Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Master's diploma thesis

in the field of study Data Science

Transfer learning for smart city data streams

Karol Krupa

student record book number 290529

thesis supervisor

PhD Maciej Grzenda

WARSAW 2022

Abstract

Transfer learning for smart city data streams

The primary objective of the thesis is to analyse existing and develop novel transfer learning methods applicable to smart city data streams. Reference problems to consider include the problem of delay prediction for public transport vehicles. In such cases, knowledge arising from models developed for e.g. bus lines can be possibly transferred to a related problem of tram delay prediction.

As a part of the thesis a survey of transfer learning methods for data streams will be performed. Next, novel methods developed for reference problems such as delay prediction will be proposed. Their implementation will be provided as a part of the thesis. A selection of existing and novel methods will be applied to reference data relying on open data such as data made public by the City of Warsaw and evaluated. Emphasis on classification tasks is assumed. The fact that the probability that an instance belongs to a class may change with time should be considered and possibly addressed when planning the development of novel methods.

Keywords: data stream, classification, transfer learning, concept drift

Streszczenie

Metody uczenia z przeniesieniem dedykowane dla strumieni danych inteligentnych miast

Głównym celem pracy jest analiza istniejących i opracowanie nowych metod uczenia z przeniesieniem, które można zastosować do strumieni danych inteligentnych miast. Problemy referencyjne do rozważenia obejmują problem przewidywania opóźnień dla pojazdów transportu publicznego. W takich przypadkach wiedza wynikająca z modeli opracowanych np. dla linii autobusowych może być ewentualnie przeniesiona do pokrewnego problemu przewidywania opóźnień tramwajów.

W ramach pracy zostanie przeprowadzony przegląd metod uczenia transferowego dla strumieni danych. Następnie zaproponowane zostaną nowe metody opracowane dla problemów referencyjnych, takich jak przewidywanie opóźnień. Ich implementacja zostanie przedstawiona jako część pracy. Wybrane istniejące i nowatorskie metody zostaną zastosowane do danych referencyjnych opartych na otwartych danych, takich jak dane upublicznione przez Miasto Stołeczne Warszawa i poddane ocenie. Zakłada się położenie nacisku na zadania klasyfikacji. Fakt, że prawdopodobieństwo przynależności instancji do klasy może zmieniać się w czasie, powinien być brany pod uwagę i ostatecznie uwzględniany przy planowaniu rozwoju nowych metod.

Słowa kluczowe: strumienie danych, klasyfikacja, uczenie z przeniesieniem, zmienna definicja pojęć

Contents

Introduction

In the age of ubiquitous IoT(internet of things) devices and mobile phones the amount of data available has been a major driving factor for the development of data science. Especially in recent years the importance of processing the available information has been realized. That is why the machine learning community has expanded significantly.

In most of the studies many assumptions about the data are made e.g. the data comes from a stationary generation process, that does not change in time. However this is not the case in many real world applications. Often the process is not stationary, may it be due to random occurrences and accidents, computing system errors, periodicity in characteristics of the process itself or aging effects in either the data collection devices or the environment. It is also the sheer amount of data that has to be dealt with when designing a processing or machine learning module. The domain that incorporates high Volume, Velocity and Variety data [17], that requires novel approaches to be effectively stored and processed is referred to as Big Data.

As the data is often arriving in sequential manner the need for it to be processed as a data stream arises. Streams are way to deal with data that is either too big to be loaded into machine memory at once or is not available as a single batch for various other reasons, such as simply its' generation process is continuous. This type of data is particularly prone to concept drifts meaning the distribution of data and its relation to the target variable changes over time. This problem has been extensively discussed in a survey [8].

Qin et al. [20] state in their definition of Internet of Things(IoT) that its main goal is to solve real world problems in essentially ever changing real world human environment. The idea of smart cities is to develop technologies that would improve the quality of life for ordinary people in growing urban areas. It uses IoT devices to monitor the city ecosystem. Smart city data may be coming from various type of infrastructure such as water supply, electricity grids, air quality and weather sensors, streets, railway systems including public transport. The type of conditions related to human activities are bound to change over time, so we expect the data to exhibit concept drifts.

As all of the data incoming from smart city datastreams is strongly connected to the rhythm of city operations questions arise: What potential benefits could be realized by sharing the data across different but interconnected aspects of smart cities? Could knowledge gained in one domain be transferable to other ones? Could model trained for tram delay predictions be able to adapt to bus delay prediction or vice versa? This thesis aims to answer these questions.

Naturally information about current position of public transport vehicles and trams is most commonly provided as a stream. Transfer learning for stationary environments has been already extensively discussed and many techniques have been developed [25]. However transfer learning in stream setting is still an uncharted territory for most data scientists. There have been a few proposed solutions especially for multiple source domains [24] [6]. This work elaborates on that topic, but focuses on the situation when there is a single source domain we want transfer the knowledge from.

1. Definitions

In the case of static datasets being used in offline learning often a strong assumption is made, that the data consists of independently and identically distributed (IID) variables.

Definition 1.1 (Independently and identically distributed variable). IID variables is data produced by a stationary process in no particular order. Each random variable comes from the same probability distribution and samples are mutually independent.

However in online learning, there are fundamental difference in data generation and acquisition process as the dataset is not fixed in size and the sample distribution may vary.

Definition 1.2 (Data stream). Data stream is a sequence of data sets $S = \langle S_1, 2, ... \rangle$ in which $S_t = \{(x_t^i, y_t^i)\}_{i=1}^{m_t} \sim_{iid} p_t(x, y)$ according to notation used in [18]. The length of the sequence may be potentially infinite. x^i constitutes a sample and y^i its label, $m_t > 0$ is a size of IID set given at time $t > 0$ so each arriving item has a timestamp. In particular at each time step m_t may be equal to 1 meaning the samples arrive one by one. It is different from static data in that it provides a notion of continuing time. Analyzing data streams is a key aspect of Big Data analytics. There is a potential to access more information, however to be able to analyze every sample one should take time constraints into consideration so that the machine learning module does not become overwhelmed.

This data may be employed for various tasks: classification, regression, clustering or frequent patter mining. The first two require the input data to be labelled which is called *supervised learning*. When the labels are not available (*unsupervised learning*) we can group the data into clusters or look for patterns. We will be considering only labelled data with an emphasis on classification task.

Definition 1.3 (Classification). Classification is a machine learning problem of assigning to sample x_i its label y_i, label being representative of one of discrete number of classes. In order to do that we train a classifier model on data samples. In traditional offline approach we are given access to any sample and its label at all times, however in stream setting the x_i samples are available only at timestep t_i and in theory we optimistically [2] assume that its respective label y_i is available at time $t_i + 1$. In practice labels can often be delayed in a case of verification latency, which we will revise later.

Definition 1.4 (Regression). Regression problem is similar to classification in that for each sample x_i we try to predict variable y_i. However case of regression y_i is a continuous variable, not a discrete label. In practice its usually impossible to reach perfectly accurate model, hence we aim to minimize prediction error.

For either of those problems there is a practical possibility to develop a model that is performing relatively well on e.g. two similar classification tasks. When the data from one task is available in large quantities and data for other task is a scarce resource, because the research may be to costly or even possibly dangerous(e.g. in radioactive area or a minefield), it might be a good idea to extensively train the model using the abundant dataset and later retrain it using the data available in small amount.

Definition 1.5 (Transfer learning). Let $D_S = \{X_S, p_S(x)\}$ be source domain, where X_S is a feature input space and $p_S(x)$ its marginal probability distribution. In that domain we are given a source task $T_S = \{Y_S, p_S(y|x)\}$ where Y_S is a label output space and $p_S(y|x)$ is a conditional probability of a label given input vector. This domain and task are represented through a dataset $S_S = \{(x_S^i, y_S^i)\}_{i=1}^N$. Consider also a target task T_T and target domain D_T represented by a dataset S_T where $D_S \neq D_T \vee T_S \neq T_T$. Transfer learning uses both datasets S_S and S_T to train a model for target task T_T. This target model is denoted as $f_T : X_T \to Y_T$. The goal is to obtain a model which is able to make more accurate predictions than a model based solely on S_T. We can distinct two types of TL:

- *Transductive transfer learning* approaches transferring knowledge between same tasks $T_S = T_T$ while only their respective domains differ $D_S \neq D_T$.
- *Inductive transfer learning* consists of transferring knowledge between different tasks $T_S \neq T_T$ while domain may or may nor differ.

The above definition is inspired by [6] however more general case with $N > 1$ source tasks is possible [18].

Above definition for offline transfer learning needs to be expanded for online learning by allowing multiple marginal probability distributions within one domain. In dynamic, non-stationary environment the stream is undergoing constant changes in distribution. This is one of the main reasons why standard transfer learning techniques cannot be applied in this setting. They are simply not designed to focus on performing on recently presented samples. As mentioned in [2] none of the assumptions of IID remain valid in this context as the data distribution is not fixed. Moreover consecutive samples are frequently correlated because of the generating process continuity.

Definition 1.6 (Concept drift). Lets consider joint probability distribution of machine learning problem $p_t(x, y) = p_t(x)p_t(y|x)$ at a time t. As defined in [18] if for any two time steps t and $t + \Delta$ the condition $p_t(x, y) \neq p_{t+\Delta}(x, y)$ is satisfied, then the concept drift has occurred. We can distinct two types of concept drifts:

- *Virtual concept drift* when only changes in $p(x)$ are involved so the x distribution changed but $p_t(y|x) = p_{t+\Delta}(y|x)$
- *Real concept drift* when the posteriori probability changes meaning there are t and $t + \Delta$ for which $p_t(y|x) \neq p_{t+\Delta}(y|x)$.

In traditional batch learning assessing model performance is done using train and test split meaning we divide the data prior to training and validation is done using unseen examples afterwards. As the data incoming in stream setting may suffer from concept drift the data distribution changes and having a static testing set, information about model performance on current distribution would not be known. While having potentially infinite stream of data cross validation may be computationally expensive as mentioned in [2]. Hence in stream setting the test-then-train approach is frequently used.

Definition 1.7 (Test-then-train evaluation). In this method as the data samples arrive they are first used for testing, to determine model performance at current time, after that the same data is used for training the model. This way model makes use of all of the data available for training, while it is measured using unseen examples.

In streaming applications labels of the data are often not available at the time data is being presented. The labels become available either after certain amount of time steps have passed or possibly never. Test-then-train approach is still applicable here, as in supervised learning training cannot proceed until labels are known, and when true labels are eventually available there is also a possibility of testing. However the prediction can be made at any point before that, which is discussed by Grzenda et al. [**delayed_label**].

Definition 1.8 (Verification latency). Verification latency occurs when for observation (x_i, y_i) predictor x_i is available at a timestamp t_1 and label y_i is available at time $t_2 = t_1 + \Delta : \Delta > 0$. The delay Δ may be specified or not known beforehand. It may be also varying with time. In an extreme case the labels are never available - this is called *Extreme verification latency* [22]. Stream learning solutions need to address this issue by managing the data until the labels arrive.

2. Literature overview

2.1. Stream mining approaches

Mining is a process of finding information, patterns, correlations and anomalies in data feeds. In [15] there are listed vital assumptions and practices of big data analytics field. In simplest case data representatives arrive one by one, each at a given time step and they need to be processed right then and there, because of the size and velocity of the stream. The main big data features were described as v-words such as *volume*, which is quantity of information, *variety* corresponds to different characteristics of data coming from different sources, it can be data form cameras, sensors, GPS devices, text, images and many more. Such data cannot be easily structured using schemes Lastly there is *velocity*, which stands for pace of information inflow and the need to process it in real time. Since Laney [17] introduced these terms in 2001 other v-words have been added upon that: Variability, Value, Validity and Veracity.

In stream mining we try to simultaneously minimize three values: error rate, memory cost and time cost. Usually each one comes at the expense of the other. When minimizing space, the data might be compressed and later decompressed but that takes time. The other option would be to sample data representatives so the time is saved, but the valuable information may be lost in the process. So there is always a trade-off but it is worth considering, when either time or resources are of the essence.

Albert Bifet at al. [2] in their book describe certain list of rules one should follow when designing stream miming pipelines. First of all every item has to be inspected at most once, at an iteration corresponding to a single point in time, which is called *one-pass learning*, whereas in an offline approach we are used to iterating over whole dataset multiple times. The module should take into consideration memory restrictions, and not forcefully store all of the data coming from potentially never ending information stream. We expect it to be able to give prediction at any point, even during the training process. Lastly the model should be able to adapt to changes in the data.

Bifet mentions smart city data streams as one of the key development fields of stream mining alongside business, technology and health. They describe data streams as "algorythmic abstraction to support real-time analytics". This abstraction however is very close to practical applications and implementations of information systems.

2.2. Non Stationary Environment

Data streams are created by processes usually originating from a constantly evolving environment. In particular, smart city data streams from public transportation are an example of such conditions. During the day, vehicle delays are affected by a numerous factors, ranging from weather conditions to human influence. We can not account for all of them, but still based on the data we have, information systems to best serve their purpose should be able to respond to these changes.

2.2.1. Reacting to concept drift

Ditzler et al. [4] have prepared a comprehensive survey describing the characteristics of the non-stationary environment, approaches to understanding them and ways to adapt to the conditions it offers. They include an extended definition and variations of concept drifts such as *permanent*, whose changes will be affecting the environment forever, and *transient* which will fade given a certain amount of time. There are two main ways to tackle the problem of non-stationarity in data streams - *active* or *passive*

The *active* approach involves actively monitoring the data for potential changes. To do so, it can use statistics gathered from observations such as either means or variances of data samples so the $p_t(x)$ or $p_t(y|x)$ from definition 1.6 by assuming increase in error rate is caused by changes in posteriori probability. One of the ways to determine the drift has occured is when a relative change in inferred mean, variance or error rate exceeds a given threshold. Whenever the drift is detected the previous model is discarded and a new one is created based on the current information. Tests for drift detection can be further divided into subcategories: Hypothesis Tests, Change-Point Methods, Sequential Hypothesis Tests, and Change Detection Tests. *Hypothesis Tests* operate on fixed number of samples and compare if the two distributions are the same according to predetermined degree of confidence. *Change-Point Methods* also operate of fixed sample size but their goal is to identify a single point in time when the drift occurred. *Sequential Hypothesis Tests* differ from Hypothesis Tests in that they are not restricted to specific set, but sequentially gather statistics until there is enough information to decide whether there has been concept change. *Change Detection Tests* also operate in sequential manner but they do not have to use hypothesis testing of the distributions.

Passive approaches to not seek for concept drift specifically. They simply accept the fact that changes in distributions may happen, and try their best to adopt to current conditions. By doing that they avoid the main hazards of active approach - false drift detection and failure to detect proper changes. Passive approaches can use decision trees and ensemble methods, which will be discussed in next section.

Both of these approaches share a common goal - to provide a model that performs well in real time. Which approach is better depends strictly on the problem being addressed. Intuitively, when we expect that there may be rapid changes in the distribution the active approach would probably be better but if there are repetitive concepts, passive approach is advisable. These two ideas however are not mutually exclusive and are often combined in state of the art models.

2.2.2. Characterizing concept drift

In fact, concept drift can not be described in binary terms(meaning it either occurs or not) and there is no strict way to determine the extent of changes that occur in the data generating process. The notion of slow or abrupt concept drift is subjective and quantitative measures are required to properly describe it. Webb at al. [23] proposed two basic measures which would help to standardize the way concept drifts are discussed. First is *magnitude*, in which the distance between previous and current distribution is calculated according to adopted measure. The other one, rather simple in concept but difficult to determine in practice is *drift duration* so the time elapsed between the beginning and the end of the drift. Based on these concepts, this work verifies other definitions, proposes and recalls terms such as *class drift*(real concept drift), *drift severity*(proportion of the domain of X for which $P(Y|X)$ changes), *covariate drift*(virtual concept drift), *minor/major drift*(with low/high magnitude), *blip drift*(sudden and short), *cyclical drift*(reoccurring, ordered drift) and others.

The same paper also examined how different algorithms perform depending on magnitude of the concept drift. The results of the study proved to be particularly revealing in case of *pure covariate drift*(only changes in $P(X)$). As it turns out, drift detection algorithms provide a minor benefit for classifiers in those conditions and neither them nor AccuracyWeightedEnsemble cannot fully recover from *pure covariate drift* and achieve pre-drift accuracy. Even more interesting were results achieved by tree-based classifiers, because as they revealed, the higher was the magnitude of the drift, the better final accuracy. It can be explained by the fact that during tree construction only certain set of feature combinations and splitting conditions is considered, hence the model will never reach 0 percent error rate. A significant concept drift encourages consideration of other tree divisions.

These results are particularly promising from a transfer learning perspective, as *virtual concept drift* is very similar conceptually to *transductive TL*. The fact that for some models the difference between domains can have a positive impact should encourage further research.

2.3. Classification and regression methods in data streams

Differing characteristics and varying conditions of non-stationary environment made most of the state of the art algorithms in the offline approach not directly transferable to data streams. For this reason, there was a need to develop new or vastly expand existing algorithms to be able to cope with different assumptions and conditions. Algorithms that would be able to adapt to concept drifts and process single data instances sequentially.

2.3.1. Hoeffding trees

Hoeffding trees were introduced by Domingos and Hulton [5] in 2000 but to this day they are a building block for many stream processing modules. The intuition behind them is similar to classic decision trees. However this solution does not require access to the whole dataset at once. It gathers statistics in the tree nodes and decides on splitting only if enough information has been accrued. It introduces attribute evaluation quality $'G'$ - a quantitative measure to determine amount of information stored in every attribute across samples in given leaf. Either information gain or Gini index may be used to asses G. If the difference between two highest G-s of different features is grater than $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$ the decision for a split is made according to attribute with the highest G. The n is a number of samples the leaf has already evaluated, $1-\delta$ is predefined probability of the split being the right one, R is a range of variable (for information gain is equal to logarithm of number of classes).

The authors also delivered a VFDT(Very Fast Decision Tree) algorithm basing on Hoeffding tree, but extending its possibilities by ability to limit G computation to every few timesteps, managing memory by limiting amount of leaves and pruning bad performing ones, detecting and ignoring attributes that algorithm determines as not informative and allowing ties when the G values are too close to each other.

2.3.2. Adaptive Random Forest

Adaptive Random Forest similarly to Hoeffding Tree is an example of utilizing and adapting well established algorithm [3] for batch processing into non-stationary setting. Even though the idea of ARF has been present for almost two decades now, the state of the art version of the algorithm was developed very recently(2017) by Gomes, Bifet et al. [11].

The algorithm 1 is very similar to traditional random forest [3], however a novelty is added to deal with concept drifts (lines 10 - 19). During training process algorithm monitors each tree and produces warnings(line 10). Whenever a warning is detected an empty backup tree is created.(line 11) After that, backup tree is trained in the background(line 18), but it does not participate in the forest voting process. When actual concept drift is detected(line 13), the main tree, which was performing poorly on recent data, is discarded and the backup tree replaces it(line 14).

The ARF has also a different sampling method. It has been proven that for large data batches number of time a sample is present in a tree adheres to $Poisson(\lambda = 1)$ distribution, so there is around 37% chance that it is not included in a tree. To leverage that by increasing resampling authors assign weight to each data sample according to $Poisson(\lambda = 6)$, and process each weighted sample one time for each tree, so there is no replacement.

The trees used in Adaptive Random Forest were developed based on ideas introduced in Hoeffding tree [5], however it also includes sever key differences. The split decision is not based on evaluation quality metric, but simply when the number of instances seen in a leaf exceeds a threshold. It results in deeper trees, and may result in overfitting. However in the scale of whole ensemble of trees this is not a drawback due to variance reduction that is a feature of ensemble, and it gives for more specialized trees. To add to that, the idea of tree pruning from VFDT is also forsaken, because the goal of the algorithm is not to find a single optimal tree and the obsolete knowledge is dispatched together with an entire tree. There is also one major change, that also differs Breiman's Random Forest[3] from Bagging, a random subset of features is selected in each tree node, and the split decision may consider only features from the subset.

2.3.3. Streaming random patches

A non-stationary setting may contain specific situations where you will need to rebuild the model sequentially using a relatively small amount of resources. Hoeffding trees accept data during construction, but do not minimize error until the partitioning condition is satisfied, as their partitioning is based on the quantity, not the quality of information. This results in reducing variance, which makes this structure a stable base model. On the other hand, one of the main advantages of ensembles is the reduction of variance, so when using Hoeffding trees, the full potential of this attribute is not realized.

For large ensembles it is encouraged to maintain high diversity among the base learners, to be able to maintain uncorrelated prediction errors. To create two diverse Hoeffding trees the

Algorithm 1 Adaptive random forest **Symbols:** n: total number of trees in ensemble; m:number of features take into accound for each split;B: set of backup trees; S: data stream; L: loss function; δ_w: warning threshold; δ_d: drift threshold

```
1: function A(d)aptiveRandomForest(n)
2:     $T \leftarrow CreateTrees(n, m)$
3:     $B \leftarrow \emptyset$
4:     $s \leftarrow S.next()$
5:     while $s$ do
6:         for $t \in T$ do
7:             $\hat{y} \leftarrow t.Predict(s.x)$
8:             $t.UpdateWeight(L(\hat{y}, s.y))$
9:             $t.Train(s.x, s.y)$
10:            if $t.WarningDetected(\delta_w)$ then
11:                $B(t) \leftarrow CreateTree(n, m)$
12:            end if
13:            if $t.DriftDetected(\delta_d)$ then
14:                $t \leftarrow B(t)$
15:            end if
16:        end for
17:        for $b \in B$ do
18:            $b.Train(s.x, s.y)$
19:        end for
20:        $s \leftarrow S.next()$
21:    end while
```

training datasets would need to differ on many items, because of their stability. Therefore batch ensembles use simpler structures in which a small change in data results in a different model.

Streaming random patches [9] counteracts these nuances of Hoeffding trees. SRP for each tree considers only a subset of features and a subset of samples - patches. This introduces diversity among base learners, as they are still stable within the subset of features, but do not have access to other variables. Similarly to ARF this algorithm uses drift detection and backup trees, but each new tree receives a new set of random features since the previous ones were not informative.

Sub-sampling in this way carries an important tradeoff, a lot of the data not to be used to train the model. Authors' study shows that because the algorithm introduce diversity among Hoeffding trees its performance is on pair with state of the art algorithm, which is ARF. Analyzing lesser data also results in higher computational capabilities.

Interestingly, the approach with feature subset selection resembles dropout, a technique used in neural networks, which also improves accuracy, despite disregarding part of the information.

2.3.4. Error Intersection Approach

As non-stationary data mining is still a developing field and there new approaches for handling concept drift are proposed every year. Most of these research endeavours however are considering classification task. This is also why most drift detection algorithms are based on aggregating binary error values. Some research problems, for example studies of social trends, are inherently a regression tasks. In this case a commonly used error metric is mean squared error.

Another issue worth considering is the use of stream mining methods in production systems. Despite the fact that tree ensemble models allow for exceptional flexibility and maintain possibly accurate predictions, it is difficult to precisely analyze the processes that take place in them, which makes them slightly unpredictable.

Error Intersection Approach(EIA) [1] proposes a completely different approach than the algorithms described so far. It does not use ensembles, but only two different models. The first one is a complex model designed to analyze data in a long-term context, the other is a simple model used to adapt to current changes in environment. Complex model is a neural network similar to networks used in static conditions, however it is periodically retrained for in not to become obsolete. Authors did not deliberately state what is used in place of simple model only that it may be any model used for timeseries forecasting.

The way the two models are connected is interesting, because the algorithm monitors the error of each model at all the time. The final prediction is entirely based on the currently more accurate model. When the error curves cross(Error Intersection), the dominant model is changed.

This approach may seem greedy, but as the authors have shown, attempting to balance both models in an ensemble results in worse accuracy due to the fact that the simple model is too heavily biased towards recent trends for most of the time.

What is worth noting, is the fact that this algorithm finds possibility of application of neural network in a stream setting. Periodic retraining works on an identical principle to transfer learning in neural networks, suggesting that multilayer perceptron may find applications in this research. Another study [12] involved architecture in which two types of models were utilized, stream model for the online data stream and multilayer perceptron for batch learning on initial part of a data stream.

2.4. Transfer Learning in Non Stationary Environment

Transfer learning has so far been considered mainly in static terms. Specifically, it has been used in various types of neural networks, and there is available a wide range of pre-trained networks ranging from convolutional networks for image analysis, through Generative Adversarial Networks to natural language processing in BERT(often used as source model for various NLP solutions). Neural networks, despite the fact that they operate in a static environment, allow for sequential processing of data and incremental model change.

It is somewhat more difficult to apply transfer learning to statistical models like Random Forest because they rely heavily on partitioning the dataset or extracting statistics in advance from all of the items. It is non-trivial to determine how data from two different domains could be combined in this situation. In this respect, a non stationary environment seems more natural for transfer learning application. In stream processing, we have statistical models capable of adaptation as they must respond to concept drift. So stream mining methods should be able to take advantage of the potential benefits that TL offers, especially when at the start of data inflow in target domain there is already a tailored model that can solve problems in a similar domain, so initial and asymptotic accuracy will benefit.

2.4.1. Online Transfer Learning

Wen et al. [24] in 2019 proposed number of novelties to existing approaches as well as employing them in the problem of transfer learning. The first enhancement OTL brings is change to VFDT [5] by enabling it to handle splits through numerical attributes via optimal dichotomy. Secondly, it also tackles a problem mentioned also in [9] of Hoeffding bound being satisfied too slowly for dynamic conditions. It does so by preserving a number of previous data items in each leaf according to sliding window. In addition, VFDT-D (name for proposed structure) allows to obtain the posteriori probability of class membership as $P(y = k|x) = \frac{n_l^k}{n_l}$

The proposed Online Transfer Learning framework uses separate models for the input domain and the target domain. Posteriori probability assessed by pretrained model is combined with the output of target domain classifier, according to formula:

$$\hat{y}_t = sgn(\alpha_{1,t}\Pi(v^\top x_t) + \alpha_{2,t}\Pi(w_t^\top x_t) - \frac{1}{2})$$

where v and w_t(index t indicates that function is constantly evolving) are prediction functions of respectively source task and target task, Π is projection function and $\alpha_{1,t}$ and $\alpha_{2,t}$ weighting parameters at the time t.

DOTL is an implementation of this framework using VFDT-D ensembles presented in paper. There is also possibility of multiple source domains (DMOTL) in which the only source domain taken into account for prediction is chosen according local similarity (LC) of its domain to current sample.

2.4.2. Melanie

It remains to be considered that there may be a situations where, similar to the target domain, the source domain is still receiving the data, so that the model can simultaneously learn from both of them.

Minku at al. [6] proposed using multiple ensembles - each for every source domain, one for target domain and additional models for every detected concept drift in target domain. This will result in a ensemble of ensembles where the voting is also performed between the ensembles. Since in that case data from different domains is received, weight corresponding to each ensemble is only updated while analyzing samples from target domain. The weights are updated based on normalized decaying average of $(1 - L(h^j(x^t)))$ where L is the loss function of model prediction h^j for sample x^t. Therefore the better the prediction, the smaller the loss, the higher the weight. It is also worth mentioning that only most recent target model is being trained on data from target domain D_T, while the source models are trained on instances from their respective domains $\langle D_{S1}, D_{S2}, ..D_{Sn}\rangle$.

Melanie uses Online Bagging and Online Boosting as the underlying ensembles due to their popularity. Interestingly, the intuition behind the use of these methods was that each of the underlying ensembles is defined in terms of subdomains. It is therefore recommended for these sub-domains to be possibly diverse. But as mentioned earlier, bagging, boosting and even random forest allow for only limited diversification. The aforementioned idea of Random Patches [9] seems to be applicable here and may be considered later in this work. SRP by diluting the data, reduces the computational cost, which can help with the complex architecture of a double-layered, constantly growing ensemble.

2.4.3. Multi-source Mapping Transfer Learning for Non-Stationary Environments

Minku et al. [7] inspired by the ideas used in Melanie have decided to expand it even further by allowing a possibility for new model to be produced each time a concept drift occurs in source domain aswell. The final result is a set of modules, each for a separate domain, inside the module there are separate models for sequentially occurring concepts within the domain and each of these concept models is also an ensemble.

With an increasing number of different domains and concepts, the problem arises as to whether the target domain data is applicable in each base model. Since it is known that each of them was created based on specific knowledge, there is a desire to use this knowledge to the best of model abilities. For this purpose MARLINE proposes a mapping operation in which a translation is done from the target domain to the other domains by determining the centroids for each class within domain and then simply transforming the point between domains according to centroid positions. Centroid is calculated for each class as a decaying average of incoming data values. Then the connecting vectors of every two classes within one concept are determined and the transformation of corresponding vectors between different concepts is a mapping function for data points. E.g. in binary case $V^S_{1,2} = M \cdot V^T_{1,2}$, where $V^S_{1,2}$ is vector connecting two classes centroids in source concept, $V^T_{1,2}$ in target concept, and M is the mapping. The intuition behind this approach is that the spread of class centroids is at least slightly similar across different contexts. Inter-class vectors may be viewed as a reference of that spread so the mapping is used to site the point in different context.

2.5. Smart City data mining

As mentioned before Smart City data is in large part produced by IoT devices. Qin et al. [20] provide an extensive overview on potential smart city applications and engineering techniques that help facilitate infrastructure for mining this type of data. It mentions potential areas of benefit among which are city traffic, management of uniformed services, energy production, security and virtually every imaginable problem connected to resource management. As can be inferred from the paper - although applications of smart city data mining are fairly intuitive, the engineering side of this problem still needs an extensive research and resources.

Due to the size of urban infrastructure, the number of users and the openness of public institutions and enterprises, urban areas are among the main producers of Big Data. Aside from the practical applications mentioned earlier, many of them are also used as a benchmark to assess performance of state of the art solutions. E.g. ARF [11] uses as a reference data from Airline traffic and electricity demand. [1] has been prepared with the Taxi demand in New York City in mind. Authors also suggested that designed model for Taxi demand should be applicable for data from other cities. MARLINE [7] is using London and Washington D.C. bike sharing data for knowledge transfer.

Data incoming from the smart city sensors, which are often of varying quality needs an appropriate infrastructure to be managed and stored effectively. Grzenda et al. [12] propose data acquisition and management system. It is implemented using various frameworks from Apache Big Data management software such as:

- Apache Flume is used to fetch the data from public API. It is able to process large amounts of data efficiently and forward it to downstream modules.
- Apache Kafka is a distributed event streaming platform that allows for high availability, scalability and fault-tolerant persistent storage of data. It is supported by vast majority of Big Data analytical tools.
- Apache Flink with its real-time stream processing capabilities is used for preprocessing and alignment of data that may be provided by vendor in unordered and unsystematic manner. A combination of data feeds from kafka or other sources is performed here, to determine labels and produce high quality data for learners.
- Records are stored in Apache Hadoop with its own Hadoop Distributed File System (HDFS) which key assumptions and goals are:
  - **Fault tolerance**
    When one of the storage devices fails, the data will be accessible from another device.
  - **Streaming data access**
    Manage whole system files continuously. Constant feed of data combined including real-time information. Memory access is focused on high volume throughput rather than on low latency.
  - **Large dataset**
    Large block size (default 64MB), which assists storing huge amounts of data.
  - **Simple coherency model**
    File should not be changed when it has been created and closed, however data can be read multiple times.

The architecture follows a three-layer lambda design, of which the *speed layer* is Flink with its real-time processing, the *batch layer* is data storage in HDFS.

Smart cities are an ideal example for the application of transfer learning. Cities can be seen as a living organism in which no two processes are completely independent of each other. An ultimate goal that researches are striving for is an artificial general intelligence system that is able to embrace the complexity of the urban ecosystem.

2.6. Stream mining frameworks

2.6.1. Massive Online Analytics

More commonly refferred to as MOA [2], Massive Online Analytics is probably the most popular machine learning framework for non-stationary environment. It is implemented in java and for this reason can be easily integrated into Apache Big Data mining software. It is often used as a place to share novel algorithm implementations by the research community. It includes implementation of previously described Hoeffding trees [5], Adaptive Random Forest [11] and many others. Extensive research on concept drift [23](paper includes detailed characterization of each drifting stream configuration) were performed using MOA benchmarks as they allow to simulate concept drifts on artificial data.

2.6.2. Scikit mutiflow

It is an fully open source project widely supported by the community. It is written in python which has been emerging in recent years as one of the dominant programming language among data scientists. It supports various types of stream mining settings and utilities such as supervised and unsupervised learning, concept drift detection, various types of output (single/multiple and classification or regression), artificial stream generation and drift detection. It includes implementation of stream ML models including variations of Hoeffding trees, ARF, Oza Bagging, Learn++, Streaming Random Patches and others. In contrast to MOA it is distributed in form of a toolkit rather than software. Scikit-multiflow recently merged with *creme* into project *River*

2.6.3. Transfer Learning resources

Unfortunately it is hard to locate any open source solution for transfer learning in streaming setting. Most of the code repositories are either no longer available or are accessible directly through the authors. However datasets used as a reference in the algorithms reviewed in it work is easily accessible. For instance [24] uses data generated in MOA framework and Nursery, Intrusion detection and Forest Covertype datasets from UCI(University of California, Irvine). In [6] and [7] bike sharing data is taken from *kaggle* platform and UCI. Grzenda et al.[12] are exploring the same Warsaw Public Transport API that will be under the scope of this research.

3. Methodology

3.1. Data for analysis

Aggregated data from the urban transportation of the city of Warsaw have a clear translation to the real needs of the residents. In addition, they are of very good quality and are suitable for stream model studies, as they have informative parameters and are both comprehensive and concise. According to the survey conducted by Gomes, Grzenda et al.[10] in practice, streaming data of such type i.e. coming from smart city domain are rare. Studies on stream mining methods rely on a limited number of data streams, not coming from smart city domain. There is an abundance of delayed unlabeled data, or data that is only partially labelled, an example of semi-supervised learning, which is of increasing interest and is being more widely studied in the batch environment, however the availability of viable resources and analysis in the streaming environment is largely just developing right now. During the preparation of this work, efforts were made to explore and obtain urban transportation data for other cities, but the availability and quality of the data is far from the transport data of the city of Warsaw and adapting such data would require a great deal of effort which is beyond the scope of this work. For the reasons discussed above in the remainder of this work we will focus on location and delay data streams of public transport vehicles in Warsaw.

3.2. Methodology overview

The overall methodology depicted on Fig.3.1 of preparing the solution consists of many consecutive and also reoccurring steps. First vital part was the data analysis which allows to determine ways to add labels to existing data and determine variables which will be of use in considered models. Equally important is the identification and selection of available transfer learning methods and state-of-the-art methods that will serve as a baseline. Also based on these models, modifications are made and new applications are proposed.

In the next step available data is processed to extract vehicle status information from the

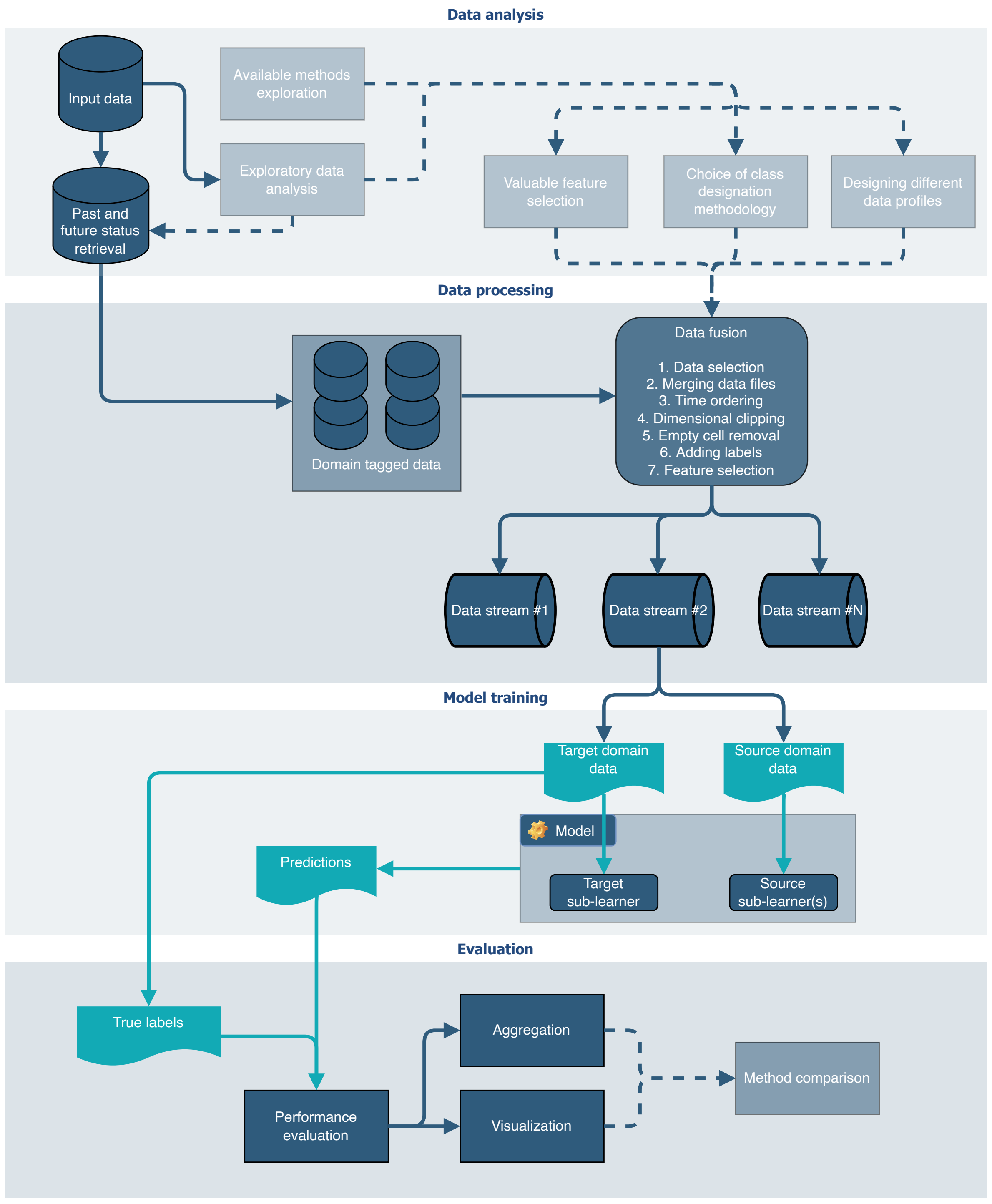

Figure 3.1: Methodology schema

future and the past. Based on this, the information context can be expanded and labels can be determined. The pre-prepared data is then combined into various profiles as in Sect.[4.3] to be tested later. During this process, the physical region of the trip data is truncated as described in Sect.[4.2], and the data is refined and organized into streams.

Within each stream every sample of data is equipped with domain identifier which is used by transfer learning models to accordingly determine the way data is processed and passed into specific sub-learners. Testing of various models analyzed and prepared during this study will take place here. Before training a model on given sample from target domain a prediction is made which will be later compared with true labels and used for evaluation through graphical techniques (plots) or arranged into tables.

3.3. Proposed models

3.3.1. Transferable data stream processing

Data in a streaming setting can come in a variety of ways. In particular, in the case of multiple domains, we want to have a tool to distinguish, separate, and pass the data to the appropriate model or sub-model, as in Melanie's case. For this purpose, the different domains in the datasets have been marked with an additional parameter. The target domain number is always 0 and the source domain numbers are consecutive natural numbers greater than zero.

3.3.2. Melanie with different configurations

In their work, authors of Melanie [6] considered only online Bagging and Boosting [19] as potential base learners for Melanie. The innate property of Melanie, which adds a new ensemble whenever a concept drift is detected, is supposed to be able to handle changes in distribution. However, it is worth exploring the benefits of using a more dynamic base model such as SRP [9] which also responds to concept drift. Additionally patches being subsets of features in SRP give for base models with higher variance, which is desirable when working with ensembles.

Let us also propose a scenario where we block Melanie's concept drift detection and let Streaming Random Patches handle it. This will result in no new SRP baseline models being created when the stream changes, only one baseline model will remain for each domain, but one that is able to respond to changes, unlike Bagging or Boosting. This will result in a more compact model and it will be tested what effect does the size reduction have on overall performance.

In Alg.[2] we propose this simplified version of Melanie. It is in essence very similar algorithm except for the concept drift response that happens implicitly in SRP.

Algorithm 2 sMelanie

Input: $(x_i^{(t)}, y_i^{(t)}) \in D_i, i \in \{S_1, S_2, ..., S_n, T\}$: training instances at time t in their respective domains i

Data: $M \subset \{S_1, ..., T\}$: set of already seen sources or target, initialised with $\emptyset$, $H = \{H_i : i \in M\}$: domain specific subclassifiers, initialised with $\emptyset$, $W = \{w_i : i \in M\}$: voting weight vectors for each set of classifiers, initialised with $\emptyset$

```
1:  function sMelanie($x_i^{(t)}, y_i^{(t)}$)
2:      if $i \notin M$ then
3:          Initialise new model $H_i$ for domain $i$
4:          $M \leftarrow M \cup \{i\}$
5:          $H \leftarrow H \cup \{H_i\}$
6:          $W \leftarrow W \cup \{w_i = 0\}$
7:      end if
8:      $H_i \leftarrow TrainOnInstance(x_i^{(t)}, y_i^{(t)}, H_i)$
9:      if $i = T$ then
10:         $W \leftarrow RecalculateWeights(H, W)$
11:     end if
```

Source code for Melanie [14] was prepared using MOA environment. However as SRP has been proposed later (Nov 2019) so to make the two models compatible a fair bit of engineering work had to be done, which also served as a starting point for further development of Melanie-like experimental approaches. Implementations of these methods can be found in the repository [16] and are described in the appendix[7].

3.3.3. MARLINE with SRP

Intuition behind this approach is the same as with Melanie. Source code for Marline was also prepared using MOA framework and is available at [13]. MARLINE was prepared after the publication of SRP, however application of SRP as base model had not been tested, and also required development effort made in this work. Implementation provided by method authors allows for only binary classification.

The way we use Streaming Random Patches in MARLINE is analogous to the way we handle Melanie. We simply change the type of model used as the base model. The other mechanisms remain the same. Domain mapping using centroids is still used as it was for MARLINE with static ensembles.

3.3.4. Melanie+

It is possible that a priori distributions for source and target domain remain similar, however the posteriori probability of $p(y)$ changes, so the prediction of model trained on source domain are not directly transferable to target domain. Melanie+ is an extension of Melanie in which additional attribute is added when training ensembles for target domain. This attribute is the class selected as a result of voting of ensembles trained for source domains. The Alg.[3] provides implementation details of this method. It is very similar to traditional Melanie [6] with addition of the lines 12-14 where additional parameter x_i^+ is created and appended to the input vector data. This is a novelty proposed in our work. As a result an argument vector $x_i^{(t)}$ is longer by 1 item in case of $i = T$, so the base learners for target domain take in one additional argument. Ultimately, the model for the target domain is trained to be able to interpret not only the current data but also the result of the source model predictions.

3.3.5. BLAST

In case knowledge transfer is of benefit only in certain periods of learning and evaluation process, it is possible to combine transfer-learning and non-transfer-learning approaches and use the one that is performing best on recent samples to make the predictions. This can be done using Blast[21] which has also been implemented as a part of MOA package.

But to make that for relevant models that do not support transfer learning it is needed to ensure that they do not get samples from source domains. Therefore, as part of our work, a modification of the BLAST model was prepared, the details of which are described in Alg.[4]. This required creating a custom model based on the currently available implementation in MOA. This model includes a split in set of models on the ones that learn on all domains and those that only learn on the target domain. Data samples are forwarded to either all models, when the data is from target domain or transfer learning models, when the data comes from source domain. Domain is indicated as one of the parameters of input vector.

Algorithm 3 Melanie+

Input: $(x_i^{(t)}, y_i^{(t)}) \in D_i, i \in \{S_1, S_2, ..., S_n, T\}$: training instances at time t in their respective domains i

Data: $M \subset \{S_1, ..., T\}$: set of already seen sources or target, initialised with $\emptyset$, $H = \{H_i : i \in M\}$: domain specific set of classifiers, initialised with $\emptyset$, $W = \{w_i : i \in M\}$: voting weight vectors for each set of classifiers, initialised with $\emptyset$

```
1:  function Melanie+($x_i^{(t)}, y_i^{(t)}$)
2:      if $i \notin M$ then
3:          $H_i \leftarrow \{\mathrm{H}_i^1\}$ (Initialise new learning set with one model initially for domain $i$)
4:          $M \leftarrow M \cup \{i\}$
5:          $W \leftarrow W \cup \{w_i = 0\}$
6:          $J_i = 1$ (number of learning ensembles for i)
7:      end if
8:      if $WarningDetected(x_i^{(t)}, y_i^{(t)})$ then
9:          Create and start training background learner $H_i^{J_i+1}$ for domain $i$
10:     end if
11:     if $DriftDetected(x_i^{(t)}, y_i^{(t)})$ then
12:         $H_i \leftarrow \mathrm{H}_i \cup \{H_i^{J_i+1}\}$
13:         $w_i = (w_i^1, w_i^2, ..., w_i^{J_i}, 0)$ (append 0 as new classifier weight)
14:         $J_i = J_i + 1$
15:     end if
16:     if $i = T$ then
17:         $x_i^+ \leftarrow \arg\max_l [y_l : y_l \in \sum_{j \neq T,k} H_j^k(x_i^{(t)})]$ (select class with highest cumulative probability from learners trained on source)
18:         $x_i^{(t)} \leftarrow [\mathrm{x}_i^{1(t)}, {}_i^{2(t)}, ...x_i^{n(t)}, x_i^+]$, append $x_i^+$ to vector $x_i^{(t)}$
19:     end if
20:     $H_i^{J_i+1} \leftarrow TrainOnInstance(x_i^{(t)}, y_i^{(t)}, H_i^{J_i+1})$
21:     if $i = T$ then
22:         $W \leftarrow RecalculateWeights(H, W)$, described in detail in Sect.[2.4.2]
23:     end if
```

Algorithm 4 TransferBLAST

Input: $(x_i^{(t)}, y_i^{(t)}) \in D_i, i \in \{S_1, S_2, ..., S_n, T\}$: training instances at time t in their respective domains i

Data: M: set of all candidate models, $M_t \in M$: set of transfer learning candidate models, C: current best classifier

```
1: function TransferBLAST($x_i^{(t)}, y_i^{(t)}$)
2:     if $i = T$ then
3:         for m in $M$ do
4:             $AddEvaluation(x_i^{(t)}, y_i^{(t)}, m)$
5:             m ←TrainOnInstance(x$_i^{(t)}, y_i^{(t)}, m$)
6:         end for
7:         $C \leftarrow SelectBestModel(M)$
8:     else
9:         for m in $M_t$ do
10:            m ←TrainOnInstance(x$_i^{(t)}, y_i^{(t)}, m$)
11:        end for
12:    end if
```

4. Experiment setup

4.1. Data analysis

The smart city data in question was obtained as a part of results from VaVeL project. Available data had been collected every day for multiple months in years 2017 and 2018. Raw data available from ZTM API offers only information about location of particular vehicle. Under the VaVeL project it has been paired with schedule data and additional variables have been added, such as:

- **Time of data collection**
- **Delay** - heuristic estimation on the basis of delay at most recent stop and time passed since leaving most recent stop
- **Next stop position**
- **Next stop distance**
- **Next stop timetable visit time**

This kind of information, as for an external observer, is very valuable for machine learning models, because it broadens the information context and allows for greater accuracy.

4.2. Data pre-processing

As the data is acquired from physical GPS devices it is prone to errors. Additionally network of public transport covers large area and gets more dispersed on the outer regions. The further from city center and urbanized areas the lesser the number of reads. On the contrary in the city center the network of communication is relatively dense and represented by multitude of reads for both trams and buses. For the above reasons in the following research smart city data has been limited to an area bounded by rectangle located in the center of longitude and latitude distributions. The width and height of the rectangle is set to 4 kilometres and inferred using following formulae.

$$1^{\circ lat} = 110.574km$$

$$1^{\circ lon} = 111.320km * cos(latitude)$$

Primary focus of this research is application of transfer learning techniques for classification task. Models task will be to estimate ahead of time whether vehicle will be delayed, hence the data in each record needs to be paired with data from the future, which is possible because in this research entire dataset is available in advance. In case of real world implementation of this solution data pipeline would need to take into account verification latency phenomenon discussed in section [1.8].

Initially prepared data labels was vehicle delay in full minutes, however for the most part, results of initial tests remained inconclusive, hence number of labels needed to be limited.

Upon expanding the scope of tested models to include MARLINE, the number of classes was reduced to two because the current MARLINE implementation only allows binary classification. The final character labels whether the vehicle arrives late or not. All previous tests had to be performed again to get the full overview of assessed models and their performance.

The final form of the data includes the following variables:

1. **Domain**

   Index of domain data is from. **0** is reserved for target domain.

2. **Current time**

3. **Current delay**

4. **Current longitude**

5. **Current latitude**

6. **Nearest stop longitude**

7. **Nearest stop latitude**

8. **Next stop distance**

9. **Speed**

10. **Class**

    The model has to predict whether the vehicle is late or not. This is a case of binary classification.

- Late (delay > 60 seconds)
- On time (otherwise)

More specifically it takes one of two values, 1 for a vehicle that is late and 0 for a vehicle arriving on time. Unlike the other parameters, which determine the status at curent time t, **Class** is determined by the delay at later time $t + dt$.

4.3. Evaluation datasets

Data from streams can be aggregated and applied in a variety of ways. It is vital to identify the method of processing data that would potentially bring best performance. In all of below cases source domain is the data collected from buses, and target domain is train data, as there is usually few times more data (depending on time of day) for buses and usually knowledge transfer is performed from richer domain to the one with lesser representation. The exception is IND dataset described in Sect.[4.3.4], in which the target domain is single bus line. In each data variant target domain data is collected within the same time frame - entire 24 hours of Friday 07.09.2018.

4.3.1. Target domain multiple lines (TAR)

This data consists solely of target data and is used as a baseline to asses the potential benefits of using transfer learning versus state of the art models that are not eligible for transfer learning. The data is collected on the span of one day from multiple tram lines (all tram lines in the clipped area). Data from all lines is presented alternately, according to the reading time from the GPS device. It consists only of target domain so transfer learning is not involved.

4.3.2. Separated domains of multiple lines (SEP)

In this data, the source domain data completely precedes the target domain data and is taken from Thursday 06.09.201. A situation arises where a model trained solely on the source domain from one day is adapted for the target domain next day. This is similar to transfer learning techniques in typical offline setting as the model is trying to reuse prior knowledge. Target data is the same as for TAR data. The source data consists of one domain, which is all the bus lines occurring interchangeably.

4.3.3. Alternating domains of multiple lines (ALT)

It is realistic to have data coming in simultaneously from the source and destination domains, which is particularly probable in stream setting. In aforementioned models Melanie[6] and MARLINE[7] source domain models learn alongside target domain models and are able to update to current trends from all domain simultaneously. The data is analyzed on the span of one day - 07.09.2018. Target domain data is the same as for TAR data. Source data is constructed analogously to SEP, however it is taken from the same day as target data.

4.3.4. Individual target line (IND)

The data in the target domain is selected for a single bus line - 157, and thus a more specialized model may be produced. Source domain still includes overall bus traffic, however with an exception of this single line. In this case target domain has significantly less instances than source domain, however the source and target should be more related as they come from the same traffic type - buses.

4.3.5. Data with historic context (*HIST)

Variants of previous datasets enriched with historical context information about location, estimated delay and speed from 5 minutes in the past. E.g. while at 9:00 a.m. a prediction is made to asses delay at 9:05 a.m., in addition to status at 9:00 a.m. there is also information about the status of the vehicle at 8:55 a.m. The intuition behind this approach is that changes in delay may form a trend. Additional variables present in both domains are:

1. **Previous delay**
2. **Previous longitude**
3. **Previous latitude**
4. **Previous next stop distance**
5. **Previous speed**

4.3.6. Data from longer time span (LARGE)

In order to better evaluate the solution, larger data sets have been prepared. In this case, the data in the source domains is from 4 days - Monday 03.09.2018 to Thursday 06.09.2018, and the target domain is from Friday 07.09.2018. This significantly extends the pre-training period, but allows for a more accurate comparison of solutions and potentially better model accuracy due to the larger amount of source data. Target domain remains the same as for TAR data.

4.3.7. Data for long term prediction (LON)

Initial data were generated for delay prediction for 5 minutes ahead. Potentially, these can often be small changes from the current delay. Therefore, a scenario will be tested where we want to determine a delay 10 minutes ahead. Naturally to determine the delay 10 minutes ahead the information has to be extracted 10 minutes from the future, so the last 10 minutes of each run of a vehicle are not included in the training streams as there are no possibility of obtaining labels. For this reason for this type of data there is less samples than for the 5 minute predictions both for source and target domains. Similarly, in the HIST-LON variant, the context-expanding data comes from 10 rather than 5 minutes before.

4.3.8. Synthetic data

Artificially generated data from normal distribution with two real value predictors and binary output, the same that was used by authors of Melanie[6] in their evaluation of Melanie method. For each artificial dataset the entire source data is presented before target domain data. In the middle of target instances stream abrupt (ABR) concept drift was simulated, meaning instant change in distribution. For incremental concept drift (INC) distribution changes gradually. Data sets without concept drift(NO) and with reocurring concept drift(REO) were also included in the tests. For REO data source was entirely processed before target as well. Various sizes were tested for target domain in which each class in each concept contains either 50, 500 or 5000 samples, whereas in each source domain each class has always 5000 samples so each source domain has 10000 samples total.

Dataset	Number of source intances	No. target instances
TAR	0	217744
SEP	291595	217744
ALT	260521	217744
TAR-IND	0	14309
SEP-IND	277330	14309
ALT-IND	246212	14309
LARGE-HIST-SEP	825815	205348
HIST-TAR	0	205348
HIST-ALT	196031	205348
HIST-SEP	232609	205348
HIST-SEP-IND	218344	14309
TAR-LON	0	216225
HIST-ALT-LON	172718	202881
HIST-TAR-LON	0	202881
HIST-ALT-LON	196031	205348
ALT-LON	251355	216225
SEP-LON	282845	216225
LARGE-HIST-SEP-LON	720790	202881
SEP-IND-LON	268595	14295

Table 4.1: Number of instances for different variants of smart city datasets

Table [??] provides an overview of the produced datasets in terms of the number of instances with distinctive domains. It can be seen that when limited to the city center only, the number of readings from buses and trams is very similar. In the case of IND-type data, the source domain is much more abundant compared to target. For 10-minute prediction there is in fact slightly less samples than for 5-minute variants.

4.4. Compared models

In the course of development, different types of models, both transfer-agnostic and supporting transfer learning (including those proposed in this work), were tested to determine the best approach for processing the prepared data variants. Below is a complete list of these models:

- **Majority**

Majority classifier which always chooses class most represented in recent 1000 samples in target domain. This serves as an absolute baseline classifier.

- **SRP**
  Streaming Random Patches
- **Melanie Boost**
  Melanie with OzaBoost ensemble as base learner.
- **Melanie SRP**
  Melanie with SRP ensemble as base learner
- **sMelanie SRP**
  Simplified version of Melanie without concept drift detection with SRP ensemble as base learner.
- **MARLINE SRP**
  MARLINE with SRP ensemble as base learner.
- **Melanie+SRP**
  Melanie+ with SRP ensemble as base learner.
- **BLAST**
  BLAST (**b**est **last**) using SRP, Melanie SRP and Melanie+ SRP as candidate models.

4.5. Performance metrics

The accuracy of assessed classifiers is measured solely on target examples, as performance on only target domain is the goal of the model. Performance on the source domain is not being optimized. As the data is coming as a stream from real-world environment it is expected for concept drift to occur. For that reason neither of potential performance measures reaches asymptotic accuracy as if it were the case in static data. Hence, it is needed to examine accuracy throughout the entire learning process using sliding window of width 1000. Additionally to take into account potential model bias, intra-rater-reliability is measured using kappa statistic on the same window, to see how informative for a give model are features presented in the input vector.

5. Results

In the following section results of testing on synthetic data and variants of city transport dataset will be presented and discussed. The aim of following tests is to determine the most appropriate way to prepare data for its use in stream learning models.

5.1. Transfer learning in stream setting

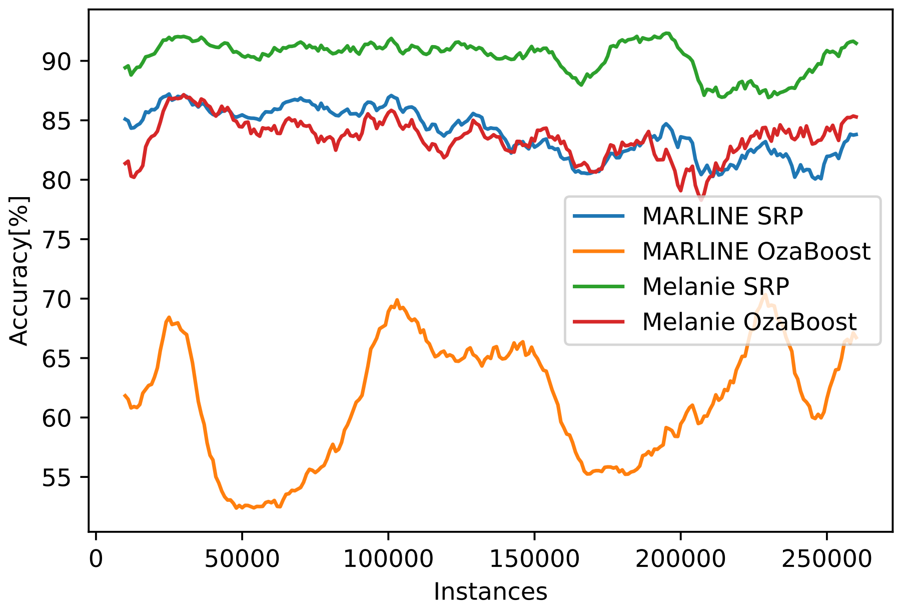

Figure 5.1: SRP and OzaBoost base comparison on SEP data set

Figure 5.1 shows accuracy of different stream learning models on SEP data set. Accuracy is calculated only for target domain and the number of instances represents instances seen in target domain. Both MARLINE and Melanie using SRP as a base learner achieve much better accuracy than their OzaBoost counterparts across the entire training process. This confirms the conjecture that for data with constantly changing distribution models with higher variance are an improvement. From this point onward all MARLINE and Melanie models tested are using

SRP as base learner.

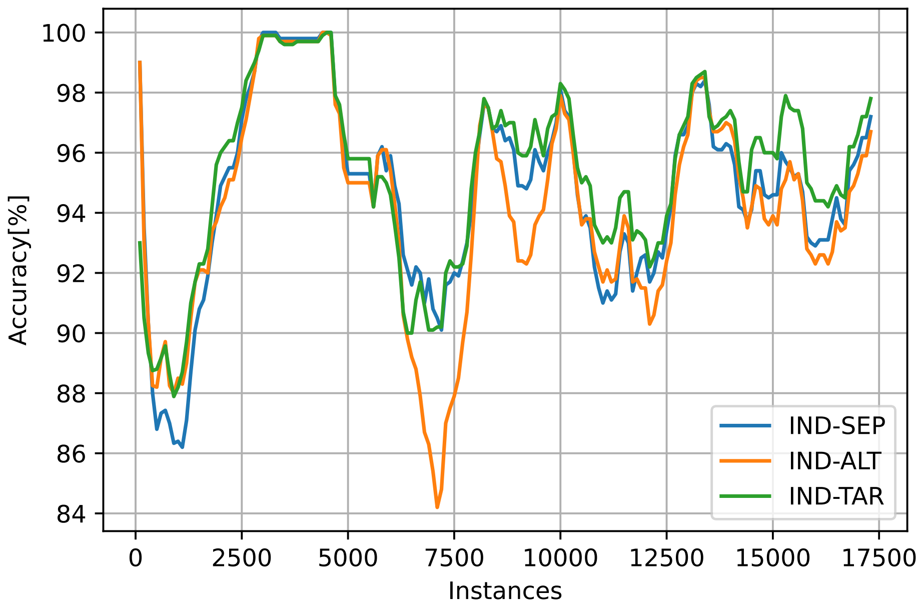

Figure 5.2: MARLINE with SRP dataset comparison

It is important to determine which form of knowledge transfer is of most benefit. Figure 5.2 includes a comparison between different variations of data in MARLINE. The use of alternating domains results in a significant decrease in accuracy. However, there is still need to explore potential application of transfer learning using MARLINE for SEP-like data.

Comparison of dataset variants on figure 5.3 is inconclusive with regards to which type of data is most suitable in that case. For each data set target data is the same. In the early morning and late evening hours it seems more appropriate to use raw data without transfer learning. This allows us to infer that there is probably less correlation between domains (busses and trams) at the time when traffic is less intense.

Despite relatively small representation of a single bus line (line 157), the application of transfer learning is not beneficial as shown in Fig.5.4. This may have to do with the fact that changes in distribution throughout the day are too frequent and unpredictable to compile with the overall state of public transportation. In this case, using an adaptive standalone SRP model is the best solution.

In the case of transferring general traffic state knowledge between buses and trams illustrated in Fig.5.5, it is more promising, because in many parts of the day, it is shown that using the transfer learning model is better than standalone SRP. Interestingly, it is the Melanie+ model proposed in this work that achieves better results than SRP. It is worth noting that Melanie+,

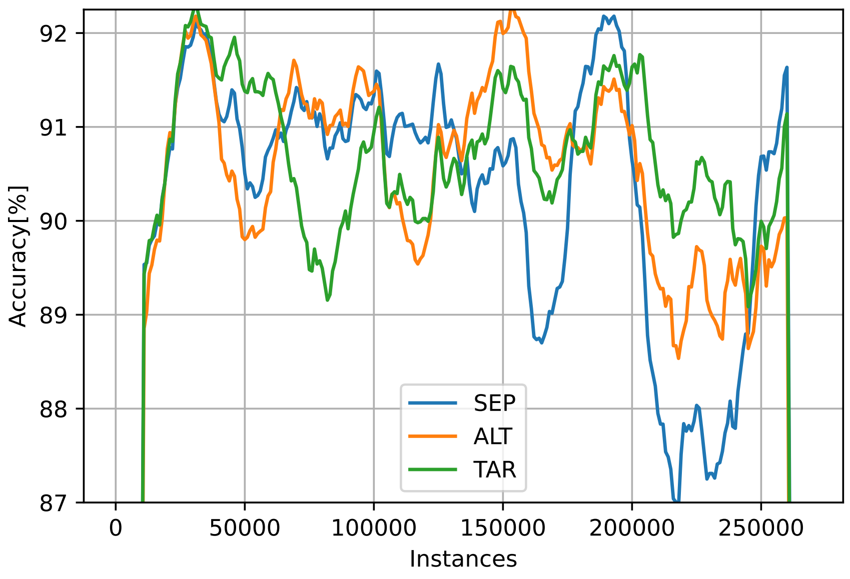

Figure 5.3: Melanie with SRP dataset comparison

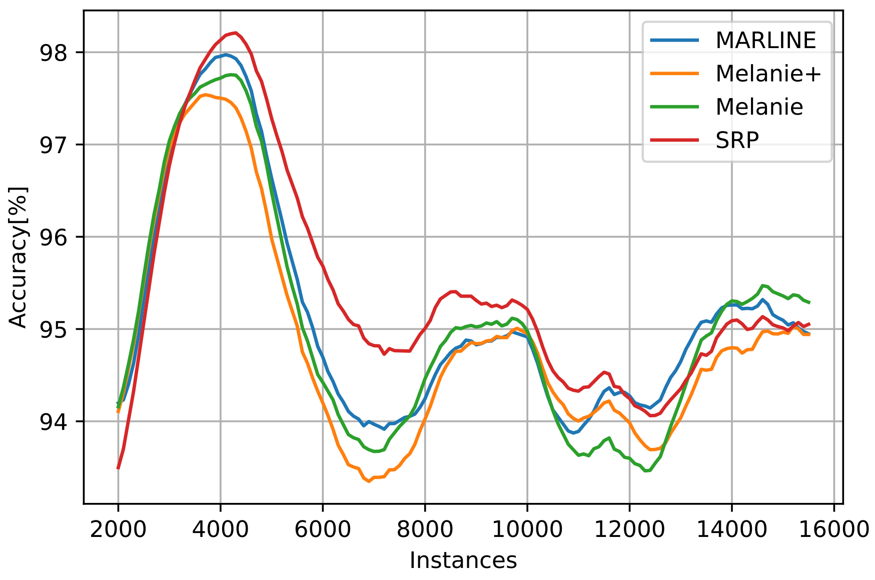

Figure 5.4: Comparison of models on SEP-IND dataset

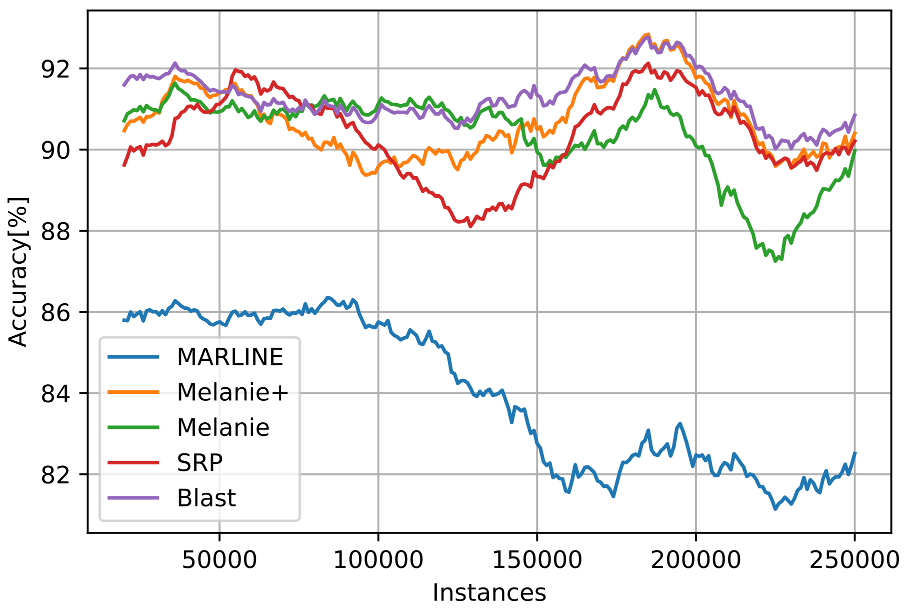

Figure 5.5: Model comparison on SEP data

which uses decisions from the source domain in a less direct way than Melanie, reduces the decrease in accuracy during the evening and early morning hours. Not surprisingly, the best solution turned out to be the use of BLAST, which consists of all the other models, but only considers the one that currently has the highest accuracy. However, this involves a longer training time which grows additively with the training time of the individual models.

Model	HIST-TAR	ALT	HIST-ALT	SEP	HIST-SEP	SEP-IND	HIST-SEP-IND	LARGE-HIST-SEP
Majority	61.95	54.43	61.95	54.43	61.95	74.85	74.85	61.95
SRP	93.35	89.84	**93.35**	89.84	**93.35**	95.69	96.01	**93.35**
Melanie Boost	85.54	82.22	84.83	81.27	86.05	95.77	96.19	86.05
Melanie SRP	93.01	89.88	91.11	90.79	91.53	95.06	96.33	91.53
Smelanie SRP	92.50	**91.21**	91.85	90.99	91.28	96.01	**96.45**	91.28
Melanie+ SRP	93.13	88.61	90.28	90.20	91.40	95.72	96.24	91.40
MARLINE SRP	92.15	83.05	83.31	77.51	78.91	94.82	96.34	78.91
Blast	**93.42**	90.89	91.59	**91.15**	91.85	**96.08**	96.41	91.85

Table 5.1: Average accuracy on 5-minute delay prediction datasets

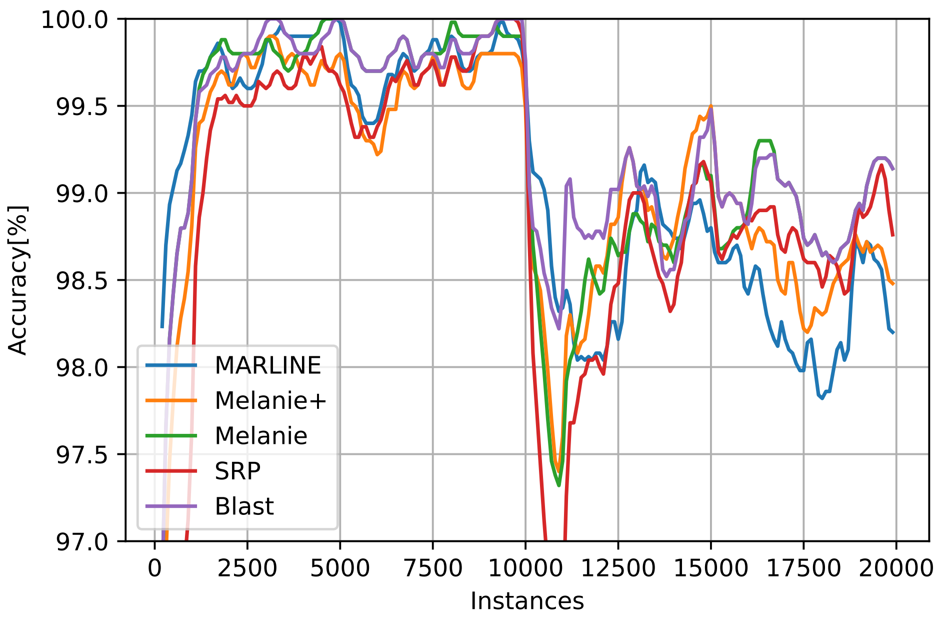

Figure 5.6: Model comparison on synthetic ABR data

As follows from Fig.5.6 for synthetically generated data in which there is sudden concept drift, the accuracy of individual models is variable and it is difficult to distinguish the best one. Great consistency can be achieved again using a combination of all models in form of BLAST.

Tables 5.1 and 5.2 summarize the mean accuracies and Kappa statistic under different model configurations. For type of data in which there is no historical context (without HIST prefix described in Sect.[4.3.5]), the application of transfer learning in the general case yielded measurable benefits. In particular, Smelanie using SRP compare favorably, specifically with regards to Kappa which is an indicative of low bias towards majority class. In each case except of IND variants (which are much smaller), using SRP as Melanie's base model was accompanied by significant improvements over Boosting.

Model	HIST-TAR	ALT	HIST-ALT	SEP	HIST-SEP	SEP-IND	HIST-SEP-IND	LARGE-HIST-SEP
Majority	0.00	0.03	0.00	0.03	0.00	2.78	2.78	0.00
SRP	**93.35**	78.14	**93.35**	78.14	**93.35**	85.20	83.51	**93.35**
Melanie Boost	85.54	62.13	67.48	81.27	86.05	95.77	96.19	86.05
Melanie SRP	93.01	78.44	81.07	90.79	91.53	95.06	96.33	91.53
Smelanie SRP	92.50	**81.35**	82.64	**90.99**	91.28	**96.01**	**96.45**	91.28
Melanie+ SRP	93.13	75.68	79.33	90.20	91.40	95.72	96.24	91.40
MARLINE SRP	92.15	63.77	65.05	77.51	78.91	94.82	96.34	78.91
Blast	85.09	80.48	82.11	81.14	82.64	86.84	87.70	82.64

Table 5.2: Average kappa on 5-minute delay prediction datasets

The MARLINE model performs comparably when the target domain is not very diverse (IND). However, when we try to model the overall traffic of the tramway network, the model records a major decrease in accuracy. Due to the limited applicability of MARLINE and its significant training time overhead, it is not used as one of the candidate models for BLAST.

In virtually every case, using SRP as Melanie's base model was associated with significant improvement. The exception was when the target domain is represented by only single bus line. Since OzaBoost does not respond to concept drift it is associated with greater retention of past knowledge. It appears that this behavior is more beneficial than high adaptability in this case.

While at the time of prediction, there is no information about the status of the tram 5 minutes earlier transfer learning models perform better than SRP. However, once there is access to such information (HIST), it is such an important and informative factor that the use of transfer learning degrades performance, so the optimal approach would be to use simply SRP and provide samples in form of HIST data stream. SRP also reaches highest average Kappa for every type of HIST data. Interestingly enough this is not the case for individual bus line (IND), for which every transfer learning method yields better overall results than SRP.

In other cases, depending on the problem presented, different models achieve the highest average accuracy. Often the differences between them are minimal. However, as can be seen in Fig.5.4 and Fig.5.5 at different times of the day, different models come to the forefront in terms of accuracy, and they are not always transfer learning models. In this case, combining all models and temporarily using the best one (BLAST) achieves the highest average accuracy and Kappa statistic in most cases. However, it is worth noting that the best candidate model is selected considering only accuracy.

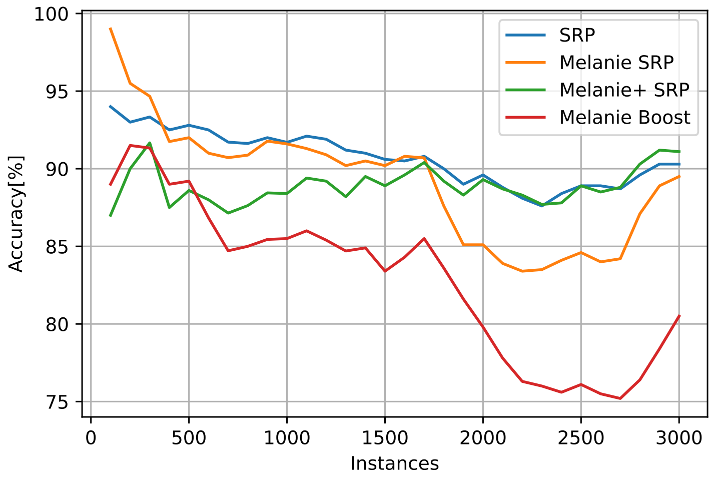

Figure 5.7: Transfer learning on initial samples

One of the benefits of transfer learning in a stationary environment is that it allows to achieve the asymptotic accuracy faster. In the case of smart city data streams from tram traffic, it appears the data from consecutive time intervals is often highly correlated, so a classifier trained on a small number of samples, without prior knowledge (SRP) is able to achieve similar or higher accuracy than transfer learning methods pretrained on bus data for previous day as seen on Fig.5.7. So in this case, the assumption suggesting benefits of transfer learning on initial data is not valid.

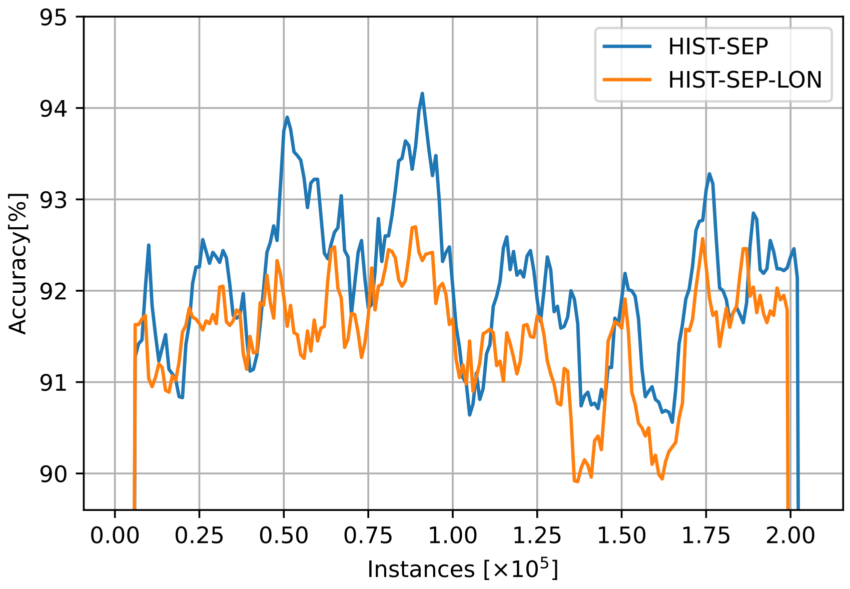

Figure 5.8: Delay comparison using BLAST

Model	HIST-TAR-LON	ALT-LON	HIST-ALT-LON	SEP-LON	HIST-SEP-LON	SEP-IND-LON	HIST-SEP-IND-LON	LARGE-HIST-SEP-LON
Majority	52.65	52.33	52.65	52.33	52.65	74.86	74.86	52.65
SRP	91.30	89.80	91.30	89.80	91.30	94.51	95.84	91.30
Melanie Boost	82.91	80.12	82.60	81.46	83.67	95.33	95.87	84.07
Melanie SRP	91.29	89.12	90.48	89.48	91.15	95.04	95.72	90.96
Smelanie SRP	90.61	**89.84**	**91.31**	88.55	90.83	95.36	95.86	90.99
Melanie+ SRP	90.70	88.34	89.91	89.48	90.84	94.85	95.37	90.56
MARLINE SRP	89.04	80.75	83.91	75.96	81.24	93.60	93.97	74.89
Blast	**91.70**	88.78	91.11	**89.93**	**91.47**	**95.39**	**96.14**	**91.49**

Table 5.3: Average accuracy on 10-minute delay prediction datasets

Model	HIST-TAR-LON	ALT-LON	HIST-ALT-LON	SEP-LON	HIST-SEP-LON	SEP-IND-LON	HIST-SEP-IND-LON	LARGE-HIST-SEP-LON
Majority	0.00	0.05	0.00	0.05	0.00	2.44	2.45	0.00
SRP	81.13	78.44	81.13	78.44	81.13	75.73	82.62	81.13
Melanie Boost	63.36	58.00	63.04	61.08	65.60	80.23	81.47	66.43
Melanie SRP	81.12	76.94	79.81	77.74	81.25	79.03	82.56	80.80
Smelanie SRP	79.59	**78.50**	**81.60**	75.75	80.64	79.73	83.14	80.95
Melanie+ SRP	79.81	75.26	78.60	77.66	80.58	77.72	80.07	80.03
MARLINE SRP	76.28	59.77	65.84	50.27	60.58	70.45	73.41	48.10
Blast	**81.99**	76.33	81.19	**78.69**	**81.93**	**80.83**	**83.96**	**81.99**

Table 5.4: Average kappa on 10-minute delay prediction (LON) datasets

We suspect that for the task of predicting a 10 minute forward delay is more difficult to constrain than predicting delay 5 minutes ahead. This is confirmed on figure 5.8 by slightly lower accuracy during the span of entire day. Both scenarios were tested using the Blast model as it proved to be most accurate and consistent.

Tables 5.3 and 5.4 include a comparison of the same models as in the 5-minute case, but this time on sets for predicting the 10-minute forward delay. Intuition would suggest, that predicting over longer timespan is significantly more difficult. The main and also the most important difference from the 5-minute variant is that there is no case in which standalone SRP achieves the best results. This is true even in the case of prediction with historical context (HIST), in which data is expanded to include information from 10 minutes before.

In this case it is already clear that the use of BLAST in general brings the best results for both accuracy and kappa measure. The exception is ALT-type data for which the use of transfer learning turns out to be detrimental in most cases, except for the simplified version of Melanie (Smelanie) proposed in this work.

In case of synthetic data sets (tables 5.5 and 5.6) which are comprised of data from differing 2D normal distributions there is apparent benefit of applying transfer learning techniques. This is especially apparent for target data with a small number of samples (50 and 500 for every class in each concept) where there is usually major difference in performance in comparison to standalone SRP, which has no knowledge at the beginning of training.

It is worth noting that in virtually every case, either of the Melanie modifications proposed in

Model	NO50	NO500	NO5000	ABR50	ABR500	ABR5000	INC50	INC500	INC5000	REO50	REO500	REO5000
Majority	47.00	49.47	49.20	43.50	49.23	49.28	43.73	48.35	49.57	45.72	48.44	49.42
SRP	92.00	96.79	99.18	95.25	98.62	98.94	86.00	83.89	87.90	95.72	97.29	99.07
Melanie Boost	**96.00**	97.48	98.41	96.50	98.80	98.96	88.89	88.36	88.52	95.00	98.59	99.32
Melanie SRP	95.00	**97.57**	98.53	95.50	98.82	99.19	86.53	88.41	88.96	95.56	97.14	99.27
Smelanie SRP	95.00	**97.57**	98.53	95.50	99.05	99.27	**91.94**	**89.49**	**89.63**	95.56	97.13	99.32
Melanie+ SRP	93.00	96.05	98.02	96.00	**99.20**	99.04	91.06	88.75	88.77	96.28	97.59	99.10
MARLINE SRP	95.00	97.14	99.09	**97.00**	99.16	99.08	90.56	88.26	88.93	**97.22**	**98.63**	**99.35**
Blast	95.00	**97.57**	**99.20**	95.50	99.03	**99.28**	91.53	89.36	89.53	95.56	97.50	99.33

Table 5.5: Average accuracy on artificial datasets

Model	NO50	NO500	NO5000	ABR50	ABR500	ABR5000	INC50	INC500	INC5000	REO50	REO500	REO5000
Majority	-6.00	-3.56	-1.70	-13.00	-2.54	-1.54	-12.53	-3.95	-0.94	-8.56	-4.22	-1.24
SRP	84.00	93.53	98.37	90.50	97.24	97.87	72.01	67.76	75.78	91.44	94.55	98.14
Melanie Boost	**92.00**	94.95	96.82	93.00	97.60	97.91	77.78	76.70	77.01	90.00	97.17	98.64
Melanie SRP	90.00	**95.12**	97.06	91.00	97.63	98.39	73.05	76.81	77.91	91.11	94.25	98.55
Smelanie SRP	90.00	**95.12**	97.06	91.00	98.10	98.55	**83.88**	**78.98**	**79.24**	91.11	94.24	98.64
Melanie+ SRP	86.00	92.10	96.04	92.00	**98.39**	98.08	82.12	77.50	77.54	92.56	95.15	98.19
MARLINE SRP	90.00	94.25	98.18	**94.00**	98.31	98.16	81.12	76.52	77.84	**94.44**	**97.25**	**98.69**
Blast	90.00	**95.12**	**98.40**	91.00	98.06	**98.56**	83.07	78.71	79.04	91.11	94.97	98.66

Table 5.6: Average kappa on artificial datasets

this work - Smelanie or Melanie+ achieves similar or better result than the original Melanie SRP. Although Marline with SRP as the base learner performs comparably or slightly worse than the other models in most cases, it clearly provides the best knowledge retention and outperforms the other models for reoccurring concept (REO) data.

5.2. Transfer learning in batch setting

The idea and application of streaming transfer learning originated from the batch environment. Although the principles of data presentation are very different, many other aspects remain the same, such as the division into different domains and the fact that data from the source domains are usually presented first. Likewise, the data format is acceptable for models in both cases. In the batch approach, however, one has access to a different kind of models, specifically neural networks, which differ significantly in their inner workings compared too very elaborate and modified, but nonetheless essentially tree ensembles of stream processing. It is for these reasons that it is worth considering batch models and what brings application of transfer learning in this particular approach.

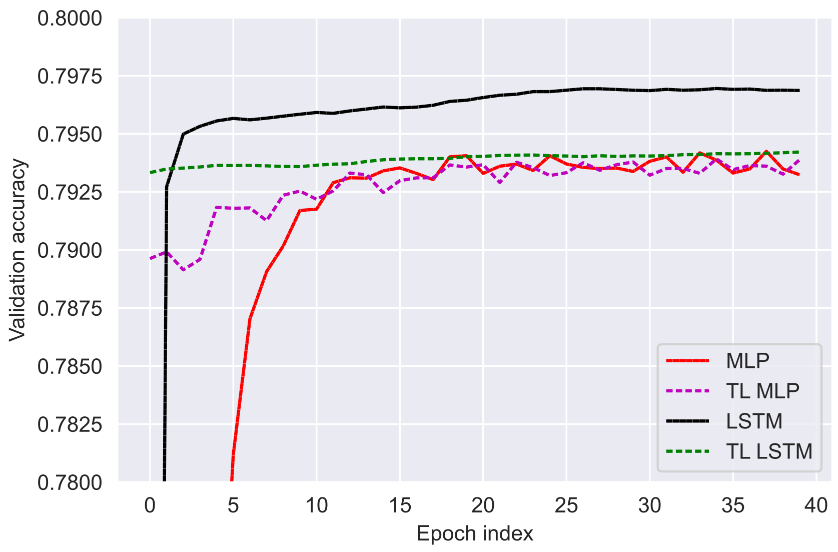

Figure 5.9: HIST-SEP testing accuracy

Two model configurations have been tested during the study. The first is a classic Multilayer Perceptron with a single hidden layer. The second model is the LSTM, which allows the use of data available in sequential form - in this case there is no shuffling of data during training as order and continuity is of the essence. Dropout layer is added to prevent overfitting. For HIST-SEP data both models are either presented with target domain data (MLP, LSTM) or are firstly pre-trained using data of busses from previous day (TL MLP, TL LSTM), for the same number of epochs as for target data and then trained and tested on target domain. For target domain being 10-minute delay prediction over entire tram traffic (figure 5.9) there is no added benefit of applying transfer learning other than higher accuracy during initial few epochs. For LSTM applying transfer learning proved to be detrimental to asymptotic accuracy. It can also be seen that for both MLP variants, the model is unstable despite the use of a small learning rate (0.0002 with Adam optimizer) and tenfold bootstrapping of train-test split.

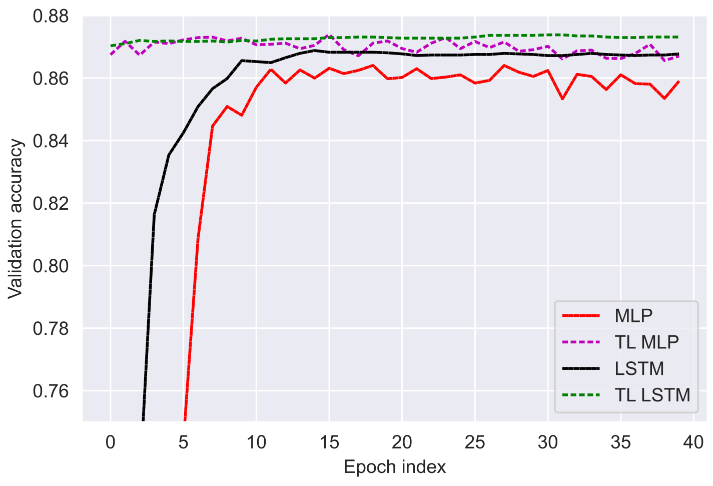

Figure 5.10: HIST-SEP-IND testing accuracy

When the target domain is more narrowly defined (figure 5.10), greater potential for transfer learning can be seen as in both cases it results in not only improved asymptotic accuracy but also great starting accuracy. As expected, LSTM having access to sequential data achieves slightly better results, as it is able to retain state of previously presented samples which is particularly relevant as for single bus line they are most often correlated. However in this view there is no possibility of evolving concepts and time is only one of the variables. More importantly, despite the fact that the parameters of both batch models were carefully chosen to achieve the highest possible accuracy, the streaming models are still able to achieve much higher accuracy in both data variants. Prepared neural network models achieve an asymptotic accuracy of less than 80% on HIST-SEP data, while on the same datasets models such as SRP exceed 93% accuracy. In the case of HIST-SEP-IND collections, the difference is less than 88% accuracy for batch models to more than 96% for streaming models, which in practice means more than 3 times lower error rate. This may be due to the fact that streaming models can adjust faster and more accurately to a changing distribution without completely losing information about past distributions which makes it more appropriate for this specific task.

6. Summary

This work analyzed the available transfer learning models in a streaming environment. New approaches that improve upon existing ones were also proposed and explored.

The streaming environment and the challenges posed by the peculiarities of this environment, such as concept drift and verification latency, as well as volume, variety and velocity of data, are described in detail first, followed by ways stream mining models deal with these issues.

Basic methods for learning on data streams, such as Hoeffding Trees, are presented with various ensemble variants, which often serve as a starting point for more complex methods - including methods for learning with knowledge transfer. Transfer learning models are also described and a survey of available approaches is conducted. These solutions aim to increase the predictive accuracy in target domain in a peculiar way - through the use of models trained on data available in related domains, which are trained either before or simultaneously with target domain classifier. It is an extremely useful feature especially in the case of stream mining.

The actual content of the work was realized in significant part by using MOA framework, which includes modern algorithms, on the basis of which new research in the field of transfer learning is constantly being conducted. This work adapted methods such as Melanie and MARLINE so as to be able to use Streaming Random Patches as a base learner, which brought tangible benefits over methods using classical ensembles such as online Bagging or Boosting. New methods based on Melanie have also been proposed. The first of one is sMelanie, which omits concept drift detection and the creation of new models for each concept drift in favor of handling the phenomenon completely by underlying models - in this case SRP. The next model is Melanie+, which uses the results from the source models to make predictions by the target models. Finally, the proposed methods were combined with previously available approaches into BLAST ensemble - which makes a prediction based on the temporarily most accurate model.

The goal of the reference task was to determine the delay of a vehicle based on its current status. Vehicle delay data from Warsaw was chosen as a reference task due to limited availability of quality smart city data, which often requires a great deal of pre-processing before it can get appropriated by a machine learning model. The smart city data was extracted and processed so that it was time-ordered and contained the necessary knowledge - such as the domain, basic information about the vehicle's present or past status, and a label that was retroactively assigned based on later samples. A distinction was made between two domains - buses and trams - and an attempt was made to determine the delay of trams with the help knowledge about bus traffic. Various data profiles were prepared in search of an optimal solution, including one in which the source domain completely preceded the target domain or occurred simultaneously in time. The experiments show that the use of SEP-type data, i.e., in which the source domains are first presented in full and the source domain models are fully trained before training the models on the target data yields better results in case of most models. The implication is that even though the domain data are not from the same period, having complete a priori knowledge of the distribution is better than modeling slightly more similar distributions simultaneously. Synthetic data was also considered for benchmarking purposes and as a result of the experiments, it was noted that the models proposed in this work and the approach using SRP as the base model has a significant impact on improving accuracy.

As a result of experiments, it has been deducted that the validity of transfer learning depends on the at-hand model and the adopted data profile. Particularly in the case of short-term prediction, sometimes the standalone SRP ensemble proves to be a very good solution - especially when knowledge of the past status from minutes before is also available. In almost every case, the use of SRP as a base learner for transfer learning methods was associated with a significant increase in accuracy. At certain times of the day, the new Melanie variants proposed in this work - sMelanie and Melanie+ - prove to be more accurate than other models. Specifically with regards to the morning, peri-morning and afternoon hours, which is when potential delay can be at its highest and prediction accuracy is then of greatest importance from a business perspective. The combination of these new methods and the previously available in form of BLAST turns out to be consistently giving best results. This is also confirmed by tests performed on synthetic data, which provide a good benchmark for determining the suitability of models in various situations, such as abrupt, incremental or reoccurring concept drift.

7. Future works

During the preparation of the solution many different approaches to the application of transfer learning were tested. A variety of data profiles were prepared and their use was explored. Available transfer learning models were modified and juxtaposed with state-of-the-art ensembles to achieve meaningful improvements. In particular, Melanie and its modifications combined with a non-transfer learning approaches such as SRP in the form of Blast ensemble proved to be the most accurate and consistent model.

In further considerations, it is worth exploring the transferability of knowledge at different scales, such as between communication networks of different cities. For this purpose, an appropriate data-mining pipeline for such cities would need to be prepared. Furthermore, it is possible to investigate transfer learning in other smart city domains such as electricity grid, waste management or water supply networks.

It is also worth developing currently available machine learning methods and proposing new unconventional approaches. One can try to combine static with streaming approaches under the assumption that the source domain data precedes the target domain. One could then model the source delay distributions over the course of a day and use this knowledge to support the target predictor.

Appendix - Source code

All the materials that were used to prepare this work have been collected in a GitHub repository [16]. The source code was prepared and shared largely on the basis of MOA software with accordance to GPL 3.0 license. Included within the repository are implementations of methods such as BLAST enabling transfer learning, Smelanie, Melanie+, and extensions of earlier implementations of SRP, Melanie, and MARLINE so that SRP can be used as the base model for these transfer learning methods. To run any of provided models you need to manually copy appropriate implementation file inside of MOA repository which you can find at https://github.com/rnetonet/moa and then one can run them using *EvaluatePrequiential* command like:

```
$ EvaluatePrequential \
-l (meta.Melanie2 -b meta.StreamingRandomPatches) \
-s (ArffFileStream -f path/to/data/stream.arff) \
-f 100 -d path/to/results.csv"
```

Melanie+ was provided in the implementation as *Melanie*2 due to the specificity of the programming syntax. The extension of the BLAST method which uses transfer learning was provided in form of *TransBlast* class. Candidate models are passed in form of comma-separated list, however they must maintain an order in which non-transfer-learning models are listed before transfer learning ones as flag $t \geq 0$ indicates index of first transfer learning model in the list.

```
$ EvaluatePrequential \
-l (meta.TransBlast -b meta.StreamingRandomPatches,meta.Melanie -t 1) \
-s (ArffFileStream -f path/to/data/stream.arff) \
-f 100 -d path/to/results.csv"
```

More details about MOA command line usage can be found in their documentation at the link https://moa.cms.waikato.ac.nz/details/classification/command-line/. The repository also contains the source code used to research smart city data processing in a batch environment.

Bibliography

[1] Lucas Baier et al. “Handling Concept Drifts in Regression Problems - the Error Intersection Approach”. In: *CoRR* abs/2004.00438 (2020).

[2] Albert Bifet et al. *Machine Learning for Data Streams with Practical Examples in MOA*. `https://moa.cms.waikato.ac.nz/book/`. MIT Press, 2018.

[3] L Breiman. “Random Forests”. In: *Machine Learning* 45 (Oct. 2001), pp. 5–32. DOI: `10.1023/A:1010950718922`.

[4] Gregory Ditzler et al. “Learning in Nonstationary Environments: A Survey”. In: *IEEE Computational Intelligence Magazine* 10.4 (2015), pp. 12–25. DOI: `10.1109/MCI.2015.2471196`.

[5] Pedro Domingos and Geoff Hulten. “Mining High-Speed Data Streams”. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’00. Boston, Massachusetts, USA: Association for Computing Machinery, 2000, pp. 71–80. ISBN: 1581132336. DOI: `10.1145/347090.347107`. URL: `https://doi.org/10.1145/347090.347107`.

[6] Honghui Du, Leandro L. Minku, and Huiyu Zhou. “MARLINE: Multi-source Mapping Transfer Learning for Non-Stationary Environments”. In: *2020 IEEE International Conference on Data Mining (ICDM)*. 2020, pp. 122–131. DOI: `10.1109/ICDM50108.2020.00021`.

[7] Honghui Du, Leandro L. Minku, and Huiyu Zhou. “MARLINE: Multi-source Mapping Transfer Learning for Non-Stationary Environments”. In: *2020 IEEE International Conference on Data Mining (ICDM)*. 2020, pp. 122–131. DOI: `10.1109/ICDM50108.2020.00021`.

[8] João Gama et al. “A Survey on Concept Drift Adaptation”. In: *ACM Computing Surveys (CSUR)* 46 (Apr. 2014). DOI: `10.1145/2523813`.

[9] Heitor Murilo Gomes, Jesse Read, and Albert Bifet. “Streaming Random Patches for Evolving Data Stream Classification”. In: *2019 IEEE International Conference on Data Mining (ICDM)*. 2019, pp. 240–249. DOI: `10.1109/ICDM.2019.00034`.

[10] Heitor Murilo Gomes et al. “A Survey on Semi-Supervised Learning for Delayed Partially Labelled Data Streams”. In: *ACM Comput. Surv.* (Feb. 2022). Just Accepted. ISSN: 0360-0300. DOI: 10.1145/3523055. URL: https://doi.org/10.1145/3523055.

[11] Heitor Murilo Gomes et al. “Adaptive random forests for evolving data stream classification”. In: *Machine Learning* 106 (Oct. 2017), pp. 1–27. DOI: 10.1007/s10994-017-5642-8.

[12] Maciej Grzenda, Karolina Kwasiborska, and Tomasz Zaremba. “Hybrid short term prediction to address limited timeliness of public transport data streams”. In: *Neurocomputing* 391 (2020), pp. 305–317. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.08.100. URL: https://www.sciencedirect.com/science/article/pii/S0925231219316182.

[13] H.Du. “Code repository for MARLINE:Multi-source mapping transfer-learning for non-stationary environments”. In: 2020. URL: https://github.com/nino2222/MARLINE.

[14] H.Du. “Code repository for MELANIE:Multi-Source Transfer Learning for Non-Stationary Environments”. In: 2019. URL: https://github.com/nino2222/Melanie.

[15] Adeel Hashmi and Tanvir Ahmad. “Big Data Mining: Tools Algorithms”. In: *International Journal of Recent Contributions from Engineering, Science IT (iJES)* 4 (Mar. 2016), p. 36. DOI: 10.3991/ijes.v4i1.5350.

[16] Karol Krupa. *Transfer learning for smart city data streams.* Version 0.0.0. Aug. 2022. URL: https://github.com/karolk98/smart_city.

[17] Douglas Laney. *3D Data Management: Controlling Data Volume, Velocity, and Variety.* Tech. rep. META Group, Feb. 2001. URL: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[18] Leandro Minku. “Transfer Learning in Non-stationary Environments: Methods and Applications”. In: Jan. 2019, pp. 13–37. ISBN: 978-3-319-89802-5. DOI: 10.1007/978-3-319-89803-2_2.

[19] Nikunj C. Oza and Stuart Russell. “Online Bagging and Boosting”. In: *In Artificial Intelligence and Statistics 2001.* Morgan Kaufmann, 2001, pp. 105–112.

[20] Yongrui Qin et al. “When Things Matter: A Survey on Data-Centric Internet of Things”. In: *Journal of Network and Computer Applications* 64 (Feb. 2016). DOI: 10.1016/j.jnca.2015.12.016.

[21] Jan N. van Rijn et al. “Having a Blast: Meta-Learning and Heterogeneous Ensembles for Data Streams”. In: *2015 IEEE International Conference on Data Mining* (2015), pp. 1003–1008.

[22] Muhammad Umer and Robi Polikar. “Comparative Analysis of Extreme Verification Latency Learning Algorithms”. In: *CoRR* abs/2011.14917 (2020).

[23] Geoffrey Webb et al. “Characterizing Concept Drift”. In: *Data Mining and Knowledge Discovery* 30 (July 2016).

[24] Yimin Wen et al. “Online transfer learning with multiple decision trees”. In: *Int. J. Mach. Learn. Cybern.* 10 (2019), pp. 2941–2962.

[25] Fuzhen Zhuang et al. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* PP (July 2020), pp. 1–34. DOI: 10.1109/JPROC.2020.3004555.

List of Figures

List of tables