The Effect of Human Thought on Data: An Analysis of Self-Reported Data in Supervised Learning and Neural Networks

Justin Lovinger and Iren Valova

Computer and Information Science Dept,
University of Mass Dartmouth, MA

Abstract

While scientific applications can gather consistent data from the natural world, psychological, sociological, and even economic applications rely on data provided by people. Since the majority of machine learning is aimed at improving the lives of people, human input is essential for useful results. In this paper, we explore datasets where input and target attributes are provided by people taking surveys. Every survey dataset, generated from human input, is reliable and self-consistent according to Cronbach's Alpha. One expects a reliable questionnaire to provide effective data for learning. It is this expectation that our analysis finds false, when applied to supervised learning. Both statistical analysis and application of several supervised learning architectures, with a focus on neural networks, are utilized to provide insight into data gathered through human input.

1 Introduction

Whether the economy, political events, product evaluation, or medicine, researchers use empirical data to solve problems. Psychologists use surveys to aid in diagnosis and checkups. Workplaces use them to screen potential employees. Websites gather survey data to enhance services and recommend the right products to the right people.

Our exploration begins with questionnaire surveys [1]. Our data preprocessing is simple, to focus on the consequences of human input in data. The person writing a survey designs questions with their own ideas of meaning and purpose. The people taking this survey each have their own unique interpretations of the questions, which can differ greatly from the intentions of the designer and other survey takers. Hence, a survey that is singular when designed, is numerous when observed by different people, resulting in a fascinating problem of noisy and inconsistent data [2, 3, 4, 5, 6] for processing and machine learning methods. The issues mentioned are not inherent to this medium, but rather a consequence of the complexities of people.

Noise comes from many sources, of which human input is only one. The prevalence of this problem has lead to much investigation by researchers [2, 4], and a number of techniques to mitigate the issue [5, 6, 7, 8, 9, 10]. Noise falls into two primary categories: label, or target, noise [2] and input noise. Target noise occurs when the target associated with an instance is wrong. In scientific applications, this most commonly occurs when an instrument malfunctions or an investigator makes a mistake. When targets are provided by untrained professionals, as in our surveys, this is far more common. Input noise is similar, and occurs when the inputs of an instance are wrong. How many inputs are wrong, and the degree of difference between the correct and observed inputs, determine the degree of input noise. Sources of input noise are similar to those of target noise.

Despite the noise generated, human input is essential for many applications. All subjective data must be gathered from people. Relying entirely on objective measurements is often impractical or impossible. Furthermore, the convenience of surveys with modern technology make them a staple of empirical data methods [11, 12, 13]. With the Internet, one can quickly connect with thousands of individuals willing to participate in a survey. With web technology, the collection of survey data can be largely automated.

The paper is structured in seven sections: next we focus on the state-of-the-art. The supervised learning architectures used in our exploration are detailed in Section 3. Included is a multilayer perceptron (MLP), a radial basis function network (RBF), a probabilistic neural network (PBNN), and a random forest classifier (RF). In Section 4 we present the surveys used in our experiments. Section 5 provides statistical analysis, to directly explore the nature of the data. Cronbach's alpha verifies the reliability of the surveys employed, and a k-nearest neighbors based noise cleaning method examines noise and consistency of the datasets. To explore the viability of these datasets for supervised learning, the architectures given in Section 3 are benchmarked in Section 6. Finally, Section 7 presents our conclusions.

2 State-of-the-Art and Applications

Many fields and applications already rely on empirical data gathered through surveys or similar questionnaires. Surveys are often employed in commercial markets and e-commerce to better understand customer needs. Psychology and medicine frequently rely on information reported by patients in questionnaires. Helping people make decisions in their everyday life can be accomplished with questionnaires supplemented with machine learning [1]. The datasets explored in this paper stem from this latter application.

In the field of psychology personality evaluation questionnaires are common [14], e.g. the Myers-Briggs type indicator. This test classifies according to extraversion (E) or introversion (I), sensing (S) or intuition (N), thinking (T) or feeling (F), and judging (J) or perceiving (P). The final classification consists of four values from these pairs, such as ISTJ, ISFJ, ESTP, ESFP, for a total of 16 classifications. Psychologists use statistical and empirical strategies to manually determine the correlation between questions and outputs, using data from people taking these questionnaires [15].

Beyond psychology, surveys are a common tool to aid the hiring process in business. Potential employees may be asked to complete a survey, which the company examines to determine if they are a good fit. Some larger companies already use such surveys as an automatic preliminary examination. When determining correlations between the questions asked and possible outputs, whether manually or automatically, our results show relying on answers provided by current employees is risky, as they are likely to be inaccurate and inconsistent.

In the field of machine learning, many famous and popular datasets rely, either partially or entirely, on human input. The human activity recognition dataset [16] uses smartphone sensors to form the input attributes, with labels for activities such as walking, laying down, and sitting. By requiring individuals to perform tasks, this dataset introduces human input to an otherwise objective set of measurements. For example, one individual may sit slumped backwards, in an almost laying position. This dataset illustrates how human input can introduce noise in difficult to predictable ways.

Although medical applications are known for rigorous data, these datasets can still require human input. The heart disease dataset [17] uses both objective measurements and data reported by the patient to predict heart disease. The patient is required to report the location, degree, and

cause of pain. This human input is likely to introduce significant noise.

While data from human input is often analyzed and criticized in sociology, business, medicine, and various other fields [18, 19, 20], the effect on supervised learning is seldom explored. The majority of literature on survey data focuses on the prediction of missing attributes [21, 22], rather than the characteristics of human input for supervised learning.

3 Explored Learning Architectures

Supervised learning architectures are chosen for their popularity, uniqueness, and effectiveness. This selection is not meant to be comprehensive, or optimal for these datasets, but rather to provide a well rounded means of exploring data.

3.1 Multilayer Perceptron

Multilayer perceptrons (MLP) have been a staple of pattern recognition and classification problems since the 1980s [23, 24], and are found in numerous applications, from financial systems to artistic expression [25]. Although MLPs are highly regarded as both relatively simple and highly effective neural networks, they do not always handle inconsistent data well. This is because a MLP must carefully position hyperplanes in order to classify data. This allows for the solution of complex, nonlinear problems. However, similar inputs can be classified very differently. In some problems, this is desirable, but it can lead to poor classification with noisy data. As such, datasets with human input provide a interesting challenge for this architecture.

With MLP so well-known, we do not summarize the classic backpropagation with momentum learning algorithm. Instead, we focus on modern enhancements, activation functions, training methods, and other extensions that have been developed due to the popularity of MLP.

While sigmoidal activation functions were originally recommended for MLP, research has found rectifier (ReLU) functions to provide more consistent training and faster convergence [26, 27, 28]. A naive rectifier has the form $f(x) = max(0, x)$. However, this function provides no gradient for backpropagation. Instead, the softplus function $f(x) = \ln(1 + e^x)$, depicted in Figure 1, approximates a rectifier while providing a gradient.

The standard backpropagation method of presenting every instance in every iteration has proven unnecessarily slow on large datasets. Stochastic gradient descent (SGD) [29] is an alternative approach that provides faster convergence and reduces the chance of becoming trapped in local minima. In SGD, a small number of patterns are randomly selected every iteration, and presented for training.

Dropout is a very recent technique that reduces overfitting by preventing co-adaptation of neurons [30]. During training, input and hidden neurons are randomly, and temporarily, deactivated. This mechanism efficiently emulates the training of an exponential number of neuron configurations. After training, weights are adjusted to the expected weights given the stochastic training mechanism, $W = pW'$, where W is the weight matrix after adjustment, W' is the weight matrix during training, and p is the independent probability of a neuron being active during training.

Since our goal is to explore the underlying data, we make no attempt to optimize the MLP architecture for each dataset, beyond hyperparameters. Instead we provide a configuration that is modern and popular, using ReLU activation functions, SGD, and dropout, in addition to standard backpropagation with momentum.

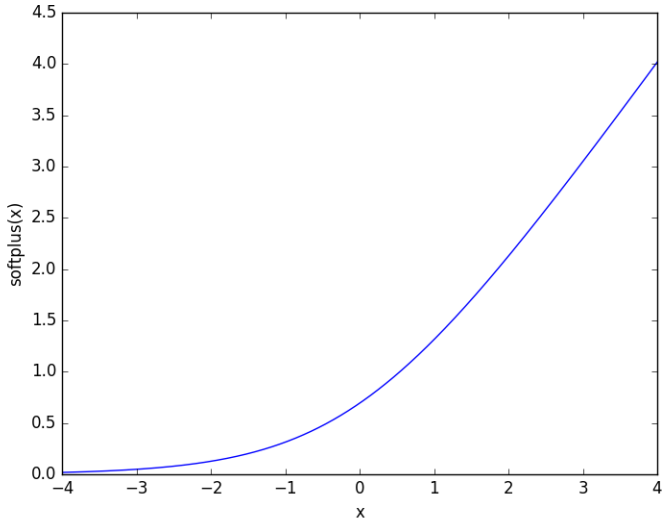

Figure 1: Softplus (ReLU) Activation Function

3.2 Radial Basis Function Network

To contrast the hyperplane based MLP, the radial basis function (RBF) network [31, 32, 33] is included. The RBF network utilizes hyperspheres to transform the problem space. These hyperspheres can be positioned with a large margin of error, minimizing the effect of noisy data. However, RBF networks do not have the same level of development and advancement as MLP. As such, they do not typically reach the same levels of accuracy in the state-of-the-art. However, noisy datasets play to the strengths of RBF networks.

The RBF network has a hidden layer of RBF neurons. Each neuron i has a center $\vec{c_i}$. During activation, this hidden layer returns a vector with one element n_i for each RBF neuron, determined by distance $d(\vec{x}, \vec{c_i}) = ||\vec{x} - \vec{c_i}||$ between input vector $\vec{x}$ and $\vec{c_i}$, and an RBF function. The Gaussian function is popular for this purpose:

$$n_i = e^{-\frac{d(\vec{x},\vec{c_i})^2}{v}} \tag{1}$$

Where v is a variance variable that determines the magnitude of n_i at a given distance.

Clustering is common for positioning the neuron centers, and any known clustering algorithm can produce a satisfactory arrangement of neurons. Our RBF architecture uses a standard Kohonen self organizing map (SOM), a state-of-the-art and very effective clustering method [34, 35, 36].

The output layer of the RBF network is similar to MLP:

$$\vec{o} = W\vec{n} \tag{2}$$

Where W is a weight matrix and $\vec{n}$ is the vector formed by all neuron outputs $\vec{n_i}$. The learning equation is likewise similar:

$$W_{t+1} = W_t + \alpha\vec{n_t}^{\intercal}(\vec{T} - \vec{o_t}) \tag{3}$$

Where $0 < \alpha \leq 1$ is a learning rate, $\vec{T}$ is the target vector, and the subscript t denotes a time step.

Note that the error term $\vec{T} - \vec{o_t}$ is not multiplied by the network output $\vec{o_t}$, as in MLP. As such, the RBF network learns to directly reproduce a given target $\vec{T}$, rather than positioning a hyperplane to separate instances. Despite this, the RBF network can simultaneously learn many patterns because the localized mechanism of the RBF neurons performs a soft partition of the input space. Therefore, different target vectors can be learned in different portions of the input space.

3.3 Probabilistic Neural Network

Probabilistic neural networks (PBNN) are shown to be useful for classification with highly noisy data [37], such as text data mined from web pages [38]. In such situations, a PBNN can have up to twice the accuracy of an MLP. As such, we expect high performance from this architecture.

Like RBF networks, the PBNN uses hyperspheres in the hidden layer, and thus has the same advantage as RBF networks with noisy data. The PBNN matches an RBF neuron to each pattern in the dataset. This allows for high accuracy with datasets that are not easily clustered. Additionally, the PBNN can output the probability of each class, rather than just a classification [39].

For each pattern p in a dataset, the PBNN creates an RBF neuron i with a center equal to the input vector of p. Additionally, the PBNN stores the target vector $\vec{t_i}$ of p for later use. During activation, the PBNN calculates the Gaussian output of each neuron i (1), as in the RBF network. The output calculation is where the PBNN diverges from the RBF network:

$$\vec{o} = \frac{\sum_{i \in N} n_i \vec{t_i}}{\sum_{i \in N} \vec{t_i}} \tag{4}$$

Where N is the set of neurons stored in the PBNN. Note the normalization by stored target vectors in the denominator, which minimizes the effect of uneven class frequencies. Optionally, the PBNN can normalize the output to obtain probabilities: $\frac{\vec{o}}{\max(o)}$.

Since the PBNN does not require an iterative process of adjustment for training, it can quickly converge in only one pass of the training data. This conveys the benefit of very fast training. However, by storing every pattern as a neuron, activation is slower than a comparable RBF network.

3.4 Random Forest

The random forest (RF) classifier [40, 41, 42] is an ensemble of decision trees (DT) [43, 44, 45]. DT is a classic classification algorithm that uses a sequence of if-then rules, applied on each input attribute, and arranged in a tree, to decide the class of an input. This is depicted in Figure 2. While some variants expand on the if-then rule formula, this description remains appropriate.

Several learning algorithms exist to generate a decision tree. Most work by determining the best attribute for a node, and recursing with a subset of instances for each child node. A child node is generated for each value of an attribute, or a range for continuous attributes, and the subset of instances for that node is those with that attribute value. The best attribute for a node is the one that best splits instances into individual classes. The ideal attribute splits a node such that each child node has instances with only 1 class. The closer an attribute is to this ideal split, the better. Most decision tree learning algorithms are differentiated by how the quality of an attribute split is determined. Information theory [47] remain popular for this purpose. ID3 [48], the classic decision tree algorithm uses information gain to determine the best attribute to split on.

In RF each DT operates on a subset of input attributes, and is trained on a subset of instances. Once trained, the RF classifies via a simple competitive vote among the DTs that make it up. The class that most DTs output, is the output of the RF. The RF algorithm, despite its simplicity, has

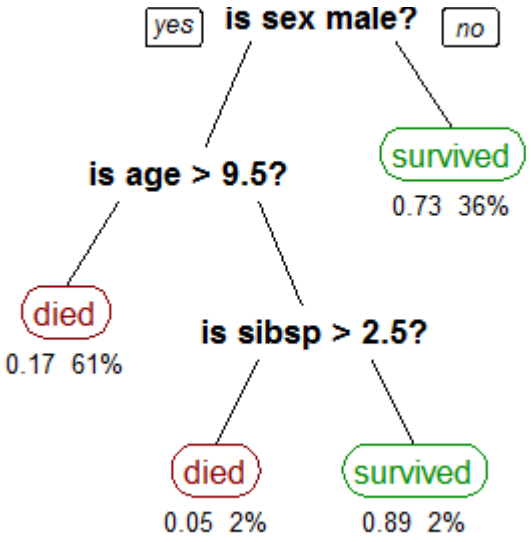

Figure 2: Decision tree showing survival of passengers on the Titanic [46]

proven extremely useful in supervised learning due to its ability to almost monotonically increase accuracy as the complexity of the ensemble increases [40].

Our RF implementation is provided by the scikit-learn Python library [49], using the information gain based DT learning algorithm provided by scikit-learn. No limit is given to tree depth or node splitting, and each terminal leaf contains only one class.

4 Surveys

With the empirical data gathering method known as a survey, we easily collect human input. Each user is presented with a number of questions, and provides or selects an answer to each. In each survey, the answer to one question forms the target data for a dataset, while the rest form the input data. Datasets span a variety of topics, from Linux distributions, to pets, to alcoholic drinks.

Table 1 provides a sample of the questions presented. While not all are provided here, this sample is representative of the questions used.

Survey questions that form the input vector for learning focus on demographics, personality, and interests. The goal is to correlate characteristics of the individual with target questions. Statistical analysis in Section 5.2 verifies the reliability of these questions.

Each question is presented in one of the following formats:

1. Radio button questions present a question and a selection of answers. Users can select one and only one answer.

2. Checkbox questions are similar to radio button questions, but allow users to select any number of answers, or no answer.

3. Scale questions present a question, and a selection of values from 1 to 7. Users can select one value.

4. Number questions present a question and an input box that accepts a number from a given minimum to maximum range.

An example of questions presented to users is provided in Figure 3

Table 1: Sample questions

Category	Questions
Demographic	What is your gender?
	How old are you?
Big 5 Personality	I know many different words (I have a rich vocabulary).
	I have excellent ideas.
	I use sophisticated words.
	I have many ideas.
	I pay attention to details.
	I like order.
	I always work to the best of my abilities.
	I like being the center of attention.
	I start conversations.
	I am interested in people.
	I take time out for others.
	I am often stressed out.
	I am easily made unhappy.
	I worry about things often.
General Personality	I like to stand out in a crowd.
	I like being challenged.
	I like taking risks.
	I like being in control.
	I am always energetic.
	I am very patient.
	Regarding rewards, do you prefer: (Quick, but lesser, rewards, A long investment, with greater payoff)
	Do you tend to focus on: (The big picture, The little details)
Interests	How much do you like watching TV?
	How much do you like listening to music?
	How much do you like reading books?
	How much do you like browsing social media (Facebook, Twitter, etc.)?
	How much do you like dancing?
	How much do you like programming?
Targets	What is your favorite Linux distribution?
	What is your favorite type of pet?
	What is your favorite type of alcoholic drink?

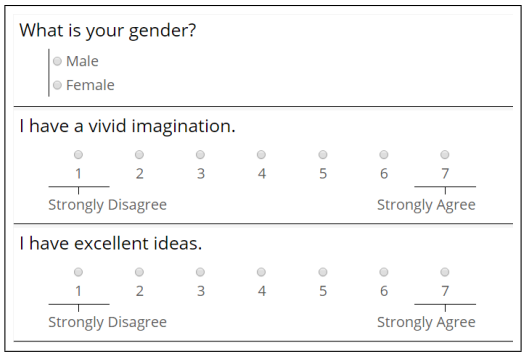

Figure 3: Example survey questions

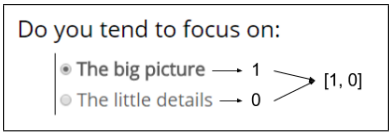

Figure 4: Conversion of Radio Question to Input

4.1 From Answers to Inputs

To facilitate use for supervised learning, raw answers are converted into numeric inputs.

Radio button questions are handled as nominal data. Each question is converted into a one-hot array of values. All unselected answers receive a value of 0, and the selected answer receives a value of 1. The total number of values is equal to the number of possible answers for the question. This conversion is visually depicted in Figure 4.

Checkbox questions are handled as binary data. Each answer is converted into an array of values, similar to radio questions. However, since multiple answers can be selected by the user, all selected answers receive a value of 1, and all unselected answers receive a value of 0. This conversion is visually depicted in Figure 5.

Scale questions are handled as ordinal data. Each question is translated into a single input. The user selected value is normalized to a value from 0 to 1. This conversion is visually depicted in Figure 6. Note that the selected value and maximum are offset by 1 because the minimum normalized value is 0.

Answers to numeric questions are simply normalized from 0 to 1, providing a single input value.

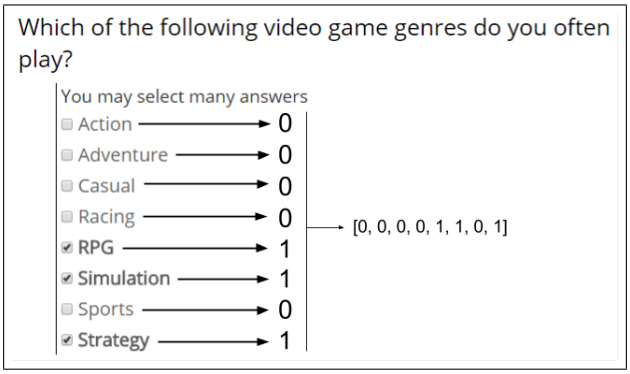

Figure 5: Conversion of Checkbox Question to Input

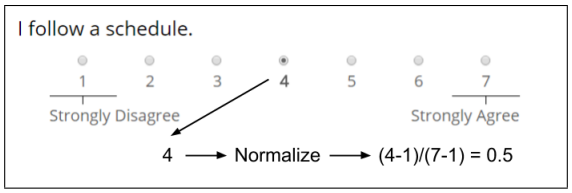

Figure 6: Conversion of Scale Question to Input

Table 2: Survey dataset statistics

Dataset	Instances	Input Size	Target Size
Linux Distros	639	68	20
Pets	105	69	16
Alcoholic Drinks	186	40	11

Since radio questions contribute an additional dimension to the input space for every answer, they are used sparingly. If a question has many potential answers, a checkbox question, or multiple scale questions are used instead, since these formats provide more information per input dimension than radio questions.

Finally, the processed answers are concatenated to form the input vector. Radio questions are used to gather target data. As such, the target vector is a one-hot array.

4.2 Explored Topics and Datasets

Datasets are generated from three surveys of varying topics. Topics are: distributions of the Linux operating system, common household pets, and alcoholic drinks.

The Linux survey [50] asks a number of questions about personality, including many from the big five personality traits [51, 52, 53] , as well as questions about user interests, such as watching movies, using social media, and reading books. The target question asks the user to select their favorite Linux distribution, and specifies that the user must have used Linux for at least 2 years, and be familiar with at least 3 Linux distributions. 19 of the most popular distributions are given as options, in addition to an "other" option. Figure 7 presents the frequency of answers given to this question.

The pets survey [54] asks similar personality questions, and adds questions specific to interacting and caring for a pet. The target question asks the user to select their favorite type of pet, and specifies that the user must have had this type of per for at least a year. Figure 8 presents the frequency of answers given to this question.

The alcohol survey [55], as with the previous surveys, asks personality questions, and includes a number of questions specific to the topic of alcohol. The target question asks the user to select their favorite type of alcoholic drink, and specifies that they must have tried at least 4 types of alcohol. Figure 9 presents the frequency of answers given to this question.

Statistics of the datasets generated from responses to these surveys are presented in Table 2. The number of instances, equal to the number of responses; the number of values in an input vector; and the number of values in a target vector, equal to the number of classes; are provided.

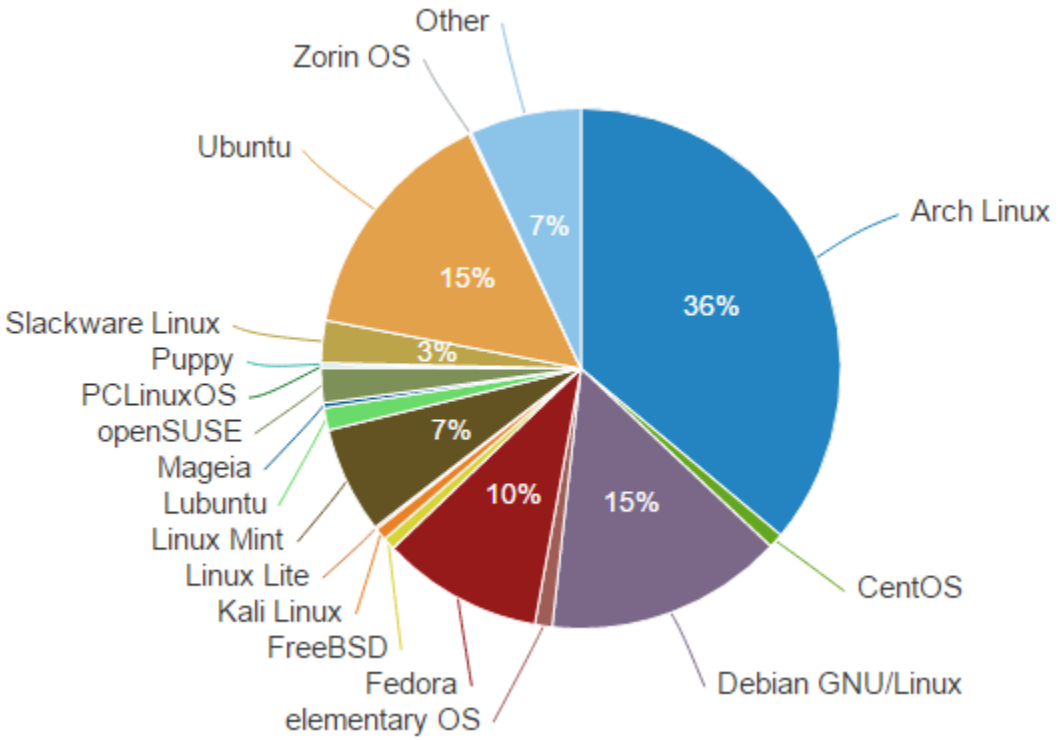

Figure 7: Distribution of answers to the Linux survey

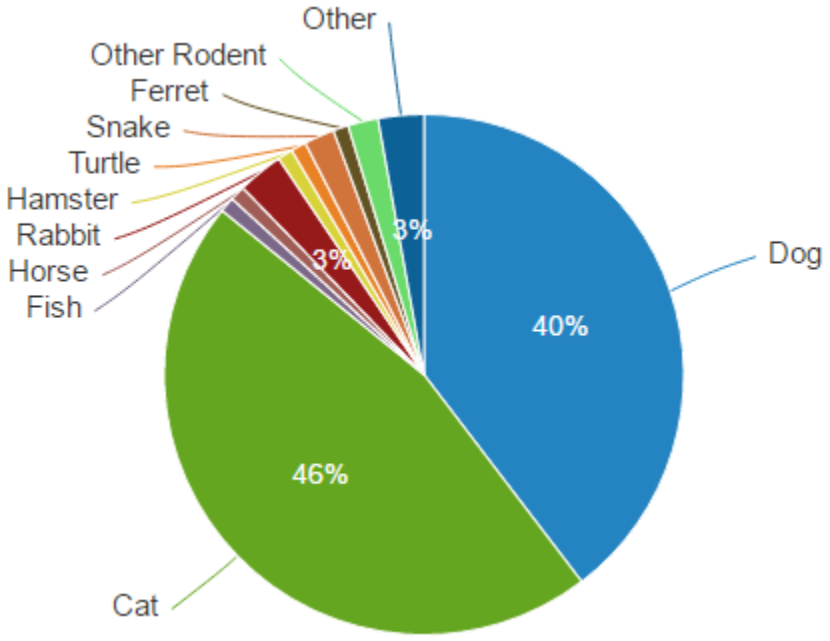

Figure 8: Distribution of answers to the pets survey

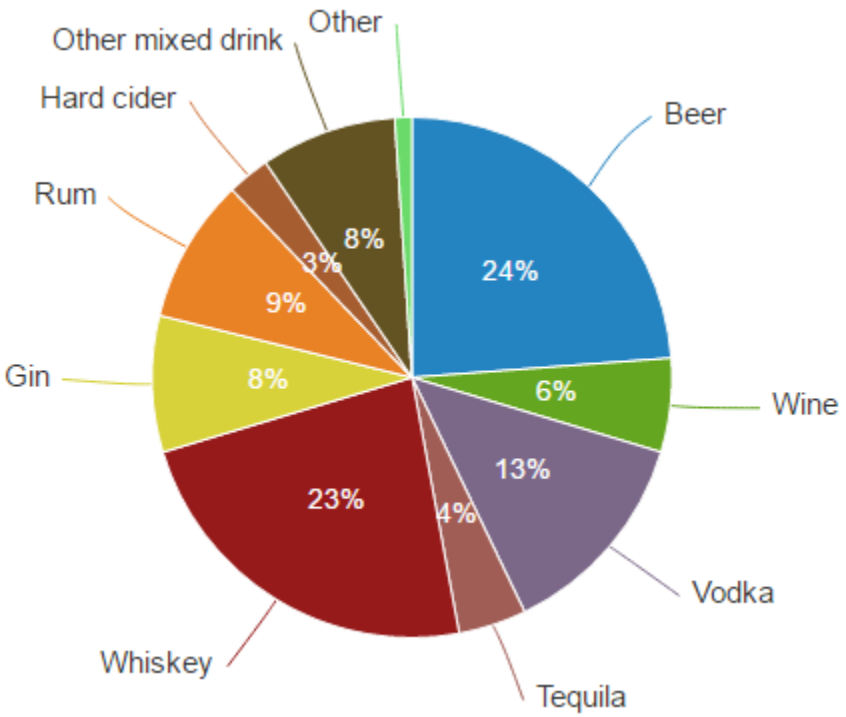

Figure 9: Distribution of answers to the alcoholic drinks survey

5 Analysis of Data

Whether through negligence, ignorance, or maliciousness, self-reported input and target data often results in noisy attributes and incorrect targets. Here, we statistically analyze datasets gathered with human input, and compare to known datasets.

Although we focus on the use of datasets generated from human input in supervised learning, we cannot directly utilize supervised learning for this analysis without introducing the bias inherit in any model. This section uses conventional statistics and preprocessing techniques to directly analyze the data, thereby decoupling the quality of the dataset from the effectiveness of the learning algorithm.

5.1 Comparison Datasets

In addition to the survey datasets described in Section 4.2, a number of low-noise datasets are presented for comparison. These datasets contrast our human input data by presenting objective numerical information, rather than target and input values reported by people. By analyzing the comparison datasets, we provide baseline results for low and acceptable levels of noise.

The well known iris dataset [56] contains input attributes describing various characteristics of a flower, including sepal and petal measurements. The goal is to classify a flower into three types of iris. This dataset is included for its precise attribute measurements, and undoubtedly correct target classes. We can confidently claim this dataset to be low noise, and highly consistent.

The similarly popular diagnostic Wisconsin breast cancer dataset [56, 57] contains measurements to aid in identifying cancer, and either a benign, or malignant classification. As with the iris dataset, this cancer dataset provides attributes gathered through careful objective measurement. As a dataset designed for medial application, we can trust the precision of the measurements and

Table 3: Comparison dataset statistics

Dataset	Instances	Input Size	Target Size
Iris	150	4	3
Cancer	569	30	2
Random	100	10	10

Table 4: Cronbach's Alpha

Dataset	Cronbach's Alpha
Linux Distros	0.762
Pets	0.747
Alcoholic Drinks	0.784

accuracy of the classifications.

A dataset of randomly generated instances is included as a control. This dataset consists of 100 instances, each with an input consisting of 10 uniformly random values, from 0 to 1, and a class randomly selected from 10 possible targets. We expect this dataset to perform very poorly, and thus we can verify the validity of our tests.

The input vector of each dataset is normalized to a vector of values from 0 to 1, and the target is presented as a one-hot vector. Table 3 presents statistics for the each comparison dataset. The number of instances; the number of values in an input vector; and the number of values in a target vector, equal to the number of classes; are provided.

5.2 Cronbach's Alpha

Before analyzing the datasets generated from our surveys, we must verify that the surveys themselves are reliable. An unreliable test is unlikely to generate a consistent and useful dataset. Cronbach's alpha (CA) [58, 59] allows us to decouple the reliability of the survey from the noise of the corresponding dataset.

CA statistically calculates the correlation between questions in a survey. When a set of questions provide correlated data, we can conclude that they measure the same concept.

CA relies on a set of responses. Each response must form a list of score values, or items, derived from question answers. See Section 4.1 for details on how we derive scores. With a matrix of score values for each response, CA is calculated as:

$$\alpha = \frac{k}{k-1}(1 - \frac{\sum s_i^2}{s_T^2}) \tag{5}$$

where k is the number of score values for each response, s_i^2 is the variance of the i^{th} item given the set of responses, and s_T^2 is the variance of a total score item obtained by summing all items for a response.

Table 4 presents the CA scores for our surveys. The maximum CA score is 1.0, and a score ≥ 0.7 is commonly considered good [59]. CA scores ≥ 0.9 are rare, and may indicate many redundant questions [58]. The high scores of our surveys indicate internal consistency and reliability of our surveys. As such, we can conclude that the choice of questions present no inherent noise.

We also note that several surveys, other than those presented, have been performed and rejected due to low reliability. The surveys and corresponding datasets presented in this paper are only those with high reliability. This decision is essential to our analysis.

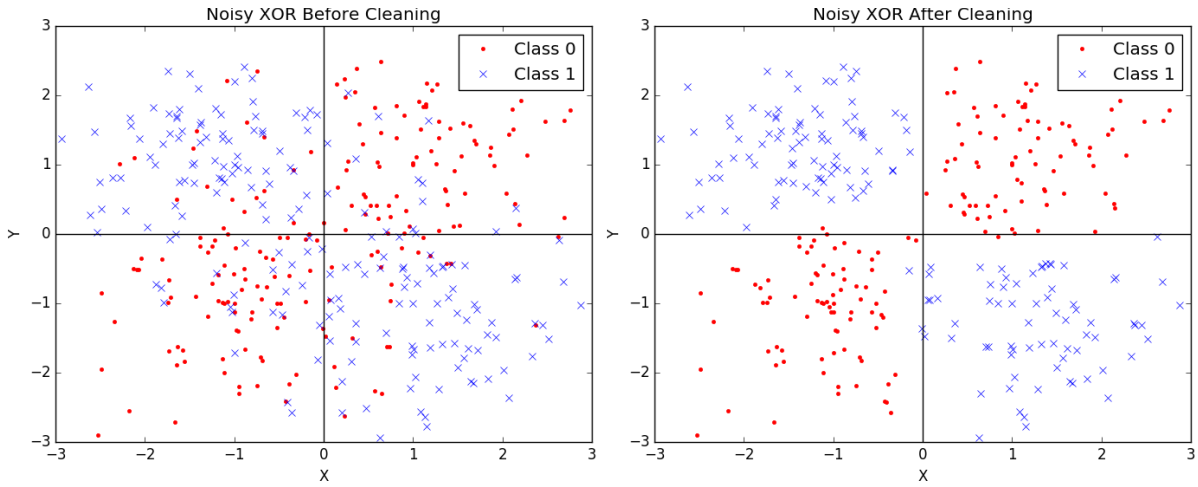

Figure 10: The Effect of Noise Cleaning on Noisy XOR

5.3 Noise Cleaning

Noise cleaning methods allows examination of datasets in the context of supervised learning, but independent of any particular architecture. This technique is important to our analysis, as it decouples the quality of a dataset from the effectiveness and bias of a supervised learning architecture.

A k-nearest neighbors based noise cleaning method, [7, 8, 9, 10] is used to correct potentially inaccurate target classes in a machine learning dataset. This method examines every instance in a dataset. For each instance, it finds the k nearest neighbors. If at least k' neighbors agree on a class, the instance receives, or maintains, this class. Otherwise, the instance is removed from the dataset. Since $(k+1)/2 \leq k' \leq k$, only one class c among the k nearest neighbors can meet the criteria that $C_c \geq k'$, where C_c is the count of class c among the k nearest neighbors. Also note that the k nearest neighbors are drawn from the original dataset, not the cleaned dataset as it is constructed.

Figure 10 provides an example of this algorithm on a noisy XOR dataset, with 100 additional instances for each original instance and Gaussian noise added to every additional instance. Before cleaning, we see many instances of the wrong class mixed with the correct instances, in all sectors. After cleaning, these instances are largely removed or corrected. Clearly, the dataset after cleaning better represents the XOR meaning of class 0 in the top right and bottom left sector, and class 1 in the top left and bottom right sector.

5.3.1 Noise Cleaning Analysis Results

For this analysis, we need only report the amount of mislabeled data in a given dataset to determine noise. Tables 5, 6, and 7 present this analysis. The percent of instances with changed target values, the percent of removed instances, and the sum of these percents, given as percent adjusted, are presented for each dataset.

This analysis confirms that the datasets gathered through human input show significantly higher levels of noise than the comparison datasets, with over 90% of instances removed in the worst case. The random dataset provides an essential benchmark for this analysis. Naturally, we except a completely random dataset to be very noisy, and difficult to generalize. The fact that datasets gathered through self-reported surveys show similar levels of adjustment is proof of the difficulties human input presents in the machine learning task.

Table 5: Noise Cleaning Results When $K = 3$ and $K' = 2$

Dataset	Percent Changed	Percent Removed	Percent Adjusted
Linux Distros	27.86%	57.75%	85.60%
Pets	34.29%	6.67%	40.95%
Alcoholic Drinks	32.80%	53.23%	86.02%
Iris	4.67%	0.00%	4.67%
Cancer	2.99%	0.00%	2.99%
Random	24.00%	72.00%	96.00%

Table 6: Noise Cleaning Results When $K = 5$ and $K' = 3$

Dataset	Percent Changed	Percent Removed	Percent Adjusted
Linux Distros	18.62%	71.99%	90.61%
Pets	29.52%	14.29%	43.81%
Alcoholic Drinks	16.67%	75.81%	92.47%
Iris	4.67%	0.00%	4.67%
Cancer	3.34%	0.00%	3.34%
Random	14.00%	85.00%	99.00%

We include various k and k' values to ensure the noise detection is not significantly influenced by these hyperparameters. For the datasets presented here, higher k and k' values almost universally increase the number of adjusted instances, allowing us to conclude that different assignment of these hyperparameters will not influence the relative levels of noise.

6 Dataset Analysis Through Supervised Learning

Although statistical analysis provides an effective and unbiased view of data, to truly understand the effect of human input on supervised learning, we must examine the behavior of learning and classification with such datasets. Datasets gathered through self-reported surveys, presented in Section 4.2, are compared to the comparison datasets presented in Section 5.1. Because our focus is the presence of noise in human input datasets, we do not attempt to clean the survey datasets for more effective learning. Several architectures, described in Section 3, are utilized, to minimize the bias an architecture provides. This analysis, although dependent on the architectures and hyperparameters used, is invaluable to understanding the effect of human provided attributes on the characteristics of a dataset.

Our diverse selection of architectures provides an unbiased examination of the quality of these

Table 7: Noise Cleaning Results When $K = 10$ and $K' = 7$

Dataset	Percent Changed	Percent Removed	Percent Adjusted
Linux Distros	2.03%	97.18%	99.22%
Pets	9.52%	77.14%	86.67%
Alcoholic Drinks	1.08%	97.85%	98.92%
Iris	1.33%	5.33%	6.67%
Cancer	1.58%	4.04%	5.62%
Random	0.00%	100.00%	100.00%

Table 8: MLP Hyperparameters

Dataset	Shape	Learning Rate	Momentum Rate	p_i	p_h	Batch Size
Linux Distros	68, 200, 150, 100, 20	0.01	0.002	0.9	0.9	200
Pets	69, 100, 50, 20, 16	0.01	0.002	0.8	0.9	40
Alcoholic Drinks	40, 120, 60, 40, 11	0.01	0.002	0.9	0.5	60
Iris	4, 32, 16, 8, 3	0.01	0.003	1.0	1.0	75
Cancer	30, 80, 40, 20, 2	0.01	0.0035	0.9	0.7	200
Random	10, 120, 60, 40, 10	0.01	0.003	1.0	0.7	40

Table 9: RBF Hyperparameters

Dataset	Neurons	Variance (v)	Learning Rate (α)
Linux Distros	40	2.0	0.9
Pets	20	1.0	0.9
Alcoholic Drinks	30	1.0	0.7
Iris	30	0.75	0.9
Cancer	60	1.0	0.9
Random	40	0.75	0.9

datasets. We note that these architectures are not intended to achieve the best possible accuracy, but rather provide a diverse set of state-of-the-art techniques, through which we can examine the quality and nature of the datasets.

Table 8 presents hyperparameters for the multilayer perceptron (MLP) architecture on each dataset. Shape is given as a list of neuron counts for each layer, in order. p_i is the active neuron probability for the input layer, and p_h is the active neuron probability for all hidden layers. Batch size is the number of patterns randomly selected during every iteration.

Table 9 presents hyperparameters for the radial basis function (RBF) architecture on each dataset. Neurons is the number of RBF neurons. Variance and learning rate are described in Section 3.2.

Table 10 presents hyperparameters for the probabilistic neural network (PBNN) architecture on each dataset. Variance is used in the Guassian equation (1), and is determined by the average distance between instances in the dataset.

Table 11 presents hyperparameters for the random forest (RF) architecture on each dataset. Trees is the number of decision trees in the RF.

Table 12 presents the supervised learning results for every architecture on each dataset. Five fold cross validation is performed, and the mean result is presented. Mean number of training epochs,

Table 10: PBNN Hyperparameters

Dataset	Variance (v)
Linux Distros	3.18
Pets	3.72
Alcoholic Drinks	2.22
Iris	1.29
Cancer	2.02
Random	1.26

Table 11: RF Hyperparameters

Dataset	Trees
Linux Distros	50
Pets	40
Alcoholic Drinks	60
Iris	10
Cancer	10
Random	80

mean accuracy on the training set, and mean accuracy on the testing set are provided. Tables 13 through 28 present mean confusion matrices on the testing sets for a sample of architecture, dataset pairs, using the same cross validation. Each column corresponds to a class predicted by the architecture, and each row corresponds to the expected class given an instance.

As we see in Table 12, the iris and cancer datasets, known for carefully and scientifically gathered attributes and targets, show high accuracy on both training and testing sets, for all architectures. By contrast, the survey datasets show significantly lower accuracy. However, it is worth noting that because these datasets have 8 to 20 classes, these results are still far better than random and are statistically significant. Unsurprisingly, the random control dataset is extremely difficult to accurately classify, with the lowest testing accuracy for all architectures. We do not claim high accuracy is impossible to achieve with these noisy datasets, only that is it difficult. Further exploration may discover architectures and hyperparameters that result in high accuracy. Likewise, this is not sufficient evidence that datasets gathered from people contain noise, but does support the hypothesis.

We also note that the hyperspheres used in RBF and PBNN tend to provide higher accuracy with these noisy datasets, proving the advantage they deliver on highly noisy and uncertain datasets. Likewise, large RF ensembles prove relatively effective on these difficult datasets.

These results clearly indicate high noise and mislabeled instances in the survey datasets, proving the hypothesis that human input leads to noisy inputs and inaccurate targets. While avoiding attributes reported by individuals is infeasible in reality, data scientists should take note, and trust such data hesitantly.

7 Conclusion

In this paper, we present both statistical analysis and exploration through supervised learning of human input datasets gathered with self-reported questionnaire surveys. Our examination shows that data reported by people shows high noise, inconsistency, and potentially mislabeled targets. This should give caution to data scientists and developers relying on data provided by users.

The statistical analysis in Section 5 asserts the reliability of the surveys themselves with Cronbach's Alpha, and thus verifies that the choices of survey questions present no inherent noise. A k-nearest neighbors based noise cleaning method is then used to detect noise and correctness of target classes in the corresponding datasets. This examination shows high noise in the human input survey datasets, and low noise in the comparison datasets gathered through careful objective measurement.

Application of supervised learning in Section 6 proves the initial statistical analysis. In all of the utilized architectures, the survey datasets have significantly lower accuracy than those gathered through objective measurement.

Table 12: Supervised Learning Results

Dataset	Architecture	Iterations	Training Accuracy	Testing Accuracy
Linux Distros	MLP	50	36.15%	36.10%
	RBF	30	36.85%	35.32%
	PBNN	1	36.62%	36.10%
	RF	1	99.22%	36.10%
Pets	MLP	50	51.43%	49.52%
	RBF	54.6	59.29%	49.52%
	PBNN	1	87.38%	57.14%
	RF	1	99.05%	56.19%
Alcoholic Drinks	MLP	50	23.93%	21.01%
	RBF	65.2	30.92%	19.87%
	PBNN	1	39.24%	30.20%
	RF	1	99.73%	24.25%
Iris	MLP	400	97.83%	96.67%
	RBF	125.4	97.50%	95.33%
	PBNN	1	92.83%	92.67%
	RF	1	99.50%	94.67%
Cancer	MLP	200	89.16%	86.76%
	RBF	62	96.75%	96.14%
	PBNN	1	89.98%	89.65%
	RF	1	99.82%	95.79%
Random	MLP	100	14.50%	13.00%
	RBF	111.8	40.25%	8.00%
	PBNN	1	36.25%	11.00%
	RF	1	100.00%	5.00%

In conclusion, people are inconsistent, and the inherent noise of human thought easily bleeds into any data they provide. It is not uncommon for computer scientists to rely on data reported by individuals. This data must be treated as noisy and potentially erroneous. With this knowledge, proper measures can be taken for effective learning, and analysis can reasonably present such data as questionable.

References

[1] Justin Lovinger. Clever surveys. `https://www.cleversurveys.com/`. Accessed: 2016-12-30.

[2] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[3] Mitra Basu and Tin Kam Ho. *Data complexity in pattern recognition*. Springer Science & Business Media, 2006.

[4] José A Sáez, Bartosz Krawczyk, and Michał Woźniak. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, 57:164–178, 2016.

[5] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[6] Michael R Smith and Tony Martinez. Improving classification accuracy by identifying and removing instances that should be misclassified. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2690–2697. IEEE, 2011.

[7] José Salvador Sánchez, Ricardo Barandela, Ana I Marqués, Roberto Alejo, and Jorge Badenas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022, 2003.

[8] Ricardo Barandela and Eduardo Gasca. Decontamination of training samples for supervised pattern recognition methods. In *Advances in Pattern Recognition*, pages 621–630. Springer, 2000.

[9] Yuan Jiang and Zhi-Hua Zhou. Editing training data for knn classifiers with neural network ensemble. In *Advances in Neural Networks–ISNN 2004*, pages 356–361. Springer, 2004.

[10] Jakramate Bootkrajang and Ata Kabán. Multi-class classification in the presence of labelling errors. In *ESANN*. Citeseer, 2011.

[11] Dietmar Harhoff and Timm Körting. Lending relationships in germany–empirical evidence from survey data. *Journal of Banking & Finance*, 22(10):1317–1353, 1998.

[12] David De Vaus. *Surveys in social research*. Routledge, 2013.

[13] Dennis F Thompson. Deliberative democratic theory and empirical political science. *Annu. Rev. Polit. Sci.*, 11:497–520, 2008.

[14] Dirk van Kampen. The 5-dimensional personality test (5dpt): Relationships with two lexically based instruments and the validation of the absorption scale. *Journal of personality assessment*, 94(1):92–101, 2012.

[15] Matthias Burisch. Approaches to personality inventory construction: A comparison of merits. *American Psychologist*, 39(3):214, 1984.

[16] Jorge-Luis Reyes-Ortiz, Davide Anguita, Alessandro Ghio, and X Parra. Human activity recognition using smartphones data set. *UCI Machine Learning Repository*, 2013.

[17] Matthias Pfisterer Robert Detrano Andras Janosi, William Steinbrunn. Heart disease data set. *UCI Machine Learning Repository*, 2013.

[18] Robert M Gonyea. Self-reported data in institutional research: Review and recommendations. *New directions for institutional research*, 127:73, 2005.

[19] Lana D Harrison. The validity of self-reported data on drug use. *Journal of Drug Issues*, 25(1):91–111, 1995.

[20] Mireille NM van Poppel, Henrica CW de Vet, Bart W Koes, Tjabe Smid, and Lex M Bouter. Measuring sick leave: a comparison of self-reported data on sick leave and data from company records. *Occupational Medicine*, 52(8):485–490, 2002.

[21] Shouhong Wang. Classification with incomplete survey data: a hopfield neural network approach. *Computers & operations research*, 32(10):2583–2594, 2005.

[22] Chao Lu, Xue-Wei Li, and Hong-Bo Pan. Application of extension neural network for classification with incomplete survey data. In *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, volume 3, pages 190–193. IEEE, 2006.

[23] Roberto Tagliaferri, Giuseppe Longo, Leopoldo Milano, Fausto Acernese, Fabrizio Barone, Angelo Ciaramella, Rosario De Rosa, Ciro Donalek, Antonio Eleuteri, Giancarlo Raiconi, et al. Neural networks in astronomy. *Neural Networks*, 16(3):297–319, 2003.

[24] Martin T Hagan, Howard B Demuth, Mark H Beale, et al. *Neural network design*. Pws Pub. Boston, 1996.

[25] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[26] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[27] László Tóth. Phone recognition with deep sparse rectifier neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6985–6989. IEEE, 2013.

[28] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 1, 2013.

[29] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[31] David Lowe and D Broomhead. Multivariable functional interpolation and adaptive networks. *Complex syst*, 2:321–355, 1988.

[32] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

[33] Ying Tan, Jun Wang, and Jacek M Zurada. Nonlinear blind source separation using a radial basis function network. *Neural Networks, IEEE Transactions on*, 12(1):124–134, 2001.

[34] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[35] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[36] Aman Mohammad Kalteh, Peder Hjorth, and Ronny Berndtsson. Review of the self-organizing map (som) approach in water resources: Analysis, modelling and application. *Environmental Modelling & Software*, 23(7):835–845, 2008.

[37] Ke Zhi Mao, K-C Tan, and Wee Ser. Probabilistic neural-network structure determination for pattern classification. *Neural Networks, IEEE Transactions on*, 11(4):1009–1016, 2000.

[38] Meijuan Gao and Jingwen Tian. Web classification mining based on radial basic probabilistic neural network. In *Database Technology and Applications, 2009 First International Workshop on*, pages 586–589. IEEE, 2009.

[39] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

[40] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[41] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):1, 2006.

[42] Victor Francisco Rodriguez-Galiano, Bardan Ghimire, John Rogan, Mario Chica-Olmo, and Juan Pedro Rigol-Sanchez. An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:93–104, 2012.

[43] Paul E Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.

[44] Helmut Schmid. Probabilistic part-ofispeech tagging using decision trees. In *New methods in language processing*, page 154. Routledge, 2013.

[45] Biswajeet Pradhan. A comparative study on the predictive ability of the decision tree, support vector machine and neuro-fuzzy models in landslide susceptibility mapping using gis. *Computers & Geosciences*, 51:350–365, 2013.

[46] Stephen Milborrow. Titanic decision tree. `https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:CART_tree_titanic_survivors.png`.

[47] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[48] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[49] scikit-learn. http://scikit-learn.org.

[50] TheStoat. Linux distributions. https://www.cleversurveys.com/surveys/survey-5763859801440256_5629499534213120/responses/-linux-distributions. Accessed: 2016-12-30.

[51] Lewis R Goldberg. The structure of phenotypic personality traits. *American psychologist*, 48(1):26, 1993.

[52] Paul T Costa and Robert R McCrae. The revised neo personality inventory (neo-pi-r). *The SAGE handbook of personality theory and assessment*, 2:179–198, 2008.

[53] Nicholas A Turiano, Daniel K Mroczek, Jan Moynihan, and Benjamin P Chapman. Big 5 personality traits and interleukin-6: Evidence for healthy neuroticism in a us population sample. *Brain, behavior, and immunity*, 28:83–89, 2013.

[54] PaintingInAir. What pet should i get? https://www.cleversurveys.com/surveys/survey-5709198289534976_5629499534213120/responses/-what-pet-should-i-get. Accessed: 2016-12-30.

[55] AvinaDiviri. Alcoholic drink predictor. https://www.cleversurveys.com/surveys/survey-6024271184789504_5668600916475904/responses/-alcoholic-drink-predictor. Accessed: 2016-12-30.

[56] M. Lichman. UCI machine learning repository, 2013.

[57] Olvi L Mangasarian, R Setiono, and WH Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. *Large-scale numerical optimization*, pages 22–31, 1990.

[58] Mohsen Tavakol and Reg Dennick. Making sense of cronbach's alpha. *International journal of medical education*, 2:53, 2011.

[59] J Martin Bland and Douglas G Altman. Statistics notes: Cronbach's alpha. *Bmj*, 314(7080):572, 1997.

Table 13: MLP Iris Confusion Matrix

10	0	0
0	9.2	0.8
0	0.2	9.8

Table 14: RBF Iris Confusion Matrix

10	0	0
0	9.2	0.8
0	0.6	9.4

Table 15: PBNN Iris Confusion Matrix

10	0	0
0	9.2	0.8
0	1.4	8.6

Table 16: RF Iris Confusion Matrix

10	0	0
0.2	9.2	0.6
0.2	0.6	9.2

Table 17: MLP Cancer Confusion Matrix

60	11.4
3.6	38.8

Table 18: RBF Cancer Confusion Matrix

70	1.4
3	39.4

Table 19: PBNN Cancer Confusion Matrix

71.4	0
11.8	30.6

Table 20: RF Cancer Confusion Matrix

70.4	1
3.8	38.6

Table 21: MLP Alcoholic Drinks Confusion Matrix

1	0	0	0	0	8	0	0	0	0	0
0.4	0	0	0	0	1.8	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
1.6	0	0	0	0	3.2	0	0	0	0	0
0.2	0	0	0	0	1.4	0	0	0	0	0
1.8	0	0	0	0	6.8	0	0	0	0	0
0.8	0	0	0	0	2.4	0	0	0	0	0
0.4	0	0	0	0	3	0	0	0	0	0
0.2	0	0	0	0	0.8	0	0	0	0	0
1	0	0	0	0	2	0	0	0	0	0
0.2	0	0	0	0	0.2	0	0	0	0	0

Table 22: RBF Alcoholic Drinks Confusion Matrix

4.2	0	0	3.2	0.2	0.4	0.6	0.4	0	0	0
0.2	0.4	0	1.2	0	0	0.2	0	0.2	0	0
0	0	0	0	0	0	0	0	0	0	0
1	0.2	0	1.2	0	1.2	0.2	0.2	0.4	0.4	0
0.4	0	0	1.2	0	0	0	0	0	0	0
2.2	0.4	0	3.4	0	1	0.8	0.2	0.4	0.2	0
1.2	0	0	1	0	0	0.4	0.2	0.2	0.2	0
0.8	0	0	1.2	0	1	0	0.2	0	0.2	0
0.2	0	0	0.2	0	0	0	0	0	0.6	0
0.6	0	0	1.4	0	0.6	0	0	0.4	0	0
0.2	0	0	0.2	0	0	0	0	0	0	0

Table 23: PBNN Alcoholic Drinks Confusion Matrix

8.8	0	0	0	0	0.2	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0.8	0	0	0	0	0
1.4	0	0	0	0	0.2	0	0	0	0	0
6.2	0	0	0	0	2.4	0	0	0	0	0
2.2	0	0	0	0	1	0	0	0	0	0
3.2	0	0	0	0	0.2	0	0	0	0	0
0.6	0	0	0	0	0.4	0	0	0	0	0
2.4	0	0	0	0	0.6	0	0	0	0	0
0.2	0	0	0	0	0.2	0	0	0	0	0

Table 24: RF Alcoholic Drinks Confusion Matrix

9	0	0	0	0	0	0	0	0	0	0
2.2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
4.8	0	0	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0	0	0
8.6	0	0	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0	0	0
3.4	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
0.4	0	0	0	0	0	0	0	0	0	0

Table 25: MLP Random Confusion Matrix

0	0	0	0	0	0.2	0	0	0.4	0.4
0	0	0	0	0	0	0	0	0.4	0.6
0	0	0	0	0.6	0.2	0	0	0.4	0.4
0	0	0	0	0.2	0.2	0	0	0.4	0.6
0	0	0	0	0.6	0.4	0	0	0.4	1
0	0	0	0	0.6	0.6	0	0	0.8	1
0	0	0	0	0.2	0.4	0	0	0	1.2
0	0	0	0	0.6	0.4	0	0	0.2	0.6
0	0	0	0	0.4	0.8	0	0	0.6	1.4
0	0	0	0	0.8	0.8	0	0	0.4	0.8

Table 26: RBF Random Confusion Matrix

0	0	0	0	0.2	0.4	0	0.2	0.2	0
0.2	0	0	0.2	0	0.2	0	0	0.2	0.2
0	0.2	0	0.2	0.4	0.4	0	0	0.2	0.2
0	0	0	0	0.4	0.2	0.4	0.2	0.2	0
0	0	0	0	0.4	0.2	0.4	0.2	0.6	0.6
0.2	0.2	0	0	0.2	0.2	0.2	0.2	1	0.8
0	0	0	0	0.4	0.2	0.2	0.4	0.4	0.2
0	0.2	0	0	0.2	0.6	0.4	0	0.2	0.2
0	0	0	0	0.6	1	0.2	0.2	0.6	0.6
0	0.2	0	0	0.6	0.6	0.2	0.2	0.8	0.2

Table 27: PBNN Random Confusion Matrix

0	0	0	0	0	0.4	0	0	0.2	0.4
0	0	0	0	0	0.2	0	0	0.2	0.6
0	0	0	0	0	0.2	0	0	1.2	0.2
0	0	0	0	0	0.4	0	0	1	0
0	0	0	0	0	0.8	0	0	1.4	0.2
0	0	0	0	0	0.4	0	0	1.2	1.4
0	0	0	0	0	0.6	0	0	0.4	0.8
0	0	0	0	0	0.4	0	0	1.2	0.2
0	0	0	0	0.2	1	0	0	1.6	0.4
0	0	0	0	0	1	0.2	0	1.4	0.2

Table 28: RF Random Confusion Matrix

1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
1.6	0	0	0	0	0	0	0	0	0
1.4	0	0	0	0	0	0	0	0	0
2.4	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	0	0
1.8	0	0	0	0	0	0	0	0	0
3.2	0	0	0	0	0	0	0	0	0
2.8	0	0	0	0	0	0	0	0	0