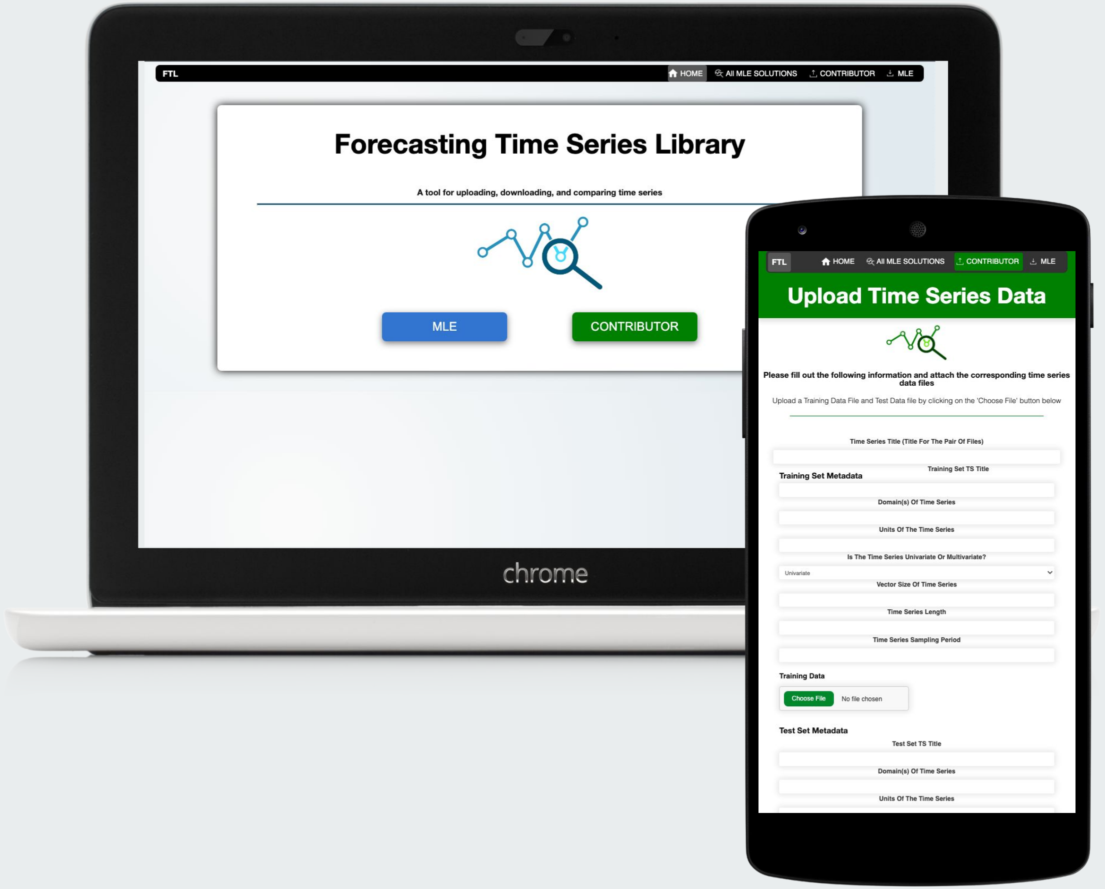
FTL - Forecasting Time Series Library

Team 5: Nate, Hannah, Carlos, Julian


FTL
HOME · All MLE SOLUTIONS · CONTRIBUTOR · MLE

Forecasting Time Series Library

A tool for uploading, downloading, and comparing time series

MLE · CONTRIBUTOR

FTL
HOME · All MLE SOLUTIONS · CONTRIBUTOR · MLE

Upload Time Series Data

Please fill out the following information and attach the corresponding time series data files

Upload a Training Data File and Test Data file by clicking on the 'Choose File' button below

Time Series Title (Title For The Pair Of Files)

Training Set Metadata

Training Set TS Title

Domain(s) Of Time Series

Units Of The Time Series

Is The Time Series Univariate Or Multivariate?

Univariate

Vector Size Of Time Series

Time Series Length

Time Series Sampling Period

Training Data

Choose File · No file chosen

Test Set Metadata

Test Set TS Title

Domain(s) Of Time Series

Units Of The Time Series

Problem Statement

-Contributors need a place to store their training and test sets

-Machine Learning Engineers (MLE's) need a place to store their forecasting solutions

-MLE’s also need to be able to view how their results compare with other forecasting solutions uploaded by other MLE’s

FTL Overview

-Web app is split into two main sections: **Contributor** and **MLE**

-Based on what role a user is at that time, they can click on that path and then upload/download the files they need

- Contributor uploads training set, test set, and specifies forecasting task

- MLE downloads training set, uploads solution, receives error analysis and accuracy comparison

Contributor Path

Need a page to upload their training sets and test sets

User is expected to provide 2 files: a training set and test set along with metadata

After uploading files, user gets a notification that their submission was successful or not

User should be able to view all the MLE solutions that have been uploaded -> Nav bar

MLE Path

Need a page where MLE can upload their forecasting solution

- After upload, should be taken to page to view how their solution compares with the test set

MLE user should easily be able to view all MLE solutions split up by training set

System Modifications

- Login System
    - Originally planned to have an account system with secure username/passwords and a google account sign in option

    - Error graph’s and original comparison plan

-Originally tried to sort accuracy by comparing error metric’s to each other instead of between different MLE user solutions

-Planned to have multiple graphs with all the different metrics instead of just one

MLE Solution Error Calculations

MLE needs a reliable way to compare their forecast to a result

Should be able to see the total sorted data, and receive a graph that details the difference between test set and forecast

The MLE can then take that data and use it however they see fit

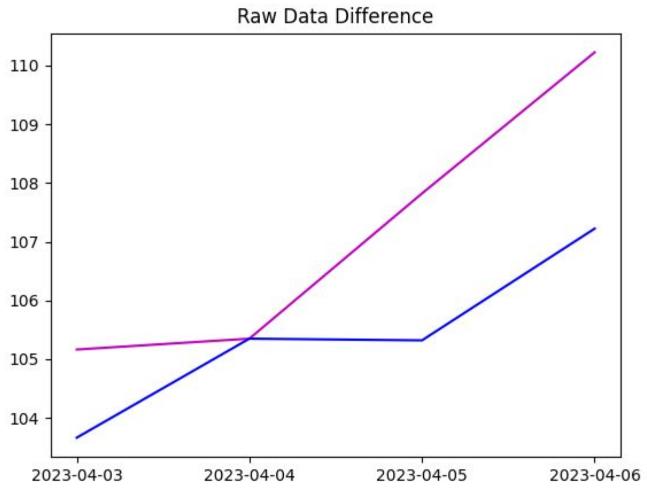
Error graph and metrics

-Used 6 algorithms

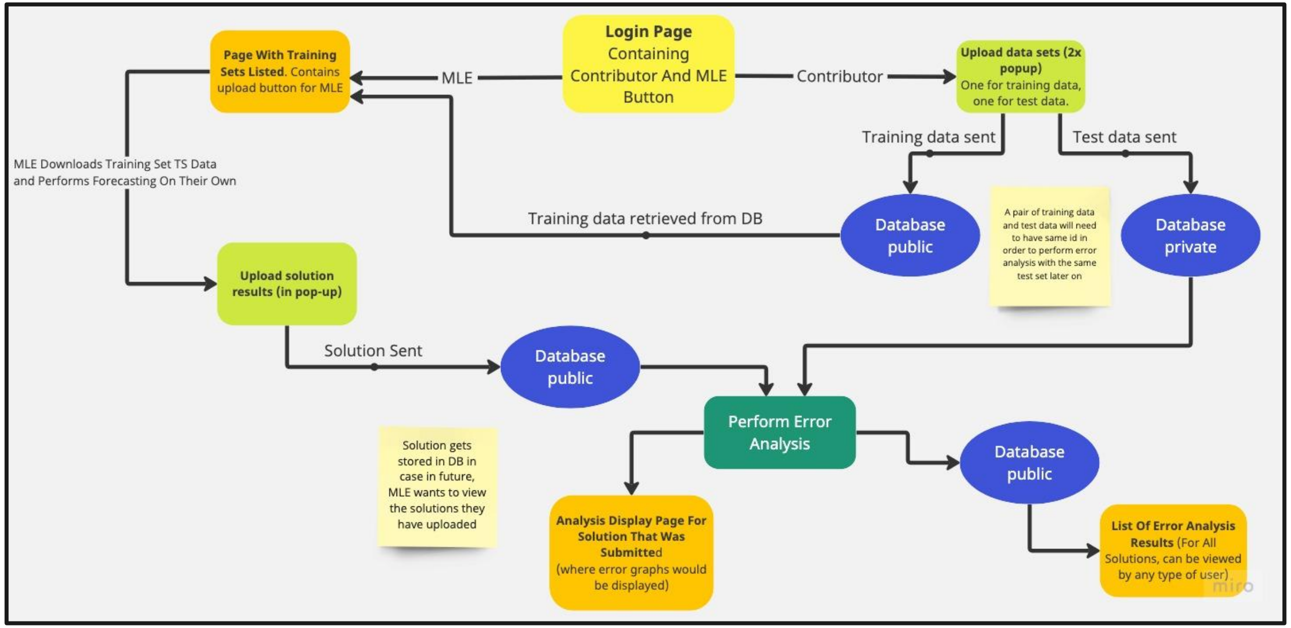
- MAE, MAPE, SMAPE
- MSE, RMSE
- Correlation-Coefficient

-User is able to see a graph comparing raw data for test set to the forecast result, with the ability to set a parameter an limit variables

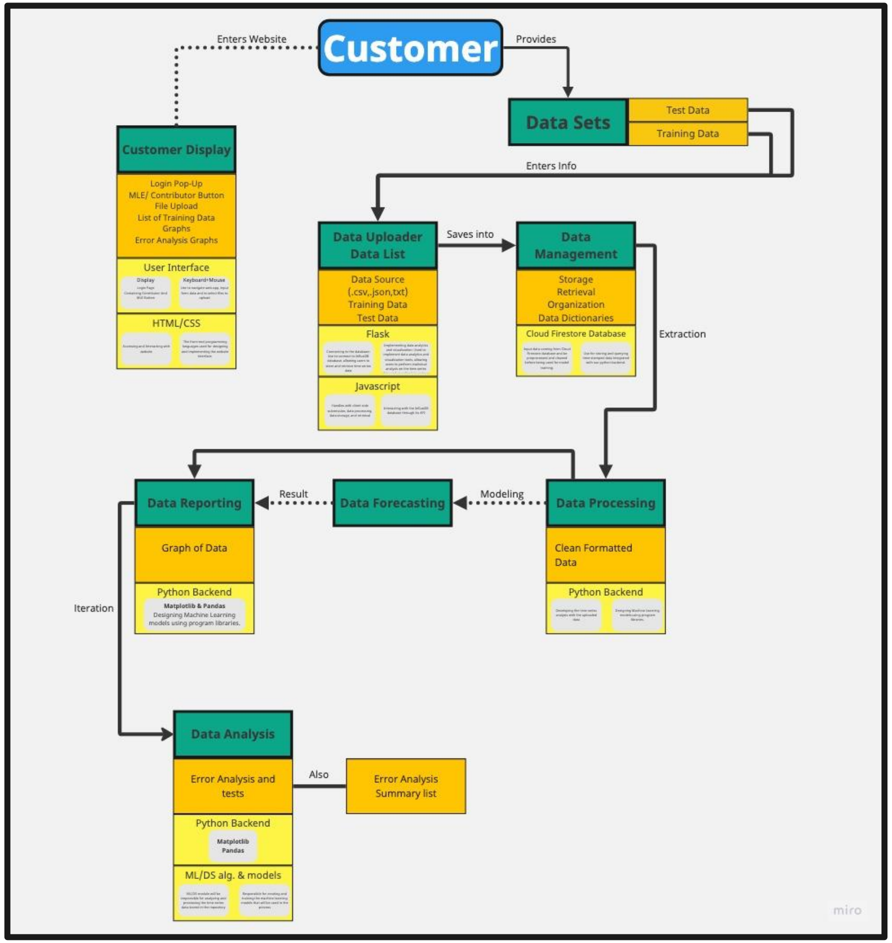
-Error metrics stored in the Flask database



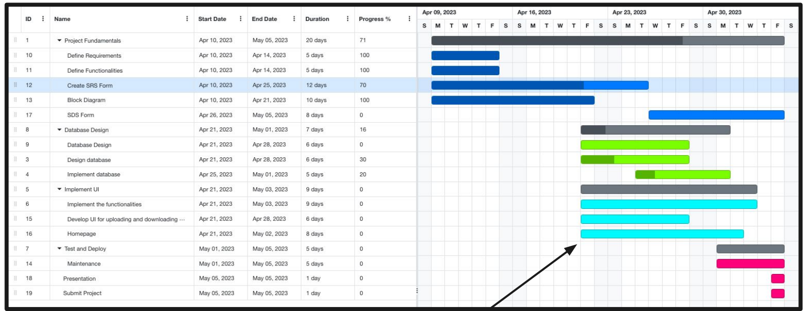
Dynamic Diagram


Login Page Containing Contributor And MLE Button
MLE
Contributor
Page With Training Sets Listed. Contains upload button for MLE
Upload data sets (2x popup) One for training data, one for test data.
Training data sent
Test data sent
MLE Downloads Training Set TS Data and Performs Forecasting On Their Own
Training data retrieved from DB
Database public
A pair of training data and test data will need to have same id in order to perform error analysis with the same test set later on
Database private
Upload solution results (in pop-up)
Solution Sent
Database public
Perform Error Analysis
Database public
Solution gets stored in DB in case in future, MLE wants to view the solutions they have uploaded
Analysis Display Page For Solution That Was Submitted (where error graphs would be displayed)
List Of Error Analysis Results (For All Solutions, can be viewed by any type of user)
miro

Static Diagram

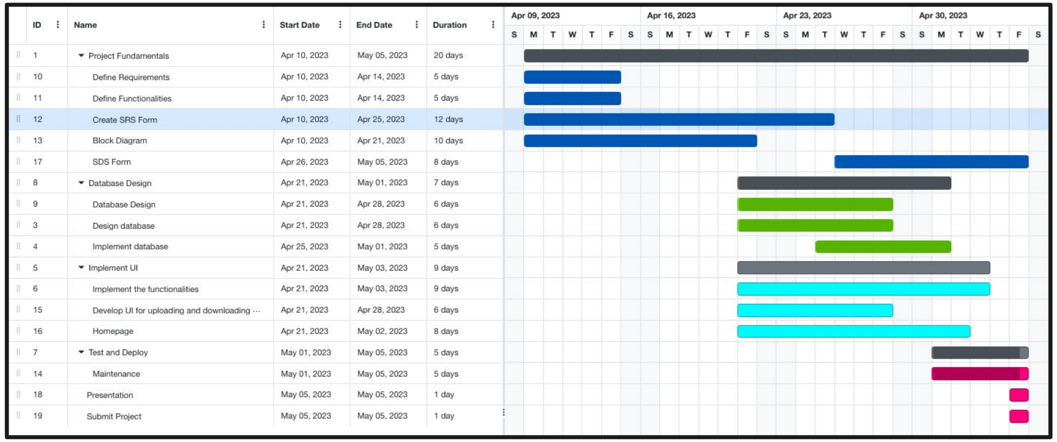


High-level schedule.



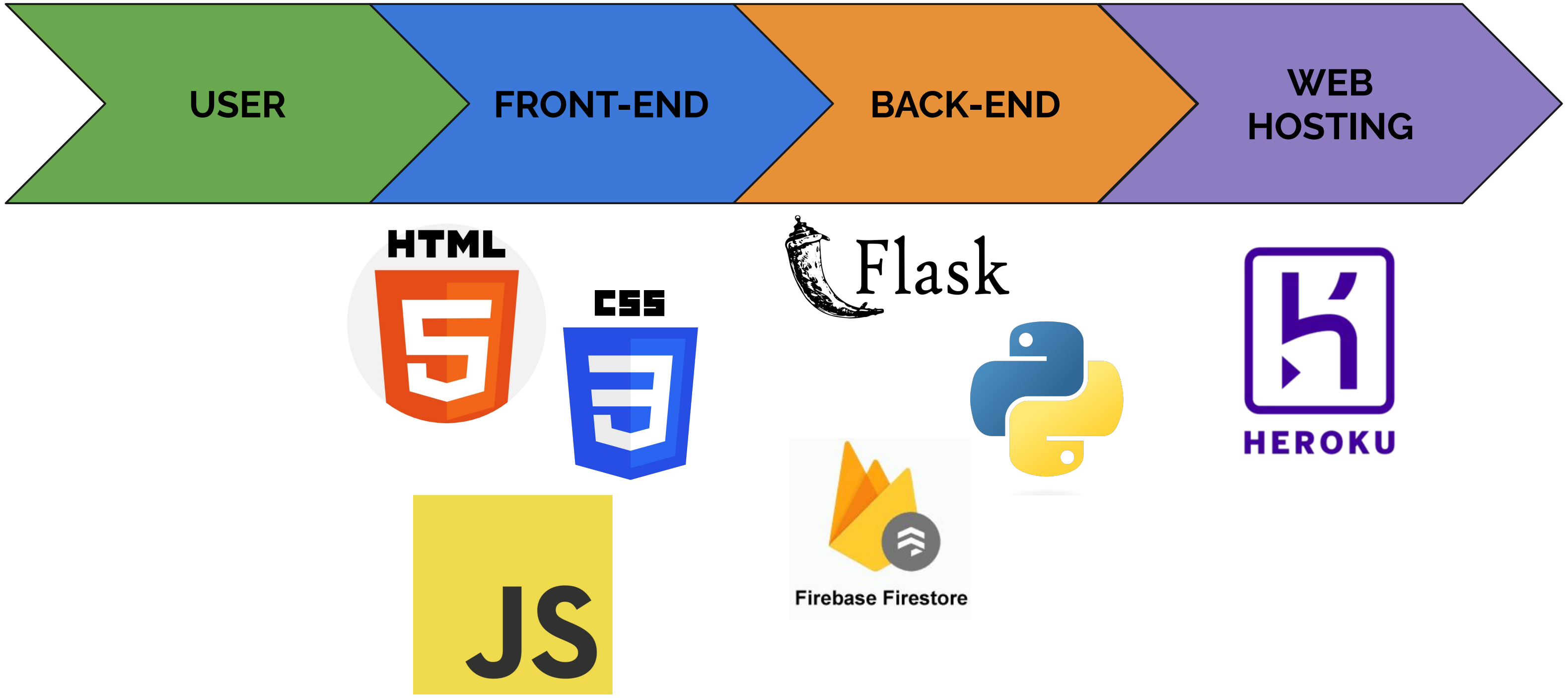
ID	Name	Start Date	End Date	Duration	Progress %
1	Project Fundamentals	Apr 10, 2023	May 05, 2023	20 days	71
10	Define Requirements	Apr 10, 2023	Apr 14, 2023	5 days	100
11	Define Functionalities	Apr 10, 2023	Apr 14, 2023	5 days	100
12	Create SRS Form	Apr 10, 2023	Apr 25, 2023	12 days	70
13	Block Diagram	Apr 10, 2023	Apr 21, 2023	10 days	100
17	SDS Form	Apr 26, 2023	May 05, 2023	8 days	0
8	Database Design	Apr 21, 2023	May 01, 2023	7 days	16
9	Database Design	Apr 21, 2023	Apr 28, 2023	6 days	0
3	Design database	Apr 21, 2023	Apr 28, 2023	6 days	30
4	Implement database	Apr 25, 2023	May 01, 2023	5 days	20
5	Implement UI	Apr 21, 2023	May 03, 2023	9 days	0
6	Implement the functionalities	Apr 21, 2023	May 03, 2023	9 days	0
15	Develop UI for uploading and downloading …	Apr 21, 2023	Apr 28, 2023	6 days	0
16	Homepage	Apr 21, 2023	May 02, 2023	8 days	0
7	Test and Deploy	May 01, 2023	May 05, 2023	5 days	0
14	Maintenance	May 01, 2023	May 05, 2023	5 days	0
18	Presentation	May 05, 2023	May 05, 2023	1 day	0
19	Submit Project	May 05, 2023	May 05, 2023	1 day	0

1st Progress Meeting with Client



ID	Name	Start Date	End Date	Duration
1	Project Fundamentals	Apr 10, 2023	May 05, 2023	20 days
10	Define Requirements	Apr 10, 2023	Apr 14, 2023	5 days
11	Define Functionalities	Apr 10, 2023	Apr 14, 2023	5 days
12	Create SRS Form	Apr 10, 2023	Apr 25, 2023	12 days
13	Block Diagram	Apr 10, 2023	Apr 21, 2023	10 days
17	SDS Form	Apr 26, 2023	May 05, 2023	8 days
8	Database Design	Apr 21, 2023	May 01, 2023	7 days
9	Database Design	Apr 21, 2023	Apr 28, 2023	6 days
3	Design database	Apr 21, 2023	Apr 28, 2023	6 days
4	Implement database	Apr 25, 2023	May 01, 2023	5 days
5	Implement UI	Apr 21, 2023	May 03, 2023	9 days
6	Implement the functionalities	Apr 21, 2023	May 03, 2023	9 days
15	Develop UI for uploading and downloading …	Apr 21, 2023	Apr 28, 2023	6 days
16	Homepage	Apr 21, 2023	May 02, 2023	8 days
7	Test and Deploy	May 01, 2023	May 05, 2023	5 days
14	Maintenance	May 01, 2023	May 05, 2023	5 days
18	Presentation	May 05, 2023	May 05, 2023	1 day
19	Submit Project	May 05, 2023	May 05, 2023	1 day

Tech Stack


USER
FRONT-END
BACK-END
WEB HOSTING
HTML
CSS
JS
Flask
Firebase Firestore
HEROKU

Flask

-Lightweight Python Framework, already had experience with compared to Django

-Page routing, rendering HTML pages

-Parsing data from files

-Communicating with Firebase Firestore Database

Team Roles

Nate

-Backend/Database

-Flask, Firebase, Python

- Slides, SRS

Hannah

-Frontend/interfacing/design

- HTML, JavaScript, CSS

-SDS, Slides

Carlos

-Backend/Error

-Python, Pandas, Matplotlib

-SDS, Slides

Julian

-Frontend/Design/Backend

-HTML, CSS, Python

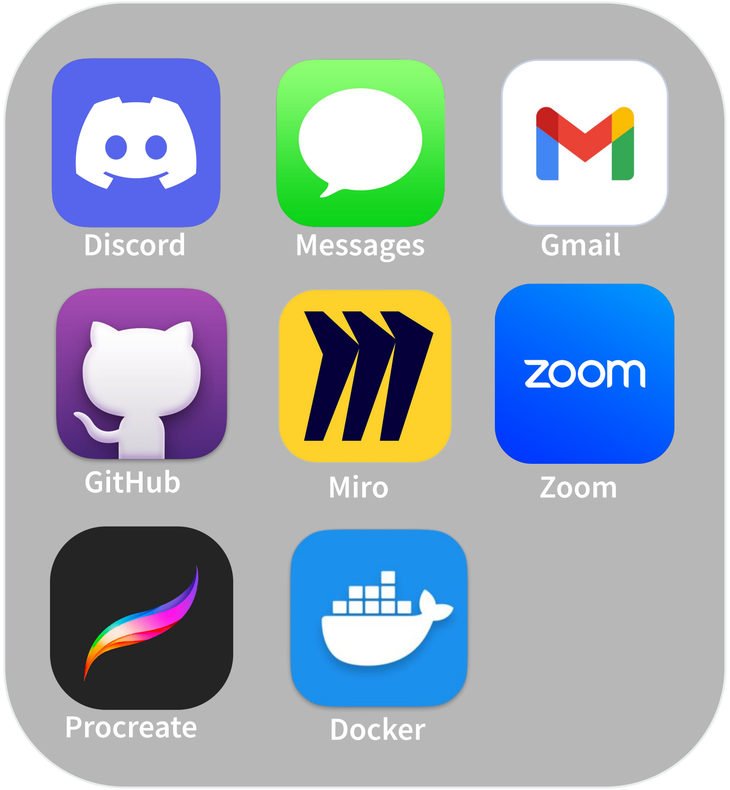
-SRS, Slides, diagrams

Tools & Group Communication

-Text group chat: fast and easy instant messaging

-Discord: Sharing files/images, code questions

-Meetings in science library and EMU



Questions We Asked

-How should system handle training set + test set? -> require 2 files from user

-Clarifying Time Series Set/File upload -> For example with stock prices, contributor would upload each file by company, those stock data files would make up a TS set

-How should error analysis be displayed and stored?->Sent to a graph for the user, and stored in a table in the database

-How should tables be structured/sorted? -> Each table refers to one TS training set, one error column will be used as the sorting column

Problems we ran into

-Meetings ending with unclear results, resulted in code that would get written then deleted/changed

-Limited synchronized meetups and work sessions resulted in slow communication

-Hands off approach meant that there was a lack of understanding about who was doing what

-Many developers did not have a good understanding of the systems/languages others were working in, resulting in a difficulty when it came to testing and combining code

Our solutions

-Started meeting more regularly, in order to make progress on the project

-Started setting predetermined work times, even without direct collaboration, being able to update each other constantly made it easier to work

-Working together more made it easier to use each others systems despite lack of knowledge

-Priorities were established, which made it easier to build around the core of the project

What would we have done differently?

-Established project goals, individual parts, and a rough draft of a schedule early on

-Set regular meeting times to code through chunks of the project together

-Communicated more about specifics of code, so all developers could be on the same page, and be able to understand each others code

Live Demo

Time Series Data Collection and Analysis