Applications of AI for Predictive Maintenance

AGENDA

- The Learning Platform
- The Team
- Workshop Perspective
- Introduction to Predictive Maintenance
- The XGBoost Algorithm
- Methodology
- Model Accuracy

WORKSHOP PERSPECTIVE

WORKSHOP SUMMARY

- Full-day workshop with hands-on labs
- Applications of different types of deep learning models across data collection, curation, and formatting, based on a real production dataset (Backblaze hard disk DB)
- The workshop covers a variety of methods/algorithms from Tree-based Machine Learning approaches to Deep Learning models.
- You'll learn how to deal with false/true positives in industrial applications by utilizing different methods such as time series prediction and anomaly detection.

WORKSHOP OUTLINE

1. **LAB 1 (2 hrs)** — Training GPU XGBoost Models with RAPIDS for Time Series
2. **LAB 2 (2 hrs)** — Training GPU LSTM Models Using Keras + Tensorflow for Time Series
3. **LAB 3 (2 hrs)** — Training Autoencoder for Anomaly Detection

INTRODUCTION TO PREDICTIVE MAINTENANCE

PREDICTIVE MAINTENANCE

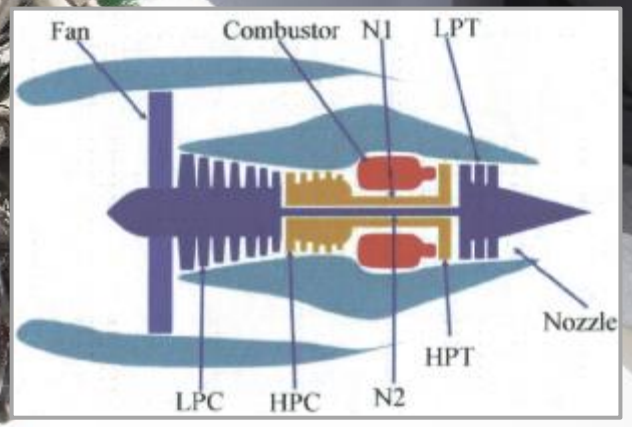
- Schedule-based maintenance provides a regularly scheduled time for part replacements and repairs. A more efficient approach is predictive maintenance: part replacements are driven by data and analytics.
- Predictive maintenance can be divided into several key steps : a) estimation of remaining useful life of critical parts, b) identification of anomalies, c) mapping of anomalies to part failures through service logs.
- Predictive maintenance helps to manage failures and avoid costly unplanned downtime.
- Other benefits include fewer unnecessary repairs, optimized spare parts inventory, and longer lifespan of equipment and parts—ultimately increasing equipment availability and improving operational efficiency.

REMAINING USEFUL LIFE PREDICTION

Reliably predicting the remaining life of mechanical equipment avoids unscheduled maintenance and extends equipment utility.

The health of an engine fleet can be modeled using data from 21 engine sensors (temperature, pressure, rpm, etc.).

State of the art deep learning models are proving capable of predicting the remaining useful life with 35-49% better accuracy than traditional methods.



NASA engine dataset: Turbofan Engine Degradation Simulation Data Set https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

OIL FIELD PREDICTIVE MAINTENANCE

- 60K Oil Wells WW equipped with Electric Submersible Pumps (ESP)
- Average Non-Producing Time (NPT) due to ESP Failure costs > $150K per day per well
  - ML techniques used historically (rule-based, fuzzy logic, traditional ML), but they don't scale
  - >50% False Alarms + detect failures too late
- Deep Learning: 93% detection accuracy with 2 months lead time at 7% False Alarm (for 200 wells)
- At 10% False Positive, DL-based Anomaly Detection yields $300K per well of lost productivity annually

WHAT DOES THIS WORKSHOP COVER?

- Data collection, curation, augmentation, defining performance metrics, training, inferencing, and iterating the whole pipeline to improve performance
- Data curation covers processing raw dataset into Pandas Dataframe, balancing heavily imbalanced dataset, data augmentation, and labeling introduction
- Several Deep Learning models, e.g. : Autoencoders and LSTMs to tackle different aspect of data analysis
- Directions on how to translate data into different categories of deep learning models. Learn how turn data in a classification problem vs. how to approach it from the anomaly detection point of view
- How to use cuDF to increase performance of your models

WORKSHOP STRUCTURE

LAB 1 (2 hrs.)

Training GPU XGBoost models with RAPIDS for Time Series

1. Discuss predictive maintenance, Blackblaze hard drive data, challenges of dealing with large noisy datasets.
2. Scope into short-term ML automation problem.
3. Ingest raw real dataset from GPU production line for XGBoost model.
4. Format DataFrames and into DMatrices to fit into model.
5. Analyze dataset statistics (i.e. false positives, true positives.)
6. Examine XGBoost performance over the model.
7. Learn how to balance imbalanced data and measure relevant KPIs to assess the model.

LAB 2 (2 hrs.)

Training GPU LSTM models using Keras+Tensorflow for Time Series

1. Discuss Recurrent Networks and Long Short-Term Memory (LSTMs).
2. Learn how to mitigate Vanishing Gradient Problem using LSTMs and get familiarized with their cell structure.
3. Learn how to create sequences of data to feed the temporal nature of RNNs.
4. Design different RNN architectures and even create your own model.
5. Learn how to create the confusion matrix and adjust the threshold to different f-1 scores.

LAB 3 (2 hrs.)

Training Autoencoder for Anomaly Detection

1. Focus on a different deep learning technique, which is called anomaly detection.
2. Learn how to use unsupervised learning algorithms like a deep autoencoder network to perform anomaly detection.
3. Learn how to create different autoencoder models in Keras.
4. General discussion on hyperparameter tuning and threshold settings.

XGBOOST ALGORITHM

WHAT IS XGBOOST

- XGBoost stands for eXtreme Gradient Boosting
- Widely known today as a "go to" model when working with structured data
- Large number of XGBoost-based winning entries of Kaggle competitions
- Based on tree ensemble methods such as Random Forests
- XGBoost is used for supervised learning problems, where we use the training data (with multiple features) x to predict a target variable y
- The actual mechanism is discussed next

XGBOOST - CLASSIFICATION TREES

Decision trees

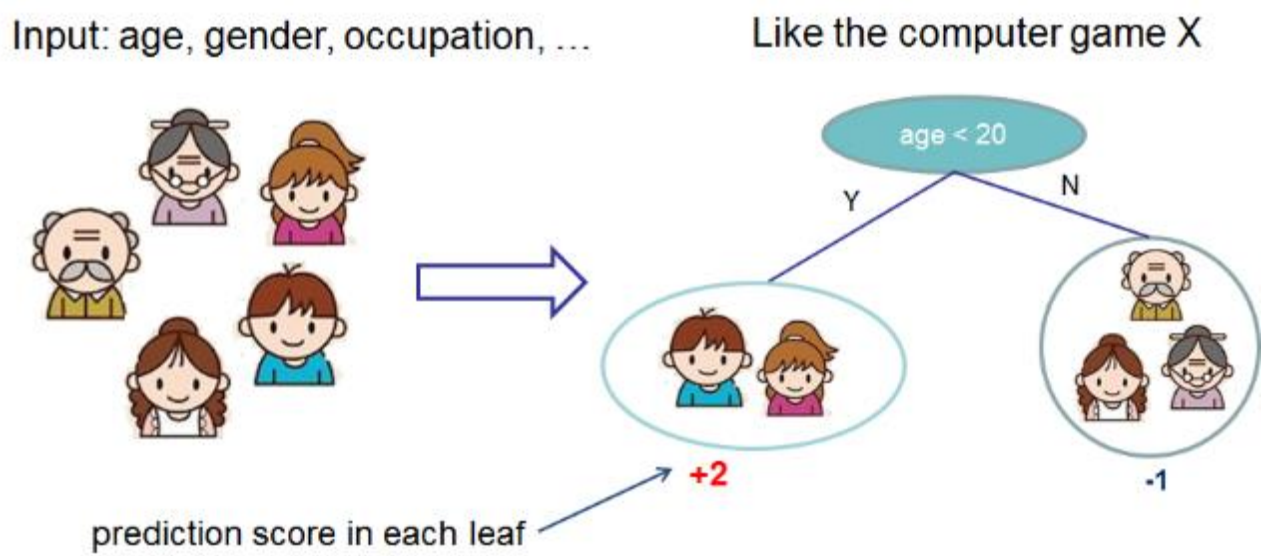


Image credit: XGBoost: A Scalable Tree Boosting System, T. Chen and C. Guestri

XGBOOST - CLASSIFICATION TREES

Decision tree ensembles

Tree ensemble of two trees. The prediction scores of each individual tree are summed up to get the final score

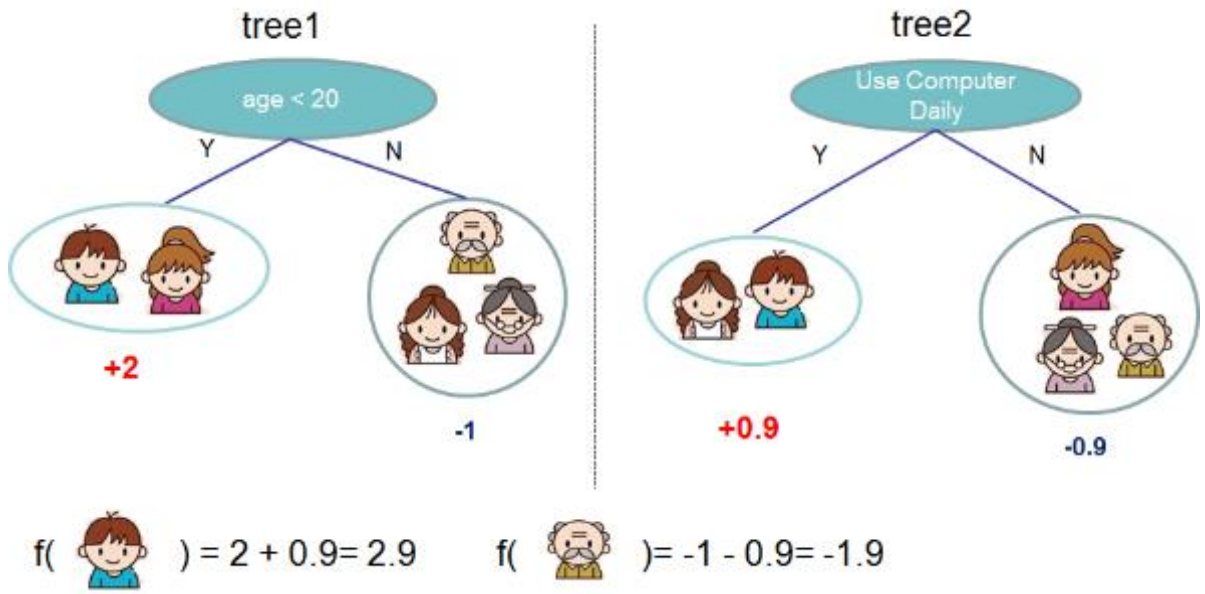
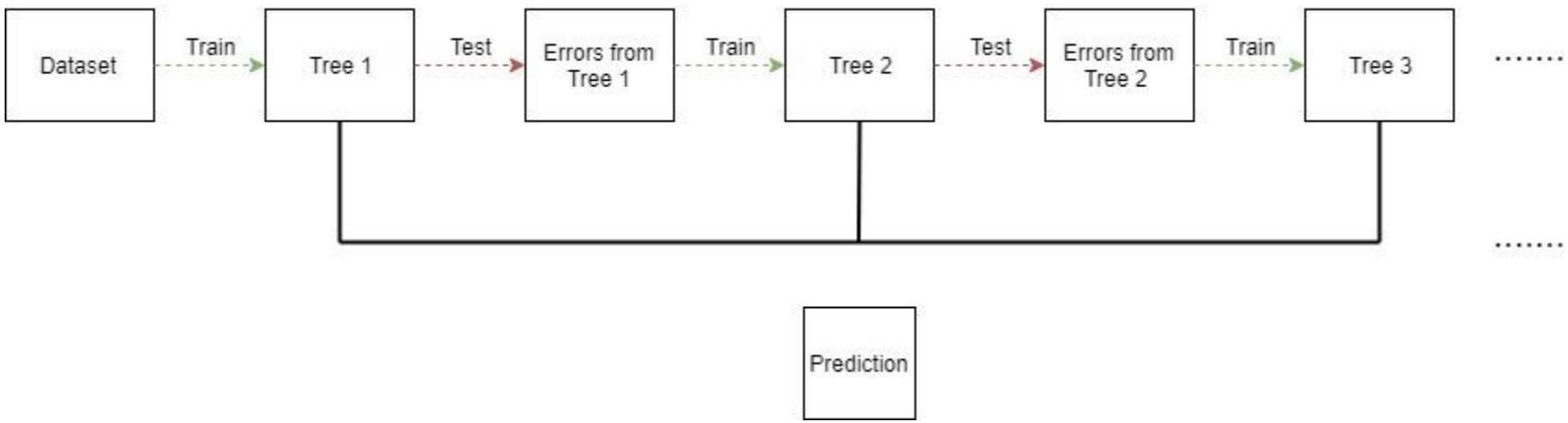


Image credit: XGBoost: A Scalable Tree Boosting System, T. Chen and C. Guestri

XGBOOST - CLASSIFICATION TREES

Decision trees



Repeatedly build new models and combine them into an ensemble. Unlike Random Forests, we build trees one at a time, where each new tree helps to correct errors made by previously trained tree

THE BACKBLAZE DATASET

THE BACKBLAZE DATASET

- *“Each day in the Backblaze data center, a snapshot of each operational hard drive is taken. This snapshot includes basic drive information along with the S.M.A.R.T. statistics reported by that drive. The daily snapshot of one drive is one record or row of data. All of the drive snapshots for a given day are collected into a file consisting of a row for each active hard drive”*

Image credit: Blackblaze website

THE BACKBLAZE DATA FORMAT

https://www.backblaze.com/b2/hard-drive-test-data.html

- **Date** – The date of the file in yyyy-mm-dd format.
- **Serial Number** – The manufacturer-assigned serial number of the drive.
- **Model** – The manufacturer-assigned model number of the drive.
- **Capacity** – The drive capacity in bytes.
- **Failure** – Contains a “0” if the drive is OK. Contains a “1” if this is the last day the drive was operational before failing.
- **2015-2017 SMART Stats** – 90 columns of data, that are the Raw and Normalized values for 45 different SMART stats as reported by the given drive. Each value is the number reported by the drive.

DATA PREPARATION THOUGHTS

https://www.backblaze.com/b2/hard-drive-test-data.html

- **Successful model creation is very dependent on data quality**
- **Time Series sensor data is notoriously ‘jittery’ and has ‘holes’**
- **While we walk through some of the preparation, for Lab 3, we filled in the holes to ensure similar time length sequences using a ‘Forward Fill’ technique**

METHODOLOGY

Exercise

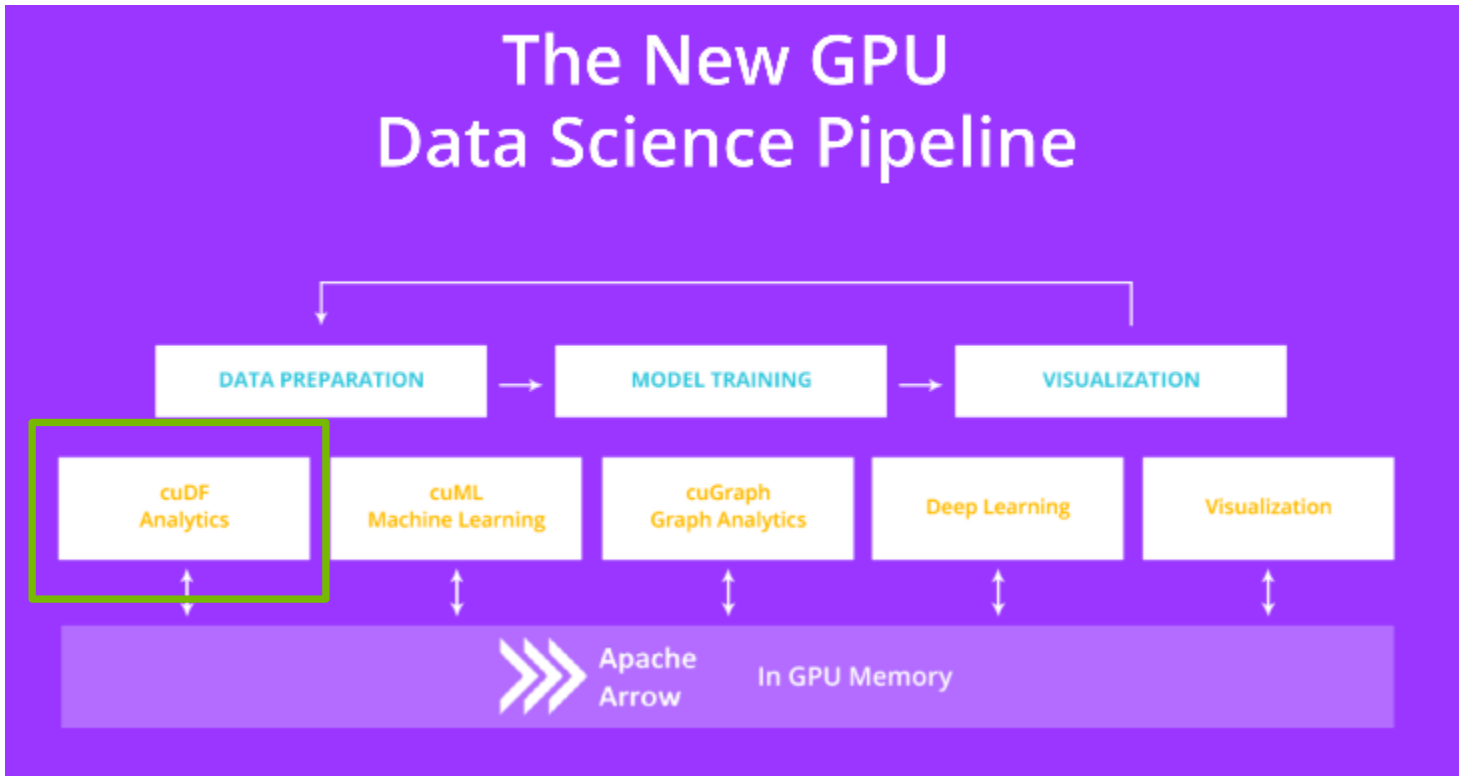
- **Train XGBoost on CPUs using Pandas' DataFrame**
- **Train XGBoost on GPUs using Pandas' DataFrame**
- **Train XGBoost on GPUs using RAPIDS' cuDF**

WHAT IS RAPIDS CUDF?

- The RAPIDS suite of open source software libraries aim to enable execution of end-to-end data science and analytics pipelines entirely on GPUs. RAPIDS uses NVIDIA CUDA primitives for low-level compute optimization and is exposed to GPU parallelism and high-bandwidth memory speed through user-friendly Python interfaces.

- Built based on the Apache Arrow columnar memory format, cuDF is a GPU DataFrame library for loading, joining, aggregating, filtering, and otherwise manipulating data. cuDF provides a pandas-like API that will be familiar to data engineers & data scientists, so they can use it to easily accelerate their workflows without going into the details of CUDA programming.

WHAT IS RAPIDS CUDF?



MODEL ACCURACY

HOW TO MEASURE MODEL ACCURACY

- Accuracy rate provides us with an overview of how well our model is performing:

  - **True Positive (TP)**: test correctly classifies the input as its true class (hard drive failed, and we classified it as fail)
  - **False Positive (FP)**: test incorrectly classifies the input as a different class (hard drive is normal, but we classified it as fail)
  - **False Negative (FN)**: test incorrectly misses to classify the input as the true class (hard drive failed but we classified as normal)
  - **True Negative (TN)**: test correctly misses to classify the input to the class (hard drive is normal, and we classified as normal)

  We need more insights into the accuracy. Especially, since our dataset is imbalanced, and we need to compromise some aspects of accuracy rate in favor of others

HOW TO MEASURE MODEL ACCURACY

Recall is the ability of the model to identify all true positive samples of the dataset

$$recall = \frac{TP}{TP + FN}$$

The ability of the model to identify the relevant samples only

$$precision = \frac{TP}{TP + FP}$$

F1 Score is generally considered a better overall measurement than accuracy when an uneven class distribution exists

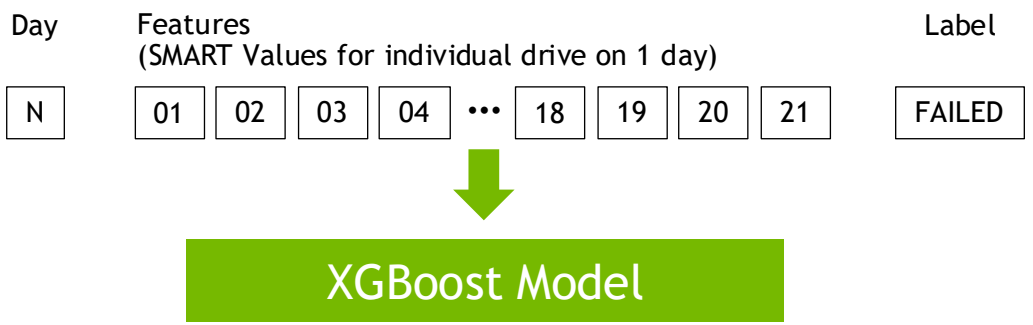
$$F_1\ score = 2 * \frac{precision * recall}{precision + recall}$$

STARTING THE LAB

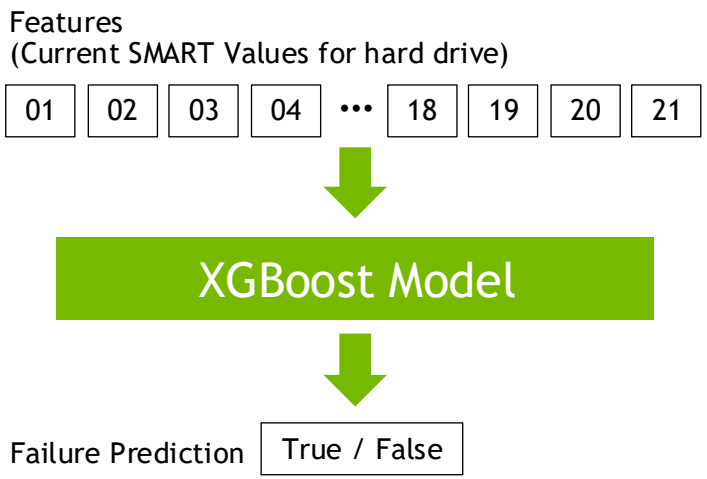
LAB 1 MODEL SUMMARY

Goal: Train ML Model to classify hard drive as “normal” or “failed” based on current SMART features

Training:
Use daily SMART features and “failed” label to train model to identify hard drives that we expect should have failed



Inference:
Pass current SMART data from hard drive through model to infer whether we believe the drive should have failed or not



LAST THOUGHTS

- Labs today will walk you through the standard workflows for Predictive Maintence use cases.
- The accuracy you see won't be optimal.
  - We are using a subset of the data – to speed up workflows.
  - We are using simpler models – to speed up workflows.
- You will notice some delays in reading data and training – anywhere from 4 minutes to 10 minutes. We have added information messages to keep you aware.
- There are a few "Exercises" where you are asked to fill in code. Look for "<<< TO DO >>>" markers. There is an answer available at the end of the task.

SUMMARY

LAB 1 SUMMARY

Training GPU XGBoost models with RAPIDS for Time Series

1. Discuss predictive maintenance, Backblaze hard drive data, challenges of dealing with large noisy datasets.
2. Scope into short-term ML automation problem.
3. Ingest raw real dataset from GPU production line for XGBoost model.
4. Format DataFrames and into DMatrices to fit into model.
5. Analyze dataset statistics (i.e. false positives, true positives.)
6. Examine XGBoost performance over the model.
7. Balance imbalanced data and measure relevant KPIs to assess the model.

WHAT'S NEXT (LAB 2)?

Training GPU LSTM models using Keras + TensorFlow for Time Series

1. Discuss Recurrent Networks and Long Short-Term Memory (LSTMs).
2. Learn how to mitigate Vanishing Gradient Problem using LSTMs and get familiarized with their cell structure.
3. Learn how to create sequences of data to feed the temporal nature of RNNs.
4. Design different RNN architectures and even create your own model.
5. Create the confusion matrix and adjust the threshold to different f-1 scores.