the use of AFE can yield down-stream cost reductions in terms of error reduction, quality decisions and less frequent re-training of models. Thus, although automatic feature engineering will incur more computational costs incurred during training, the overall cost-benefit tradeoff would be in favor of automated feature engineering, especially in production scenarios where long-term performance benefit would be more than upfront computation cost.

4.4. Interpretability Considerations

Interpretability is a key factor in automated feature engineering, especially in the area of application that is required to be transparent, trustful and compliant by regulation. The rule-based and statistical features are more explainable generally as they are obtained to be explicit mathematical transformations or better known statistical operations. Values like ratios, differences, aggregate counts, and normalized values possess simple semantic meanings, which make it easy to trace the impact of input variables on model predictions on a practitioner. This openness aids in

ISSN: 3071-3315
PP: 01-15.

model debugging, validation and sharing of results with domain experts and stake holders. The feature representation of deep learning systems, including embeddings and latent vectors generated by neural networks, on the contrary, are usually black-box representations. Even though these representations can take non-linear interactions into account and vastly enhance the forecasting capabilities, they cannot be directly interpreted. Their unspecified feature semantics may restrict their use in sensitive areas like healthcare, finance and legal systems where explainability is frequently essential. The accurate contribution of deep features is still not clear even with the post hoc explanation methods. Strategies that provide a balance between performance and transparency Hybrid methods that integrate interpretable feature engineering methods with strong representation learning should be promising. In these methods, rule based, statistical and relational characteristics are initially constructed and filtered so as to maintain interpretability and the deep representations are used selectively to model intricate patterns not found with conventional methods. This tiered approach enables models to have an advantage of human-understandable properties and expressive latent representations. Moreover, hybrid frameworks should be able to take advantage of the tools of feature importance analysis and explainability to offer insights on several levels of abstraction. Hybrid automated feature engineering methods can achieve this by having a greater degree of interpretability and predictive power that allows them to be more widely applicable to different domains with different degrees of transparency needs. They are useful in the production of intelligent choices since they allow a certain level of interpretability without compromising on performance thus increasing user confidence and making machine learning systems useable in the real world.

5. CONCLUSION

This paper paper outlined a detailed, methodological research on automated feature engineering (AFE) to tabular data with recognition of increasing significance in recent machine learning pipelines. It has been known that feature engineering can play a vital role in determining the performance of a model, but the conventional manual methods are time-consuming, domain-specific and hardly scalable. AFE helps to counteract these limitations and allows the creation of powerful and well-performing models with a minimum amount of human intervention by automating the feature generation, transformation, selection, and evaluation processes. The presented literature review followed the development of the methods of feature engineering, where simple rule-based methods were used in the past, sophisticated search-based methods advanced, and deep learning-based approaches integrated with AutoMLs, which created an excellent framework background. The systematic approach by which the paper was presented illustrated how a flexible automated feature engineering system could be used to benefit various tabular data. Every step of the framework data profiling, automated generation of features, feature transformation and encoding, feature selection and iteration evaluation were developed to function in a data-driven and optimization-based fashion. This structural design makes the features generated not only expressive, but also relevant, compact and downstream learning algorithms friendly. The effectiveness of the suggested approach was confirmed through the use of experimental assessments using finance, healthcare, and marketing data sets that demonstrated a consistent increase in predictive performance across the various model

ISSN: 3071-3315
PP: 01-15.

kinds. These findings prove that automated feature engineering can improve simple, as well as complex models, significantly with better input representations. Although it has these benefits, the research also noted some major difficulties that are related to automated feature engineering. Higher cost of computation in training and lower interpretability especially where deep representations are used are still a major concern. Nevertheless, the experimental discussion proved that the extra computational cost is mostly limited to the offline training stage and that can be offset at the inference time. Moreover, an effective tradeoff between transparency and performance can be achieved by hybrid schemes that incorporate explainable rule-based schemes with effective representation learning techniques. In the future, one of the paths to take in research is to develop explainable AFE models giving a better understanding of feature construction and contribution, or adaptive feature learning models, which can change dynamically with different data distributions. The inclusion of automated feature engineering with live and streaming data systems also constitutes a potential direction of expanding its use. In sum, automated feature engineering will become a core part of developing scalable, reproducible, and high-performing machine learning systems of all sizes in all data-heavy settings.

REFERENCES

[1] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3, 1157–1182.
[2] Liu, H., & Motoda, H. (1998). Feature selection for knowledge discovery and data mining. Springer.
[3] Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer.
[4] Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Data preprocessing for supervised learning. International Journal of Computer Science, 1(2), 111–117.
[5] Jensen, D., & Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. ICML.
[6] Deepayan, C., & Getoor, L. (2006). Link-based classification. SIGKDD Explorations, 8(2), 14–23.
[7] Kanter, J. M., & Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. IEEE DSAA.
[8] Markovitch, S., & Rosenstein, D. (2002). Feature generation using general constructor functions. Machine Learning, 49(1), 59–98.
[9] Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection. MIT Press.
[10] Neshatian, K., Zhang, M., & Andreae, P. (2012). A survey on evolutionary computation approaches to feature selection and construction. IEEE Transactions on Evolutionary Computation, 16(6), 817–840.
[11] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798–1828.
[12] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504–507.
[13] Guo, C., Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737.
[14] He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622.
[15] Feurer, M., et al. (2015). Efficient and robust automated machine learning. NeurIPS, 2962–2970.