

Master's Thesis

---

# Augmented Data Discovery

Implementation of an Augmented Data Discovery solution

---

Arnout R. Berkenbosch

Company Supervisor: R. Bakkenes

Graduation Supervisor: Prof. Dr. A.E. Eiben

Second Reader: Dr. E.R. Dugundji

Vrije Universiteit Amsterdam

Avanade Netherlands BV

August 31<sup>st</sup>, 2020

---

# Augmented Data Discovery

Implementation of an Augmented Data Discovery solution

---

A.R. Berkenbosch  
2620585

Master Project Business Analytics

Vrije Universiteit Amsterdam  
Faculty of Science  
Business Analytics  
De Boelelaan 1081a  
1081 HV Amsterdam

Host organization:  
Avanade Netherlands B.V.  
Orteliuslaan 1000  
3528 BD Utrecht

August 2020

# Preface

Before you lies the thesis “Augmented Data Discovery”, which targets the automation of the data science process. It has been written to fulfill the graduation requirements of the Master Business Analytics at the Vrije Universiteit Amsterdam. I was engaged in researching and writing this thesis from March until August 2020.

The project was undertaken at Avanade BV in Utrecht, where I undertook an internship. My research question was formulated together with my supervisor, Robin Bakkenes. The research took place under difficult circumstances due to COVID-19 as I could not be physically present at the office and the university during the entire internship period. Fortunately, Avanade and the university showed excellent flexibility during this pandemic, which allowed me to properly conduct my research. Using communication platforms, such as Microsoft Teams and Zoom, I was able to get the support I needed to complete my research.

I would like to thank Avanade as a whole for providing me with the tools needed to conduct this research. I would also like to thank my company supervisor Robin Bakkenes in particular for his guidance.

I would like to thank my supervisor from the university Prof. Dr. A.E. Eiben for his guidance during the internship. I would also like to thank my fellow students for their support and advice in my study group, who also fall under the guidance of Prof. Dr. A.E. Eiben. Also, I would like to thank Dr. E.R. Dugundji for being the second reader.

I hope you enjoy your reading.

Arnout Berkenbosch

Utrecht, August 31<sup>st</sup>, 2020

# Executive Summary

**Problem Definition:** Data discovery is the process of collecting and analyzing data to gain insights into the different trends and patterns that are within the data. Augmented data discovery aims to automate this data discovery process. Much time is spent by data scientists in analyzing the data. By automating the iterative steps that a data scientist must perform, the time that is spent on the data science process can be shortened significantly. Augmented data discovery is expected to play a more important role in the future. Therefore, Avanade as a leading digital innovator would like to do research on this topic as this will most likely be an important part of their process in the nearby future.

**Approach:** To explore the possibilities of augmented data discovery solution, a prototype is created in Python. In the prototype, several data mining methods are implemented to create an automated workflow. Missing values are handled using a simple imputation method, imputing the median for numeric values and the mode for categorical values. Outliers are handled using the interquartile range (IQR) method. A framework called Featuretools is used for automatically creating new features. Two automated modeling frameworks are used, Tree-based Pipeline Optimization Tool (TPOT) and Auto-sklearn. Both frameworks handle several pre-processing steps, such as data transformations, decomposition and feature selection. Next to that, they both use an automated modeling approach where they pick a model and also tune its hyperparameters. This is done for either a classification or regression problem.

**Results:** The obtained results, by testing the prototype on a variety of different datasets, showed that this automated solution was capable of achieving some good results in online data competitions, such as Kaggle. The major advantage of such an augmented data science solution is efficiency. With minimal effort, such a solution is still able to obtain good results.

**Conclusion:** The prototype showed that this automated solution worked quite well. With some improvements and modifications, this prototype could evolve into a well working augmented data discovery tool. Such a tool could save valuable time within Avanade's data science process.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	About Avanade . . . . .	1
1.2	Business context and problem statement . . . . .	1
1.3	Research question . . . . .	3
1.4	Report Structure . . . . .	3
<b>2</b>	<b>Literature review</b>	<b>4</b>
2.1	Pre-processing . . . . .	4
2.2	Feature engineering . . . . .	5
2.3	Modeling . . . . .	5
<b>3</b>	<b>Augmented Data Science Tool</b>	<b>6</b>
3.1	Process description . . . . .	6
3.2	Prototype . . . . .	7
<b>4</b>	<b>Azure implementation</b>	<b>8</b>
4.1	Microsoft Azure . . . . .	8
4.2	Used Azure resources . . . . .	8
4.2.1	Azure Databricks . . . . .	8
4.2.2	Azure Data Lake Storage Gen2 . . . . .	9
4.2.3	Azure Data Factory . . . . .	9
4.3	Prototype . . . . .	9
<b>5</b>	<b>Data</b>	<b>10</b>
5.1	Conditions . . . . .	10
5.2	Dataset 1 - Titanic . . . . .	10
5.3	Dataset 2 - Mobile price classification . . . . .	11
5.4	Dataset 3 - Australia rain prediction . . . . .	11
5.5	dataset 4 - Random Acts of Pizza . . . . .	12
5.6	Dataset 5 - Pump it up . . . . .	12
5.7	Dataset 6 - Richter's Predictor: Modeling Earthquake Damage . . . . .	12
5.8	Dataset 7 - Insurance prediction . . . . .	13
5.9	Dataset 8 - House Prices: Advanced Regression Techniques . . . . .	13
5.10	Dataset 9 - Restaurant Revenue Prediction . . . . .	13

5.11	Dataset 10 - DengAI: Predicting Disease Spread . . . . .	14
5.12	Dataset 11 - Bike Sharing Demand . . . . .	14
5.13	Dataset 12 - Urban Air Pollution . . . . .	15
<b>6</b>	<b>Approach</b>	<b>16</b>
6.1	Pre-processing . . . . .	16
6.1.1	Recognizing Data types . . . . .	16
6.1.2	Data Cleaning . . . . .	17
6.1.3	Data Transformations . . . . .	18
6.1.4	Dimensionality reduction . . . . .	19
6.2	Feature Engineering . . . . .	19
6.3	Feature Selection . . . . .	19
6.4	Modeling . . . . .	20
<b>7</b>	<b>Missing Values</b>	<b>21</b>
7.1	Simple imputation . . . . .	21
7.2	k-Nearest Neighbors . . . . .	22
7.3	Prototype . . . . .	24
<b>8</b>	<b>Outliers</b>	<b>25</b>
8.1	Median Absolute Deviation (MAD) . . . . .	25
8.2	interquartile range (IQR) . . . . .	25
8.3	Z-score . . . . .	26
8.4	Prototype . . . . .	26
<b>9</b>	<b>Feature Engineering</b>	<b>27</b>
9.1	Deep Feature Synthesis . . . . .	27
9.1.1	Example . . . . .	28
9.1.2	Feature encoding . . . . .	28
9.2	Prototype . . . . .	29
<b>10</b>	<b>Modeling</b>	<b>30</b>
10.1	Tree-based Pipeline Optimization Tool (TPOT) . . . . .	30
10.1.1	Optimizing Tree-Based Pipelines . . . . .	32
10.2	Auto-sklearn . . . . .	32
10.2.1	Automated ensemble construction of models evaluated during optimization . . . . .	34
10.3	Comparison of TPOT against Auto-sklearn . . . . .	35
10.4	Prototype . . . . .	35

<b>11 Data Transformation</b>	<b>37</b>
11.1 Normalization . . . . .	37
11.1.1 Min-Max Normalization (Min-Max Scaling)) . . . . .	37
11.1.2 Z-score normalization (Standardization) . . . . .	37
11.1.3 Robust Normalization (Robust scaling) . . . . .	37
11.1.4 Prototype . . . . .	38
11.2 Transformation . . . . .	38
11.2.1 Linear transformation . . . . .	38
11.2.2 Box-cox transformation . . . . .	38
11.2.3 Polynomial feature transformation . . . . .	39
11.2.4 Prototype . . . . .	39
<b>12 Feature Selection</b>	<b>40</b>
12.1 Correlated features . . . . .	40
12.2 Low variance features . . . . .	41
12.3 Select k best features . . . . .	41
12.4 Select top n percentile features . . . . .	41
12.5 Recursive feature elimination . . . . .	41
12.6 Classification-based feature selection with Extremely Randomized Trees	42
12.7 Prototype . . . . .	42
<b>13 Dimensionality reduction</b>	<b>43</b>
13.1 Principal Component Analysis (PCA) . . . . .	43
13.2 Truncated SVD . . . . .	43
13.3 Kernel PCA . . . . .	43
13.4 Independent component analysis (ICA) . . . . .	44
13.5 Prototype . . . . .	44
<b>14 Classification and Regression Algorithms</b>	<b>45</b>
14.1 Naive Bayes . . . . .	46
14.2 Decision Tree . . . . .	46
14.3 Random forest and Extremely Randomized Trees . . . . .	47
14.4 Gradient Boosting . . . . .	47
14.5 Support Vector Machines (SVM) . . . . .	48
14.6 Logistic regression . . . . .	48
14.7 Generalized Linear Models (GLM) . . . . .	49
14.8 Discriminant Analysis . . . . .	50
14.9 K-Nearest Neighbors (KNN) . . . . .	50
14.10 Classical linear regressors . . . . .	50

<b>15 Experimental setup</b>	<b>52</b>
15.1 Training/validation/test split . . . . .	52
15.2 Modeling . . . . .	52
15.3 Evaluation metrics . . . . .	53
15.3.1 Confusion matrix . . . . .	54
15.3.2 Accuracy . . . . .	54
15.3.3 Precision . . . . .	55
15.3.4 Recall . . . . .	55
15.3.5 F1 score . . . . .	55
15.3.6 Area Under the Receiver Operating Characteristic Curve (AU-ROC) . . . . .	55
15.3.7 Mean absolute error (MAE) . . . . .	56
15.3.8 Mean squared error (MSE) . . . . .	56
15.3.9 R2 score . . . . .	56
15.3.10 Online data competition submission score . . . . .	57
<b>16 Results</b>	<b>58</b>
16.1 Titanic . . . . .	58
16.2 Mobile price classification . . . . .	59
16.3 Australia rain prediction . . . . .	60
16.4 Random Acts of Pizza . . . . .	61
16.5 Pump it up . . . . .	62
16.6 Richter’s Predictor: Modeling Earthquake Damage . . . . .	63
16.7 Insurance prediction . . . . .	64
16.8 House Prices: Advanced Regression Techniques . . . . .	65
16.9 Restaurant Revenue Prediction . . . . .	66
16.10DengAI: Predicting Disease Spread . . . . .	66
16.11Bike Sharing Demand . . . . .	67
16.12Urban Air Pollution Challenge . . . . .	68
16.13Discussion . . . . .	68
<b>17 Conclusion</b>	<b>70</b>
<b>Bibliography</b>	<b>72</b>



# List of Figures

1	Flowchart of the data science process . . . . .	2
2	Example of changing the datatype of the column age from object to a numeric data type . . . . .	22
3	Example k-Nearest Neighbors with $k = 3$ . . . . .	23
4	Example of label encoding . . . . .	23
5	Example of the label encoder with shifted values . . . . .	24
6	Example of Deep Feature Synthesis for creating new features at different depths, with $d = 3$ , by using the relationships between the tables . . . . .	28
7	Example of a One Hot Encoding . . . . .	29
8	Example of a decision tree, where it is decided whether someone should take the bus or go walking. This is decided by first looking at the weather, then the time that is left and whether he or she is hungry so that he or she can walk by a caf[Pleaseinsertintopreamble] for some food. . . . .	47
9	Example of the support vector machine algorithm in two dimensions. The two classes, indicated as squares and circles, are separated with the two support vectors creating the maximum margin possible. . . .	48
10	Example of a logistic regression in machine learning. This is a binary classification problem where everything below the threshold gets classified as 0 and above the threshold gets classified as 1. The curve is called a sigmoid function. . . . .	49
11	train/validation/test split . . . . .	52
12	Confusion matrix . . . . .	54
13	Example of ROC curve and AUC . . . . .	56

# List of Tables

1	Confusion matrices Titanic dataset. TPOT (left) and Auto-sklearn (right). . . . .	58
2	Evaluation metrics Titanic dataset . . . . .	58
3	Confusion matrices mobile phone price classification. TPOT (on top) and Auto-sklearn (below). . . . .	59
4	Evaluation metrics mobile phone price classification . . . . .	60
5	Confusion matrices Australia rain dataset. TPOT (left) and Auto-sklearn (right). . . . .	60
6	Evaluation metrics Australia rain dataset . . . . .	61
7	Confusion matrices acts of pizza dataset. TPOT (left) and Auto-sklearn (right). . . . .	61
8	Evaluation metrics acts of pizza dataset . . . . .	62
9	Confusion matrices pump it up dataset. TPOT (on top) and Auto-sklearn (below). . . . .	62
10	Evaluation metrics pump it up dataset . . . . .	63
11	Confusion matrices earthquake dataset. TPOT (on top) and Auto-sklearn (below). . . . .	63
12	Evaluation metrics earthquake dataset . . . . .	64
13	Confusion matrices insurance prediction dataset. TPOT (left) and Auto-sklearn (right). . . . .	64
14	Evaluation metrics insurance prediction dataset . . . . .	65
15	Evaluation metrics House price prediction . . . . .	65
16	Evaluation metrics restaurant dataset . . . . .	66
17	Evaluation metrics disease spread prediction . . . . .	67
18	Evaluation metrics Bike Sharing Demand . . . . .	67
19	Evaluation metrics urban air pollution challenge . . . . .	68

# 1 Introduction

## 1.1 About Avanade

Avanade is founded in the year 2000 by Accenture LLP and Microsoft Corporation. Avanade wants to take the lead as a digital innovator. They provide business solutions, innovative digital services and design-led experiences for its customers and clients. This is delivered by using Microsoft products and the power of people. Their professionals combine business, technology and industry expertise to build and deploy business solutions in order to realize results for Avanade's clients and also their customers. Globally, Avanade has around 29,000 people digitally connected divided over 23 countries [Avanade, nd].

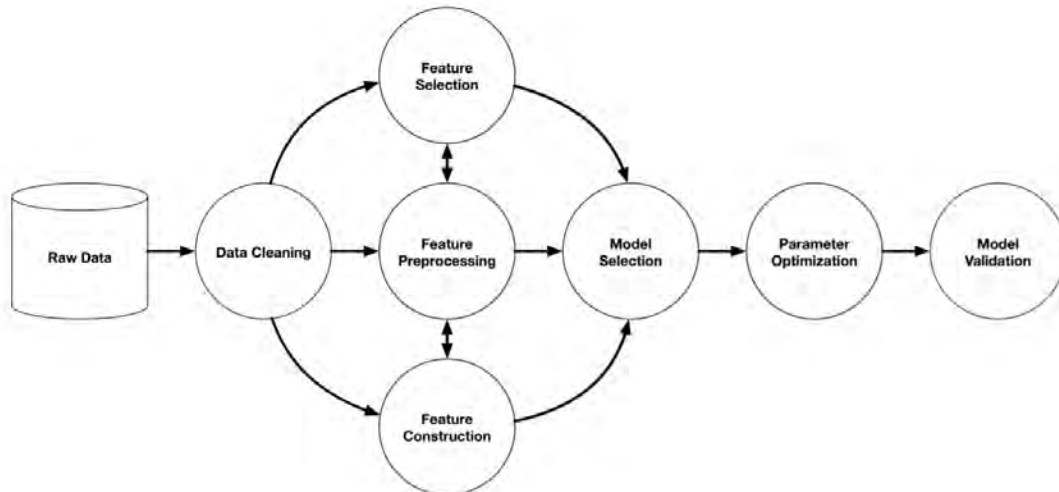
## 1.2 Business context and problem statement

Data discovery is the process of collecting and analyzing data to gain insights into the different trends and patterns that are within the data. It is an important first step in making critical business decisions based on the organization's data. During the data discovery process data is firstly gathered, then prepared (Data Preparation) and finally analyzed (Analytics, generating insights and visualize those insights). A business could act upon the insights generated from their data. Data discovery unlocks essential information which is important when making business decisions [MicroStrategy, nd].

Augmented data discovery (also named Augmented Data Science) is an increasingly applied Business Intelligence (BI) option for automatically preparing and processing business data. This is especially challenging for unstructured data from sources, such as Data Lakes, IoT feeds and customer service interactions. This can in return save valuable time for a business when a great part of this process is automated.

Augmented data discovery covers the entire automated data science process. One part of this process involves data preparation, such as cleaning and feature engineering. A second part of this process is data analytics where a model is selected and the hyperparameters are set. In the process of augmented data discovery, these

parts could be named augmented data preparation and augmented analytics. The flowchart in Figure 1 shows the steps within a data science process that we aim to automate.



**Figure 1:** Flowchart of the data science process

Augmented data preparation processes data for profiling, quality, cleaning and modeling, which is providing data for data discovery and analytics. Augmented analytics is the automation of creating insights or predictive models using Machine learning methods. This is motivated by the idea that augmented data discovery is a "rising BI capability for automatically preparing and organizing enterprise data for BI and Data Science." Avanade expects that augmented data discovery will play a more important role in the future. Gartner identifies Augmented Analytics among the top trends of Data & Analytics at the Gartner Data & Analytics Summit [Gartner, 2019].

Within the field of data science, much time is spent on the process of preparing and collecting the data. According to the estimations of experts and interviews, a data scientist can spend up to 80% of their time in collecting and preparing data before it can be explored and analyzed [Lohr, 2014]. This is a manual process that is very time-consuming. Nowadays data volumes are increasing and also becoming more complex, which makes analyzing the data even more difficult. By automating the iterative steps that a data scientist must perform, the time that is spent on the data discovery and preparation can be shortened by 50-80% [Agarwal, nd].

Due to these beneficial effects, Augmented data discovery is expected to play a more

important role in the future. Therefore, Avanade, as a leading digital innovator, is interested in researching this topic as this will most likely be an important part of their process in the nearby future. The deliverables of this research can afterwards be expanded and applied to their activities, which in return can be used with their clients.

### 1.3 Research question

The objective is to research the possibilities of augmented data discovery. In addition, this research is supported by creating a prototype. The goal of this prototype is to design and create an augmented data discovery solution with the Python programming language, which could be implemented into the Microsoft Cloud environment using Azure resources. Hence, the following research question emerges:

*"How and to what extent can a data discovery process be automated with an augmented data discovery solution using Azure resources?"*

For this research the following sub-questions are determined:

- What is Augmented Data Discovery and how is this deployed within an organization?
- Which pre-processing steps can be automated?
- How can feature engineering and feature selection be automated?
- How can model-building be automated?
- What is the trade-off between efficiency and quality?

### 1.4 Report Structure

The following chapters are discussed in the remainder of this thesis. Firstly, Chapter 2 provides the literature research, in which the related work of this topic is discussed. Secondly, Chapter 3 gives an explanation how such an automated data science tool could appear. Chapter 4 explains the implementation in Azure. Chapter 5 presents which datasets are used for this research. Chapter 6 gives a general overview of the approach in creating the prototype. The upcoming Chapters 7, 8, 9, 10, 11, 12, 13 and 14 will go deeper into the globally discussed approach per topic. Next in Chapter 15, the experimental setup is explained in which also the evaluation metrics of the prototype are discussed. In Chapter 16 the results of the performed experiments are presented and discussed. Finally, in Chapter 17 the conclusion of this research is given.

## 2 Literature review

This section discusses relevant literature on the subject. Augmented data discovery is a relatively new topic and is seen as an important application in the near future. Therefore, not too much literature is available on the subject. There are not many researchers that have done similar research on this topic. One similar research on augmented data science was found. This research is conducted by [Uzunalioglu et al., 2019]. Their solution introduces a set of modules that is uncovering data structure and data quality, generating new features and identifying and tuning a model for the specific task. To find sufficient literature, the subject is split into separate parts. More literature could be found on the following individual topics:

- Pre-processing
- Feature engineering
- Modeling

### 2.1 Pre-processing

Data pre-processing is the first step in the data science process. Here the raw data is transformed into a more suitable format for machine learning models to understand. [García et al., 2015] have published an educational book that describes the pre-processing part of data mining in a very extensive manner. This can be used to get a clear view of all pre-processing steps that could be considered in the augmented data discovery solution. From there it can be investigated whether these steps can be automated or not and which pre-processing methods are best to be automated. These are pre-processing steps like handling missing values, extreme values and data transformations. [Bilalli et al., 2016] show that pre-processing can be done by an automated approach, using ideas from meta-learning. They considered many pre-processing techniques and data mining algorithms. They built a tool in Weka, which makes it possible for non-experts in data science to perform data pre-processing.

## 2.2 Feature engineering

Another important and time-consuming data science procedure is feature engineering. Various research is done to automate this process, whereby several different systems are developed. [Kanter and Veeramachaneni, 2015] developed the Data Science Machine. This machine can derive predictive models automatically from raw data. To achieve that, they first developed the Deep Feature Synthesis algorithm to automatically generate features for relational datasets. Secondly, they implemented a machine learning pipeline which is generalizable. They created a framework called Featuretools, which is implemented into Python. [FeatureLabs, 2019] provides the documentation of deep feature synthesis. [Lam et al., 2017] introduced a system that they called One Button Machine. This system performs automatic feature discovery in relational databases. The One Button Machine automatically applies advanced data transformations and performs the joining of database tables to extract useful features from the data. [Khurana et al., 2016] presents a novel system that performs feature engineering automatically for supervised learning. This system is called "Cognito". Cognito explores various feature constructions, meanwhile it is maximizing the accuracy.

## 2.3 Modeling

When the dataset is fully prepared, a model can be constructed. There are many different algorithms to choose from and it is the data scientist's task to determine what is the best model for the problem and set its hyperparameters. [Thornton et al., 2013] showed that selecting a model and setting the hyperparameters can be done using an automated approach. They do this using recent innovations within Bayesian optimization. They show that their method performed better in most cases than standard selection methods. Their solution is built in WEKA. [Feurer et al., 2015a] introduced a new robust AutoML system, which is based on scikit-learn. They follow and extend the method introduced by [Thornton et al., 2013]. Their method is called Auto-Sklearn. Similarly, Auto-Sklearn uses Bayesian optimization in order to find the best combination of models, feature pre-processors and hyperparameters to maximize the accuracy. [Olson et al., 2016] introduced a tree-based pipeline optimization tool in Python, which uses a version of genetic programming in order to maximize the classification accuracy. This was one of the first automated machine learning methods. They wrote several papers on the topic in 2016 and 2017 and won two best paper awards. Both Auto-sklearn and tree-based pipeline optimization tool are available in Python and are improved since their original paper was published.

## 3 Augmented Data Science Tool

This chapter discusses how an augmented data science tool could appear. Its step-by-step process is explained. Which steps can be automated and when does the user of the tool need to provide input or manual adjustments. A tool like this will not be fully automated as the input of a data scientist is important. It is not the intention for such a tool to be the replacement of a data scientist, but rather a supporting tool. For describing this tool let's assume it has a working graphical user interface (GUI) in which the user can perform certain operations.

### 3.1 Process description

The first step in a data science project is to formulate a question within the data problem. This question can either belong to a classification or a regression problem, in which a certain variable must be predicted based on the available data. Secondly, the data must be fed to the tool, this can be done using the GUI in which a dataset for training can be uploaded to the tool. Python can handle different types of files so multiple options are possible here, for example CSV files, JSON files, excel files or a table from an HTML document. Python has also the ability to interact with databases and retrieve tables from there. Also, when available, an unlabeled dataset (e.g. test set for prediction) can be uploaded.

Once the dataset of choice is uploaded, the user should specify the index column and the target variable column. It should be indicated by the user when no unique index column exists, then an index will be created. The target variable is the variable that needs to be predicted.

When the index and target variable are specified, an overview will be presented to the user with some information about the data per feature, such as the data types, percentage of missing values and some descriptive statistics. Now the user can make decisions based on the presented data, but is not obligated to do so. Features can be removed if the user finds those irrelevant. Features can be removed with a missing value percentage above a certain threshold specified by the user. The user can check whether the data types are incorrectly inferred. If this is the case, the user has the



option to adjust the data type of the feature in question into the correct data type, which will be beneficial for the remaining data science process.

At this point, the data will most likely not be clean and immediately ready for a model to train. So the data needs to be pre-processed. The pre-processing of the data will deal with missing values, extreme values (outliers), performing data transformations and perhaps with help of natural language processing. This will be done using certain methods that are intended for such actions. This will be an automated process.

After pre-processing, the next step is feature engineering. For feature engineering, an automated method discussed in the literature review will be used. How this exactly works will be explained more extensively in Chapter 9. Before modeling a selection out of the existing features must be made. Therefore several feature selection methods are used.

Finally, a model can be built from the resulting dataset. This is done using an automated model building method, several of these methods exist and are discussed in the literature review. The final created model with its hyperparameters settings will be exported so the model can be used at another time.

The trained model can be used on the test set, that is the unlabeled dataset for which a prediction is desired. After following the same pre-processing steps the model can be used to create a prediction on the test set. The generated prediction will be exported as a CSV file. That prediction contains the index and the predicted values.

## **3.2 Prototype**

For this research, a prototype will be developed to show whether an augmented data science tool as described is possible. The goal is to create a tool that comes close to the process explained above. The prototype will be slightly less extensive and simpler than discussed in the process description. This is due to the limited time for this project. In the upcoming chapters, the approach of the prototype will be explained, which methods are considered, which methods are chosen and why those methods are chosen. In addition, it is investigated how such a solution could be operative using Azure resources.

## 4 Azure implementation

Avanade, as a Microsoft partner, is using Azure resources for their working activities. One condition of the prototype is that it should be able to run in the Microsoft cloud environment using Azure resources.

### 4.1 Microsoft Azure

Microsoft Azure is a set of cloud services that is growing constantly and is helping organizations providing in all sorts of business needs, which helps businesses taking on their challenges. Azure gives the freedom to use your favorite tools and frameworks in order to develop, manage and deploy applications on a massive global network [Microsoft, ndc].

### 4.2 Used Azure resources

In order to run the created prototype, three Azure resources are used. These are the following resources:

- Azure Databricks
- Azure Data Lake Storage Gen2
- Azure Data Factory

#### 4.2.1 Azure Databricks

Databricks is an analytics platform, which is optimized for the Microsoft Azure cloud services platform. It enables collaboration between data engineers, business analysts and data scientists by providing streamlined workflows and an interactive workspace integrated into Azure. It appears similar to Jupyter notebook, in which code can run one piece at a time in a notebook environment. It supports multiple programming languages, such as R and Python [Microsoft, ndd].

### **4.2.2 Azure Data Lake Storage Gen2**

Azure Data Lake Storage Gen2 is used for storing data. It is a cost-effective and highly scalable solution. Data Lake Storage Gen2 is optimized for analytics workloads. Here for example csv files are stored, which can be processed by the script created in Databricks [Microsoft, ndb].

### **4.2.3 Azure Data Factory**

Azure Data Factory is a cloud-based extraction, transformation and load (ETL) and data integration service. Data Factory allows for created data-driven workflows, called pipelines within Data Factory, which can perform ETL operations on the data. Complex ETL processes can be build by using services such as Azure Databricks and Azure Data Lake Storage Gen2 [Microsoft, nda].

## **4.3 Prototype**

For the created prototype the datasets are stored in Azure Data Lake Storage Gen2. The Python code is created in Azure Databricks. Azure Data Factory is used to create the workflow. Thus the pipeline, which in this case is the Databricks notebook. This pipeline retrieves the data from the Data Lake Storage and performs analysis on the data. In Azure Data Factory, certain parameters could be specified for the pipeline to be processed. This could be parameters such as the index, target variable and other parameters that need to be manually set.

# 5 Data

In this chapter, the conditions that the datasets should meet and the different datasets that are used in the experiments are described. For this research open-source datasets are used. The used datasets are from the following public data platforms:

- Kaggle (<https://www.kaggle.com/>)
- Driven Data (<https://www.drivendata.org/>)
- Zindi (<https://zindi.africa/competitions>)

## 5.1 Conditions

There are several conditions when selecting a dataset. One of these conditions is that there should be diversity in the datasets. The different datasets contain for example not solely numerical values or categorical values, but rather a mix of different data types. Both binary and multiclass classification problems should be treated. Next to that, the dataset should be large enough to support the process. Also, the dataset should not be perfect. It should contain noise such as outliers or missing values, here lies the challenge in automating these pre-processing steps. Contrarily, it should not be a dataset that is too "dirty" that it is not usable at all.

In addition, there are some limitations to the prototype which creates some additional conditions which the dataset should meet in order to be processed by the created prototype. The dataset must be a single CSV or JSON file for the training set and if available an additional CSV or JSON file for the test set. Also, the task must be a classification or regression problem, containing one single target variable. This can be a binary or multi-class problem in case of a classification problem. Also, natural language processing is not taken into account by the prototype.

## 5.2 Dataset 1 - Titanic

The first dataset that is used, is the Titanic dataset from Kaggle. This was the dataset that was first used to create this tool. This dataset was chosen because it

is simple, it is not clean and it is small. A small dataset was very convenient for creating the tool, this speeds up running all created methods and functions during the testing phase.

This is a simple competition especially for people that want to get acquainted with data science. It is a small dataset that has different data types: text, numeric, categorical and boolean data. It is not a clean dataset, it also contains missing values. It is a binary classification problem, in which the target is to predict whether a passenger has survived or not based on the provided data. The data contains information about the passenger such as age, class, gender, etc.

The data competition is available on:  
<https://www.kaggle.com/c/titanic/>

### **5.3 Dataset 2 - Mobile price classification**

The second dataset was also a small dataset from Kaggle, named mobile price classification. This dataset was mainly chosen to try another dataset than the Titanic dataset to see how the tool would handle this. There was not a competition that could be entered for this dataset.

This dataset consists of all sorts of mobile phone information. The target variable is categorically named price range. This price range varies from low = 0 till 3 = very high. To predict this, the data set provides information on the mobile phones, such as bluetooth, Wi-Fi, 4G, RAM, battery power, etc.

The dataset is available on:  
<https://www.kaggle.com/iabhishekofficial/mobile-price-classification>

### **5.4 Dataset 3 - Australia rain prediction**

The third dataset is about predicting the rain in Australia. This dataset was from Kaggle but it was not a competition. Also, this dataset was used to see how the tool would handle this different dataset. It was interesting as this was a significantly larger dataset than the first two.

The dataset contains about 10 years of daily weather observations from multiple weather stations, so quite a large dataset. The target is to predict whether it is going to rain tomorrow or not, so yes or no. In the dataset various information is given,

such as location, rainfall that day, temperature, etc.

The dataset is available on:

<https://www.kaggle.com/jsphyg/weather-dataset-rattle-package>

## 5.5 dataset 4 - Random Acts of Pizza

The fourth dataset is from a data competition that is organized purely for fun, where people on the Reddit community requested a pizza via a posted message. All kind of data was gathered from these posts the people made, such as the number of reactions to the post, number of up-votes, number of down-votes, time of post, etc. Eventually, the goal is to predict whether the person that posted the request for a pizza successfully received a pizza from someone. This dataset is quite unbalanced as most people did not receive a pizza, which creates an additional challenge.

The data competition is available on:

<https://www.kaggle.com/c/random-acts-of-pizza>

## 5.6 Dataset 5 - Pump it up

The fifth dataset contains data on the state of water pumps in Tanzania. Based on the provided data the goal is to classify the pump as either: functional, functional needs repair or non-functional. There are records of almost 60.000 different water pumps available in the training data. To classify this multiclass problem there is data available such as what kind of pump is operating, when the pump was installed, how the pump is managed, region, etc.

The data competition is available on:

<https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/>

## 5.7 Dataset 6 - Richter's Predictor: Modeling Earthquake Damage

The sixth dataset relates to the earthquake in Nepal in 2015. The goal is to predict the level of damage to the buildings caused by that earthquake where 1 = low damage, 2 = medium damage and 3 = almost complete destruction. The training data has information about more than 260.000 buildings. The data consists of all sorts of information about the building characteristics, such as the number of floors, age of the building, what the building is made of, etc. There are 38 explanatory

variables in total.

The data competition is available on:

<https://www.drivendata.org/competitions/57/nepal-earthquake/>

## 5.8 Dataset 7 - Insurance prediction

The seventh dataset holds information on insurances for buildings in Nigeria. A company offers insurance policies for buildings in that area for damage caused by, for example, fire, vandalism or storm. The goal of this data competition is to predict for a building if it will have an insurance claim during a specified period. This is a binary classification problem where it should be predicted if there is at least one claim in the insured period (1) or none (0). In order to predict this, a predictive model is build based on the building characteristics with a total of 14 features. The training data contains information little over 7000 buildings, thus not a large dataset. The data is slightly unbalanced. Most buildings do not file a claim in the given period.

The data competition is available on:

<https://zindi.africa/competitions/data-science-nigeria-2019-challenge-1-insurance-predi>

## 5.9 Dataset 8 - House Prices: Advanced Regression Techniques

Dataset 8 is a playground data competition on Kaggle, which is convenient for the first time testing the automated regression approach. It involves a regression problem where the target is to predict the house price. In order to predict this price, the dataset provides all sorts of information about the houses, such as location, build year, information on the rooms, etc. 79 explanatory variables are describing the houses. The training data contains data of 1460 houses, so not a large dataset.

The data competition is available on:

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview>

## 5.10 Dataset 9 - Restaurant Revenue Prediction

The ninth dataset is from a data competition that is created by TFI. TFI is the company behind many popular brands such as Burger King. The goal of this competition is to predict the annual revenue of a restaurant based on the provided information. The training data consist of demographic, real estate and commercial

data. Remarkably, in this data competition training data only contains information on 137 restaurants while the test set contains over 100.000 restaurants. There are 41 features available in the dataset to train on. So there is not much data to train on, but a lot to predict.

The data competition is available on:

<https://www.kaggle.com/c/restaurant-revenue-prediction/overview>

## 5.11 Dataset 10 - DengAI: Predicting Disease Spread

The tenth dataset is about the Dengue fever. Dengue fever is a disease that is carried by mosquitoes that occurs in tropical parts of the world. It causes flu-like symptoms in mild cases. Severe cases can cause severe bleeding or even death. The target is to predict the weekly cases of dengue fever in a city in a certain week. These cities are in San Juan, Puerto Rico, Iquitos and Peru. To predict this, environmental data is used. The goal of this competition is to get an understanding of the relationship between dengue fever and climate. It is a really small dataset, where the training set consists of only 1456 records with 24 explanatory variables.

The data competition is available on:

<https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/page/80/>

## 5.12 Dataset 11 - Bike Sharing Demand

Dataset 11 is about a bike sharing system. A bike-sharing system is where people can rent a bike from a particular location and they can return it to another location. These systems are often introduced in large cities around the world. For this competition, the data is used from the bike sharing system in Washington. With this competition it is the goal to study the mobility in the city by predicting the hourly demand of bicycles based on the weather data. The data to work with consists of the date and weather data, such as temperature, wind speed and humidity. The training set contains data of every hour from the 1<sup>st</sup> of January 2011 until 19 December 2012.

The data competition is available on:

<https://www.kaggle.com/c/bike-sharing-demand/overview>



## 5.13 Dataset 12 - Urban Air Pollution

Dataset 12 is about air pollution. This is a data competition on Zindi that has recently started (3 July). This competition is about the air quality in Africa. In many African cities, the air quality is getting worse. The challenge within this competition lays in finding ways to track the air quality by diving deeper into the data. In order to predict the air quality, the dataset consists of weather data and observations from a satellite that is tracking pollutants in the atmosphere. The air quality is measured in PM2.5 concentration, this is a common measure for air quality. In the training set, there are over 30.000 daily observations with 80 explanatory features.

The data competition is available on:

<https://zindi.africa/competitions/zindiweekendz-learning-urban-air-pollution-challenge>

# 6 Approach

This chapter explains the approach of the data science process, which we aim to automate. The steps from pre-processing until modeling are discussed. Multiple methods will be suggested per topic. In the upcoming chapters, it will be explained per topic which methods are used within the created prototype and why those methods are used.

## 6.1 Pre-processing

The data mining steps described in this chapter are mainly based on [García et al., 2015]. From this, the following pre-processing steps emerge:

- Recognizing Data types
- Data Cleaning
  - Missing Value Imputation
  - Outlier Handling
- Data Transformation & Normalization
- Dimensionality reduction

### 6.1.1 Recognizing Data types

Within the same dataset, there are usually different data types. In order to perform proper pre-processing these data types should be recognized. In Python there are parsers to automatically infer the datatype. Using pandas in Python the following data types can be automatically inferred:

- Integer
- Float
- Boolean
- Object (string or mixed type)

There is also the data type category, but this data type will not be automatically recognized. This will be recognized as an integer or object data type depending on how the categorical feature is represented in the column. This parser in Python works well, however this type inference does not allow wrong records. For example, a text value which has been filled in incorrectly in a numerical column, such as age, will cause the column to be labeled as an object type. Therefore, a manual step must be included in the prototype to ensure the data types will be correctly recognized.

### **6.1.2 Data Cleaning**

When the data is successfully integrated into a single dataset, the data must be cleaned as it is most likely not error-free. A chunk of the data may be "dirty data". Dirty data include noisy data, such as outliers, or missing data. When a large proportion of the data is dirty, less reliable models will be produced. Therefore, dirty data should be cleaned.

#### **Missing Values**

A real-life dataset contains most likely missing values. There are several schemes to handle these missing values. The simplest approach is to discard the instances with missing values, which could be a reasonable method if the dataset contains only a small amount of missing values. Another approach is the imputation of missing values, which aims to fill the missing values with estimated values. These could be a very basic imputation method, such as imputing the mean or median for numeric values and the mode for categorical values. Also, machine learning based methods could be used. An algorithm, such as K-Nearest Neighbors or Support Vector Machines, could be used to predict the value of the missing records.

Not a single imputation method performs best on all classifying problems, so for each different dataset, another method could achieve better results. A further explanation of this topic is given in Chapter 7.

#### **Outlier Detection and handling**

An outlier is an observation point that differs significantly from the other observations within the dataset. An outlier could be caused by a measurement error or due to the variability of what is observed or measured. Possible methods of handling outliers that are considered for applying in the prototype are:

- Median Absolute Deviation (MAD)
- Interquartile range (IQR)

- Z-score

More on these methods is explained in Chapter 8.

### 6.1.3 Data Transformations

The collected data, as it is, may not be ideal for obtaining accurate predictive models. The attributes are "raw" and they still have a meaning from their original domain. Transforming (also referred to as scaling) the original attributes, could help to improve the predictive model.

#### Data Normalization

A basic transformation strategy is normalization. With normalization, no new attributes are generated, but the distribution of the original values is transformed into a new set of values. Three popular normalization methods are:

- Min-Max Normalization (Min-Max Scaling)
- Z-score Normalization (Standardization)
- Robust Normalization (Robust Scaling)

In Chapter 11 these methods are explained.

#### Data Transformation

In contrast to normalization, data transformation aims to create new attributes. This is often called transforming the attributes. Usually, the original raw attributes are converted into another format using various mathematical formulas. There are many different transformation methods, some of those are:

- Linear Transformation
- Box-Cox Transformation
- Polynomial Feature Transformation

For normalization and transformation, there is no universally best approach. It has to be investigated what transformation method(s) suits an automated solution best. More on these methods is explained in Chapter 11.

#### 6.1.4 Dimensionality reduction

One major problem in data mining for large datasets, which has many potential predictors, is the "curse of dimensionality". Because of the computational complexity of the data mining algorithms, dimensionality can become a real hurdle for efficiency. To tackle this problem, there are some dimension reducer methods developed. Some of these methods are:

- Principal Component Analysis (PCA)
- Truncated SVD
- Kernel PCA
- Independent component analysis (ICA)

From these methods, Principal Component Analysis is perhaps the most well-known method. More on this is discussed in Chapter 13.

## 6.2 Feature Engineering

In the literature research, various papers were mentioned that proposed an automated feature engineering solution. One of those solutions was called "Cognito" [Khurana et al., 2016]. They constructed software whereby the user should upload a file and then features could be created. So this cannot be used within Python, in which this augmented data science solution is developed. Another paper discussed the One Button Machine [Lam et al., 2017]. This framework is not available to the public. The last method was called Deep Feature Synthesis [Kanter and Veeramachaneni, 2015]. The framework they created, called Featuretools, is available in Python. Therefore, this accessible method is most promising for an augmented data science solution build in Python. More on Deep Feature Synthesis and how it works is discussed in Chapter 9.

## 6.3 Feature Selection

During feature selection, the features that eventually will be used for training the model are selected. When there are too many features this could negatively influence the model. After performing deep feature synthesis many different features are created, this could be several hundred additional features. Not all of these features are relevant to the model. The benefits of decreasing the number of features are: it reduces overfitting, it shortens training time and it can improve the model performance [Shaikh, 2018]. The following methods are considered when selecting features:

- Removing highly correlated features
- Removing features with low variance
- Select k best features
- Select top n percentile of features
- Recursive feature elimination

This will be further explained in Chapter 12.

## 6.4 Modeling

There are some automated Machine learning packages within Python, which are discussed in the related literature. Two promising methods for modeling found during the literature research are Auto-sklearn [Feurer et al., 2015a] and Tree-based Pipeline Optimization Tool (TPOT) [Olson et al., 2016]. These automated machine learning methods frees the data scientist from selecting the model and its hyperparameters. Both are available within Python and are a great addition to the augmented data discovery solution, which is solely built in Python. These frameworks also cover some parts of the pre-processing. In Chapter 10 these two modeling approaches are extensively explained.

## 7 Missing Values

In this chapter, the missing value imputation methods, that are considered using, are discussed. These methods consist of a simple imputation method and a machine learning based method, namely k-Nearest Neighbors.

### 7.1 Simple imputation

One of the methods that is considered for implementing in the prototype is a simple imputation method. With this simple imputation method, the median is imputed for numeric values and the mode for categorical values. The median is preferred over the mean because the data is not known upfront. Therefore, nothing could be assumed about possible extreme outliers which can drastically influence the mean. Thus, the median seems to be more appropriate in this case. Although this is a very simple and fast method. This method does have some disadvantages. One of the limitations of this imputation method is that imputing the same value, median or mode, for all missing values will lead to a reduced variance and resulting in an unbalanced dataset. This is especially the case when there are many missing values.

For the optimal performance of this method, it is important that the data types of the columns are correctly inferred. Because this approach will be automated, the parser in Python is used to retrieve the datatype and based on the data type either the median for numeric values or the mode for categorical values is used. The parser in Python cannot handle dirty records as was mentioned earlier in Chapter 6. For example, a numeric column could be incorrectly labeled as an object type. When this occurs the mode will be imputed in that column instead of the median. This could negatively influence the results while modeling. To avoid this issue, an additional manual step is added in which the user can select columns where the data type needs to be changed. When the datatype is wrongly labeled as an object, where it should be numeric, then the column could be indicated by the user to convert the data type to numeric. The values that are not numeric in that column will be deleted, so a missing value remains which will be imputed again. An example is presented in Figure 2. Here the datatype was recognized as an object due to the dirty record 'a' in the age column. When the datatype gets converted this value is

removed from the dataset.

Age
32
25
41
a
18

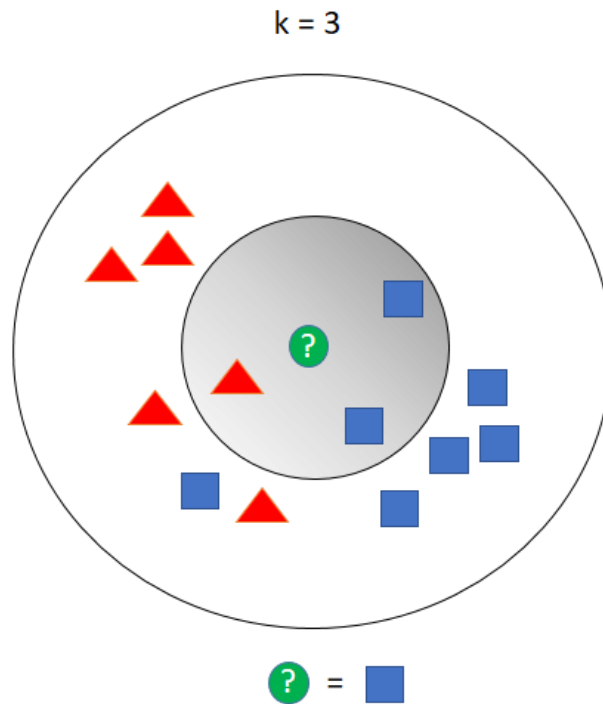
Age
32
25
41
NaN
18

**Figure 2:** Example of changing the datatype of the column age from object to a numeric data type

## 7.2 k-Nearest Neighbors

Another method that is considered for imputing missing values is the k-nearest neighbors imputation approach. The k-Nearest Neighbors algorithm is a classification and regression method which finds the k nearest training examples within the given feature space, where k is specified by the user. A visual example is presented in Figure 3 with  $k = 3$ . In the image, the green circle is the unknown data point. When looking at the three closest neighbors, the majority are blue squares. Thus, the unknown green circle gets classified as a blue square. Sklearn has a class that performs this imputation named the KNNImputer. With this method, the missing values are imputed using k-Nearest Neighbors. This class uses the euclidean distance metric by default that supports missing values. The missing values are imputed using values from the nearest neighbors that do have a value for that feature [Scikit-Learn, ndb].





**Figure 3:** Example k-Nearest Neighbors with  $k = 3$

For the KNNImputer it is required that all data is numeric. Thus all categorical/text features have to be encoded. This can be done by encoding every category of the categorical features using a label encoder [Scikit-Learn, ndf]. Then the target labels are encoded with values between 0 and  $n-1$ . With  $n$  the different number of labels within the categorical feature. Figure 4 provides an example of this process. Here the three different fruits are encoded by 0, 1 and 2.

fruit
apple
pear
banana
apple
banana


→

fruit_enc
0
2
1
0
1

**Figure 4:** Example of label encoding

The main drawback of encoding categorical features with the label encoder is that it can cause errors when performing on the test set that has an additional category within a feature. Using the label encoder this can cause the different categories to be

numbered in another fashion. An example with the fruit is given in Figure 5. Here one apple has been replaced by avocado which causes the numbering to be shifted. During training, banana was 1, but in the test set banana is now 2.



fruit
apple
pear
banana
avocado
banana

fruit_enc
0
3
2
1
2

**Figure 5:** Example of the label encoder with shifted values

This will result in bad modeling because the model is trained on different values. This can be solved by encoding entirely on the training and test set. However, then the model must be trained again every time there is a new test set to get the correct encoding. This is not very efficient for business purposes. Especially with large datasets which can lead to enormous training times.

### 7.3 Prototype

For the prototype the simple imputation method and the KNN imputation method were considered. Based on the arguments discussed above, using the KNN imputation in an augmented data discovery solution is not very practical. This is also the case with other imputation methods that use a machine learning based approach, such as k-means clustering. It can be used with datasets that only contain numerical values or are already encoded when retrieved from the data source. Therefore the simple approach is preferred for the prototype. This method will work on all sorts of datasets. It will not be the best imputation method for every dataset, but it is overall a well-working imputation method.

# 8 Outliers

In this chapter, the different outlier handling methods that are considered are discussed. These are the following methods:

- Median Absolute Deviation (MAD)
- Interquartile range (IQR)
- Z-score

## 8.1 Median Absolute Deviation (MAD)

This is a relatively simple outlier detection method. The median of the residuals is calculated in order to detect outliers. First, the absolute difference between all values and the median is calculated. From those values, the median is taken, which yields the MAD (Equation 1). This MAD value is used for detecting outliers by observing whether a number is a certain value away from the MAD. The default threshold for this distance is 3 MAD.

$$MAD = \text{median}(|X_i - \text{median}(X)|) \quad (1)$$

This method could be effective. However, it can also be too aggressive when classifying values that are not that much different than the other values. Another downside is that if half of the data has the same value, then MAD would be 0, causing all values that are different to be detected as an outlier [Oracle, nd].

## 8.2 interquartile range (IQR)

Another method for detecting outliers is the interquartile range (IQR) method [Brownlee, 2018]. This method defines the middle half of the data between the 25th and 75th percentiles (Equation 2).

$$IQR = Q_3 - Q_1 \quad (2)$$

The IQR method identifies outliers by observing which observation points are outside the lower bound and upper bound. The lower and upper bound are determined by

taking the IQR times a factor  $k$  (commonly 1.5) below the 25th percentile or above the 75th percentile. This yields:

$$\text{Lowerbound} = Q_1 - k * IQR \quad (3)$$

$$\text{Upperbound} = Q_3 + k * IQR \quad (4)$$

The advantage of using the interquartile range with augmented data preparation is that this method can be used regardless of the distribution, which is useful when generalizing a method for different datasets.

### 8.3 Z-score

The Z-score is a metric that helps to understand whether a data point is smaller or greater than the mean and how far away from the mean. The Z-score can tell how many standard deviations a data point is from the mean. The Z-score is given by Equation 5, where  $\mu$  is the mean and  $\sigma$  is the standard deviation [Maini, nd].

$$Z_{score} = \frac{x - \mu}{\sigma} \quad (5)$$

When a data point is more than 3 standard deviations away from the mean, then this point is considered an outlier. A disadvantage of this method is, it assumes the data is normally distributed. This is not ideal for an augmented data science solution as we do not know the distribution beforehand.

### 8.4 Prototype

After investigating the three methods explained in this chapter, the IQR method is implemented in the prototype. The IQR method does not depend on the distribution of the data, which is unknown. Unlike Z-score which assumes the data is normally distributed. The MAD method also has its disadvantages when many data points have the same values. Therefore, the IQR method seems the most appropriate method for automated data preparation.

## 9 Feature Engineering

This chapter discusses the method used for automated feature engineering. For feature engineering, the method proposed by [Kanter and Veeramachaneni, 2015] named Deep Feature Synthesis is used.

### 9.1 Deep Feature Synthesis

Featuretools [FeatureLabs, 2019] is the framework build in Python that performs Deep Feature Synthesis (DFS). DFS is an algorithm that can automatically generate features derived from the relationships between the data points within a dataset. The algorithm follows the relationships within the data to a base field, then it applies mathematical functions along that path to create a new feature. DFS stacks calculations consecutively, each feature can be defined as having a certain depth  $d$ . DFS works especially well with relational tables, but can also work well with single data tables [Kanter, 2018]. [Gordon, 2018] gives an extensive example of the use of deep feature synthesis and several encountered problems on the way using a single data table. His full notebook is available on [Github, 2018].

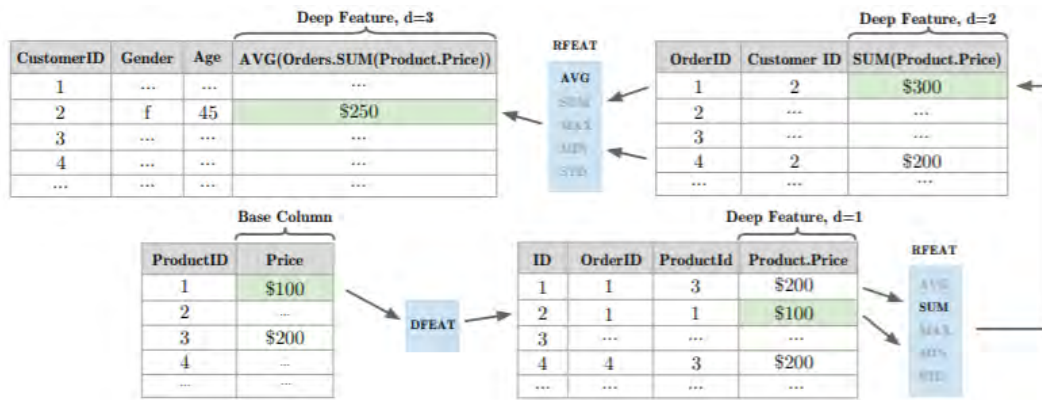
Featuretools takes instances with a relationship as input and outputs a single newly created feature. Some of the aggregation function used within DFS are:

- Sum
- Mean
- Max/Min
- Standard deviation
- Mode
- Count

Featuretools can also derive multiple features from a single feature, such as date. For a date object, it will automatically generate the features Day, Month, Year, DayOfWeek and IsWeekend features. These are often helpful features in modeling.

### 9.1.1 Example

An example is given in Figure 6 to illustrate how DFS works. The data in this example is about a store that has orders, customers and products that all have their own table. In the example, the average order size per customer is calculated. Firstly they start with the product, which holds the ProductID and the Price. The direct feature (dfeat) is calculated to add the price to the product orders table. Then the relational feature (rfeat) is calculated by taking the sum of the product prices per OrderID,  $SUM(Product.Price)$ . Then another relational feature is calculated to calculate the average order size per CustomerID,  $AVG(Orders.SUM(Product.Price))$  [Kanter and Veeramachaneni, 2015].



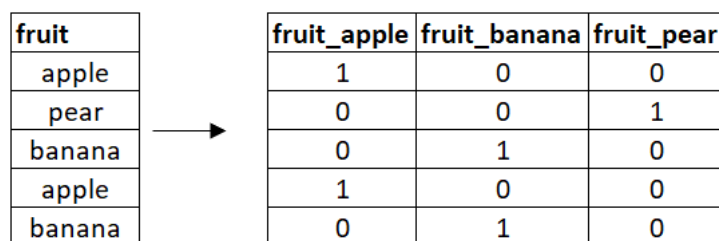
**Figure 6:** Example of Deep Feature Synthesis for creating new features at different depths, with  $d = 3$ , by using the relationships between the tables

In this small example, it is only calculating the average product price per customer, which is added as a feature in the dataset. In larger datasets DFS will create many more features. Then more aggregation functions are used and there are more features. For example, the aggregation functions mentioned above. This can cause DFS to create hundreds of new features, or even more if desired. This is done in a relatively short period of time whereas someone would do this manually, it would take quite some time to create that many features.

### 9.1.2 Feature encoding

Machine learning models need numeric data in order to work. Therefore text data should be encoded. Featuretools can do this for you, using a One Hot Encoding. This One Hot Encoding creates a binary column for each category within the categorical feature. A visualized example is presented in Figure 7. Because there are three types of fruit, it creates three binary columns with 0's and 1's indicating whether a specific

fruit is in that instance or not.



fruit
apple
pear
banana
apple
banana

fruit_apple	fruit_banana	fruit_pear
1	0	0
0	0	1
0	1	0
1	0	0
0	1	0

**Figure 7:** Example of a One Hot Encoding

## 9.2 Prototype

DFS is implemented in the prototype. Although in the prototype only single tables are used, DFS can still create some valuable features. A manual step is included in the prototype in which the user should indicate what feature it wants to aggregate on in relation to the other features. The user could indicate a single feature, but also more or none.

# 10 Modeling

In this chapter, two automated modeling approaches are discussed, namely TPOT and Auto-sklearn. There are two main challenges with automated modeling in data science. One challenge is that no single machine learning algorithm performs best on all datasets. Another problem is that some machine learning algorithms are highly dependent on hyperparameter optimization. The two automated modeling approaches, discussed in the next two paragraphs, do tackle these problems. For each of the approaches, the used methods are briefly explained. A further explanation of some of those methods is explained in the upcoming chapters.

## 10.1 Tree-based Pipeline Optimization Tool (TPOT)

TPOT [Olson et al., 2016] is an automated machine learning tool within Python that optimizes tree-based machine learning pipelines using genetic programming. There are several pipeline operators implemented in TPOT. All of these operators use existing implementations available in the scikit-learn package. The four main types of implemented pipeline operators are:

- Pre-processors
- Decomposition
- Feature Selection
- Models

### Pre-processors

TPOT implements the following four pre-processing operators from Sklearn:

- Min-Max Normalization (MinMaxScaler)
- Z-score Normalization (StandardScaler)
- Robust Normalization (RobustScaler)
- Polynomial Transformation (PolynomialFeatures)

These four transformation methods are explained in Chapter 11.



## Decomposition

TPOt implemented a Principal Component Analysis (PCA) using a randomized Singular Value Decomposition (SVD). This method and other decomposition methods are explained in Chapter 13.

## Feature Selection

The feature selection methods that are implemented in TPOt are:

- Select top k features (SelectKBest)
- Select top n percentile of features (SelectPercentile)
- Remove features with low variance (VarianceThreshold)
- Recursive feature elimination (RFE)

These four feature selection methods are explained in Chapter 12.

## Models

TPOt is focused solely on supervised learning models. They implemented the following supervised classification and regression operators:

- Decision Tree
- Random Forest
- Gradient Boosting
- Extreme Gradient Boosting (XGBoost)
- Linear Support Vector Machine (SVM)
- Logistic Regression
- k-Nearest Neighbors
- Linear regression
- Ridge regression
- Lasso regression

These models are further explained in Chapter 10.

### 10.1.1 Optimizing Tree-Based Pipelines

TPOT uses a genetic programming algorithm that is implemented in the Python package DEAP [Fortin et al., 2012]. The default settings of TPOT runs ten generations with a population size of 100. With these settings, the genetic programming algorithm is generating 100 random tree-based pipelines from which their accuracy is validated on the dataset. From these 100 pipelines, the top 20 from the population are selected for the next generation. This is done with the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) selection scheme, which is an evolutionary algorithm. Here pipelines are selected to maximize the accuracy while also minimizing the number of operators used in the pipeline. From the 20 selected pipelines, that are selected for the next generations, each creates five offspring, these are five copies of the pipeline. 5% of those created offspring cross over with other offspring using the one-point crossover method. Remaining offspring that are unaffected are randomly changed by a single point in the pipeline, insert mutation or shrink mutation is applied, each with a probability of 1/3. With mutation, some part of the pipeline gets altered. This causes to create slightly different pipelines, which are perhaps better. After each generation again the top 20 pipelines get selected and the process repeats for the new generations. Each generation the pipeline gets slightly altered, which has a chance of improving that pipeline [Olson et al., 2016].

## 10.2 Auto-sklearn

Auto-sklearn [Feurer et al., 2015a] is the other automated machine learning approach. It uses Bayesian optimization in order to find the best combination of models. Auto-sklearn uses an ensemble construction that is automated, this steps allows using all different classifiers or regressors that are found during the Bayesian optimization. The same pipeline operators as for TPOT are discussed for this approach:

- Pre-processors
- Decomposition
- Feature Selection
- Models

### Pre-processors

Auto-sklearn implemented the following three pre-processing operators quite similar to TPOT:

- Min-Max Normalization (MinMaxScaler)

- Z-score Normalization (StandardScaler)
- Polynomial Transformation (PolynomialFeatures)

Compared to TPOT, Auto-sklearn excludes Robust Normalization. The transformation methods are explained in Chapter 11.

### **Decomposition**

Unlike TPOT, Auto-sklearn implemented multiple decomposition methods, these are the following methods:

- Principal Component Analysis (PCA)
- Truncated SVD
- Kernel PCA
- Independent component analysis (ICA)

These methods are explained in Chapter 13.

### **Feature Selection**

In Auto-sklearn the following feature selection methods are implemented:

- Select top k features (SelectKBest)
- Select top n percentile of features (SelectPercentile)
- Recursive feature elimination
- Classification-based feature selection with Extremely Randomized Trees

These methods are explained in Chapter 12.

### **Models**

Auto sklearn implemented considerably more modeling methods than TPOT did. They implemented the following 15 supervised classification and regression operators:

- Gaussian Naive Bayes
- Bernoulli Naive Bayes
- Multinomial Naive Bayes
- Decision Tree

- Random Forest
- Extremely Randomized Trees
- Adaptive Boosting (AdaBoost)
- Gradient Boosting
- Linear Support Vector Machine (SVM)
- Kernel Support Vector Machine (SVM)
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Linear Classification via online stochastic gradient descent (SGD)
- Maximum Margin classification via Online Passive Aggressive algorithms
- k-Nearest Neighbors (kNN)
- Linear regression
- Ridge regression
- Lasso regression

These different classification and regression methods are briefly explained in Chapter 14.

### **10.2.1 Automated ensemble construction of models evaluated during optimization**

Auto-sklearn uses Bayesian hyperparameter optimization. This is a data-efficient method in finding the best hyperparameters for the model. However, with the goal of simply making a good prediction, it is quite a wasteful method due to losing all models during the optimization process, although these models did perform almost as good as the best-performing model. Auto-sklearn uses the created models rather than discarding those. They store the models and afterwards these models are used to construct an ensemble out of them. This automatically constructed ensemble does not have one single hyperparameter setting and is therefore less sensitive to overfitting and more robust.

Individual models are often outperformed by ensembles. Ensembles perform especially well when the individual models, of which the ensemble is constructed, are strong on

their own and if the individual models make uncorrelated errors. These individual models are different in nature and therefore correlated errors are unlikely. They found that the models found by Bayesian optimization creating a uniformly weighted ensemble out of them did not work well. Therefore, they use ensemble selection in order to optimize these weights in correspondence with [Caruana et al., 2004]. This is a greedy approach. It starts with an empty ensemble and then it iteratively adds a model that maximally increases the validation performance of the ensemble [Feurer et al., 2015a].

### 10.3 Comparison of TPOT against Auto-sklearn

In this chapter, two automated modeling approaches are discussed, TPOT and Auto-sklearn. These frameworks use a different approach. Each method has its own advantages and disadvantages.

The scaling operators used in both methods are almost the same. Concerning the composition methods, Auto-sklearn has the choice of 4 different operators while TPOT uses solely PCA. This creates more possible different pipelines for Auto-sklearn. The amount of models is also significantly larger within Auto-sklearn than TPOT. This is a advantage as this enables Auto-sklearn to deal with a larger variety of datasets.

Auto-sklearn uses an ensemble of multiple models. This often yields better modeling results. A disadvantage is that the model is rather complex, containing dozens of different models. This makes the best ensemble model difficult to interpret. TPOT on the other hand chooses for the best model a single classifier or regressor (e.g. Decision Tree or Random Forrest), which is easier to understand.

Another disadvantage of Auto-sklearn is that it can only run on Linux, unlike TPOT that can run on each operating system. This could be a rather inconvenient limitation.

The most important metric to compare these methods is to validate both the modeling performance. This is done in Chapter 16.

### 10.4 Prototype

For the prototype, both automated modeling approaches are used. This is done in order to make a comparison between the two frameworks based on the results. From the literature, it is not clear whether one method is better than the other.

Thus, both will be used and then a comparison can be made based on the results. So two versions of the prototypes are created, with one supporting TPOT and one supporting Auto-sklearn. The results are discussed in Chapter 16.

# 11 Data Transformation

In Chapter 6 several possible transformation methods are discussed, from which many are implemented in either TPOT or Auto-sklearn. In this chapter, those methods are explained more extensively.

## 11.1 Normalization

The three proposed normalization methods in Chapter 6 were:

- Min-Max Normalization (Min-Max Scaling)
- Z-score Normalization (Standardization)
- Robust Normalization (Scaling)

### 11.1.1 Min-Max Normalization (Min-Max Scaling)

Min-max normalization scales all the values of the feature within a certain scale, usually between 0 and 1 or -1 and 1. Equation 6 shows the formula of this method.

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (6)$$

### 11.1.2 Z-score normalization (Standardization)

This normalization method transforms the attribute values that they now present a standard deviation of 1 and a mean equal to 0. This normalization method centers the data by using the following formula seen in Equation 7, where  $\mu$  is the mean and  $\sigma$  is the standard deviation [García et al., 2015].

$$x_{scaled} = \frac{x - \mu}{\sigma} \quad (7)$$

### 11.1.3 Robust Normalization (Robust scaling)

This method is similar to Min-max normalization, it uses the interquartile range instead of the min and the max for scaling the data. Using the interquartile range instead of the min-max method makes it less sensitive for extreme values in the data. This method has the following formula seen in Equation 8.

$$x_{scaled} = \frac{x - Q_1(x)}{Q_3(x) - Q_1(x)} \quad (8)$$

#### 11.1.4 Prototype

Not a single normalization method performs best on every dataset. In the automated modeling approaches, that are used in the prototype, these methods are already implemented. They use all the methods in different pipelines to observe whether the model improved. So, this makes these frameworks very convenient for using transformations in an augmented data science solution.

## 11.2 Transformation

The discussed data transformation methods were:

- Linear Transformation
- Box-Cox Transformation
- Polynomial Feature Transformation

### 11.2.1 Linear transformation

The linear transformation is the simplest form of transformations. Some linear transformations are average, sum, rotations, etc. These are meaningful transformations. Lets explain this with the following example. If A is a set of attributes and B a subset of the attributes in A, with r being a set of real numbers. Then apply the following expression:

$$Z = r_1B_1 + r_2B_2 + \dots + r_mB_M \quad (9)$$

This derived a new attribute Z, by taking a linear combination of the attributes within B. It should be noted that even with a linear transformation, to find a proper transformation often some experiments are needed. Which makes this method not optimal for an automated approach [Lin, 2002].

### 11.2.2 Box-cox transformation

The Box-cox transformation is designed to fit the data automatically by transforming the continuous variable in an almost normal distribution [García et al., 2015]. This



seems useful for an automated solution. The big drawback is that this however only works for non-negative values, which makes this method not ideal for many datasets.

### 11.2.3 Polynomial feature transformation

This is an operator that generates features that are interacting using polynomial combinations between the numerical features. These transformations are generally created when one wants to include the assumption there is a nonlinear relationship between the numerical features and the target variable. This method is mostly used to add complexity to linear models with little features or when a dependency is suspected between one and another feature [Dorpe, 2018].

So from the given set of attributes  $x_1, x_2, \dots, x_n$  a derived attribute  $Y$  is computed from the set of existing features. If there is for example a two dimensional input sample of the form  $[x_1, x_2]$ . Then the created degree 2 polynomial features are:  $[1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$  [Dorpe, 2018].

[Lin, 2002] shows that when no knowledge is available, a transformation could be approximated with a polynomial transformation. This makes it a suitable method for an automated approach.

### 11.2.4 Prototype

From the discussed methods the polynomial features method seems the most promising for an automated data science solution. The polynomial features transformation is available in Sklearn and, similar to the normalization methods mentioned above, implemented in TPOT and Auto-sklearn. This, combined with the explanation above, makes this method the most convenient for implementing into the prototype.

# 12 Feature Selection

Selecting fewer features to eventually train the model, could positively influence the model. After DFS many features could be created, therefore feature selection methods are of great use to reduce some irrelevant features. In the approach (Chapter 6), some methods were mentioned for selecting features. Those were the following methods:

- Removing highly correlated features
- Removing features with low variance
- Select k best features
- Select top n percentile of features
- Recursive feature elimination

In this chapter, these methods will be explained more extensively.

## 12.1 Correlated features

To check for correlated features, a correlation matrix is created. In the correlation matrix, every feature is compared to each other feature using Pearson correlation coefficient. Pearson correlation coefficient is defined as follows:

$$\rho_{x,y} = \frac{E[(X - \mu_x)(Y - \mu_y)]}{\sigma_x \sigma_y} \quad (10)$$

Where:

- $\rho$  is Pearson's correlation coefficient between x and y
- $E$  is the expectation
- $\mu_x$  is the mean of x
- $\mu_y$  is the mean of y
- $\sigma_x$  is the standard deviation of x

- $\sigma_y$  is the standard deviation of  $y$

Features with a correlation coefficient of higher than 0.7 can be considered highly correlated [Calkins, 2005]. Therefore, 0.7 is the threshold that is set to remove correlated features.

## 12.2 Low variance features

This is a simple feature selection method, implemented in TPOT, where it removes features when their variance does not meet a certain threshold. This will get rid of the features that have the same value for almost every instance. The variance is given by Equation 11.

$$\text{Var}(X) = p(1 - p) \quad (11)$$

Where  $p$  is the probability, so when one wants to delete all features that have the same value in 90% of all instances then  $p$  should be set to 0.9 [Scikit-Learn, ndd].

## 12.3 Select k best features

This method selects the  $k$  best features. The  $k$  best features are determined by some function. This could be a regression or classification function. The eventual  $p$ -values per feature will rank the importance of the features. Then the  $k$  top-ranked features are selected. Both TPOT and Auto-sklearn use this method in their framework [Scikit-Learn, ndd].

## 12.4 Select top n percentile features

This is similar to the select  $k$  best features method. Instead of selecting  $k$  number of features, it selects a specified highest scoring percentage of features. The scores are equivalent to select  $k$  best features, determined by either a regression or classification function. This method is also used in both automated modeling approaches [Scikit-Learn, ndd].

## 12.5 Recursive feature elimination

With recursive feature elimination (RFE) the eventual set of features that are selected are obtained by recursively trying smaller sets of features. This method is used in TPOT and Auto-sklearn. Firstly, the initial set of features is used for training. Then, the importance of the features is obtained through an external

estimator which assigns weights to each of the features, for example a linear model (e.g. Support Vector Machine) or the coefficients. Next, the least important features are removed from the set of features. This process is recursively repeated on the current set of features until the desired number of features remain [Scikit-Learn, ndd].

## 12.6 Classification-based feature selection with Extremely Randomized Trees

With this approach, used in Auto-sklearn, feature selection is done based on the classifier Extremely Randomized Trees. Using this classification method the feature is scored and the best scoring ones are selected to create the model. Recursive feature elimination is also classification-based as this uses a linear model to assign the weights. How this classifier works in general is explained in Chapter 14. For selecting the features, Extremely Randomized Trees uses the Gini index for scoring, this score can also be named the Gini Importance of the feature. From this scoring metric the top k features are selected.

## 12.7 Prototype

For the created prototype the highly correlated features will be deleted before using the automated modeling approach. This will often reduce the number of features, which is favorable because so many features could be created during deep feature synthesis. Passing fewer features to the automated modeling approach reduces training time. Also having too many correlated features could possibly negatively influence the model. With the automated modeling approaches TPOT and Auto-sklearn the other discussed feature selection methods are used, depending on the created pipeline.

# 13 Dimensionality reduction

In Chapter 6, several dimensionality reduction methods are discussed, all implemented in either TPOT or Auto-sklearn. In this chapter, the mentioned methods are briefly explained. Those are the following methods:

- Principal Component Analysis (PCA)
- Truncated SVD
- Kernel PCA
- Independent component analysis (ICA)

## 13.1 Principal Component Analysis (PCA)

PCA is performing a linear dimensionality reduction. This is done using approximated Singular Value Decomposition (SVD) of the data, it then keeps only the most significant singular vectors in order to project the data to a lower-dimensional space [Scikit-Learn, ndg].

## 13.2 Truncated SVD

This method is similar to PCA performing a linear dimensionality reduction. Contrary to PCA, it performs the dimensionality reduction using truncated SVD. The data is not centered before computing the SVD by this estimator. Therefore, truncated SVD can work with sparse matrices [Scikit-Learn, nde].

## 13.3 Kernel PCA

Kernel PCA also performs principal component analysis, it is an extension of regular PCA. But with kernel PCA it is performing PCA in a reproducing kernel Hilbert space, this allows for non-linear mapping [Feurer et al., 2015b].

## 13.4 Independent component analysis (ICA)

ICA finds basis vectors so that the original data projected on these basis vectors are maximally statistically independent. It can be seen as an extension of PCA. Classically, ICA is used to separate mixed signals, this problem is known as the "blind source separation" problem [Jutten and Comon, 2010]. Also, ICA can be used as a linear decomposition method to find components [Scikit-Learn, ndc].

## 13.5 Prototype

All of these methods are implemented in either TPOT or Auto-sklearn. From these methods, PCA is implemented in both automated modeling approaches. Auto-sklearn, on the other hand, has also implemented the other three discussed approaches, Truncated SVD, Kernel PCA and ICA. So, all of these methods are used in the prototype due to both approaches are used for comparison.

# 14 Classification and Regression Algorithms

From the automated modeling approaches discussed in chapter 10 many different methods were mentioned. In this chapter, each of those methods will briefly be explained. This concerns the following classification and regression algorithms:

- Gaussian Naive Bayes
- Bernoulli Naive Bayes
- Multinomial Naive Bayes
- Decision Tree
- Random Forest
- Extremely Randomized Trees
- Gradient Boosting
- Extreme Gradient Boosting (XGBoost)
- Adaptive Boosting (AdaBoost)
- Linear Support Vector Machine (SVM)
- Kernel Support Vector Machine (SVM)
- Logistic Regression
- Linear Classification via Online Stochastic Gradient Descent (SGD)
- Maximum Margin classification via Online Passive Aggressive algorithms
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- k-Nearest Neighbors (kNN)
- Linear regression

- Ridge regression
- Lasso regression

## 14.1 Naive Bayes

The Naive Bayes classifier is a simple probabilistic classifier used in machine learning, based on Bayes theorem (Equation 12). Bayes theorem is used to find the probability that A occurs given that B has already happened. Bayes theorem makes the assumption that the predictor, or in this case the features, are independent. That is the reason it is called "naive".

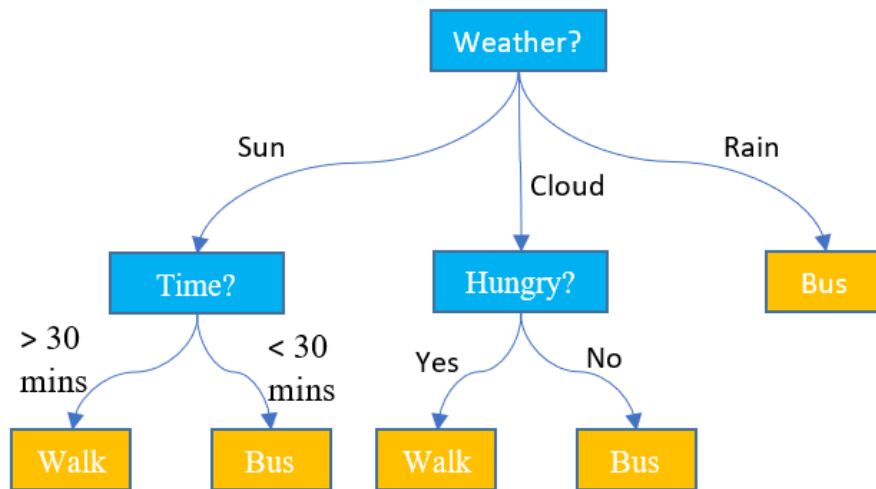
$$P(A|B) = \frac{P(B|A)PA}{P(B)} \quad (12)$$

As mentioned earlier, there are three variants of Naive Bayes implemented in Auto-sklearn. The difference between those variants is about how the data appears. Bernoulli Naive Bayes is used for binary data. Multinomial Naive Bayes is used for discrete data. Gaussian Naive Bayes is used for continuous data [Gandhi, 2018a].

## 14.2 Decision Tree

A decision tree uses a tree-shaped model to make decisions with the possible consequences. This includes the probability of an outcome. A decision tree starts in a single node. This node branches into different possible outcomes. Then again those nodes lead to other additional nodes. This goes on, which forms a treelike shape. A decision tree contains only conditional statements at each node, where the decision is based on. A decision tree can be used for regression and classification problems. Figure 8 shows an example of a decision tree [Brid, 2018].





**Figure 8:** Example of a decision tree, where it is decided whether someone should take the bus or go walking. This is decided by first looking at the weather, then the time that is left and whether he or she is hungry so that he or she can walk by a café for some food.

### 14.3 Random forest and Extremely Randomized Trees

Random forest and extremely randomized trees are methods that use an ensemble of multiple individual decision trees. Both of these algorithms can also be used for regression and classification purposes. Both methods are trained with the method bagging. Bagging stands for Bootstrap Aggregation, which aims to decrease the variance. The idea of bagging is that the combination of those multiple models, in this case the decision trees, will together increase the result of the model. The difference between the two methods exists in the splitting rule. With random forest, the split is determined by the information gain or Gini impurity, which are metrics for determining the best. For extremely randomized trees the splitting rules are randomly generated, then the best one gets selected [Yiu, 2019] [Feurer et al., 2015b].

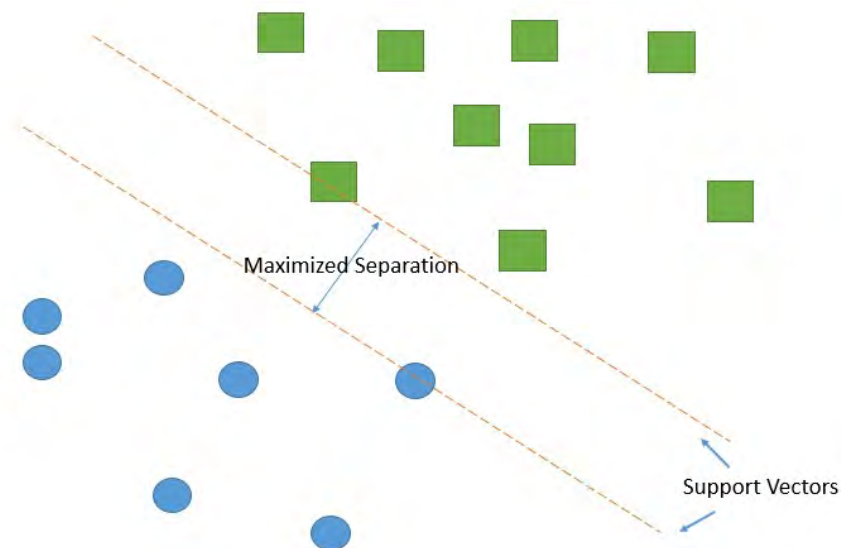
### 14.4 Gradient Boosting

Similar to Random Forest, Gradient Boosting is an ensemble of decision trees. This can also be used for either regression or classification. Instead of Bagging, now boosting is used. Boosting is the strategy of combining multiple simple models into one. These simple models are referred to as "weak learners", which are boosted to

create a strong model. Gradient boosting uses gradient descent to minimize the loss function of the model. Gradient descent is an optimization algorithm for finding a local minimum of a function which can be differentiated. AdaBoost and XGBoost are a variant on gradient boosting. The popularity of these methods has grown due to their good performance in data competitions, such as Kaggle [Mujtaba, 2020].

## 14.5 Support Vector Machines (SVM)

The support vector machine algorithm aims to find the optimal hyperplane in a dimensional space of size  $n$  so that it can classify the different classes. It can handle both classification and regression problems. The optimal hyperplane is the one with the largest margin (or distance) between the data points of the distinctive classes (Figure 9) [Gandhi, 2018b].



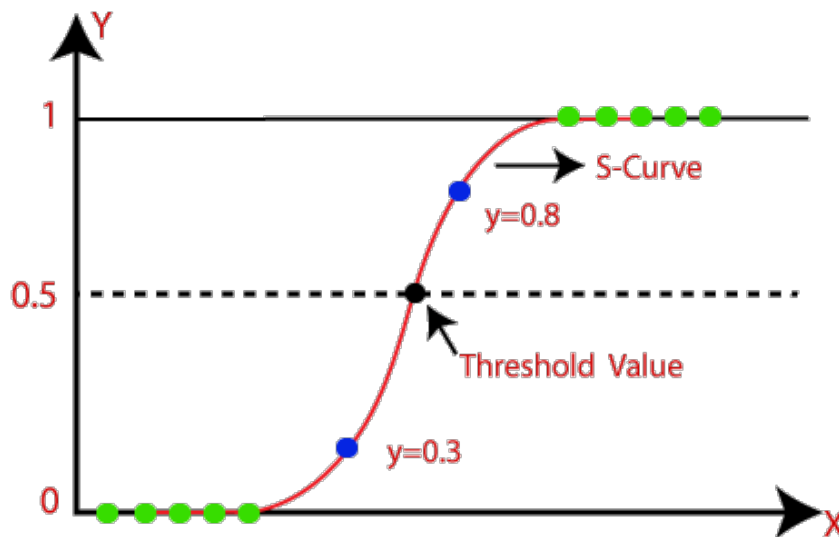
**Figure 9:** Example of the support vector machine algorithm in two dimensions. The two classes, indicated as squares and circles, are separated with the two support vectors creating the maximum margin possible.

In the classifying algorithms listed above, linear SVM and kernel SVM are mentioned. The difference is that linear SVM uses linear support vectors and kernel SVM uses the kernel trick which allows for non-linear classification.

## 14.6 Logistic regression

The logistic regression classifier is a method used in TPOT. Logistic regression is similar to linear regression, but linear regression solves regression problems and

logistic regression solves classification problems. This method is especially useful for binary classification problems. Logistic regression aims to maximize the likelihood function. An example is given in Figure 10.



**Figure 10:** Example of a logistic regression in machine learning. This is a binary classification problem where everything below the threshold gets classified as 0 and above the threshold gets classified as 1. The curve is called a sigmoid function.

## 14.7 Generalized Linear Models (GLM)

Generalized Linear Models can be used for either regression or classification purposes. Under GLM fall the two methods listed above used in Auto-sklearn, namely linear classification via online stochastic gradient descent (SGD) and maximum margin classification via online passive aggressive algorithms. These are both linear classification algorithms. In Auto-sklearn, they use only online learning algorithms from the GLM category since they were interested in scaling their automated machine learning framework to large datasets. With online algorithms, the input is processed piece by piece instead of that the entire input is available from the beginning.

Linear classification via online stochastic gradient descent (SGD) uses some loss function which is either a hinge, negative log likelihood or a Huber loss. Hinge is a soft-margin linear SVM (higher tolerance for misclassification), Huber loss a smoothed version of the hinge and negative log likelihood results in logistic regression. Maximum margin classification via online passive aggressive algorithms solves constrained optimization problems iteratively in order to update the weights of the model, which guarantees small steps and retains a larger margin [Feurer et al., 2015b].

## 14.8 Discriminant Analysis

Discriminant Analysis is used for classification and quite similar to logistic regression. In Auto-sklearn they implemented two variants from the discriminant analysis family. These are the two classifiers: linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). LDA uses a linear decision surface and assumes that the different classes have an identical covariance. QDA uses a quadratic decision surface, which makes it more flexible than LDA, and has the assumption that the values of all features are normally distributed [Scikit-Learn, nda].

## 14.9 K-Nearest Neighbors (KNN)

This algorithm was already explained in Chapter 7 as a missing value imputation method. KNN can be used for a classification or regression problem. It finds the  $k$  nearest training examples within the given feature space, where  $k$  is specified by the user.

## 14.10 Classical linear regressors

The three classical linear regressors used in both TPOT and Auto-sklearn are Linear regression, Ridge regression and Lasso regression. Linear regression fits a straight line between the data points which minimizes the residual sum of squares. The formula is given by Equation 13.

$$Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i \quad (13)$$

Ridge regression is a linear least squares regression with l2 regularization. Lasso regression is a linear least squares regression with l1 regularization. L2 avoids overfitting issues and l1 works better with a huge amount of features because it shrinks the less important features. The difference between these methods is that the ridge and lasso regression add a penalty term to the loss function of the ordinary least squares (OLS) method (Equation 14). Ridge regression adds the summation of the squared weights to the OLS loss function with some factor lambda as a penalty term (Equation 15). Lasso regression adds the summation of the absolute values of the coefficients to the OLS loss function with some factor lambda as a penalty term (Equation 16) [Nagpal, 2017].

$$OLS_{loss} = \sum_{i=1}^n (y_i - \lambda \sum_{j=1}^p x_{ij} \beta_j)^2 \quad (14)$$

$$Ridge_{loss} = \sum_{i=1}^n (y_i - \lambda \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (15)$$

$$Lasso_{loss} = \sum_{i=1}^n (y_i - \lambda \sum_{j=1}^p x_{ij} \beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (16)$$

# 15 Experimental setup

To test the prototype's performance, some experiments will be executed. For that multiple datasets are used which are discussed in Chapter 5. In this chapter, the setup of these experiments is discussed and also how the performance of the prototype is evaluated.

## 15.1 Training/validation/test split

Each dataset is split into a training set (67%) and a validation set (33%). In the Python notebook, a random state is added which sets a seed for the random generator, causing the dataset to make the training validation split in the same manner every time. This is done in order to compare TPOT to Auto-sklearn in a fair manner. When the dataset is from a data competition, then a separate testing set is provided. That testing set can be used to evaluate the model in the data competition where the score can be compared to other competitors.



Figure 11: train/validation/test split

## 15.2 Modeling

For modeling both the automated modeling approaches, TPOT and Auto-sklearn, are used. Both are used in order to investigate whether one framework performs better than the other.

For TPOT the number of generations and population size should be set. This is set to 5 generations with a population size of 50. This is lower than the default

setting, but this causes the algorithm not to have extremely large training times for the larger datasets. For Auto-sklearn no such additional information has to be provided. We can however provide the scoring function for both methods. The default scoring function is accuracy, this can be changed to for example f1-score or precision. This will be an option for the user of the tool to define the scoring function. This is especially useful for highly unbalanced datasets where the accuracy is a rather irrelevant metric. For regression problems, the mean squared error is used as a scoring function.

### 15.3 Evaluation metrics

Several evaluation metrics are used to evaluate the performance of the constructed model. These evaluation metrics are based solely on the training and validation data. The following evaluation metrics are used for classification problems:

- Confusion matrix
- Accuracy
- Precision
- Recall
- F1-score
- Area Under the Receiver Operating Characteristic Curve (AUROC)
- Online competition submission score

To evaluate regression models, other evaluation metrics are used. The three metrics used are [Karbhari, 2018]:

- Mean absolute error (MAE)
- Mean squared error (MSE)
- R2 score

These evaluation metrics, combined with a online data competition submission score (if possible), will provide a good insight on how the augmented data science tool performed.

### 15.3.1 Confusion matrix

The output of a machine learning model for a classifying problem can be represented in a confusion matrix (Figure 12). This metric can visualize the performance of the machine learning model.

	Predicted: No	Predicted: Yes
Actual: No	True Negative (TN)	False Positive (FP)
Actual: Yes	False Negative (FN)	True Positive (TP)

**Figure 12:** Confusion matrix

The example in Figure 12 is a 2x2 confusion matrix which is for a binary classification problem. In this case, the output can be either "Yes" or "No". Within this confusion matrix there are the following terms:

- True Positives (TP): These are cases that are correctly predicted "Yes".
- True Negatives (TN): These are cases that are correctly predicted "No".
- False Positives (FP): These are cases that are wrongly predicted "Yes" because they are actually "No".
- False Negatives (FN): These are cases that are wrongly predicted "No" because they are actually "Yes".

These terms mentioned above can be used to determine the accuracy, precision and recall [Markham, 2014].

### 15.3.2 Accuracy

The accuracy is the fraction of the correctly predicted cases out of all cases (Equation 17). Accuracy is an important metric, but in some cases it is not. In an unbalanced dataset, where for example 90% of all cases are classified true, a model that classifies everything as false obtains a high accuracy of 90%, but is however a bad model. Therefore, including other metrics is important.

$$Accuracy = \frac{TP + TN}{Total} \quad (17)$$



### 15.3.3 Precision

The precision is the fraction of the correctly predicted cases from all the cases that are predicted true (Equation 18). So, when the cases are predicted as true, how often is this correct [Markham, 2014].

$$Precision = \frac{TP}{TP + FP} \quad (18)$$

### 15.3.4 Recall

The recall, also known as sensitivity, is the fraction of the correctly predicted cases from the cases that are actually true. So, when the cases are actually true, how often is this correctly predicted.

$$Recall = \frac{TP}{TP + FN} \quad (19)$$

### 15.3.5 F1 score

The F1 score, or F1 measure, is a metric that seeks the balance between precision and recall. The F1 score is the harmonic mean of those two metrics (Equation 20) [Markham, 2014].

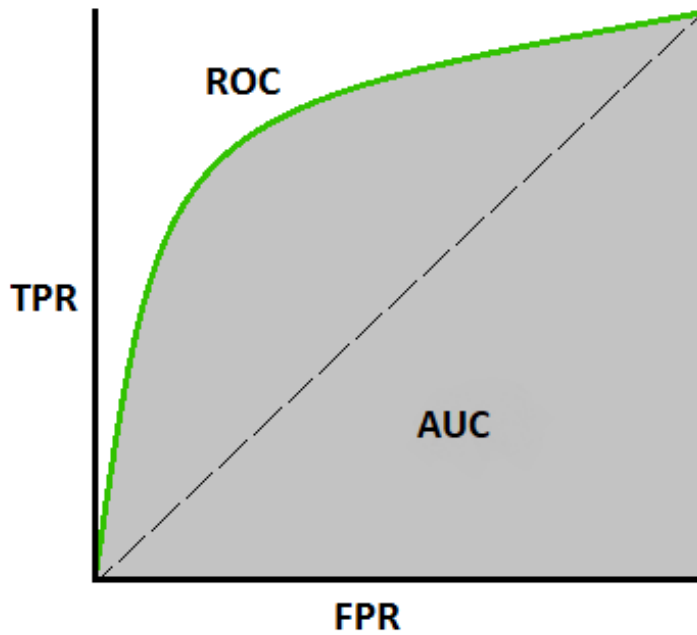
$$F1score = 2 * \frac{precision * recall}{precision + recall} \quad (20)$$

### 15.3.6 Area Under the Receiver Operating Characteristic Curve (AUROC)

This is an important metric for checking a classification model's performance. It consists of the AUC (Area Under The Curve) and ROC (Receiver Operating Characteristics) curve. Combined it can also be written as AUROC. This metric explains how much the model is capable of distinguishing between different classes. A perfect model has an AUC of 1, when the AUC is 0.5 then the model has no class separation whatsoever. The ROC curve is the True Positive Rate (TPR) (Equation 21) plotted against the False Positive Rate (FPR) (Equation 22) where the TPR is actually the recall. In Figure 13 an example is given of the ROC curve where is indicated what the AUC is. [Narkhede, 2018].

$$TPR = \frac{TP}{TP + FN} \quad (21)$$

$$FPR = \frac{TN}{TN + FP} \quad (22)$$



**Figure 13:** Example of ROC curve and AUC

### 15.3.7 Mean absolute error (MAE)

The mean absolute error calculates the sum of the absolute difference between the predicted and the actual values (Equation 23).

$$MAE = \frac{1}{n} \sum_{t=1}^n |actual_t - predicted_t| \quad (23)$$

### 15.3.8 Mean squared error (MSE)

The MSE is similar to MAE, except instead of the absolute difference the squared difference is taken (Equation 24) [Karbhari, 2018].

$$MSE = \frac{1}{n} \sum_{t=1}^n (actual_t - predicted_t)^2 \quad (24)$$

### 15.3.9 R2 score

Perhaps the most important score for evaluating a regression model is the R2 score. This score indicates how well the regression fits the data. For the R2 score, 1.0 is perfect and 0 is the worst. The formula for the R2 score is shown in Equation 25.

$$R2 = 1 - \frac{SS_{Res}}{SS_{total}} \quad (25)$$

Where  $SS_{Res}$  is the sum of squares of residuals and  $SS_{total}$  is the total sum of squares.

### **15.3.10 Online data competition submission score**

When the data is from a data competition, then a prediction can be submitted. This scored submission can be compared to other competitors that have made a submission on the data competition platform. This is a good way to evaluate the model in comparison to others. The metric used to score the submission varies per competition. This can for example be the accuracy or AUROC.

# 16 Results

In this chapter, the results obtained from the created prototype are discussed. The augmented data science tool is used with the different datasets discussed in Chapter 5. The models are evaluated by observing the evaluation metrics discussed in Chapter 15. Two versions of the prototype are used, one with the TPOT framework and one with Auto-sklearn. This is done in order to make a comparison between the two frameworks, so it can be determined which framework suits such a solution better. The final selected classifier or regressor that is used on the dataset, constructed by the automated modeling approach, will also be mentioned.

## 16.1 Titanic

The titanic dataset was the first dataset processed through the tool. After the pre-processing steps were done, both TPOT and Auto-sklearn were used for modeling. The accuracy is used as a scoring function. The best found classifier by TPOT is a Decision Tree Classifier. Auto-sklearn constructed an ensemble out of 21 different models. Most of these individual models, with the highest weights, are Random Forrest and Gradient Boosting classifiers. Table 1 shows the confusion matrices of these models and Table 2 shows the evaluation metrics.

**Table 1:** Confusion matrices Titanic dataset. TPOT (left) and Auto-sklearn (right).

Actual/Predicted	No	Yes	Actual/Predicted	No	Yes
No	162	13	No	163	12
Yes	44	76	Yes	39	81

**Table 2:** Evaluation metrics Titanic dataset

	TPOT	Auto-sklearn
Accuracy	0.807	0.827
Recall	0.780	0.803
Precision	0.820	0.839
F1-score	0.789	0.813
AUROC	0.780	0.803

From the confusion matrices, it can be observed that it predicts who survived quite good for both modeling approaches. Most of the cases are correctly predicted. This is also supported by the values obtained from the evaluation metrics. It has a high accuracy of 80% and 82%. Also, the other metrics are around that value, thus overall it seems a sufficient model. Auto-sklearn does perform slightly better, approximately 2% better on all metrics.

This was an open Kaggle competition. Kaggle used the accuracy as a scoring metric in this competition. The prediction was uploaded to Kaggle resulting in a score of 0.785 (TPOT) and 0.790 (Auto-sklearn). So, quite similar scores but slightly better for Auto-sklearn. The score achieved by Auto-sklearn is approximately the top 22% on the leaderboard, which is pretty good for the little effort that goes into running it through the tool.

## 16.2 Mobile price classification

The second dataset was the mobile price classification. This was also a small dataset, just like the Titanic data. As a scoring function, the accuracy is used on this dataset. Using the tool on this data, resulted in Linear Support Vector Classifier (Linear SVC) for TPOT. Auto-sklearn constructed an ensemble out of 28 models. Similar to TPOT, from those models all are Linear Support Vector Machine classifiers. So according to both approaches, this algorithm seems to be the best. The confusion matrices are shown in Table 3 and Table 4 shows the evaluation metrics.

**Table 3:** Confusion matrices mobile phone price classification. TPOT (on top) and Auto-sklearn (below).

Actual/Predicted	1	2	3	4
1	165	4	0	0
2	0	155	7	0
3	0	11	143	8
4	0	0	9	158
Actual/Predicted	1	2	3	4
1	165	4	0	0
2	1	160	1	0
3	0	3	153	6
4	0	0	2	165

**Table 4:** Evaluation metrics mobile phone price classification

	TPOT	Auto-sklearn
Accuracy	0.941	0.974
Recall	0.940	0.974
Precision	0.941	0.974
F1-score	0.940	0.974
AUROC	0.957	0.973

By observing the confusion matrices, it is clear that the tool performed very well on the validation data with both TPOT and Auto-sklearn. Almost all cases are predicted correctly. This translates to very high accuracy, recall, precision and F1-score of around 0.94 and 0.97 for all metrics. Both models also have a very high AUROC, which indicates the goodness of the model. Auto-sklearn does outperform TPOT again. This time with approximately 3%, which is quite a lot considering how close to 100% it is. There was no Kaggle competition for this dataset, so no comparison to other competitors could be made.

### 16.3 Australia rain prediction

This dataset was large compared to the already discussed datasets. This was also a slightly unbalanced dataset. TPOT uses a Gradient boosting classifier in its best pipeline. Auto-sklearn created an ensemble out of eight different models where the model with the highest weight is a Random Forest classifier with 32 % followed by a Linear SVM with 28%. Table 5 shows the confusion matrices of these models and Table 6 shows the evaluation metrics.

**Table 5:** Confusion matrices Australia rain dataset. TPOT (left) and Auto-sklearn (right).

Actual/Predicted	No	Yes
No	34616	1850
Yes	4997	5461

Actual/Predicted	No	Yes
No	34400	2066
Yes	4781	5677

**Table 6:** Evaluation metrics Australia rain dataset

	TPOT	Auto-sklearn
Accuracy	0.854	0.854
Recall	0.736	0.743
Precision	0.810	0.806
F1-score	0.762	0.767
AUROC	0.736	0.743

The augmented data discovery solution worked reasonably well on this dataset. TPOT and Auto-sklearn performed very similarly, there is not much between the two. Both models that are created are quite decent in predicting the rain for tomorrow. Auto-sklearn is a little better in predicting the "Yes" cases, in which we are interested. Therefore Auto-sklearn seems to be a slightly better overall model, which also can be seen in the 1% higher AUROC metric. This was not a Kaggle competition, thus no comparison to others could be made.

## 16.4 Random Acts of Pizza

With this relatively small dataset, the data was slightly unbalanced where three-quarters of the target variable is classified as "False". So, it is interesting to see how this is handled by the augmented data science solution. As a scoring function, the f1-score is used, because of the slightly unbalanced dataset. TPOT fits a Random Forest classifier as the best model on the data. Auto-sklearn constructed an ensemble of six models. The highest weighted model, making up 84% of the total ensemble, is a linear classification via online stochastic gradient descent classifier. The confusion matrices are shown in Table 7 and Table 8 shows the evaluation metrics.

**Table 7:** Confusion matrices acts of pizza dataset. TPOT (left) and Auto-sklearn (right).

Actual/Predicted	False	True
False	987	16
True	210	121

Actual/Predicted	False	True
False	915	88
True	172	159

**Table 8:** Evaluation metrics acts of pizza dataset

	TPOT	Auto-sklearn
Accuracy	0.831	0.805
Recall	0.675	0.696
Precision	0.854	0.743
F1-score	0.707	0.713
AUROC	0.675	0.696

This dataset is processed fairly well by the augmented data science solution. Both TPOT and Auto-sklearn have high accuracy which is expected due to the dataset being rather unbalanced. Therefore, accuracy is not a very important metric in assessing the models. Observing the other metrics, Auto-sklearn seems to have constructed a slightly better model than TPOT did, in which it is capable of classifying the "True" cases correctly more often. The AUROC is the metric that is used in the Kaggle competition for scoring. TPOT achieve a Kaggle score of 0.651, approximately the top 32% of the competition. Whereas Auto-sklearn achieves a score of 0.664, which is in the top 27% of the competition. These scores are pretty decent. Thus, for this dataset Auto-sklearn again performs better than TPOT.

## 16.5 Pump it up

This was a decent sized dataset and also a multiclass problem. During the automated process, TPOT fits a Random forest classifier on the dataset. Auto-sklearn created an ensemble of size six, existing predominately out of random forest classifiers. For this dataset the accuracy was used as a scoring function. In Table 9 the confusion matrices of these models are shown and Table 10 shows the evaluation metrics.

**Table 9:** Confusion matrices pump it up dataset. TPOT (on top) and Auto-sklearn (below).

Actual/Predicted	Functional	Functional needs repair	Non functional
Functional	9592	226	901
Functional needs repair	754	463	208
Non functional	1627	91	5740
Actual/Predicted	Functional	Functional needs repair	Non functional
Functional	9373	380	966
Functional needs repair	709	515	201
Non functional	1597	161	5700



**Table 10:** Evaluation metrics pump it up dataset

	TPOT	Auto-sklearn
Accuracy	0.806	0.795
Recall	0.663	0.667
Precision	0.744	0.707
F1-score	0.689	0.683
AUROC	0.773	0.765

From the confusion matrices, it can be observed that the tool performed well in classifying the different water pumps. Both automated modeling approaches have very similar scores when observing the evaluation metrics. TPOT seems to be slightly better, but this is not a major difference. The competition score on driven data was 0.795 for TPOT and 0.799 for Auto-sklearn, the accuracy was used as a scoring metric for this competition. This was approximately the top 21% of the competition, so quite good. For this dataset, it could not be determined which modeling approach performs better.

## 16.6 Richter’s Predictor: Modeling Earthquake Damage

Also, this was a multiclass problem and quite a large dataset. As a scoring function, the accuracy was used. When processing this dataset through the augmented data science solution, TPOT used a Random Forest classifier on this problem. Auto-sklearn used an ensemble out of six models, in which it predominately consists out of Adaboosting models. The confusion matrices are shown in Table 11 and Table 12 shows the evaluation metrics.

**Table 11:** Confusion matrices earthquake dataset. TPOT (on top) and Auto-sklearn (below).

Actual/Predicted	Low damage	Medium damage	Complete destruction
Low damage	3665	4552	78
Medium damage	1438	42059	5430
Complete destruction	107	11195	17475
Actual/Predicted	Low damage	Medium damage	Complete destruction
Low damage	4105	4099	91
Medium damage	1653	41647	5627
Complete destruction	139	10893	17745

**Table 12:** Evaluation metrics earthquake dataset

	TPOT	Auto-sklearn
Accuracy	0.735	0.738
Recall	0.636	0.654
Precision	0.730	0.729
F1-score	0.669	0.682
AUROC	0.728	0.739

When looking at the confusion matrices and evaluation metrics, it appears that the tool performed decently. TPOT and Auto-sklearn perform relatively similar on this problem. On the driven data competition the F1 score was used as a scoring metric. TPOT achieved a score of 0.734 and Auto-sklearn achieved a score of 0.735, here the difference is hardly noticeable. This was approximately the top 11% of the competition, which is really good. Thus, it seems that although the metrics are not very high it still is a very good model compared to other competitors.

## 16.7 Insurance prediction

The insurance prediction competition considers quite a small dataset. This data problem is a binary classification problem. This is a somewhat unbalanced dataset. Therefore, the AUROC is used as a scoring function instead of the accuracy. TPOT fitted a linear support vector machine to the data. Auto-sklearn constructed an ensemble out of four different models, where the two most relevant models (54% and 42%) were both a linear classification via online stochastic gradient descent classifier. The confusion matrices are shown in Table 13 and Table 14 shows the evaluation metrics.

**Table 13:** Confusion matrices insurance prediction dataset. TPOT (left) and Auto-sklearn (right).

Actual/Predicted	False	True
False	1648	172
True	352	191

Actual/Predicted	False	True
False	1507	313
True	285	258

**Table 14:** Evaluation metrics insurance prediction dataset

	TPOT	Auto-sklearn
Accuracy	0.778	0.747
Recall	0.629	0.652
Precision	0.675	0.646
F1-score	0.642	0.649
AUROC	0.629	0.652

The augmented data science tool performed poorly on this dataset. It has quite a high accuracy, but the other metrics are a fair amount lower. It seems that the tool had difficulties classifying whether there will be a claim or not. The competition scores on Zindi were 0.601 for TPOT and 0.624 for Auto-sklearn. This score is based on the AUROC. Auto-sklearn did perform slightly better than TPOT. This score ended up in approximately the top 50% on the leaderboard of the competition. This is not very impressive compared to the previous results with the other datasets, but also not extremely bad.

## 16.8 House Prices: Advanced Regression Techniques

This was the first dataset that was used to test the automated regression method. After pre-processing was completed, the remaining dataset was processed by TPOT and Auto-sklearn. TPOT fitted a Gradient Boosting Regressor to the data. Auto-sklearn created an ensemble out of two models, which were both ridge regressions. The evaluation metrics are shown in Table 15.

**Table 15:** Evaluation metrics House price prediction

	TPOT	Auto-sklearn
Mean absolute error	16415.804	17196.212
Mean squared error	907966311.828	1013412307.341
R2 score	0.876	0.862

The evaluation metrics seem pretty good. Both approaches have a good R2 score. The mean absolute error indicates that the approaches are approximately 17.000 off the actual price on average. This was a Kaggle competition where the submission is scored on the Root-Mean-Squared-Error of the logarithmic prices instead of the actual prices. This resulted in a Kaggle score of 0.138 for TPOT and 0.150 for Auto-sklearn. TPOT ended up in the top 43% of the competition and Auto-sklearn ended up in the top 56%. This is not too bad, but it is worse compared to the

classification tasks. Remarkably, TPOT outperforms Auto-sklearn quite clearly when looking at the Kaggle leaderboard.

## 16.9 Restaurant Revenue Prediction

With this dataset, there was not a lot of data to train on. After processing through the augmented data science tool both automated modeling approaches were used to create a regression model. TPOT fitted a random forest regressor. Auto-sklearn fitted an ensemble out of four models, in which the most relevant model (90%) was a support vector machine regressor. So again both a different regressor. The evaluation metrics are shown in Table 16.

**Table 16:** Evaluation metrics restaurant dataset

	TPOT	Auto-sklearn
Mean absolute error	1018011.955	1073760.025
Mean squared error	4015351467572.691	4753409570059.237
R2 score	0.525	0.438

The training set was really small, which became smaller after the training/validation split, so these metrics are not that trustworthy. Nevertheless, the values are not great, there are large errors and a bad R2 score. On Kaggle the root mean squared error was used as a metric. TPOT achieved a score of 1908153.655, which was the top 67% of the competition. Auto-sklearn achieved a score of 2262881.226, which was the bottom 10% of the competition. So these results are not very good, in which especially Auto-sklearn constructed a very bad regression model. Again, pretty disappointing results for a regression problem.

## 16.10 DengAI: Predicting Disease Spread

This data competition concerns a small training set. After pre-processing was done, TPOT and Auto-sklearn were used for modeling. TPOT fits a k-nearest neighbors regressor as the best model on the data. Auto-sklearn constructed an ensemble of nine models. The two highest weighted models are a k-nearest neighbors regressor (32%) and gradient boosting regressor (22%). The evaluation metrics of this training data is presented in Table 17.

**Table 17:** Evaluation metrics disease spread prediction

	TPOT	Auto-sklearn
Mean absolute error	8.655	9.797
Mean squared error	354.374	453.249
R2 score	0.849	0.806

The evaluation metrics look quite good. This is however a small training set, so these results could be unreliable. TPOT seems to have the upper hand over Auto-sklearn. On driven data, the mean absolute error was used as a scoring metric. TPOT achieved a competition score of 30.175 and Auto-sklearn a score of 26.593. These scores are quite a bit higher than the MAE obtained from the evaluation metrics on the validation set, which could be a result of the small training set. Auto-sklearn ended up in the top 22% of the competition, which is good. Auto-sklearn outperformed TPOT on this dataset quite clearly.

## 16.11 Bike Sharing Demand

This relatively small dataset was processed by the prototype in order to predict the bike-sharing demand. In this process, both TPOT and Auto-sklearn are used for modeling. TPOT constructed a gradient boosting regressor for this dataset. Auto-sklearn constructed an ensemble of size four, where the highest weighted regressor was a ridge regression with 60%. Table 18 shows the evaluation metrics of this regression problem.

**Table 18:** Evaluation metrics Bike Sharing Demand

	TPOT	Auto-sklearn
Mean absolute error	31.382	28.074
Mean squared error	2027.596	2307.454
R2 score	0.937	0.929

When looking at the evaluation metrics, both modeling approaches seem to have created quite a good model. The error for both modeling approaches is not that high considering the demand is on average approximately 200. Both R2 values are really high, indicating a good model. Both models were uploaded to Kaggle where the Root Mean Squared Logarithmic Error (RMSLE) was used for scoring. TPOT achieved a submission score of 0.436, which is really good and with that score ended up in the top 22% of the competition. Auto-sklearn scored 0.665 in the competition, which is a lot worse than TPOT scored. With this score, it only ended up in the top

73% of the competition. So, also with this regression problem, TPOT outperforms Auto-sklearn.

## 16.12 Urban Air Pollution Challenge

With this competition, the air pollution is predicted through a regression model. The data was processed through the prototype to create such a model. Both TPOT and Auto-sklearn are used for modeling after pre-processing was done. TPOT fitted an XGboost regressor to the data and Auto-sklearn constructed an ensemble of three models: a ridge regression, a K-nearest neighbors regressor and an AdaBoost regressor. Table 19 shows the evaluation metrics of this regression problem.

**Table 19:** Evaluation metrics urban air pollution challenge

	TPOT	Auto-sklearn
Mean absolute error	20.017	26.051
Mean squared error	851.600	1288.641
R2 score	0.603	0.400

The evaluation metrics do not look very impressive. Especially Auto-sklearn appears to have constructed a bad predictive model with only an R2 value of 0.4. TPOT did a lot better with an R2 value of 0.6, but that is still not very high. Both solutions were submitted on the Zindi data competition, where the root mean squared error is used for scoring. TPOT achieve a score of 34.053 and Auto-sklearn achieved a score of 39.544. At the time of submission, only 13 competitors were on the leaderboard, so the comparison is rather difficult to make. Auto-sklearn ended on place 10 out of 13 with that score, which is not good. TPOT achieved fourth place out of 13, which is decent. Again TPOT outperforms Auto-sklearn on a regression problem.

## 16.13 Discussion

The results presented in this chapter show that an augmented data science solution can achieve good results for different classification and regression problems. For most classification problems it was able to distinguish between the different classes and to construct a good predictive model. In the data competitions it often performed fairly well against other competitors, it was regularly in the top quarter of the competition. This is quite good considering the effort that goes into processing a dataset through the tool. For the regression problems, the prototype was not working as well as it did for the classification problems. There were some decent results, but also some

disappointing ones.

One of the sub-questions was about the trade-off between efficiency and quality. From these results, it can be concluded, that the efficiency is very high as the effort that goes into processing a dataset through the prototype is minimal. On the other hand, the quality did not suffer too badly under this automatic approach, especially for classification problems where some valuable results were obtained. However, spending more time on the individual problems by getting a better understanding of them will most likely enable one to obtain better results at the cost of time.

When comparing the two frameworks, TPOT and Auto-sklearn, it is noticeable that Auto-sklearn did perform similar or better than TPOT with each of the classification tasks. All metrics were often higher and it also scored better in the data competitions. When looking at the regression problems, TPOT does often perform better than Auto-sklearn with one exception, which was the Disease Spread competition. On all other problems, TPOT performs better and this was often a lot better. It seemed that Auto-sklearn was regularly not capable of constructing a good regressor.

An advantage of Auto-sklearn compared to TPOT is the computation time. When processing large datasets, such as the earthquake dataset, the prototype with TPOT would run for approximately 15 hours. On the same dataset where Auto-sklearn was used, it only runs for a bit longer than an hour, which is significantly faster. Considering the enormous difference in running time and also achieving better results most of the time, Auto-sklearn is preferred over TPOT for classification problems. For regression problems, TPOT is preferred. TPOT has longer training times, but the results were much more consistent and often better than Auto-sklearn.

## 17 Conclusion

In this final chapter, the main conclusions of the conducted research are discussed. Also, the research question will be answered. Finally, possible future research for improving this work is addressed.

This research aimed to do research on the possibilities of augmented data discovery (or augmented data science). Thus, the automation of the data science process. Augmented data discovery is expected to play a more important role in the future. Avanade is interested in this topic as they expect this will play a more important role within their data mining process. For this thesis the following research question is defined:

*"How and to what extent can a data discovery process be automated with an augmented data discovery solution using Azure resources?"*

For this research, the augmented data discovery process is split into three parts, namely pre-processing, feature engineering and modeling. A great deal of research has been conducted in order to automate those three parts. In addition, a prototype is created to show whether an augmented data science solution is possible.

In the created prototype some relatively basic pre-processing steps are implemented. Handling missing values is done through a simple imputation method, namely imputing the median or the mode. Extreme values are handled using the interquartile range method. During the literature research, some helpful frameworks were found in automating some parts of the data science process. Those frameworks are implemented in Python. The framework Featuretools uses deep feature synthesis in order to create new features from the original features. This framework can, with little input from the user of the prototype, create a great number of new features that could be useful for modeling. Two similar frameworks TPOT and Auto-sklearn are both automated modeling approaches that can automatically choose a machine learning model for classification or regression and also tune its hyperparameters. In addition, TPOT and Auto-sklearn also handle some pre-processing steps, such as data transformations, decomposition and feature selection.



Based on the literature research and the created prototype, it can be concluded that an augmented data discovery solution is possible. The obtained results by testing the prototype showed that this automated solution was capable of achieving good results in the online data competitions. A comparison is made between the two frameworks Auto-sklearn and TPOT. From this comparison, it is concluded that from the two frameworks Auto-sklearn works better on classification problems and TPOT performs better on regression problems. Also, the prototype is able to run using Azure resources which is desired by Avanade. The major advantage of such an augmented data science solution is efficiency. With minimal effort, such a solution is able to obtain quite good results.

The time available for this project was not enough to develop a fully developed and operational tool. While the results have shown that an augmented data science solution is possible, still many improvements could be introduced in addition to the created prototype. The prototype supports classification and regression problems. In the future, it could be improved by also supporting clustering and time series forecasting problems. Also, the tool does not support image recognition tasks which could be a favorable feature. The prototype, as it is, does not use natural language processing on the data. Implementing an automated natural language processing pipeline enables the tool to handle different datasets that contain much textual data. Another limitation of the prototype is that it does not support multi-label predictions. It can now only predict one single feature. To make the prototype fully operational, an Graphical User Interface could be created that makes it easy to use.

Augmented data discovery will play a more important role in the near future. In a relatively short period of time, it was possible to create a working prototype that could perform some decent analysis. With the proposed improvements that are mentioned, together with the frameworks, such as TPOT, Auto-sklearn and Featuretools that are still being improved, such an augmented data science solution could be improved, achieve better results and be part of Avanade's data science process.

# Bibliography

- [Agarwal, nd] Agarwal, N. (n.d.). Augmented analytics: The future of data & analytics. <https://www.analyticsinsight.net/augmented-analytics-the-future-of-data-analytics/>. Accessed: 18-03-2020.
- [Avanade, nd] Avanade (n.d.). Avanade website. <https://www.avanade.com/>. Accessed: 2020-05-06.
- [Bilalli et al., 2016] Bilalli, B., Abelló, A., Aluja-Banet, T., and Wrembel, R. (2016). Automated data pre-processing via meta-learning. In Bellatreche, L., Pastor, Ó., Almendros Jiménez, J. M., and Aït-Ameur, Y., editors, *Model and Data Engineering*, pages 194–208, Cham. Springer International Publishing.
- [Brid, 2018] Brid, R. S. (2018). Decision tree - a simple way to visualize a decision. <https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb>. Accessed: 2020-07-01.
- [Brownlee, 2018] Brownlee, J. (2018). How to use statistics to identify outliers in data. <https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data/>. Accessed: 2020-04-15.
- [Calkins, 2005] Calkins, K. G. (2005). Correlation coefficients. <https://www.andrews.edu/~calkins/math/edrm611/edrm05.htm>. Accessed: 2020-05-22.
- [Caruana et al., 2004] Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the Twenty-First International Conference on Machine Learning, ICML '04*, page 18, New York, NY, USA. Association for Computing Machinery.
- [Dorpe, 2018] Dorpe, S. V. (2018). Preprocessing with sklearn: a complete and comprehensive guide. <https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9>. Accessed: 2020-06-15.

- [FeatureLabs, 2019] FeatureLabs (2019). What is featuretools? <https://docs.featuretools.com/en/stable/>. Accessed: 2020-04-23.
- [Feurer et al., 2015a] Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M., and Hutter, F. (2015a). Efficient and robust automated machine learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc.
- [Feurer et al., 2015b] Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M., and Hutter, F. (2015b). Supplementary material for efficient and robust automated machine learning. pages 1–13.
- [Fortin et al., 2012] Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.
- [Gandhi, 2018a] Gandhi, R. (2018a). Naive bayes classifier. <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c>. Accessed: 2020-06-30.
- [Gandhi, 2018b] Gandhi, R. (2018b). Support vector machine — introduction to machine learning algorithms. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Accessed: 2020-07-02.
- [García et al., 2015] García, S., Luengo, J., and Herrera, F. (2015). *Data Preprocessing in Data Mining*, volume 72. Springer.
- [Gartner, 2019] Gartner (2019). Gartner identifies top 10 data and analytics technology trends for 2019. <https://www.gartner.com/en/newsroom/press-releases/2019-02-18-gartner-identifies-top-10-data-and-analytics-technolo>. Accessed: 11-03-2020.
- [Github, 2018] Github (2018). Predict titanic survival rates. [https://github.com/ag2816/TitanicKaggle/blob/master/Titanic\\_FeatureTools.ipynb](https://github.com/ag2816/TitanicKaggle/blob/master/Titanic_FeatureTools.ipynb). Accessed: 2020-04-24.
- [Gordon, 2018] Gordon, A. (2018). Tool review: Lessons learned from using featuretools to simplify the process of feature engineering. <https://medium.com/dataexplorations/tool-review-can-featuretools-simplify-the-process-of-feature-engineering-5d165100b0c3>. Accessed: 2020-04-24.

- [Jutten and Comon, 2010] Jutten, C. and Comon, P. (2010). Chapter 1 - introduction. In Comon, P. and Jutten, C., editors, *Handbook of Blind Source Separation*, pages 1 – 22. Academic Press, Oxford.
- [Kanter and Veeramachaneni, 2015] Kanter, J. M. and Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10.
- [Kanter, 2018] Kanter, M. (2018). Deep feature synthesis: How automated feature engineering works. <https://innovation.alteryx.com/deep-feature-synthesis/>. Accessed: 2020-05-28.
- [Karbhari, 2018] Karbhari, V. (2018). How to evaluate regression models? <https://medium.com/acing-ai/how-to-evaluate-regression-models-d183b4f5853d>. Accessed: 2020-07-02.
- [Khurana et al., 2016] Khurana, U., Turaga, D., Samulowitz, H., and Parthasarathy, S. (2016). Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 1304–1307. IEEE.
- [Lam et al., 2017] Lam, H. T., Johann-Michael, Thiebaut, Sinn, M., Chen, B., Mai, T., and Alkan, O. (2017). One button machine for automating feature engineering in relational databases. *arXiv*, pages 1–9.
- [Lin, 2002] Lin, T. (2002). Attribute transformations for data mining I: Theoretical explorations. *International Journal of Intelligent Systems*, 17(2):213–222.
- [Lohr, 2014] Lohr, S. (2014). For big-data scientists, ‘janitor work’ is key hurdle to insights. *The New York Times*.
- [Maini, nd] Maini, E. (n.d.). Z score for outlier detection – python. <https://www.geeksforgeeks.org/z-score-for-outlier-detection-python/>. Accessed: 2020-07-17.
- [Markham, 2014] Markham, K. (2014). Simple guide to confusion matrix terminology. <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>. Accessed: 2020-06-09.
- [Microsoft, nda] Microsoft (n.d.a). Azure data factory. <https://docs.microsoft.com/en-us/azure/data-factory/introduction>. Accessed: 2020-07-16.

- [Microsoft, ndb] Microsoft (n.d.b). Azure data lake storage. <https://azure.microsoft.com/en-us/services/storage/data-lake-storage/>. Accessed: 2020-07-16.
- [Microsoft, ndc] Microsoft (n.d.c). What is azure. <https://azure.microsoft.com/en-nl/overview/what-is-azure/>. Accessed: 2020-07-16.
- [Microsoft, ndd] Microsoft (n.d.d). What is azure databricks. <https://docs.microsoft.com/en-us/azure/databricks/scenarios/what-is-azure-databricks>. Accessed: 2020-07-16.
- [MicroStrategy, nd] MicroStrategy (n.d.). Data discovery explained. <https://www.microstrategy.com/us/resources/introductory-guides/data-discovery-explained>. Accessed: 13-03-2020.
- [Mujtaba, 2020] Mujtaba, H. (2020). What is gradient boosting and how is it different from adaboost? <https://www.mygreatlearning.com/blog/gradient-boosting/>. Accessed: 2020-06-30.
- [Nagpal, 2017] Nagpal, A. (2017). L1 and l2 regularization methods. <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>. Accessed: 2020-07-02.
- [Narkhede, 2018] Narkhede, S. (2018). Understanding auc - roc curve. <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. Accessed: 2020-06-10.
- [Olson et al., 2016] Olson, R. S., Bartley, N., Urbanowicz, R. J., and Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 485–492, New York, NY, USA. Association for Computing Machinery.
- [Oracle, nd] Oracle (n.d.). Outlier detection methods. [https://docs.oracle.com/cd/E17236\\_01/epm.1112/cb\\_statistical/frameset.htm?ch07s02s10s01.html](https://docs.oracle.com/cd/E17236_01/epm.1112/cb_statistical/frameset.htm?ch07s02s10s01.html). Accessed: 2020-07-17.
- [Scikit-Learn, nda] Scikit-Learn (n.d.a). 1.2. linear and quadratic discriminant analysis. [https://scikit-learn.org/stable/modules/lda\\_qda.html](https://scikit-learn.org/stable/modules/lda_qda.html). Accessed: 2020-07-01.
- [Scikit-Learn, ndb] Scikit-Learn (n.d.b). 6.4. imputation of missing values. <https://scikit-learn.org/stable/modules/impute.html>. Accessed: 2020-06-04.

- [Scikit-Learn, ndc] Scikit-Learn (n.d.c). Decomposing signals in components (matrix factorization problems). <https://scikit-learn.org/stable/modules/decomposition.html#ica>. Accessed: 2020-06-29.
- [Scikit-Learn, ndd] Scikit-Learn (n.d.d). Feature selection. [https://scikit-learn.org/stable/modules/feature\\_selection.html#univariate-feature-selection](https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection). Accessed: 2020-06-18.
- [Scikit-Learn, nde] Scikit-Learn (n.d.e). `sklearn.decomposition.truncatedsvd`. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html#sklearn.decomposition.TruncatedSVD>. Accessed: 2020-06-29.
- [Scikit-Learn, ndf] Scikit-Learn (n.d.f). `sklearn.preprocessing.labelencoder`. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>. Accessed: 2020-06-04.
- [Scikit-Learn, ndg] Scikit-Learn (n.d.g). `sklearn.preprocessing.randomizedpca`. <http://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/modules/generated/sklearn.decomposition.RandomizedPCA.html>. Accessed: 2020-06-18.
- [Shaikh, 2018] Shaikh, R. (2018). Feature selection techniques in machine learning with python. <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>. Accessed: 2020-05-22.
- [Thornton et al., 2013] Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *KDD '13: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. Association for Computing Machinery.
- [Uzunalioglu et al., 2019] Uzunalioglu, H., Cao, J., Phadke, C., Lehmann, G., Akymac, A., He, R., Lee, J., and Able, M. (2019). Augmented data science towards industrialization and democratization of data science. *arXiv*, pages 1–10.
- [Yiu, 2019] Yiu, T. (2019). Understanding random forest. <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>. Accessed: 2020-06-30.

