FEATURE SHAPLEY: A GENERAL FRAMEWORK TO DISCOVERING USEFUL FEATURE INTERACTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

A machine learning system is typically composed of model and data. In many applications, feature is the input of models so as to generate a meaningful prediction. While a large amount of model-centric solutions are proposed to improve the capabilities of models, there is very limited exploration on how to discover useful feature interactions[1] from a data-centric perspective. In this work, we propose a general framework named Feature Shapley with the purpose of discovering useful high-order feature interactions based on Feature Shapely values and thereby generating new features. Since computing exact Feature Shapley values is computationally infeasible, Monte-Carlo approximation and early truncation trick are applied for efficient estimation of Feature Shapley values in this work. Experimental results indicate that the decisive feature interactions exploited by Feature Shapley are of vital importance for the Click-through rate (CTR) prediction and asset pricing task. With decisive feature interactions exploited by Feature Shapley, even simple models (e.g., linear regression (LR) or shallow neural network) could achieve similar or even better performance comparing with more complex approaches and keep their superior interpretability at the same time.

1 INTRODUCTION

Progress in machine learning has been driven by efforts to improve performance on benchmark datasets. To improve the performance of machine learning systems, model-centric solutions focus on improving the capabilities of models iteratively while less effort is observed from the data-centric perspective. As data is the fuel of modern machine learning systems, there is another natural and promising way to boost the performance of machine learning systems, commonly known as data-centric solutions. In other words, one can deliver a performance-enhanced machine learning system by improving the input data Northcutt et al. (2021).

In this work, we propose a general framework named Feature Shapley to discovering useful feature interactions from the data-centric perspective. The learned feature interactions are then selected as the input. Specifically, Feature Shapley values are estimated for input features when creating a new feature interaction term. In this way, only features that have considerable marginal contribution to the model performance will be included to the feature interaction. Note that the proposed Feature Shapley is very flexible as it can be applied on any task that takes feature as the input of machine learning systems. We demonstrate the effectiveness of Feature Shapley on both CTR task and asset pricing task in this work.

However, computing exact Feature Shapley values is computationally infeasible. We therefore find two keys for efficient estimation of Feature Shapley values: (1) Extending Monte-Carlo approximation methods Maleki et al. (2013) to our feature valuation setting. (2) Applying early truncation trick to avoid futile model training. This enables Feature Shapley to discover useful interactions with affordable computational cost. Without bells and whistles, by simply considering feature interactions learned by our Feature Shapley, a linear regression model is able to achieve a solid improvement. As shown in Fig. 1, by taking only three feature interactions exploited by Feature Shapley into account, LR (Shapley) (i.e., a linear regression model that linearly sums all features and learned feature interactions) is able to achieve competitive performance compared with some well-designed baselines.

[1]Feature interactions are a major type of feature transformations, where multiplication is performed over input features.

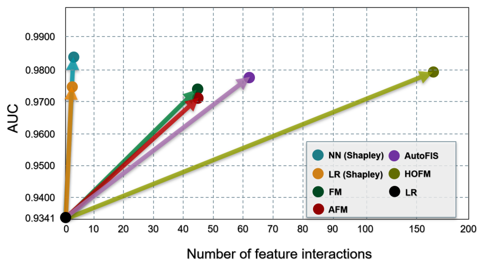


Figure 1: The additional gain between the linear regression (LR) and comparing methods for CTR prediction on the Frappe Dataset. By considering only three feature interactions, our methods (i.e., LR (Shapley) and NN (Shapley)) significantly boost the AUC performance. Best viewed in color.

Meanwhile, by employing a single layer perceptron over the concatenated representations of feature interactions, NN (Shapley) is able to outperform all comparing methods on the Frappe dataset.

In a nutshell, this work makes the following main contributions: **Conceptual:** We emphasize the importance of feature interactions from the data-centric perspective. By modeling such feature interactions, one could achieve better performance and keep the superior interpretability of linear models. **Algorithmic:** We propose a novel framework Feature Shapley, which estimates Feature Shapley values of features and further generates useful feature interactions without manual effort. To decrease computational complexity, Monte-Carlo approximation methods and early truncation trick are used to approximate Feature Shapley values. **Empirical:** We conduct extensive experiments on three CTR datasets and US Stock Markets, to demonstrate the advantages of our Feature Shapley on the effectiveness of discovering useful feature interactions.

2 Related works

This section first briefly reviews existing data-centric approaches. As our proposed Feature Shapley aims to measure the marginal contribution of input features when creating feature combinations, we then present existing feature-importance methods.

2.1 Data-centric Methods

Data-centric approaches emphasize the role of good data as the cause of good model performance, rather than the design of the model. For example, the problem of how to identify and learn with noisy labels have received increasing attention from researchers Northcutt et al. (2021); Feng et al. (2020); Cordeiro & Carneiro (2020). Meanwhile, feature selection techniques and feature extraction techniques have demonstrate their effectiveness to decrease the high dimensionality of the input feature vector Zebari et al. (2020). Another typical successful case is data augmentation which aims to increase the amount of training data for deep neural networks Zoph et al. (2020); Zhong et al. (2020). In this work, we focus on exploiting useful feature interactions based on Shapley value.

2.2 Shapley value for interpretability

Recent studies demonstrate the effectiveness and interpretability of Shapley value for various applications Lundberg & Lee (2017); Shapley (2016). For example, Ghorbani et al. propose to quantify the Shapley value of each data sample to the model Ghorbani & Zou (2019), and then Neuron Shapley Ghorbani & Zou (2020) is proposed to identify responsible filters of convolutional neural networks for the facial recognition tasks and adversarial attacks. Our work lays on the foundation of feature-importance methods, which aim to estimate the marginal contribution of each feature to model performance. Research has focused on efficient approximation of feature importance estimation and developing model-specific methods Lundberg & Lee (2017); Strumbelj & Kononenko (2010); Mase et al. (2019); Chen et al. (2018); Lundberg et al. (2020). For example, SHAP Lundberg & Lee (2017) provides model-specific methods to explain how an individual feature affects the model prediction, leading to interpretable machine learning systems. Different from prior works which focus on explaining how much a feature, a neuron, or a data sample affects the model per-

Algorithm 1: The Feature Shapley Algorithm

```
Input: Initial model $f^0$; evaluation metric $P$; feature set $X$; and number of random permutations $k$.
Output: The optimal feature set $X^*$.
Initialize iteration $t = 0$; $X^* = X$;
while Convergence criteria not met do
    $p^t \leftarrow P(f^t(X^*))$ ;
    $t \leftarrow t + 1$ ;
    for each target feature $x_i \in X$ do
        Initialize Feature Shapley value of candidate features $\phi_z = 0, z \in \{i+1, \cdots, n\}$ ;
        Obtain k random permutations over candidate feature set $\{x_{i+1}, \cdots, x_n\}$;
        for each permutation $S_k$ do
            for $j$th feature in $S_k$ do
                Obtain a temporal feature interaction term $x_c = x_i \times x_1^{S_k} \times \cdots \times x_j^{S_k}$;
                Train $f_j^t(X^* \cup \{x_c\})$ ;
                Evaluate $p_j^t \leftarrow P(f_j^t(X^* \cup \{x_c\}))$;
                $\phi_j^{S_k} = p_j^t - p^t$;
                $p^t \leftarrow p_j^t$;
            end
        end
        Calculate Feature Shapley values $\phi_z, z \in \{i+1, \cdots, n\}$ by averaging Feature Shapley values of
          candidata features obtained from different permutations $S_k$;
        Select features whose Feature Shapley value is over the tolerance and obtain a feature set $S_m$ ;
        Form a new feature interaction term $x_m = x_i \times x_1^{S_m} \times \cdots \times x_{|S_m|}^{S_m}$ ;
        Update model $f^t(X^*) \leftarrow f^t(X^* \cup \{x_m\})$;
        Update feature set $X^* \leftarrow X^* \cup \{x_m\}$
    end
end
```

formance, we focus on exploiting the interaction effect among input features based on the Feature Shapley value. By taking the learned feature interactions as new "features", even simple models (e.g., linear regression or shallow neural network) could achieve superior experimental results.

3 Feature Shapley

3.1 Shapley Values

As a concept from cooperative game theory, Shapley values can be feature importances for machine learning models, to explain how much an individual feature contributes to the prediction Lundberg & Lee (2017); Štrumbelj & Kononenko (2014); Lipovetsky & Conklin (2001); Shapley (2016). A typical paradigm to estimate Shapley values is to retrain the model on all feature subsets $S \subseteq X$, where X is the set of all features. Specifically, the Shapley value of j-th feature can be estimated:

$$\phi_j = \sum_{S \subseteq X \backslash \{j\}} \frac{|S|!(|X| - |S| - 1)!}{|X|!} [P(S \cup \{j\}) - P(S)] \tag{1}$$

where $P(S)$ denotes the performance of the model which takes feature subset S as input, and $P(S \cup \{j\})$ represents the performance of the model which takes j-th feature into account. Note that the differences are computed for all possible subsets $S \subseteq X \backslash \{j\}$, because the influence of the j-th feature depends on other features. In our context, we will refer to ϕ_j as Feature Shapley value.

3.2 Estimating Feature Shapley Value

Given a basic linear model f^0, an evaluation metric P, and a feature set $X = \{x_1, \cdots, x_n\}$, our goal is to discover useful feature interactions by estimating the Feature Shapley value of feature candidates. As shown in Algorithm 1, for each feature in the feature set X, one target feature x_i is set at a time, and features $x_z, z \in \{i+1, \cdots, n\}$ are treated as candidate features. Candidate features with high Feature Shapley value will "join" the target feature x_i, a new feature combination can be constructed by multiplying these features (see line 19 in the Algorithm 1).

As Eq. (1) depicts, computing Shapley value is computationally expensive with respect to the number of features. Inspired by recent works which efficiently estimate Shapley values for data or neurons Ghorbani & Zou (2019; 2020), we alleviate the above problem by applying Monte-Carlo approximation methods and early truncation trick to approximate Feature Shapley values. Specifically, for a model with feature set X, the Feature Shapley value of the j-th feature could be estimated:

$$\phi_j = \mathbb{E}_{\pi \sim \mathcal{N}}[P(X^* \cup \{x_i \times x^{S_k} \times x_j\}) - P(X^* \cup \{x_i \times x^{S_k}\})] \tag{2}$$

where $\mathcal{N}$ is a uniform distribution over $n!$ permutations of the features; X^* is the previous feature set; x_i is the target feature; and x^{S_k} is the product of features that appear before the j-th feature in a permutation S_k (i.e., $x_j^{S_k}, j \in \{1, \cdots, j-1\}$). In this way, the Feature Shapley value of candidate features can be unbiasedly approximated, whose error analysis has been well studied Maleki et al. (2013). The main computational expense in the Eq. (2) is computing the marginal contribution of each candidate feature for a feature combination: $P(X^* \cup \{x_i \times x^{S_k} \times x_j\}) - P(X^* \cup \{x_i \times x^{S_k}\})$. In order to alleviate the problem, we apply early truncation in this work. Specifically, the number of learned feature interactions in the early stage is small, and the performance of model usually increases steadily as the number of learned feature interactions increases. However, the contribution $P(X^* \cup \{x_i \times x^{S_k} \times x_j\}) - P(X^* \cup \{x_i \times x^{S_k}\})$ degrades to zero or negligible when a near optimal set of feature interactions is learned by the Feature Shapley, which means that it is hard to find useful feature interactions to improve the performance continuously. Based on the above observation, we define a performance threshold to guide the learning process of Feature Shapley, i.e., the learning process will be terminated if no useful feature interaction is observed in an iteration.

3.3 Time Complexity

Here we analysis the time complexity of our proposed algorithm. There are three loops in the algorithm from line 5 to line 21. The outer loop would iterate all features, thus contributing $O(|X|)$. The middle loop would iterate all k permutations, thus contributing $O(k)$. The inner loop would iterate all features in S_k with a fixed x_i, thus contributing $O(|S_k|) = O(n-i) = O(n)$. In line 11, we train f_j^t based on the linear model as shown in Eq. (1), thus contributing $O(NTW)$, where N is the number of the training examples, T is the epoch, and W is the the number of weights. The operations in line 17-21 are $O(1)$ operations. Overall, the time complexity of the proposed algorithm is $O(|X|knNTW)$.

4 Application to CTR Prediction

In this section, we apply Feature Shapley to CTR prediction task, which is a representative task in learning feature interactions.

4.1 Task Description

CTR prediction plays an important role in recommender systems, which aims to estimate the probability of a user to click on a given item. The main challenge is to effectively capture useful feature interactions Lian et al. (2018). However, finding meaningful feature interactions comes at a high cost, since it requires tedious engineering efforts and heavily relies on domain experts Song et al. (2019); He & Chua (2017). In response, Factorization Machines (FM) Rendle (2010) is proposed to model second-order feature interactions without manual engineering by parameterizing the weight of a feature interaction as the inner product of the representations of the raw features. Given a real valued feature vector $\mathbf{x} \in \mathcal{R}^n$, where n is the number of features, FM is formulated as follows:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{j=i+1}^{n} \mathbf{v}_i^T \mathbf{v}_j \cdot x_i x_j \tag{3}$$

where w_0 is the global bias; w_i is learned to weight the i-th feature; and $\mathbf{v}_i^T \mathbf{v}_j$ term denotes the interaction effect between the i-th and j-th feature, which is estimated by the inner product between their representations.

4.2 Background

Despite its effectiveness to model feature interactions and generalization to predict unseen feature interactions, FM suffers two major drawbacks. First, FM only models second-order feature interactions and regrettably ignores high-order feature interactions. Second, FM enumerates all pairwise interactions explicitly, ignoring the fact that not all feature interactions are useful to the finial prediction Xiao et al. (2017). To alleviate the former concern, approaches with the purpose of modeling high-order interactions are proposed in recent years. For example, HOFM Blondel et al. (2016) proposes to approximate all possible feature interactions through the ANOVA kernel. NFM He & Chua (2017), PNN Qu et al. (2016), and Deep Crossing Shan et al. (2016) leverage deep neural networks (DNNs) to capture patterns of high-order feature interactions. Furthermore, ensemble methods such as Wide&Deep Cheng et al. (2016), DCN Wang et al. (2017), DeepFM Guo et al. (2017), and xDeepFM Lian et al. (2018) propose to combine a shallow component (e.g. LR and FM) and a deep component (i.e., DNNs) to model low-order interactions and high-order interactions simultaneously.Another group of works focus on looking for useful features interactions. For instance, AFM Xiao et al. (2017) proposes to weight all second-order feature interactions via an attention network. Autoint Song et al. (2019) and AFN Cheng et al. (2020) learn the weight or orders of each feature in feature interactions via self-attention or logarithmic neurons, which take into account of all features when generating a feature combination. Empowered by advanced designs in Neural Architecture Search (NAS), Autocross Luo et al. (2019), AutoFIS Liu et al. (2020b), AutoGroup Liu et al. (2020a) are proposed to generate high-order feature interactions. However, the order of such feature interactions need to be defined in advance due to the large search space, restricting models' potential in looking for discriminative feature interactions.

This motivate us to apply our proposed Feature Shapley approach to discover useful feature interactions for the CTR task. It is advantageous in threefold. First, it only considers "useful" features when creating a feature combination, which reduces noisy information for both accuracy and interpretability. Second, feature interactions of arbitrary order can be learned efficiently. Compared with methods which numerate all possible feature interactions (their search space is $n!$), the search space of Feature Shapley is reduced to kn^2. Third, different from deep learning models which implicitly model feature interactions via multilayer perceptron. In this way, high-order feature interactions can be generated in an explicit manner. This leads to superior interpretability while achieving the improvement of accuracy. Consequently, we apply Feature Shapley on CTR datasets to show its effectiveness, which is denoted as LR (Shapley). To further improve the performance for CTR prediction, we feed the concatenated representations of learned feature interactions into a single layer perceptron, which is then added to the output of LR model. This approach is presented as NN (Shapley) in this paper.

4.3 Comparing Methods

(1) LR: It is the linear combination of raw features. No interaction effect is modeled in this method. **(2) FM** Rendle (2010): FM enumerates second-order feature interactions. **(3) AFM** Xiao et al. (2017): It distinguishes second-order feature interactions via attention networks. **(4) HOFM** Blondel et al. (2016): It approximates high-order feature interactions through the ANOVA kernel. **(5) PNN** Qu et al. (2016): PNN proposes a product layer upon the embedding layer, to model pairwise interactive patterns. **(6) AFN** Cheng et al. (2020): AFN learns adaptive-order (power) feature interactions via a logarithmic transformation layer. **(7) DCN** Wang et al. (2017): It jointly learns explicit and implicit high-order feature interactions. **(8) DeepFM** Guo et al. (2017): It simultaneously models low-order feature interactions via FM and high-order feature interactions via DNN. **(9) xDeepFM** Lian et al. (2018): It proposes compressed interaction network to replace the cross network in the DCN. **(10) AutoFIS** Lian et al. (2018): AutoFIS automatically identifies useful second-order and third-order feature interactions. The learned feature interactions are then fed into the DeepFM model for prediction.

4.4 Experimental Settings

We conduct experiments based on three public datasets: Movielens, Frappe and Avazu[2], whose details are summarized in Table 1. Their instances are randomly split by 7:2:1 for training, validation

[2]https://github.com/WeiyuCheng/AFN/-AAAI/-20

Table 1: Data statistics.

Dataset	Number of Fields	Number of Features	Number of Instances
MovieLens	3	90,445	2,006,859
Frappe	10	5,382	288,609
Avazu	22	1,544,250	40,428,967

Table 2: The performance Comparison on all datasets. ”#FI” is short for number of explicit feature interactions. We use $'*'$ to denote statistically significant improvements (paired t-test with p-value < 0.05). Intuitively, better performance with fewer feature interactions represent a better model, we hence highlight the best performance and the least feature interactions in this table. For each empirical result, we run the experiments for 5 times and report their average value.

Model Class	Model	Movielens			Frappe			Avazu		
		AUC	Logloss	#FI	AUC	Logloss	#FI	AUC	Logloss	#FI
First-Order	LR	0.9328	0.2972	N.A.	0.9341	0.2916	N.A.	0.7350	0.3806	N.A.
Second-Order	FM	0.9392	0.2780	3	0.9726	0.1925	45	0.7445	0.3758	231
	AFM	0.9359	0.2893	3	0.9707	0.2068	45	0.7420	0.3775	231
High-Order	HOFM	0.9442	0.2637	4	0.9790	0.1584	165	0.7497	0.3745	1771
	PNN	0.9409	0.2833	N.A.	0.9749	0.1825	N.A.	0.7498	0.3741	N.A.
	AFN	0.9455	0.2668	N.A.	0.9735	0.1874	N.A.	0.7491	0.3739	N.A.
Ensembled Methods	DCN	0.9421	0.2771	N.A.	0.9677	0.1825	N.A.	**0.7514***	0.3743	N.A.
	DeepFM	0.9445	0.2736	N.A.	0.9733	0.1868	N.A.	0.7492	0.3735	N.A.
	xDeepFM	0.9504	0.2588	N.A.	0.9766	0.1759	N.A.	0.7506	**0.3729**	N.A.
NAS Method	AutoFIS	0.9560	0.2365	4	0.9769	0.1718	62	0.7501	0.3730	1669
Our Methods	LR (Shapley)	0.9418	0.2747	**2**	0.9746	0.1780	**3**	0.7469	0.3746	**32**
	NN (Shapley)	**0.9693***	**0.1714***	**2**	**0.9833***	**0.1122***	**3**	0.7471	0.3746	**32**

and test, respectively. **(1) Movielens:** It includes user’s preference base on the following feature fields: user ID, movie ID and tag. **(2) Frappe:** This dataset contains a context-aware app usage log, which includes 10 feature fields: user ID, item ID, daytime, weekday, isweekend, homework, cost, weather, country and city. **(3) Avazu:** It contains users’ tagging records on mobile advertisements. It includes 22 feature fields such as users id, user information and advertisement date.

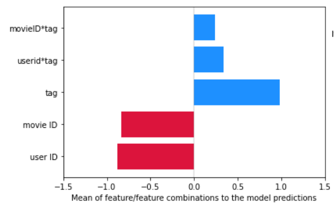
We implement all methods via Pytorch[3]. Besides, Adaptive Moment Estimation (Adam) Kingma & Ba (2014) and early stopping is used to train all models. Following Cheng et al. (2020), we set the maximum order of HOFM as 3. For methods employed multilayer perceptron, the hidden layers size has been tuned from one to three layers, and the size of hidden number is chosen from [16, 32, 64, 128, 256, 512, 1024]. For our proposed methods LR (Shapley) and NN (Shapley), the range of learning rate, weight decay, and embedding dimension, are selected from [$1e^{-4}$, $1e^{-3}$], [$1e^{-5}$, $1e^{-4}$,$1e^{-3}$] and [3, 10, 32, 64, 128, 256], respectively. The batch size used for MovieLens, Frappe, and Avazu are 4096, 128, 4096, respectively. In addition, the dropout rate varies in the range [0.1, 0.5] stepped by 0.1, and the performance threshold tried lies in the interval [1e-3, 5e-3] stepped by 0.001. We set the dimension of feature embedding layer to 256 and use a single layer perceptron whose hidden layer size is 512 for NN (Shapley). The initial model f^0 and evaluation metric P we used to exploit useful feature interactions for LR (Shapley) are LR model and AUC.

Lastly, We use two popular metrics for performance evaluation: AUC (Area Under the ROC Curve) and Logloss. Note that a slightly higher AUC or lower Logloss at 0.001-level is known to be a significant improvement for the CTR prediction task and lead to substantial revenue Cheng et al. (2016); Guo et al. (2017); Cheng et al. (2020).

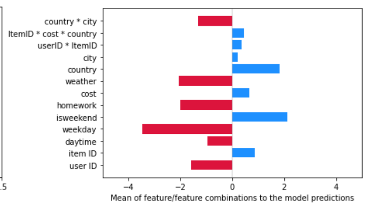
4.5 Performance Comparison

Table 2 presents the overall performance of proposed LR (Shapley), NN (Shapley) and comparing methods on three datasets. As a whole, all methods that aim to model feature interactions are able to improve the performance for CTR prediction, comparing with the LR which ignores the interaction effect among features. This verifies the effectiveness of modeling feature interactions for the CTR

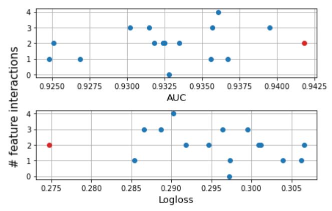
[3]https://pytorch.org/



(a)



(b)



(c)

Figure 2: (a)(b) Mean of feature/feature interactions to the model predictions on MovieLens and Frappe dataset, respectively. (c) The toy test on MovieLens dataset. The red points are the result of feature interaction set $\{[1, 3], [2, 3]\}$, achieving the best performance evaluated by AUC and Logloss.

task. Moreover, the performance of LR (Shapley) surprisingly achieves competitive results with learned feature interactions compared with more complicated models on three datasets. In particular, LR (Shapley) has learned 32 feature interactions on avazu dataset, while comparing methods modeling hundreds or thousands of feature interactions to achieve similar performance. The underlying reason could be that, different from comparing methods which enumerating all possible feature interactions (i.e., FM, AFM, HOFM) or deleting some ”useless” feature interactions (i.e., AutoFIS), LR (Shapley) iteratively discovers useful feature interactions based on the Feature Shapley values. This leads to the improvement of LR model with less noisy feature interactions but more decisive feature interactions. As shown in Fig. 2(a-b), the ”new set of features” (original features and learned feature interactions) and their mean marginal contribution to the model prediction are visualized, indicating the final prediction is driven by original features and learned feature interactions together. Another interesting result is that NN (Shapley) achieves the best performance on the Movielens and Frappe datasets. This points to a promising direction: one can firstly use our Feature Shapley framework to exploiting useful feature interactions, and then apply a more powerful prediction model to further boost the performance. In this way, both accuracy (i.e., performance boosted by more powerful models) and interpretability (i.e., the importance of features and feature interactions learned with the LR model) can be achieved simultaneously. To summarize, the satisfying results of LR (Shapley) and NN (Shapley) verifies the effectiveness of Feature Shapley to exploit useful feature interactions and indicates applicable input data (i.e., features and learned feature interactions) is of great importance to achieve more accurate predictions.

4.6 A Toy Test on MovieLens dataset

To verify whether our proposed Feature Shapley is able to discover useful feature interactions, we hence design a toy test on the Movielens dataset. Specifically, we enumerate all possible feature interaction sets for the Movielens dataset, and to see whether the best set of feature interactions matches the set of feature interactions learned by the Feature Shapley. To this end, we use linear regression as the basic model, and equip it with all possible feature interactions. For example, $\{[1, 2]\}$ represents the interaction between the first feature field and the second feature field is added to the LR model, and we can see $\{[1, 2], [2, 3], [1, 3]\}$ is exactly the FM model. The results are shown in Fig. 2(c), which indicate the feature interaction set $\{[1, 3], [2, 3]\}$ improves the performance of the LR model the most. Note that the above optimal set $\{[1, 3], [2, 3]\}$ (e.g., {[MovieID, Tag], [UserID, Tag]}) is exactly what Feature Shapley learned on the Movielens dataset (See Fig. 2a). This verifies the effectiveness of Feature Shapley framework being able to discover useful feature interactions.

5 Application to Asset Pricing

In this section, we use Feature Shapley to discover valuable feature interactions for asset pricing.

5.1 Task Description

From the machine learning perspective, asset pricing is exactly a regression task. Well-designed factors[4] are used as features of the input and the corresponding return is the target. Specifically,

[4]In the context of asset pricing, we will refer to the features as factors.

Table 3: The Performance Comparison on US stock dataset. One additional factor of FFM(3) and FFM(5) is learned by our Feature Shapley.

Model	MSE Loss	R^2 Loss	Mean of α	Number of Factors
CAPM	1.247E-03	3.921E-02	-6.270E-03	1
FFM3	1.207E-03	7.069E-02	-6.024E-03	3
FFM5	1.205E-03	7.158E-02	-6.006E-03	5
FFM3 (Shapley)	1.204E-03	7.286E-02	-5.922E-03	4
FFM5 (Shapley)	**1.203E-03**	**7.363E-02**	**-5.913E-03**	6

given a set of factors X, the return of an asset can be predicted by the weighted sum of input factors. It is important to note that the goal of asset pricing is not to learn a model with the least regression loss. Economists focus on discovering insightful factors, which are helpful to explain the underlying patterns of the financial market. To this end, a simple linear regression model is usually used to test the quality of the factors. Taking the well-known Fama & Frenchfive-factor model (FF5) Fama & French (2015) as an example, the prediction is made by considering five fators:

$$R_i - R_f = \beta(R_m - R_f) + sSMB + hHML + rRMW + cCMA + \alpha \tag{4}$$

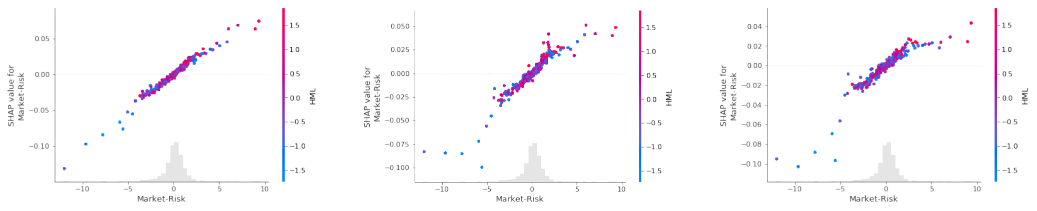
where R_i, R_f, R_m denote the asset's return; risk free rate of return, and the market return, repectively; SMB is the return on a diversified portfolio of small stocks minus the return on a diversified portfolio of big stocks; HML is the difference between the returns on diversified portfolios of high and low B/M (Book-to-market) stocks; RMW is the difference between the returns on diversified portfolios of stocks with robust and weak profitability; CMA is the difference between the returns on diversified portfolios of the stocks of low (conservative) and high (aggressive) investment companies; and α is the bias term which should be zero if the factors and the weights (β, s, h, r, and c) are able to capture all variation in expected returns.

5.2 Background

Asset pricing, which aims to understand the behavior of risk premia through return prediction, is a fundamental problem in finance for many decades. The birth of asset pricing theory starts from a single-factor model called capital asset pricing model (CAPM) Sharpe (1964). In CAPM, the expected return of a financial asset is defined as a function of the covariance between the asset's return R_i and the market return R_m, which is commonly referred as the asset's "beta". Even though CAPM provides powerful and intuitively pleasing predictions between expected return and risk, economists find that "beta" alone is not enough to explain financial return in many cases Breeden et al. (1989). Fama & French (1993) observed two interesting facts in the stock market: 1) value stocks usually outperform growth stocks. 2) stocks with small capital have the tendency to outperform large capital stocks. Motivated by these two insightful observations, the Fama & French three-factor model (FF3) Fama & French (1993) is proposed with two additional factors related to company size and value. FF3 significantly outperforms CAPM and leads to a Nobel Prize to the authors in 2013. Later on, Fama & French (2015) found that FF3 fails to capture the variation in asset return related to profitability and investment, and proposed the Fama & French five-factor model (FF5) Fama & French (2015) with two additional factors. Even though multi-factor models have achieved great success in asset pricing, the universal assumption that there are no interactions between the factors is tested not reasonable in some markets Godfrey (2018). Developing asset pricing models with high-order interaction between factors is a promising research direction.

5.3 Comparing Methods

(1) CAPM Sharpe (1964). is a classic asset pricing model that only considers the market risk factor. **(2) FF3** Fama & French (1993). FF3 is an influential three-factor asset pricing model consisted of a market risk factor, a company size factor (SMB) and a company value factor (HML). **(3) FF5** Fama & French (2015). FF5 is an extension of FF3 with additional profitability (RMW) and investment factors (CMA).



(a) stocks of high size (b) stocks of medium size (c) stocks of low size

Figure 3: The correlation between the market-risk feature and the HML feature.

5.4 Experimental Settings

To construct the stock dataset, we rank stocks from the US markets that have transaction records between 01/06/2018 and 31/05/2021, and randomly sample 100 stocks based on their market capitalization. In this way, the transaction data from 33 stocks of high market capitalization, 34 stocks of medium market capitalization, and 33 stocks of small market capitalization are selected from Yahoo Finance[5], respectively. For these stocks, we use the transaction data before 2021 (from 01/06/2018 to 31/12/2020) as the training dataset, and transaction data in 2021 (from 04/01/2018 to 28/05/2021) are treated as test dataset. The Fama & French factor data is also available[6].

We implement the asset pricing experiments via sklearn [7]. Meanwhile, linear regression model (i.e., initial model f^0) is chose as prediction model and mean squared error (MSE) (i.e., evaluation metric P) is the loss function. All methods are trained until convergence for fair comparisons.

5.5 Results

Table 4 reports the results of our experiments. Firstly, we observe that our proposed FFM5 (Shapley) is the top performing model evaluated by MSE and R^2 loss Gu et al. (2018). We can observe that the mean of α obtained by FFM5 (Shapley) is closest to zero, indicating FFM (Shapley) achieves stronger interpretability by considering the learned feature interactions. In addition, FFM3 (Shapley) not only improves FFM3 model, but also performs better than FFM5, indicating the learned feature interactions is even important than the RMW factor and CMA factor in our experimental setting. This ascertains the effectiveness of our proposed Feature Shapley framework to discovering useful feature interactions. Moreover, We also find that the interactions learned in FFM3 (Shapley) and FFM5 (Shapley) is exactly the same (i.e., the market-risk factor and HML factor). Such consistent results motivates us to discover the underlying relationships between the above two factors. To this end, we use the SHAP tool Lundberg & Lee (2017) to plot their correlations across companies with different market capitalizations. As shown in Fig.3, we found a small value of market-risk factor is more likely to be accompanied by small values of HML (marked blue), while a bigger value of market-risk factor is more likely to be accompanied by large values of HML (marked red). Their interaction effect is able to improve the performance of both FF3 and FF5 model.

6 Conclusions and Future Works

Input data is of important influence to the performance of machine learning systems. In this work, we propose a game theoretic framework Feature Shapley to discovering useful feature interactions. By input the original feature along with the learned feature interactions data, we can see even a linear regression model is able to achieve surprisingly performance on the CTR task and asset pricing scenarios. Hence, it is a promising direction to feed the "new feature set" into more powerful models, to achieve both accuracy and interpretability. However, estimating Feature Shapley values is still quite expensive, which motivates us to improve the efficiency of Feature Shapley in the future.

[5]https://finance.yahoo.com/

[6]https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

[7]https://scikit-learn.org/

REFERENCES

Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*, pp. 3351–3359, 2016.

Douglas T Breeden, Michael R Gibbons, and Robert H Litzenberger. Empirical tests of the consumption-oriented capm. *The Journal of Finance*, 44(2):231–262, 1989.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610*, 2018.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

Weiyu Cheng, Yanyan Shen, and Linpeng Huang. Adaptive factorization network: Learning adaptive-order feature interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3609–3616, 2020.

Filipe R Cordeiro and Gustavo Carneiro. A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations? In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 9–16. IEEE, 2020.

Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33:3–56, 1993.

Eugene F Fama and Kenneth R French. A five-factor asset pricing model. *Journal of financial economics*, 116(1):1–22, 2015.

Lei Feng, Senlin Shu, Zhuoyi Lin, Fengmao Lv, Li Li, and Bo An. Can cross entropy loss be robust to label noise? In *IJCAI*, pp. 2206–2212, 2020.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019.

Amirata Ghorbani and James Zou. Neuron shapley: Discovering the responsible neurons. *arXiv preprint arXiv:2002.09815*, 2020.

Chris Godfrey. The interactions between size, book-to-market and momentum in the uk. *Book-to-Market and Momentum in the UK (January 15, 2018)*, 2018.

Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. Technical report, National bureau of economic research, 2018.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 355–364, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1754–1763, 2018.

Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. Autogroup: Automatic feature grouping for modelling explicit high-order feature interactions in ctr prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 199–208, 2020a.

Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2636–2645, 2020b.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanming Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Autocross: Automatic feature crossing for tabular data in real-world applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1936–1945, 2019.

Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

Masayoshi Mase, Art B Owen, and Benjamin Seiler. Explaining black box decisions by shapley cohort refinement. *arXiv preprint arXiv:1911.00467*, 2019.

Curtis G Northcutt, Anish Athalye, and Jonas Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021.

Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1149–1154. IEEE, 2016.

Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pp. 995–1000. IEEE, 2010.

Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 255–262, 2016.

Lloyd S Shapley. *17. A value for n-person games*. Princeton University Press, 2016.

William F Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1161–1170, 2019.

Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pp. 1–7. 2017.

Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.

Rizgar Zebari, Adnan Abdulazeez, Diyar Zeebaree, Dilovan Zebari, and Jwan Saeed. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70, 2020.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.

Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European Conference on Computer Vision*, pp. 566–583. Springer, 2020.