**Mechanism for the use of machine learning to make suggestions to software users based on prior usage patterns**.

As our software pertains heavily to visualizations, that is the context that will be provided. However, this is applicable to any software from which behavior can be recorded and recommendations can be made.

**Context:**

When opening a visualization program such as MicroStrategy or Tableau, users are typically presented with the following view. Metrics and attributes are available for selection, and the visualization library of possible chart types is visible.

A screenshot of a cell phone

Description automatically generated

Users then build a visualization from the ground up, which can be challenging.

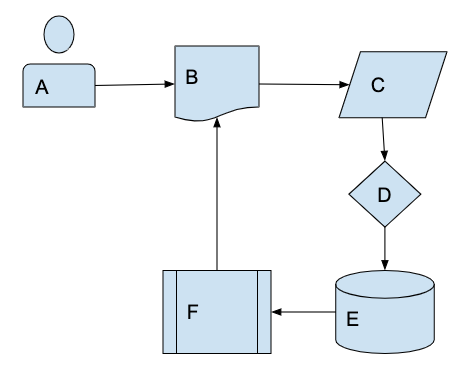
**Solution:**

By storing the prior actions of users with a program to create visualizations a model can be trained to suggest new visualizations. Selecting these models can then be used as model feedback for additional human in the loop training.

A screenshot of a cell phone

Description automatically generated

**Process Flowchart:**



A: User

B: New Document

C: Data

D: Decision (individual history)

E: Data Storage

F: Model

When a user (A) creates a new document (B) and imports data (C), this process (D) checks to see if the user has an individual history and trained model.

Then, using either their individual history or the organizational history for the associated project, the model constructs visualizations using parameters created from the relevant history. These are then displayed in the new document and available to the user.

**Model Implementation**:

E: Data Storage

F: Model

G: All Users

H: Decision: allocate priority to users

I: High Priority Users

J: Low Priority Users

K: Parameterization of Stored Data

A picture containing clock, object

Description automatically generated

The users (G) for the project must be sorted (H) into high priority (I) and low priority (J) use cases. High priority users will have an individual model trained for them on their own history, while low priority users will be considered in aggregate. This reduces the computational expense of maintaining numerous parallel models. This attribute would be kept in storage and considered for model construction and training.

Next, the existing data must be parameterized (K). Metadata for Attributes and Metrics, such as data type (numerical, strings, dates, currency, range, etc) would be notated in binary format for model training.

*Details on parameterization:*

*If a dataset that is frequently used contains currency data and data ranges, and is frequently used in a line chart to show profits over time, having the metadata for the inputs allows a new dataset with new variables names to be imported, the metadata extracted, and the model is then able to suggest a profit chart. Otherwise, no suggestions would be available until the new dataset had been used in visualizations and the model training updated. The generalizations across datasets are a major value-component of this system.*

Last, a model must be trained (F). The specific design of the model is relatively unimportant for the process design: any classification model will work. We initially considered Artificial Neural Networks (ANN) as shown in our board-drawings, but a Bayesian belief network, support vector machine, K nearest neighbors, decision tree, random forest, naïve Bayes, etc., would all likely function. This selection can be made on resource availability, simplicity of set up, and accuracy desired.

The inputs for the model would be the existing visualizations within a project and the associated datasets that were previously parameterized. The outputs, or target classifications, would be the selected visualizations and the associated inputs. For examples, from the note on parameterization above, the classification made would be the following JSON format object:

{

Vis type: Line Chart,

Metric: $amount,

Attribute: Time

}

Implementing this is technically an implementation of several models in parallel to determine each individual item in the output object. The results would then be aggregated so that the top X recommendations are stored and pre-rendered. Those that do not render (for example, if an attribute that is incompatible with a visualization type is recommended) would be discarded, and the remaining results would be available in the ‘suggested visualizations’ portion of the sample screen shown in the Solution § on page 1.