The large diffusion of consumer-level wearable devices has opened many possibilities related to activity monitoring. Smart watches and devices such as Fitbit [Fit17] are increasingly used by people to track their daily motor activity, whereas a wide variety of biosensors is starting to play an important role in patient monitoring. The task of detecting motor activities such as walking from sensor data is thus becoming very popular in the fields of data science and machine learning. 

Model development for prediction in general is an iterative process which model to deploy. Therefore, comparing and reasoning about prediction results of alternative classifiers is a crucial step in the process of iterative model development. This process is, however, inherently difficult in the case of sensor-based activity prediction, where intricacy of long temporal sequences, high prediction frequency, and imprecise labeling introduce an additional layer of complexity, making standard evaluation methods less effective or even misleading. To improve the development cycle of sensor data classifiers, we introduce Track Xplorer—an interactive visualization system to query, analyze and compare the classification output of activity detection at multiple levels of granularity. Track Xplorer tackles the scalability and interpretability issues associated with sensor data classification through multiscale, contextual visual analysis and supports classifier debugging, versioning and collaborative analysis. Our system visualizes the results of different classifiers and ground truth labels as temporally-aligned linear tracks, and provides a coordinated video playback of activities to contextualize their predictions. Users can interactively explore the results of different classifiers, and assess their accuracy with respect to the ground truth labels and video. To this end, we also contribute an algebra over track representations to filter, compose, and compare classification outputs, enabling users to effectively reason about the performance of classifiers. We demonstrate how our system helps data scientists debug misclassifications and improve the prediction performance in developing activity classifiers for real-world, multi-sensor data collected from individuals with Parkinson’s disease.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Human-centered computing]: Human computer interaction (HCI)—Interactive systems and tools

1. Introduction

The large diffusion of consumer-level wearable devices has opened many possibilities related to activity monitoring. Smart watches and devices such as Fitbit [Fit17] are increasingly used by people to track their daily motor activity, whereas a wide variety of biosensors is starting to play an important role in patient monitoring. The task of detecting motor activities such as walking from sensor data is thus becoming very popular in the fields of data science and machine learning.

Model development for prediction in general is an iterative process where data scientists start with some alternative models along with associated hyperparameters, train them on a subset of the available data and then evaluate their performance on test data. During this evaluation step, aggregate performance metrics such as accuracy score, precision and recall are often used to establish how well a classifier can identify specific activity events and to compare the efficiency of different prediction models and parameters. Then, data scientists use these insights to go back to their code and tune their algorithms and parameters, restarting the cycle and iteratively improving their models. Analyzing the output of predictive models is a crucial step in the life cycle of iterative model development (Fig. 1).

When dealing with motor activity detection based on sensor data, common evaluation methods suffer from scalability and interpretability issues due to the temporal nature of the data. Each motor activity detection is in fact based on the aggregation over time windows of many model predictions computed at high frequency, and is generally validated against labels generated through manual video annotation. The need for evaluating a classifier’s output based on this time overlap, significantly affected by human interpretation, purpose and bias in labeling motor activities, often makes aggregate performance metrics alone fall short in conveying insights into why one prediction model seems to perform better than the other. Given this inherent additional layer of complexity, it would instead be desirable to be able to analyze activity prediction results at different levels of granularity, with the possibility to interactively drill down to the context of each motor activity prediction to better interpret the causes of mispredictions.

Figure 1: Improving model development life cycle. Track Xplorer enhances the comparative evaluation of multiple predictive models through contextual visual analysis.
We introduce a novel visualization system, Track Xplorer (also referred to as Xplorer for short), to interactively analyze the classification results of sensor-based predictive models. Xplorer enables users to debug and compare multiple classifiers up to the level of granularity of a single prediction, providing different qualitative and quantitative ways to validate the performance of each model. Xplorer facilitates the interpretation of classification results in application context, enabling data scientists to reason about the causes of misclassifications and to improve their predictive models. We note that our system does not aim at gaining insights on the internal behavior of a predictive model, rather it serves the purpose of analyzing its output.

Our contributions include 1) a novel visualization system for visual analysis of classification performances on temporal data, 2) a set of methods to integrate contextual information and model metadata to support performance analysis, and 3) a visual track algebra for filtering, composing, analyzing and comparing classification results.

To illustrate the usefulness of our system, we present a use case involving the development of predictive models to detect specific motor activities in individuals with Parkinson’s disease. We study the usage of Xplorer through a group of fourteen participants, data scientists and business managers, working on the same project. We demonstrate how Xplorer proved to be essential for visually validating and comparing predictive models, for reasoning on the causes of mispredictions, and for understanding the trade-offs in the usage of different sensors – improving the overall predictive model development life cycle. We further observe how the system facilitated the discussion among data scientists as well as between data scientists and business managers in general.

In the following, we first give a synopsis of prior work, followed by a brief discussion of our system design. We then provide details on Track Xplorer’s interactions and visual design along with its track algebra, command line set and classification validation support. Next, we discuss the usage of Track Xplorer in the development of classifiers for detecting movement patterns in Parkinson’s disease. We conclude the paper by summarizing our contributions.

2. Related Work
Our work is related to earlier research in systems infrastructure for improving machine learning (ML) model development cycle [RBER17, RDSW*16, VSL*16, MLDD17, MCD17, vdWPS*17], interactive analysis of classifier performances [RAL*17, ACD*15, AHH*14], sequential and temporal data querying and visualization [KSF*02, RTW*11, SGM*04, FWBB10, DHHH13, PMR*96, MLAo*13, WGGP*, ZDFD15, All83, FKSS06, JS09, SAA*94, JS99, Kar94, HOB, BSM04], and systems that facilitate visual analysis through algebraic operations (e.g., [Wil05, STH02, DHHH13]).

2.1. Systems Infrastructure of Improving ML Model Development
The practical success of ML and of deep learning in particular has dramatically increased the demand for applying ML models to solve
problems across domains. However, ML model development is iterative and time-consuming, often relies on trial-and-error, and requires high engineering skills and large training data, which are often expensive and difficult to acquire and maintain [SHG*15].

Recent database research proposes systems to improve ML modeling life cycle. ModelDB [VSL*16] stores machine learning models along with associated pipelines and parameter metadata, enabling the exploration of these models through SQL queries. ModelHub [MLD17] specializes in deep learning models, providing a custom model versioning system to track the models and a domain specific language to configure deep networks and their hyperparameters.

Training data collection and management is also critical for end-to-end machine learning. Snorkel [RBER17] supports training data generation at scale for ML models using weak supervision through user-defined labeling functions [RDSW*16]. DataHub [BCH*15] supports managing datasets and their versions over time. Van der Weide et al. [vdWPS*17] tie data versioning to ML pipelines, tracking data resulting from intermediate stages of ML pipelines to reduce the redundancy of computations and improve their robustness. ProvDB [MCD17] tracks the provenance of artifacts (e.g., data, scripts, results) from data science experiments using git and a graph database. Similarly to VisTrails [CFS*06], ModelDB, ProvDB, and [vdWPS*17] also use workflows (pipelines) to represent modeling or analysis processes. Unlike VisTrails, where users manually design the workflows, these three tools generate or infer the workflows for users.

Research in ML model management is nascent and sparse, limited in applying visual analysis and interactive exploration. Data visualization has to be part of any usable ML model management stack. Track Xplorer is similar to the work above in part in using a git-based versioning. It is, however, the first system that enables visual and quantitative analysis, comparison and tracking of temporal multi-sensor data classifications across models.

2.2. Visual Analysis of Classifier Performances

Researchers have introduced interactive tools, e.g., [RAL*17, ACD*15, AHH*14] to help data scientists make sense of their classifier performances. Squares [RAL*17] supplements summary performance statistics with instance-level distribution information, uncovering distinct characteristics of classifiers having comparable aggregate performance. Similarly, ModelTracker [ACD*15] and Confusion Wheel [AHH*14] adopt a tighter coupling of performance with data instances to enable multiscale analysis. Xplorer complements earlier work on classifier performance analysis by focusing on temporal data classifications, integrating additional “human soft knowledge” (e.g., activity videos and expert labels), and introducing a visual algebra over classification results that enables composable and rigorous performance analysis.

2.3. Sequential Data Visualization and Querying

The visual design of Xplorer draws from genomic data browsers (e.g., [KF*02, RTW*11, SGM*04, FWBB10]) and multimedia editors [App17, Ado17], using visual encoding along a linear axis (track) of data and metadata sequences as the basic unit of representation. Genomic browsers enable the visualization of molecular sequences from various sources as aligned, linear tracks—which can be added, removed and reordered on demand. Genomic browsers support interactions such as zooming and panning to enable fine-grained exploration of the data, often encoded as horizontal bars of variable length. These features are also common in multimedia editors, where tracks typically represent audio or video sources, and are shared by many other tools from the temporal and sequential data visualization literature [PMR*96, WGGP*, HOB, Kar94, BSM04].
3.2. User Interface

The interface of Xplorer (Fig. 2) is composed of a main view, where classification results and labels are represented as linear tracks stacked vertically. A track visually corresponds to a sequence of non-overlapping colored blocks, positioned over a common timeline. We categorize tracks into two types based on the form of the data they represent: classifier tracks and label tracks. A classifier track contains probability scores associated to each event and can be visualized either as an area chart or as a horizontal bars ("blocks"), whereas a label track contains only information about time intervals. A classifier track can be converted into a label track by applying a threshold on the classifier track’s prediction scores.

Figure 5: A track in Xplorer corresponds to a list of non-overlapping time-periods (“events”). There are two types of tracks based on the form of the data that they represent: classifier tracks and label tracks. A classifier track contains probability scores associated to each event and can be visualized either as an area chart or as a horizontal bars ("blocks"), whereas a label track contains only information about time intervals. A classifier track can be converted into a label track by applying a threshold on the classifier track’s prediction scores.

Our system tackles these issues through automated precomputation and data compression. In particular, we rely on a set of procedures that periodically take care of accessing the available predictive models, of running them on the right subset of the sensor data, and of compressing and storing their output (Fig. 3). Data scientists are required to include in their code a standardized set of attributes, which hold meta-information about the nature, version and requirements of a classifier. This information is complemented by the git version control system, which holds data about user commits and modifications in the code. An analytics pipeline checks the code repository for new versions of the models and takes care of running them when a change is detected, performing the computation according to the meta-information extracted from each classifier. Results are then stored in a centralized database together with the classification results. By merging similar predictions close to each other in time (Fig. 4), we compress the results and store them in a JSON-based file (.BSX) that can be later opened from the public, web-based Xplorer user interface.

An important design criteria preserved by our system consists in preserving data confidentiality while allowing an easier distribution of classification results. Since data science projects are often run across groups or companies, a common requirement is to maintain separate access privileges to prediction results, sensor data and source code. Through the distribution of BSX files, any user can be easily granted access to prediction results through the publicly available Xplorer interface, without being exposed to any confidential information. On the other hand, data scientists can easily enable an optional, protected ssh tunneling connection (Fig. 3) to access confidential information without having to deal with BSX files.

3.3. Interactions

Track information can be analyzed at different levels of granularity through zooming and panning, which are performed with the mouse wheel and drag actions. By hovering on a block, information about the correspondent prediction or label is shown (e.g. author, duration) as a tooltip. For classifier tracks, the tooltip shows classifier-specific variables associated to the prediction (e.g. “tremor frequency,” “angular velocity”), that data scientists can use to debug their algorithms (Fig. 6).

Each classifier track also includes four buttons enabling the user to 1) increase its height for better visibility, 2) play consecutively the videos of all detected activities, 3) display information about the underlying predictive model (e.g. sensors, prediction window and threshold used), and 4) switch between two different visualization modes. Fig. 5 explains how a classifier track can also be represented as an area chart, visualizing outputs continuous probability scores, the block is generated by applying a classifier-specific threshold to the score values (Fig. 5). The opacity of a block encodes the associated probability score, increasing with high score values and decreasing with low score values. When dealing with multi-class classifiers, Xplorer generates by default a separate track for each predicted class, leaving an option to aggregate them visually.

As for label tracks (Fig. 2b), each block corresponds to a textual label (e.g. “Walking,” “Person is sitting”), characterized by a start and end time which determine its position and length. Labels can be either algorithmically generated or manually defined by a human, and are often used as ground-truth or as a reference for validating classifier tracks. A particular type of label track, called protocol track (Fig. 2c), can hold different unique labels on the same timeline, given they do not overlap with each other. A protocol track is generally used as a reference to the data collection process, where each block corresponds to a specific task performed by a subject.

While all other temporal data is represented as a linear track, video is shown in a separate undocked window (Fig. 2d), which can be dragged across the interface and freely resized by the user. The interface of Xplorer also includes a left sidebar, from which users can decide which tracks to visualize and easily zoom to specific events contained in the protocol track, and three auxiliary modal windows. The latter can be used for (1) analyzing classifier performance, (2) inspecting model information and code revisions, and (3) accessing raw sensor data associated to a particular prediction or time interval.
While analyzing each track separately may be sufficient for some with the labels associated to “Standing”. To enable reasoning beyond the video functionality, it also enables the consecutive playing of all false events that were not identified by the predictive model (false negatives). Similarly, we can define the difference between track and a label track used as a classifier track and as a label track containing ground-truth labels. By computing the intersection and the subtraction of the two tracks, a user can quickly identify the correct and incorrect predictions.

4. Visual Track Algebra

While examining each track separately may be sufficient for some applications, in many cases the possibility to combine different tracks could be essential. For instance, a user may want to analyze the output of a slow classifier only when a different classifier is predicting no walking movement. Similarly, a user may want to consider all moments in which a subject is stationary, thus needing to unify the labels associated to “Sitting” with the labels associated to “Standing”. To enable reasoning beyond the scope of single classifiers and labels, we define a visual algebra that allows to generate new tracks as a combination of existing tracks. Operations such as addition, subtraction, logic conjunction and disjunction can be applied to both classifier and label tracks with different semantic meaning.

Fig. 7 illustrates how the most common operators can be used for classifier validation. If we denote a classifier track by A and a label track used as ground-truth by B, A ∩ B corresponds to the intersection of both tracks, that is to the events that were correctly predicted by the model (true positives). Similarly, we can define the difference between track A and track B as a new track were all block instances of B are removed from A. This way, the track A − B will contain all classifications that do not match any ground truth label (false positives), while B − A will conversely represent labeled events that were not identified by the predictive model (false negatives).

The power of the track algebra consists in enabling users to quickly combine tracks to validate complex hypotheses about the classification process. In particular, in presence of ground-truth labels, it makes the identification of misclassified events visually straightforward. In combination with the video functionality, it also enables the consecutive playing of all false positive and all false negative predictions for a particular classifier. This way, the user can visually validate the performance of his predictive model and reason on the causes of each single misprediction based on its context.

4.1. Command Line: Combining, Filtering and Ordering

Track Xplorer interface features a command line interface for enabling users to quickly perform complex interactions, such as track manipulation through visual algebra. Fig. 8 shows a list of the most common commands that can be executed from the command line. Each command is composed of one operator and one or two operands, which can be track identifiers or numerical values. A track identifier is automatically generated as a combination of the track name, author and version (e.g. the first version of the “SleepingJohn” classifier created by author “John” will generate the ID “SleepingJohn1.0”) and is made available through autocompletion. For instance, while typing “threshold sle” the command line will automatically infer which available track is best suited for the operator “threshold”, highlight it in the main view and offer the suggested completion “threshold SleepingJohn1.0”. Tracks can also be referred to based on their order of appearance (e.g. “union 1” will generate the union of the first two tracks) or by string wildcards (e.g. “show %walk” will make visible all tracks related to walk). When a command generates a new track, this one is added to the main view and its name and identifier are automatically defined based on the operation performed. The command line can also be used for computing performance metrics, ordering tracks and filtering classification events based on their attributes. For instance, the command “filter TurningErhan1.2 angle>60&duration>2” will create a new track with only slow turn events that last more than two seconds and in which the subject rotates by more than sixty degrees. After selecting a moment t along the timeline with the cursor, the command “order” performs instead a selection of all tracks containing an event overlapping the time interval $(t−\epsilon,t+\epsilon)$, and orders them based on their temporal match.

4.2. Classifier Validation

While observing a classifier track A and a ground-truth label track B next to each other, it becomes intuitive to understand that the performance of the predictive model depends on how much the blocks of each track are aligned with each other. Optimally, for each block in A there should exist...
Figure 9: Common performance metrics computed through track algebra. A represents a classifier track and B represents a label track containing ground-truth labels. Accuracy, precision and recall scores can be visualized as a “container” track with partial color fill. Mispredictions affecting a specific performance metric are localized in the non-filled (blank) regions of the track.

Figure 10: Performance metrics (top) and classifier information (bottom) modal windows. Xplorer features a modal window (top) to display different performance measures for a classifier track and includes an interactive ROC (Receiver Operating Characteristic) curve to help the user choose an adequate threshold for the selected model. The information modal (bottom) displays model metadata and summarizes commits and modifications performed on the code repository.
We present here the application of our system to a project involving Xplorer was used as a companion tool over most of the project by a team.

5.1. Classifier Development and Comparison

5. Use Case: Detecting Motor Activities in Parkinson’s Subjects

We present here the application of our system to a project involving the development of predictive models to automatically detect specific motor activities performed by subjects affected by Parkinson’s disease. Xplorer was used as a companion tool over most of the project by a team of fourteen data scientists and business people.

A total of six wearable IMU sensors were used, worn by twenty-five Parkinson’s disease subjects over multiple sessions (i.e. visits) of about one hour. The sensors measured accelerometer, gyroscope and magnetometer information at 128Hz and were placed on the wrists, feet, chest and back of the patients. During each visit, all subjects performed the same set of predefined tasks, according to a single clinical protocol. A group of external technicians took care of recording sessions, labeling specific activities and time-stamping the execution of tasks. Sensor data, ground-truth labels and video files were all stored in a single database, that all data scientists could access during the development of their predictive models.

Each data scientist developed their algorithms in Python, in accordance with the specifications of our analytics pipeline. Their code was pushed to a private Github repository and automatically loaded and executed by our system. The results of the computation were systematically made available to the team through the Xplorer user interface, without any additional input from the group members. The analysis of classification results was performed both individually and collaboratively during weekly team meetings.

5.1. Classifier Development and Comparison

Developing a predictive model is an iterative process based on trial and error, which comprehends analyzing a first set of classification results, tuning model parameters and then recomputing predictions. Xplorer proved to be valuable for data scientists for discovering insights during the analysis of classification results.

Video Playback and Track Algebra. The playback functionality, in combination with the track algebra, proved to be a fundamental feature for quickly identifying mispredictions. For example, by subtracting the “Walk” label track from the “Walk” classifier track and by playing the resulting track, it was possible to observe all cases in which the classifier did not detect that the subject was walking (false positives). By observing the video and the task labels, data scientists realized that, since the model was using the sensor worn on the chest, it was incorrectly detecting movements such as arising from the chair and coat buttoning (Fig. 12a).

Similarly, the “Step detector” classifier track (based on sensors worn on the shoes) showed false positives in correspondence with feet tremor (Fig. 12b), particularly common when subjects were sitting with their legs crossed. Based on these insights, data scientists decided to re-train their classification models with data from different sensors or by including additional features.

Model Debugging. Another widely used feature was the ability to inspect information about each single prediction. After noticing that the two hand classifiers “Pronation-supination” and “Tremor” were biased by the action of walking, data scientists were able to mouse over mispredicted events and observe the attributes computed by their predictive models (Fig. 12d). In this case, each prediction held numerical information about hand rotation angle, hand rotation speed, tremor frequency and tremor amplitude. By analyzing these attributes, data scientists were able to filter out movements happening at specific frequencies associated with walking, thus making their model more robust.

Similarly, the filtering function was used to check the validity of step detections, revealing several events characterized by unexpectedly long durations and unrealistic speed (Fig. 12b). By isolating these events and playing their associated video, it was discovered that the step detection algorithm couldn’t detect properly the landing phase of the foot. The possibility to dynamically visualize and download the raw data associated to a time period was respectively used to verify the correct behavior of algorithms and to precisely isolate sensor data for further offline analysis.

Classifier Versioning. A key feature of Xplorer is represented by the possibility to compare the results of different classifiers, but also of different versions of the same predictive model. Not always updating a model may result in a better performance, and keeping track of changes is important for understanding which modifications have led...
to an improvement or need to be reverted. A typical use of version comparison was made to tune model parameters to balance the amount of false positives and false negatives, in particular for classifiers such as “Sit2Stand” and “Tremor” (Fig. 12d).

5.2. Label Accuracy and Validation

While human-generated labels may seem to represent a valid source of ground-truth information, blindly assuming their completeness and correctness can easily lead to wrong insights and to inaccurate performance estimates. If a time period is mislabeled, for instance, it can erroneously reveal a false positive or false negative prediction, thus decreasing the classifier’s performance and creating a bias in interpreting its classification results.

Partial Labeling. The first issue encountered within the project consisted in the discrepancy between the ground-truth labels provided by video annotators (who were instructed based on a pre-existing medical protocol) and the ones requested or expected by data scientists (more aware of machine learning requirements for training). In many cases, these types of problems could be easily solved by combining labels through track algebra. For instance, while raters defined partial labels such as “ShortWalk” and “Long-Walk” or “Turn90” and “Turn>90”, data scientists were able to combine them in single tracks (“ShortWalk+LongWalk”, “Turn90+Turn>90”) and then use them for evaluating the performance of their classifiers (Fig. 6).

Missing Labels. Another problem related to manual video annotations was the sporadic absence of labels, either due to distraction of the technician or to the subject being off camera. While evaluating the performance of the “Walking” classifier through track algebra, the team noticed a percentage of false positives much higher than expected. By playing the corresponding parts of the video, it was discovered that the subject had walked outside of the camera view, where the video annotator couldn’t tell what actions were performed. Here, the annotator was asked to generate a new track of labels indicating all the moments in which the patient was off-camera, which were then excluded from the performance evaluation via track algebraic subtraction (Fig. 12c).

Label Definition and Human Bias. While inspecting mispredictions at a more granular label, the team observed that one of the causes of lower performance was related to the definition of motor activities themselves. For instance, while the “Walking” classifier was trained to recognize any horizontal movement involving feet motion, the annotator’s definition of the motor activity assumed a minimum of three steps to be made by the patient in the same direction. Because of this, the track algebraic difference showed a large set of false positives associated to small movements, such as performing few steps for reaching an object. Similarly, an inconsistent definition of “turn” was used for generating labels. During team meetings, typical questions included “Should a larger rotation of the chest be considered a turn, even if the legs don’t move?”, “Which is the minimum angle of rotation that defines a turn?”, and “To what extent is it useful to consider such details for the purpose of the project?”. Fig 12e shows how these inconsistencies in labeling were handled through track algebra. By visually matching the alignment of predictions, data scientists also noticed a human bias in the annotation of motor activities. For instance, it was discovered that some “walking” labels occurred before the patient actually started moving his feet. We assume this is related to the video annotator already expecting the intention to move of the subject, whose actions were dictated by a specific protocol. During performance evaluation, data scientists were able to limit these issues by considering patterns in detection through the “~” operator (Fig 11c). In order to partially overcome the problem, the team of data scientists asked the technicians to provide multiple ground truth labels for the same activity, each produced by a different video annotator. Thanks to Xplorer’s track intersection functionality (Fig. 11e), data scientists were then able to run performance evaluation only in the regions of agreement of the annotators (Fig. 12e).
5.3. Team Collaboration
While data scientists made individual use of Xplorer for classifier development, the visualization tool was also used collectively by the entire team during weekly meetings. In particular, its visual output was projected on a large screen or shared to remote participants via video conference. Every week, the tool would be used to show the progress on the acquisition of new patient data, on the development of classifiers and on the generation of new ground truth labels. One person often acted as a moderator, using Xplorer to go through the team progress and leading the conversation.

Collaborative Interpretation. Due to its abstraction and simple visuals, Xplorer proved to be an efficient medium for discussion among people with very different backgrounds. Even without being familiar with machine learning or knowing the details of each predicted model, it was sufficient for team members to visually check the alignment of tracks and further validate them with the video. Each data scientist could observe and give their feedback about models built by other developers, without knowing all their implementation details. Domain experts as well were able to share their knowledge about Parkinson’s disease, and suggest data scientists how to best handle certain motor activity events. On the other hand, data scientists could express through Xplorer the need for additional training samples or better quality ground-truth labels.

Decision Making. From the managerial perspective, Xplorer was used to track the progress of the project with respect to deadlines and to assess the quality of each predictive model before deployment. Based on project requirements, human resources were dynamically allocated in order to compensate predictive models that showed weaknesses in Xplorer. For instance, the tool opened a discussion on the quality and reliability of labels, making the management consider hiring new personnel for manually annotating videos. Similarly, false negatives shown in Xplorer were used to justify the acquisition of new data from additional patients, in order to have enough samples for each Parkinson’s phenotype. Based on classifiers’ accuracy and video footage, a subset of the sensors was also excluded from the scope of the project, with the decision of making data scientists focus more on wrist and lumbar sensors.

6. Discussion
6.1. Improving Scalability At Multiple Levels
There are five main data components that a user can benefit from while performing visual analysis of classification results: (1) the actual results of the analysis, (2) ground truth labels, (3) multimedia footage, (4) raw sensor data and (5) additional metadata information. While (3) can be consumed via streaming and (5) generally consumes a limited amount of memory, the number of instances of (1), (2) and (3) can cause serious scalability issues, affecting the interactivity of the system. We solve these issues by reducing the frequency of (1) and (2) events through the compression algorithm illustrated in Fig. 4, and by dynamically loading (3) only for small subsets of the dataset, based on user request.

A different scalability limitation lies instead in the number of tracks to visualize on screen, which amounted to almost one hundred in our use case scenario. Too many classifier or label tracks displayed at the same time can make visual search and comparison cumbersome for the user, who would rather keep a small, ordered set of tracks in order to achieve the desired visual insights. For this reason, we combine the use of smart autocompletion and wildcard selectors in the command line with a smart ordering functionality.

Smart track ordering addresses the situation in which the user is interested in a particular moment in time and wants to visually compare all tracks that contain relevant information in that particular time window. For instance, the user is focusing on the moment in which the subject is standing up from the chair and is interested in seeing tracks that detect events related to this action: smart ordering automatically reorders classifier and label tracks, bringing Arising from chair, Sit-to-Stand, Sitting and Standing close to each other. Smart ordering further keeps memory of previous tracks of interest, so that if the user focuses on a consecutive action in which the user sits back on the chair, the track Stand-to-sit is ordered on top and the track Sit-to-stand (no more involved) is still kept visible.

6.2. A First Step Towards Model Versioning
With increasing maturity of machine learning technology, research has more and more focused on creating systems to more effectively build models. However, despite the fast expansion of machine learning applications into new domains, many aspects of the predictive modeling life cycle have not been properly addressed yet. Track Xplorer relies on enforcing a standardized model definition which all data scientists will commit to use, but leaves them free to continue using git as a versioning system. In particular, Xplorer’s analytics pipeline extracts automatically model metadata information from git commits and presents it in the front-end user interface (Fig. 10, bottom) while comparing multiple versions of the same classifier. Data scientists can use the track algebra to identify differences between model versions 13 and then inspect which parameters and commits are associated to the best results. This way, data scientists can easily go back to their code and revert or confirm modifications to their model.

6.3. Running Meaningful Performance Evaluations
As shown in Fig. 9, our track algebra also supports visualizing the contribution of predictions over time to the computation of common performance metrics. One key aspect of evaluating the performance of a predictive model involves choosing which metrics to adopt for assessing its quality — and the decision is often based on the desired outcome of the prediction. For instance, let’s suppose we are interested in detecting the event of a subject standing from a seated position, and we are not really interested in detecting a precise time window for this movement. In case (1) shown in Fig. 14, we would predict correctly one Sit-to-Stand event out of two, after a long period in which no events are detected. If we consider the fact that the classifier correctly predicted that no other events happened over that period, the accuracy score metric would tell us that our model is performing extremely well (beyond 95% correct predictions over time). However, for our purposes, this metric is extremely misleading, since the classifier missed half of the interesting events. Similarly, the Jaccard index computed in case (2) of Fig. 14 would suggest a 50% overlap, which is not very meaningful if we are interested in event detection only. The simple concepts of

\begin{center}
\begin{tabular}{|c|c|c|c|}
  \hline
  A & B & A - B & v1.1 Loss \\
  \hline
  B & A & B - A & v1.1 Gain \\
  \hline
  A \land B & & & v1.1 Comparison \\
  \hline
\end{tabular}
\end{center}

Figure 13: Classifier versioning. Track Xplorer supports a hybrid versioning based on the combination of classic version control systems (e.g. git) and a standardized model definition. Track algebra can also be used to visualize differences between classifier versions.
false positives and false negatives assume indeed subjective meanings in
time-series data classifications. Fig. 14.3 shows how the detection of four
events would by definition generate seven sequences of mispredictions,
with a very negative impact on most performance metrics. Xplorer enables
data scientists to visually assess the suitability of a particular metric, but
also helps them gain insights to develop new, effective performance metrics
tailored to the classification task at hand.

Figure 14: Enhanced performance analysis through visual track algebra.
Track Xplorer helps data scientists reason on the tradeoffs of each
performance metric, helping them decide which ones to adopt. It also
helps them gain insights to develop new, effective performance metrics
tailored to the classification task at hand.

Figure 15: A probabilistic approach to ground truth information. Labels
could be better modeled as a function over time to take into account the
confidence on the event and the uncertainty in precisely identifying its
start and end timepoints.

6.4. A New Model For Activity Labeling

Another aspect affecting performance evaluation is the quality of ground
truth labels, which we simply treated as actions defined through a start
and an end moment in time. As we discussed in our use case, video
annotators cannot always find precise time boundaries for a motor activity
and sometimes they are not even sure if that action has to be labeled or not,
eventually affecting the consequent process of validation. It may be useful
instead to approach ground truth labels with a probabilistic approach,
similarly to what is currently done for classifiers. For instance, a ground
truth label could be defined as a function whose value depends on the con-
fidence of the video annotator, with time boundaries modeled as to keep
into account the uncertainty in determining the exact moment in which the
action started or ended (Fig. 15). This would enable the definition of new
performance metrics and of a continuous version of our current track alge-
bra, with broader possibilities in terms of model analysis and manipulation.

7. Conclusion

We introduce Track Xplorer, a system for interactive visual analysis of
predictions of classifiers modeled to detect events in temporal sensor
data. Our system enables the user to visually and quantitatively analyze
and compare results from multiple classification models, improving the
model development and debugging experience of data scientists. Track
Xplorer couples contextual information such as ground truth labels, expert annotations and event videos together with track visualizations of
predictions through interaction and visual encoding, thereby empowering
users with diverse backgrounds to better interpret, debug, and enhance
the performance of classifiers.

We also introduce an extensible visual algebra over track representa-
tions, enabling composable and rigorous performance comparison and
analysis by data scientists.

We demonstrate the usefulness of our tool through its application in
a collaborative project for developing classifiers to discern motor activity
patterns for scoring the degree of disease progression among Parkinson’s
disease patients. Track Xplorer enables the project team members to
identify early on possible systemic errors in the data, reason about and
pinpoint the causes of misclassifications, and effectively compare the
results of different classifiers and, hence, improve the classification
performances by selecting better models and parameters.

References

IEEE TVCG 20, 12 (2014), 1703–1712. 2, 3
Commun. ACM 26, 11 (1983), 832–843. 2, 3
recreation/storage tradeoff. Proceedings of the VLDB Endowment 8, 12 (2015),
1346–1357. 3
[BSM04] Bade R., Schlichtweg S., Miksch S.: Connecting time-oriented
data and information to a coherent interactive visualization. In Proc. CHI
(2004), ACM, pp. 105–112. 2, 3
sequences using rules, aggregations, and selections. IEEE TVCG (Proc. InfoVis)
(2017). 3
Exploring high-dimensional rss sequences from in vitro selection. In IEEE
BioVis (2013). 2, 3


© 2018 The Author(s)
Computer Graphics Forum © 2018 The Eurographics Association and John Wiley & Sons Ltd.