mathematics

Article

Ranking the Importance of Variables in a Nonparametric Frontier Analysis Using Unsupervised Machine Learning Techniques

Raul Moragues [1,2], Juan Aparicio [1,3,*] and Miriam Esteve [1]

1 Center of Operations Research (CIO), Miguel Hernandez University of Elche (UMH), 03202 Elche, Spain; rmoragues@umh.es (R.M.)
2 Ph.D. Program in Economics (DEcIDE), Miguel Hernandez University of Elche (UMH), 03202 Elche, Spain
3 Joint Research Unit, Valencian Graduate School and Research Network of Artificial Intelligence (valgrAI), 46022 Valencia, Spain
* Correspondence: j.aparicio@umh.es

Abstract: In this paper, we propose and compare new methodologies for ranking the importance of variables in productive processes via an adaptation of OneClass Support Vector Machines. In particular, we adapt two methodologies inspired by the machine learning literature: one involving the random shuffling of values of a variable and another one using the objective value of the dual formulation of the model. Additionally, we motivate the use of these type of algorithms in the production context and compare their performance via a computational experiment. We observe that the methodology based on shuffling the values of a variable outperforms the methodology based on the dual formulation. We observe that the shuffling-based methodology correctly ranks the variables in 94% of the scenarios with one relevant input and one irrelevant input. Moreover, it correctly ranks each variable in at least 65% of replications of a scenario with three relevant inputs and one irrelevant input.

Keywords: data envelopment analysis; feature ranking; model specification; unsupervised machine learning; technical efficiency; overfitting

MSC: 90C08

check for updates

Citation: Moragues, R.; Aparicio, J.; Esteve, M. Ranking the Importance of Variables in a Nonparametric Frontier Analysis Using Unsupervised Machine Learning Techniques. *Mathematics* **2023**, *11*, 2590. https://doi.org/10.3390/math11112590

Academic Editor: Andrea Scozzari

Received: 17 May 2023
Revised: 31 May 2023
Accepted: 1 June 2023
Published: 5 June 2023

1. Introduction

A topic which has attracted large amounts of interest from the machine learning and statistical communities is the importance which certain variables have when building models to predict or explain a response variable. This can be encountered in a large variety of fields, such as the measurement of the technical efficiency of a set of homogeneous entities (companies, public organizations, etc.), related to microeconomics and operations research. Some of the early contributions in this area of study can be traced back to the work by Cobb and Douglas [1], who empirically estimated a production function. Later, Koopmans proposed a formal definition of technical efficiency [2], and Debreu and Farrell introduced a way to measure it, following an input-oriented or output-oriented radial direction in [3,4], respectively. A link between the measures of efficiency and production technologies was introduced by Shephard in [5]. Building on these foundations, a variety of approaches have been proposed, which are usually split in the literature into parametric and nonparametric methodologies. Representative examples of each approach, which are two of the most well-known techniques, are the Data Envelopment Analysis (DEA) in the nonparametric family [6,7], and the Stochastic Frontier Analysis (SFA) in the parametric one [8,9].

In this article, we focus on the nonparametric approach due to some of its characteristics, such as its flexibility and its natural multi-input, multi-output treatment. Whereas the parametric approach assumes some functional form for the production frontier, the nonparametric approach takes, as a foundation, only some properties of the underlying production frontier. In particular, the DEA methodology proposes a linear optimization program which can be used to estimate the technical efficiency of a unit with respect to a production technology which satisfies the postulates of envelopment, free disposability of inputs and outputs, and convexity. It does so by applying, at the last stage, the postulate of minimal extrapolation, which estimates the smallest among all possible sets satisfying the above postulates [7].

The postulate of minimal extrapolation is the cause of one of the criticisms that has been leveraged against DEA, which is that it is a data-driven technique which is descriptive in nature, and thus may not generalize well [10]. In particular, it does not allow for statistical inference tasks to be performed unless the sample size is large enough. Various authors have attempted to overcome this limitation. Among them, Simar and Wilson have adapted bootstrapping procedures to estimate bias, variance, and to construct confidence intervals [11,12]. Other properties studied include the consistency and speed of convergence of the DEA estimators [13], which is deeply related to the problem of the curse of dimensionality, which results in too many units being considered efficient when the number of dimensions is large, relative to the number of units available. Some recent contributions to the DEA literature are [14,15].

In this paper, we turn our attention to a field related to that of operations research and optimization, which is the field of machine learning and data analytics. This is an area of knowledge which builds estimators from available data. These estimators can be broadly classified into two families: supervised and unsupervised learning. In supervised learning, some variables are used in order to estimate one or multiple objective variables. Depending on the nature of the predicted variable(s), supervised learning tasks can be a regression of a continuous variable or a classification of elements into various classes when the target variable is discrete. On the other hand, in unsupervised learning, all variables are used in order to obtain information about the process which generated the data, which includes tasks such as clustering, anomaly detection, or estimating probability densities and their supports.

Until relatively recently, there had been little contact between the fields of machine learning and the measurement of technical efficiency fields, but some examples of their proximity can be seen in the work by Kuosmanen and Johnson, who used piecewise linear estimators of a production function via the Corrected Concave Nonparametric Least Squares [16]. Another contribution by Parmeter introduced nonparametric kernel estimators to the frontier problem [17], while Daouia et al. proposed a procedure using constrained polynomial splines to obtain smooth frontiers [18]. Other authors have adapted decision tree-based techniques such as Classification and Regression Trees (CART) in [10], or probabilistic regression trees (with panel data) in [19]. Furthermore, Valero-Carreras et al. adapted Support Vector Regression in this context [20], while Olesen and Ruggiero proposed a representation of production frontiers using hinging hyperplanes [21]. Finally, Guerrero et al. combined DEA with machine learning techniques through the Structural Risk Minimization principle [22].

One feature that the works mentioned above have in common (except DEA itself) is that they are methods which use some of the variables available in the data (inputs) in order to predict the values of one (or more) output variable(s). This is usually associated with the supervised learning paradigm in machine learning. In the production context, this is justified by the natural split into inputs and outputs present, where the inputs are used to predict or explain the values of the output(s). However, these methods have drawbacks, such as the requirement to a priori partition the variables into two subsets, and they present difficulties when being extended to multi-output contexts [23].

In contrast, unsupervised learning makes no such distinction between variables. Instead, it treats all variables homogeneously, and attempts to obtain information about the underlying Data Generating Process (DGP) that yielded the data. In this context, the estimation objective of DEA becomes to estimate the production technology, which can be seen as the support of the underlying Data Generating Process [24] and, from this point of view, DEA resembles an unsupervised learning technique more closely. These observations enable the use of methodologies for the estimation of the support of a distribution in order to estimate production technologies.

Among the methods for estimating the support of a probability distribution, a relevant family is that of kernel methods, such as the Kernel Density Estimation [25,26]. Kernels are transformations of the data which endow estimators with flexibility. They can also provide smoothing to estimators of probability densities of random variables, and their corresponding supports. At the intersection of machine learning methodology and the statistical learning theory, there lies a family of kernel-based estimators called Support Vector Machines (SVM) [27,28]. First introduced by Vapnik, SVMs adapt the flexibility of kernel methods to varied tasks such as classification and regression in the supervised learning area. Furthermore, SVMs can be adapted to unsupervised machine learning tasks such as the estimation of support of high-dimensional distributions via, for example, the OneClass Support Vector Machines (OCSVM) estimator of [29]. We choose this method as the basis for our estimator of the production technology.

An important topic in data-driven methodologies is the phenomenon of the decrease in the quality of the model as the dimensionality of the data increases when compared to the number of data points available [30]. This problem is related to the rate of convergence, which depends on the number of units as well as on the dimensionality of the problem. In a nonparametric frontier analysis, this phenomenon is often called the curse of dimensionality and takes form in an increase in the number of Decision Making Units (DMUs) considered efficient, and the subsequent lack of discrimination between them. It is also related to the question of model specification, and an important task in this context is measuring the importance of each variable in the production process, which allows the ranking of the variables according to this importance.

Approaches to the evaluation of the importance of variables and their selection from the DEA literature include proposals based on regressions between input variables and efficiency scores [31], or partial correlations among variables [32], as well as methods evaluating the contribution of each variable to the estimated efficiency scores [33]. Statistical hypothesis tests have been proposed to evaluate the significance of input variables [34], as well as comparisons between the number of efficient units estimated by various models [35]. We refer the reader to a more thorough discussion and comparison of these and other techniques in [36]. Other contributions propose methods which evaluate the importance of subsets of variables, such as those that enrich the optimization programs using binary variables to model the inclusion or exclusion of variables [37–39]. Along these lines, criteria such as Akaike's Information Criteria [40] or game-theoretic measures such as the Shapley value [41] have been used to choose among models. In addition to the selection among the original variables, other authors have proposed methods for the aggregation of variables into new variables, such as those which use the Principal Component Analysis [42,43], or techniques based on bootstrapping [12]. Some other approaches to attempt to increase the discriminating power of DEA involve super-efficiency models [44], which omit the unit whose efficiency is being evaluated from the reference set of the technology, or the use of the distance to anti-efficient frontiers [45], among others.

More recently, there are contributions for the selection of variables such as [46], who provide a more recent overview of methods as well as a methodology using contribution loads; methodologies based on statistical tests such as [47,48]; as well as methods which enrich other estimators such as SCNLS [16] with LASSO-based regularization terms [49–51].

From the machine learning perspective (without contact with the technical efficiency measurement field), many varied approaches have been proposed for the ranking of importance of variables and their selection; see for example [52–54]. In this paper, we will focus on two of the most well-known methods. One involves the random shuffling of features, first introduced with Random Forest in [55]; and the other is an SVM-specific method which measures the importance of variables via their effect on the objective value of the dual formulation of the estimator. It was introduced in [56].

In summary, in this paper, we propose an adaptation of the OneClass Support Vector Machine algorithm to the estimation of production technologies which generalize those of DEA, aiming to overcome the deterministic and overfitting nature of DEA. We furthermore endow it with methods for the measurement of the importance of each variable in the production process, as well as obtaining a ranking of these variables. We propose a feature shuffling method and an approach based on the objective function of the dual formulation of the model. This paper presents a new link between the ranking of the importance of variables in efficiency measurement and machine learning. Furthermore, this paper proposes, for the first time, the use of unsupervised machine learning methodologies to rank the importance of variables in production processes.

The rest of this article is structured as follows. Section 2 introduces the main concepts of the Data Envelopment Analysis, OneClass Support Vector Machines, and describes the feature importance methods which we will adapt. Section 3 proposes an adaptation of the OneClassSVM estimator to the nonparametric frontier estimation context and equips it with the proposed approaches for ranking the importance of features. Section 4 describes and presents the results of a computational experiment performed to compare these methods. Finally, Section 5 presents the conclusions of this article, as well as an outline of potential future research lines.

2. Background

In this section, we briefly introduce the main notions used in the article related to the Data Envelopment Analysis, OneClass Support Vector Machines, and the feature ranking methods which we will adapt.

2.1. Data Envelopment Analysis

The Data Envelopment Analysis (DEA) is a nonparametric methodology for the estimation of the technical efficiency of members of a set of Decision Making Units (DMUs) in a multi-input, multi-output context. It uses a dataset consisting of n DMUs, where DMU i consumes $\boldsymbol{x}_i = (x(1), \ldots, x(m)) \in \mathbb{R}^m_+$ inputs in order to produce $\boldsymbol{y}_i = (y(1), \ldots, y(s)) \in \mathbb{R}^s_+$ outputs, and estimates the relative efficiency of each DMU with respect to an underlying production technology satisfying certain microeconomic axioms (we represent vectors using bold characters, while scalars are not bold).

Whereas the standard formulation in DEA assumes non-negative values for both inputs and outputs, yielding different objectives for each type of variable, these variables can be considered with the direction of flow to and from the unit under study [2]. This difference in flow direction is formalized in the netput notation, which considers inputs to have non-positive values, while the values of output variables are non-negative, i.e., the netput representation of DMU i is $\boldsymbol{z}_i = (-\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^m_- \times \mathbb{R}^s_+$ [57,58]. We denote the netput dataset by $\mathcal{Z}$. With this notation, increasing efficiency of a DMU corresponds to maximizing each component (without reducing any other component). Furthermore, the axiom of free disposability is simplified, since now any variable can have its values freely reduced while remaining feasible. The *netput* notation in the production context allows for a homogeneous treatment of all variables. The efficiency of each DMU is evaluated with respect to a technology or production possibility set, which is defined by those netput bundles whose outputs are producible given a set of inputs.

$$\Psi = \{\boldsymbol{z} \in \mathbb{R}^m_- \times \mathbb{R}^s_+ : \boldsymbol{z} \text{ is feasible}\}.$$

We will denote by Ψ the unknown underlying technology, and work with estimations of this technology, which we denote by $\hat{\Psi}$. In the case of the DEA estimator, the technology $\hat{\Psi}_{\text{DEA}}$ is the smallest set satisfying: the envelopment of the data, that is, $\hat{\Psi}_{\text{DEA}}$ contains all DMUs in $\mathcal{Z}$; convexity, that is, $\hat{\Psi}_{\text{DEA}}$ is a convex set; and free disposability of inputs and outputs, that is, for every $\boldsymbol{z} \in \hat{\Psi}_{\text{DEA}}$, whenever $\boldsymbol{z}' \leq \boldsymbol{z}$, we have $\boldsymbol{z}' \in \hat{\Psi}_{\text{DEA}}$ [7]. In particular, the DEA estimated technology $\hat{\Psi}_{\text{DEA}}$ is, following the minimal extrapolation principle, the unique set satisfying these properties.

Given a technology Ψ, a usual characterization of technically efficient units is as those points which are not strictly Pareto dominated by any other element of Ψ, i.e., there is no feasible DMU with strictly greater netput values. The corresponding set of interest is the *weak efficient frontier* of Ψ, which consists of the (weak) technically efficient points in Ψ. It is defined by $\partial^W(\Psi) := \{\boldsymbol{z} \in \Psi : \hat{\boldsymbol{z}} > \boldsymbol{z} \Rightarrow \hat{\boldsymbol{z}} \notin \Psi\}$. The technical efficiency of a DMU can then be measured as the distance to $\partial^W(\Psi)$ with respect to some permitted improvement path. One such measure of efficiency is the *Directional Distance Function* (DDF), introduced in [57,59]. The DDF projects each DMU along some pre-specified direction $\boldsymbol{g} \in \mathbb{R}^{m+s}_{+}$ such that $\boldsymbol{g} \neq \boldsymbol{0}$, as follows:

$$\delta(\boldsymbol{z}, \boldsymbol{g}) = \max\{\delta : (\boldsymbol{z} + \delta \boldsymbol{g}) \in \Psi\}. \tag{1}$$

Thus, $\delta(\boldsymbol{z}, \boldsymbol{g})$ measures the distance from $\boldsymbol{z}$ to the boundary of a given technology Ψ along a direction $\boldsymbol{g}$. The DDF takes the value 0 for efficient DMUs (any change along the $\boldsymbol{g}$ direction leaves the technology) and, whenever $\boldsymbol{z} \in \Psi$, we have $\delta(\boldsymbol{z}, \boldsymbol{g}) \geq 0$. Therefore, $\delta \in [0, +\infty)$, and it can be seen as a measure of technical inefficiency of a given DMU $\boldsymbol{z}$.

A particular choice of vector $\boldsymbol{g}(\boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{y}) = (0, \ldots, 0, z(m+1), \ldots, z(m+s)) \in \mathbb{R}^{m+s}_{+}$ yields the output-oriented Farrell measure of efficiency, which is defined as the largest possible proportional increase in all outputs simultaneously, which results in a feasible bundle [4]:

$$\lambda(\boldsymbol{z}) = \max\{\lambda : (-\boldsymbol{x}, \lambda \boldsymbol{y}) \in \Psi\}. \tag{2}$$

The coincidence of both measures can be seen explicitly via the relationship $\lambda = \delta - 1$, since $\boldsymbol{z} + \delta \boldsymbol{g} = (-\boldsymbol{x}, \boldsymbol{y}) + \delta(\boldsymbol{0}, \boldsymbol{y}) = (-\boldsymbol{x}, (1+\delta)\boldsymbol{y})$.

DEA is a very interesting approach to determine technical efficiency, partly due to being based on few assumptions. However, the estimates obtained using this technique are very dependent on the variables utilized [30] and suffer from overfitting to the data due to the minimal extrapolation principle [10]. We introduce a technique which aims to overcome these limitations and use it to measure the relative importance of variables (inputs and outputs).

2.2. OneClass Support Vector Machines

We now turn our attention to the OneClass Support Vector Machine (OCSVM) estimator [29], in machine learning, which we will adapt to measure technical efficiency. This estimator belongs to the family of Support Vector Machines (SVM) [27,28], which are based on the statistical learning theory, with the aim to bound not only the classification error, but also the generalization error via a weighing hyperparameter $C = 1/\nu n$. The OCSVM estimates the region in which this data lives, separating it from the rest of the space via a hyperplane in a transformed space via a transformation function $\boldsymbol{\phi}$. In other words, it estimates the support of the underlying Data Generating Process. The OCSVM estimator is based on the following quadratic program:

$$\begin{array}{lll} \displaystyle\min_{\boldsymbol{w} \in \mathbb{R}^{m+s+h}, \boldsymbol{\xi} \in \mathbb{R}^n, \rho \in \mathbb{R}} & \displaystyle\frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho & \\ \text{subject to} & \langle \boldsymbol{w} \cdot \boldsymbol{\phi}(z_i) \rangle \geq \rho - \xi_i, & \forall i \in \{1, \ldots, n\} \\ & \xi_i \geq 0, & \forall i \in \{1, \ldots, n\} \end{array}. \tag{3}$$

This program involves a regularization error term $\frac{1}{2}\|\boldsymbol{w}\|^2$, an empirical error term involving one ξ_i for each DMU, and a scalar value ρ indicating where the frontier is located. The tradeoff between the various terms is weighted by a hyperparameter ν. When solving this program, we obtain $(\boldsymbol{w}^*, \boldsymbol{\xi}^*, \rho^*)$, which determine the estimated support of the dataset, defined by:

$$\hat{\Psi}_{\text{OCSVM}} = \{\boldsymbol{z} \in \mathbb{R}^{m+s} : \langle \boldsymbol{w}^* \cdot \boldsymbol{\phi}(\boldsymbol{z}) \rangle \geq \rho^*\}. \tag{4}$$

This estimator can obtain sets with various properties according to the choice of kernel or transformation function $\boldsymbol{\phi}(\boldsymbol{z})$. This kernel maps the feature space into a higher dimensional space and separates the data in this transformed space, giving more flexibility to the method. By parallelism with the DEA-estimated technology, we choose for this task the following adaptation of the PieceWise Linear (PWL) transformation function described in ([60], Expression (12)), which will result in polyhedral sets being estimated:

$$\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}) = \begin{cases} -z(k), & \text{for } k \in \{1, \dots, m+s\} \\ -\max\{\mu, \langle \boldsymbol{p}_k \cdot \boldsymbol{z} \rangle + q_k\}, & \text{for } k \in \{m+s+1, \dots, m+s+h\} \end{cases}. \tag{5}$$

Figure 1 illustrates the role played by the PWL transformation function (5) in the proposed approach in an example with two variables. In this example, four hyperplanes are defined, each of which splits the variable space into two half-spaces. These hyperplanes are called basis functions in [60]. Each component of the transformation function achieves the value $-\mu$ in one such half-space, and the value $-\langle \boldsymbol{p}_k \cdot \boldsymbol{z} \rangle - q_k$ elsewhere. Thus, the region where each hyperplane is equal to hyperparameter μ causes the corresponding component of $\boldsymbol{\phi}_{PWL}$ to change the function chosen in the maximum, which results in a turning point of a potential estimated boundary of $\hat{\Psi}_{\mathrm{OCSVM}}$, weighted by the corresponding component of $\boldsymbol{w}$. Thus, we can observe that the resulting boundary consists of a piecewise linear region with turning points at each of the defined hyperplanes. Given a particular transformation function, the OCSVM algorithm will estimate values of $\boldsymbol{w}$ so that the corresponding region is an estimate of the support of the observed data. A restriction on the values of $\boldsymbol{w}$ will enable the estimation of a convex set.

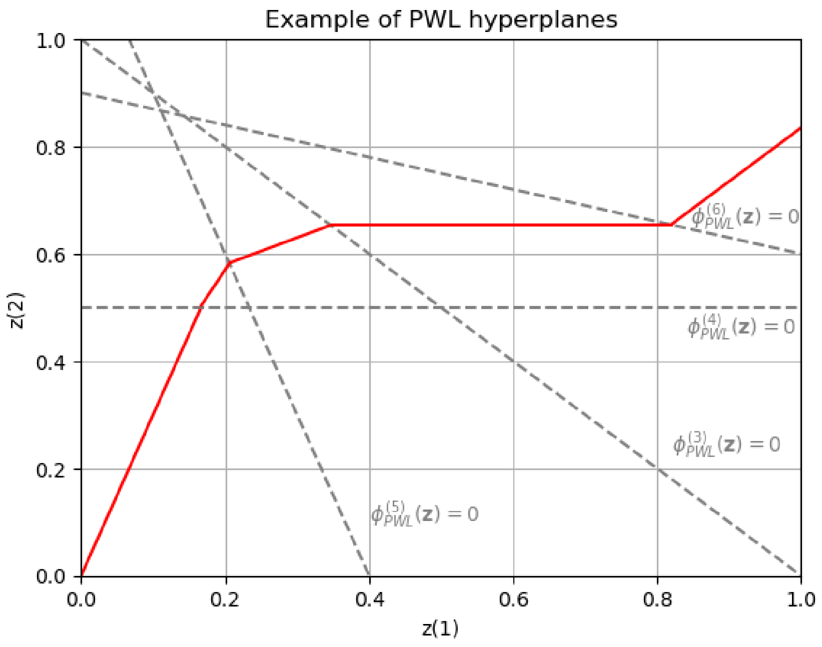

Figure 1. Example of hyperplanes and of a classification boundary.

This feature mapping involves an offset hyperparameter μ, as well as a number h of hyperplanes determined by their coefficients $\boldsymbol{p}_k$ and $q_k \in \mathbb{R}$. The regions where each hyperplane takes the value μ are those regions of the netput space where the boundary of the support can have turning points.

2.3. Feature Ranking in Machine Learning

We now turn our attention to two of the large variety of methods available in the machine learning literature to measure and rank the importance of variables. We choose two of the most well-known methods which are appropriate to the OneClass Support Vector Machine estimator. These are methods based on the random shuffling of the values of a variable, introduced in [55], as well as an approach based on the change in the objective value of the dual problem to (8), as proposed in [56] specifically for Support Vector Machines.

2.3.1. Feature Shuffling

Feature Shuffling, also known as Permutation Feature Importance, is one of the classical methods for measuring the relative importance of features for a given estimator in machine learning. It is based on the increase of the error of a model when the values of a feature are randomly shuffled. This methodology was introduced together with the Random Forest algorithm [55].

The idea behind this algorithm is that, for some estimators, the magnitude of the error values may depend on the number of features used in the estimator, so that if two models are compared where a feature has been completely dropped from one of them, there may be effects from the changes in both the number of features used (related to the rate of convergence), and the effect of the feature itself which we aim to measure. Instead, randomly shuffling the values of the variable being evaluated makes this variable, on average, unrelated to the rest of features. The effect is then similar to dropping the variable from the estimator, while keeping the number of features used in the estimator unchanged.

Based on this observation, a backward stepwise procedure is proposed which, at each step, obtains a measure of the relative importance of the features for the current estimator, and ranks the least important one last among the remaining features. At each step, this method fits the appropriate estimator to the data with every remaining variable, and obtains the Mean Squared Error (MSE) of the estimator, E_0, that is, the error of the model. It then iterates over each remaining variable l one at a time, and shuffles randomly the values of this variable, obtaining a dataset $\mathcal{Z}_l$. Then, the estimator is fitted again with the shuffled data, yielding an estimator error of E_l.

The shuffling of values of a variable that are very important to the production process should result in a large increase of values from E_0 to E_l, whereas an irrelevant variable being shuffled would have little to no effect on the error of the estimator. Thus, we calculate the normalized difference:

$$\Delta E_l = \frac{E_l - E_0}{E_0}. \tag{6}$$

The variable l is considered more important the larger the ΔE_l is. Thus, at each step, the variable attaining the minimum value of ΔE_l is considered the least important, and it is ranked last among the remaining variables. The process is then iterated with those variables not yet added to the ranking until the last variable remaining is considered the most important one. This yields a complete ranking of the features under consideration.

2.3.2. Dual Objective Variation

This SVM-inspired approach, introduced in [56], is based on the variation in the objective value of the dual formulation of a Support Vector Machine when a variable is dropped. It considers the dual formulation of an SVM, as calculated using standard quadratic programming tools. The use of the objective value J of the dual as a measure of the importance of a variable is justified since this objective value can be seen as a measure of the error of the model.

Thus, we can evaluate the importance of each variable via the effect that removing this variable has on the objective value J of the dual formulation of an SVM. The methodology begins by solving the Support Vector Machine (in its dual formulation) and obtaining its objective value J. This quantity involves the dual variables, which are then fixed in order to simplify the computation. Then, the values of the variable being considered are converted to 0, simulating the effect of its removal, and the model is solved once again, obtaining a new value J_l. Then, the normalized squared difference ΔJ_l^2 is calculated:

$$\Delta J_l^2 = \left(\frac{J_l - J}{J}\right)^2. \tag{7}$$

This value ΔJ_l^2 is the corresponding measure of importance of the variable l. If the evaluated variable l is irrelevant to the production process, its elimination will cause a small increase in the error of the model, that is, it will yield a small value of ΔJ_l^2, whereas the removal of an important variable will cause the error to increase, yielding larger values of ΔJ_l^2.

At each step, the variable with a minimum ΔJ_l^2 is considered as the least important for the estimator, and it is ranked last among the remaining variables. The method is then iterated until the final variable remaining is considered the most important variable in the process.

3. New Methods for Ranking Variables in Production Processes Using OneClass Support Vector Machines for Efficiency Measurement

In this Section, we adapt the OCSVM algorithm introduced in Section 2 with the piecewise linear kernel (5) to the task of estimating production technologies via appropriate modifications to satisfy convexity and other relevant microeconomic properties. We, furthermore, propose two approaches for ranking the importance of the variables involved in the production process.

As the basis of the proposal, we follow the approach from [24] that a technology arises from an underlying Data Generating Process (DGP) which is assumed to have some statistical properties. These assumptions are that the observed DMUs are random samples of identically and independently distributed random variables with an underlying probability density function satisfying some regularity conditions. In this context, the problem of estimating a production technology has a natural interpretation, such as the task of estimating the support of a probability distribution, which enables the use of tools from that literature, such as OCSVM.

We begin by adapting the OneClass Support Vector Machine estimator (3) to the task of estimating a production technology as follows. The resulting model, which we call OneClass for Efficiency Measurement (OCEM) is the following:

$$\min_{\boldsymbol{w}\in\mathbb{R}^{m+s+h},\ \boldsymbol{\xi}\in\mathbb{R}^n,\ \rho\in\mathbb{R}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \tag{8}$$

$$\begin{aligned} \text{subject to } & \langle \boldsymbol{w}\cdot\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i)\rangle \geq \rho - \xi_i, && \forall i \in\{1,\dots,n\} && \text{(8a)}\\ & \xi_i \geq 0, && \forall i \in\{1,\dots,n\} && \text{(8b)}\\ & w_j \geq 0, && \forall j \in\{1,\dots,m+s+h\} && \text{(8c)}\\ & \langle \boldsymbol{w}\cdot\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{0})\rangle = \rho, && && \text{(8d)} \end{aligned}$$

Model (8) is a quadratic program, where the objective function and restrictions (8a) and (8b) are identical to those of the OCSVM model (3). We use the PWL transformation function $\boldsymbol{\phi}_{\mathrm{PWL}}$ from (5), with $\boldsymbol{p_k} \geq \boldsymbol{0}$ for all hyperplanes. The algorithm involves two hyperparameters: ν and μ, which we will now characterize. These will be fine-tuned via a train-test split in order to obtain the best ones for each dataset. Restriction (8c) will ensure convexity of the estimated technology, defined by:

$$\hat{\Psi}(\mu,\nu) := \{\boldsymbol{z} \in \mathbb{R}_-^m \times \mathbb{R}_+^s : \langle \boldsymbol{w}^* \cdot \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z})\rangle \geq \rho^*\}.$$

The corresponding efficient frontier is defined by the boundary of the polyhedral technology, that is, $F(\hat{\Psi}(\mu,\nu)) := \{\boldsymbol{z} \in \mathbb{R}_-^m \times \mathbb{R}_+^s : \langle \boldsymbol{w}^* \cdot \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z})\rangle = \rho^*\}$. Thus, restriction (8d) will guarantee that the efficient frontier will pass through the origin, that is, $\boldsymbol{0} \in \hat{\Psi}(\mu,\nu)$. We calculate the technical inefficiency of a DMU with respect to this technology by adapting the DDF formulation (1), with $\boldsymbol{g} \in \mathbb{R}_+^{m+s}$:

$$\delta_{\mathrm{OCEM}}(\boldsymbol{z},\boldsymbol{g}) = \max\{\delta \in \mathbb{R} : (\boldsymbol{z}+\delta\boldsymbol{g}) \in \hat{\Psi}(\mu,\nu)\} = \max\{\delta : \langle \boldsymbol{w}^* \cdot \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}+\delta\boldsymbol{g})\rangle \geq \rho^*\}. \tag{9}$$

With this setup, it can be proved that the estimated technology $\hat{\Psi}(\mu,\nu)$ satisfies the usual microeconomic axioms of production technologies. The convexity of $\hat{\Psi}(\mu,\nu)$ follows as in ([60], section 5), given that the defined $\boldsymbol{\phi}_{\text{PWL}}$ is concaved and $\boldsymbol{w} \geq \mathbf{0}$. Free disposability of inputs and outputs is satisfied as in CNLS (see ([16], section 2.2)) when imposing the condition that $\boldsymbol{p}_k \geq \mathbf{0}$ for all hyperplanes, so we will determine hyperplanes satisfying this property. As a consequence of the properties of the OCSVM algorithm ([29], Proposition 3), we obtain a bound on the fraction of outliers n_{OL} in terms of the hyperparameter ν, given by

$$n_{OL} \leq \nu n \leq n_{SV}.$$

Here, n_{SV} is the number of Support Vectors, that is, those DMUs which are important to determine the frontier. Therefore, we observe that, if $\nu < 1/n$, then $\hat{\Psi}_{\text{DEA}} \subseteq \hat{\Psi}(\mu,\nu)$, as a consequence of the principle of minimal extrapolation in DEA. In other words, the technology estimated by DEA is, when ν is small enough, a subset of the estimated technology.

The property that ν is a lower bound for the fraction of Support Vectors allowed, and an upper bound for the fraction of outliers, leads us to choose ν in the range $[1/(n+1), 0.1]$, except when $n \leq 10$, where we choose $\nu \in [0.1, 0.3]$. This choice results in a minimum of 0 outliers, and a maximum of 10% of DMUs being outliers.

We now describe the role of the hyperplanes involved in the transformation function (5), which is closely related to the appropriate range of values of the hyperparameter μ. The hyperplane coefficients $\boldsymbol{p}_k \geq \mathbf{0}$ and $q_k \in \mathbb{R}$ involved in the model parameterize a set of hyperplanes which will allow the polyhedral frontier turning points, that is, where the edges of the faces of the polyhedral technology will be located. Since the goal is to estimate an efficient frontier which is close to the data but without overfitting, thus being close to the theoretical frontier, we are interested in hyperplanes which lie in the region enveloping the data from above. A known set of hyperplanes in this region is given by the faces of the convex closure, which we remark can be estimated by the DEA methodology. Hence, we obtain a set of hyperplanes by solving the following linear DEA problem with directional function $\boldsymbol{g} = \mathbf{1}$ corresponding to the Chebyshev norm l_∞ (see [61]). Using the netput notation, the linear program to solve for DMU $\boldsymbol{z}_i$ is:

$$\begin{array}{rlll} -\mu_i = & \min\limits_{\boldsymbol{p}_k, q_k} & -\langle \boldsymbol{p}_k \cdot \boldsymbol{z}_i \rangle - q_k & \\ & \text{subject to} & \langle \boldsymbol{p}_k \cdot \boldsymbol{z}_r \rangle + q_k \leq 0, & \forall r \in \{1,\ldots,n\} \\ & & \langle \boldsymbol{p}_k \cdot \boldsymbol{g} \rangle = 1, & \\ & & \boldsymbol{p}_k \geq \mathbf{0}, & \end{array} . \tag{10}$$

By solving these n linear programs, we obtain, for each DMU $\boldsymbol{z}_i$, a corresponding hyperplane defined by $(\boldsymbol{p}_k, q_k)$ which is a hyperplane at distance $-\mu_i$ from this DMU located on the DEA-estimated efficient frontier along the direction $\boldsymbol{g} = \mathbf{1}$. Thus, we choose $h = n$. Regarding μ, it is a hyperparameter which offsets the defined hyperplanes simultaneously, so we choose the value $\mu_{min} = \min\{\mu_i : i \in \{1,\ldots,n\}\}$, i.e., the negative of the largest distance from a DMU to the efficient frontier, as a lower bound for potential values of μ, yielding a potential range of values for $\mu \in [\mu_{min}, 0)$. We remark that, by the first constraint of (10), the objective value of (10) (i.e., $-\mu_i$) is non-negative, hence, $\mu_i \leq 0$ for each DMU. We assume that there is at least one DMU which is strictly inefficient with respect to model (10), which implies that there is some DMU $\boldsymbol{z}_i$ such that the objective value of program (10) is strictly positive, hence $\mu_i < 0$. Under this hypothesis, we have that $\mu_{min} < 0$.

At this stage, we have all the information required to set up the quadratic program (8) which will be solved in order to obtain $(\boldsymbol{w}^*, \boldsymbol{\xi}^*, \rho^*)$. It remains to tune the hyperparameters $\nu \in [1/(n+1), 0.1]$ when $(n \geq 10$ and $\mu \in [\mu_{min}, 0)$. For this task, we choose five equally spaced values for each variable from these intervals. We then split the data into train and test sets, where we use 70% of the data as a training set $\mathcal{Z}_{train}$ and the remaining 30% as a test set $\mathcal{Z}_{test}$ in order to evaluate the fit of each model trained. We denote their respective cardinalities by n_{train} and n_{test}. Then, for each pair of hyperparameter values, we fit model (8) with the train set in order to obtain an estimate of the technology $\hat{\Psi}(\mu,\nu)$. In order to choose among these candidate technologies, we evaluate them using their Mean Squared Error on the test set as follows. We evaluate model (9) using Farrell's output distance, that is, with $\boldsymbol{g}(\boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{y}) = (0, \ldots, 0, z(m+1), \ldots, z(m+s))$, in order to obtain the efficiency $\delta(\boldsymbol{z}, \boldsymbol{g})$ of DMU $\boldsymbol{z}$, and its projection $\hat{\boldsymbol{z}} = \boldsymbol{z} + \delta(\boldsymbol{z}, \boldsymbol{g})\boldsymbol{g}$ to the estimated efficient frontier. We then calculate the MSE between the observed values and the estimated output values of each DMU in the test set, via

$$\frac{1}{n_{test}} \sum_{z \in \mathcal{Z}_{test}} \sum_{k \in \{m+1, m+s\}} (\hat{z}(k) - z(k))^2.$$

We choose those hyperparameters μ^*, ν^* which lead to the smallest Mean Squared Error on the predictions on the test set. With these hyperparameters fixed, we again fit model (8) using the whole dataset $\mathcal{Z}$ to obtain the final estimate of the technology $\hat{\Psi} := \hat{\Psi}(\mu^*, \nu^*)$.

The algorithm described above involves the tuning of appropriate values for hyperparameters μ^*, ν^*. However, for computational reasons, we sometimes already have appropriate valid hyperparameters from a previous estimate of the technology, which we fix for comparison.

Since Program (8) is quadratic, we can use the standard tools of quadratic programming to obtain the following dual formulation, with hyperparameters μ^*, ν^*. We remark that the dual of Program (8), which is a minimization problem, is a maximization problem. However, the objective function of the maximization problem is the negative of the presented one, and is thus equivalent to the following minimization problem (this is standard in the SVM literature, see e.g., ([29], eq. (3.11))):

$$\begin{array}{lll} \min\limits_{\boldsymbol{\alpha}, \boldsymbol{\gamma}, \alpha_0} & J = \dfrac{1}{2}\left\| \sum\limits_{i=1}^{n} \alpha_i \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i) + \boldsymbol{\gamma} + \alpha_0 \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{0}) \right\|^2 & \\ \text{subject to} & 0 \leq \alpha_i \leq 1/\nu^* n, & \text{for } i \in \{1, \ldots, n\} \\ & \sum\limits_{i=1}^{n} \alpha_i + \alpha_0 = 1, & \\ & \boldsymbol{\gamma} \geq \boldsymbol{0}, & \end{array} \quad . \tag{11}$$

We remark that, in terms of the variables of the primal program (8), we have the equality $\boldsymbol{w} = \sum\limits_{i=1}^{n} \alpha_i \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i) + \boldsymbol{\gamma} + \alpha_0 \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{0})$, so that the objective value $J = \frac{1}{2}\|\boldsymbol{w}\|^2$ is a measure of the error of the model.

We now adapt the previously described methods for ranking the relative importance of the variables to the context of ranking the variables involved in productive processes using the OCEM algorithm.

We first describe the Shuffling-OCEM methodology involving the random shuffling of variable values, before moving to the Dual-OCEM proposal, which is based on the variation of the objective function of the dual program (11).

3.1. Shuffling-OCEM

We proceed to adapt the methodology involving the shuffling of values of a feature to the OCEM context as follows. We first solve the full OCEM model with the original dataset $\mathcal{Z}$, with the tuning of hyperparameters. This yields values for μ^*, ν^*, which we use to obtain the MSE of the fitted estimator, denoted by E_0. We then iterate over each variable l being tested for inclusion, randomly shuffle its values to obtain dataset $\mathcal{Z}_l$, and solve the OCEM model with the previously established hyperparameters μ^*, ν^* and dataset $\mathcal{Z}_l$. We calculate the error of this model, E_l, and calculate the importance measure ΔE_l of the variable l using (6).

We remark here that we keep the hyperparameters μ^*, ν^* obtained in the first model fixed for the shuffled models, since we are interested in evaluating the effect of the change in each candidate variable on the estimator. Furthermore, we performed some preliminary testing, where we observed that the version with further hyperparameter tuning takes longer computational time, around 5 times longer than without it, but does not yield better results. Thus, we consider the version where the hyperparameter tuning procedure is only performed on the full model at each step of the process, and solve the shuffled models with the hyperparameters obtained from the full model.

Once the measure of importance ΔE_l has been calculated for every variable, the least important variable is that with the smallest value of ΔE_l. We add this variable to the current ranking, and iterate the method without this variable, in order to continue ranking the rest of the variables. At each iteration of the method, the hyperparameters μ^*, ν^* are recomputed. The final variable, that is, the variable which is never considered as the least important among those remaining, is then considered the most important variable in the production process. Algorithm 1 shows the steps followed by Shuffling-OCEM.

Algorithm 1 Shuffling-OCEM algorithm implementation

```
procedure CALCULATE_RANKING_SHUFFLING_OCEM($\mathcal{Z}$,variables_to_rank)
    remaining_variables ← variables_to_rank
    ranking, $\Delta E$ ← [ ]
    while |remaining_variables| > 1 do
        n_var = |remaining_variables|
        for $k$ ← 1 to $n$ do
            $\boldsymbol{p}_k, q_k, \mu_i$ ← solve_program_(10)
        end for
        $\mu_{min} = \min_i\{\mu_i\}$
        $\mathcal{Z}_{train}, \mathcal{Z}_{test}$ ← create_train_test($\mathcal{Z}$)
        $\hat{\Psi}, \boldsymbol{w}^*, \boldsymbol{\xi}^*, \rho^*, \mu^*, \nu^*$ ← full_OCEM($\mathcal{Z}, \boldsymbol{p}_k, q_k, \mu_{min}$)
        $E_0$ ← calculate_MSE_OCEM($\mathcal{Z}, \hat{\Psi}$)
        for $l$ ← 1 to $n\_var$ do
            $\mathcal{Z}_l$ ← permute_var($\mathcal{Z}, l$)
            $\hat{\Psi}_l, \boldsymbol{w}_l^*, \boldsymbol{\xi}_l^*, \rho_l^*$ ← OCEM_no_crossvalidation($\mathcal{Z}_l, \boldsymbol{p}_k, q_k, \mu_{min}, \mu^*, \nu^*$)
            $E_l$ ← calculate_MSC_OCEM($\mathcal{Z}_l, \hat{\Psi}_l$)
            $\Delta E_l$ ← $\frac{E_l - E_0}{E_0}$
        end for
        $v$ ← $arg\,min\{\Delta E\}$
        ranking ← add_variable_to_ranking($v$)
        remaining_variables ← remaining_variables − $v$
        $\mathcal{Z}$ ← remove_variable($\mathcal{Z}, v$)
    end while
    final_variable ← remaining_variables
    ranking ← add_variable_to_ranking(final_variable)
    return ranking
end procedure
```

3.2. Dual-OCEM

This method evaluates the effect that the omission of a variable has on the model via the variation in the objective value J of the dual problem (11) of the OCEM model. The idea is that the removal of a variable will result in a larger effect on J the more important the removed variable was on the model. In order to obtain values for the parameters involved in the dual program, we begin by setting up and solving the full OCEM primal problem (8), obtaining optimal hyperparameters μ^*, ν^* for this problem, as well as a set of h hyperplane parameters $\boldsymbol{p}_k$, q_k involved in the feature mapping $\boldsymbol{\phi}_{\mathrm{PWL}}$. We then fix these hyperparameters throughout this iteration and turn our attention to program (11), the dual model to the OCEM method.

We solve the dual program (11) and then, in order to make the computations feasible, [56] proposes that we keep the solutions $(\boldsymbol{\alpha}^*, \gamma^*, \alpha_0^*)$ of the dual constant. In order to evaluate the change in the objective function J when removing a variable l, since we keep the hyperparameters and the dual variables constant, the only changes in J will come from the effect of removing feature l in the transformation function $\boldsymbol{\phi}_{\mathrm{PWL}}$. We denote this transformation, applied to DMU $\boldsymbol{z}_i$ by $\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i(-l))$, and we calculate it by eliminating all contributions of variable l to $\boldsymbol{\phi}_{\mathrm{PWL}}$, i.e., by setting $z(l) = 0$ whenever it appears. That is,

$$\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}(-l)) = \begin{cases} -z(k), & \text{for } k \in \{1, \dots, m+s\},\ k \neq l \\ 0, & \text{for } k = l \\ -\max\{\mu^*, \sum\limits_{j=1, j\neq l}^{m+s} p_k(j) z(j) + q_k\}, & \text{for } k \in \{m+s+1, \dots, m+s+h\} \end{cases}. \tag{12}$$

We remark that this has no effect on the vector $\mathbf{0}$, i.e., $\boldsymbol{\phi}_{\mathrm{PWL}}(\mathbf{0}) = \boldsymbol{\phi}_{\mathrm{PWL}}(\mathbf{0}(-l))$. The change in the objective function J of the dual problem (11) when removing variable l is then:

$$\begin{aligned} DJ_l = J_l - J =& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^* \alpha_j^* \Big[\langle \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i(-l)) \cdot \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_j(-l)) \rangle - \langle \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i) \cdot \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_j) \rangle \Big] \\ &+ \sum_{i=1}^{n} \alpha_i^* \langle [\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i(-l)) - \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i)] \cdot \gamma^* \rangle \\ &+ \sum_{i=1}^{n} \alpha_i^* \alpha_0^* \langle [\boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i(-l)) - \boldsymbol{\phi}_{\mathrm{PWL}}(\boldsymbol{z}_i)] \cdot \boldsymbol{\phi}_{\mathrm{PWL}}(\mathbf{0}) \rangle. \end{aligned} \tag{13}$$

We further remark that, in Equation (13), the terms involving the DMUs with $\alpha_i^* = 0$ vanish, so that we only need to take into account those DMUs $\boldsymbol{z}_i$ with $\alpha_i^* \neq 0$ (i.e., the Support Vectors) in order to calculate DJ_l. Therefore, we only need to consider a subset of the data, which may be smaller than the original dataset. We then calculate DJ_l as l runs over every variable that remains to be ranked. The measure of importance of variable l used in this method is (7):

$$\Delta J_l^2 = \left(\frac{J_l - J}{J} \right)^2 = \left(\frac{DJ_l}{J} \right)^2.$$

At each step, we calculate ΔJ_l^2 for each variable l remaining. The variable that is considered the least important is the variable l which attains the minimum value of ΔJ_l^2, so we add this variable to the ranking as the next least important variable. We then iterate the method without this variable being considered in order to continue ranking the rest of the variables. At each iteration, we recalculate the hyperparameters, hyperplane parameters, and dual variables. Finally, the last variable remaining is considered the most important variable for the production process. The steps of Dual-OCEM are shown in Algorithm 2.

Algorithm 2 Dual-OCEM algorithm implementation

```
procedure CALCULATE_RANKING_DUAL_OCEM($\mathcal{Z}$,variables_to_rank)
    remaining_variables $\leftarrow$ variables_to_rank
    ranking, $\Delta J^2 \leftarrow$ [ ]
    while |remaining_variables| $> 1$ do
        n_var = |remaining_variables|
        for $k \leftarrow 1$ to $n$ do
            $\boldsymbol{p}_k, q_k, \mu_i \leftarrow$ solve_program_(10)
        end for
        $\mu_{min} = \min_i\{\mu_i\}$
        $\mathcal{Z}_{train}, \mathcal{Z}_{test} \leftarrow$ create_train_test($\mathcal{Z}$)
        $\hat{\Psi}, \boldsymbol{w}^*, \boldsymbol{\xi}^*, \rho^*, \mu^*, \nu^* \leftarrow$ full_OCEM($\mathcal{Z}, \boldsymbol{p}_k, q_k, \mu_{min}$)
        $J, \boldsymbol{\alpha}^*, \gamma^*, \alpha_0^* \leftarrow$ solve_dual_OCEM_(11)($\mathcal{Z}$)
        for $l \leftarrow 1$ to $n\_var$ do
            $\boldsymbol{\phi}_{PWL}(-l) \leftarrow$ calculate_updated_transformation_(12)
            $DJ_l \leftarrow$ calculate_using_(13)
            $\Delta J_l^2 \leftarrow \left(\frac{DJ_l}{J}\right)^2$
        end for
        $v \leftarrow arg\,min\{\Delta J^2\}$
        ranking $\leftarrow$ add_variable_to_ranking($v$)
        remaining_variables $\leftarrow$ remaining_variables $- v$
        $\mathcal{Z} \leftarrow$ remove_variable($\mathcal{Z}, v$)
    end while
    final_variable $\leftarrow$ remaining_variables
    ranking $\leftarrow$ add_variable_to_ranking(final_variable)
    return ranking
end procedure
```

4. Computational Experience

In order to evaluate and compare the performance of the proposed algorithms, we use scenarios inspired by [62], with Cobb-Douglas production functions with Variable Returns to Scale with multiple inputs and a single output. With these production functions, the magnitude of the exponent of each input represents its level of theoretical marginal importance, with higher values indicating a more important variable. An irrelevant variable can be considered to have exponent 0. Furthermore, the sum of the exponents of the inputs is associated with the returns to scale of the production process. When the sum of the exponents is less than one, the corresponding production process exhibits non-increasing returns to scale, whereas a sum greater than one is associated to non-decreasing returns to scale. Consequently, a sum of one is associated with constant returns to scale. In this computational experience, we consider functions whose exponents add up to less than one, thus exhibiting non-increasing returns to scale. Other returns to scale could be considered, but this extension is beyond the scope of this paper. In particular, we consider scenarios with 1, 3, or 5 relevant inputs $\boldsymbol{x}$, one additional irrelevant input a, and one output y, in order to test how each methodology ranks the variables by their importance. We investigate the effect on the tests of the specification of the following factors: (1) sample size, (2) inefficiency distribution, (3) median inefficiency level, and (4) production function.

To calculate the output level of DMU j given inputs $\boldsymbol{x}_j$, we use a production function $f(\boldsymbol{x}_j) = q_j$ and an inefficiency term θ_j, so that $y_j = q_j/\theta_j = f(\boldsymbol{x}_j)/\theta_j$. The production functions $q_j = f(\boldsymbol{x}_j)$ which we simulate are Cobb-Douglas functions, with a variety of parameters and different exponents in each of the relevant variables, are presented below. Each function represents the maximum producible output given input profile $\boldsymbol{x}_j$ of relevant variables in a scenario. The true inefficiency term θ_j is defined, as in [62,63], using $\theta_j = 1 + \psi_j$, where ψ_j is a non-negative random variable taking a variety of probability distributions. We, furthermore, include an additional input variable a_j which is not involved in the production function. Thus, a_j is irrelevant to the production process. The values for all the inputs $\boldsymbol{x}_j = (x_j(1), x_j(2), \ldots, x_j(m-1))$ and a_j of DMU j are generated independently from $Uni[5, 15]$. The production functions used are:

- In the case of one relevant input ($m = 2$):

$$f(\boldsymbol{x}) = 10x(1)^{\gamma}, \tag{14}$$

  where γ takes values between 0.1 and 0.9 by 0.1.
- The technologies with three relevant inputs are ($m = 4$):

$$f(\boldsymbol{x}) = 10x(1)^{0.45}x(2)^{0.35}x(3)^{0.1}, \tag{15a}$$

$$f(\boldsymbol{x}) = 10x(1)^{0.5}x(2)^{0.25}x(3)^{0.05}, \tag{15b}$$

$$f(\boldsymbol{x}) = 10x(1)^{0.4}x(2)^{0.2}x(3)^{0.1}, \tag{15c}$$

$$f(\boldsymbol{x}) = 10x(1)^{0.3}x(2)^{0.2}x(3)^{0.15}. \tag{15d}$$

- In the case of five relevant inputs ($m = 6$):

$$f(\boldsymbol{x}) = 10x(1)^{0.3}x(2)^{0.2}x(3)^{0.15}x(4)^{0.1}x(5)^{0.05}, \tag{16a}$$

$$f(\boldsymbol{x}) = 10x(1)^{0.4}x(2)^{0.2}x(3)^{0.15}x(4)^{0.1}x(5)^{0.05}, \tag{16b}$$

$$f(\boldsymbol{x}) = 10x(1)^{0.25}x(2)^{0.2}x(3)^{0.15}x(4)^{0.1}x(5)^{0.05}, \tag{16c}$$

$$f(\boldsymbol{x}) = 10x(1)^{0.35}x(2)^{0.3}x(3)^{0.15}x(4)^{0.1}x(5)^{0.05}. \tag{16d}$$

With each production function, we considered sample sizes of 50 and 100 DMUs, each of them with two different probability distributions for the inefficiency terms (half-normal and exponential), and in each case, in both a low and a high level of median inefficiency, yielding a total of 136 different scenarios, each of which was replicated 100 times, for a total of 13,600 datasets. Regarding the inefficiency term ψ_j, we consider the following configurations. For the half-normal distribution, the low inefficiency setting considers $\psi_j \sim |N(0, 0.4)|$, whereas for the high inefficiency setting, we choose $\psi_j \sim |N(0, 0.8)|$. Similarly, for the exponential distribution, we consider $\psi_j \sim E(0.4)$ and $\psi_j \sim E(0.8)$, respectively, for the low and high inefficiency settings. With these parameters, the median values for the inefficiency θ_j are approximately 1.27 and 1.54, respectively.

With these scenarios, we execute the Shuffling-OCEM and Dual-OCEM algorithms in order to investigate whether they are able to adequately detect the relative importance of each variable, obtaining a full ranking of the input variables, which should be ordered by the values of the exponents, with higher values indicating a higher importance. The position of the irrelevant variable in the ranking should be last, as it could be considered to have exponent 0 in the Cobb-Douglas functions below.

We first consider the aggregated results of the overall experiments in Table 1. In all subsequent tables, we present on the left-hand side the results for the Shuffling-OCEM methodology and those for the Dual-OCEM on the right. Furthermore, we split them according to the number of input variables due to the different nature of each ranking table. Overall, we observe that Shuffling-OCEM outperforms the Dual-OCEM methodology at this task.

Table 1. Aggregated ranking results of the shuffling and dual methodologies according to number of variables.

	Shuffling-OCEM						**Dual-OCEM**					
Overall Results												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	2%	2%	3%	7%	23%	63%	7%	7%	10%	15%	25%	36%
$x(2)$	5%	6%	9%	19%	39%	22%	12%	12%	13%	18%	24%	22%
$x(3)$	10%	13%	20%	32%	19%	8%	17%	16%	17%	19%	17%	14%
$x(4)$	17%	20%	28%	20%	10%	4%	19%	18%	20%	17%	13%	12%
$x(5)$	27%	30%	22%	13%	6%	2%	20%	23%	20%	16%	12%	9%
a	39%	29%	18%	9%	4%	1%	25%	24%	20%	14%	10%	8%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	1%	3%	16%	80%	6%	11%	27%	56%		
		$x(2)$	6%	15%	64%	15%	12%	21%	42%	25%		
		$x(3)$	27%	54%	15%	4%	31%	39%	19%	12%		
		a	66%	28%	5%	1%	52%	29%	12%	7%		
					2	**1**	**2**	**1**				
				$x(1)$	6%	94%	18%	82%				
				a	94%	6%	82%	18%				

From the results, when we consider the effects of comparisons according to various factors, we observe the following patterns:

(1) We begin with the effects of sample size. In Table 2, we can observe that both methods improve as the number of DMUs increases from 50 to 100 DMUs, and that the Shuffling-OCEM methodology with 50 DMUs already outperforms the Dual-OCEM methodology with 100 DMUs.

(2) We now study the effects of changing the inefficiency distribution. Table 3 reports the aggregated percentages for the scenarios with exponential and half-normal distributions separately. We can observe a slightly better performance for both methods with a half-normal distribution than with an exponential one, but the difference is very small. In fact, these two tables are almost exactly the same as the overall results in Table 1. Thus, we can conclude that both methods are robust to the inefficiency distribution.

(3) Next, we compare the performance in the scenarios with low and high average inefficiency levels. We can observe in Table 4 that both methods perform worse when the average inefficiency level is high. Furthermore, we observe that the Shuffling-OCEM with a high average inefficiency still performs better than the Dual-OCEM even with a low average inefficiency.

Furthermore, we consider the separate effects of low and high average inefficiency levels with each of the two inefficiency distributions, and, as in the inefficiency distribution comparison (2), we observe very similar values when changing the inefficiency distribution. Hence, we can conclude again that the type of inefficiency distribution does not affect the performance in either the low or high inefficiency. We do not report these tables, as their values are very similar to those already presented. They are available upon request.

(4) Finally, we consider the results obtained with each of the production functions used separately. The results for scenarios (16a)–(16d), i.e., those production functions with $m = 6$ can be observed in Table 5, while those corresponding to scenarios (15a) and (15d), that is, with $m = 4$, are presented in Table 6. The results from scenarios with $m = 2$ according to the value of γ, that is, with production functions (14), are summarized in Figure 2. We can observe that the values of the exponents highly affect the quality of the rankings, where the larger the difference between two consecutive exponents, the smaller the confusion between them. As before, Shuffling-OCEM clearly outperforms Dual-OCEM throughout. We now focus on the Shuffling-OCEM behavior according to the values of the exponents.

Table 2. Results depending on the sample size.

	Shuffling-OCEM						**Dual-OCEM**					
50 DMUs												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	3%	3%	4%	9%	24%	57%	10%	9%	9%	14%	24%	35%
$x(2)$	6%	8%	11%	19%	33%	23%	15%	14%	14%	17%	22%	19%
$x(3)$	12%	14%	20%	27%	18%	9%	20%	18%	16%	17%	16%	14%
$x(4)$	19%	20%	24%	20%	12%	6%	19%	18%	19%	18%	14%	13%
$x(5)$	26%	27%	22%	14%	7%	4%	18%	21%	21%	17%	13%	11%
a	34%	28%	19%	12%	6%	2%	19%	21%	22%	17%	12%	9%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	2%	4%	19%	75%	7%	13%	28%	52%		
		$x(2)$	7%	16%	58%	18%	14%	21%	39%	26%		
		$x(3)$	28%	50%	16%	5%	32%	36%	18%	13%		
		a	62%	29%	7%	1%	47%	30%	15%	8%		
					2	**1**	**2**	**1**				
				$x(1)$	7%	93%	20%	80%				
				a	93%	7%	80%	20%				
100 DMUs												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	1%	1%	2%	6%	21%	70%	4%	5%	10%	17%	26%	38%
$x(2)$	3%	4%	7%	19%	45%	21%	9%	9%	12%	19%	26%	25%
$x(3)$	7%	11%	19%	37%	19%	6%	14%	15%	18%	21%	17%	15%
$x(4)$	14%	21%	33%	21%	9%	2%	19%	19%	21%	17%	13%	11%
$x(5)$	29%	34%	22%	11%	4%	1%	23%	25%	20%	15%	11%	7%
a	46%	29%	16%	6%	2%	0%	31%	26%	18%	11%	7%	6%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	0%	1%	13%	86%	4%	10%	26%	60%		
		$x(2)$	4%	13%	71%	12%	10%	21%	45%	25%		
		$x(3)$	26%	58%	13%	2%	29%	41%	19%	11%		
		a	69%	27%	3%	0%	57%	28%	10%	5%		
					2	**1**	**2**	**1**				
				$x(1)$	5%	95%	17%	83%				
				a	95%	5%	83%	17%				

Table 3. Results depending on the inefficiency distribution.

	Shuffling-OCEM						**Dual-OCEM**					
Exponential Inefficiency Distribution												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	2%	2%	3%	8%	23%	61%	8%	7%	10%	15%	25%	35%
$x(2)$	5%	6%	10%	19%	39%	22%	13%	11%	13%	18%	23%	22%
$x(3)$	10%	12%	20%	31%	18%	8%	18%	16%	17%	19%	16%	14%
$x(4)$	16%	21%	28%	21%	10%	4%	19%	18%	19%	17%	14%	12%
$x(5)$	28%	30%	21%	12%	6%	3%	20%	23%	20%	17%	11%	9%
a	38%	29%	18%	9%	4%	1%	23%	24%	21%	14%	10%	8%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	1%	3%	16%	79%	7%	12%	27%	54%		
		$x(2)$	6%	15%	63%	16%	12%	23%	40%	25%		
		$x(3)$	29%	52%	15%	4%	31%	36%	20%	13%		
		a	64%	29%	6%	1%	50%	29%	14%	8%		
					2	**1**	**2**	**1**				
				$x(1)$	6%	94%	17%	83%				
				a	94%	6%	83%	17%				

Table 3. *Cont.*

	Shuffling-OCEM						**Dual-OCEM**					
Half-Normal Inefficiency Distribution												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	2%	2%	3%	7%	23%	63%	6%	7%	9%	16%	24%	37%
$x(2)$	5%	7%	9%	19%	38%	23%	12%	12%	13%	18%	24%	22%
$x(3)$	10%	13%	20%	31%	19%	7%	17%	17%	17%	19%	17%	14%
$x(4)$	18%	20%	27%	20%	10%	4%	20%	19%	21%	18%	13%	11%
$x(5)$	26%	30%	23%	13%	6%	2%	20%	23%	21%	16%	12%	9%
a	39%	28%	18%	10%	4%	1%	26%	23%	20%	15%	10%	7%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	1%	3%	15%	81%	5%	11%	26%	57%		
		$x(2)$	6%	16%	64%	14%	13%	21%	42%	24%		
		$x(3)$	27%	53%	16%	4%	30%	38%	19%	12%		
		a	66%	28%	6%	1%	52%	30%	12%	6%		
					2	**1**	**2**	**1**				
				$x(1)$	6%	94%	17%	83%				
				a	94%	6%	83%	17%				

Table 4. Results depending on the average inefficiency setting.

	Shuffling-OCEM						**Dual-OCEM**					
Low Average Inefficiency												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	0%	1%	2%	5%	22%	70%	5%	5%	8%	15%	26%	42%
$x(2)$	2%	3%	7%	20%	47%	21%	10%	9%	12%	18%	27%	23%
$x(3)$	6%	10%	21%	40%	18%	5%	17%	16%	17%	20%	16%	13%
$x(4)$	14%	20%	35%	20%	8%	2%	19%	19%	21%	18%	12%	10%
$x(5)$	29%	36%	21%	10%	3%	1%	22%	24%	21%	15%	11%	7%
a	48%	30%	15%	5%	1%	0%	27%	26%	20%	13%	8%	5%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	0%	1%	12%	86%	4%	9%	25%	62%		
		$x(2)$	2%	13%	74%	11%	10%	20%	46%	24%		
		$x(3)$	22%	63%	12%	2%	30%	42%	18%	10%		
		a	75%	22%	2%	0%	57%	29%	10%	4%		
					2	**1**	**2**	**1**				
				$x(1)$	4%	96%	14%	86%				
				a	96%	4%	86%	14%				
High Average Inefficiency												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	4%	3%	5%	9%	23%	56%	9%	9%	11%	15%	24%	31%
$x(2)$	8%	9%	11%	18%	31%	24%	14%	14%	14%	17%	21%	21%
$x(3)$	13%	15%	19%	24%	19%	9%	18%	17%	16%	17%	17%	15%
$x(4)$	19%	21%	22%	20%	12%	6%	19%	18%	19%	17%	14%	13%
$x(5)$	26%	25%	23%	15%	8%	3%	18%	21%	20%	17%	13%	11%
a	31%	27%	20%	13%	6%	2%	22%	21%	20%	16%	12%	10%
			4	**3**	**2**	**1**	**4**	**3**	**2**	**1**		
		$x(1)$	2%	5%	18%	75%	8%	14%	28%	50%		
		$x(2)$	10%	19%	54%	17%	15%	23%	36%	25%		
		$x(3)$	32%	43%	19%	6%	31%	34%	20%	15%		
		a	56%	34%	9%	1%	46%	29%	16%	9%		
					2	**1**	**2**	**1**				
				$x(1)$	8%	92%	22%	78%				
				a	92%	8%	78%	22%				

In the scenarios with $m = 6$, we observe that when variables are misclassified, they are almost always placed in the relative position of other variables with small differences between the values of the respective exponents. For example, less confusion arises when the differences between consecutive exponents are larger, such as between $x(1)$ and $x(2)$ in Scenario (16b) and between $x(2)$ and $x(3)$ in Scenario (16d), whereas the rest of the consecutive differences are smaller than 0.1 and some confusion arises in the rankings.

In the scenarios with $m = 4$, we observe that there is still some confusion among variables when the difference between their exponents is up to 0.10, while larger differences result in a clear separation between their positions in the ranking. For example in Scenario (15a), where the exponents of $x(1)$ and $x(2)$, as well as between $x(3)$ and a differ by 0.1, there is some confusion, whereas variable $x(3)$ is almost always classified in either 3rd or 4th place, and hardly ever as more important. In this case, the difference in exponent between $x(2)$ and $x(3)$ is relatively large at 0.25. Similar trends can be observed throughout.

Table 5. Scenarios with $m = 6$.

	Shuffling-OCEM						**Dual-OCEM**					
Scenario (16a)												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	3%	2%	3%	8%	21%	63%	7%	9%	11%	16%	24%	33%
$x(2)$	6%	7%	12%	21%	36%	18%	15%	13%	14%	18%	21%	19%
$x(3)$	10%	13%	19%	28%	21%	9%	17%	16%	16%	18%	17%	16%
$x(4)$	18%	20%	26%	19%	12%	5%	19%	18%	19%	16%	15%	13%
$x(5)$	26%	30%	22%	13%	6%	3%	18%	22%	21%	16%	13%	10%
a	38%	29%	17%	10%	4%	1%	24%	22%	19%	15%	11%	9%
Scenario (16b)												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	1%	1%	1%	3%	12%	82%	3%	4%	6%	13%	24%	50%
$x(2)$	7%	8%	10%	23%	44%	8%	14%	14%	15%	19%	24%	14%
$x(3)$	10%	12%	20%	31%	22%	4%	17%	17%	17%	20%	17%	12%
$x(4)$	16%	19%	29%	21%	12%	3%	18%	19%	20%	18%	14%	10%
$x(5)$	27%	30%	21%	13%	7%	2%	22%	22%	21%	16%	12%	8%
a	39%	30%	18%	9%	4%	1%	25%	24%	21%	14%	10%	6%
Scenario (16c)												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	3%	4%	6%	12%	26%	50%	11%	11%	14%	17%	22%	25%
$x(2)$	4%	6%	11%	20%	31%	28%	13%	12%	14%	17%	22%	21%
$x(3)$	8%	12%	18%	28%	22%	12%	16%	15%	15%	17%	18%	17%
$x(4)$	15%	21%	28%	19%	10%	6%	18%	17%	18%	18%	14%	15%
$x(5)$	27%	31%	21%	11%	6%	3%	19%	22%	19%	16%	12%	11%
a	42%	26%	17%	9%	5%	2%	23%	22%	20%	14%	11%	10%
Scenario (16d)												
	6	**5**	**4**	**3**	**2**	**1**	**6**	**5**	**4**	**3**	**2**	**1**
$x(1)$	2%	2%	2%	6%	31%	56%	6%	5%	8%	15%	30%	35%
$x(2)$	3%	3%	4%	11%	45%	34%	6%	7%	9%	17%	28%	34%
$x(3)$	10%	14%	22%	39%	10%	5%	19%	18%	19%	20%	14%	11%
$x(4)$	18%	21%	29%	22%	7%	3%	21%	19%	23%	18%	11%	9%
$x(5)$	29%	31%	23%	13%	3%	2%	21%	26%	21%	17%	9%	6%
a	39%	30%	19%	9%	3%	1%	27%	26%	21%	13%	8%	5%

Finally, in the scenarios with $m = 2$, both methodologies improve their performance as the value of the exponent γ of the relevant variable increases, as shown in Figure 2. In particular, Shuffling-OCEM accurately identified the relevant variable as the most important in over 95% of simulations whenever γ was at least 0.3, outperforming the Dual-OCEM method. When γ is small, the importance of the relevant variable decreases, and the proportion of successful ranking decreases, until when $\gamma = 0.1$, Shuffling-OCEM was still able to identify the correct order in 67% of replications, while Dual-OCEM basically guessed randomly, at around a 50% success rate.

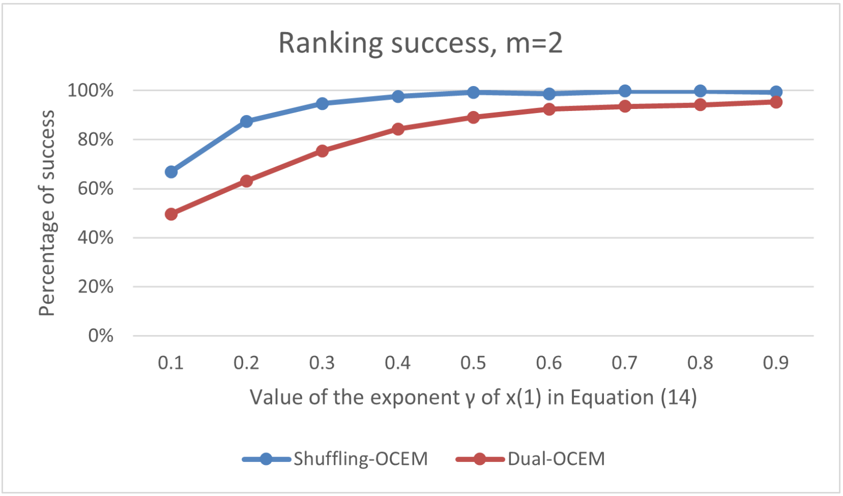

Figure 2. Ranking results in the scenarios (14) with $m = 2$, according to the value of the exponent γ of $x(1)$.

Table 6. Scenarios with $m = 4$.

	Shuffling-OCEM				**Dual-OCEM**			
Scenario (15a)								
	4	**3**	**2**	**1**	**4**	**3**	**2**	**1**
$x(1)$	1%	2%	26%	71%	4%	8%	33%	55%
$x(2)$	2%	5%	67%	27%	5%	13%	48%	35%
$x(3)$	29%	65%	5%	1%	34%	49%	11%	6%
a	69%	28%	2%	0%	58%	31%	8%	4%
Scenario (15b)								
	4	**3**	**2**	**1**	**4**	**3**	**2**	**1**
$x(1)$	0%	1%	6%	93%	3%	7%	21%	69%
$x(2)$	4%	9%	81%	6%	10%	19%	51%	20%
$x(3)$	38%	53%	8%	1%	40%	39%	15%	6%
a	57%	37%	5%	0%	48%	35%	13%	5%
Scenario (15c)								
	4	**3**	**2**	**1**	**4**	**3**	**2**	**1**
$x(1)$	1%	2%	10%	88%	5%	11%	24%	59%
$x(2)$	8%	19%	65%	9%	16%	26%	38%	20%
$x(3)$	26%	53%	19%	3%	29%	36%	22%	13%
a	65%	27%	7%	1%	50%	27%	15%	8%
Scenario (15d)								
	4	**3**	**2**	**1**	**4**	**3**	**2**	**1**
$x(1)$	3%	7%	21%	69%	11%	19%	29%	41%
$x(2)$	9%	27%	45%	18%	17%	26%	31%	26%
$x(3)$	16%	46%	27%	11%	20%	31%	26%	23%
a	73%	19%	7%	1%	51%	24%	14%	10%

We remark that part of the higher confusion in the scenarios between variables in the scenarios with a larger number of input variables can be attributed to the smaller exponents that lead to smaller differences while retaining the characteristics of non-increasing returns to scale of the Cobb-Douglas functions.

Regarding the position of the irrelevant variable a in the production processes under study, while it is sometimes classified higher in the rankings than some relevant variables, it is almost always placed in the relative position of variables with exponents of up to 0.2.

This could indicate that exponents of such magnitudes may not be very important to the production processes being considered.

Finally, regarding the computational time taken by both methods, we mention that the programming language used was Python and that CPLEX v12.8 was utilized for solving the optimization programs. We observe that Shuffling-OCEM takes between 13% and 42% longer than Dual-OCEM, depending on the number of inputs (m) and the number of DMUs (n). Average computation times for each combination of number of inputs and number of DMUs are reported in Table 7. As the number of inputs increases, the computational time of both methods increases, with Dual-OCEM scaling better than Shuffling-OCEM. However, we can observe that both methods take longer with larger sample sizes, but as the number of samples increases, the time taken by Dual-OCEM increases faster than that of Shuffling-OCEM, resulting in slightly smaller ratios in the cases with 100 DMUs than in those with 50 DMUs. In particular, Dual-OCEM takes around 2.5 times longer to execute with 100 DMUs as with 50 DMUs, while Shuffling-OCEM takes around 2.4 times longer. The simulations were executed on a PC with a 1.8 GHz dual-core Intel Core i7 processor, 8 Gigabytes of RAM, and operating system Microsoft Windows 10 Enterprise.

Table 7. Computational times.

m	n	**Shuffling-OCEM**	**Dual-OCEM**	**Ratio**
2	50	64 s	55 s	1.15
	100	154 s	140 s	1.135
4	50	147 s	112 s	1.307
	100	356 s	280 s	1.269
6	50	244s	171s	1.423
	100	590 s	430 s	1.370

These results show that the Shuffling-OCEM method is significantly better than the Dual-OCEM method at establishing a correct ranking of variables, while taking, on average, 1.25 times the computational time to execute, so it can be considered a better method.

5. Conclusions and Future Work

In this paper, we have presented some methods for measuring the importance of variables in production processes based on an adaptation of the OneClass Support Vector Machine estimator to Efficiency Measurement (OCEM), and evaluated their ability to rank the variables by relative importance to efficiency measurement in a production process. This adapted estimator applies data-centric optimization to the estimation of the production technology, which can be viewed as the region of the space where the observed data lies, while attempting to improve the generalization capability of standard DEA. Based on the production technologies estimated, we evaluate the importance of variables in determining this technology, and thus for the production process.

This is an important topic in the efficiency estimation literature due to the effect that including additional variables has on the estimations, particularly when they are not very relevant. This allows for the consideration of whether to remove some of the less important variables from a productive process in order to obtain better model specifications.

In particular, we adapt two classic methodologies from the machine learning literature, based on shuffling the values of a variables (Shuffling-OCEM) and the dual formulation of the OCEM estimator (Dual-OCEM). We compare them using Cobb Douglas functions in the single-output setting, with Variable Returns to Scale and with independently generated inputs.

In these simulated scenarios, we observe that the Shuffling-OCEM methodology outperforms Dual-OCEM, with a slightly higher computational cost. Both methods improve their performance as the number of DMUs increase. They are relatively robust to the compared types of inefficiency distributions, obtaining similar results with both a half-normal and an exponential distribution. Both methods decrease their accuracy as the average level of inefficiency increases, since the effect of the high average inefficiency can surpass the small relative difference in importance between variables. We observe that the performance of both methods depends on the exponents of the variables, with higher differences between these exponents yielding more correct rankings. In particular, as the exponents decrease as more variables are included, the rankings worsen as the number of variables increase. Comparing them to each other, the Shuffling-OCEM methodology clearly outperforms the Dual-OCEM methodology overall, and in each of the comparisons. In fact, Shuffling-OCEM performs better when the average inefficiency is high or when the sample size is small than Dual-OCEM in the corresponding lower average or high sample size scenarios. In particular, we observe that Shuffling-OCEM correctly ranks the variables in 94% of the scenarios with one relevant and one irrelevant input, while Dual-OCEM achieves an overall success rate of 82%. When the exponent of the relevant variable is low, the performance of the Dual-OCEM methodology deteriorates more than that of the Shuffling-OCEM. Regarding the scenarios with $m = 4$, Shuffling-OCEM is capable of ranking each variable correctly in at least 65% of replications of scenario (15a), with varying results according to the relative differences between consecutive values of the exponents, while the respective Dual-OCEM percentages lie between 48% and 58%.

Therefore, we conclude that the Shuffling-OCEM methodology should be used over the Dual-OCEM methodology, at least in situations similar to those considered. This is further supported by a computational cost that is only slightly higher for the Shuffling-OCEM than the Dual-OCEM methodology. Further studies could evaluate whether these conclusions hold in more general cases, such as those with correlations among variables, or with different production functions which may not satisfy convexity or other properties.

Finally, we mention some possible avenues for further research. These are just some of the methods available in the literature for the ranking of variables, and other methods could be adapted to the OCEM estimator in order to compare their performance. The proposed OCEM estimator uses a PieceWise Linear (PWL) transformation function, but this is not the only possible choice. There is a variety of kernels and transformation functions which could be used, such as polynomial, Gaussian, or sigmoid kernels, among others. The proposed methodology treats both inputs and outputs homogeneously, so these methodologies could be considered for the ranking of outputs, or even for the ranking of both inputs and outputs simultaneously, while still taking into account the characteristics of production processes, which could be an area worth exploring. In practice, the methods proposed in this paper could be used to measure the importance of variables in real-life datasets, and as a basis to obtain models for the measurement of efficiency involving only those variables considered more important by the methods. Furthermore, in addition to the ranking of variables, these methods could be enriched with stopping rules or thresholds to determine when a variable is relevant or not, turning them into methods for selection of variables in production processes. An interesting line of future work in this direction could be the usage of the proposed methodologies with real-life datasets, to evaluate whether some variables can be considered irrelevant. Another potential line of future work is to evaluate the performance of the proposed methods in a wider variety of scenarios, with broader characteristics such as different assumptions about returns to scale, a variety of production functions, or relationships between the variables among others. Another interesting future line of research is the comparison of the new methods with other methodologies available in the literature in order to compare their performance.

Author Contributions: Conceptualization, R.M. and J.A.; methodology, R.M. and J.A.; software, M.E.; validation, R.M. and M.E.; formal analysis, R.M., J.A. and M.E.; investigation, R.M., J.A., and M.E.; resources, R.M., J.A. and M.E.; data curation, M.E.; writing—original draft preparation, R.M.; writing—review and editing, R.M., J.A. and M.E.; visualization, R.M. and M.E.; supervision, J.A.; project administration, J.A.; funding acquisition, J.A. All authors have read and agreed to the published version of the manuscript.

Funding: R.M. acknowledges financial support from Cátedra Santander en Eficiencia y Productividad.

Data Availability Statement: Datasets were simulated.

Acknowledgments: The authors are grateful to the four anonymous reviewers for providing constructive comments and helping in improving the contents and presentation of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

DEA	Data Envelopment Analysis
DMU	Decision Making Unit
DDF	Directional distance function
OCSVM	OneClass Support Vector Machine
OCEM	OneClass for Efficiency Measurement
$\boldsymbol{x}$	Inputs
$\boldsymbol{y}$	Outputs
$\boldsymbol{z}$	Netputs
m	Number of inputs
s	Number of outputs
n	Number of DMUs
Ψ	Production Possibility Set
$\hat{\Psi}$	Estimated Production Possibility Set
$\boldsymbol{g}$	Directional vector
μ_i	(Minus) DDF DEA distance of DMU i
$\mathcal{Z}$	Dataset in netput notation
$\mathcal{Z}_{train}$, $\mathcal{Z}_{test}$	Train-test split of dataset $\mathcal{Z}$
$\boldsymbol{w}, \boldsymbol{\xi}, \rho$	Variables of the OCSVM algorithm
ν	OCSVM hyperparameter
$\boldsymbol{\phi}$	Transformation function of the OCSVM algorithm
$\boldsymbol{\phi}_{PWL}$	PieceWise Linear Transformation function
$\boldsymbol{p}_k, q_k$	Hyperplane parameters of the PWL transformation
μ	Offset hyperparameter of the PWL transformation function
E	MSE of the OCEM estimator in Shuffling-OCEM
J	Objective value of the Dual OCEM program
$\boldsymbol{\alpha}, \boldsymbol{\gamma}$	Variables of the Dual OCEM program.

References

1. Cobb, C.W.; Douglas, P.H. A theory of production. *Am. Econ. Rev.* **1928**, *18*, 139–165.
2. Koopmans, T.C. Efficient allocation of resources. *Econometrica* **1951**, *19*, 455. [CrossRef]
3. Debreu, G. The Coefficient of Resource Utilization. *Econometrica* **1951**, *19*, 273–292. [CrossRef]
4. Farrell, M.J. The Measurement of Productive Efficiency. *J. R. Stat. Soc. Ser. A Gen.* **1957**, *120*, 253–290. [CrossRef]
5. Shephard, R.W. *Cost and Production Functions*; Princeton University Press: Princeton, NJ, USA, 1953.
6. Charnes, A.; Cooper, W.W.; Rhodes, E. Measuring the efficiency of decision making units. *Eur. J. Oper. Res.* **1978**, *2*, 429–444. [CrossRef]
7. Banker, R.D.; Charnes, A.; Cooper, W.W. Some models for estimating technical and scale inefficiencies in data envelopment analysis. *Manag. Sci.* **1984**, *30*, 1078–1092. [CrossRef]
8. Aigner, D.; Lovell, C.A.K.; Schmidt, P. Formulation and estimation of stochastic frontier production function models. *J. Econometr.* **1977**, *6*, 21–37. [CrossRef]
9. Meeusen, W.; van Den Broeck, J. Efficiency Estimation from Cobb-Douglas Production Functions with Composed Error. *Int. Econ. Rev.* **1977**, *18*, 435–444. [CrossRef]
10. Esteve, M.; Aparicio, J.; Rabasa, A.; Rodriguez-Sala, J.J. Efficiency analysis trees: A new methodology for estimating production frontiers through decision trees. *Expert Syst. Appl.* **2020**, *162*, 113783. [CrossRef]
11. Simar, L.; Wilson, P.W. Sensitivity analysis of efficiency scores: How to bootstrap in nonparametric frontier models. *Manag. Sci.* **1998**, *44*, 49–61. [CrossRef]
12. Simar, L.; Wilson, P.W. A general methodology for bootstrapping in non-parametric frontier models. *J. Appl. Stat.* **2000**, *27*, 779–802. [CrossRef]
13. Kneip, A.; Park, B.U.; Simar, L. A Note on the Convergence of Nonparametric DEA Estimators for Production Efficiency Scores. *Econom. Theory* **1998**, *14*, 783–793. [CrossRef]

14. Lee, P.F.; Lam, W.S.; Lam, W.H. Performance Evaluation of the Efficiency of Logistics Companies with Data Envelopment Analysis Model. *Mathematics* **2023**, *11*, 718. [CrossRef]
15. Ratner, S.V.; Shaposhnikov, A.M.; Lychev, A.V. Network DEA and Its Applications (2017–2022): A Systematic Literature Review. *Mathematics* **2023**, *11*, 2141. [CrossRef]
16. Kuosmanen, T.; Johnson, A.L. Data envelopment analysis as nonparametric least-squares regression. *Oper. Res.* **2010**, *58*, 149–160. [CrossRef]
17. Parmeter, C.F.; Racine, J.S. Smooth Constrained Frontier Analysis. In *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis: Essays in Honor of Halbert L. White, Jr*; Chen, X., Swanson, N.R., Eds.; Springer: New York, NY, USA, 2013; pp. 463–488. [CrossRef]
18. Daouia, A.; Noh, H.; Park, B.U. Data envelope fitting with constrained polynomial splines. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2016**, *78*, 3–30. [CrossRef]
19. Tsionas, M.G. Efficiency estimation using probabilistic regression trees with an application to Chilean manufacturing industries. *Int. J. Prod. Econ.* **2022**, *249*, 108492. [CrossRef]
20. Valero-Carreras, D.; Aparicio, J.; Guerrero, N.M. Support vector frontiers: A new approach for estimating production functions through support vector machines. *Omega* **2021**, *104*, 102490. [CrossRef]
21. Olesen, O.; Ruggiero, J. The hinging hyperplanes: An alternative nonparametric representation of a production function. *Eur. J. Oper. Res.* **2022**, *296*, 254–266. [CrossRef]
22. Guerrero, N.M.; Aparicio, J.; Valero-Carreras, D. Combining Data Envelopment Analysis and Machine Learning. *Mathematics* **2022**, *10*, 909. [CrossRef]
23. Borchani, H.; Varando, G.; Bielza, C.; Larranaga, P. A Survey on Multi-Output Regression. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2015**, *5*, 216–233. [CrossRef]
24. Daraio, C.; Simar, L. *Advanced Robust and Nonparametric Methods in Efficiency Analysis: Methodology and Applications*; Studies in Productivity and Efficiency; Springer: New York, NY, USA, 2007. [CrossRef]
25. Rosenblatt, M. Remarks on some nonparametric estimates of a density function. *Ann. Math. Stat.* **1956**, *27*, 832–837. [CrossRef]
26. Parzen, E. On estimation of a probability density function and mode. *Ann. Math. Stat.* **1962**, *33*, 1065–1076. [CrossRef]
27. Vapnik, V. *Statistical Learning Theory*; John Wiley & Sons: Chichester, UK, 1998.
28. Vapnik, V. *The Nature of Statistical Learning Theory*; Information Science and Statistics; Springer: New York, NY, USA, 2013.
29. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the support of a high-dimensional distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef] [PubMed]
30. Charles, V.; Aparicio, J.; Zhu, J. The curse of dimensionality of decision-making units: A simple approach to increase the discriminatory power of data envelopment analysis. *Eur. J. Oper. Res.* **2019**, *279*, 929–940. [CrossRef]
31. Ruggiero, J. Impact assessment of input omission on DEA. *Int. J. Inf. Technol. Decis. Mak.* **2005**, *4*, 359–368. [CrossRef]
32. Jenkins, L.; Anderson, M. A multivariate statistical approach to reducing the number of variables in data envelopment analysis. *Eur. J. Oper. Res.* **2003**, *147*, 51–61. [CrossRef]
33. Pastor, J.T.; Ruiz, J.L.; Sirvent, I. A statistical test for nested radial DEA models. *Oper. Res.* **2002**, *50*, 728–735. [CrossRef]
34. Banker, R.D. Hypothesis tests using data envelopment analysis. *J. Product. Anal.* **1996**, *7*, 139–159. [CrossRef]
35. Fanchon, P. Variable selection for dynamic measures of efficiency in the computer industry. *Int. Adv. Econ. Res.* **2003**, *9*, 175–188. [CrossRef]
36. Nataraja, N.R.; Johnson, A.L. Guidelines for using variable selection techniques in data envelopment analysis. *Eur. J. Oper. Res.* **2011**, *215*, 662–669. [CrossRef]
37. Peyrache, A.; Rose, C.; Sicilia, G. Variable selection in data envelopment analysis. *Eur. J. Oper. Res.* **2020**, *282*, 644–659. [CrossRef]
38. Benítez-Peña, S.; Bogetoft, P.; Morales, D.R. Feature selection in data envelopment analysis: A mathematical optimization approach. *Omega* **2020**, *96*, 102068. [CrossRef]
39. Limleamthong, P.; Guillén-Gosálbez, G. Mixed-integer programming approach for dimensionality reduction in data envelopment analysis: Application to the sustainability assessment of technologies and solvents. *Ind. Eng. Chem. Res.* **2018**, *57*, 9866–9878. [CrossRef]
40. Li, Y.; Shi, X.; Yang, M.; Liang, L. Variable selection in data envelopment analysis via Akaike's information criteria. *Ann. Oper. Res.* **2017**, *253*, 453–476. [CrossRef]
41. Li, Y.; Liang, L. A Shapley value index on the importance of variables in DEA models. *Expert Syst. Appl.* **2010**, *37*, 6287–6292. [CrossRef]
42. Ueda, T.; Hoshiai, Y. Application of principal component analysis for parsimonious summarization of DEA inputs and/or outputs. *J. Oper. Res. Soc. Jpn.* **1997**, *40*, 466–478. [CrossRef]
43. Adler, N.; Golany, B. Including principal component weights to improve discrimination in data envelopment analysis. *J. Oper. Res. Soc.* **2002**, *53*, 985–991. [CrossRef]
44. Andersen, P.; Petersen, N.C. A procedure for ranking efficient units in data envelopment analysis. *Manag. Sci.* **1993**, *39*, 1261–1264. [CrossRef]
45. Shen, W.f.; Zhang, D.q.; Liu, W.b.; Yang, G.l. Increasing discrimination of DEA evaluation by utilizing distances to anti-efficient frontiers. *Comput. Oper. Res.* **2016**, *75*, 163–173. [CrossRef]

46. Fernandez-Palacin, F.; Lopez-Sanchez, M.A.; Munõz-Márquez, M. Stepwise selection of variables in DEA using contribution loads. *Pesqui. Oper.* **2018**, *38*, 31–52. [CrossRef]
47. Sharma, M.J.; Yu, S.J. Stepwise regression data envelopment analysis for variable reduction. *Appl. Math. Comput.* **2015**, *253*, 126–134. [CrossRef]
48. Jitthavech, J. Variable elimination in nested DEA models: A statistical approach. *Int. J. Oper. Res.* **2016**, *27*, 389–410. [CrossRef]
49. Lee, C.Y.; Cai, J.Y. LASSO variable selection in data envelopment analysis with small datasets. *Omega* **2020**, *91*, 102019. [CrossRef]
50. Chen, Y.; Tsionas, M.G.; Zelenyuk, V. LASSO+DEA for small and big wide data. *Omega* **2021**, *102*, 102419. [CrossRef]
51. Duras, T.; Javed, F.; Månsson, K.; Sjölander, P.; Söderberg, M. Using machine learning to select variables in data envelopment analysis: Simulations and application using electricity distribution data. *Energy Econ.* **2023**, *120*, 106621. [CrossRef]
52. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
53. Zhang, Y.; Yang, A.; Xiong, C.; Wang, T.; Zhang, Z. Feature selection using data envelopment analysis. *Knowl.-Based Syst.* **2014**, *64*, 70–80. [CrossRef]
54. Al-Tawil, M.; Mahafzah, B.A.; Al Tawil, A.; Aljarah, I. Bio-Inspired Machine Learning Approach to Type 2 Diabetes Detection. *Symmetry* **2023**, *15*, 764. [CrossRef]
55. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
56. Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V. Gene selection for cancer classification using support vector machines. *Mach. Learn.* **2002**, *46*, 389–422. [CrossRef]
57. Luenberger, D.G. New optimality principles for economic efficiency and equilibrium. *J. Optim. Theory Appl.* **1992**, *75*, 221–264. [CrossRef]
58. Cherchye, L.; De Rock, B.; Walheer, B. Multi-output profit efficiency and directional distance functions. *Omega* **2016**, *61*, 100–109. [CrossRef]
59. Chambers, R.G.; Chung, Y.; Färe, R. Benefit and distance functions. *J. Econ. Theory* **1996**, *70*, 407–419. [CrossRef]
60. Huang, X.; Mehrkanoon, S.; Suykens, J. Support vector machines with piecewise linear feature mapping. *Neurocomputing* **2013**, *117*, 118–127. [CrossRef]
61. Briec, W. Hölder distance function and measurement of technical efficiency. *J. Product. Anal.* **1999**, *11*, 111–131. [CrossRef]
62. Sirvent, I.; Ruiz, J.L.; Borrás, F.; Pastor, J.T. A Monte Carlo evaluation of several tests for the selection of variables in DEA models. *Int. J. Inf. Technol. Decis. Mak.* **2005**, *4*, 325–343. [CrossRef]
63. Banker, R.D.; Chang, H. A simulation study of hypothesis tests for differences in efficiencies. *Int. J. Prod. Econ.* **1995**, *39*, 37–54. [CrossRef]