# DeepNotebooks: Deep Probabilistic Models Construct Python Notebooks for Reporting Datasets

Claas Völcker[1], Alejandro Molina[1], Johannes Neumann[2],
Dirk Westermann[2], and Kristian Kersting[1,3]

[1] Machine Learning Group, CS Department, TU Darmstadt
c.voelcker@stud.tu-darmstadt.de molina@cs.tu-darmstadt.de
[2] University Heart and Vascular Center Hamburg
jo.neumann@uke.de d.westermann@uke.de
[3] Centre for Cognitive Science, TU Darmstadt
kersting@cs.tu-darmstadt.de

**Abstract.** Machine learning is taking an increasingly relevant role in science, business, entertainment, and other fields. However, the most advanced techniques are still in the hands of well-educated and -funded experts only. To help to democratize machine learning, we propose DeepNotebooks as a novel way to empower a broad spectrum of users, which are not machine learning experts, but might have some basic programming skills and are interested data science. Within the DeepNotebook framework, users feed a cleaned tabular datasets to the system. The system then automatically estimates a deep but tractable probabilistic model and compiles an interactive Python notebook out of it that already contains a preliminary yet comprehensive analysis of the dataset at hand. If the users want to change the parameters of the interactive report or make different queries to the underlying model, they can quickly do that within the DeepNotebook. This flexibility allows the users to interact with the framework in a feedback loop — they can discover patterns and dig deeper into the data using targeted questions, even if they are not experts in machine learning.

## 1 Introduction

Data science has enjoyed considerable successes in recent years, both in creating more powerful models and broadening the range of potential applications. However, behind all these exceptional success stories, there are troves of human experts, including machine learning and domain experts, statisticians, and computer scientists, among others. These experts focus on different aspects aspect of the data analysis pipeline, from data acquisition and feature engineering to modeling selection, training, and evaluation. As the complexity of each of these tasks increases, even experts can lose track of all the details and nuances of each part of the pipeline. As for non-experts, they might not be aware of best practices nor have a chance to keep track of the rapidly evolving state-of-the-art.

These difficulties have given rise to a new area of research focused on building off-the-shelf machine learning methods that can easily be used by non-experts – automatic machine learning (AutoML).

Indeed, several different approaches to AutoML do already exist, ranging from automatic building blocks for feature engineering [1, 2, 3] and model learning [4, 5], to automatic reporting as in the Automatic Statistician [6, 7, 8] and interactive machine learning notebooks[4]. There is also work on interpreting the computations of modern machine learning models [9, 10, 11] as this is of crucial importance to non-experts. However, to the best of our knowledge, there is no framework yet that incorporates modeling, learning, reporting, explainability and interactivity at the same time. The question whether such an exploratory automatic statistician, which is not limited to only investigating a single target variable, is possible, was the seed that grew into the present paper.

Specifically, triggered by the recent successes of deep and tractable probabilistic models, we introduce DeepNotebooks[5] — an interactive system that automatically constructs data reports in the form of Python notebooks using mixed sum-product networks (MSPNs) [12, 13]. Similar approaches for modelling heterogeneous data have been explored before [14], leading to the BayesDB system for exploring databases [15]. Instead of exploring a method for approximate inference in databases, DeepNotebooks are built on models for tractable, exact inference for single table data. In addition, they are not only a framework for user interaction with data, but also reporting tool, which performs a preliminary array of common statistical tests. Python notebooks provide an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots, and rich media. A DeepNotebook is therefore not just a data report as it allows non-experts to interactively answer complex queries using tractable inference within the underlying MSPN.

We proceed as follows: We start off by briefly reviewing MSPNs. We then introduce DeepNotebooks. Before concluding, we illustrate them on several datasets, including one on myocardial infarction diagnosis.

## 2 Automatic Statisticians via Deep Probabilistic Models

The vision of an automatic statistician [6, 7, 8] is to build statistical models with minimal input from experts in statistics and machine learning. Probabilistic graphical models (PGMs) [16] are arguably a promising tool for realizing this vision. They can solve many ML tasks by estimating a distribution and then answering probabilistic queries. Consider, e.g. predictive modeling. One may train a PGM and use inference to obtain probabilistic answers to queries; for multi-class classification answering the query $\arg\max_c P(\text{Class} = c|\text{data})$ gives us the most likely class according to our model. Alternatively, we can ask which is

[4] `www.h2o.ai/h2o-old/h2o-flow/`

[5] Code available at `https://github.com/cvoelcker/DeepNotebooks`, an example of a generated DeepNotebooks at `https://cvoelcker.github.io/DeepNotebooks/demo/deepnotebook.html`

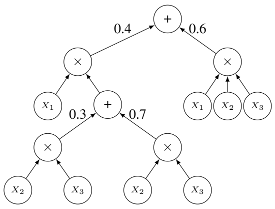


Fig. 1: An example of a valid SPN. Here, $x_1$, $x_2$ and $x_3$ are random variables modelled by histograms. The structure represents the joint distribution $P(x_1, x_2, x_3)$.

the most likely value of any feature: $\arg\max_x P(X = x|\text{evidence})$. Unfortunately, inference in unrestricted PGMs is intractable.

### 2.1 Deep and Tractable Probabilistic Models

Motivated by the importance of efficient inference for large-scale applications, a substantial amount of work has been devoted to learning probabilistic models for which inference is guaranteed to be tractable. Examples of these model classes include sum-product networks (SPNs) [17] and in particular mixed sum-product networks (MSPNs) [12], hinge-loss Markov random fields [18], and tractable higher-order potentials [19]. In this work, we have focused on SPNs.

Being instances of Arithmetic Circuits (ACs) [20], SPNs are a deep architecture that can represent high-treewidth models [21] and facilitate fast, exact inference for a range of queries in time linear in the network size [17]. They inherit universal approximation properties from mixture models – a mixture model is simply a "shallow" SPN with a single sum node. Consequently, SPNs can represent any prediction function, very much like deep neural networks. However, having exact probabilistic inference at hand offers an advantage not present in other PGMs and deep neural networks. One can compare the probabilities computed by different models and not only solve classification or regression problems, but also do anomaly detection at the same time while taking into account the statistical nature of the data. Also, instead of e.g. classical deep neural networks, SPNs are not only trained to predict the probability of a single target variable. This makes them especially useful in a data exploration context, where the true target of the investigation might not be known prior. Furthermore, any measures based on probabilities such as entropy, mutual information, and information gain can be computed efficiently. In the present paper, we will also show this for Shapley values [10].

### 2.2 Mixed Sum-Product Networks (MSPNs)

DeepNotebooks resort to MSPNs, as they are currently the only model able to build an SPN structure in a likelihood-agnostic way – using piecewise polynomials to encode the leaf distributions – therefore being suitable for our heterogeneous setting. MSPN learning is also able to deal with missing or unknown values out of the box, by implicit marginaliyation ove the missing features. This makes them applicable in contexts where common inputation methods might not be able to easily fill in a lot of the data.

**Representation of MSPNs:** Formally, an MSPN is a rooted directed acyclic graph, comprising *sum*, *product*, and *leaf* nodes as seen in Fig. 1. The scope of an MSPN is the set of random variables appearing on the network. More precisely, an MSPN can be defined recursively as follows:

1. a tractable univariate distribution is an MSPN.
2. a product of MSPNs defined over different scopes is an MSPN, and
3. a convex combination of MSPNs over the same scope is an MSPN.

Here, a product node encodes a factorization over independent distributions defined over different random variables, while a sum node stands for a mixture of distributions defined over the same variables. From this definition, it follows that the joint distribution modeled by an MSPN is a valid probability distribution, i.e. each complete and partial evidence inference query produces a consistent probability value [17, 22]. This also implies that we can construct multivariate distributions from simpler univariate ones. To build the structure in a likelihood-agnostic way, we make a piecewise approximation to the leaf distributions. In their purest form, piecewise constant functions are often adopted in the form of histograms or staircase functions. More expressive approximations are comprised of mixtures of truncated polynomials and exponentials.

**Tractable Inference in MSPNs.** To answer probabilistic queries in an MSPN, we evaluate the nodes starting at the leaves. Given some evidence, the probability output of querying leaf distributions is propagated bottom up. For product nodes, the values of the child nodes are multiplied and propagated to their parents. For sum nodes, instead, we sum the weighted values of the child nodes. The value at the root indicates the probability of the asked query.

To compute marginals, i.e., the probability of partial configurations, we set the probability at the leaves for those variables to 1 and then proceed as before. All these operations traverse the tree at most twice and therefore can be achieved in linear time w.r.t. the size of the MSPN.

**Learning MSPNs.** Existing SPN learning works focus on learning the SPN parameters given a structure [23, 24, 25] or jointly learn both the structure and the parameters [26, 27, 28]. A particular prominent approach is LearnSPN [29, 30], which recursively partitions a data matrix using hierarchical co-clustering. In particular, LearnSPN alternates between partitioning features into independent groups, inducing a product node, and clustering the data instances, inducing a sum node. As the base step, univariate likelihood models for single features are induced. To learn MSPNs, the LearnSPN algorithm is adapted to deal with the lack of parametric forms by performing a partitioning over mixed continuous and discrete data by exploiting a randomized approximation of the Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient (RDC) [31]. Thus, MSPNs maintain their expressiveness while representing a wide range of statistical data types, which can even be estimated automatically from data [13], resulting in an automatic exploratory density estimation approach.

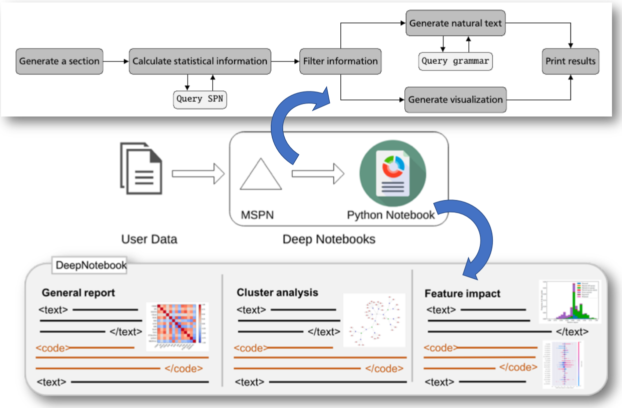


Fig. 2: Illustration of DeepNotebooks. A user feeds data into the system, and an MSPN is trained. The MSPN along with analysis information computed from the MSPN are then embedded into a Jupyter Python notebook, producing an interactive data report that uses the MSPN as "virtual statistical machine".

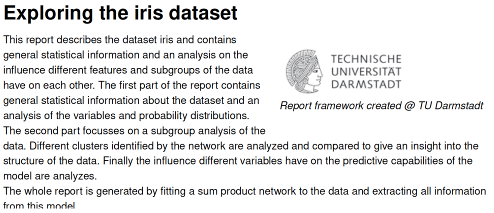
## 3 DeepNotebooks – Constructing Data Reports in the form of Python Notebooks based on MSPNs

Generally, the idea behind DeepNotebooks can be defined as follows: *A DeepNoteboook for a dataset D is a Jupyter notebook [32] with an embedded (deep) probabilistic model encoding the distribution of D. The model is used used to precompute the cells of the notebook describing the dataset D.*

The workflow of DeepNotebooks is depicted in Fig. 2: A user loads a dataset into the system, which automatically creates a probabilistic model that encodes the distribution of the data, in our case an MSPN. Using the MSPN, the system proceeds without user interaction and performs statistical analysis based on the model. Both the model and the analysis are then stored in a Jupyter notebook, the DeepNotebook. The user then opens the DeepNotebook where they can see the automatically generated report, as well as interact with the model. Changing the parameters of the reports is easy, as well as doing further analysis or even evaluating other datasets with the given model. All those options and more are available to the user from within the DeepNotebook. Each generated DeepNotebook contains three major sections, cf. Fig. 2:

**Section 1:** a general report on descriptive statistics and feature marginals,
**Section 2:** an analysis of the clusters encoded by the SPN structure
**Section 3:** report of the impact of features on conditional probabilities,

which are wrapped into natural text produced by templates and by a probabilistic grammar making the report more user-friendly, cf. Fig. 3. Normally, the notebook reports the top 5 results (variables with the highest correlation,



**Exploring the iris dataset**

This report describes the dataset iris and contains general statistical information and an analysis on the influence different features and subgroups of the data have on each other. The first part of the report contains general statistical information about the dataset and an analysis of the variables and probability distributions.
The second part focusses on a subgroup analysis of the data. Different clusters identified by the network are analyzed and compared to give an insight into the structure of the data. Finally the influence different variables have on the predictive capabilities of the model are analyzes.
The whole report is generated by fitting a sum product network to the data and extracting all information from this model

*Report framework created @ TU Darmstadt*

Fig. 3: Textual introduction to the DeepNotebook automatically written for Iris.

variables with the most impact on a prediction), but the user can also provide thresholds for reporting or specify which variables they are specifically interested in. We will now describe each section of a DeepNotebook in more detail.

**DeepNotebook Section 1 - General Report.** The general report provides a description of the data at hand. It includes descriptive statistics like correlations, dependencies, and mutual information among the random variables. Since the network represents the full joint probability density function of all variables, it allows efficient computation of these common statistical measures. Each calculation only requires one full bottom-up evaluation of the MSPN. The expectations of each variable can be computed by propagating expectations from the leaves to the root and treating each sum node as a weighted sum of expectations. The variable correlations can be computed similarly by recursively evaluating the covariance of the variables at each node. Covariance and correlation are only defined for ordered attributes in the data. Therefore, these measures cannot be computed for all variable combinations when the dataset contains categorical features. For the coupling between a categorical and a continuous variable, the notebook reports the coefficient of variation. For categorical variables without an ordering, normalized mutual information (MI) is reported.

Due to the efficient inference, it is possible to present partial dependency plots [33] for categorical and continuous variables. By default, a DeepNotebook selects all those features, which show a linkage (either due to covariance or MI) above a certain threshold. This can be adapted dynamically by the users according to their specific needs. Similar to the other sections, the notebook also contains a written explanation of the visualization, constructed from the probabilistic grammar shown in Fig. 4.

**DeepNotebook Section 2 - Cluster Analysis.** The MSPN structure contains an implicit hierarchical clustering due to the sum nodes. A DeepNotebook uses this to explain sub-clusters of the data and to provide descriptions for them. Furthermore, the distribution of variance for each variable between the clusters is calculated and presented. This allows users to understand the different parts of the underlying model better. Intuitively, if a particular cluster explains a large part of the variance of an interesting feature, this cluster and its constituent nodes are important for a more in-depth analysis.

**DeepNotebook Section 3 - Feature Impact.** Finally, a DeepNotebook computes different conditional probabilities for important variables and analyzes the influence of the features on predictions. To do this, each categorical variable is treated as a target variable separately, and the SPN is used as a pre-

```
<start>                  ::= <fullClause> | ... omitted for brevity
<fullClause>             ::= <subject_feature> <object_dependency> |
 ↪   <subject_dependency> <object_feature>
<subject_dependency>     ::= There is a {strength} {neg_pos} <dependency> | The model
 ↪   shows a {strength} {direction} <dependency>
<object_feature>         ::= <conjunctionFor> <features?> "{x}" and "{y}"
<subject_feature>        ::= <features?> "{x}" and "{y}" have
<object_dependency>      ::= a {strength} {neg_pos} <dependency>
<conjunctionFor>         ::= between | for
<dependency>             ::= <linear?> dependency | <linear?> relation | <linear?>
 ↪   relationship
<features>               ::= the features
<conjunctionWhile>       ::= , while | , but | . on the other hand
```

Fig. 4: Parts of the grammar used for producing correlation descriptions. Variables with a "?" at the end are randomly included/omitted for variation. Variables in curly brackets are replaced with the computed values from the MSPN.

dictive model. These predictions are then analyzed using the methods proposed by Robnik-Šikonja and Kononenko [34], Baehrens et al. [35] and Štrumbelj and Kononenko [36]. These explanation approaches allow the users to understand how different variables change the conditional probability of others and to estimate the importance of a feature for classification. They require only the computation of marginals, or gradients on the network. Due to the graph structure of the MSPN, gradients can be computed by a simple backward pass through the network using automatic differentiation. Marginalization of features is also easy in MSPNs [17]. For more details about the computations, we refer to Sec. 4.

The information is aggregated and normalized to provide an easy overview. Using Baehrens et al. [35]'s approach, the gradients are computed for each point in the original dataset and then normalized. This normalization is important, as the piecewise linear structure of the MSPN can result in very sharp edges with correspondingly large gradients. The normalized gradients represent relative importance for each feature and datapoint. These are then aggregated into one plot per feature and prediction, which describes the relative importance for the prediction. With significant computational resources, users can also use Shapley values [37] to estimate feature importance, which generalize the gradient approach but requires a Monte Carlo sampling step. Overall feature importance for each possible prediction attribute is then summarized using the mean squared distance of each gradient component to zero. This assures that features which never have any local impact on the classification are also assumed to not contribute to the impact globally. Finally, the Shapley values are visualized.

Since DeepNotebooks are Jupyter notebooks, the users can add more cells, access the model and the data, and add arbitrary Python code for further queries and analysis. A DeepNotebook therefore serves as an easy and accessible introduction to a dataset and enables a user to employ the reported results in their own data analytics pipeline later on. The data science loop does not start with an empty but with a pre-filled Python notebook, "programmed" by the machine.

## 4 Computing Statistical Measures using MSPNs

Each section of a DeepNotebook provides a different view on the data at hand and is based on statistical measures computed form MSPNs as underlying "virtual statistical machine". Showing how to compute them is one of our main technical contributions.

**Computing Covariance using MSPNs.** Calculating the covariance of two variables in a distribution can be decomposed into computing the joint expectation and the marginal expectation of each variable. The graph structure of an SPN allows an algorithm to calculate the moments of a probability function directly from the network, in one bottom-up pass. At a sum node, the moment of the distribution can be calculated as follows: $m_k = \mathbb{E}[x^k] = \sum p_i \mathbb{E}[x^k] = \sum p_i m_i^k$ . At a product node, the moments of independent variables are also independent, and therefore the moment of the child nodes can just be combined in a vector. At a leaf node, the moments need to be calculated according to the distributions. In the case of MSPNs, all leaves are piecewise linear density approximations, therefore it is very easy to calculate the required integral for the estimation, since all moments are polynomial functions of the base points. Using the moments as computational blocks, it is possible to compute the correlation matrix in closed form from the covariance matrix of the whole probability density function.

The joint expectation can also be calculated efficiently in a similar manner: First, the means of all leaf nodes are calculated independently. These are then combined at the sum and product nodes in a botoom-up-pass. At a product node, due to the assumption of independence, the joint expectation is equal to the product of the expectation: $\mathbb{E}[xy] = \mathbb{E}[x] \cdot \mathbb{E}[y]$. At a sum node, the expectations are multiplied by the weights and summed together. Finally, the correlation can be obtained by normalizing the covariance matrix by the variances of the features.

**Dependencies among Variables using the Law of Total Variance.** When dealing with general tabular data, it is necessary to deal with categorical variables. In this case, the covariance is not defined, and therefore we use the coefficient of determination as a measure for categorical-continuous variables and the mutual information for categorical-categorical dependence. Both are normalized between 0 and 1 and serve a similar purpose of estimating variable dependency as the correlation. The coefficient of determination between a categorical variable X and a continuous variable Y can be calculated as follows. If the categorical variable is assumed to correspond to clusters in the continuous one, the coefficient shows that the total variance of Y results from adding the intra- and inter-cluster variance:

$$1 = \frac{E[\sigma^2(p(Y|X))]}{\sigma^2(p(Y)} + \frac{\sigma^2(E[p(Y|X)])}{\sigma^2(p(Y))} \ .$$

The first term approaches zero as the clusters become smaller and more and more separated, while the second term approaches zero if the conditional means of the clusters do not vary significantly. To compute them, we extended the algorithm for computing mean and variance to conditional probabilities in MSPNs.

**Compiling Conditional MSPNs from MSPNs** Calculating moments as presented above, marginalizes the MSPNs over and over. We can optimize that by compiling a network that computes the conditional probability function $p(x|y)$. At a product node, the probability of $p(x)$ and $p(y)$ are independent of each other and therefore the conditional probability $p(x|y)$ equals the marginal $p(x)$. The leaf node representing $p_i(y)$ is omitted from the graph. At a sum node, the conditional probability function reads as

$$p(x|y) = p(y)^{-1} \sum\nolimits_i \alpha_i p_i(x, y) = \sum\nolimits_i \alpha_i p_i(y) p(y)^{-1} p(x) \ .$$

The probability $p_i(y)$ for each child serves as an update on the weights of the sum node. This assumes that the probabilities of $x$ and $y$ are independent for each child of the sum node, which is guaranteed if the algorithm is run bottom-up on the network as $p(y)$ has already been removed for all children. After extracting the conditional MSPN, the algorithm detailed above can be run to get the conditional means and variances of the continuous variable. Likewise, one can compute the mutual information:

$$\mathrm{MI}(x, y) = \frac{I(x, y)}{\sqrt{H(x)H(y)}} = \frac{\sum_x \sum_y p(x, y)(\log(p(x, y)) - \log(p(x)p(y)))}{\sqrt{\sum_x p(x)\log(p(x)) \sum_y p(y)\log(p(y))}} \ .$$

Indeed, mutual information has been used in the context of evaluating (M)SPNs [38], to visualize the connection between two variables. But evaluating the equation above using a numeric method by repeatedly calculating the needed probabilities for continuous features is potentially slow since the probability functions represented by MSPNs can be non-smooth. This is the reason this framework uses correlation for the dependency between continuous variables since approximating the mutual information becomes practically intractable for more complex marginal distributions, which would incur a runtime overhead not feasible in data exploration settings. For categorical variables, where the possible states are finite, and often few, the mutual information can be calculated precisely using the equation above.

**Estimating Shapley Explanation Values from MSPNs.** SHAP values [36, 10] estimate the impact of a feature on a single classification using the game-theoretic concept of Shapley values. These values represent the contribution each feature brings to the outcome of the classification by looking at subsets of features. In general, a complete calculation of these values for a classifier is intractable, because the number of possible subsets of features grows exponentially, but they can be estimated using Monte Carlo sampling [36]. Since MSPNs have a natural way of marginalizing over missing features, this can be done efficiently without using interpolation or summation over missing values.

## 5 Illustrations of DeepNotebooks

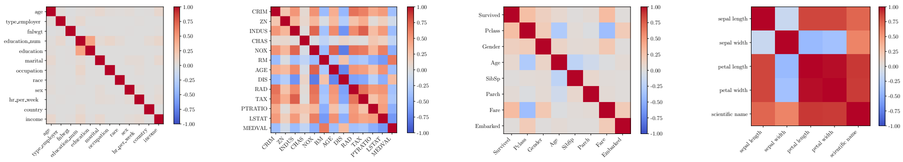
To investigate DeepNotbooks empirically, we implemented the system in Python using the SPFlow library [39] for learning the MSPN. Then we generated DeepNotebooks for four well known UCI datasets [40]. We used the Iris dataset to develop and test all algorithms and descriptions. We then generated DeepNotebooks on the Titanic, Boston Housing and Adult datasets. As a final validation, we generated a DeepNotebook for a real-world medical dataset comprised of information on heart infarct patients, which has not been studied in the context of machine learning or exploratory data analysis before.

**Experimental Protocol.** The Iris and Boston Housing datasets were used as provided by the sklearn library [41], while the Titanic and Adult datasets were cleaned and preprocessed. Finally, the medical dataset considers diagnosis of myocardial infarction using high-sensitivity troponin l 1-hour [42]. The dataset contains a large number of variables and a lot of missing values since not all patients are subject to all test procedures. We filtered this dataset by estimating the 20 most relevant attributes for diagnosis by using gradient tree boosting. We then generated a DeepNotebook to further investigate the relationship of these features to each other and the final diagnosis. For each dataset, we investigated three questions to assess the usability of the report for exploratory data analysis: Does the report reflect patterns in the data as expected from prior knowledge?, are there unexpected results?, and do these reflect genuine information discernible from the data?. Together, these questions aim at deducing whether the generated report provides reasonable insight into the data. Since the datasets are originally intended for classification or regression, we focused on understanding the relationship between the features and the label.

**Correlation and Statistical Measures.** For all datasets, we found that an overview of the marginal distributions can help the user to assess the general shape of the data quickly. The histograms calculated from the MSPN correspond to empirical histograms of the data, albeit smoothed by the training process.

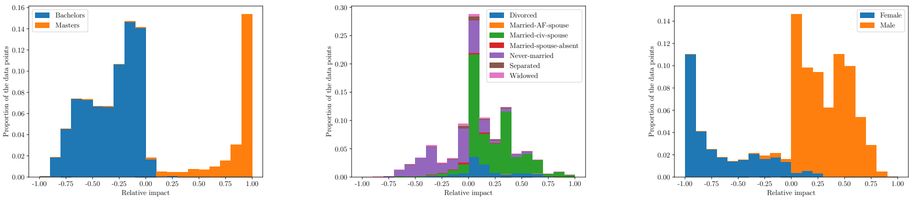
Fig. 5 shows the correlation and determination coefficients for the UCI datasets. For the Boston housing data, the correlation already captures many important relationships within the data. This is to be expected, as the Boston housing data is specifically intended to showcase simple regression models and linear correlations. Similarly, the Iris dataset contains a lot of well known dependencies. On the Titanic and Adult dataset, the descriptive statistics are non-informative, since the MSPN finds nearly no correlations within the data. The only significant finding in the Adult data results from two redundant attributes, the "education level" and a numeric representation thereof.

For the medical dataset, the mutual information (not shown here) indicates a clear dependence between the diagnosis and the troponin levels of a patient, which is a well known medical indicator for cardiovascular disease. The troponin levels of a patient at different test times were also clearly correlated with each other and less strongly with most other features, indicating their medical relevance among the tests. Other weaker dependencies do not warrant a closer inspection on the first pass, but might be interesting for a second, more thor-



(a) Adult (b) Boston housing (c) Titanic (d) Iris

Fig. 5: MSPN Correlations reported in the DeepNotebooks on four UCI datasets.



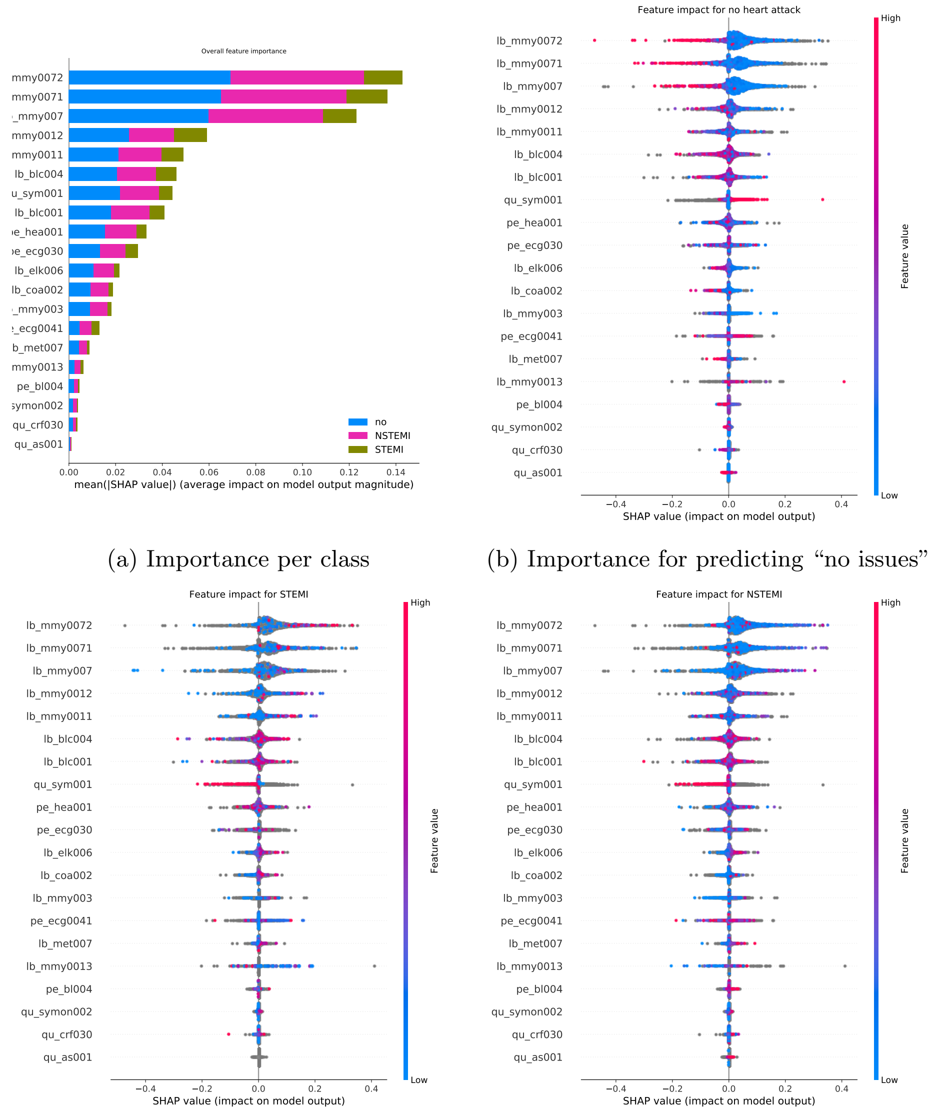
(a) Education (b) Marital relation (c) Gender

Fig. 6: Relative impact of the features with the widest spread on classifying high income in the Adult dataset as reported in the DeepNotebook.

ough, investigation. Since all features were already selected as being predictive for the diagnosis, this was counter-intuitive, but mutual information and correlation consider only pairwise interactions. This highlights the importance of non traditional statistical tests, like feature importance analysis for prediction.

**Explaining Predictions.** We found that explaining predictions can highlight variable interactions, which are not directly evident from correlation measures alone. On the Adult dataset, the variable most commonly chosen as the target for classification analysis is the variable “income”, which has two possible values, representing a yearly income below or above USD $50,000$ respectively. The predictive precision of DeepNotebook for this target was 76.28%. To assess the usefulness of the feature importance and impact measures, this classification was investigated in more detail.

Fig. 6 shows that the normalized gradient strengths for these features computed by the DeepNotebook. One can clearly see that ”education level”, ”sex”, and ”marital status” are informative features for the prediction with a clear impact. Also, by inspecting the visual explanations automatically created in a DeepNotebook, a user is able to assess that the values ”bachelors” and ”masters” for the variable ”education” have a pronounced, and opposed impact on the conditional probability. The histogram is clearly separated, with the value ”masters” generally increasing the probability of a person earning more than USD 50,000, and ”bachelors” decreasing this probability. Another feature which results in a relatively strong difference in probability is gender. Males are more likely to earn



(a) Importance per class

(b) Importance for predicting “no issues”

(c) Importance for predicting “STEMI”

(d) Importance for predicting “NSTEMI”

Fig. 7: Shapley impact values for the diagnosis prediction on the medical dataset. Grey dots visualize values for other class predictions.

more than females. Both of these results are not unexpected from the domain of the data and conform with prior expectations and as well as inspection of the underlying data. Another, less intuitive result of the DeepNotebook is that the ”marital status” has a recognizable effect on the probability of the income. A further investigation of this phenomenon shows that this pattern is also reflected in the data. This shows that it is indeed possible to quickly glean facts and avenues for further investigation from the automated analysis that might not have been expected a priori.

For the medical dataset we show the Shapley values computed by the DeepNotebook. As one can see in Fig. 7, different features are important for the separate classifications. This is a distinct case when compared to the Adult example, since in a multi-class prediction, the Shapley values and explanation vectors will not be symmetrical. Overall, the troponin levels (features lb_mmy) are the most important predictive features, which aligns with the strong mutual information coupling to the diagnosis and the medical background knowledge. The next important features contain information about symptoms, admission to the ICU, and ECG patterns detected. This conforms clearly to the expectation, for example in cases where the symptoms leading to ICU admission are long passed (feature qu_sym001 with high values) the chances of an acute infarction are far lower, while a specific ECG signal (low values for feature pe_ecg0041) are indicative of a STEMI type myocardial infarction (this signal is very typical for STEMI infarctions). The signal for the strong indicator troponin is only relevant for excluding heart attacks, STEMI and NSTEMI type occurrences both lead to high levels, although very high levels seem mostly indicative of a STEMI type infarction. For differentiating between different types of infarction, looking for example at the ECG result can help. This is a good starting point for a more in depth analysis of the detailed differences between the different heart attack types. Overall, DeepNotebooks can yield several findings that were not obvious to non-medics but conformed to medical knowledge.

## 6 Conclusions

We have presented DeepNotebooks — a novel way of interacting with data using a deep probabilistic model in the background. The generated reports automatically capture several insights from the data presented in a natural, comprehensible way. The automatic evaluation presented in written and graphical form enables domain experts that are not machine learning experts to get feedback instantly and to explore their data at their own pace. Changing the parameters of the generated report allows a user to choose between an in-depth analysis and a quick overview of the most important patterns. Also, since the data reports are Jupyter notebooks, the results reported are highly interactive and flexible. Overall, the example DeepNotebooks show that they allow one to find patterns in the data, which conform to prior expectation, but also result in novel findings. Given these insights, the user can directly investigate the model and data further using standard Python.

Nevertheless, DeepNotebooks are only a starting point and many things are left to be done. One should extend DeepNotebooks to include other statistical measures. Currently, DeepNotebooks are also specifically focused on analyzing categorical datasets. They should be extended to analyzing regression and time series data. Extending SPNs to provide better or even counterfactual explanations of the underlying model is another interesting avenue. Incorporating other model agnostic or developing similar measures specifically for SPNs could provide additional insight into the predictive capabilities of the network. Likewise, extracting Bayesian and Markov networks from SPNs would also lead to additional insights into the underlying data. Finally, there are many SPN learning algorithms with different properties. A thorough investigation of these, especially concerning how easy they are to use and tune for a non-expert user, would greatly improve the usability of the method for a wide audience.

**Acknowledgements:** The authors thank the reviewers for their valuable feedback. They also thank Antonio Vergari, Robert Peharz, and Isabel Valera for valuable discussions on DeepNotebooks. They also acknowledge the support of the BMBF project "Artificial intelligence for diagnosis of myocardial infarction" (B19-003 EXT) as well as of the Rhine-Main Universities Network for "Deep Continuous-Discrete Machine Learning" (DeCoDeML). Major parts of the work have been done as part of the B.Sc. thesis of CV [43].

## References

[1] Hoang Thanh Lam et al. "One button machine for automating feature engineering in relational databases". In: *arXiv preprint arXiv:1706.00327* (2017).

[2] Michael R Anderson et al. "Brainwash: A Data System for Feature Engineering." In: *CIDR*. 2013.

[3] Yingbo Zhou et al. "Parallel feature selection inspired by group testing". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3554–3562.

[4] Matthias Feurer et al. "Efficient and robust automated machine learning". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2962–2970.

[5] Hector Mendoza et al. "Towards automatically-tuned neural networks". In: *Workshop on Automatic Machine Learning*. 2016, pp. 58–65.

[6] David K. Duvenaud et al. "Structure Discovery in Nonparametric Regression through Compositional Kernel Search". In: *Proceedings of ICML*. 2013, pp. 1166–1174.

[7] James Robert Lloyd et al. "Automatic Construction and Natural-Language Description of Nonparametric Regression Models". In: *Proceedings of AAAI*. 2014, pp. 1242–1250.

[8] Hyunjik Kim and Yee Whye Teh. "Scaling up the Automatic Statistician: Scalable Structure Discovery using Gaussian Processes". In: *AISTATS*. 2018, pp. 575–584.

[9] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you? Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[10] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems.* 2017, pp. 4765–4774.

[11] Sebastian Lapuschkin et al. "Unmasking Clever Hans predictors and assessing what machines really learn". In: *Nature Communications* 10.1096 (2019).

[12] Alejandro Molina et al. "Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains". In: *AAAI* (2018).

[13] Antonio Vergari et al. "Automatic Bayesian Density Analysis". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).* 2019.

[14] V. K. Mansinghka et al. "Crosscat: A fully Bayesian, nonparametric method for analyzing heterogeneous, high-dimensional data". In: *Journal of Machine Learning Research* 17.138 (2016), pp. 1–49.

[15] V. K. Mansinghka et al. "BayesDB: A probabilistic programming system for querying the probable implications of data". In: *arXiv preprint arXiv:1512.05006* (2015).

[16] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

[17] Hoifung Poon and Pedro Domingos. "Sum-Product Networks: a New Deep Architecture". In: *UAI* (2011).

[18] Stephen H. Bach et al. "Hinge-Loss Markov Random Fields and Probabilistic Soft Logic". In: *JMLR* 18 (2017), 109:1–109:67.

[19] Daniel Tarlow, Inmar E. Givoni, and Richard S. Zemel. "HOP-MAP: Efficient Message Passing with High Order Potentials". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010.* 2010, pp. 812–819.

[20] Arthur Choi and Adnan Darwiche. "On Relaxing Determinism in Arithmetic Circuits". In: *Proceedings of ICML.* 2017, pp. 825–833.

[21] Han Zhao, Mazen Melibari, and Pascal Poupart. "On the Relationship between Sum-Product Networks and Bayesian Networks". In: *ICML.* 2015.

[22] Robert Peharz et al. "On Theoretical Properties of Sum-Product Networks". In: *AISTATS.* 2015.

[23] Robert Gens and Pedro Domingos. "Discriminative Learning of Sum-Product Networks". In: *NIPS.* 2012.

[24] Martin Trapp et al. "Safe Semi-Supervised Learning of Sum-Product Networks". In: *UAI* (2017).

[25] Han Zhao, Pascal Poupart, and Geoffrey J Gordon. "A Unified Approach for Learning the Parameters of Sum-Product Networks". In: *NIPS.* 2016, pp. 433–441.

[26] Aaron Dennis and Dan Ventura. "Learning the Architecture of Sum-Product Networks Using Clustering on Varibles". In: *NIPS.* 2012.

[27] Aaron Dennis and Dan Ventura. "Greedy Structure Search for Sum-product Networks". In: *IJCAI'15.* Buenos Aires, Argentina: AAAI Press, 2015, pp. 932–938. ISBN: 978-1-57735-738-4.

[28] Robert Peharz, Bernhard Geiger, and Franz Pernkopf. “Greedy Part-Wise Learning of Sum-Product Networks”. In: *ECML-PKDD 2013*. 2013.

[29] Robert Gens and Pedro Domingos. “Learning the Structure of Sum-Product Networks”. In: *ICML*. 2013, pp. 873–880.

[30] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. “Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning”. In: *Proceedings of ECML-PKDD*. 2015, pp. 343–358.

[31] David Lopez-Paz, Philipp Hennig, and Bernhard Schölkopf. “The Randomized Dependence Coefficient”. In: *NIPS*. 2013, pp. 1–9.

[32] Thomas Kluyver et al. “Jupyter Notebooks – a publishing format for reproducible computational workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87–90.

[33] Christoph Molnar. *Interpretable Machine Learning*. `https://christophm.github.io/interpretable-ml-book/`, accessed June 4, 2019. 2018.

[34] Marko Robnik-Šikonja and Igor Kononenko. “Explaining classifications for individual instances”. In: *IEEE Transactions on Knowledge and Data Engineering* (2008).

[35] David Baehrens et al. “How to explain individual classification decisions”. In: *Journal of Machine Learning Research* (2010).

[36] Erik Štrumbelj and Igor Kononenko. “Explaining prediction models and individual predictions with feature contributions”. In: *Knowledge and information systems* (2014).

[37] Scott M Lundberg et al. “Explainable machine-learning predictions for the prevention of hypoxaemia during surgery”. In: *Nature Biomedical Engineering* 2.10 (2018), p. 749.

[38] Alejandro Molina et al. “Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains”. In: *AAAI*. 2018.

[39] Alejandro Molina et al. *SPFlow: An Easy and Extensible Library for Deep Probabilistic Learning using Sum-Product Networks*. 2019. eprint: `arXiv:1901.03704`.

[40] Dua Dheeru and Efi Karra Taniskidou. *UCI machine learning repository*. Tech. rep. University of California, Irvine, School of Information and Computer Sciences, 2017.

[41] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[42] Johannes Tobias Neumann et al. “Diagnosis of myocardial infarction using a high-sensitivity troponin I 1-hour algorithm”. In: *JAMA cardiology* 1.4 (2016), pp. 397–404.

[43] Claas Völcker. “DeepNotebooks – Interactive data analysis using Sum-Product Networks”. B.Sc. thesis. TU Darmstadt, 2018.