DATA VALIDATION FOR MACHINE LEARNING

**ABSTRACT**

Machine learning is a powerful tool for gleaning knowledge from massive amounts of data. While a great deal of machine learning research has focused on improving the accuracy and efficiency of training and inference algorithms, there is less attention in the equally important problem of monitoring the quality of data fed to machine learning. The importance of this problem is hard to dispute: errors in the input data can nullify any benefits on speed and accuracy for training and inference. This argument points to a data-centric approach to machine learning that treats training and serving data as an important production asset, on par with the algorithm and infrastructure used for learning. In this paper, we tackle this problem and present a data validation system that is designed to detect anomalies specifically in data fed into machine learning pipelines. This system is deployed in production as an integral part of TFX – an end-to-end machine learning platform at Google. It is used by hundreds of product teams use it to continuously monitor and validate several petabytes of production data per day. We faced several challenges in developing our system, most notably around the ability of ML pipelines to soldier on in the face of unexpected patterns, schema-free data, or training/serving skew. We discuss these challenges, the techniques we used to address them, and the various design choices that we made in implementing the system. Finally, we present evidence from the system’s deployment in production that illustrate the tangible benefits of data validation in the context of ML: early detection of errors, model-quality wins from using better data, savings in engineering hours to debug problems, and a shift towards data-centric workflows in model development.

**EXISTING SYSTEM**

While a great deal of machine learning research has focused on improving the accuracy and efficiency of training and inference algorithms, there is less attention in the equally important problem of monitoring the quality of data fed to machine learning. The importance of this problem is hard to dispute: errors in the input data can nullify any benefits on speed and accuracy for training and inference. This argument points to a data-centric approach to machine learning that treats training and serving data as an important production asset, on par with the algorithm and infrastructure used for learning.

**Disadvantages of Existing System:**

1. Accuracy problem
2. Efficiency problem.

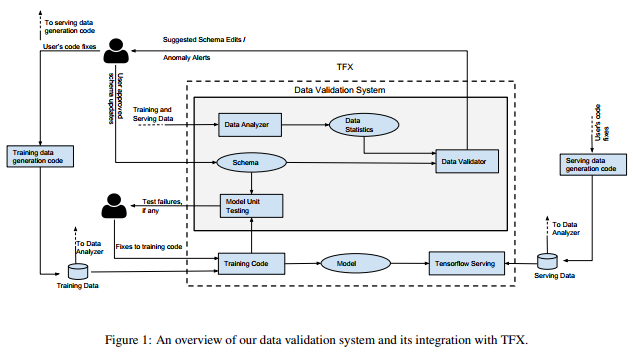
**PROPOSED SYSTEM**

In this paper, we tackle this problem and present a data validation system that is designed to detect anomalies specifically in data fed into machine learning pipelines. This system is deployed in production as an integral part of TFX an end-to-end machine learning platform at Google. In this paper we focus on the problem of validating the input data fed to ML pipelines. The importance of this problem is hard to overstate, especially for production pipelines. Irrespective of the ML algorithms used, data errors can adversely affect the quality of the generated model.

**Advantages of Proposed system:**

1. It is used by hundreds of product teams use it to continuously monitor and validate several petabytes of production data per day.
2. The tangible benefits of data validation in the context of ML: early detection of errors, model-quality wins from using better data, savings in engineering hours to debug problems, and a shift towards data-centric workflows in model development.

**SYSTEM CONFIGURATIONS**

****

# Modules:

**Data Analyzer**

Data Analyzer that computes predefined set of data statistics sufficient for data validation.

**Data Validator:**

Data Validator that checks for properties of data as specified througha Scheme.

**Model Unit Tester:**

Model Unit Tester that checks for errors in the training code using synthetic data generated through the schema.

**Hardware Requirements:**

# Processor - Pentium –IV

* Speed - 1.1 GHz
* Ram - 256 MB
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**Software Requirements:**

* Operating System - Windows XP
* Coding Language - Java