Automating interpretable feature engineering for predicting human behavior

submitted in partial fulfillment for the degree of master of science

Jan Osolnik
11653523

master information studies
data science
faculty of science
university of amsterdam

June 2018

	Internal Supervisor	External Supervisor
Title, Name	prof. dr. T.M. (Tom) van Engers	Derek Willemsen
Affiliation	UvA	BUX B.V.

Universiteit van Amsterdam

Amsterdam
Data Science

ABSTRACT

In this research a system for automating prediction of human behavior is described. It leverages Deep Feature Synthesis for automated feature engineering and is built as an API which combines a generalizable and a custom script that provide the building blocks for building a Training and a Scoring pipeline. These can be used on problems in the specified problem space, characterized by the data that can be structured in an entity set which includes Cohorts, Individuals and Interactions entities. The training pipeline focuses on establishing trust into the model and provides the answer to the question: "Can we predict this using our data?". It includes performance metrics, feature importances and explanations of individual instance predictions (using Local Interpretable Model-Agnostic Explanations). The Scoring pipeline loads the model and the most relevant features extracted from the Training pipeline and scores on individuals that can be without available target values. The combination of both aim to provide trust into the predictions and deploying them in production with a re-usable API. The system is evaluated on a prototypical problem in the problem space of customer lifetime value prediction and shows in the report that it can explore a large feature space that would otherwise be explored by human intuition.

KEYWORDS

ML 2.0, Automated feature engineering, Deep Feature Synthesis, Local Interpretable Model-Agnostic Explanations, Customer Lifetime Value Prediction

1 INTRODUCTION

With machine learning and artificial intelligence technologies becoming one of the key driving forces of economic growth in today's data economy, it is worth considering the bottlenecks in the stages of machine learning pipelines. As it is noted by Pedro Domingos, the most important factor of whether a machine learning project succeeds or fails is often the features used. He adds that this is where most of the effort in the machine learning projects goes. It's the part where intuition and creativity are as important as the technical knowledge. [15]

Machine learning can provide insight for the increasing need of organizations to better understand their customers' behavior by detecting behavioral patterns that are computationally unfeasible to be perceived by humans. However, despite a large development of new machine learning discoveries in the research, most companies still often struggle how to deploy their predictive models to solve real business problems. This execution gap needs to be addressed in order to provide value from machine learning to various aspects of human activity. The problem data scientists often face is not the lack of ideas to consider but rather the available resources to test them. By automating the more time-consuming stages in the pipeline more insight from data can be extracted to provide value to various organizations, whether in the medical, financial, educational or other domains. [3]

Consequently the research question of this research is:

To what extent can we automate a generalizable and interpretable system for predicting human behavior?

In order to answer this question the focus will be on building a generalizable automated pipeline with the core component being automated interpretable feature engineering on relational data. The outcome of the pipeline answers the crucial question that is often asked when considering a problem that could be solved with the use of data: "Can we predict this with our data?" The goal is not to outperform other algorithms but rather solve the problem as soon as possible with extracting most of the feature space that would be otherwise explored with human intuition, after which it is to be estimated whether it is worth investing more human effort to improve the performance with an iterative approach by using intuition and domain expertise. By minimizing the time to first model consequently more questions can be answered which affects the business aspect of machine learning. The research is performed in programming language *Python*, while the main supporting libraries include *Pandas* for data manipulation, *Matplotlib* for data visualization, *Scikit – Learn* for machine learning and *Featuretools* for automated feature engineering.

Given the wide problem space that can be addressed using machine learning systems, combined with the time constraint of this research, a specific problem space was chosen with its underlying data structure - prediction of human behavior. Another reason is that the ability to automate a pipeline often has an inverse relationship with the scope of the problem space that the pipeline addresses. For this reason the system automation requires some assumptions about the problem space that it is aiming to answer questions for. The problem space is in this case defined by the available data that can be structured in a specific way in order to derive insight from it. For the produced pipeline to be re-used, the available data of the problem space needs be structured into tables with features for each of the entities (Cohorts, Individuals and Interactions):

(1) Cohorts table that includes features about the environment when the individuals started using the product
(2) Individuals table that includes features about the individuals that interact with the product
(3) Interactions table that includes raw features on interactions between the product and the individuals

Whereas the available Individuals and Interactions entities satisfy the necessary and sufficient conditions of the problem space that is addressed, Cohorts entity can provide some additional information gain to derive insight from, but it is not a necessary condition for reusability of the pipeline. A feature is defined as an individual measurable property of a phenomenon being observed [6]. It often includes transformations but not necessary (as it is used in Cohorts and Individuals). A raw feature is defined in this

case as a feature that doesn't include any transformations apart from possible time-based (e.g. daily) aggregations on an individual level (Interactions) to decrease the computational complexity. These tables, combined with the relationships between them define the entity structure of the problem space and can be found visualized in Figure 1.

The research is structured as follows. Section 2 focuses on giving the context of the research question and the building blocks in order to answer it. Section 3 focuses on explaining the approach that was taken to answer the research question both through a generalizable lens in the given problem space and also applying and evaluating it on the chosen prototypical problem. Section 4 explains how well the produced system performed and some specifics of the problems that were encountered in the process of building it, followed by Section 5 which elaborates the outcome through the context of the research question. Section 6 extends that to incorporate some ideas on how this system could be improved further.

2 BACKGROUND

As mentioned in section 1 there is often a need to rethink the way machine learning projects are approached in order to increase their impact. The new approach was explored in the definition of a new paradigm shift called ML 2.0 [12] which leverages Deep Feature Synthesis (DFS) algorithm to automate the feature engineering [13]. The approach used with DFS is called Expand Reduce and it is one of the approaches for automated feature engineering, such as hierarchical organization of transformations [21], meta learning [8] and reinforcement learning [22]. However, as DFS has been shown to consistently perform well on diverse datasets in combination with recently open-sourced library Featuretools which leverages DFS, this has been chosen as the approach to automate the arguably most time-consuming step of answering the research question - feature engineering.

2.1 ML 2.0

Most current approaches to machine learning (also referred as ML 1.0.) focuses on "discovery first". In order to provide an impact of these activities the discovery phase has to be subsumed by the more targeted goals of delivery and impact. This implies having a faster, 8-week process of development, understanding, validation and deployment. A core facet of this paradigm is ability to re-use the APIs, either by developers or subject matter experts. This means increasing the business impact of machine learning projects rather than only having it as a separate R&D activity with limited scope of ability to affect the decision-making process. [12]

With this approach a "minimum viable data-driven model" is built which can quickly answer the question "Can this problem be solved using a predictive model?"

There are multiple goals and requirements for an ML 2.0 system:

(1) Subsume discovery (deliver a machine learning model that hasn't been solved before or partial steps have not been taken).
(2) Be end-to-end (enable a developer to execute multiple steps - from an extraction of relevant data subset to learning the model and also building and validation of the model)
(3) Enable confidence (be able to test the end-to-end system under different simulated scenarios and conditions)
(4) Provide deployable code (to be able to deploy the model on the new data)
(5) Provide provisions for updates (to update, re-asses/validate and re-deploy it)

All of this has to be done with a minimal amount of coding, effort and machine learning expertise. This enables the democratization of AI with increasing the speed of building new AI applications. In order to satisfy the increasing demand for predictive models, ML 2.0 focuses on ways to automate parts of the machine learning systems that are human labor intensive. [12]

2.2 Deep Feature Synthesis (DFS)

Feature engineering, using domain specific knowledge to extract predictive patterns from a raw dataset often requires an insight from domain experts which slows down the process of building a predictive model. While generalized feature extraction algorithms have been well studied for machine vision and natural language processing, this has not been the case with relational and human behavioral data which remains driven by intuition and that way time-consuming. The problem with the current best performant "black-box" algorithms such as neural networks is that they are not as interpretable and scalable (more computationally expensive) as traditional machine learning algorithms. This led to the creation of Deep Feature Synthesis (DFS) algorithm for feature engineering. The Data Science Machine, a system that builds the features automatically using DFS is evaluated by comparing it to the performance of manually built machine learning pipelines in existing data science competitions (e.g. from data science competition platform Kaggle). The competitions focus on predicting a repurchase of a customer, whether a project will get funded by donors or where a taxi rider will choose to go. In 2 of these 3 competitions the automatically built pipeline beats the majority of competitors, and in the third one it achieves 94th percentile of the best competitors' score. They have also observed that many data science problems released by Kaggle and competitions at conferences have some common characteristics. Firstly, the data is structured, and relational (usually as a set of tables with relational links), secondly the data captures some aspect of human interactions with a complex system and thirdly the presented problem tries to predict some aspect of human behavior.

There are three core concepts behind DFS:

- Features are derived using the relationships between the data points in a dataset: The computation of feature engineering is usually found in databases or log files.
- Across datasets, many features are derived using similar mathematical operations: What needs to be calculated can be described in natural language in a different way, however the mathematical operation can be the same. In deriving the average customer order value or the average pilot flight delay, in both cases all the numerical values in a columns of a customer or a pilot are collected after which the values can be computed (as it is visualized for the case for the customer in 2). These mathematical operations, which are consequently data-agnostic, are also called "primitives".

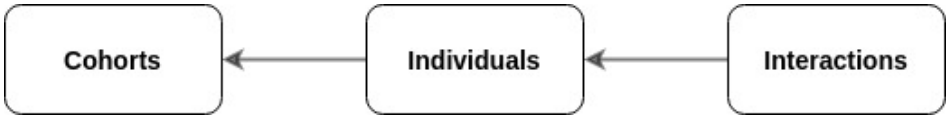

Figure 1: Illustration of the entity set schema. The directed arrow indicates a relationship between two entities

- New features are often composed using previously derived features: Primitives are the building blocks, in order to construct complex features they can be stacked to mimic the ones that are created by humans.

In order to perform the feature engineering at various depths there three feature abstractions used:

- $efeat$: Computed on 1 entity using transformation primitives on either a single row (calculation of a weekday, ratios and differences) or multiple rows (derivatives or bag of words).
- $dfeat$: Computed between 2 entities on a forward relationship to reflect metadata. No computation is performed.
- $rfeat$: Computed between 2 entities using aggregation primitives on a backward relationship to compute aggregate rows and apply functions, such as count, mean, standard deviation and others. These are computed across one-to-many (parent-to-child) relationships that group by the parent (one) and compute the values for the child (many).

The example of applying of all the feature abstractions is available in Figure 2.

2.3 Local Interpretable Model-Agnostic Explanations (LIME)

The features generated automatically with DFS are intuitive to understand, however it is crucial to add another component to interpretability of the prediction model: to gain trust into the model by showing how the model makes decisions upon specific instances. The reason for this is obvious: if the users don't trust the model or a prediction, they will most likely not use it. Whereas this is easier to achieve with simple models like linear regression and decision trees they usually lack the performance of more complex models. Ensemble methods like Random Forest and XGBoost provide a better performance but are harder to interpret on an instance level.

Accuracy or some other aggregate metrics provide implicit trust into the model but they can sometimes be misleading. An example of this is accidental leaks into the training data. For this reason a Local Interpretable Model-Agnostic Explanations (LIME) was developed, a technique to evaluate usefulness of a machine learning classifier in various tasks related to trust.

The toy example in Figure 3 below presents the intuition behind LIME. The black-box model's complex decision function f which is not known to LIME is represented by blue/pink background, and it is not possible to approximate with a linear model. The model is focused on explaining the instance that is represented by a bright bold red cross. LIME samples instances and predicts using f, while taking into account proximity to the instance that is being explained (in the Figure 3 determined by the size of individual instances). The learned explanation is represented by a dashed line and while not globally, it is locally faithful.

The result of this process is an interpretable model such as a linear model with various coefficients like in the example above. The general reasoning behind it is that it is easier to approximate any model (even a black-box model) by a simple model locally (close to the prediction that we want to explain) than to try to approximate it globally. Another characteristic of LIME is being model-agnostic which means that technique can be used on different models, from Random Forest predictions on text to neural network predictions on image classification. The latter example includes using the technique to divide an image into interpretable components in order to explain the output of a neural network. This process of individual instance interpretability becomes increasingly important with the importance of a decision, for instance in using machine learning for medical diagnostics or terrorism detection as the predictions cannot be acted upon on blind faith. [11]

2.4 Applications of ML 2.0 and DFS

There are currently two instances when ML 2.0 and DFS were applied in an industrial environment. These will be presented with the final outcome on the organizations where they were built, followed by how both are applied to a new problem in this research.

2.4.1 *Solving the "false positives" problem in fraud prediction*

One example of applying Deep Feature Synthesis at an Industrial Scale is at BBVA, a Spanish financial institution to dramatically reduce false positives in fraud prediction. It is estimated that only 1 in 5 of transactions declared as fraud are actual fraud and 1 in 6 have had a valid transaction declined in the past year. By using DFS the authors generated 237 features for every transaction and used a Random Forest classifier for prediction. On an unseen set of 1.85 million transactions they were able to reduce the false positives by 54% and provide savings of 190.000 âĆň. [23]

2.4.2 *AI project manager*

The first AI product delivered using ML 2.0 and Deep Feature Synthesis is The AI Project manager, a system the authors have built jointly with Accenture, a leading professional services company. The problem of many project managers is described. The analysis of an overwhelming number of projects is done after the project is done, to elaborate on what went right or wrong to identify the root cause. They built a machine learning model to identify overarching patterns in time series data to predict critical problems before they happen. This provides value to project managers to better allocate their energy to a subset of projects that need their attention the most and provide the most value to the organization they're a part of. The project manager can see the predictions of the projects that they are responsible for managing in the system's user interface. Extra emphasis was put on the value of answering the question of whether it is possible to predict the outcome of interest with

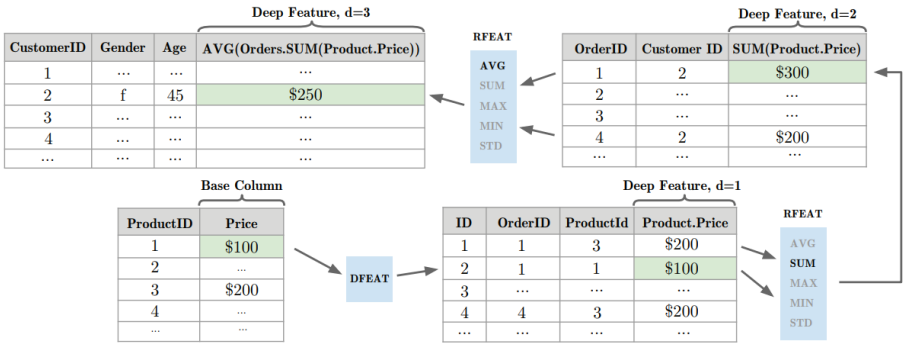

Figure 2: Illustration of generating $rfeat$, $dfeat$ and $efeat$ features at different depths, d [13]

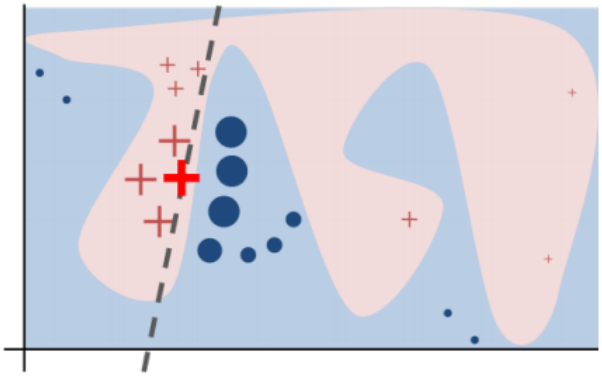

Figure 3: Intuition behind LIME's local approximation [11]

the available data. The end-to-end project to develop and deliver spanned over 8 weeks due to the practice of ML 2.0 in live testing. The end result is a system that predicts potential issues 80% of the time which enables an improvement of key performance indicators that are relevant to project delivery. [4]

2.4.3 *Customer lifetime value prediction*

The system in this research will be applied to predicting Customer Lifetime Value (CLV), which measures all the future profits that a given customer will generate in their entire lifetime for a given business. This knowledge is crucial to maximize efficiency of a business. It provides insight into how much it is reasonable to spend to acquire, or retain a customer and also which segments of users the business should focus on to maximize its returns. The underlying dynamics that organizations often face is a power law distributed generated revenue in the customer base - that a small percentage of customers (also known as "whales") generate the majority of the profits for the organization.

In the case of a digital product business being able to identify these customers early in their start of using the product, it provides actionable insight that can be incorporated in the decision-making. One example is tracking the future cash flow of new cohorts which enables to see how different acquisition channels perform and adjust the marketing spending to channels that acquire users with a relatively higher CLV (while taking into account customer acquisition cost - CAC). Another is personalized cross-selling or up-selling of existing customers with various incentives (e.g. push notifications, e-mail campaigns) to increase their CLV. As there is evident value in being able to predict the future spending of individuals, there is also existing research in using various machine learning approaches to solve this problem. With the change of the monetization strategy of a business there also comes a different way to measure behavioral patterns of customers. The available data, insight and actionability vary across different business models while the structure of the data often does not. An auto repair and maintenance company [5] might have a similar data structure as an e-commerce company [2] or an real-estate company [9], the common theme being transactions of an individual with the business's product or service. In the case when that is not true, it would mean a change in the structure of the entity set where the presented system would not be applicable for re-use.

The available data used in this research is provided by BUX, a Dutch financial technology startup that enables simple and affordable trading of various asset classes - stocks, indices, currencies,

cryptocurrencies and commodities. Their product focuses on trading contracts for difference (CFDs) which offers European traders and investors an opportunity to profit from price movement without owning the underlying asset [10]. BUX users trade in virtual currency (using the product as a fun BUX user) and after which they might decide to switch to real money (using the product as a serious BUX user), with all the risk involved. Serious BUXers, the users that convert (decide to start trading with real money) generate the vast majority of the revenue for the business, either from commissions (transaction fee on each trade) or financing fees (using a multiplier which allows to effectively multiply the amount invested by borrowing money from BUX). In addition to that, just like in the provided examples above, the generated revenue of the user base follows a power law distribution - a minority of serious BUX users (whales) generate the majority of revenue for the business. For all of these reasons BUX faces the same question as many other businesses: "How much will a user spend in their entire lifetime of using our product or service"? The estimation is performed on a large amount of data about the usage of the product that can be used to detect behavioral patters of different types of users, one being a classification of whether a user is likely to become a whale in their lifetime (a high spender) or not. This is what the system with the underlying data will be explained in Section 3.2 and evaluated in Section 4.

3 METHODOLOGY

The approach to answer the research question is addressed in this Section. The aim is to produce an automated system that is evaluated in Section 4 on a specific prototypical problem in the problem space. Section 5 provides a general view of the re-use of the system. This Section is split into two parts, Research (Section 3.1) and Application (Section 3.2). Research describes the overall system that can be re-used to build predictions in the problem space defined in Section 1 and Application focuses on applying the system on a specific problem, predicting customer lifetime value (CLV). Both Research and Application include two seperate pipelines:

- Training pipeline (Section 3.1.1, Section 3.2.1)
- Scoring pipeline (Section 3.1.2, Section 3.2.2)

Each has its own steps:

The system adapts automatically by changing the inputs - a set of hyper-parameters that affect specific parts of the computation (e.g. prediction problem type).

Featuretools provides an approach to applying a cutoff value to prevent data leakage - using data in the training process that shouldn't be available, and is done after loading the data into memory. While this has been proven useful in a large number of use-cases (and can be made more efficient by loading datasets sequentially by tools like Dask or Spark), the filtering in this research is performed on the database level to minimize the amount of data loaded into the memory.

3.1 Research

The use-case of a prediction problem can be structured to include behavioral data across a specific time frame to predict a target value.

As mentioned in Section 1, *Featuretools*, an open-source library for automated feature engineering is used. The library leverages the Deep Feature Synthesis algorithm to build the features automatically based on the entity set and defined relationships between individual entities. For building the Random Forest algorithm the open-source library $Scikit - Learn$ is used.

Figure 4 provides an illustration of the two scripts used as a part of the API, each with its own set of functions.

The generalizable script can be used on any problem in the problem space. It includes functions for each of the steps in both of the pipelines, such as connecting to the database, training the model, generating the report and loading the scores in the database.

The custom script provides functions specific to the chosen problem, and extracts and transforms the data so that it is aligned with the specified data structure, a necessary condition to use the system. The custom script leverages functions from the generalizable script (such as a connection to the database), and can be changed to any problem in the problem space.

These two scripts and their corresponding functions are used to build the end-to-end system as described in the steps in this section.

3.1.1 *Training pipeline*

Step 1: Defining the hyper-parameters

The hyper-parameters for building the pipeline are defined first. These are specific parts of the pipeline that can be manipulated to adjust to different demands of the final outcome, such as the amount of data included and the decision boundaries to make predictions on. Whereas data structure is usually static in a specific data set, there is a need for certain characteristics of the system to be dynamic. These hyper-parameters include:

- Time range of the individuals that are included in the sample based on the time they started using the product
- The limit of the number of individuals per months (cohort size) to reduce the complexity of computation and simultaneously be able to take a larger time range for better generalization to out-of-sample users
- The time range of the behavioral data for each of the individuals included to calculate the features on
- Prediction problem type (regression, binary classification or multi-class classification)
- The decision boundaries (specific to binary and multi-class classification)
- Number of the most important features to extract and retrain the algorithm on (to minimize over-fitting)
- Revenue generated by acting on the predictions (specific to binary and multi-class classification)
- Cost incurred by acting on the predictions (specific to binary and multi-class classification)

Step 2: Building the entity set and labels

A connection to the database is established which includes all the relevant data. With the connection established the relevant features of Cohorts, Interactions and Individuals entities are extracted using the database connection. Choosing the right subset of tables and columns to use as entity sets is done manually and involves intuition of which raw (untransformed) features could be useful to predict the target values.

Steps	Training pipeline	Scoring pipeline
1	Defining the hyper-parameters	Defining the hyper-parameters
2	Building the entity set and labels	Building the entity set and labels
3	Deep feature synthesis	Deep feature synthesis
4	Training and saving the model with top features	Loading the model and top features
5	Prediction (test data)	Prediction (on all data)
6	Performance report	Load predictions into the database

Table 1: Illustration of the steps in Training and Scoring pipelines

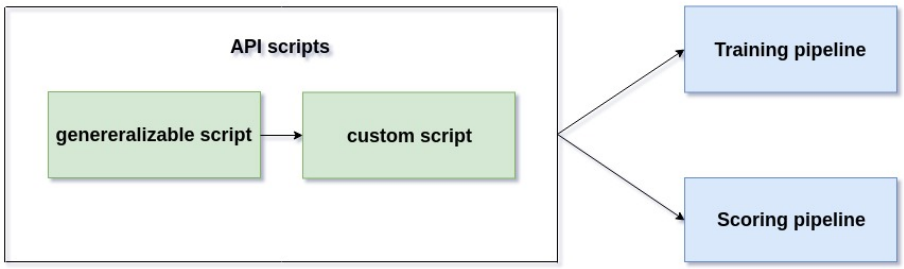

Figure 4: Visualization of an API script collection as a combination of using a generalizable script and a custom-built script, adapted to the a data structure in the specified problem space

The limitation of number of individuals per cohort is done by creating a temporary table which can be used to query the sample of individuals on which all the entities are extracted for.

Next the entity set is created, followed by defining the names of the entities, the entity tables and the possible time indexes. After this the relationships between the entities are specified. Relationships are encoded as a list of parent entity, child entity and the key by which the entities are related. This is fed into a function using *Featuretools* capabilities to create the entity set.

Step 3: Deep feature synthesis

The entity set is the input into the Deep Feature Synthesis algorithm. The primitives are defined. Transformation primitives include Day, Week, Month, Weekday, Weekend. Aggregation primitives include Sum, Std, Max, Min, Mean, Count, PercentTrue and NUnique. The primitives can be changed by the user of the system

Step 4: Training and saving the model with top features

The next step is to create the labels based on the prediction problem specified as a pipeline hyper-parameter in the first step of the pipeline. The data is then split into training and testing, followed by training a Random Forest algorithm on the training data.

A number of most relevant features for the prediction problem are extracted based on the pipeline hyper-parameter specified in Step 1. This is done to avoid over-fitting by using too many features. After this, the fitted model is saved so that it can be reused later in Section 3.1.2.

Step 5: Prediction (test data)

The trained Random Forest model is used to predict on the testing set.

Step 6: Report

The report includes:

- Performance metrics (R^2 and *RMSE* for regression, accuracy, F1 score, precision, recall and accuracy for classification). The performance metrics are evaluated using 5-fold cross-validation to quantify possible over-fitting.
- Feature importances of the most relevant features.
- Explanation of predictions for a sample of individuals by taking 5 individuals that have the highest value and 5 individuals with the lowest value for the most relevant feature. This is done to ideally extract individuals with both ends of extremes of target values.

The core component for binary classification is also applying a domain-specific cost function that takes into account the impact of predictions and changing the decision threshold accordingly. The question such as "what is the impact of applying the predictions in practice?" are crucial in deciding whether to use the predictions in the decision-making process. The cost function is adopted to a typical question when applying machine learning models in marketing: "How much revenue is generated or cost incurred by taking or not taking an action on the predictions?" Figure 11 in Appendices provides the visualization of cost-benefit analysis of using the system. A function is applied to find the optimal decision threshold (from 0.1 to 0.9 with the increment of 0.1 which takes into account the vast majority of the decision threshold space) to maximize the expected profit of using the model.

With this report a decision is made whether the performance of the model is valid to provide value.

3.1.2 *Scoring pipeline*

Step 1: Defining the hyper-parameters

The model and features are available and the decision has been made that the model is valid. This in turn means that the time frame is guided to score on out-of-sample individuals for which the target values are not necessarily available yet (e.g. prediction on users that started using the product in the last week). The hyper-parameters are the similar as described in section 3.1.1, the only difference being that there is no limit applied to the cohort size as the scores on all users need to be calculated to be able to provide actionable insight.

Step 2: Building the entity set and labels

The process is the same as described in Section 3.1.1

Step 3: Deep Feature Synthesis

The process is the same as described in Section 3.1.1

Step 4: Loading the model and top features

Loading the model and the most relevant features, both saved in Section 3.1.1.

Step 5: Prediction (all data)

The loaded model is used to predict on the feature matrix.

Step 6: Load predictions into the database

The predictions are loaded into the database with which they can be used to support decision-making.

3.2 Application

As was mentioned in 2.4.3, customer lifetime prediction represents an important problem to solve in many organizations. Consequently it has been chosen as a prototypical prediction problem to test the system on. That means using the system for binary classification on whether a user will become a whale or not. The transactional data includes all interactions with the product in the first 3 weeks of using the product as a user. This time frame is chosen to both gather enough data (1 week was not enough for sufficient performance) and also have the insight into cohort performance soon after the start of using the product. It is used to build features automatically to describe behavioral patterns which are used to predict the probability of a user becoming a whale. As there is no information about the actual customer lifetime value, a customer value after 6 months of usage is used as a proxy because the majority of variability in CLV is explained in that period of time.

Figure 5 provides a visualization of the implementation for both pipelines. The model is trained with the Training pipeline every month, whereas the scores are produced daily with the Scoring pipeline when a new set of users acquire 3 weeks of usage of the product. The last trained model is used to score the users. The users included to build training features have started using the product between -6 months and -18 months before the time of execution of the Training pipeline. For both pipelines 3 weeks of usage is used to build the features, the only difference is the time frame and that Training pipeline randomly samples users from each cohort, while Scoring pipeline extract all of the users available to produce scores for.

Figure 6 provides the application of the API scripts. The *utils* script can be used on any problem in the problem space whereas the *bux_utils* script provides functions specific to the chosen problem at BUX.

3.2.1 *Training pipeline*

Step 1: Defining the hyper-parameters

The specified hyper-parameter for the time frame to extract data from is 1 year, from October 2016 till September 2017, such that all the users have at least 6 months of usage data available to calculate the target values. The cohort size (the number of individuals in each of the months of data) is set to 3000, which means that 36000 (12*3000) users are extracted. The hyper-parameters for revenue and cost of acting on a classification of a whale are set at 25 and 5 units (as illustrated in the cost matrix in Figure 12). The prediction problem varies as all three possible variants are tested (whereas as the evaluation is performed only on binary classification):

- Regression: Prediction of the continuous target value (6 month customer value)
- Binary classification: Prediction of the class of the user, either as a whale or a non-whale (whale - a users that are in 99th percentile of generated revenue for the business based on 6 month customer value. The 99th percentile is calculated across all the user base and is defined as a static cutoff point even when the time frame of the users changes, that way also static in both Training and Scoring pipeline)
- Multi-class classification: Prediction of the class of the user (low value user, medium value user and high value user based on 6 month customer value)

Step 2: Building the entity set and labels

A connection with the database is established, and a function is executed for building each of the entities:

(1) Cohorts (raw features describing the environment of the week when the users started using the product - volatility of each of the asset classes that can be traded in the product, including the values normalized by the average 2-year volatility value for each of the asset classes)
(2) Users (raw features describing the individual users)
(3) Transactions (raw features describing daily summaries of interactions with the product, such as the session length, the use of leverage, and which asset classes are traded).

The target values are then extracted for each of the users for each of the prediction problem types. This is followed by specifying the entity tables, time indexes, relationships between the entities and the name of the entities.

There are two relationships defined, the first one of Cohorts entity as a parent entity and Users entity as a child entity, the second one of Users entity as a parent entity and Transactions entity as a child entity. With both entity tables and relationships defined the entity set is created.

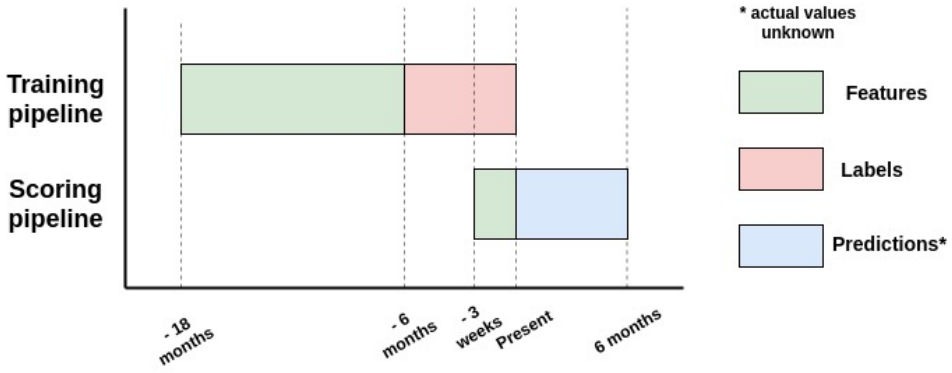

Figure 5: Visualization of time-scales for Training and Prediction pipeline (based on the CLV machine learning system implemented at ASOS [2])

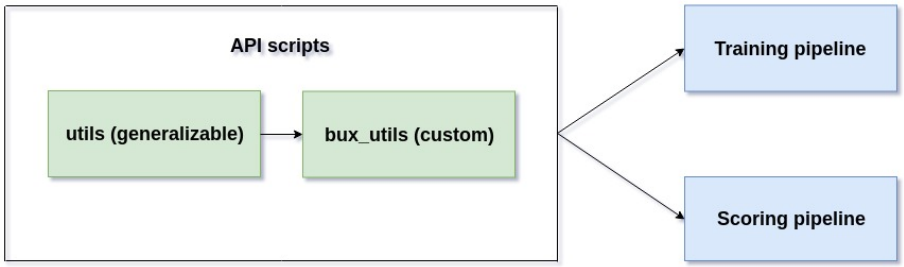

Figure 6: Visualization of an API script collection as a combination of using a generalizable script (*utils*) and a custom-built script, adapted to the data in the specified problem space (*bux_utils*)

Step 3: Deep Feature Synthesis

The created entity set is an input into DFS to build the features automatically. A feature matrix is created with 292 features out of 10 columns in Cohorts entity table, 17 columns in Transactions entity table and 31 columns in Users entity table.

Step 4: Training the model

The feature matrix (X) and target values (y), which vary by prediction problem type are split into training and testing set (in which 30% of users are included, a default value increased by a margin from 25% being the default value in $Scikit - Learn$). The model is then trained on the training set. This is followed by fitting the model to predict on the testing data.

Step 5: Saving the model and extraction of the most relevant features

The model is re-built and saved with the most relevant features, chosen based on feature importance. The number was chosen by experimenting with different numbers of features and noticing a high diminishing value of adding additional features, while there was also an increase in the difference in performance on different folds (increased variance).

Step 6: Report

A report is generated with all the components mentioned in section 3.1.1. The report produced for the binary classification is available in section 4. Figure 12 in Appendices provides the visualization of cost-benefit analysis of using the system on the specific use-case at BUX by making the benefit and cost of the incentive for both targeting and not targeting users. The optimal threshold is calculated by taking into account benefits and costs of acting on the predictions when a user is classified as a whale. The crucial part is that revenue (25 units) is much higher than cost (5 units) with which there is implicitly more weight given to recall than precision. The values are chosen to reflect the business goals, however the actual values are unknown at the time of this research. The threshold with the maximum expected profit is chosen.

3.2.2 *Scoring pipeline*

Step 1: Defining the hyper-parameters

Whereas the time range of the extracted users span across a year in section 3.2.1 it is used here to score on users that have at least 3 weeks of usage, with 6 months of usage not being a necessary condition anymore.

Step 2 - Step 6

The same process as described in 3.1.2 performed by using functions from the *utils* and the *utils_bux* scripts.

Scoring is done on a weekly basis to score new users that have been using the product for 3 weeks while training is done on a monthly basis which is enabled by running both pipelines as a job.

There were also two sets of features built manually via an iterative approach by getting feedback from the domain experts. The first set includes features in Users entity calculating time to various events (extracted from different tables so the calculations had to be done using SQL). The second set includes features in Cohorts entity calculating the average volatility of each of the asset classes for each of the cohorts (a week of data when the users started using the product), including the values normalized by the average volatily in the last 2 years.

A crucial part of building the features was also to understand clearly how the data is generated to avoid data leakage (information from outside the training dataset that is used to create the model) as this can create an overly optimistic predictive model which oftentimes generalizes poorly on out-of-sample data. It would often be easy to overlook this without a clear feedback on the generated features from the data experts at BUX. So without doubt there is a need to either have a clear understanding of the data or creating a feedback loop on the modeling process with someone who does.

4 RESULTS

This Section addresses the results of the system on the prototypical problem of customer lifetime value prediction. Whereas the system can be applied to either of the prediction problem types it is evaluated on a binary classification task - classifying users as either a whale or a non-whale.

The training pipeline is built with the hyper-parameters mentioned in Section 3.2.1. As there is an apparent high class imbalance in the dataset the performance metrics that the pipeline is evaluated on include F1 score, precision and recall.

Figure 7 in Appendices provides an illustration that the system before applying the cost function (decision threshold set at the default value, 0.5) has larger issues with false negatives (identifying a whale as a non-whale, influencing recall) than false positives (identifying a non-whale as a whale, influencing precision).

Figure 8 in Appendices provides an illustration of the confusion matrix of the system after applying the cost function to maximize expected profit (after which the decision threshold set at 0.2). Expected profit is calculated as an element-wise summed product of the cost function (Figure 12) and the confusion matrices that change with the changing decision threshold. As it's defined implicitly by providing the cost function, the priority of the system is to provide business value by correctly identifying as many whales as possible. Consequently recall is of higher priority than precision. After changing the threshold to 0.2, 65 % of whales are identified after 3 weeks of usage (at the cost of a lower precision, 50 %).

The core performance metrics using a 5-fold cross-validation (using the default threshold) are: (the two numbers are the mean performance metric and the mean standard deviation of that performance metric over all 5 folds)

$$F1score = 0.56 \pm 0.03$$

$$Precision = 0.73 \pm 0.05$$

$$Recall = 0.48 \pm 0.05$$

The above metrics show that the performance is consistent through different folds which implies that it is unlikely that there is a problem of model overfitting.

Table 2 shows a sample of the 5 most relevant features for predicting whether a user will become a whale. The full feature set with feature importances for all most relevant features used is available in Table 4 in the Appendices.

All of the features can be explained intuitively, such as the number of unique trades as a serious BUXer or the maximum amount of open positions at the end of the day as a serious BUXer. The explanations for all of the most relevant features are available in Table 3 in the Appendices. Having intuitively interpretable features establishes additional trust into the predictions of the system.

As mentioned in Section 3.2 there was a set of features built manually. There is 1 manually built feature in the 20 most relevant features and while it cannot be estimated how much more feature space can be explored manually, it is clear that a large portion has already been explored with DFS (considering the intuitive explanations behind the features).

Section 1 discusses a big challenge that organizations often face in how to embed the output of the models into the existing processes to improve the decision-making process. The aim of the applied system was to evaluate different cohorts of users through time and how many whales there are currently in each of the cohorts (as that has a large impact on the revenue generated per each cohort). Arguably even more importantly is to answer how many of the users will become a whale as it is more actionable. By producing scores with the system on the user base of the last 30 months the scores were calculated which was then used to evaluate different cohorts. This was visualized to get an intuitive sense of how the cohorts evolve through time.

Apart from that the report also includes explaining predictions on individual instances. Figure 9 in the Appendices provides an explanation of an individual prediction with the most relevant features influencing the prediction. There is a low confidence in the classification of either of the classes in the chosen instance. It is possible to interpret that the user is 12% more likely to become a whale because of a high largest open position in the end of the day, combined with only taking a day to deposit its own money and other features that can be seen in the figure. Figure 10 in the Appendices provides an explanation of an individual prediction with the most relevant features influencing the prediction. There is a higher certainty of the user becoming a whale based on the feature values of the user (feature values in orange color indicating a characteristic of a whale). There is a 14% increase in probability of becoming a whale based on the dispersion of open positions at the end of the day, followed by a 12% increase because of the amount of the maximum open position at the end of the day.

Feature name	Feature importance
SUM(transactions.trades_sb_invested_amount)	0.044
NUM_UNIQUE(transactions.trades_sb_commission)	0.032
MAX(transactions.trades_sb_invested_amount)	0.029
Conversion Completed_hours_till_event	0.028
SUM(transactions.total_session_duration)>	0.023

Table 2: The 5 most relevant features for classifying whales

5 CONCLUSION

This Section recaps the research with a general overview of the system. It discusses the value of re-using it on other problems in the problems space from the perspective of generalizability, satisfying the requirements for an ML 2.0 system and performance on the prototypical problem.

The final outcome is a re-usable automated system in the specified problem space that minimizes the time to first model. It is enabled by using Deep Feature Synthesis algorithm to automate arguably the most labor-intensive part of applied machine learning - feature engineering. This means building a minimum viable data-driven model that determines whether a problem can be solved with available data combined with the report for explanation of individual predictions to establish trust into the model. The model can be used for scoring on new instances in a productionized environment by using the same API as was used in the training.

The system was built while keeping in mind the goals and requirements of an ML 2.0 system (defined in Section 2). The discovery is subsumed for a problem that hasn't been solved using this methodology. The pipeline is end-to-end from the extraction of raw features for each of the entities to scoring on out-of-sample users and loading the data into the database. The system enables confidence by using LIME to explain predictions on users that are randomly sampled and using performance metrics to showcase predictive power of the algorithm. In addition to that, it provides a deployable code which can be re-used on other problems in the specified problem space. The system also provides provisions for updates which are enabled by transforming the notebooks into scripts that can be used to score on new users in the form of scheduled jobs.

By evaluating the pipeline it is clear that the features built have a discriminative power between whales and non-whales as 65% of whales can be correctly classified with 50% confidence when classifying a user after 3 weeks of usage. The feature set can additionally be complemented with features using intuition that extract feature space that hasn't been extracted with DFS. The most relevant features selected intuitively make sense, such as the sum of all the session time of a customer, sum of the transaction amount of a customer and the number of times a customer viewed their positions. This clearly indicates that Deep Feature Synthesis algorithm has capability to extract a considerable amount of the feature space that would otherwise be extracted using intuition, taking considerably more human involvement. The end-to-end system was built with the first scores produced in 6 weeks which would not be possible without the use of *Featuretools* for automated feature engineering.

In regard to the research question, it has been shown that a large part of the end-to-end system can be automated when the problem space that the system can be used for is limited. In the Section 6 there are also examples where this assumption is not made and more advanced methods are used.

The repository with the code used in this research and the information on re-use of the system can be found on: https://github.com/josolnik/msc-thesis-bux. The content of *bux_utils* script functions are omitted for confidentiality reasons.

6 FUTURE WORK

While there is clear value in interpretability of models, the inherent nature of human reasoning needs to be taken into account to maximize the likelihood that the interpretations are correct. Cognitive biases are defined as systematic, often irrational reasoning patterns which allow humans to make fast and risk-averse decisions that happen due to cognitive limitations, motivational factors and/or adaptations to natural environments [1]. The cumulative impact of these biases in machine learning should not be understated as algorithmic decision-making becomes more prevalent with time. There are over 50 cognitive biases discovered to date [16] in large part by Tversky and Kahneman starting in 1970s [7].

Cognitive biases have been shown to be reliable, systematic and hard to eliminate which poses a difficult problem when it comes to induction of rules for decision-making with the use of algorithms. There is an existing research in possibly debiasing techniques of 20 cognitive biases. The aim is to systematically relate cognitive biases to the interpretation of machine learning results. The examples include base-rate fallacy (unable to to correctly process conditional probabilities) and confirmation bias (tendance to look for evidence supporting the current hypothesis). The focus is on asking questions such as: "Which cognitive biases affect understanding of symbolic machine learning models?". By implementing various debiasing techniques in the process of interpreting machine learning algorithms the positive impact of the improved decisions can be maximized.

To improve the system's performance it is possible to go a step further and automatically choose the best algorithm for the problem and tune its hyper-parameters ([18] and [14]). Other problems spaces can also be explored with a similar approach. In addition to LIME, other methods can provide additional interpretation of the system predictions, such as SHAP (SHapley Additive exPlanations) that draws insights from game theory to feature attribution methods in machine learning [17].

7 ACKNOWLEDGEMENT

I would like to thank both Tom van Engers and Derek Willemsen for their help and guidance on this research. It has been a fun ride.

REFERENCES

[1] Amos Tversky, Daniel Kahneman. 1974. Judgment under uncertainty: Heuristics and biases, Science 185 1124âĂŞ1131.

[2] Benjamin Paul Chamberlain, Angelo Cardoso, C.H. Bryan Liu et al. 2017. Customer Lifetime Value Prediction Using Embeddings. Retrieved May 10, 2018 from https://arxiv.org/pdf/1703.02596.pdf

[3] Benjamin Schreck, Max Kanter, Kalyan Veeramachaneni, Sanjeev Vohra, Rajendra Prasad. 2018. Getting Value from Machine Learning Isn't About Fancier Algorithms âĂŤ It's About Making It Easier to Use. Retrieved May 20, 2018 from https://hbr.org/2018/03/getting-value-from-machine-learning-isnt-about-fancier-algorithms-its-about-making-it-easier-to-use

[4] Benjamin Schreck, Shankar Mallapur, Kalyan Veeramachaneni et al. 2018. The AI Project Manager. Retrieved April 18, 2018 from https://www.featuretools.com/wp-content/uploads/2018/03/AIPM.pdf

[5] C.J.Cheng, S.W. Chiu, C.B.Cheng, J.Y. Wu. 2012. Customer lifetime value prediction by a Markov chain based data mining model: Application to an auto repair and maintenance company in Taiwan. Retrieved May 1, 2018 from https://www.sciencedirect.com/science/article/pii/S1026309812000107?via%3Dihub

[6] Christopher Bishop. 2006. Pattern recognition and machine learning. Berlin: Springer. ISBN 0-387-31073-8

[7] Daniel Kahneman, Amos Tversky. 1970. Prospect Theory: An Analysis of Decision under Risk: Econometrica, 47(2), pp. 263-291

[8] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana et al. 2017. Learning Feature Engineering for Classification. Retrieved May 20, 2018 from https://www.ijcai.org/proceedings/2017/0352.pdf

[9] Jian Wang, Murtaza Das, Amar Duggasani. 2016. Predicting and measuring customer lifetime values for apartment tenants. Retrived May 2, 2018 from https://link.springer.com/content/pdf/10.1365%2Fs41056-016-0015-0.pdf

[10] Khader Shaik. 2014. Managing Derivatives Contracts: A Guide to Derivatives Market Structure, Contract Life Cycle, Operations, and Systems. Apress. p. 74. ISBN 978-1-4302-6275-6

[11] Marco Tulio Ribeiro, Sameer Singh and Carlos Guestrin. 2016. Why Should I Trust You? Explaining the Predictions of Any Classifier. Retrieved May 15, 2018 from https://arxiv.org/pdf/1602.04938.pdf

[12] Max Kanter, Benjamin Schreck, Kalyan Veeramachaneni. 2018. Machine learning 2.0 Engineering data driven AI products. Retrieved April 15, 2018 from https://www.featuretools.com/wp-content/uploads/2018/03/ml20.pdf

[13] Max Kanter, Kalyan Veeramachaneni. 2015. Deep Feature Synthesis: Towards Automating Data Science Endeavors. Retrieved April 15, 2018 from http://groups.csail.mit.edu/EVO-DesignOpt/groupWebSite/uploads/Site/DSAA_DSM_2015.pdf

[14] Matthias Feurer, Aaron Klein, Katharina Eggensperger et al. 2015. Efficient and Robust Automated Machine Learning. Retrieved May 14, 2018 from https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

[15] Pedro Domingos. 2012. A few useful things to know about machine learning:. Retrieved April 10, 2018 from https://homes.cs.washington.edu/p̃edrod/papers/cacm12.pdf

[16] Rűdiger Pohl. 2017. Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory, Psychology Press. 2nd ed.

[17] Scott M. Lundberg, Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. Retrieved June 1, 2018 from http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[18] Thomas Swearingen, Will Drevo, Bennett Cyphers et al. 2017. ATM: A distributed, collaborative, scalable system for automated machine learning. Retrieved May 20, 2018 from https://ieeexplore.ieee.org/document/8257923/

[19] Tomáš Kliegr, Štěpán Bahník, Johannes Fűrnkranz. 2018. A review of possible effects of cognitive biases on interpretation of rule-based machine learning models. Retrieved May 10, 2018 from https://arxiv.org/pdf/1804.02969.pdf

[20] The Economist. 2017. Data is giving rise to a new economy. Retrieved April 10, 2018 from https://www.economist.com/news/briefing/21721634-how-it-shaping-up-data-giving-rise-new-economy

[21] Udayan Khurana, Deepak Turaga, Horst Samulowitz. 2016. Cognito: Automated Feature Engineering for Supervised Learning. Retrieved May 17, 2018 from https://ieeexplore.ieee.org/document/7836821/

[22] Udayan Khurana, Horst Samulowitz, Deepak Turaga. 2017. Feature Engineering for Predictive Modeling using Reinforcement Learning. Retrieved May 18, 2018 from https://arxiv.org/pdf/1709.07150.pdf

[23] Wedge, Max Kanter, Kalyan Veeramachaneni, Santiago Moral Rubio and Sergio Iglesias Perez. 2017. Solving the "false positives" problem in fraud prediction. Retrieved May 21, 2018 from https://arxiv.org/pdf/1710.07709.

Appendices

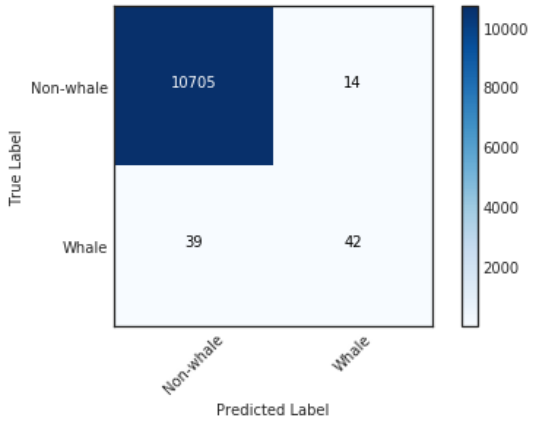

Figure 7: Computed confusion matrix for classification into whales and non-whales before applying the cost function

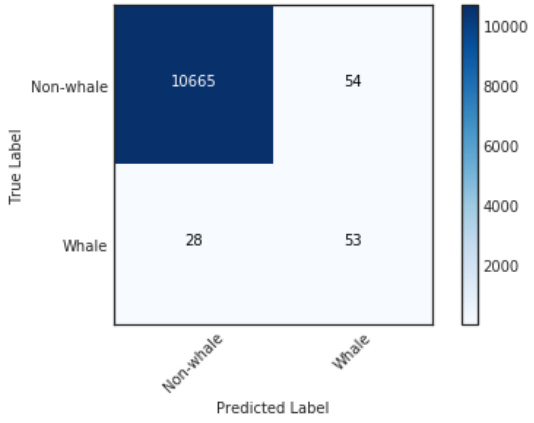

Figure 8: Computed confusion matrix for classification into whales and non-whales after applying the cost function

, ,

Feature name	Feature intuition
SUM(transactions.trades_sb_invested_amount)	The total amount of held open positions as a serious BUXer*
NUM_UNIQUE(transactions.trades_sb_commission)	Unique trading days as a serious BUXer
MAX(transactions.trades_sb_invested_amount)	The maximum amount of a held open position as a serious BUXer*
Conversion Completed_hours_till_event	Number of hours till conversion into a serious BUXer
SUM(transactions.total_session_duration)>	The total session duration
STD(transactions.conversion_to_sb)	Standard deviation of conversion to BUX
SUM(transactions.trades_sb_open_positions)	The total number of open positions as a serious BUXer*
MAX(transactions.trades_sb_open_positions)	The maximum number of open positions as a serious BUXer*
MAX(transactions.total_session_duration)	The maximum total session duration
MEAN(transactions.total_session_duration)	The average session duration per day
SUM(transactions.view_position)	Total viewed positions of a user
STD(transactions.total_session_duration)	The dispersion of the duration of sessions per day
MEAN(transactions.view_position)	The average viewed positions per day
MIN(transactions.total_session_duration)	The minimum length of a session in a day
MEAN(transactions.trades_sb_short)	The average short positions per day as a serious BUXer
STD(transactions.view_position)	The dispersion of the viewed positions in a day
MAX(transactions.view_position)	The maximum number of view positions in a day
SUM(transactions.trades_sb_long)	The average open long positions as a serious BUXer*
MEAN(transactions.trades_sb_open_positions)	The average number of open positions as a serious BUXer*
STD(transactions.education_topic_read)	The dispersion of the educational topics read per day

* at the end of the day

Table 3: The 20 most relevant features for classifying whales with intuitive explanation

Feature name	Feature importance
SUM(transactions.trades_sb_invested_amount)	0.044
NUM_UNIQUE(transactions.trades_sb_commission)	0.032
MAX(transactions.trades_sb_invested_amount)	0.029
Conversion Completed_hours_till_event	0.028
SUM(transactions.total_session_duration)>	0.023
STD(transactions.conversion_to_sb)	0.022
SUM(transactions.trades_sb_open_positions)	0.022
MAX(transactions.trades_sb_open_positions)	0.020
MAX(transactions.total_session_duration)	0.020
MEAN(transactions.total_session_duration)	0.019
SUM(transactions.view_position)	0.019
STD(transactions.total_session_duration)	0.018
MEAN(transactions.view_position)	0.017
MIN(transactions.total_session_duration)	0.017
MEAN(transactions.trades_sb_short)	0.016
STD(transactions.view_position)	0.016
MAX(transactions.view_position)	0.015
SUM(transactions.trades_sb_long)	0.015
MEAN(transactions.trades_sb_open_positions)	0.013
STD(transactions.education_topic_read)	0.012

Table 4: The 20 most relevant features for classifying whales with feature importance

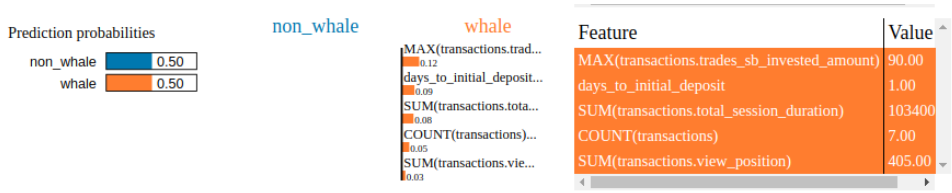

Figure 9: Explanation of a prediction on an individual user using LIME - uncertain prediction of the user becoming a whale

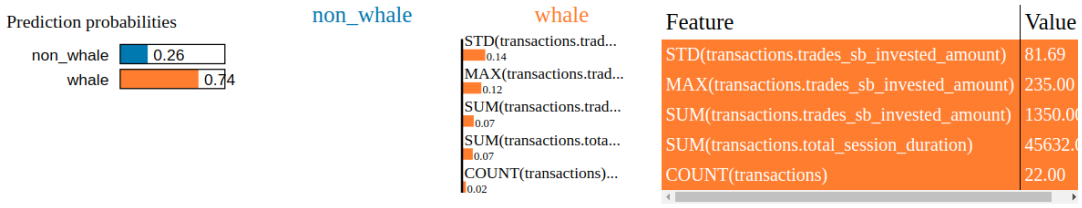

Figure 10: Explanation of a prediction on an individual user using LIME - predicting a high likelihood of the user becoming a whale

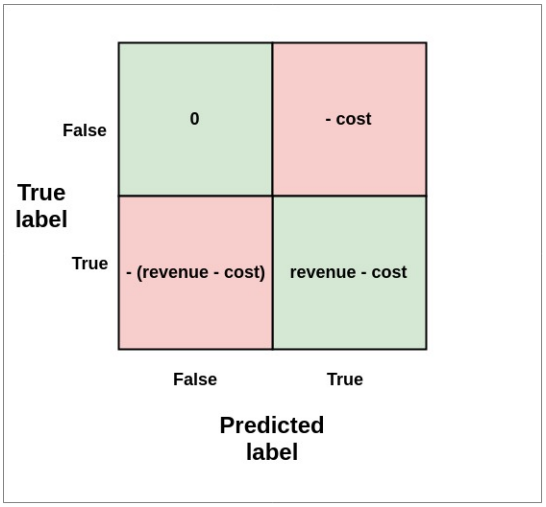

Figure 11: Cost-benefit analysis of the domain-specific cost function - generalizable to marketing domain

, ,

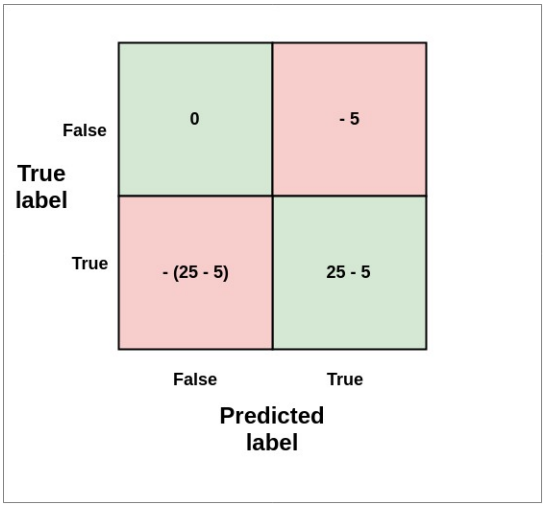

Figure 12: Cost-benefit analysis of the domain-specific cost function applied to BUX