# LEARNING VARIABLE IMPORTANCE WITH A BANDIT-BASED APPROACH

**Eddie (Yidi) Wu**
Brown University

**ABSTRACT**

Understanding variable importance is a critical task in statistical modelling, especially in high-dimensional data analysis. Traditional methods such as LASSO and tree-based approaches tend to struggle with accuracy, robustness and computational efficiency in feature selection. Building on existing combinatorial bandit frameworks, this paper proposes a novel pure exploration bandit-based method for learning feature importance in a model-agnostic manner. The iterative algorithm is inspired by best-arm identification and integrates permutation importance as a model-agnostic reward mechanism. Numerical simulations show that the approach achieves higher accuracy and faster convergence compared to existing methods, and can adapt to both offline and online settings. Additionally, applying the framework to empirical asset pricing demonstrates that it recovers key predictors of stock returns while efficiently filtering out noises. These results suggest that the proposed method provides a robust, interpretable and computationally scalable solution for learning feature importance.
*Click here for the GitHub repository*

***K*eywords** Variable importance · Model selection · Multi-armed bandit · High-dimensional data

## 1 Introduction

In the era of big data, a growing abundance of high-dimensional datasets is both a blessing and a curse for researchers and practitioners. Traditional methods such as linear regression and generalized linear models break down in the presence of high dimensional covariates. Regularization techniques like ridge and LASSO regressions can potentially overcome this issue but they are heavily dependent on choosing the appropriate penalty coefficient. Furthermore, model performance is compromised when signal-to-noise ratio is low. Sophisticated machine learning techniques, such as tree-based bagging and boosting algorithms which can capture nonlinear relationships, might work well on high dimensional data, but face challenges in computation and interpretability. In the presence of many covariates, understanding which variables have the greatest influence on outcome is of particular interest to stakeholders. For instance, economists can uncover economic insights from real-world data and improve policy effectiveness and efficiency by specifically targeting the important variables.

Every model has its own method for assessing variable importance. In LASSO, important variables can be inferred as those with non-zero coefficients. In random forest, one can rank variables based on average reduction in some impurity measures. When the number of covariates is large and data is noisy, these model-specific methods might not accurately pick out the important variables. Moreover, computation becomes highly demanding for many machine learning algorithms. As such, Durand and Gagne (2014) formulate online feature selection as a combinatorial bandit problem and explores feature importance iteratively using fully connected neutral network run on a subset of features every time. [1] Subsequently, Liu and Rockova (2021) propose a variable subset selection method based on combinatorial bandits, whose rewards are given by a Bayesian additive regression tree (BART), that yields sizable computational gains and strong empirical performance on large datasets. [2]

In this paper, building on the earlier work, I develop a model-agnostic exploratory multi-armed bandit framework for learning variable importance and running feature selection on high-dimensional datasets with sparsity where the number of covariates far exceeds the number of observations but most of the covariates have no influence on the

outcome. While previous work integrates multi-armed bandits with a specific statistical model such as BART, I employ model-agnostic feature importance measures that are compatible with any statistical model under consideration by the researcher. Furthermore, in previous work, the bandit objective is to minimize expected cumulative regret during experimentation and this can be achieved by the Thompson sampling algorithm. In my set-up, the objective is to learn variable importance with high confidence as quickly as possible. Hence, my sampling algorithm is inspired by the best arm identification literature which considers reward gained during sampling to be insignificant relative to reward earned from implementing the optimal actions post sampling.

In numerical experiments, I use simulated data whose true data generating process (DGP) is known and given by

$$y = f(X) + \epsilon$$

where $\epsilon$ is zero mean noise with constant variance. $X$ is a $p \times 1$ vector of features and $f(\cdot)$ introduces sparsity where only $q$ features in $X$ have significant effect on $y$ and $q << p$. I demonstrate several key advantages of my iterative approach and the result is robust across multiple types of DGPs. First, the algorithm attains a higher accuracy of separating the true important variables from noise compared to model-specific methods. Second, the algorithm enables the posterior distributions of the truly important variables to converge at a faster rate than using Thompson sampling. Lastly, on large datasets e.g. those with more than 10,000 observations and more than 1000 covariates, the algorithm has considerable computational gains versus model-specific variable importance methods.

On the empirical front, I apply my method to the asset pricing study by Gu, Kelly and Xiu (2020), who construct a panel data of 94 firm-level features of all publicly companies in the US. [3] Their study aims to compare the out-of-sample predictive performances of various mainstream machine learning algorithms. Using my approach, I analyze variable importance in an online setting i.e. data arrives at a monthly frequency, and find a largely similar set of important predictors for stock returns as identified in their paper.

This research aims to contribute to the repertoire of algorithms for learning variable importance in high-dimensional data that are robust, reliable and computationally efficient. The remainder of the paper is organized as follows: literature review, detailed explanations of the framework, numerical experiments, empirical study, discussions, and conclusion.

## 2 Literature Review

Several papers have forged a connection between feature selection and reinforcement learning. Gaudel and Sebag (2010) set up feature selection as a Markov Decision Process (MDP) whose state space is given by the power set of the feature set and choose features to minimize the generalization error of the learned model. [4] They solve the MDP with Monte Carlo tree search. Rasoul et al. (2021) tackle a similar MDP but employ a temporal difference algorithm to find the best subset of features. [5] Ashtiani et al. (2014) partition the sample space into sub-regions and select features within each locality using Upper Confidence Tree. [6]

To the best of my knowledge, Durand and Gagne (2014) are the pioneer in formulating feature selection as a combinatorial multi-armed bandit problem. [1] Their algorithm represents each feature as a bandit arm and iteratively selects subsets of features with constant subset size via Thompson sampling and computes rewards based on the accuracy of predictions by a neural network. Liu and Rockova (2021)'s framework combines Thompson sampling with a median probability model (MPM) computational oracle to choose the set of features at every time step and obtains rewards from BART. More importantly, they provide theoretical results on the regret bounds and consistency of their algorithm. [2] Thompson sampling is known to attain the optimal logarithmic in-sample expected regret for the stochastic multi-armed bandit problem, [7] as well as a logarithmic in-sample expected regret for the combinatorial bandit problem. [8] However, it only achieves a polynomial rate of convergence of posterior distributions of the arm parameters, [9] and bandit algorithms with logarithmic expected regret bound is shown to be far from optimal for the best arm identification problem. [10]

In many settings where researchers are more concerned with finding the best subset of variables as quickly as possible rather than minimizing the cumulative error of selecting wrong variables during feature selection, algorithms biased towards exploration might be preferred. A number of papers have proposed stochastic (non-combinatorial) bandit algorithms where the probability of selecting a sub-optimal arm decays at an exponential rate. [11] [12] [13] [14] Notably, Russo (2016) proposes a class of top-two algorithms that converges at an exponential rate with the best possible exponent among all allocation rules. [9] Kasy and Sautmann (2021) develop an exploration sampling algorithm which transforms the Thompson sampling probabilities and attains exponential rate of convergence. [15] Caria et al. (2023) introduce Tempered Thompson Sampling which alternates between uniformly random assignment and Thompson sampling with a fixed probability at every iteration. [16] The combinatorial bandit implementation for feature selection by Liu et al (2021) takes a similar approach for exploration: their sampling algorithm has probability $\epsilon$ of uniformly choosing a random subset of features, and probability $(1 - \epsilon)$ of choosing $K$ features via Thompson sampling. [17]

Several papers also address the question of best subset identification in combinatorial bandit. Wang and Zhu (2022) develop a pure exploration sampling algorithm which enhances exploration by pulling subsets of arms with the largest reward gaps and the least number of observations. [18] Nakamura and Sugiyama (2024) modify the algorithm of Wang and Zhu (2022) to attain a tighter upper bound on the sample complexity in a fixed confidence setting. [19] In this paper, I propose a top-two Thompson sampling process for combinatorial bandit based on Russo (2016) because the Bayesian framework is more interpretable and compatible with feature selection.

While previous work on adaptive feature selection such as Durand and Gagne (2014) and Liu and Rockova (2021) wed combinatorial bandit to a specific statistical model e.g. using tree-based models to compute bandit rewards, this paper aims to develop a model-agnostic framework. A common model-agnostic way of evaluating feature importance is the partial dependence plot first proposed by Friedman (2001), which computes the expected outcome for each value in the support of the feature of interest, where the expectation is taken over the joint distribution of all other features. [20] Then, the differential in expected outcome at different values of the feature of interest conveys the importance of the feature. Another method is the Shapley value which originates from cooperative game theory studied by Shapley (1953). [21] It evaluates the contribution of a feature to outcome prediction by examining all feature subsets including and excluding the feature of interest. Several subsequent papers have overcome the computational intractability of Shapley value via efficient approximations. [22] [23]

The popular permutation importance method, proposed by Breiman (2001), randomly permutes the values of a feature and examines the change in predictive performance of the model. [24] As variable importance assessment based on tree splits in tree-based models is biased, it serves as a corrected measure. [25] Molnar et al. (2023) study the theoretical link behind permutation importance and the true DGP and formalize permutation importance as a statistical estimator of the ground truth estimand. [26] Other commonly used model-agnostic feature importance method include Locally Interpretable Model-agnostic Explanations (LIME) proposed by Ribeiro et al. (2016). [27] The idea is to learn an interpretable model locally around the prediction to inspect variable importance. In this research, I use permutation feature importance which is the most suited for the purpose of iterative feature selection, as explained in the subsequent sections. The potential limitations of this method and possible ways to overcome those are also discussed in Section 6.

## 3 The Bandit-based Approach

In this section, I present my bandit-based feature selection framework in depth, and provide an overview of combinatorial bandit, Thompson sampling, and model-agnostic feature importance measures.

### 3.1 Combinatorial multi-armed bandits

The canonical stochastic multi-armed bandit (MAB) problem is a single player game with the following elements:

- Arms: there is a finite set of actions ("arms") indexed as 1, 2, ... K.
- Rewards: each arm is associated with a reward distribution, parameterized by $\theta_1, \theta_2, ...\theta_K$ respectively (these are constants). The distribution type may be known but the parameters are unobserved. The distribution type may or may not be the same across arms.

At every time step, the player chooses an action $k$ and samples a reward $r_t$ from the associated distribution. The objective is to maximize expected cumulative reward $E[\sum_{t=1}^{T}(r_t)]$ or minimize expected cumulative regret over time. Let $\theta^* \coloneqq \arg\max_k \theta_k$, regret is defined as $(\theta^* - \theta_k)$, and expected cumulative regret is $\sum_{t=1}^{T}(\theta^* - \theta_{i(t)})$, where $i(t)$ is a mapping from time step $t$ to the index of the arm chosen at $t$. The two mainstream classes of algorithms, Thompson sampling [28] and Upper Confidence Bound (UCB) [29], are developed for this purpose.

**Algorithm 1** Thompson sampling

**Input :** Prior distribution $P_{k,0}$ for the unknown parameters $\theta_k$, for $k = 1, ..., K$.
**Output :** Posterior distribution $P_{k,T}$ for $k = 1, ..., K$

**for** $t \in [1, T]$ **do**
  Sample $\theta_{k,t} \sim P_{k,t-1}$ for $k = 1, ..., K$.
  $s_t \leftarrow \arg\max_k \theta_{k,t}$
  Sample $r_t \sim Q_{s_t}$ where $Q_{s_t}$ is the true reward distribution of arm $s_t$.
  Update $P_{s_t,t-1}$ using $r_t$ to obtain $P_{s_t,t}$.
  For all arms $k \neq s_t$, $P_{s_t,t} \leftarrow P_{s_t,t-1}$.
**end for**

To illustrate stochastic MAB and Thompson sampling with an example, let's consider the simplest case of Bernoulli bandit. Suppose there are 3 arms, each giving a reward of one with probability $\theta_k$, for $k = 1, 2, 3$, and $\theta_k$ is not observed by the player. Following algorithm 1, the player assumes a Beta distribution for each $\theta_k$ with $\alpha_{k,0} = \beta_{k,0} = 1$ i.e. uniform prior. At every time step, the player samples a $\hat{\theta}_k$ from each Beta distribution, and play the arm given by $\arg\max_k \theta_k$. He then observes a reward $r_t \in \{0, 1\}$ from the arm that is played, and update the Beta distribution of the arm accordingly using the reward. This process repeats for $T$ time steps and, in theory, maximizes the expected cumulative reward collected till $T$.

An alternative way to define the objective of MAB is to identify the best arm with a specified confidence level as quickly as possible (fixed confidence) or to attain as high confidence that an arm is optimal as possible within a given $T$ (fixed budget). Pure exploration algorithms such as Track-and-Stop [12] and top-two Thompson sampling [9] are designed to accomplish such objectives.

Combinatorial multi-armed bandit has a similar setup but differs from the canonical MAB in arm selection and reward structure. In combinatorial MAB, the player chooses and plays a subset of actions $S_t$, termed the "super-arm", rather than a single action at every time step $t$. The reward of a super-arm $R(S_t)$ is a function of the set of actions chosen. A common assumption is linear reward aggregation:

$$R(S_t) = \sum_{k \in S_t} r_t^k$$

$R_t(S)$ can also be a nonlinear function of the base arm rewards. *Bandit feedback* setting is where the player only observes the total reward of the selected super-arm $R(S_t)$, while *semi-bandit feedback* setting is where the player only observes the reward of the chosen base actions, but not $R(S_t)$. The term "global reward" refers to $R(S_t)$ while "local reward" refers to $r_t^k$.

### 3.2 Bandit feature selection

I present how the algorithm works and provide a detailed explanation of its design. The algorithm treats each feature as a base arm and adopts a semi-bandit feedback mechanism i.e. local rewards are observed while global rewards are not directly observed. To begin with, I assume that each feature $k$ in the feature vector $X$ has a latent variable $\theta_k \in [0, 1]$ which encapsulates its level of influence on the outcome $y$. The player's goal is to learn the posterior distributions of $\theta_k$ from data using a Bayesian approach:

1. First, the player places Beta distribution priors with parameters $\alpha_{k,0}, \beta_{k,0}$ on each $\theta_k$. These parameters govern the player's initial knowledge of feature importance. Having $\alpha_{k,0} = 1$ and $\beta_{k,0} = 1$ is equivalent to uniform prior on $[0, 1]$ while adjusting the parameter values can encode prior knowledge.
2. At the first time step $t = 1$, the player samples a $\theta_{k,1}$ for every feature, and define a super-arm $S$ to include features with $\theta_{k,1} \geq 0.5$ (if step 2 stops here, this is Thompson sampling). Next, the player generates a random number $u$ from $U[0, 1]$.
   - If $u \leq 0.5$, he plays the super-arm $S_t = S$.
   - If $u > 0.5$, he samples a $\theta'_{k,1}$ for every feature and define a super-arm $S'$ to include features with $\theta'_{k,1} \geq 0.5$. If $S' = S$, he samples another $S'$ until $S' \neq S$. Then, he plays the set $S_t = (S \cup S') \backslash (S \cap S')$, i.e. the union minus the intersection of the two super-arms.
3. The player fits a statistical model of his choice using the chosen features $S_t$ in the data, and observe the binary 0-1 base-arm reward $r_t^k$ for each feature computed using permutation feature importance. $P(r_t^k = 1)$ is assumed to be strictly increasing in the true level of importance of a variable (more on that in Section 3.3).
4. Update the Beta distribution of each feature $k$ using the binary $r_t^k$, to obtain the posterior distributions of $\theta_k$ at this time step.
5. Repeat step 2 - 4 for a pre-specified number of iterations, or till convergence. An example of convergence criterion could be that for all features with $E(\theta_k) \geq 0.5$ where expectation is with respect to the posterior Beta distribution, the ranking of these variables according to their mean importance $E(\theta_k)$ stays stable for a certain number of iterations.

There are several details worth highlighting. **First**, the top-two Thompson sampling process for combinatorial bandit in step 2 is motivated by the class of top-two algorithms in Russo (2016). Since Thompson sampling is optimal for minimizing cumulative regret, top-two Thompson sampling limits the exploitation of features with high known importance to encourage greater exploration. Taking the union minus the intersection of $S$ and $S'$ is essentially shifting

away from features that are more likely to be selected and focusing on the under-explored features with a $0.5$ probability at every time step. Russo (2018) shows that for canonical MAB, although a constant threshold of $0.5$ for $u$ does not exactly lead to the optimal exponent in the posterior convergence rate, the $0.5$ threshold guarantees that the exponent in top-two algorithms is always within a factor of 2 of the optimal exponent. [9] As Russo (2018) only conjectures that the same result holds for top-m arms identification, the formal proof extending this result to combinatorial MAB can be a direction for future research.

**Second**, in the combinatorial bandit literature, an oracle refers to the computational procedure that selects the best subset of arms when $\theta_{k,t}$'s are provided. In step 2, after sampling the $\theta_{k,t}$'s, the oracle in this algorithm selects features with $\theta_{k,t} \geq 0.5$ into the super-arm. This design is derived from the objective of maximizing the accuracy of selecting the true important variables ((Liu-Rochova (2021))). Suppose the global reward is defined as $R_t(S_t) = \left[\prod_{k \in S_t} r_t^k\right] \left[\prod_{k \notin S_t}(1 - r_t^k)\right]$, if $S_t$ coincides with the set of true important variables, $R(S_t)$ is more likely to be 1 than if $S_t$ is not equal to the set of true important variables. Then, the expected global reward can be shown to be $E[R_t(S_t)] = \left[\prod_{k \in S_t} E(\theta_k)\right] \left[\prod_{k \notin S_t}(1 - E(\theta_k))\right]$ assuming that $r_t^k$'s are independent. and $S_t = \{k : \theta_{k,t} \geq 0.5, k = 1, ...K\}$ maximizes the expected global reward. The validity of the assumption that $r_t^k$'s are independent depends on the choice of base-arm reward mechanism. More discussions on that can be found in section 3.3 and section 6.

**Third**, after obtaining the posterior distributions of $\theta_k$, the next-step is to build a parsimonious model by selecting a threshold $\nu \in [0, 1]$ such that variables with posterior mean $E(\theta_k) \geq \nu$ are included in the final statistical model. A natural choice for the threshold is $0.5$ because in this combinatorial bandit set-up, the posterior probability of a model is given by $P(M_L|\mathcal{D}) = \prod_{k=1}^{K} E(\theta_k)^{l_k}(1 - E(\theta_k))^{1-l_k}$ where $L = [l_1, l_2, ...l_K]$ and $l_k \in \{0, 1\}$ indicates whether a variable $k$ is included, and $\mathcal{D}$ is data. A model $M$ with subscript $L$ means it includes variables whose $l_k = 1$. As such, the model with the highest posterior probability, i.e. the MAP model, is one which encompasses variables with posterior mean importance greater than 0.5. Furthermore, the MAP model exactly coincides with the median probability model, defined as the model which includes only variables with posterior mean greater than $0.5$ in Barbieri and Berger (2004), who show that under certain conditions, the median probability model is the best predictive model. [30]

### 3.3 Local reward mechanism

I use model-agnostic methods of assessing feature importance to compute local rewards in the semi-bandit feedback setting. I provide an overview of three popular methods - partial dependence (PD), Shapley value (SV), and permutation importance (PI) - and explain the advantages of PI over the other methods.

**PD** measures the marginal effect of a feature $X_k$ on the predicted outcome $f(X_k)$ where $f(\cdot)$ is the true functional form of the model. Given a point $x_k$ in the support of $X_k$, the PD function $f_{PD}(\cdot)$ is defined as:

$$f_{PD}(x_k) = E_{X'} f(x_k, X') = \int f(x_k, X') dP(X')$$

i.e. the average predicted outcome when $X_k = x_k$. In terms of implementation, the sample PD function can be computed by selecting a set of $D$ points in the support of $X_k$ and using bootstrap for each of the points. Then, the PD feature importance for a feature $X_k$ is given by:

$$\sqrt{\frac{1}{D-1} \sum_{d=1}^{D} \left( \hat{f}_{PD}(x_k^{(d)}) - \frac{1}{D} \sum_{d=1}^{D} \hat{f}_{PD}(x_k^{(d)}) \right)}$$

i.e. the variance in the predicted outcome over the set of $D$ points. There are two major drawbacks to using this feature importance measure as the reward function. First, PD does not capture the interactions between features. Suppose a feature $k$ affects the outcome only via interaction with other features, there is a possibility that PD show a flat relationship between the outcome and feature $k$. Second, since computing PD requires bootstrap, when data is high dimensional and the number of observations is small relative to the number of features, PD estimation tends to be imprecise. Thus, PD is not suitable as a reward function in my setup.

**SV** assesses the amount of contribution of a feature to the predicted outcome. Let $F$ be the set of all features, and $f_S(\cdot)$ be a model that includes only features in the subset $S$, SV of a feature $k$ is defined as:

$$\phi_k = E_{X_{S \cup \{k\}}} \left[ \sum_{S \subseteq F \setminus \{k\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[ f_{S \cup \{k\}}(X_{S \cup \{k\}}) - f_S(X_S) \right] \right]$$

i.e. the expectation of the weighted average difference in predicted outcome with feature $k$ and predicted outcome without feature $k$. As the exact formula involves an exponential number of subsets of $F$, several computationally

**Algorithm 2** Bandit feature selection

**Input :** Beta prior $\alpha_{k,0}, \beta_{k,0}$ for $k = 1, ..., p$; Base estimator $M$; reward threshold $\nu$; number of iterations $T$; data $D$.
**Output :** Beta posterior $\alpha_{k,T}, \beta_{k,T}$ for $k = 1, ..., p$.

Initialize $p$ Beta distributions using the beta priors.
**for** $t \in [1, T]$ **do**
  *Top-two Thompson sampling*:
  Sample $\theta_{k,t} \sim$ Beta$(\alpha_{k,t-1}, \beta_{k,t-1})$ for $k = 1, ..., p$.
  $S \leftarrow \{k : \theta_{k,t} \geq 0.5\}$.
  Sample $u \sim U[0, 1]$.
  **if** $u \leq 0.5$ **then**
    $S_t \leftarrow S$
  **else if** $u > 0.5$ **then**
    $S' \leftarrow S$
    **while** S = S' **do**
      Sample $\theta'_{k,t} \sim$ Beta$(\alpha_{k,t-1}, \beta_{k,t-1})$ for $k = 1, ..., p$.
      $S' \leftarrow \{k : \theta'_{k,t} \geq 0.5\}$
    **end while**
    $S_t = (S \cup S') \backslash (S \cap S')$
  **end if**

  *Permutation importance reward*:
  $D_{in}, D_{out} \leftarrow$ train_test_split$(D)$
  Fit estimator $M$ using features $S_t$ and data $D_{in}$.
  **for** feature $k \in [1, p]$ **do**
    $\pi_k \leftarrow$ permutation_importance$(M, D_{out}, k)$
    $r_t^k \leftarrow 1(\pi_k \geq \nu)$
  **end for**

  *Update Beta parameters*:
  **for** feature $k \in [1, p]$ **do**
    $\alpha_{k,t} \leftarrow \alpha_{k,t-1} + r_t^k$
    $\beta_{k,t} \leftarrow \beta_{k,t-1} + (1 - r_t^k)$
  **end for**
**end for**

**Algorithm 3** Permutation importance

**Input :** Fitted estimator $M$; test data $D_{out}$; feature $k$.
**Output :** Feature importance $\pi_k$

Initialize the number of times to repeat permutation $B$.
**for** $b \in [1, B]$ **do**
  $\hat{y} \leftarrow M(D_{out})$
  $R^2 \leftarrow L(y, \hat{y})$ where $L(\cdot)$ is the function for computing $R^2$.
  Permute feature $k$ in the design matrix of $D_{out}$ to obtain $D'_{out}$.
  $\hat{y} \leftarrow M(D'_{out})$
  $R'^2 \leftarrow L(y, \hat{y})$
  $\pi_{b,k} \leftarrow R^2 - R'^2$
**end for**
$\pi_k \leftarrow \frac{1}{B} \sum_{b=1}^{B} \pi_{b,k}$

tractable methods such as SHAP have been proposed to approximate SV. [22] While SV appears to be a good measure of feature importance, it depends strongly on the estimated model $\hat{f}(S \cup \{k\})$. Consider a trained model which overfits to noise in training data and produces poor out-of-sample predictions, some noise variables will have large SV due to their contribution to the predicted outcome. SV will consider those variables important even though they are just noise. As such, SV requires careful model choice and hyperparameter tuning.

**PI** measures the drop in out-of-sample predictive performance of a model if feature $k$ is randomly permuted in the out-of-sample data. Let $X_k$ be a feature whose importance we are interested in, $\tilde{X}_k$ be feature $k$ after permutation, $X'$ be all the other features, $P(\tilde{X}_k|X')$ be the conditional density of $\tilde{X}_k$, and $L(\cdot)$ a loss function, then PI is defined as: [26]

$$\pi_k = E_{X',Y}\left[E_{\tilde{X}_k|X'}\left(L(Y, f(\tilde{X}_k, X'))\right)\right] - E_{X,Y}\left[L(Y, f(X))\right]$$

Permutation breaks the relationship between feature $k$ and the outcome. If feature $k$ is truly important, the change in loss will be significant. For actual implementation, the dataset is divided into a training and a test set. The training set is used to obtain the estimated model $\hat{f}(\cdot)$ while the test set is used to compute PI. Given a choice of model, PI does not require optimal hyperparameter tuning because it examines the relative change in loss on out-of-sample data. Even if the model overfits to noise, permuting noise variables does not influence out-of-sample loss considerably while permuting true important variables might relatively worsen the out-of-sample model performance. On simulated data, Altmann et al. (2010) show that PI works very well in deciding the significance of variables and distillating the important variables. More details on computation are presented in algorithm 3. I use predictive $R^2$ as the performance metric in my implementation.

There are multiple ways of obtaining a 0-1 binary reward $r_t^k$ from $\pi_k$ which is continuous between $[0, 1]$ when $R^2$ is the metric. For example, $r_t^k$ can be sampled from $Ber(\pi_k)$ and larger PI leads to higher expected reward. However, in most empirical applications, $\pi_k$ tends to be much smaller than $0.5$, meaning that most of the times an important feature does not receive a positive reward. Another way to binarize $\pi_k$ is to define an importance threshold $\nu$. Suppose a researcher deems a variable to be important only if it leads to a $5\%$ or more drop in predictive $R^2$, then he can set $\nu = 0.05$, and $r_t^k = 1(\pi_k \geq \nu)$. The threshold can be for either absolute or relative change in out-of-sample performance, depending on the nature of the application. If a prediction problem is inherently difficult i.e. out-of-sample $R^2$ is small, a relative change threshold is preferred. The drawbacks of PI, together with ways to address them, are discussed in section 6. More discussions on why I choose to observe local reward rather than global reward can also be found in section 6.

## 4 Simulation Experiments

Using simulated data from numerous known DGPs, I demonstrate several merits of bandit feature selection. First, it attains a higher accuracy of identifying the true important variables from noise compared to model-specific methods. Second, using top-two Thompson sampling which aids exploration achieves a faster rate of convergence than using Thompson sampling, and is less likely to be trapped in local minima. Lastly, the iterative learning algorithm is highly versatile because it is compatible with a broad range of machine learning algorithms and can work in both offline and online settings. In this section, I present the results that are based on the more challenging DGPs, and other results can be obtained from the GitHub code repository.

### 4.1 Bandit versus model-specific methods

I compare the performance of bandit feature selection against other methods using two DGPs in this subsection. Both DGPs are based on examples presented in Friedman (1991). [31]

The first DGP is:

$$f(x_i) = 10\sin(\pi x_{i,1} x_{i,2}) + 20(x_{i,3} - 0.5)^2 + 10x_{i,4} + 5x_{i,5} \tag{1}$$

$$y_i = f(x_i) + \epsilon_i \tag{2}$$

where $1 \leq i \leq n$, $x_i \sim [0, 1]^p$ and $\epsilon_i \sim N(0, 1)$. As the outcome is nonlinear in the first three features and linearly additive in the next two features, this is a rather challenging DGP for variable selection. To demonstrate the ability of bandit feature selection to learn from high-dimensional data, I set $n = 300$ and $p = 1000$.

I generate a dataset $\mathcal{D}$ and use random forest regressor as the base estimator. Random forest is a ensemble method which predicts outcome as the averaged predicted outcome of a set of regression trees:

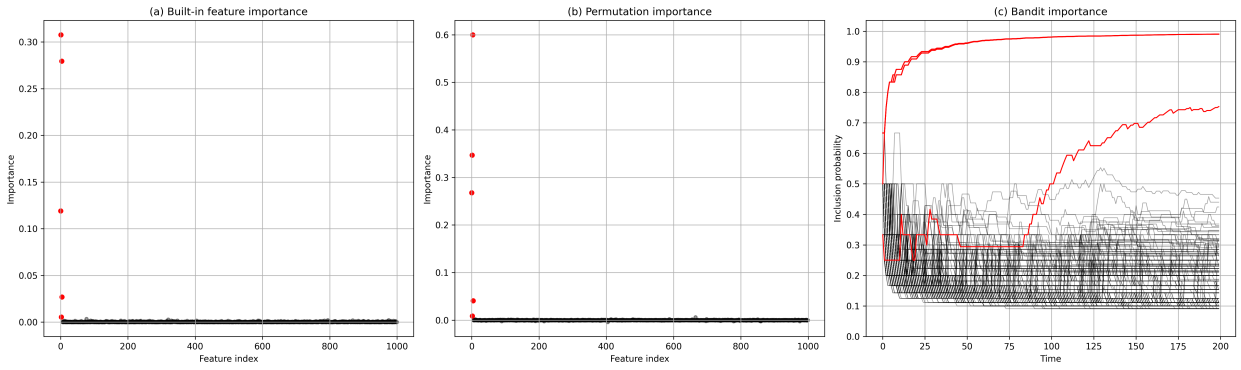


Figure 1: Comparing feature importance using data generated from (1). Red dots and lines are the true important features while black dots and lines are noise.

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} T_m(x)$$

where $M$ is the number of regression trees and $T_m(X)$ is the prediction of a tree $m$ conditioning on $x$.

First, I fit random forest on the full data sample $\mathcal{D}$ using the optimal hyperparameters, which are found via grid search and k-fold cross validation. With the fitted model, I obtain the built-in impurity based feature importance, i.e. the importance of a feature is the mean reduction in variance across all splits on that feature in the forest. (Figure 1 panel (a))

Next, using the same fitted model, I compute the permutation feature importance by algorithm 3 for each feature. (Figure 1 panel (b))

Lastly, using the same $\mathcal{D}$, I run bandit feature importance by algorithm 2 with a uniform prior and 200 iterations. (Figure 1 panel (c)) To simulate a real-world situation where one does not have sufficient data for k-fold cross validation and hyperparameter tuning, I do not use the optimal hyperparameters found previously, but specify the hyperparameters based on heuristics, e.g. `n_estimators`=100 and `max_depth`=10, because bandit feature selection works without the need for optimally tuned hyperparameters. More discussions can be found in section 6.

In Figure 1, panel (a) shows built-in impurity-based importance and panel (b) shows permutation importance computed from the optimally tuned random forest regressor. Both methods are unable to separate $x_3$ (I drop the $i$ subscript to indicate that $x_3$ refers to this feature in general, and not of any particular $i$) from noise and attribute a low importance to $x_5$. Panel (c) shows the inclusion probabilities of each feature over time. All noise features have inclusion probabilities suppressed below 0.5, while $x_1, x_2, x_4, x_5$ have inclusion probabilities rising to above 0.5 in mere 15 iterations. $x_3$ has inclusion probability fluctuating around 0.3 initially, but once the algorithm is highly confident about the other important features, it is also able to learn the importance of $x_3$ in later iterations. By the end of 200 iterations, we see a clear separation between noise and all the true important variables.

The second DGP in this experiment is:

$$f(x_i) = 0.1e^{4x_{i,1}} + \frac{4}{1 + e^{-20(x_{i,2} - 0.5)}} + 4x_{i,3} + 3x_{i,4} + 2x_{i,5} \tag{3}$$

$$y_i = f(x_i) + \epsilon_i \tag{4}$$

The specifications of $n, p, x_i, \epsilon_i$ are identical to the first DGP. (3) is nonlinearly additive in the first two features and linearly additive in the next three features. I use random forest estimator and perform the same comparison. Results are shown in Figure 2. In this DGP, the most challenging variables to be identified are $x_4$ and $x_5$. Impurity-based importance fails to pick out $x_5$ and assigns a very small importance to $x_4$, while permutation importance computed directly on the fitted random forest with optimal hyperparameters misses out on both $x_4$ and $x_5$. On the other hand, bandit feature importance identifies $x_1, x_2, x_3$ and $x_4$ within 20 iterations, and manages to pick out $x_5$ after several more iterations.

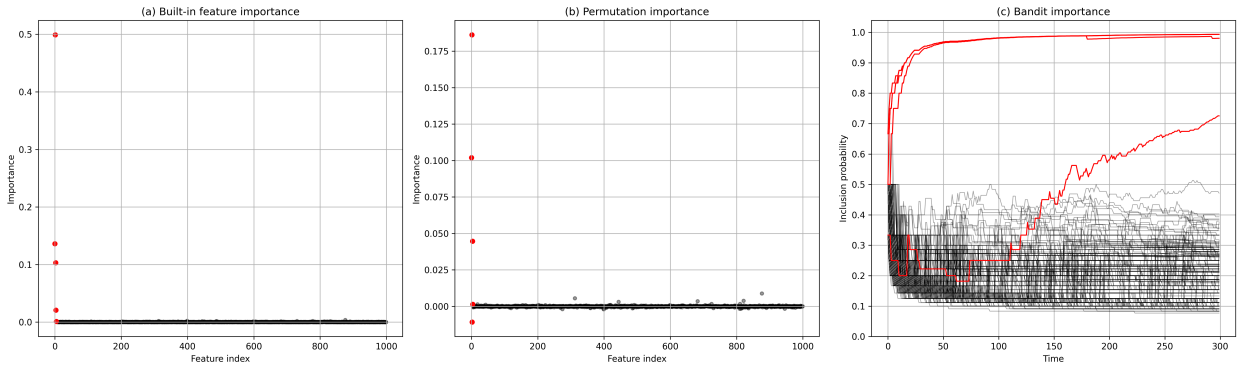


Figure 2: Comparing feature importance using data generated from (3). Red dots and lines are the true important features while black dots and lines are noise.

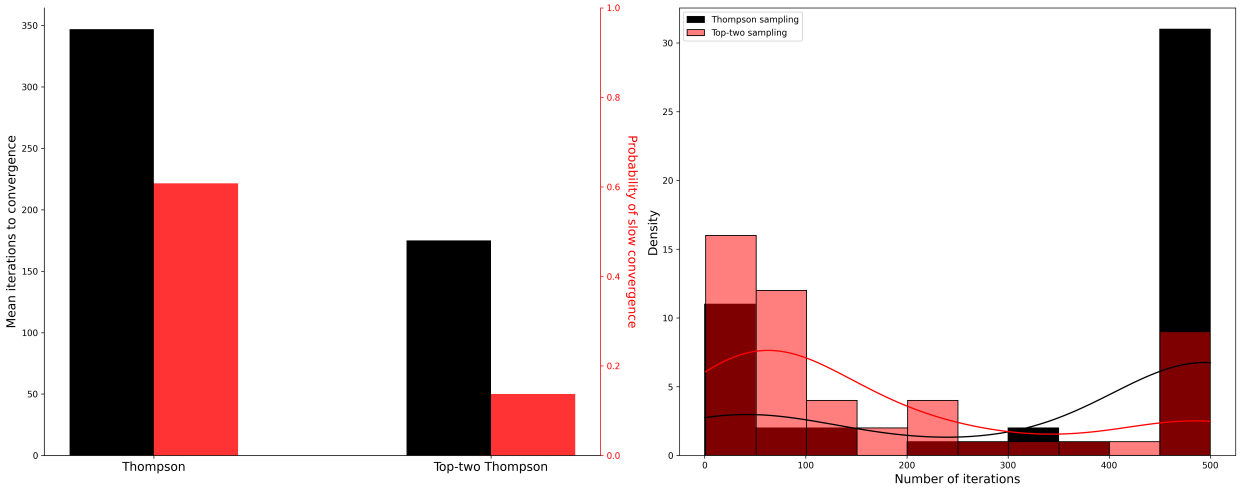


Figure 3: Comparing rate of convergence between Thompson sampling and top-two Thompson sampling using data generated from (1), with random forest as the base estimator. Left: left y-axis is the number of rounds to iteration while the right y-axis is the probability of not converging within 500 iterations. Right: the density plot of time taken to converge.

### 4.2 Exploration and convergence

I compare the number of iterations taken for Thompson sampling to converge versus top-two Thompson sampling. In the first experiment, I generate data from (1) and use random forest as the base estimator. Over 50 trials, I keep track of the number of iterations to convergence for each algorithm, and cap the maximum number of iterations at 500, and set $n = 300$ and $p = 500$, to limit the amount of computation. Convergence is defined as all true important variables having posterior inclusion probabilities above 0.5 and all noise variables having posterior inclusion probabilities below 0.5. I report the mean number of iterations taken for convergence and the probability of not converging within 500 iterations.

In Figure 3, focusing on the RHS panel which shows the density plots of the number of iterations to convergence, in $60\%$ of the trials, Thompson sampling does not converge within 500 iterations, implying it is stuck in a local minimum that is difficult to escape. On the other hand, top-two Thompson sampling is much more robust to local minima and only less than $20\%$ of the trials do not converge within 500 iterations.

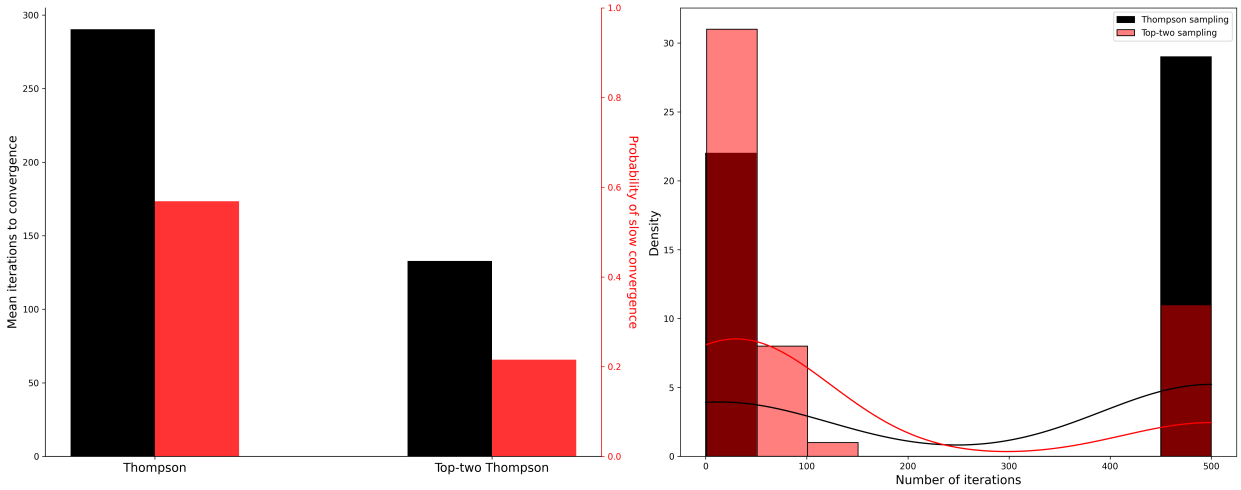


Figure 4: Comparing rate of convergence between Thompson sampling and top-two Thompson sampling using data generated from (3), with LASSO as the base estimator. Left: left y-axis is the number of rounds to iteration while the right y-axis is the probability of not converging within 500 iterations. Right: the density plot of time taken to converge.

To demonstrate the robustness of this result, in the second experiment, I generate data from (3) and use LASSO as the base estimator. LASSO is linear regression with L1 penalty in the loss function:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\theta)^2 + \lambda||\theta||_1$$

The reason for picking LASSO is that the variables in (3) are additively separable. Despite being mis-specified, LASSO is still able to learn variable importance via bandit feature selection. Similarly, we observe lower probability of getting stuck in a local minimum and faster rate of convergence.

The outperformance of top-two Thompson sampling originates from its tendency to explore variables with lower inclusion probabilities. By selecting the union minus the intersection of two super-arms, the variables that are more frequently selected are filtered out while the variables that are less likely to be selected are played.

### 4.3 Online setting

So far, this paper has been focusing on offline learning, where a single dataset $\mathcal{D}$ is available for learning feature importance. Very often, practitioners are interested in online feature learning, where data comes in batches at a fixed time interval: $\mathcal{D}_1, \mathcal{D}_2 ... \mathcal{D}_n$. In every $\mathcal{D}_t$, either all features or a subset of features can be observed. In the case where all features are observed at every time step, online learning is analogous to offline learning with bootstrap, but the difference lies in that bootstrap produces the same data points as the original $\mathcal{D}$, while online learning receives new data points from the same DGP. Alternatively, the case where the observer only chooses to observe a subset of observations at every time step has crucial real-world relevance because observing all features might be expensive or impossible in some settings. Then, bandit feature selection offers a solution to learning feature importance under such scenarios.

In this section, I demonstrate the ability of bandit feature selection to learn feature importance from data arriving in batches. I consider three different DGPs, each using a different base estimator, to show the model-agnostic nature of the algorithm. The first two DGPs are from equation (1) and (3) from Friedman (1991), [31] and the last DGP is a challenging case from Liang et al. (2018) [32] as introduced in Liu-Rockova (2021). [2]

In the last DGP:

$$f(x_i) = \frac{10x_{i,2}}{1 + x_{i,1}^2} + 5\sin(x_{i,3}x_{i,4}) + 2x_{i,5} \tag{5}$$

$$y_i = f(x_i) + \epsilon_i \tag{6}$$

where $x_i = \frac{e_i + z_i}{2}$, and $\epsilon_i \sim N(0, 0.5)$, for $i = 1, ...n$. $z_i$ is a $p \times 1$ vector whose every entry is i.i.d. $N(0, 1)$, and $e_i$ is a $p \times 1$ vector constructed by first sampling a $\eta \sim N(0, 1)$ and then set every entry of $e$ to be equal to $\eta$. This

ensures that all features in $x_i$ are mutually correlated with a correlation coefficient of 0.5. The DGP is challenging not only because of its nonlinearity and feature interactions, but also the correlation between features which attenuates permutation importance (see section 6 for more discussion).

I use gradient boosting regressor (GBR) for the first DGP, feedforward neural network (FFNN) for the second DGP, and random forest for the third DGP. GBR is an ensemble method like random forest (RF), but instead of building a set of trees in parallel, it builds a sequence of trees to minimize a loss function. Each next tree is fitted to the residuals of the previous tree:

$$F_m(x) = F_{m-1}(x) + \gamma h_m(x)$$

where $x$ is the feature vector, $F_{m-1}(x)$ is the prediction based on all $m-1$ fitted trees so far, $h_m(x)$ is the fitted value of the $m$-th decision tree, and $\gamma$ is the learning rate. It is a sequential process where at each step, a weak learner is trained on the residuals obtained from all previous weak learners.

FFNN is the most basic type of neural network where all the nodes are fully connected and information only flows in one direction. Suppose the first layer has $J_1$ nodes, the $J_1 \times 1$ vector $a^{(1)}$ is given by:

$$z^{(1)} = W_1^T x + b_1$$

$$a^{(1)} = \sigma(z^{(1)})$$

where $W_1$ is a $p \times J_1$ matrix of weights, $b_1$ is a $J_1 \times 1$ vector of biases (analogous to the intercept in linear regression), $\sigma(\cdot)$ is a nonlinear activation function applied element-wise to $z^{(1)}$. The vector of nodes in subsequent layers are given by the same iterative process, but with the vector from the previous layer as input. Essentially, FFNN takes the input, iteratively performs affine transformation and applies nonlinear transformation, to arrive at the output.

For GBR, I set `n_estimators` = 100, `max_depth` = 3, and `n_iter_no_change` = 10. For FFNN, I use two hidden layers with 64 and 32 neurons, `ReLU` activation, adaptive learning rate, and early stopping. For RF, I choose `n_estimators` = 100, `max_depth` = 5, and `min_samples_split` = 2. The choice of hyperparameters are arbitrary and based on heuristics. More discussions can be found in section 6.

Figure 5 presents the results. In both panels (a) and (b), as the DGPs are relatively less challenging and the base estimators are powerful, I report only the first 50 iterations. We observe that inclusion probability of the true important variables quickly converge towards 1 while the noise variables are suppressed below 0.5. In panel (a), towards iteration 50, there is one noise variable that has inclusion probability rising above 0.5 temporarily due to inherent randomness in data, and it goes below 0.5 after several more iterations.

Panel (c) plots the inclusion probabilities for the third DGP which is highly challenging and I report 500 iterations. The rate of convergences of the five true important variables are ranked from fastest to slowest as: $x_2, x_5, x_1, x_4, x_3$. Since local reward is based on permutation importance which assesses the drop in out-of-sample predictability after permuting a feature, it is notable that the eventual inclusion probabilities represent the relative influence of the five true predictor on the outcome in terms of predictive performance. $x_3$ and $x_4$ interact with each other in a $\sin(\cdot)$ function, resulting in the smallest amount of influence among all five predictors. Moreover, because $x_3$ is essentially indistinguishable from $x_4$ in the DPG, their inclusion probability trajectories are very similar. We observe that the inclusion probabilities do not converge to 0. This is because I set the threshold for local reward to be at least a $1\%$ drop in out-of-sample $R^2$. An eventual inclusion probability of $0.6$ implies that permuting the feature leads to a $1\%$ drop in out-of-sample $R^2$ $60\%$ of the times. If a lower threshold is set, the inclusion probabilities of $x_3$ and $x_4$ can converge to a higher value. However, a researcher might not be too concerned with a true important variable if its influence on outcome is extremely small. As such, the choice of threshold depends on the nature of the application and the researcher's belief of how much predictability should a feature have to be considered important.

## 5 Application to Asset Pricing

In this section, I apply bandit feature selection to the empirical asset pricing study conducted by Gu, Kelly and Xiu (2020) to understand what variables are important for driving cross-sectional asset returns. Gu, Kelly and Xiu (2020) compares the out-of-sample predictive performance of a wide range of mainstream machine learning algorithms used in both academia and industry today, including regularized regressions, tree-based methods, and neural networks, on the stock returns of over 30,000 publicly listed companies in the US between 1957 and 2016. They divide the data into training, validation and testing samples, and roll the split points forward in time iteratively to study the average predictability of each algorithm during the sample period. They find that the best performing algorithms are neural networks, gradient boosted trees and random forest, which attain an out-of-sample predictive $R^2$ of $0.34 - 0.40\%$ for monthly stock returns, and $3.09 - 3.60\%$ for annual stock returns. [3]

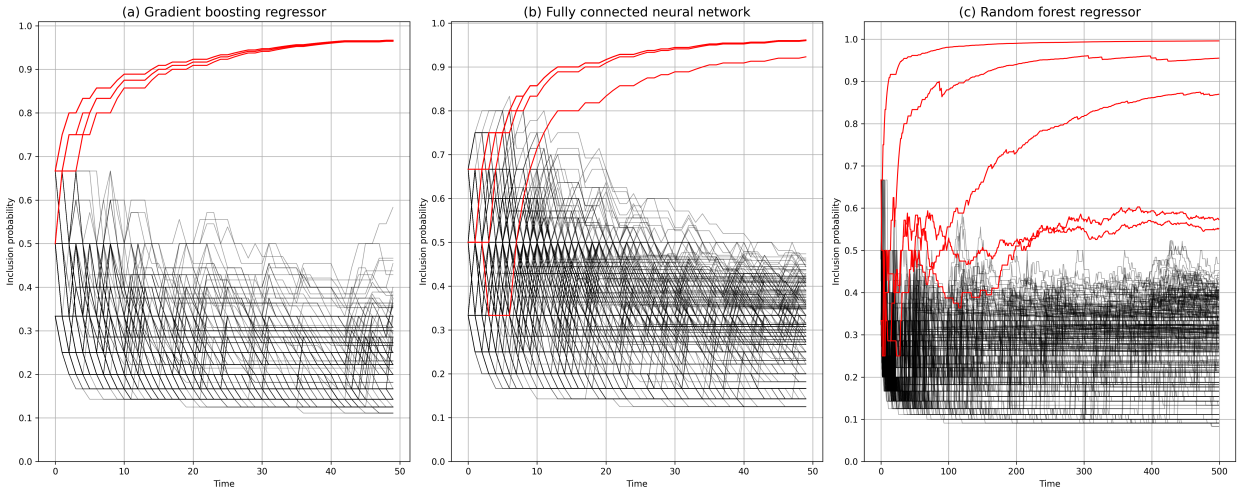


Figure 5: Inclusion probabilities over time for three different DGPs, each using a different base estimator.

Gu, Kelly and Xiu (2020) also investigate the features that are crucial for returns predictability by setting all values of a feature of interest to 0 and calculating the reduction in predictive $R^2$. Then, they average this across all training samples to obtain an overall importance measure for each feature. I aim to compare the output of bandit feature selection against their results, using gradient boosted trees as the base estimator. The reason for choosing gradient boosted trees over random forest or neural networks is primarily because gradient boosted trees are computationally much more efficient than the other two methods given the volume of data at hand. Moreover, for complicated datasets like stock returns, hyperparameter tuning plays a crucial role in the performance of neural networks which require the tuning of a myriad of parameters including the number of layers, number of nodes, learning rate, batch size, optimizer settings, and early stopping. In contrast, gradient boosted trees typically demand fewer tunable parameters, making them more practical and efficient in this application.

### 5.1 Data and method

As the number of observations is very large relative to the number of features in the full data sample, the high-dimensionality assumption that I have been assuming in the numerical experiments does not hold. As such, I run bandit feature selection in an online setting where data arrives at a monthly frequency.

In Gu, Kelly and Xiu (2020), they compile a large collection of 94 stock level features covering characteristics related to momentum, liquidity, volatility, valuation, and fundamentals, covering every month from Mar 1957 to Dec 2016. Additionally, they construct eight macroeconomic predictors following the definitions given in Welch and Goyal (2008). [33] These are the variables they use for predicting cross-sectional stock returns. All explanatory variables are lagged by 1 month at least relative to stock returns.

I download the firm level features dataset which is updated till Jun 2021 from Xiu's personal website. Since the macroeconomic predictors are updated on a monthly basis and take on the same value across all observations within a cross section, I do not include the macroeconomic predictors in my online setting. Finally, I download stock returns data from CRSP for all firms in the Gu, Kelly and Xiu (2020) dataset, and risk free rate from Kenneth French's Data Library to compute excess return.

To give an overview of my data preprocessing steps, first, following Gu, Kelly and Xiu (2020), rather than using the raw stock feature values, I rank each stock feature and map the ranks into the interval $[-1, 1]$ to facilitate the optimization of the machine learning algorithms. Second, while Gu, Kelly and Xiu (2020) generate dummy variables for the industry membership of each firm and feature interactions, I skip this step because the `HistGradientBoostingRegressor` class in `sklearn` is able to recognize the industry variable as a categorical variable and gradient boosting trees can account for the interactive effects of features by increasing model complexity without having to explicitly create interactions. Third, instead of running the full sample from 1957 to 2021, during which the significant drivers of stock returns might have changed drastically due to variations in the underlying economic dynamics and market environment, I take a subset of 10 years of the full sample from 2010 - 2019, the period after the Great Recession and before the COVID19 recession, during which drivers of cross sectional stock returns are likely to remain stable, and are more

relevant to the present day markets. Finally, even when running the dataset with online setting, every monthly cross section still has over 5000 stocks on average, but only 94 stock level features and 74 industry memberships. Hence, in each cross section, I generate 5000 noise variables from $N[0, 1]$, most values of which fall in the interval $[-1, 1]$, to artificially enforce the dimensionality to be similar to the number of observations in each monthly dataset. This gives me a dataset with over 5000 stock returns and over 5000 features arriving each month.

As returns prediction is a challenging problem and predictability is very small as seen from miniscule out-of-sample $R^2$, using an arbitrary set of hyperparameters might yield predictive $R^2$ very close to 0 or even negative $R^2$. Therefore, I use the first month i.e. Jan 2010 for hyperparameter tuning. The key parameters are learning rate, maximum number of trees, maximum number of leaf nodes per tree, maximum depth per tree, minimum number of samples required for split, and amount of L2 regularization. After obtaining the optimal hyperparameters by gridsearch and k-fold cross validation, I use the same set of hyperparameters for all subsequent months.

### 5.2 Results

I run bandit feature selection for 600 iterations, i.e. 5 iterations for each of the 120 months between 2010 and 2019, and set local reward threshold to be a $2\%$ relative drop in predictive $R^2$ when a feature is permuted. Panel (a) in figure 6 shows the change in inclusion probabilities over time, while panel (b) shows the features corresponding to the top 25 inclusion probabilities. We observe that 6 features have inclusion probabilities above 0.6 most of the times: industry momentum (indmom), bid-ask spread (baspread), 12-month momentum (mom12m), sensitivity to market factor (beta), 1-month momentum (mom1m), and idiosyncratic return volatility (idiovol). All of these variables are highly crucial cross sectional return predictors in the finance literature, and coincide with the ranking of variable importance by Gu, Kelly and Xiu (2020). They identify all six aforementioned predictors to have strong predictability in all of their machine learning algorithms (with the only exception that baspread and beta are unimportant for partial least squares and principal component regression).

Next, we observe more than 20 features to have inclusion probabilities fluctuating between 0.6 and 0.5 over time. All of these features, with the exception of cash holdings (cash) and volatility of liquidity in terms of dollar trading volume (std_dolvol), coincide with the set of features identified to have strong predictability by Gu, Kelly and Xiu (2020). Note that the ranking of feature importance in Gu, Kelly and Xiu (2020) is the average importance across all statistical models. In my application, although I only use GBR as the estimator, I manage to identify features that are deemed unimportant for GBR but important for other algorithms in Gu, Kelly and Xiu (2020), suggesting that bandit feature selection is able to learn the true important variables in the actual DGP, rather than just the important variables in the context of the model.

Panel (a) also shows the inclusion probability trajectories of the artificial noise variables, all 5000 of which are hovering between 0.1-0.4 since around 50 iterations. This indicates that bandit feature selection is highly effective in identifying the noise variables from signals and suppressing the probability of picking these variables again in future iterations.

There are two comments worthy of highlighting. First, my ranking of variable importance does not align exactly with the ranking of variable importance in Gu, Kelly and Xiu (2020). This could be because they use the full sample which is much larger than my subset of samples and it is highly likely that cross sectional drivers of stock returns have changed considerably over the decades. Second, my results are similar to Gu, Kelly and Xiu (2020) in the sense that while several predictors like indmom, mom1m, mom12m, mvel1 and retvol have posterior inclusion probabilities above 0.55, a lot more other predictors have inclusion probabilities hovering close to 0.5. This can be explained by the fact that in predicting cross sectional stock returns, there are a few strong predictors and many more mediocre but nevertheless useful predictors, each of which explains a little bit of cross sectional returns, as implied by the variable importance results of Gu, Kelly and Xiu (2020). Re-running this set-up using a more lenient threshold, e.g. assigning local reward to a $1\%$ relative drop in predictive $R^2$ when a feature is permuted, achieves a better separation of signal from noise. A lower threshold might be preferred if there is a large number of weak predictors.

## 6 Discussions

This section discusses the rationale behind certain design choices of bandit feature selection, as well as potential drawbacks and ways to overcome them.

**First**, permutation importance is unable to identify important variables when they are highly correlated. This is because when two variables are highly correlated, permuting one variable does not reduce predictive performance of the model much given the presence of the other variable. From section 4.3, we see that bandit feature selection is able to handle nonlinear DGPs involving features with correlations of 0.5. If correlation goes to a larger value like 0.8 or above, permutation importance does not work well.

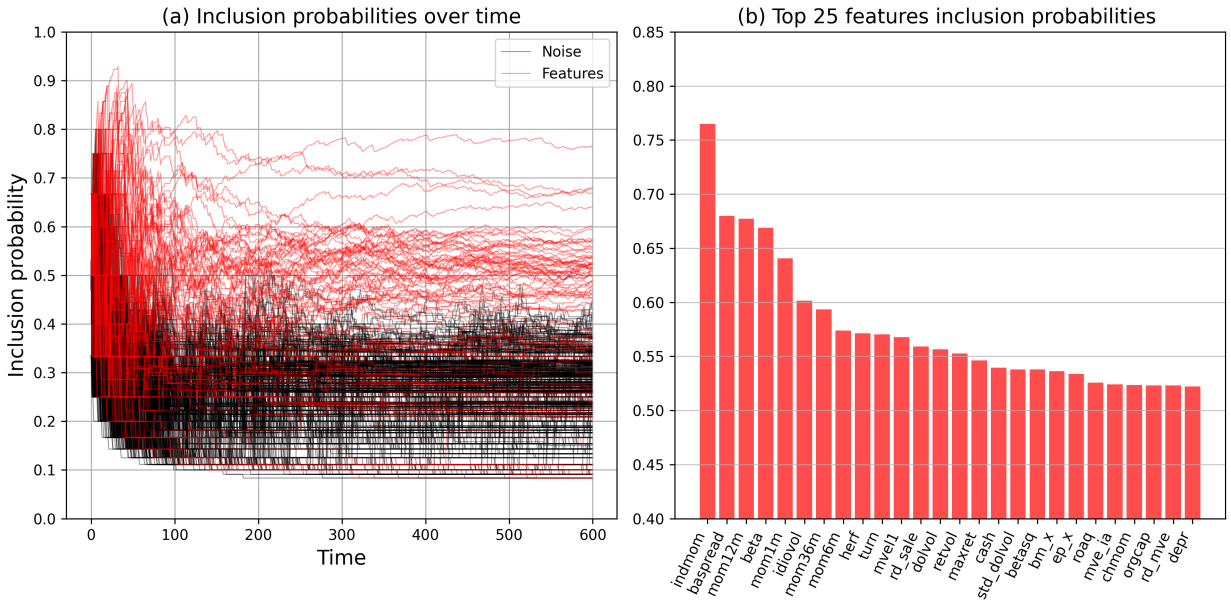


Figure 6: Left: inclusion probabilities over time. Right: Features corresponding to top 25 inclusion probabilities.

Figure 7 shows the correlation matrix of all 94 firm level features in Gu, Kelly and Xiu (2020) in my subset sample. In all $94C2 = \frac{94\times93}{2} = 4371$ pairs of variables, less than 15 pairs have absolute value of correlation coefficient greater than 0.75, and most pairs have absolute correlations well below 0.25, as such the conclusions in section 5.2 are unlikely to suffer from correlated variables. Additionally, in most of the highly correlated pairs, the features are identified to be unimportant by Gu, Kelly and Xiu (2020). The only four exceptions, i.e. highly correlated pairs of important variables, are (baspread, retvol), (baspread, idiovol), (maxret, retvol), (dolvol, mvel1). Nevertheless, they are identified to be important by bandit feature selection, most likely because in some subsets at certain time steps, only one variable in the pair is selected, enabling permutation importance to assign it positive local reward.

One solution to overcome this limitation is to bundle the highly correlated features and permute the entire bundle when assigning local reward. This requires an additional step in the bandit algorithm which constructs the correlation matrix after having selected the super-arm at time $t$, and either bundles the correlated variables as one block or performs dimensionality reduction such as $PCA$ to end up with a single variable for computing permutation importance.

**Second**, in bandit feature selection, it is possible to use bandit feedback instead of semi-bandit feedback, i.e. global reward is observed for the super-arm but base arm rewards are not observed. In Liu et al. (2021), they use classification loss to obtain the global reward $r_t$ of a super-arm, and compare it to the global reward obtained in the previous iteration $r_{t-1}$ to determine the next set of arms to be played. If $r_t > r_{t-1}$, the search for the best subset of arms is in the right direction. [17] While this framework seems appealing, the predictive performance of the model and hence its global reward depend on its hyperparameters at time $t$. This strong dependence on hyperparameter tuning not only means that the search process is computationally intensive, but the eventual result hinges heavily on how hyperparameters are chosen.

On the other hand, using permutation importance for computing local reward has good interpretation in the sense that permutation importance is measured as the change in predictive performance after randomly permuting the feature of interest, i.e. it is implicitly rewarding features that have significant influence on generalization error of the model. The set of features with the highest inclusion probabilities is also the set that contributes the most to out-of-sample predictability. This set-up aligns the bandit reward with the objective of the practitioner if he wants to obtain a model with strong generalizability. It also has a more intuitive interpretation in comparison to model-specific methods such as examining the weights associated with a feature in a neural network or how often a feature is used in decision tree splits.

**Third**, Molnar et al. (2023) formulate the theory of permutation importance by drawing connection to the underlying DGP. Recall the PI formula in section 3.3, in actual computation, $f(\cdot)$ is approximated by $\hat{f}(\cdot)$ and expectation is approximated by Monte Carlo integration. This implies that true PI based on the DGP is biased by the errors in model approximation and Monte Carlo approximation. Molnar et al. (2023) show that if the model estimator is unbiased

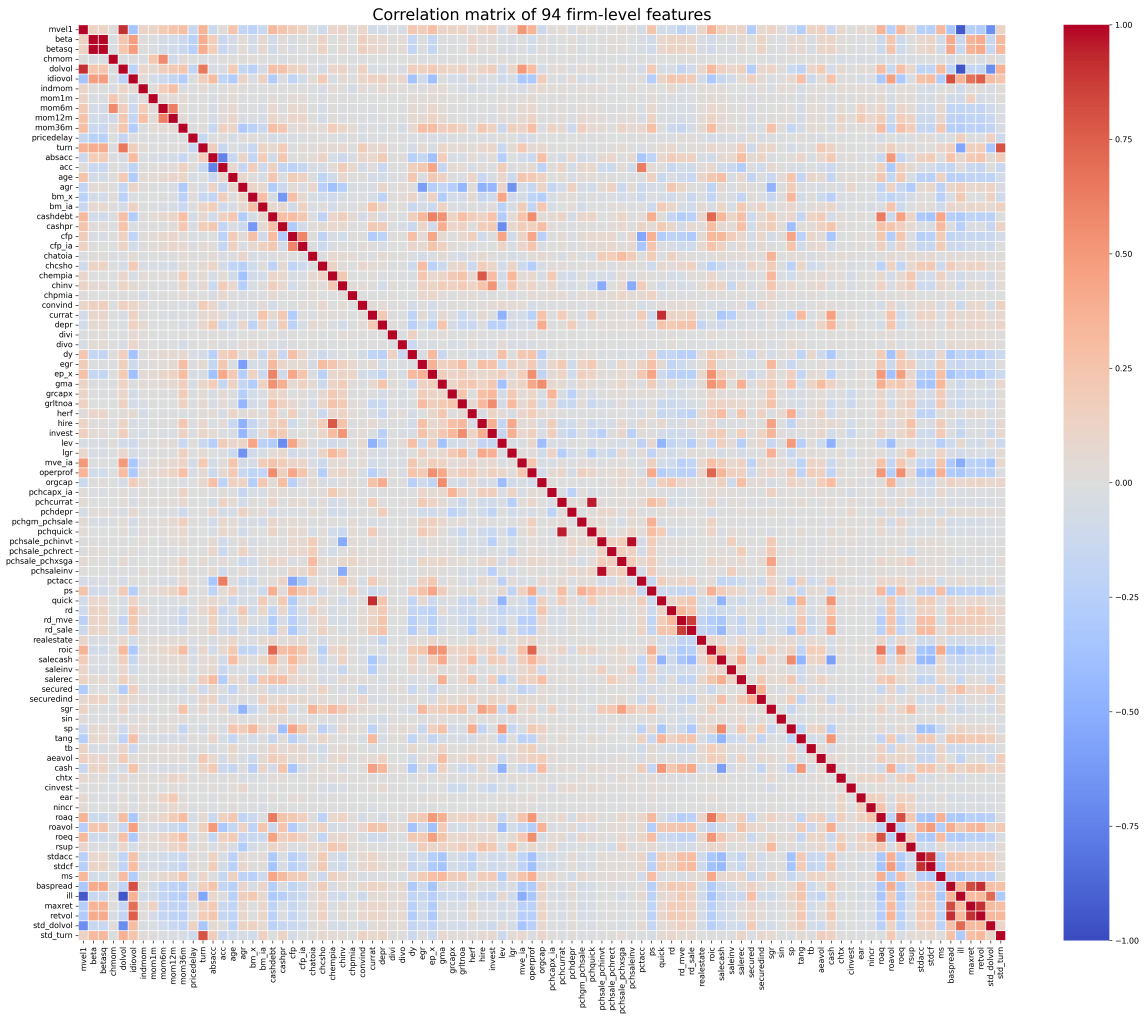


Figure 7: Correlation matrix of 94 firm level features in the empirical application

i.e. $E(\hat{f}) = f$, then sample PI is an unbiased estimator of population PI. Moreover, they show that the gap between population PI and sample PI is a function of the expected variance of $f$ and covariance of $f$ and $\hat{f}$. [26] In light of this, they propose a sample PI computed from averaging multiple model refits and corrected confidence intervals.

Bandit feature selection has a similar flavor as the refitting PI estimator of Molnar et al. (2023). In the offline setting, it computes PI using bootstrapped data and different sets of features at every time step to iteratively correct the sample PI which is affected by bias and data randomness, to arrive at a more accurate importance estimate eventually.

The theorems in Molnar et al. (2023) might offer theoretical backing to why using a base estimator with a mis-specified functional form (e.g. using LASSO to learn nonlinear additive DGP in section 4.2) enables feature importance to be learned, and why hyperparameter tuning at every time step is not necessary for learning feature importance. Since the gap between sample PI and population PI is bounded by the covariance between $f$ ad $\hat{f}$, when this quantity is small, sample PI does not deviate too far from population PI. Moreover, for nonparametric machine learning algorithms, specifying a more complex functional form tends to increase variance and reduce bias As such, when specifying hyperparameters of a nonparametric base estimator, it is recommended to go for a moderately complex model to reduce the bias of sample PI, and the point estimate can be made more accurate via iterative learning.

Intuitively, when hyperparameters are not optimal and the estimator overfits to training data, because PI is evaluated on out-of-sample data, noise variables do not impact out-of-sample predictability anyways while true important variables

will lead to a change in predictability, albeit small. If computation is not a major constraint, it would also be possible to conduct hyperparameter tuning at every time step within the bandit feature selection algorithm.

**Four**, one drawback of bandit feature selection is that it does not have a good convergence criterion, i.e. when to stop the algorithm when running on real-world dataset. One potential solution is to monitor the inclusion probability graph at fixed time intervals, and stop when the practitioner observes stability or a clear separation in the inclusion probabilities. In my Python implementation, I define a stopping condition i.e. the ranking of all variables with inclusion probabilities above 0.5 remains unchanged for $K$ iterations, where $K$ is set to be 50 as default.

**Lastly**, there are several points to note when implementing bandit feature selection in the real world. First, bandit feature selection requires the user to specify a threshold $\nu$ for the drop in out-of-sample metrics after permuting a feature, for obtaining local rewards. The choice is at the discretion of the user i.e. how much he believes that a true important feature should impact out-of-sample predictability. A lower threshold would lead to higher inclusion probabilities for the variables that play a role in the true DGP, and vice versa. Note that even if the threshold is very lenient, noise variables are extremely unlikely to have inclusion probability above 0.5. It is often the case that randomly permuting a noise variable leads to increase in out-of-sample predictability, hence earning a negative PI. In my experiments and applications, I have used $\nu$ ranging from $1\%$ to $5\%$ drop in predictive performance. When the number of true important variables is large and each has a small influence on outcome, it is recommended to set a more lenient $\nu$. Second, because bandit feature selection involves bootstrap and row permutation, when using it on time series data, it is essential to preserve the temporal order of dataset when fitting estimators and computing PI to avoid look-ahead bias.

## 7 Conclusions

In this research, I develop a model agnostic framework for learning variable importance and selecting models in both online and offline settings. The model is robust to the true DGP and the choice of base estimator. In comparison to model specific methods, it attains a higher accuracy of picking out the true important variables from noise, and enjoys greater computational efficiency by learning from a subset of a large number of features at every time step.

However, it remains to be rigorously proven that top-two Thompson sampling for combinatorial bandits indeed attains a faster rate of convergence than Thompson sampling. Additionally, one can study the theoretical properties of permutation importance and its link to the true DGP, and characterize the asymptotic properties of bandit feature selection in terms of its ability to learn the population permutation importance.

## References


[1] Audrey Durand and Christian Gagné. Thompson sampling for combinatorial bandits and its application to online feature selection. *AAAI Workshop: Sequential Decision-Making with Big Data*, 2014.

[2] Yi Liu and Veronika Rockova. Variable selection via thompson sampling. *Journal of the American Statistical Association*, 00(0):1–18, 2021.

[3] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33:2223–2273, 2020.

[4] Romaric Gaudel and Michele Sebag. Feature selection as a one-player game. *International Conference on Machine Learning*, pages 359–366, 2010.

[5] Sali Rasoul, Sodiq Adewole, and Alphonse Akakpo. Feature selection using reinforcement learning. *arXiv preprint arXiv:2101.09460*, 2021.

[6] Mohammad-Hassan Zokaei Ashtiani, Majid Nili Ahmadabadi, and Babak Nadjar Araabi. Bandit-based local feature subset selection. *Neurocomputing*, 138:371–382, 2014.

[7] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *JMLR: Workshop and Conference Proceedings*, 23:39.1–39.26, 2012.

[8] Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. *Proceedings of the 35th International Conference on Machine Learning, PMLR*, 80, 2018.

[9] Daniel Russo. Simple bayesian algorithms for best arm identification. *29th Annual Conference on Learning Theory, PMLR*, 49, 2016.

[10] Sebastien Bubeck, Remi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. *International Conference on Algorithmic Learning Theory*, pages 23–47, 2009.

[11] Sebastien Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple identifications in multi-armed bandits. *Proceedings of the 30th International Conference on Ma- chine Learning, JMLR*, 28, 2013.

[12] Aurelien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. *JMLR: Workshop and Conference Proceedings*, 49:1–30, 2016.

[13] Jiecao Chen, Xi Chen, Qin Zhang, and Yuan Zhou. Adaptive multiple-arm identification. *Proceedings of the 34th International Conference on Machine Learning, PMLR*, 70, 2017.

[14] Peter Glynn and Sandeep Juneja. Selecting the best system and multi-armed bandits. *arXiv preprint arXiv:1507.04564*, 2018.

[15] Maximilian Kasy and Anja Sautmann. Adaptive treatment assignment in experiments for policy choice. *Econometrica*, 89(1):113–132, 2021.

[16] A. Stefano Caria, Grant Gordon, Maximilian Kasy, Simon Quinn, Soha Shami, and Alexander Teytelboym. An adaptive targeted field experiment: Job search assistance for refugees in jordan. *Journal of the European Economic Association*, 2023.

[17] Kunpeng Liu, Haibo Huang, Wei Zhang, Ahmad Hariri, Yanjie Fu, and Kien Hua. Multi-armed bandit based feature selection. *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 316–323, 2021.

[18] Siwei Wang and Jun Zhu. Thompson sampling for (combinatorial) pure exploration. *Proceedings of the 39th International Conference on Machine Learning, PMLR*, 162, 2022.

[19] Shintaro Nakamura and Masashi Sugiyama. Thompson sampling for real-valued combinatorial pure exploration of multi-armed bandit. *The Thirty-Eighth AAAI Conference on Artificial Intelligence*, 2024.

[20] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine1. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[21] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2.28:307–317, 1953.

[22] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *31st Conference on Neural Information Processing Systems*, 2017.

[23] Benedek Rozemberczki, Lauren Watson, Peter Bayer, Hao-Tsung Yang, Oliver Kiss, Nilsson Sebatstian, and Sarkar Rik. The shapley value in machine learning. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022.

[24] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[25] Andre Altmann, Laura Tolosi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *BIOINFORMATICS*, 26(10):1340–1347, 2010.

[26] Christoph Molnar, Timo Freiesleben, Gunnar Konig, Julia Herbinger, Tim Reisinger, Guiseppe Casalicchio, Marvin N. Wright, and Bernd Bischl. Relating the partial dependence plot and permutation feature importance to the data generating process. *Explanable Artificial Intelligence CCIS 1901*, pages 456–479, 2023.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[28] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[29] T.L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *ADVANCES IN APPLJED MATHEMATTCS*, 6:4–22, 1985.

[30] Maria Maddalena Barbieri and James O. Berger. Optimal predictive model selection. *The Annals of Statistics*, 32(3):870–897, 2004.

[31] Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

[32] Faming Liang, Qizhai Li, and Lei Zhou. Bayesian neural networks for selection of drug sensitive genes. *Journal of the American Statistical Association*, pages 955–972, 2018.

[33] Ivo Welch and Amit Goyal. A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, pages 1455–1508, 2008.