jamk.fi

Regular data analysis methods

Jesse Raitapuro

Bachelor's thesis
February 2019
Technology, communication and transport
Degree Programme in Information and Communications Technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Raitapuro, Jesse	Bachelor's thesis	February 2019
		Language of publication: English
	Number of pages: 62	Permission for web publication: yes

Title of publication

Regular data analysis methods

Degree programme

Information and Communications Technology

Supervisor(s)

Rantonen Mika, Saarisilta Juha

Assigned by

Moventas Gears Oy

Abstract

Data analysis project was assigned by windmill gear constructing company Moventas Gears Oy. The purpose was to find new information from windmill gear data, which Moventas Gears Oy provided. Energy price is decreasing constantly on the wind energy field, so the project was an attempt to find new cost savings for Moventas Gears Oy.

The data analysis was conducted by using regular tools, methods and formulas from common data science field. All used methods and how they work with mathematical formulas is explained. The data was viewed in multiple different perspectives to get a result that would be the most reliable. Variable name anonymization was made for the data provided, because the naming was considered as confidential information.

All results of the different data analysis method supported the same conclusion: single data variables were unable to give new supporting information against different variables under the scope. When looking at multiple variable affecting at interesting variables, no linear relationship was found. Non-linear multiple variable relationship against single variables under the scope research would have taken so long time, that was not done.

This data analysis project was instructive research for real data, even though a new relationship between the data points was not found. There are some methods left to use for this case, however, the time limits were exceeded. The data analysis methods that are left could be used on a new research on same data.

Keywords/tags (subjects)

Data analysis formulas, data analysis methods, data mining, Python

Miscellaneous (Confidential information)

Data naming is considered as confidential information and they are changed to random names

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Raitapuro, Jesse	Opinnäytetyö, AMK	Helmikuu 2019
	Sivumäärä 62	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: kyllä

Työn nimi
Perinteiset data-analytiikkamenetelmät

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Mika Rantonen, Juha Saarisilta

Toimeksiantaja(t)
Moventas Gears Oy

Tiivistelmä

Data-analytiikkaprojekti on toteutettu tuulimyllyjen vaihteita valmistavalle Moventas Gears Oy:lle. Tarkoitus oli etsiä uutta tietoa tuulimyllyjen vaihdedatasta, jonka Moventas Gears Oy toimitti. Energian hinta putoaa koko ajan tuulienergian tuotantoalalla, joten projektin tarkoitus oli etsiä säästöjä tuotannossa Moventas Gears Oy:lle.

Data-analytiikka toteutettiin käyttäen perinteisiä työkaluja, menetelmiä ja kaavoja yleisesti käytössä olevilla menetelmillä. Kaikki menetelmät on selitetty, kuinka ne toimivat matemaattisten kaavojen kera. Dataa on katseltu monesta eri perspektiivistä, jotta tulokset olisivat mahdollisimman luotettavia. Muuttujien nimet on anonymisoitu datalle, koska nimeäminen niille oli luottamuksellista tietoa.

Kaikki eri data-analytiikan menetelmät johtivat samaan lopputulokseen, että yksittäiset muuttujat datassa eivät voi selittää uutta tietoa eri muuttujista, jotka olivat tarkastelussa. Kun monen eri muuttujan vaikutusta tutkittiin kiinnostaviin muuttujiin, ei lineaarista suhdetta löytynyt. Monen muuttujan ei-lineaaristen vaikutusten tutkiminen kiinnostaviin muuttujiin olisi vienyt liikaa aikaa, joten sitä ei tehty.

Data-analytiikkaprojekti oli opettavainen tutkimus oikealle datalle, vaikka uutta tietoa datapisteiden vaikutuksista toisiinsa ei löytynyt. Muutamia menetelmiä jäi käyttämättä, mutta aikaraja tuli vastaan. Data-analytiikkamenetelmät jotka jäivät jäljelle voisi toteuttaa uuteen tutkimukseen samaan dataan.

Avainsanat (asiasanat)
Data-analytiikan kaavat, data-analytiikan menetelmät, tiedon louhiminen, Python

Muut tiedot (salassa pidettävät liitteet)
Datan nimeämispolitiikka on salaista tietoa, joten data on nimetty satunnaisesti

Contents

Figures

Tables

Terminology

DataFrame	Format for handling multiple dimensional data in Python
Formula	Mathematical equation
Jupyter Notebook	Development environment, mostly used with Python
Method	Way for processing
Plot	Figure generated from data
Python	Programming language
Scalar	Single value from data series
Variable	Data feature

1 Introduction

This thesis was assigned by Moventas Gears, a windmill gear construction company. Moventas Gears has multiple different locations; and this particular project was carried out at the factory/headquarters located in Jyväskylä.

Moventas Gears interest in data analytic/research part was in gathering new information about the relationships between different windmill gear parts data and how they affect the outcome. A great amount of different windmill gear data was provided by Moventas Gears and domain knowledge was needed to get a better understanding of the data and its different variables. Without any knowledge or specific target about what was wanted from data, preliminary data analysis and research would take too long for every variable, because then it would be more like bruteforcing some information out of the data. Hence, some specific variables were given by Moventas Gears, and the focus was on all different data analytic and research parts.

Data analytics and research focused on traditional methods instead of neural networks. All analysis was conducted in Python 3 programming language with its third party supporting data analysis, research and visualization libraries e.g. Pandas, Numpy, Matplotlib, Statsmodels, Hypertools and Scikit-learn. All used libraries were open source software. Data analytic methods for linear and non-linear data were used to provide multiple perspectives for the analysis.

All data specific field names (variable names) and specific data are considered private and will handled with aliases such as *"data1-x", "data1-y"* etc.

Data used in analysis can be thought as described in Table 1:

Table 1 Example structure from data

Data1-x(1)	*Data1-x(2)*	*Data1-y(1)*	*Data1-i(1)*	*Data1-j(1)*	*Data1-j(2)*
0.6	0.02	0.006	1.1	2.2	2.6
0.2	0.06	0.025	0.5	0.5	0.1

Two different datasets were provided with similar variables, called *data1* and *data2* in this thesis. The data shape was hundreds of rows and few hundreds of columns. The data shape varied in the analysis because of gaps in the data.

Examples from source code (Python 3) are not located in the actual documentation to keep things clear, instead one Jupyter Notebook from decision tree variable analysis example is provided as an attachment (Appendix 1) and one Python 3 script from generating tree models (Direct source code, not Jupyter Notebook) is also found as attachment in Appendix 2.

2 Data analysis

Data analysis is a process for gathering new information out of data and a main component of data mining. Data analysis is usually conducted, when the purpose is to improve some already existing process, get some completely new information or evaluate a process. (Galetto 2016; Imanuel n.d.)

> *"Data analysis involves asking questions about what happened, what is happening, and what will happen (predictive analytics)."*
> *(Galetto 2016.)*

2.1 Data gathering

This phase is most of the time relatively easy, since nothing needs to be done for the data. Of course, there are exceptions, for example if there is a process and data is to be collected from that, it does not have any data collecting/generating system attached to it. Collecting data can also be just downloading data from a website or receiving it from a client. If one has a process, the data of which data one likes to collect, then one would need to set up the whole thing, and that can take a very long time. After the data is gathered, one should just save it securely for upcoming phases in data analysis process.

2.2 Clean and integrate data

Sometimes data can be very sparse and contain plenty of unnecessary data and outliers (Sridhar 2018.). Data can also be in a wrong shape in provided files, so that it needs to be fixed. Data quality needs to be improved using data interpolation or removal methods. Gaps in data can be filled with interpolation, if one knows how the data behaves. Empty and duplicate parts can be removed if it is not possible to use interpolation methods or analyze only certain parts of data. Clean and integrate data part can easily take 50-60% of a project's total worktime.

It is possible to even get new data so fast that the user does not have enough processing power to clean and integrate data fast enough, so it needs to be decided which data will be completely ignored (Granville 2014, 46).

2.3 Evaluate data

When data cleaning and integrating part are finished, evaluation is needed to find out if data can be used in the analytic phase. There are some methods for this such as histograms, standard deviation, mean value and skewness (Imanuel n.d.).

Depending on the data, sometimes data will have a lot of false information in it, such a social media platform with user feedback and actions. This can cause bad results from machine learning algorithms, so some mechanism needs to be used on that kind of platforms to detect spam and things like that. (Granville 2014, 47-48).

This phase can also have some hard things to implement. Example how to visualize and check data if you have very much information in it, like millions of points (Granville 2014, 57).

User should also think about the used features in the data for the problem, it is not needed to use every feature in the data for getting good results. In fact, some bad features can hide effectiveness from proper features in dataset. (Granville 2014, 68-70).

If the data looks fine and has nothing strange, then it is possible to start selecting different data analysis methods. If data is skewed, then it is needed to go back to clean

and integrate data phase, and if that is not enough, then it is required to collect more data.

2.4 Analytic phase

The analytic phase can be divided into two categories: initial data analysis phase and the main analysis phase (Imanuel n.d.).

When starting the analytic phase, the user should pick up proper tools for the project. The tools depend on what kind of the project it is, e.g. how much data users have, what kind of data it is (Live stream or data dump), what is needed to get out of from the data etc. (Granville 2014, 113-114)

The needed tools will include a machine learning algorithm (e.g. decision tree or neural network) and a visualization tool. These tools depend on the problem. (Granville 2014, 115-117)

In initial analytics, there are multiple measures that can be taken, for example:

- Correlation analysis
- Univariate statistics
- Different kind of plots
- Associations
- Computing new variables

(Imanuel n.d.)

The main analysis contains data analytic methods used for generating end results, which provides new information gained from data, e.g. models and a check on the stability of the results. (Ibid.)

3 Data analysis formulas and methods

3.1 Min-Max scaling

Min-Max scaling is used when there are variables with different value scales, and the selected algorithms/methods require data that needs to be in a similar range. (Asaithambi 2017.)

Min-Max scaling is a way to scale data series values between in a certain range, for example from 0 to 1, where 0 is the lowest value from data series and 1 is the highest value from data series. This is also known as data "normalization". Min-Max scaling is performed using the following formula:

$$Xnorm = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

Here *X* is a single scalar value from data series. As example, Min-Max scaling is conducted for the following data series:

$$[0, 10, 20]$$

This would lead for following outcome after Min-Max scaling:

$$[0, 0.5, 1]$$

(Raschka 2014.)

3.2 Standard deviation

Standard deviance is showing how far data series data is scattered out. It works for data series, for example a one-dimensional array such as:

$$[0, 1, 2, 3, 4, \dots, N]$$

The formula for calculating standard deviation is following:

$$\sqrt{\frac{(X1-M)^2 + (X2-M)^2, \dots, +(Xn-M)^2}{N}} \quad (2)$$

Here *X* is the scalar value in data series, *M* is the mean of data series values and *N* is the count of samples in data series. This formula is for the whole data series. If the data is a sample from larger data series, then the calculation is slightly different.

For the sample data it is necessary to remove one from count *N*, hence, the formula is presented as follows:

$$\sqrt{\frac{(X1-M)^2 + (X2-M)^2, \dots, +(Xn-M)^2}{N-1}} \quad (3)$$

The result from calculating standard deviation is single scalar value. A lower value means that the data points in array are grouped more closely to each other and higher means the data points are more spread out. (Standard Deviation and Variance n.d)

3.3 Removing outliers

The formula for detecting outliers:

$$|Xn - m([X1, X2, \dots, Xn])| > (x * std([X1, X2, \dots, Xn])) \quad (4)$$

First from value (Data groups one value) *Xn*, the mean(*m*) of column is subtracted. After that, the value is compared if it is smaller than the column's standard deviation (*std*) (Optionally deviation can be increased by coefficient *x*). If the value is smaller or the same as standard deviation (**x*), then value is accepted. If it is higher, then it will be removed from data (current row).

Depending on data accuracy, standard deviation with multiplier 0.5-3 is used, or also with a value greater than, and equal comparison is as follows:

$$|Xn - m([X1, X2, \dots, Xn])| \geq (2||3 * std([X1, X2, \dots, Xn])) \quad (5)$$

(Detect and exclude outliers in pandas dataframe. n.d.)

3.4 Manhattan distance

Manhattan distance is an algorithm, which works for *X* dimensional data arrays. It is used to detect how far the different data points are from each other in the data array; for example, if 2-D data series are gained with the following one data point: *[X, Y]* against another similar series with similar data points *[X, Y].* The algorithm for Manhattan distance is following:

$$|\mathrm{X1} - \mathrm{X2}| + |\mathrm{Y1} - \mathrm{Y2}|, \dots, + |\mathrm{N1} - \mathrm{N2}| = \text{Manhattan distance} \quad (6)$$

Where *Xn* is x-axis data point from array, and *Yn* is y-axis data point from array. All dimensional data points can be used in calculation (*Nn*). The result is the distance between the data points. For example, from point A:

$$(10, 20, 10)$$

To point B:

$$(10, 20, 20)$$

The result is 10 which shows that only single value is returned, instead of *N* dimensional data. (Polamuri 2015.)

3.5 Data interpolation

Data interpolation is method to fill sparse data with generated data. Data is generated within different interpolation algorithms such as linear- and polynomial interpolation. It is the value estimation between known values (Rouse 2015.).

Linear interpolation is used in one data analysis method (PCA to classified data) to fill gaps between sparse data. It is a quite simple algorithm and can (and will) cause errors for data that is being analyzed; hence, interpreting the results will be more uncertain.

If two different data points (Two-dimensional data points) are known (*x1*, *y1*) and (*x2*, *y2*), linear interpolation equation is following:

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \quad (7)$$

Where points (*x1*, *y1*) and (*x2*, *y2*) are known and *x* is between *x1* and *x2*. (Linear interpolation with Excel. n.d.)

When interpolating simpler one-dimensional data with one missing value, interpolation fill is following (Where *X* is the missing value):

$$[2, X, 6, 8] \rightarrow [2, 4, 6, 8]$$

3.6 Covariance and covariance matrix

Covariance is a way to calculate how much two different variables (Or dimension) vary from each other. The formula for calculating covariance:

$$\sigma(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})^2 \quad (8)$$

(Janakiev 2018.)

Where *n* is sample count, $x_i|y_i$ is scalar from one dimension and $\bar{x}|\bar{y}$ is the mean from one dimension´s values.

Covariance matrix is a matrix, which contains covariance values between *X* amounts of different variables. The formula for calculating covariance matrix, with two-dimensional data is as follows:

$$C = \begin{matrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{matrix} \quad (9)$$

(Ibid.)

3.7 Eigenvalue and eigenvector

Eigenvalues can be calculated from the following formula:

$$\det(\mathrm{A} - \lambda \mathrm{I}) = 0, \quad (10)$$

Where *A* is the filled matrix and *I* is the identity matrix. When solving equation, all solved λ values are going to be eigenvalues. Once the eigenvalues are found, then eigenvectors can be calculated. Eigenvector can be calculated from the following equation:

$$(A - \lambda I)x = 0, \quad (11)$$

x is eigenvector (for λ) to be calculated. Eigenvector is calculated with Gaussian elimination from that formula. (finding eigenvalues and eigenvectors n.d.)

Gaussian Elimination is way to solve systems of equations (Kase & Kuang n.d.).

3.8 Linear regression

> *Regression analysis is a way of mathematically sorting out which of those variables does indeed have an impact. It answers the questions: Which factors matter most? Which can we ignore? How do those factors interact with each other? And, perhaps most importantly, how certain are we about all of these factors?*

(Gallo 2015.)

The relationship between two variables, for example *X* and *Y* is defined from following linear regression equation:

$$\hat{y} = b_0 + b_1 x_1 \quad (12)$$

In that equation, $\hat{y}$ is dependent variable that is going to be predicted. b_1 is slope, x_1 is independent variable value and b_0 is constant value, usually called Y-intercept. For example, value of $\hat{y}$ from variable x_1. can be predicted. Information can be gained about how much variable y increases, when variables x_1 increases by one (1) unit.

Variable b_0 is calculated from this equation:

$$b_0 = \overline{y} - b_1 \overline{x} \quad (13)$$

Where $\overline{y}$ is mean of the dependent variable, b_1 is slope and $\overline{x}$ is mean of the independent variable.

In addition, $b_1 x_i$ can be calculated from this equation:

$$b_1 = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sum(x_i - \overline{x})^2} \quad (14)$$

Where $\overline{y}|\ \overline{x}$ is mean of the independent/dependent variable and $y_i|x_i$ is value of independent/dependent variable. This equation works also for multiple variable regression as follows:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 \quad (15)$$

(Bronshtein 2017; Foltz 2013; Foltz 2014.)

3.9 Clustering (Mean Shift)

Clustering algorithms are for labeling data from data series to different clusters. The meaning for that is to detect different groups from data, when those are unknown.

"Meanshift is a clustering algorithm that assigns the datapoints to the clusters iteratively by shifting points towards the mode".
(Meanshift Algorithm for the Rest of Us (Python) 2016.)

Centroids for each iteration are calculated from the following equation:

$$X_i^{t+1} = m(X_i^t) \tag{16}$$

Where *Xi* is one possible centroid, *t* is iteration count and function *m()* produces so called Mean Shift vector. Mean Shift vector is calculated from the following equation:

$$m(X_i) = \frac{\sum_{X_j \in N(X_i)} K(X_j - X_i) X_j}{\sum_{X_j \in N(X_i)} K(X_j - X_i)} \tag{17}$$

Where $N(X_i)$ are neighbor points of X_i, $K(X_j - X_i)$ is kernel to use in Mean Shift algorithm and $X_j - X_i$ is distance between those points. (Sklearn MeanShift documentation. n.d; Meanshift Algorithm for the Rest of Us (Python) *2016*.)

The algorithm automatically calculates count of clusters from given data and produces following kind of clusters (See Figure 1).

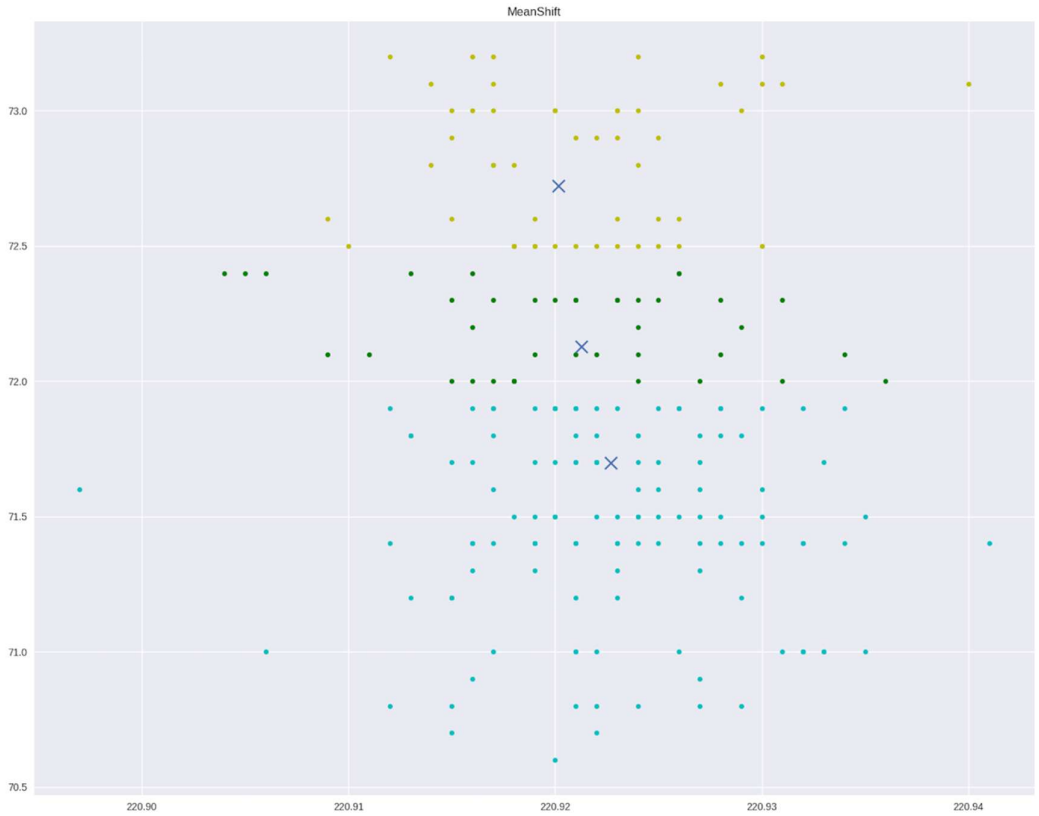


Figure 1 Clustering example

Based on Figure 1, three different clusters are created. Blue, yellow and green.

3.10 Dimension reduction (Principal component analysis (PCA))

The purpose of dimension reduction is to reduce the amount of dimensions from data, so it is easier to visualize data and increase the speed of machine learning algorithms. After dimension reduction, the same data is in smaller format and some data is lost on conversion. (Galarnyk 2017.)

Data loss is going to occur every time, when dimension reduction is performed to data. Data will have *X* amount less features than original, so data loss will depend on how many dimension are reduced. More reduced dimension will cause more loss of data.

PCA is a good method for finding patterns in high dimensional data or compressing data (Example images). The patterns are very hard to find from high dimensional data without reducing dimension (PCA fits this purpose well). (Smith 2002.)

3.11 Dimension reduction process (PCA)

First, each dimension´s scalars need to be reduced by mean value of scalar's own dimension values.

$$Scalar_{new} = Scalar - Mean(Scalar\ own\ dimension) \quad (18)$$

An example, when doing to one dimensional data, follows:

$$\begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \rightarrow \begin{matrix} -1 \\ 0 \\ 1 \end{matrix}$$

Then covariance matrix (See chapter 3.6) is calculated for data. After the covariance matrix is generated, then eigenvectors and eigenvalues (Eigenvalue and eigenvector) need to be calculated from the covariance matrix. Then feature vector is generated from eigenvectors and the largest column is selected from the feature vector. Now the reduced data can be created from the following equation:

$$FinalData = RowFeatureVector \times RowDataAdjust \quad (19)$$

(Ibid.)

Following is an example of doing dimension reduction from three (3) dimension data to two (2) dimension data for the following data serie:

$$[(A, B, C), (D, E, F), (G, H, I)]$$

The output would be the following:

$$[(J, K), (L, M), (O, P)]$$

3.12 Decision tree regressor

Decision tree regressor models are built from predictable variable (dependent variable) and feature variable(s) (independent variable(s)). These models can be used in regression and classification problems and also with extracting most effective independent variables from those tree models. (Sayad n.d a & b.)

Decision tree regressor models can be used in linear and non-linear data, however, they work better than most of regular methods for non-linear data (Nevavuori 2018a.).

Models are built from feature variables, their values and “decisions”, which leads to a structure that looks like a tree (Sayad n.d a & b.). That is why they are called decision tree models. An example of a decision tree structure can be seen in Figure 2.

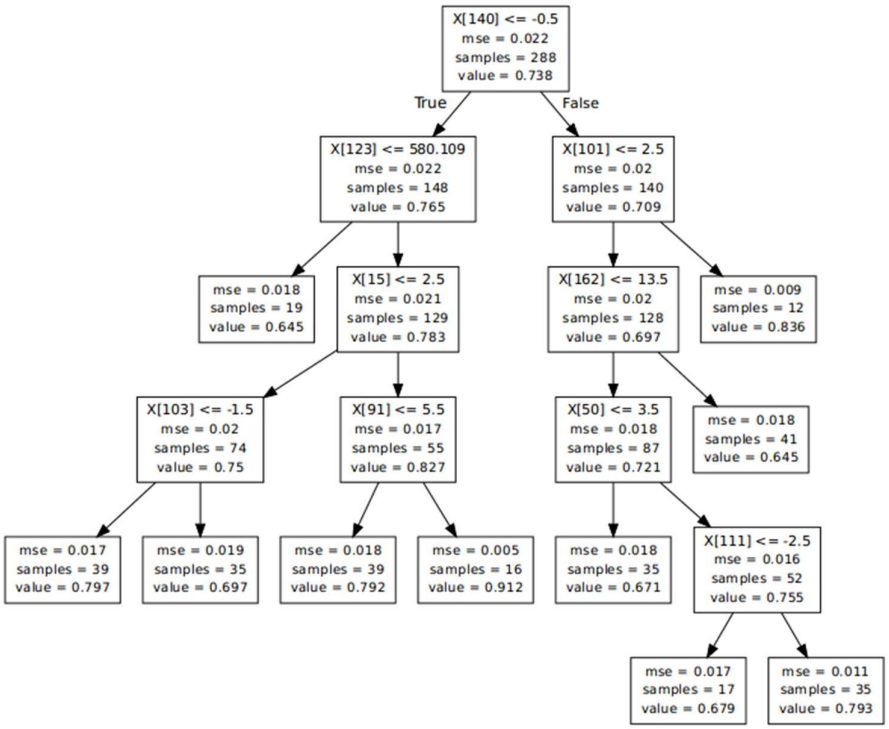


Figure 2 Decision tree example

Decision tree classifier is used predicting categorical data such as:

$$[dog, cat, horse]$$

Regressor for numeric data as follows:

$$[25, 100, 27, 43]$$

It is possible to extract the most affecting variables from the tree, when looking how much they affect result (For example, predicting value). Most affecting variables usually occur at the top of the tree more often than variables that does not affect that much. These features can be considered as most important variables in the created tree model. These models can also be used in predicting value/or classification tasks, however, this is not the purpose here; hence, this part is not discussed here any further. (Nevavuori 2018a.)

3.13 Construction of tree model

ID3 is the algorithm (core) used to construct decision trees. The construction of decision trees usually contains information gain (with entropy); however, with regression, the information gain is replaced with standard deviation reduction. (Sayad n.da; Rishab 2017.)

Standard deviation, when using one attribute reads as follows:

$$Standard\ deviation = S(T) = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} \quad (20)$$

Where *x* is one scalar value, $\bar{x}$ is mean of scalars and *n* is count of scalars.

As for two attributes

$$S(T,X) = \sum_{c\in X} P(c)S(c) \quad (21)$$

Where *P(c)* is $\frac{count\ of\ single\ type\ scalar\ in\ one\ attribute}{all\ scalar\ count\ in\ one\ attribute}$ and *S(c)* is same as the earlier mentioned *S(T)* for that one type of scalar in one attribute.

And standard deviation reduction formula comes from:

$$SDR(T,X) = S(T) - S(T,X) \quad (22)$$

The most important attribute is one with the highest standard deviation reduction. (Sayad n.da.)

3.14 Simple example of building tree model without math

The process of building a tree model consists of two kinds of variables, which are the earlier mentioned independent variable(s) and dependent variable(s). An example follows in Table 2:

Table 2 Example datatable

Bark	Weight	Tail length	Dog
Yes	15 kg	10 cm	Yes

Yes	25 kg	3 cm	Yes
No	10 kg	20 cm	Other animal
Yes	10 kg	30 cm	Other animal

Following variables can be considered as independent variables: bark, weight and tail length. The last variable, not an independent variable, is processed as a dependent variable (Dog).

When starting a tree model construction process, the first job is to select an independent variable that has most effect on a dependent variable and select it as the first node. In this case bark called column (First column in Table 2) is most effecting to result, so Table 2 looks like Table 3 now.

Table 3 Shrinked datatable

Weight	Tail length	Dog
15 kg	10 cm	Yes
25 kg	3 cm	Yes
10 kg	20 cm	Other animal
10 kg	30 cm	Other animal

After this operation is done, it is looped until there are no independent variable columns left. The output for this data is illustrated in Figure 3:

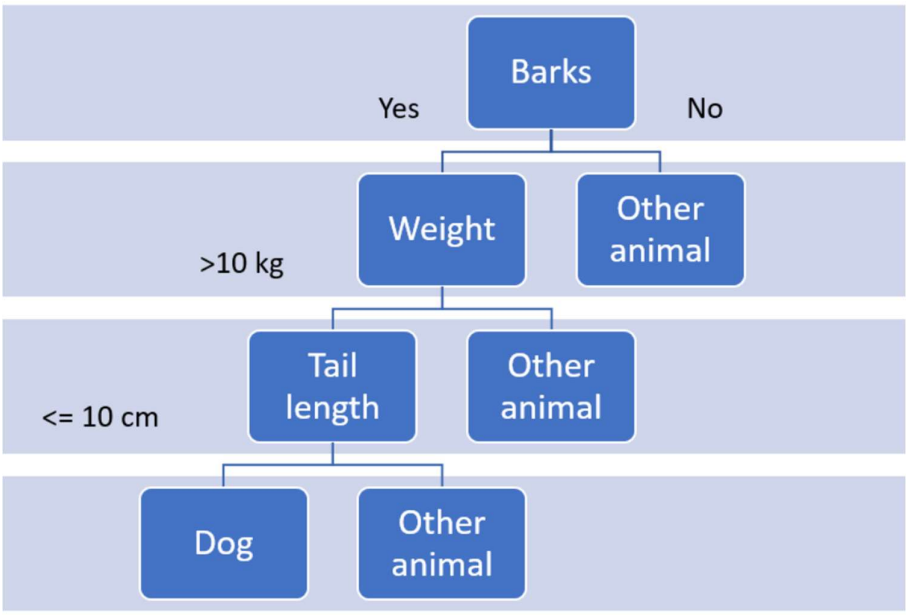


Figure 3 Decision tree construction

It is possible to see in Figure 3 structure that the most meaningful independent variable is named barks, because it is on the at top of the tree.

4 Data mining

4.1 General

Data mining is a process for gathering new information from data, which can help to create a more efficient process (Example manufacturing). Data can possibly have hidden information in it, which could be gathered with different methods focused on analyzing the data. Understanding the data and what is wanted from it for the business aspect is a requirement. (Data Mining Processes n.d.)

The part of data mining in this project is to clarify methods and their results. The data used in the analysis is a generated CSV file from the local database containing all data provided by Moventas Gears.

Before the data analysis methods could be carried out, decisions how to integrate data needed to be done. Multiple ways were used to get integrated data and have a good DataFrame for data analysis.

A good DataFrame should contain almost all rows filled with data, so no further actions are required to integrate data. Mostly, when working with real data, this is not reality. Almost every dataset, which is created from real processes has some errors occurring when saving/creating data. This is the reason why removing and filling processes are needed. All variables (example "*A*") from data cannot be used, because so many values/or all values are missing from some of the variables. The variables with few values or no values at all need to be removed when running algorithms. Data filling methods somewhat affect some analytic phases, because then the data is not "real" and data analytic results are skewed. If all data is handled at once, then all rows should be deleted. The solutions for handling all data at once is only to handle the needed variables at the time, remove columns with a high count of missing data or use interpolation techniques, so it is not needed to remove so much data from DataFrame.

Examples from good and bad DataFrames are shown in Table 4 and Table 5

Table 4 Good DataFrame for analysis

	Variable1	Variable2	Variable3
0	5	562	0.000000010617872603038
1	3	221	0.0000186631548658056
2	7	566	0.0000000138793185720678
3	8	212	0.0000000106134122603038

Table 5 Bad DataFrame for analysis

	Variable1	Variable2	Variable3	Variable4	Variable5
0	5	562	5		
1		221	2	0.0000187	
2	7		3		
3		212	9	0.0000000106	

4.2 Method 1. Clustering (Meanshift) with standard deviation and Manhattan distance finding

When finding similar behavior variables from test results, similar behavior can help to choose variables which are possibly worth more accurate inspection. Information from variables with similar behavior can also help when using other data analysis methods.

When finding similar behavior between variables first clusters (Mean Shift algorithm is used as clustering method) are created to help select/process data, which has some different kinds of data groups in it. The cluster count is selected by an algorithm.

After clustering has been performed, then standard deviation of y-axis variable is calculated for every cluster. If a cluster's standard deviation has a certain amount of change (Static selected value, based on test results), then that data is selected for later processing. Data with similar clusters are not used, because they most likely do not contain any new info.

Following figures (Figure 4 and Figure 5) have examples about cluster deviation. Figure 4 shows that all data clusters (blue, yellow and green) have a similar deviation. For that reason, it is most likely that those variables do not have any new data between each other. Hence, those variables are not used in further analysis with this method. Figure 5 shows that the deviation of data clusters (green, blue and yellow) differs between each other, so variables with that kind of deviation might contain some new information and could be used in later processing.

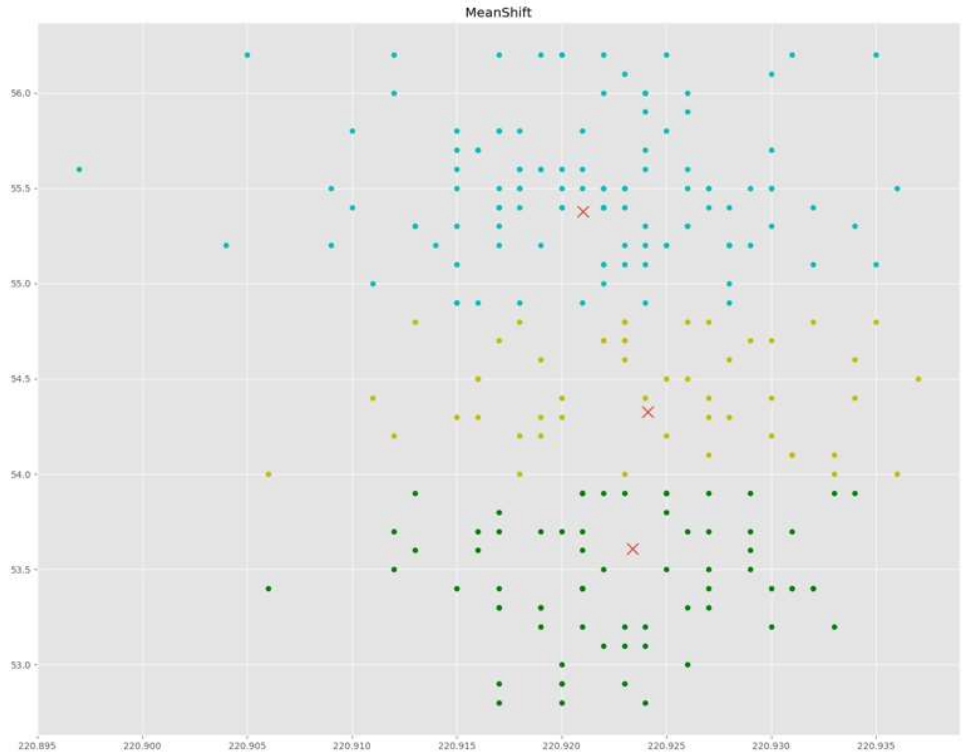


Figure 4 Not much deviation between clusters

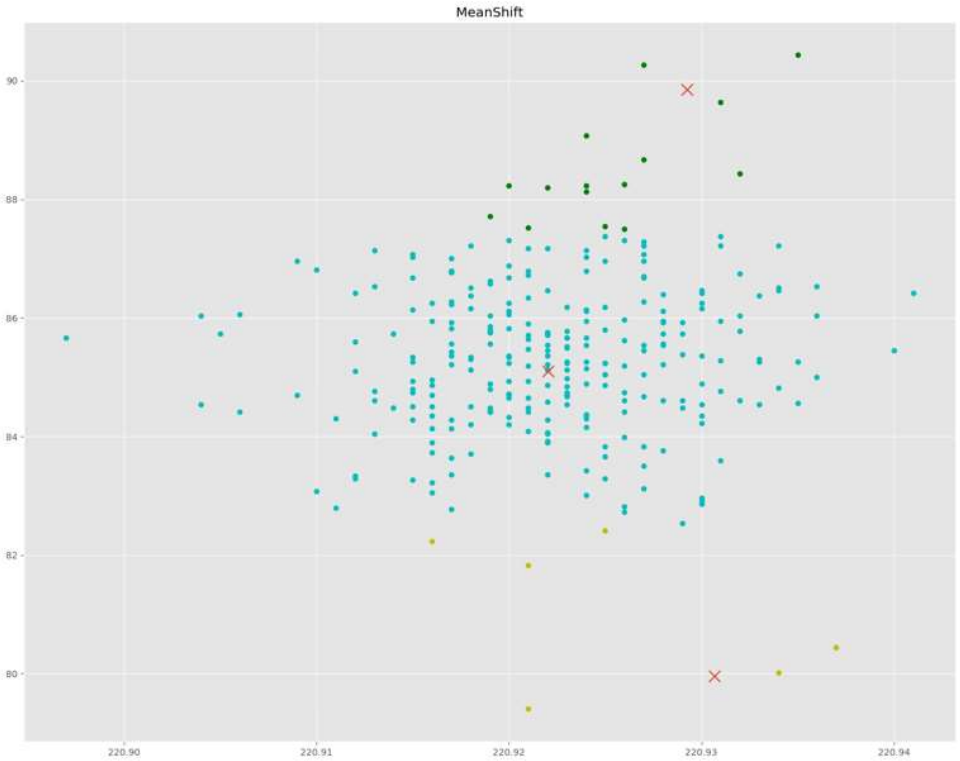


Figure 5 Great deal of deviation between clusters

The next phase is Manhattan distance algorithm, which checks a location difference from two data points as shown in Figure 6, between them (Polamuri 2015.).

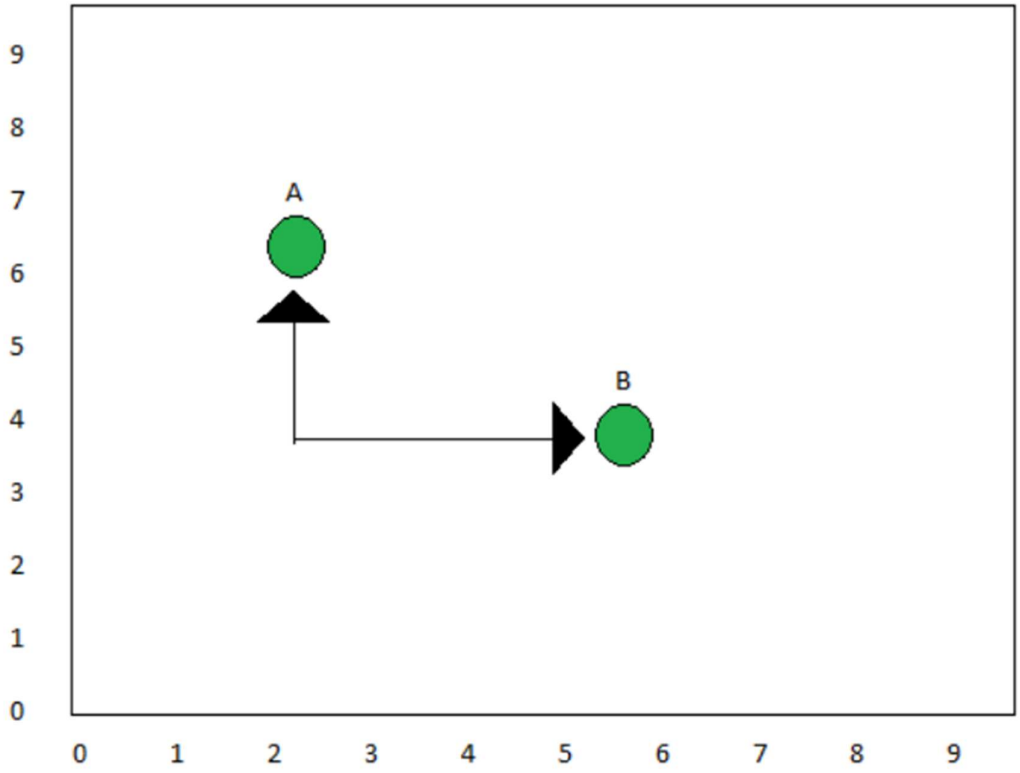


Figure 6 Two datapoint example for Manhattan distance

First, normalization is applied to data for creating similar data series environments. Without normalization this process does not work, because scale does not match. Then the distance between data points is calculated with Manhattan distance algorithm.

The whole process is performed to all of *data1(x)* values against *data1(y, z, i, j, k*) values and vice versa. Then series with less than 15 percentage difference are selected. Those combinations are worth inspecting or could be used for supporting other methods. The upcoming figure has an example with similar patterns (Figure 7).

These results make it possible to check which variables have some similarities in behavior patterns (See Figure 7):

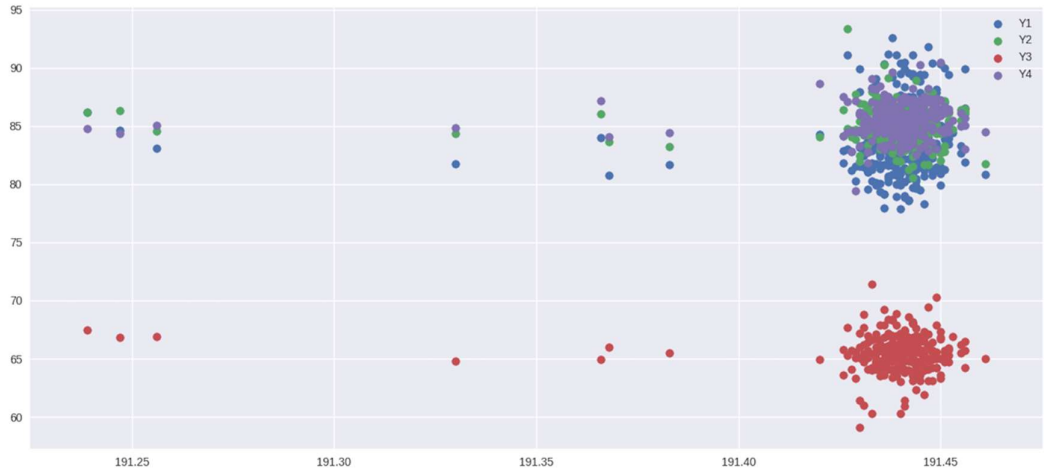


Figure 7 Data similarity example

Plotting the example variables at same figure (Figure 7), it is possible to see similar behavior (different colored groups have same style shape). The first combination found in the *data1* had 12 variables and the second combination had 19 variables. At this point result does not help to gather new information from the data.

4.3 One variable linear regression

The purpose of regression analysis was to check the coefficients of each of these variables and declare the relationship between variables more specifically. One variable regression analysis is processed only on selected input/output variables (Those variables were provided by Moventas Gears from *data1*), because straight correlation between each variable has already been carried out.

CSV format dataset is generated from these variables (An example from CSV can be seen in Table 6), so it is also manually possible to check results with supporting regression line images (Example of regression line can be seen in Figure 8).

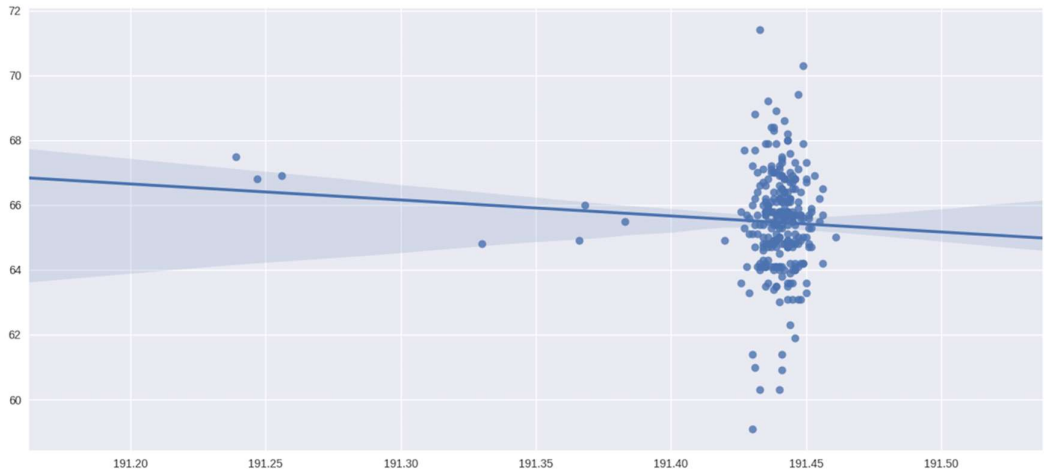


Figure 8 Regression line example

Example rows/columns from CSV (All data could not be inserted to this example due to row and columns count (Table 6)).

Table 6 Example of regression CSV data

	coefficient	dataLength	dependentVariable
0	-19.141798	56	Variable10
1	6.72067733	241	Variable10

The meanings of the abbreviations (This information is extracted from linear regression analysis) in columns in regression data are listed as follows:

- coefficient: Coefficient how much independent variable needs to increase to dependent variable increases one unit
- dataLength: Example 56 means that there are 56 *x* and 56 *y* data points
- dependentVariable: Variable which increases one unit if independent variable changes amount of coefficient
- eigenValue: How long data eigen vector is and how much data has stretched
- fValue: Indicates if variance between variables differs significantly
- independentVariable: Variable which increases independent variable by one unit if this is increased by a coefficient
- pValue: If p value is smaller than value *A* (most of times 0.05), result is usually ok, if it is higher, then null hypothesis cannot be passed and data needs to be checked for same kind of properties
- rSquared: How much regression line value explains based on data. Most of the time, bigger is better, because then data group is nearer to regression line

- rsquaredAdj: Percentage variance about independent variables, which affects the dependent variable. A higher count of independent variables usually decreases this result.
- ssrValue: Value which is mathematical operation residual (SST-SSE). No need to check this
- standardError: Coefficient average error

(Bronshtein 2017; F Statistic / F Value: Simple Definition and Interpretation. 2018; Foltz 2013.)

4.4 Multiple variable regression with PCA plots

Multiple variable regression analysis is used for finding relationships between multiple independent variable and single dependent variable. The method for this analysis is the same as in single variable regression (Ordinary least squares), however, the visualization for two dimensions is performed with PCA (More about dimension reduction in chapter 3.10).

First, combination lengths of six variables were created from independent variables (From *data1*). More than combinations of six would require much more computing power for calculating the results, the length of six total combinations is 1 344 904, so total processing count is 1 344 904 * 2=2 689 808. Multiple independent variable regression was computed to every combination of six against both dependent variables. Combinations with r-squared (Previous chapter 4.3) value more than 0.2 were picked for later processing. After the regression processing was ready, then components from combinations were picked to an individual list. From that list, new list combinations were generated with the length of two, three, four and five variables, which are processed through regression analysis again. This time the combinations were selected using the p-value (Previous chapter 4.3), because r-squared is so low and it does not give information anymore. When p-value is lower than 0.05, then images are printed from independent variables against dependent variable. Two-dimensional plots are possible with PCA dimension reduction. The required info is retained for plotting after reduction. When inspecting few of the plots, it is possible to see that there is no kind of reasonable coefficient (Shrink units are not shown in images, because they are not needed for this visualization purpose).

Upcoming figures (Figure 9 and Figure 10) are example plots of dimension reduced variable combinations. The purpose for those figures is only to show that when adding a new variable, the shape of data changes noticeably.

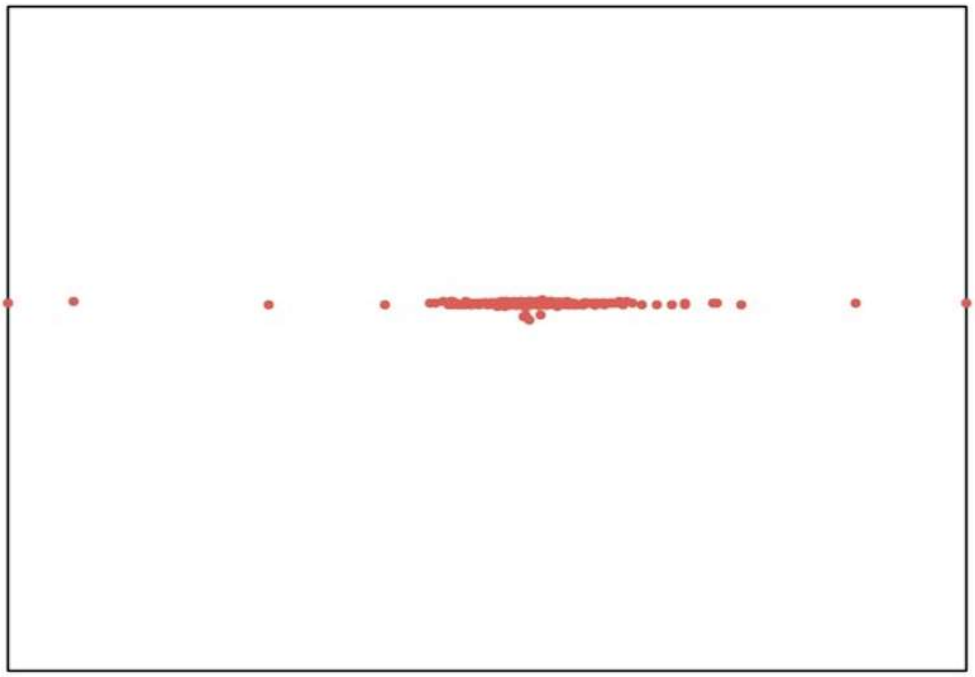


Figure 9 PCA example with only independent variables

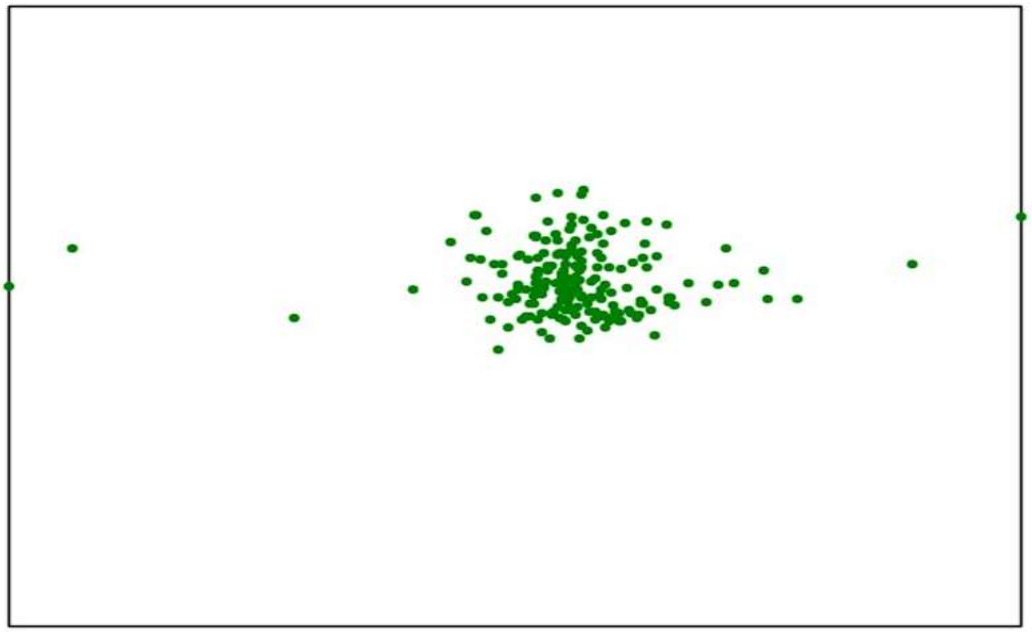


Figure 10 PCA example with independent variables and dependent variable

All results show that when independent variable plot is going somehow straight/linear, then with an added dependent variable, the image has no information and vice versa.

4.5 PCA to classified data

Purpose for using PCA to classified data was to detect how data variance differs between different ranks (ranked to three classes: worst, medicore and best based on sum of data rows).

The PCA dimension reduction method was used for data variables (From *data1*) and when PCA process was finished, the dimension reduced data was used for generating figures. (Galarnyk 2017.)

PCA is used at visualization of more than three variables or making machine-learning algorithms faster; in this case, it is used for visualization. The new shrinked components are not any "real" variables but same data in smaller format with some data loss. The purpose of this method is to check if data variables have significantly different values between each other, taken from top 60 highest values from rows (sum values of *data1*). After that, the largest value will have tag "worst", middle values

have "medicore" tag and lowest "best" tag. Now the data have different labels. Then dimension reduction is applied to all columns in the dataset to make two new "principal" components for plotting reasons, then plotting those 60 rows in figure will show rows as new components. The problem with processing data is missing values. All data needs to be solid before the analysis; thus, filling methods are used; static variable to missing values, linear interpolation to rows and linear interpolation to columns.

The result from static data fill (See Figure 11):

- 17662-17707 range (Ranking value) and standard deviation: 11.76, with static value of 25. That range is already quite high considering that there is not much variance between results. There is no extra info in this result.

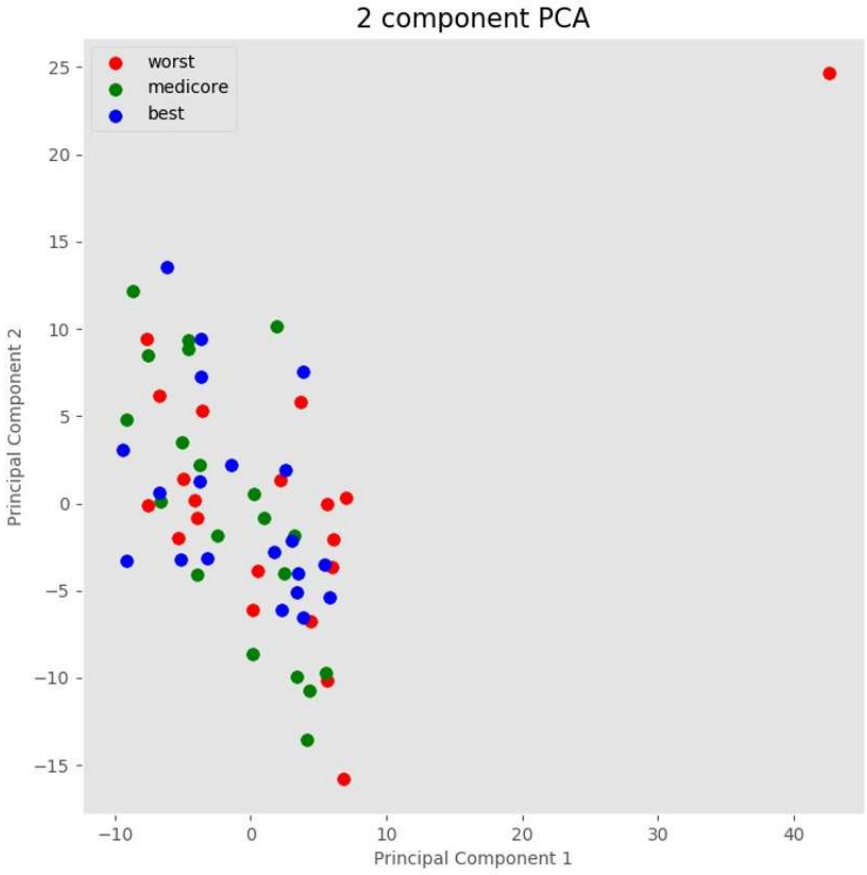


Figure 11 PCA, static data fill

The result from interpolation applied to rows (See Figure 12):

- 17670-35826 range (Ranking value), mean 24937 and standard deviation 8936, with interpolation applied to rows. This range is very high, yet, still the result have similar look (different colored points) hence, there is so no extra info (except one outlier).

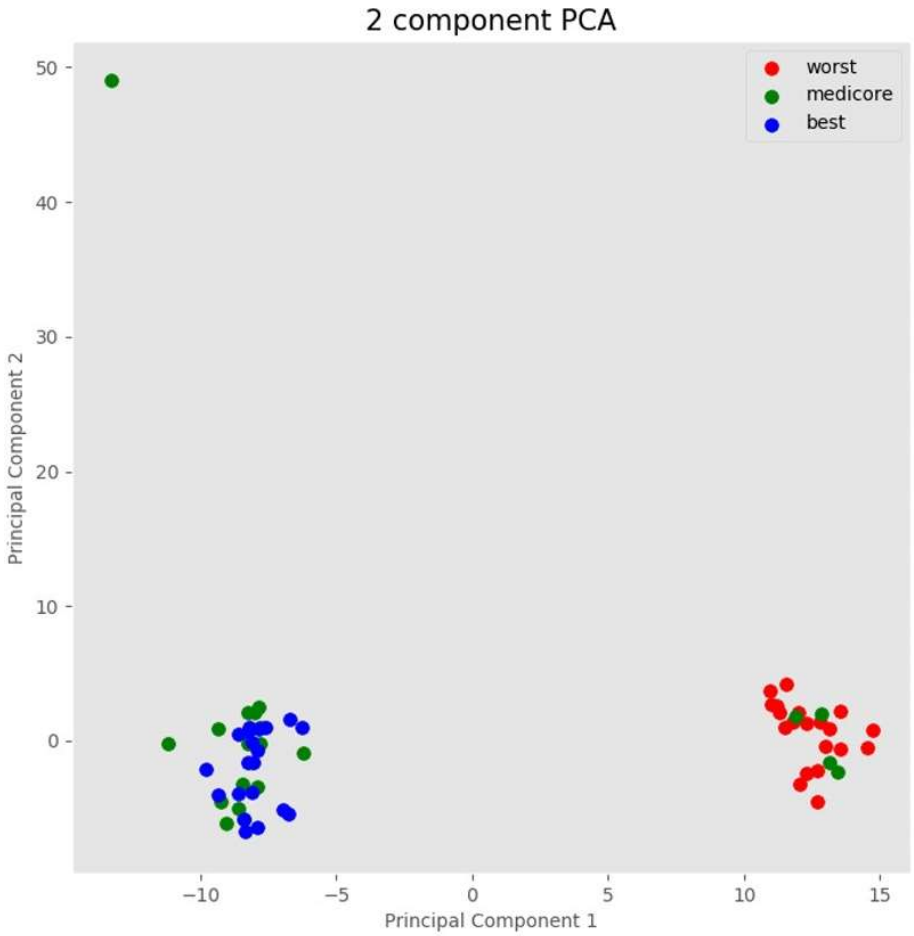


Figure 12 PCA, interpolation applied to rows

The result from interpolation applied to columns (See Figure 13):

- 17666-17707 range (Ranking value), mean 17677 and standard deviation 11, with interpolation applied to columns. The range is not very high but still results goes to the same group. There is no extra info here.

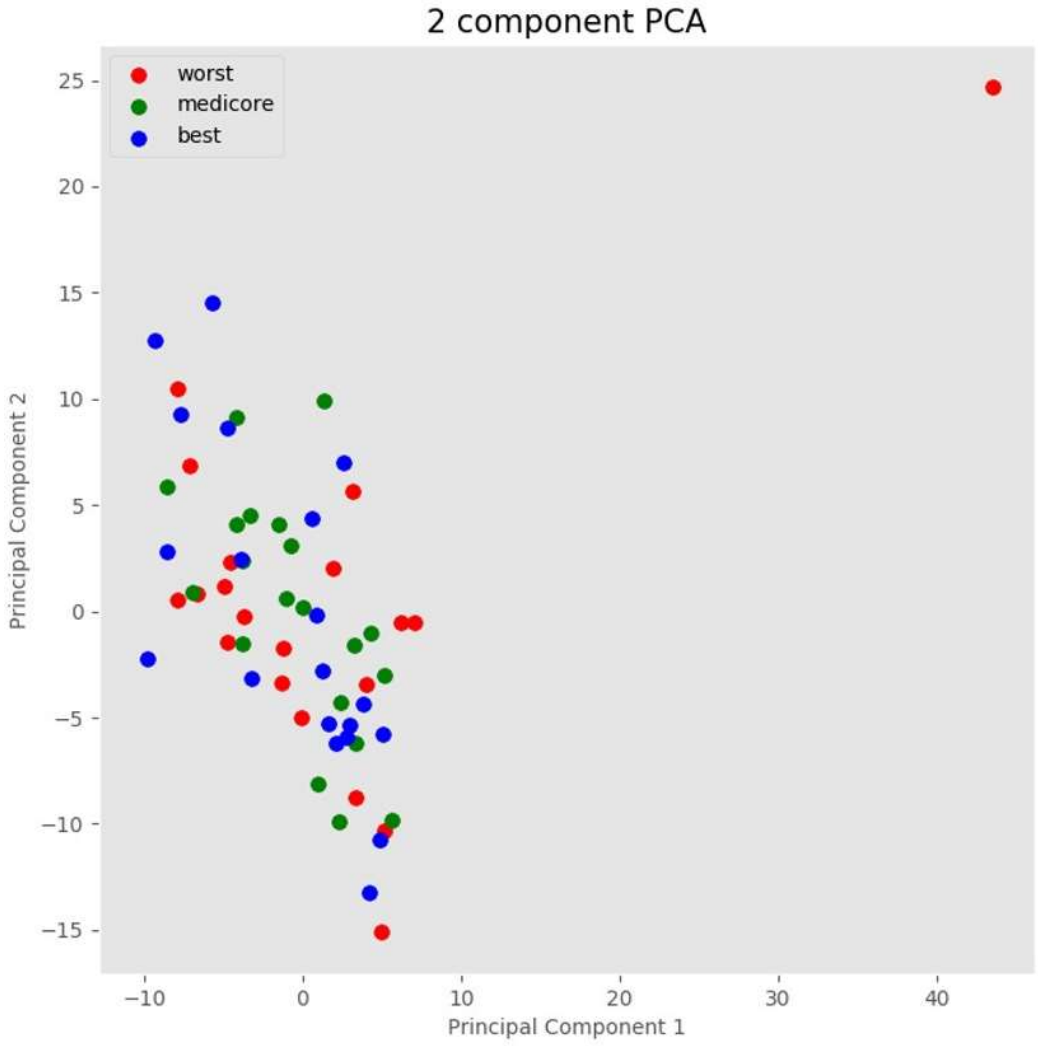


Figure 13 PCA, interpolation applied to columns

The results from PCA do not give any new information. The data is so similar and there are missing values with every row, so filling methods are applied to unite the data. If row dropping was applied with the missing values to whole dataset, the dataset would be empty and has no use for the analysis.

4.6 Tree modeling

The decision tree model, which differs a great deal from earlier methods, is used for mining important features (variables, which affect the result most) from data. Tree modeling with several different hyper parameter options is used for more reliable results along with random forest algorithm. This method works well for non-linear data also. (Nevavuori 2018a.)

4.6.1 Tree model optimization

Decision trees as standalone without tweaking can be very sensitive for small changes in data, so result can differ a lot because of that. Tree models can also become very complex if a very deep tree is constructed from data. Random forest algorithm is a powerful tool for creating multiple trees from selected dataset. (Nevavuori 2018a.)

4.6.2 Random forest algorithm

Random forest starts creating new trees from randomizing original data. Algorithm shuffles/stacks data for every "new" generated tree. If another example is created from Table 2, the shuffled dataset can look like this (Table 7):

Table 7 Randomized data table for random forest algorithm

Barks	Weight	Tail length	Dog
Yes	15 kg	10 cm	Yes
Yes	25 kg	3 cm	Yes
No	10 kg	20 cm	Other animal
Yes	15 kg	10 cm	Yes

What differs Table 7 from Table 2 now is the last row that has been replaced with the first row. The new table is generated by randomly picking rows from the original data table until the row count is similar to the original data table. This operation is repeated so many times that the forest tree count has been filled in. Then new models

are created as explained in an earlier chapter (Chapter 3.14), and their results are used to vote for the most effect variable(s) and prediction result. (Starmer 2018.)

When using numerical data, trees can also become very deep and the decision might not have a very important effect anymore. It is possible to limit tree depth directly, so that the result is obtained earlier. Tree depth/complexity can also be reduced by giving every node a minimum amount of data before a split is possible (For example, at least 50 data samples are needed to create a new node split). (Nevavuori 2018a.)

4.6.3 Tree model analysis

This case has been analysed with following hyper parameter combinations:

- Max depth sizes = 2, 5, 10, 15, 20, 30, 50
- Minimum sample splits = 2, 5, 10, 15, 20, 30, 50
- Forest tree counts = 5, 10, 25, 50, 100, 200, 300, 500

The data is filtered based on variables affecting at least 5 percentage of the result and/or manual inspection from generated figures. Figures (Showing how hyper parameter tuning affect result) are interpreted following way:

X-axis show the maximum depth of a tree (/or how many samples are required to make a new node split) and y-axis how many independent variables there are in total with the current hyper parameter combination to predict the dependent variable (the same variable can occur multiple times).

Then forest tree count, maximum depth and minimum sample count for split are selected, when the variable count stays at roughly the same level at the y-axis. In the analysis part, more loose values are also tested.

Then the variables are extracted from forests with all previously explained hyper parameter combinations; thus, randomly showing variables are reduced to a minimum amount and the most often detected variables are selected.

Example from independent variable count stabilization using maximum depth size/minimum sample count and forest tree count for one dependent variable (See Figure 14 and Figure 15):

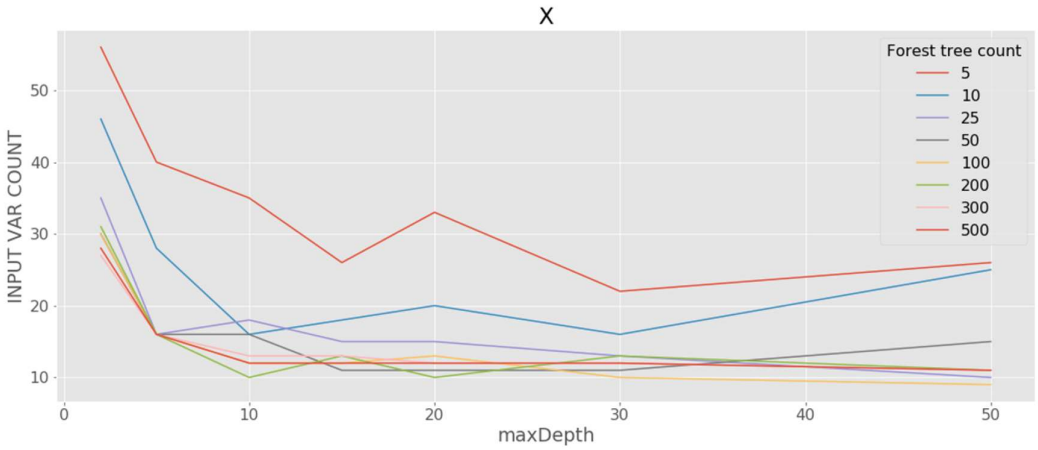


Figure 14 Example of decision tree maximum depth effect

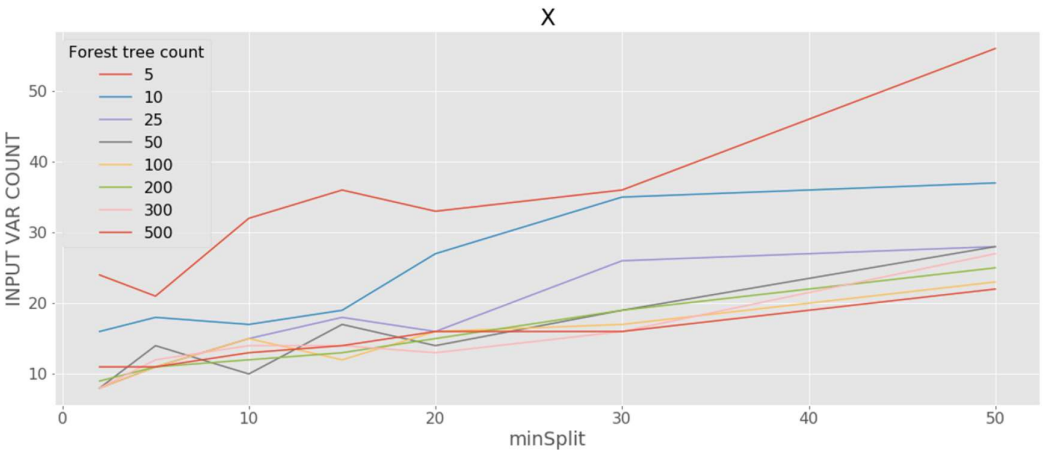


Figure 15 Example of decision tree minimum node split effect

Based on these figures, following hyper parameters are considered optimal for selected variable:

- Forest size of 100 trees (Higher selected count does not affect the analysis negatively)
- Tree node depth of 10
- Minimum sample count for the split of 10

4.7 Significant variables (based on tree models)

Using tree models and hyper parameter tuning, few variables have been found to have the most effect on dependent variables; these are not considered groups but instead variables, which appear most of the times in each of modeling phases. For data, the following tables 8 and 9 show the difference between independent variable (Which has most impact) count for the different datasets (Table 8 and Table 9):

Table 8 Result table for data1

data-y variable one	data-y variable two
data-x var1	data-x var9
data-x var2	data-x var10
data-x var3	data-x var11
data-x var4	data-x var8
data-x var5	data-x var12
data-x var6	data-x var4
data-x var7	data-x var1
data-x var8	data-x var13

Table 9 Result table for data2

data-y variable one	data-y variable two
data-x var14	data-x var18
data-x var15	data-x var19
data-x var16	data-x var20
data-x var17	data-x var21

When checking these tables, it is possible to see that *data2* does not have as many variables showing up as in the other data. This can mean that other data has more

information in it, however, when counting the data amount done to the analysis, it is not possible to give any new recommendations.

4.8 Feature importance

Feature importance is a way to check how much each variable affects the predicted result in decision trees. For example; the following three variables and predicted variable are shown as follows:

$$[variable1, variable2, variable3] \; and \; [predicted \; variable]$$

The result could be something like this:

- Variable1, 50%
- Variable2, 25%
- Variable3, 25%

The example result tells that variable1 value affect predicted variable result by 50% and other two variables value 25% each.

4.8.1 Importance for all important variables against dependent variables

All affecting variables (with optimal random forest hyper parameters) against dependent variables have been extracted from tree models. How much each variable increases dependent variables based on models is shown in bar charts as follows (Figure 16 & Figure 17 & Figure 18 & Figure 19). In the bar charts, independent variables are at x-axis (Without naming), the percentage effect is shown in y-axis.

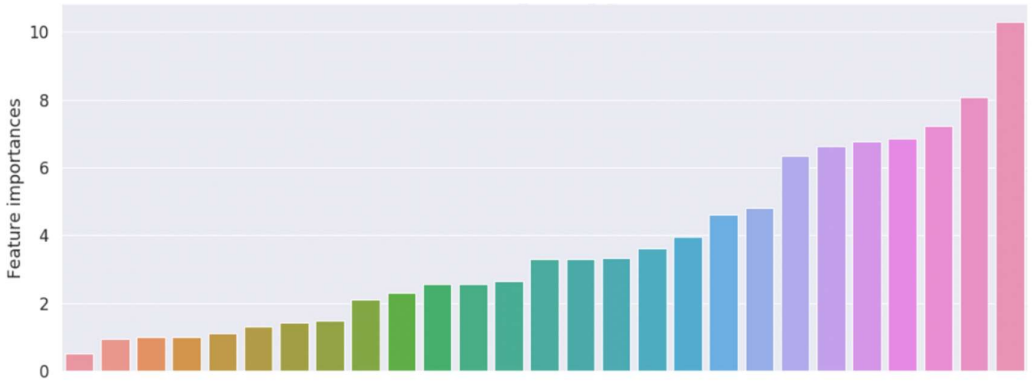


Figure 16 Feature importance for data1 against data-y variable one

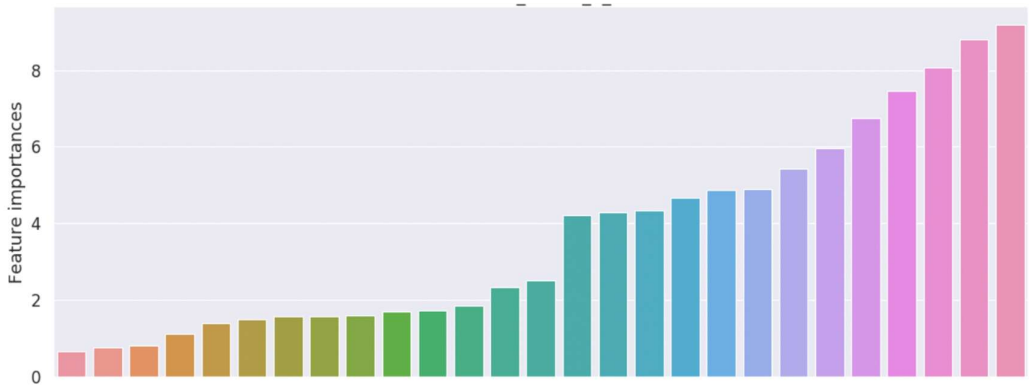


Figure 17 Feature importance for data1 against data-y variable two

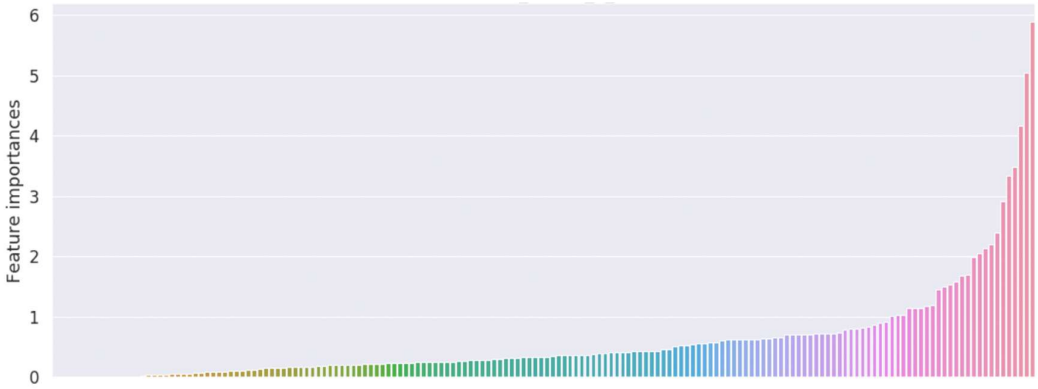


Figure 18 Feature importance for data2 against data-y variable one

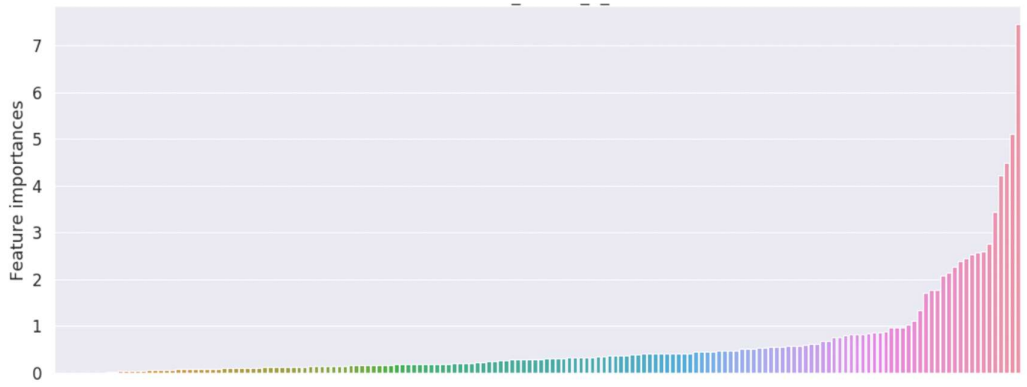


Figure 19 Feature importance for data2 against data-y variable two

4.8.2 Independent and dependent variable ratio

When independent variable, dependent variable and their ratio are plotted in the same figure with specific order, it is easy to detect how independent variable behaves when dependent variable increases (Nevavuori 2018c.). In this case, ordering was dependent variable value from lowest to highest.

Data-y ratio against *data-x* is checked for gears in the following way:

$$normalized\ data_y - normalized\ data_x$$

With that equation, the ratio value can be extracted, showing which one is higher, *data-y* or *data-x* or if they are equal. When checking the ratio value against independent variable and dependent variable differ from mean, it is possible to detect information about their ratio and ratio tells behavior between independent and dependent variables. (Nevavuori 2018c.)

Figure 20 shows information about values and contains three different plots. The first plot is how much dependent variable value differs from the mean of dependent variable value. The second plot is how much independent variable value differs from the mean of independent variable value. And last plot shows their normalized ratio, the formula of which is mentioned earlier in this chapter. All plots are in the same order, which is order by dependent variable value (Ascending).

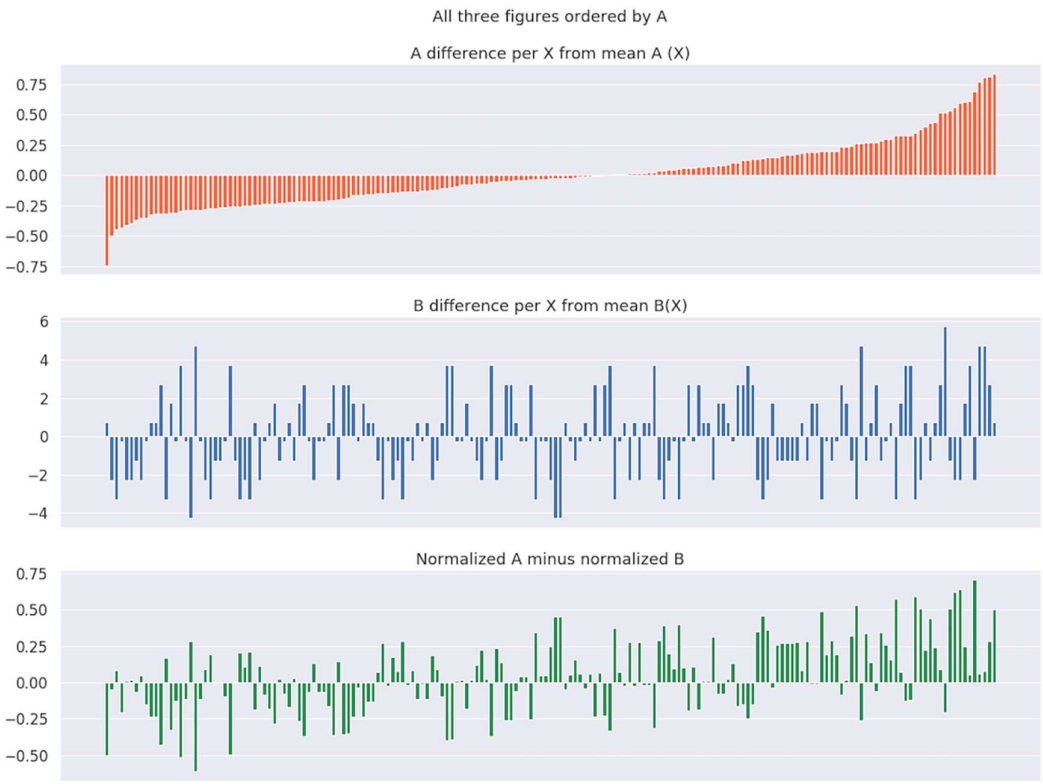


Figure 20 Independent and dependent variable ratio plots

4.8.3 Results based on chapter 4.8.2 style figure inspection

When the comparison was conducted with independent variable values and dependent variable values from figures manually, there were some notes that could be done. These notes from the tree model provided variables such as in the following examples:

- With a small amount of data at mean *data-y, data-x* follows *data-y*
- With *data-x* above mean, much of the gear *data-y* is lower and with *data-x* below the mean, the *data-y* is higher

However, mostly there was not enough proof in the figures to see which variable alone impacted another (and mostly the figures just had random noise).

4.8.4 Data distribution

Data distribution for each variable (Dependent and independent (Which had most effect to selected dependent variables)) is generated to see which point *data-x*/*data-y* is distributed mostly in current data. If tolerance limits are known, then it would be easy to check the points where the data is located in known limits.

In the bar chart, *data-x* spread is at x-axis and y-axis indicates how many data points are in the selected point in x-axis. Following (Figure 21) shows that most of the *data-x* is distributed between in a range from 0.6 to 1.0.

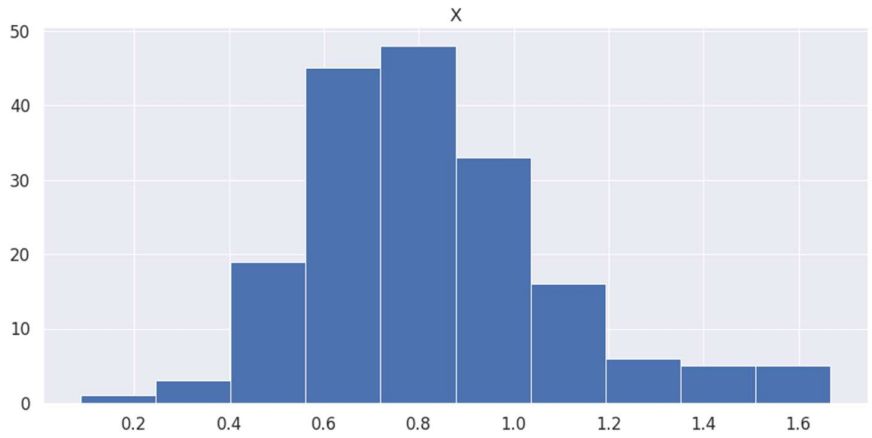


Figure 21 Example from data distribution

5 Results of data analysis

The results of the linear regression analysis do not provide any new information on which variable could cause the other variable values to change directly. Variable combinations (1 or more) do only seem to have random noise, when using linear methods. Visualization with PCA does not help to provide new information either in grouped or ranked variables. Therefore, as a conclusion from these linear methods no new information was gained from the data.

The tree modeling provided *data-x* variables, which affected mostly the selected *data-y* values. From the tree model results, a further analysis was carried out for those *data-x* variables against *data-y* variables. The distribution for *data-x* and *data-y* was also generated for additional information.

Based on the models and this analysis some notifications can be done; however, these notifications not significant to gain any new correlation information about single variables. Hence, as conclusion it can be stated that single variables do not cause the investigated *data-y* effects.

For further research for investigating how multiple *data-x* (non-linear data) combinations affect *data-y* there are some methods. As one method, it is possible to plot *data-x* as line plots into the same figure and manually check if *data-x* have some kind of pattern. A more advanced way could be to investigate the method to generate new linear components from non-linear *data-x*, and then conduct a linear regression analysis for the new linear components and generate coefficients to see how they might affect dependent variable. (Nevavuori 2018b.)

6 Summary

Domain knowledge was provided by Moventas Gears, which limited the variables for the data mining process; since otherwise it could have taken too much time. With domain knowledge, specific variables were under the scope in process.

Data analytic was done for data, which was already in CSV file format instead of raw data in separate files. That made possible to focus on the data mining process for the data and time was not wasted on parsing data from files/sources.

In this project a total of five different main methods were researched, namely the regression analysis, decision trees (Random forest regressor), PCA dimension reduction, Mean Shift clustering and Manhattan Distance finding algorithm. The used supporting methods, which were also used in some of the main methods were the following: data distribution, data interpolation, normalization, manual plot checking and outlier removal.

The main purpose of this project was to detect how selected *data-x* (and also, *data-z*, *data-i*, *data-j* and *data-k*) variables affect the selected *data* variables. One dataset (*data1*) was used in the regression analysis, PCA dimension reduction, Mean Shift clustering and Manhattan distance finding algorithm with supportive methods. Two

datasets (*data1* and *data2*) were used for researching the results from tree modeling (Random forest regressor).

Data analysis to this data provided knowledge that the selected single/multiple variable(s) do not cause any straight linear effect to data variables, and single variables also do not have any non-linear effects. Multiple variable non-linear affect could not be researched within the time limits of this project.

References

Asaithambi, S. 2017. Why, How and When to Scale your Features. Medium website article. Accessed on 03.02.2019. Retrieved from https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e

Bronshtein, A. 2017. Simple and Multiple Linear Regression in Python. Towards Data Science website article. Accessed on 25.12.2018. Retrieved from https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9

Data Mining Processes. N.d. Article at Zentut website. Accessed on 03.02.2019. Retrieved from http://www.zentut.com/data-mining/data-mining-processes/

Detect and exclude outliers in pandas dataframe. N.d. Stackoverflow forum post. Accessed on 25.12.2018. Retrieved from https://stackoverflow.com/questions/23199796/detect-and-exclude-outliers-in-pandas-dataframe

F Statistic / F Value: Simple Definition and Interpretation. 2018. Statistics how to website article. Accessed on 27.01.2019. Retrieved from http://www.statisticshowto.com/probability-and-statistics/f-statistic-value-test/

Finding eigenvalues and eigenvectors. N.d. Article at scss.tcs.ie website. Accessed on 01.01.2019. Retrieved from https://www.scss.tcd.ie/~dahyotr/CS1BA1/SolutionEigen.pdf

Foltz, B. 2013. Statistics 101: Simple Linear Regression, The Least Squares Method. Video at YouTube website. Accessed on 02.01.2019. Retrieved from https://www.youtube.com/watch?v=Qa2APhWjQPc

Foltz, B. 2014. Statistics 101: Multiple Regression, The Very Basics. Youtube website. Accessed on 02.01.2018. Retrieved from https://www.youtube.com/watch?v=dQNpSa-bq4M

Galarnyk, M. 2017. PCA using Python (scikit-learn). Towards Data Science website article. Accessed on 25.12.2018. Retrieved from https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

Galetto, M. 2016. What Is Data Analysis?. Article at NG Data website. Accessed on 30.12.2018. Retrieved from https://www.ngdata.com/what-is-data-analysis/

Gallo, A. 2015. Refresher on Regression Analysis. Article at hbr.org website. Accessed on 14.09.2018. Retrieved from https://hbr.org/2015/11/a-refresher-on-regression-analysis

Granville, V. 2014. Developing Analytic Talent : Becoming a Data Scientist. Indianapolis: John Wiley & Sons, Inc.

Imanuel. N.d. What is data analysis ?. Article at Predictive analytics today website. Accessed on 30.12.2018. Retrieved from https://www.predictiveanalyticstoday.com/data-analysis/

Janakiev, N. 2018. Understanding the Covariance Matrix. Article at datasciense+ website. Accessed on 01.01.2019. Retrieved from https://datascienceplus.com/understanding-the-covariance-matrix/

Kase, E & Kuang, Y. N.d. How to use gaussian elimination to solve systems of equations. Article at dummies.com website. Accessed on 01.01.2019. Retrieved from https://www.dummies.com/education/math/calculus/how-to-use-gaussian-elimination-to-solve-systems-of-equations/

Linear interpolation with Excel. N.d. Article at Data digitization website. Accessed on 31.12.2018. Retrieved from https://www.datadigitization.com/dagra-in-action/linear-interpolation-with-excel/

Meanshift Algorithm for the Rest of Us (Python). 2016. Chioka website article. Accessed on 31.12.2018. Retrieved from http://www.chioka.in/meanshift-algorithm-for-the-rest-of-us-python/

Nevavuori, P. 2018a. Discussion about decision trees. Researcher at Jyväskylä University of Applied Scienses. Jyväskylä. Discussion 15.10.2018

Nevavuori, P. 2018b. Discussion about further actions. Researcher at Jyväskylä University of Applied Scienses. Jyväskylä. Discussion 23.11.2018

Nevavuori, P. 2018c. Discussion about visualization. Researcher at Jyväskylä University of Applied Scienses. Jyväskylä. Discussion 15.11.2018

Polamuri, S. 2015. Five most popular similarity data-x implementation in python. Data aspirant website article. Accessed on 25.12.2018. Retrieved from http://dataaspirant.com/2015/04/11/five-most-popular-similarity-data-x-implementation-in-python/

Raschka, S. 2014. About Feature Scaling and Normalization. Sebastian Raschka´s website article. Accessed on 25.12.2018. Retrieved from http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-standardization

Rishab, J. 2017. Decision Tree. It begins here. Article at Medium website. Accessed on 29.10.2018. Retrieved from https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134

Rouse, M. 2015. Extrapolation and interpolation. Article at Whatis Techtarget website. Accessed on 30.12.2018. Retrieved from https://whatis.techtarget.com/definition/extrapolation-and-interpolation

Sayad, S. N.da. Decision Tree – Regression. Author website. Accessed on 23.10.2018. Retrieved from https://www.saedsayad.com/decision_tree_reg.htm

Sayad, S. N.db. Decision Tree – Classification. Author website. Accessed on 23.10.2018. Retrieved from https://www.saedsayad.com/decision_tree.htm

Sklearn MeanShift documentation. N.d. Sklearn documentation. Accessed on 25.12.2018. Retrieved from https://scikit-learn.org/stable/modules/clustering.html#mean-shift

Smith, L. 2002. A tutorial on Principal Components Analysis. Article at Otago website. Accessed on 01.01.2019. Retrieved from http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Sridhar, J. 2018. What Is Data Analysis and Why Is It Important?. Article at makeuseof website. Accessed on 31.12.2018. Retrieved from https://www.makeuseof.com/tag/what-is-data-analysis/

Standard Deviation and Variance. N.d. Math is fun website article. Accessed on 11.10.2018. Retrieved from https://www.mathsisfun.com/data/standard-deviation.html

Starmer, J. 2018. StatQuest: Random Forests Part 1 - Building, Using and Evaluating. Youtube website. Accessed on 23.10.2018. Retrieved from https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

Appendices

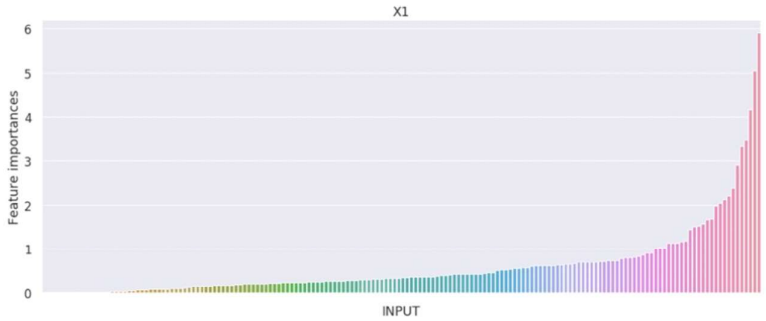
Appendix 1. Jupyter Notebook from one data analysis phase

In [2]:

```
import pandas as pd
import numpy as np
import ast
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
from matplotlib import style
style.use("seaborn")
sns.set(font_scale=1.5)


from scipy import stats
from collections import OrderedDict
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor


csvFile = 'data2.csv'
columns = 'columns.txt'


result1 = open(columns, 'r')
resultList = result1.read()
data2items = ast.literal_eval(resultList)
data2items = data2items + ["data-y_var1"]+ ["data-y_var2" ]


df = pd.read_csv(csvFile, header=0)
df = df[data2items]
df = df.dropna(thresh=len(df) - 30, axis=1)
df = df.dropna(axis=0).reset_index(drop=True)
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]


CSV_PATH = 'tree.csv'

DF_TREES = pd.read_csv(CSV_PATH, header=0)
DF_TREES = DF_TREES.drop(DF_TREES.columns[0], axis=1)
```

In [3]:

```
DF_TREES_VAR1 = DF_TREES.loc[(DF_TREES['FORESTCOUNT'] == 100) & (DF_TREES['maxDe
pth'] == 10) & (DF_TREES['minSplit'] == 20) & (DF_TREES['OUTPUT'] == 'data-y_var
1')]
DF_TREES_VAR1 = DF_TREES_VAR1.sort_values(by=['EFF'])
DF_TREES_VAR1['EFF'] = DF_TREES_VAR1['EFF'].map(lambda x: x* 100)

plt.figure(figsize=(18,7))
ax = sns.barplot(DF_TREES_VAR1['INPUT'], DF_TREES_VAR1['EFF'])
ax.set(xticklabels="")
ax.set(ylabel="Feature importances")
plt.title('X1')
plt.savefig("images/X1")
plt.show()
```

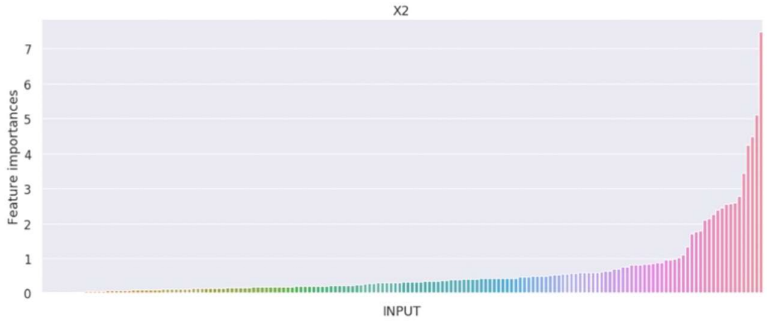


In [4]:

```
pd.options.display.max_rows = 1000
print_df = DF_TREES_VAR1.reset_index()
print_df.drop(columns=["FORESTCOUNT", "index"], inplace=True)
print_df = print_df.iloc[:,0:2]
```

In [5]:

```
DF_TREES_VAR2 = DF_TREES.loc[(DF_TREES['FORESTCOUNT'] == 100) & (DF_TREES['maxDe
pth'] == 10) & (DF_TREES['minSplit'] == 20) & (DF_TREES['OUTPUT'] == 'data-y_var
2')]
DF_TREES_VAR2 = DF_TREES_VAR2.sort_values(by=['EFF'])
DF_TREES_VAR2['EFF'] = DF_TREES_VAR2['EFF'].map(lambda x: x* 100)

plt.figure(figsize=(18,7))
ax = sns.barplot(DF_TREES_VAR2['INPUT'], DF_TREES_VAR2['EFF'])
ax.set(xticklabels="")
ax.set(ylabel="Feature importances")
plt.title('X2')
plt.savefig("images/X")
plt.show()
```



In [6]:

```
print_df = DF_TREES_VAR2.reset_index()
print_df.drop(columns=["FORESTCOUNT", "index"], inplace=True)
print_df = print_df.iloc[:,0:2]
```

Variables for analysis

data-y_var1

- data-x_var14
- data-x_var15
- data-x_var16
- data-x_var17

data-y_var2

- data-x_var18
- data-x_var19
- data-x_var20
- data-x_var21

In [7]:

```
data-y_var1_interesting_Vars = ["data-x_var14", "data-x_var15", "data-x_var16",
"data-x_var17"]

data-y_VAR1 = "data-y_var1"

data-y_var2_interesting_Vars = ["data-x_var18", "data-x_var19", "data-x_var20",
"data-x_var21"]

data-y_VAR2 = "data-y_var2"
```

In [8]:

```
# Correlation table for variables

corr_table = list(OrderedDict.fromkeys(data-y_var2_interesting_Vars+data-y_var1_
interesting_Vars+ [data-y_VAR1]+ [data-y_VAR2]))


corr_df = df[corr_table]
corr_df = corr_df.dropna(axis=0)
corr_df.corr().iloc[:-2:,-2:]
df = corr_df
```

In [9]:

```
# Normalization for values

df_values = corr_df.values
min_max = MinMaxScaler()
df_values_scaled = min_max.fit_transform(df_values)
df_norm = pd.DataFrame(df_values_scaled, columns=corr_df.columns)
```

In [10]:

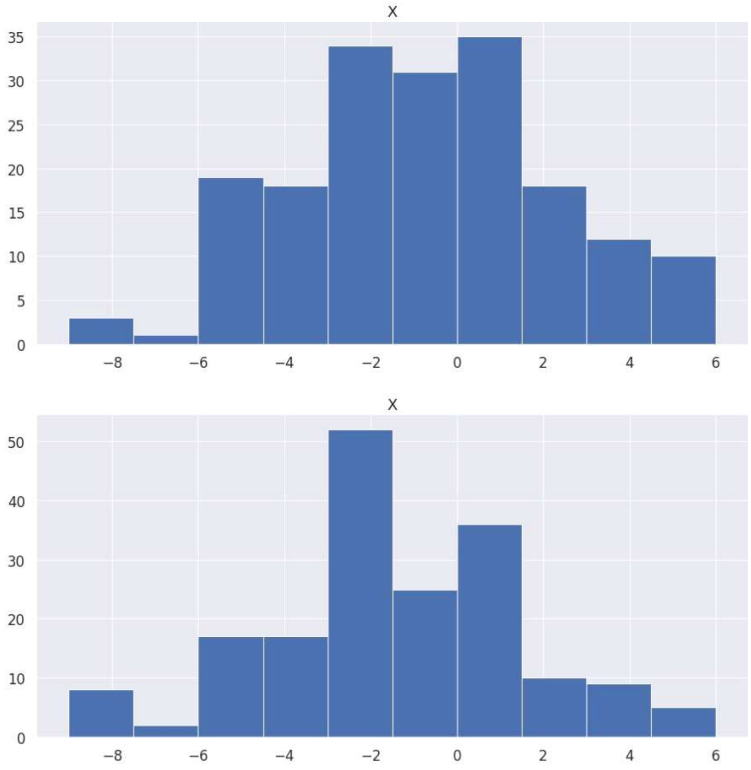
```
#Drop old index
df.reset_index(inplace=True, drop=True)
df_norm.reset_index(inplace=True, drop=True)
```

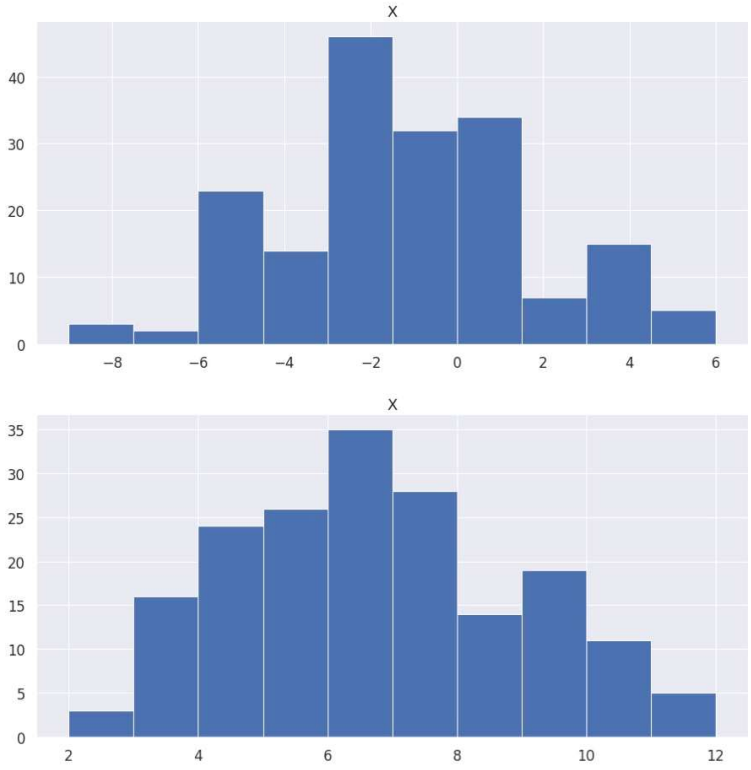
In [11]:

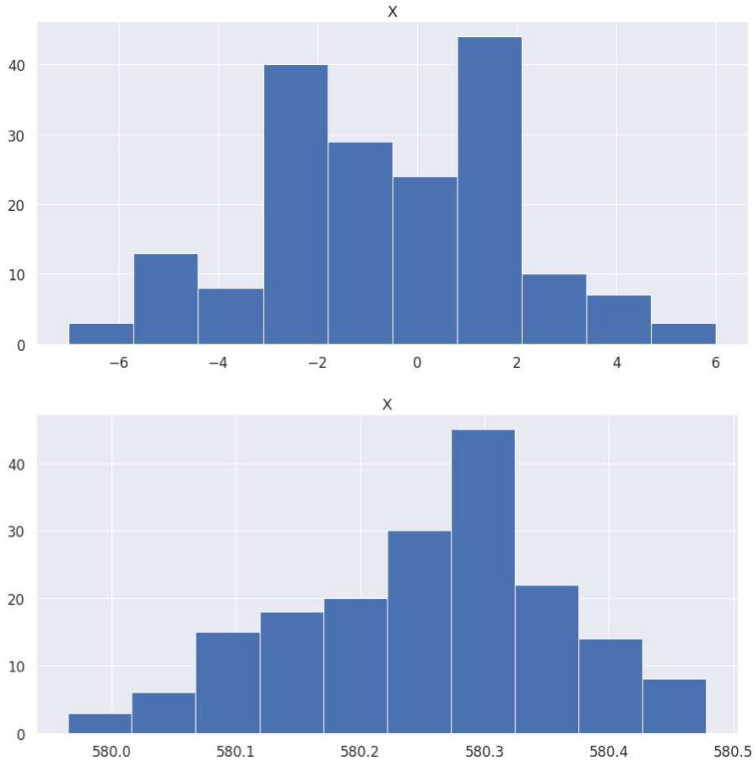
```
def parsefigure(name):
    name = name.replace("/", "-").replace("*","-").replace(" ", "_").replace("."
,"")
    name = name.replace("/", "-").replace("*","-").replace(" ", "_").replace("."
,"")

    return name
```

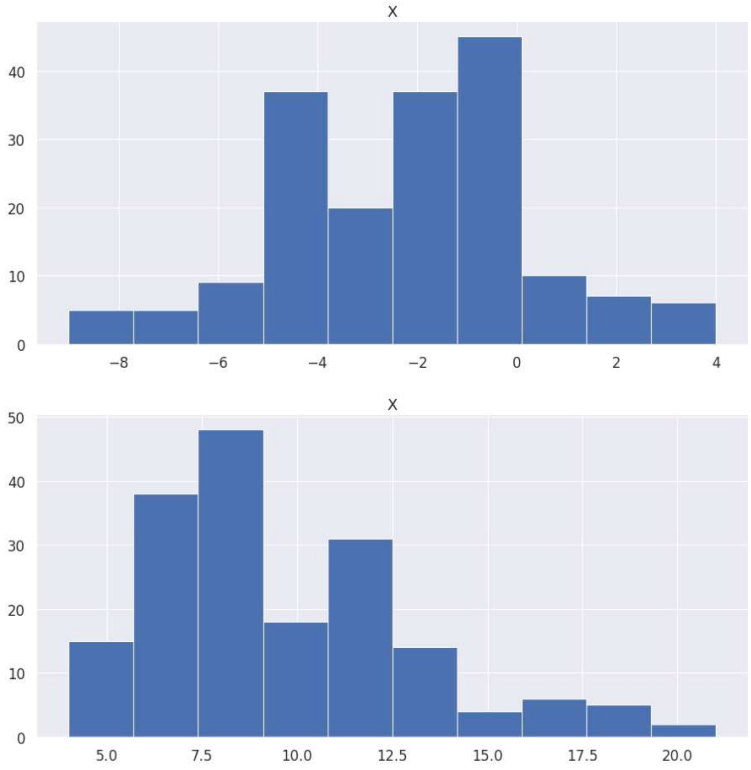
In [12]:

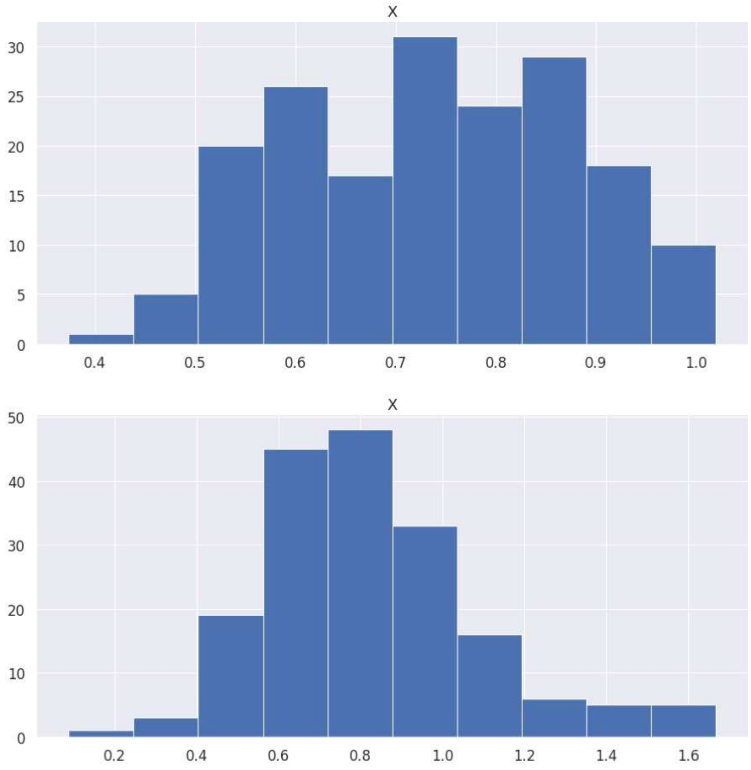
```
# Data distribution plots

for column in list(df_norm):
    plt.figure(figsize=(15,7))
    plt.hist(df[column])
    savename = parsefigure(column)
    plt.title("X")
    plt.savefig("images/distribution-X3".format(savename))
```


X
35
30
25
20
15
10
5
0
−8
−6
−4
−2
0
2
4
6
X
50
40
30
20
10
0
−8
−6
−4
−2
0
2
4
6


X
40
30
20
10
0
−8
−6
−4
−2
0
2
4
6
X
35
30
25
20
15
10
5
0
2
4
6
8
10
12

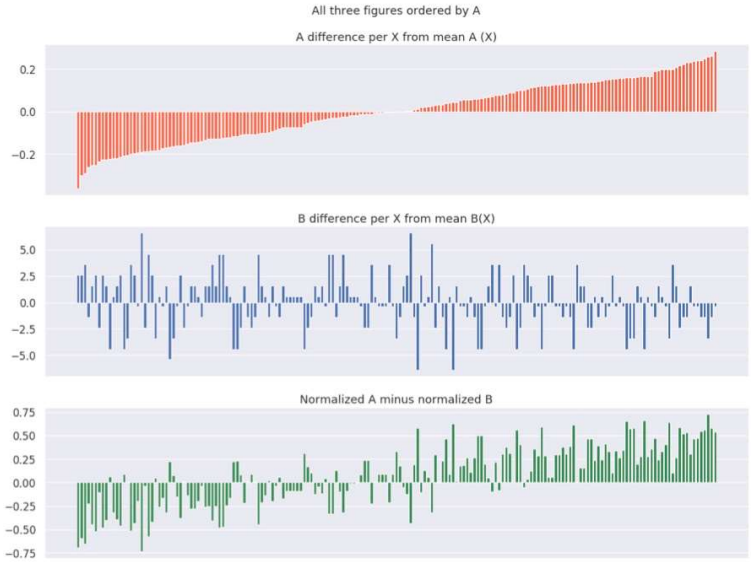

X
40
30
20
10
0
−6
−4
−2
0
2
4
6
X
40
30
20
10
0
580.0
580.1
580.2
580.3
580.4
580.5

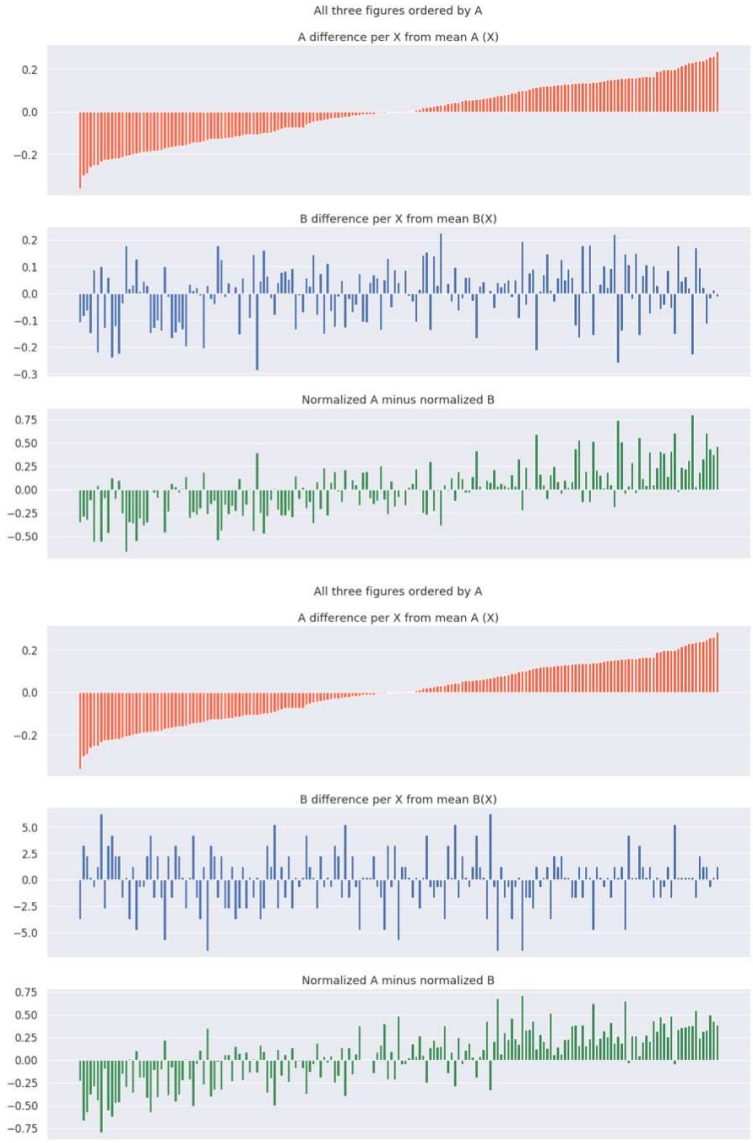

X
40
30
20
10
0
−8
−6
−4
−2
0
2
4
X
50
40
30
20
10
0
5.0
7.5
10.0
12.5
15.0
17.5
20.0

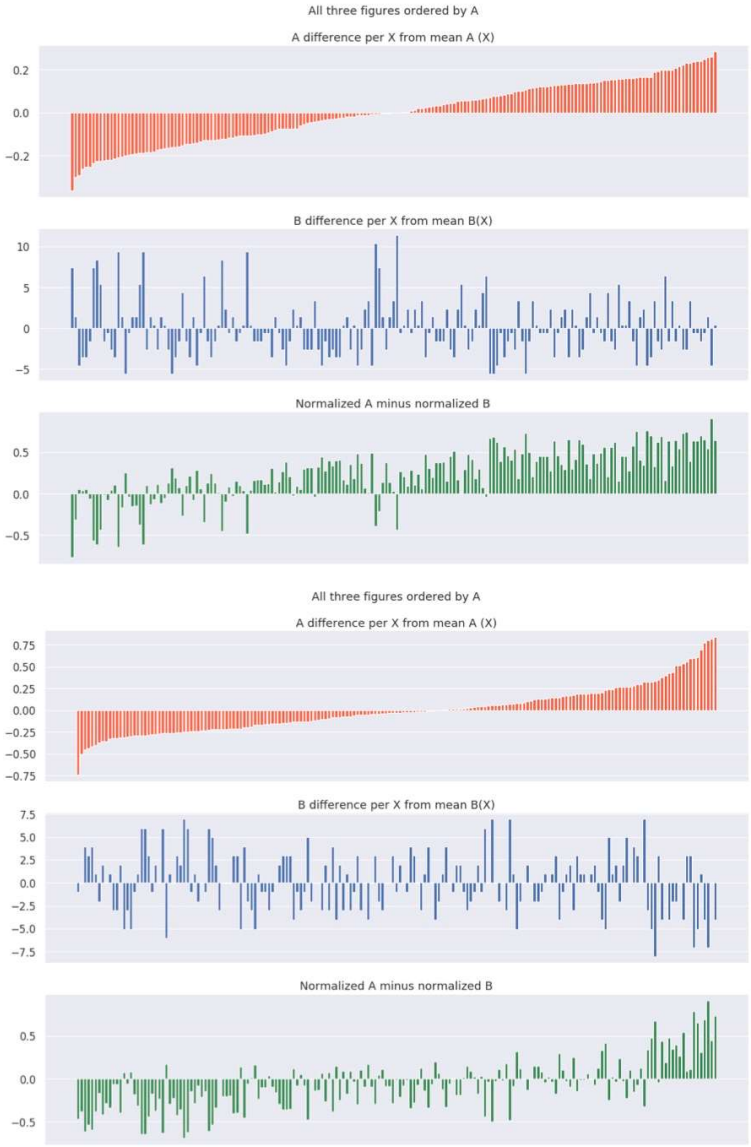

X
30
25
20
15
10
5
0
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X
50
40
30
20
10
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

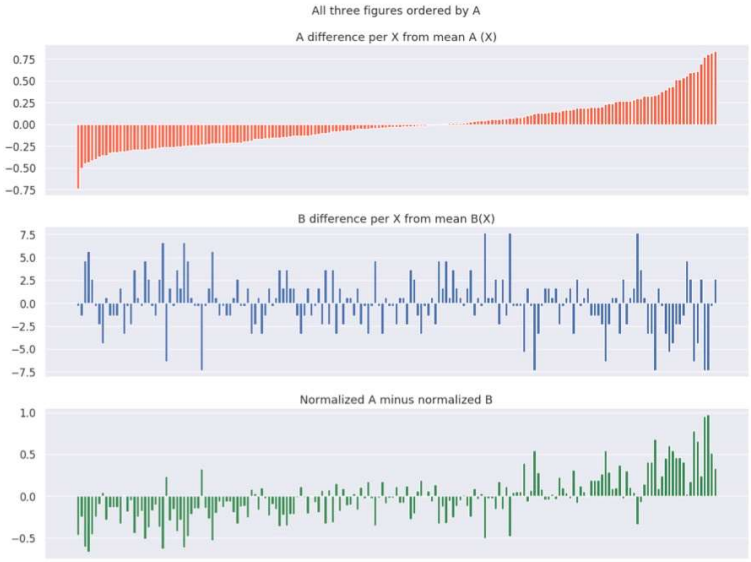
In [13]:

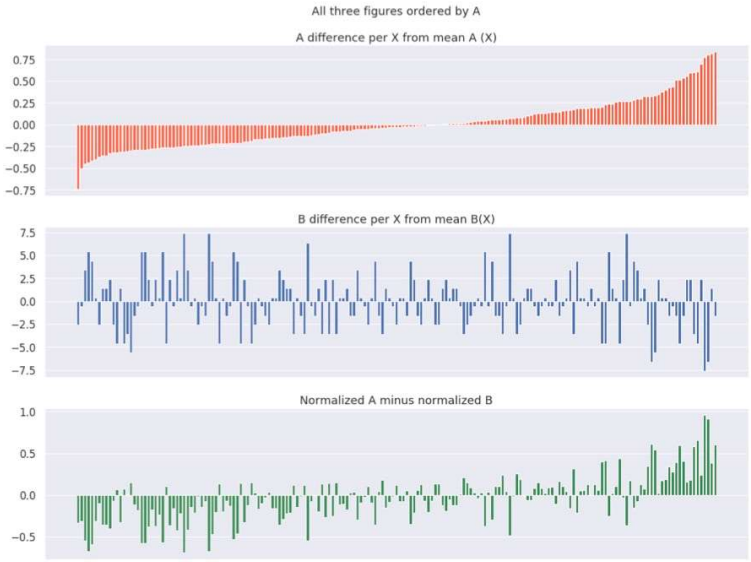
```
# Variable ratio plots

itemRange = list(range(0, len(df_norm)))
data-ys = [data-y_VAR1]+[data-y_VAR2]

for data-y in data-ys:

    def plot_data(column_data):
        for column in column_data:

            test = df_norm[data-y]-df_norm[column]
            df_norm[data-y+"+"+column+"_N"] = test
            df[data-y+"+"+column+"_N"] = test


            df_norm.sort_values(by=[data-y], ascending=True,inplace=True)
            df.sort_values(by=[data-y], ascending=True,inplace=True)


            plt.rcParams["figure.figsize"] = (20,15)
            f, axarr = plt.subplots(3, sharex=True)
            axarr[2].bar(itemRange,df_norm[data-y+"+"+column+"_N"], color="seagr
een")
            axarr[1].bar(itemRange,df[column]-np.mean(df[column]), label=column)
            axarr[0].bar(itemRange,df[data-y]-np.mean(df[data-y]), label=data-y,
 color='tomato')


            axarr[0].set_title("All three figures ordered by A\n\nA difference p
er X from mean A ({0})".format("X"))
            axarr[1].set_title("B difference per X from mean B({0})".format("X"
))
            axarr[2].set_title("Normalized A minus normalized B")

            axarr[0].axes.get_xaxis().set_ticks([])
            axarr[1].axes.get_xaxis().set_ticks([])
            axarr[2].axes.get_xaxis().set_ticks([])


            savename = parsefigure(column+"-and-"+data-y)
            f.savefig("images/{0}".format("threePlots"))
            plt.show()

    if data-y == 'data-y_var1':
        plot_data(data-y_var1_interesting_Vars)
    else:
        plot_data(data-y_var2_interesting_Vars)
```

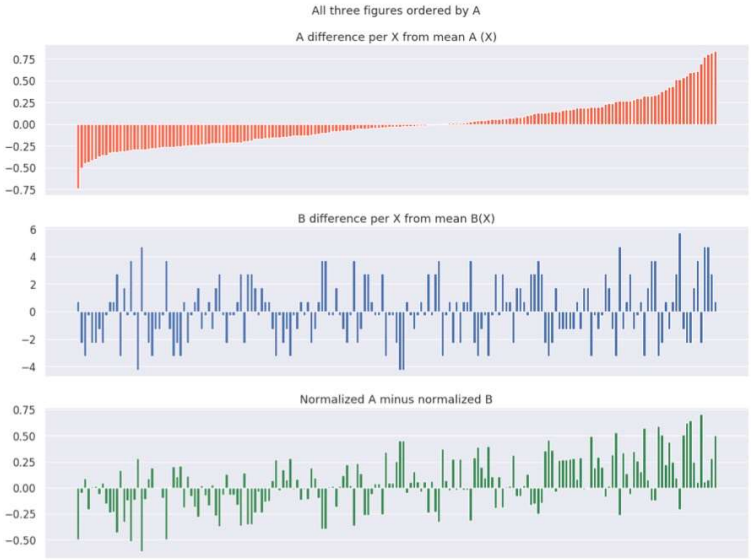

All three figures ordered by A
A difference per X from mean A (X)
0.2
0.0
−0.2
B difference per X from mean B(X)
5.0
2.5
0.0
−2.5
−5.0
Normalized A minus normalized B
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


All three figures ordered by A
A difference per X from mean A (X)
0.2
0.0
−0.2
B difference per X from mean B(X)
0.2
0.1
0.0
−0.1
−0.2
−0.3
Normalized A minus normalized B
0.75
0.50
0.25
0.00
−0.25
−0.50
All three figures ordered by A
A difference per X from mean A (X)
0.2
0.0
−0.2
B difference per X from mean B(X)
5.0
2.5
0.0
−2.5
−5.0
Normalized A minus normalized B
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


All three figures ordered by A
A difference per X from mean A (X)
0.2
0.0
−0.2
B difference per X from mean B(X)
10
5
0
−5
Normalized A minus normalized B
0.5
0.0
−0.5
All three figures ordered by A
A difference per X from mean A (X)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
B difference per X from mean B(X)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Normalized A minus normalized B
0.5
0.0
−0.5


All three figures ordered by A
A difference per X from mean A (X)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
B difference per X from mean B(X)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Normalized A minus normalized B
1.0
0.5
0.0
−0.5


All three figures ordered by A
A difference per X from mean A (X)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
B difference per X from mean B(X)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Normalized A minus normalized B
1.0
0.5
0.0
−0.5


All three figures ordered by A
A difference per X from mean A (X)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
B difference per X from mean B(X)
6
4
2
0
−2
−4
Normalized A minus normalized B
0.75
0.50
0.25
0.00
−0.25
−0.50

Appendix 2. Decision tree model generation source code

```
import pandas as pd
import numpy as np
import ast
from scipy import stats
from sklearn.ensemble import RandomForestRegressor

csvFile = 'data2.csv'
columns = 'columns.txt'
result1 = open(columns, 'r')
resultList = result1.read()
data2items = ast.literal_eval(resultList)
data2items = data2items + ["data-y_var1"]+ ["data-y_var2" ]
df = pd.read_csv(csvFile, header=0)
df = df[data2items]
df = df.dropna(thresh=len(df) - 30, axis=1)
df = df.dropna(axis=0).reset_index(drop=True)
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]

analysisColumns = list(df)[:-2]
analysisVars = list(df)[-2:]
resultDf = pd.DataFrame()

maxDepths = [2, 5, 10, 15, 20, 30, 50]
minSampleSplit=[2, 5, 10, 15, 20, 30, 50]
forestCounts = [5, 10, 25, 50, 100, 200, 300, 500]

for outVar in analysisVars:
   for depth in maxDepths:
      for split in minSampleSplit:
         for forest in forestCounts:

            regressor = RandomForestRegressor(n_estimators=forest,
max_depth=depth,min_samples_split=split)
            regressor.fit(df[analysisColumns], df[outVar])

            featureImportances = regressor.feature_importances_

            topValues = list(featureImportances)
            featureIndex = list(df[analysisColumns])

            for item in range(0, len(featureIndex)):
               tempDf = pd.DataFrame({"INPUT":featureIndex[item], "OUTPUT":outVar,
"EFF":topValues[item], "maxDepth":depth, "minSplit": split, "METHOD":"TREE",
"ROWCOUNT":len(df[analysisColumns]), "FORESTCOUNT":forest}, index=[0])

               resultDf = resultDf.append(tempDf)
resultDf.to_csv("tree.csv")
```