Time Series Forecasting & Benchmarks Repository
Project Plan

Aiden Duval, Zane Globus-O'Harra, Erin Stone, Kareem Taha, Zachary Weisenbloom
Team 1 ("zeakz")
https://github.com/zfgo/CS_422_p1
CS 422 Spring 2023 Project 1 – 2023-05-05

1 Revision History

Date	Author	Description
2023-04-10	Aiden D	Add meeting notes.
2023-04-13	Aiden D	Add meeting notes.
2023-04-17	Aiden D	Add meeting notes.
2023-04-20	Aiden D	Add meeting notes.
2023-04-24	Aiden D	Add meeting notes.
2023-04-27	Aiden D	Add meeting notes.
2023-05-01	Aiden D	Add meeting notes.
2023-05-02	Zane G-O	Add 2, 3.1, 4, 5, 6, and 7.
2023-05-04	Zane G-O	Add 3.2.

2 Application Overview

The fields of data science (DS), machine learning (ML), and artificial intelligence (AI) have exploded in recent years. The engineers in these fields analyze many types of data, creating models and algorithms to classify, categorize, predict, etc. One type of data is the time series (TS), which has time, t, as the independent variable, and the magnitude of some other variable, x, with respect to the time domain. The recording of this data happens at equally spaced instances of time, Δ. TS forecasting, using models created by DS or ML engineers, aims to determine an estimate for the value of $x_{N+\Delta}$.

With the rise in popularity of DS, ML, and AI, so too has TS forecasting risen in popularity, especially with an abundance of data from Internet of Things (IoT) devices. However, many engineers in this field find themselves going back and forth across the internet, spending a great deal of time searching for places to download TS data.

Despite the importance and rise of popularity of TS forecasting and analysis, there is no central repository where TS data and benchmarks are stored. The DS and ML engineers need such a location where they can obtain data to test their models, and then compare the results of their models against other methods.

Another issue is that the current repositories that hold TS data are spread across the web, and the data that they hold has many different characteristics. It would hence be desirable to create a central repository that would fix these issues and be a main hub for TS forecasting data.

3 Management Plan

3.1 Initial Plans and Organization

Our team separated the project into client-side and server-side components. We divided the team up by component, with two people working on the client-side, and three people working on the server-side. Within each of these two components, we created subcomponents, or modules, and assigned work on these modules to each team member. The layout of the components and modules is as follows:

- Client components:
    - Login/sign up page.
    - Contributor pages:
        - Upload forecasting task and TS training and test data sets.
    - Participant pages:
        - Download TS training data set related to a forecasting task.
        - Upload solution data set for a given forecasting task and get comparison metrics with the task's TS test data.
- Server components:
    - Database:
        - User accounts.
        - Forecasting tasks:
            - Related TS data sets.
    - Convert file format module.
    - Comparison metrics module.

In-depth explanations of these components can be found in SDS section 3.1, as well as SDS section 4.

On the client side, we designated the UI design and layout to Erin, with Aiden acting as the project manager, as well as helping Erin with integration of the client and server components. On the server side, we designated Zach to work on the upload and download functionalities of the pages and the file format conversion, Zane to work on the database design and the file format conversion, and Kareem to work on the comparison metrics. We decided that these designations would not be rigid, and

Our plan was to meet frequently (twice a week on Mondays and Thursdays) to distribute tasks, get input from other team members, and decide on goals to work on before the next meeting. These meetings were initially in-person while we were designing our system and figuring out what software systems to use to create our system, but they moved to Discord calls later on in the process. This allowed us to share our screens, and increased the amount of progress we were able to make on the software components. We used Discord as our main form of communication, where we asked each other questions or got help between our meetings.

We used GitHub to keep track of our code, and committed frequently after we completed a module or submodule. We also created a shared Google Drive folder to store all our documentation and diagrams.

3.2 Modified Plans and Organization

As the project progressed, it became clear that we would not be able to accomplish everything that we had laid out in our initial design. We decided to scrap the login/sign up page, simply

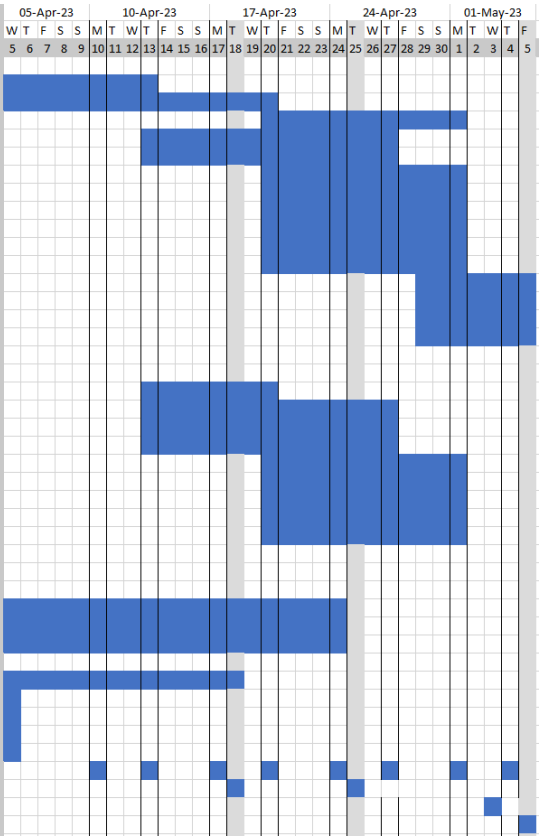
replacing it with a “user choose” functionality, where the user would choose to be either a contributor or a participant. We also only used the convert file format module to convert files so that they could be properly accepted by the comparison metrics module. Instead of allowing users to download files in any format they desired, we made it so that users would download a ZIP archive of the training files that had been uploaded by the contributor.

We also changed how our team was organized. Erin had created the UI design and frontend almost entirely on her own. Kareem had worked almost exclusively on the comparison metrics module. Zane wrote the SRS and SDS almost entirely on his own, as well as adding details and context to the Project Plan document and writing other documentation. Aiden noted down the meeting notes in the Project Plan document. Zane and Zach had spent most of their time doing pair programming working on the upload and download functionality with the Django server-side code. When we got to integrating the frontend and backend, there were several long days spent in Discord calls where Erin and Zach worked on integration while Zane worked on updating the SDS and SRS to reflect the current state of our system.

4 Project Timeline

To make consistent progress, we created a timeline, dividing our tasks into different categories, and mapping out how much time we thought each task would take, as well as when we thought that those tasks should be completed by. This timeline is available in Timeline.xlsx, as well as shown below:

Task	assigned to	start	end
Backend & functionality			
decide on framework	Team	2023-04-05	2023-04-13
set up a DB	Zane & Zach & kareem	2023-04-05	2023-04-20
download data for testing	Zane & Zach	2023-04-20	2023-05-01
functionality to save user account information	Zane & Zach	2023-04-13	2023-04-27
functionality for login screen	Zane & Zach & Team	2023-04-13	2023-04-27
Store/download .csv/database files	Zane & Zach & Team	2023-04-20	2023-05-01
functionality to compare test/solution data accurac	Kareem	2023-04-20	2023-05-01
create method for contributor to upload TS data & f	Zane & Zach & Team	2023-04-20	2023-05-01
create method for participant to download forecast	Zane & Zach & Team	2023-04-20	2023-05-01
create method for participant to upload solution da	Zane & Zach & Team	2023-04-20	2023-05-01
Integration and Debugging			
Integrate frontend/backend	Team	2023-04-29	2023-05-05
debug views	Team	2023-04-29	2023-05-05
debug comparison metrics	Team	2023-04-29	2023-05-05
debug file conversion	Team	2023-04-29	2023-05-05
Frontend/UI design			
create UI design	Erin	2023-04-13	2023-04-20
Create login page	Aiden & Erin & Team	2023-04-13	2023-04-27
create contributor page	Aiden & Erin & Team	2023-04-13	2023-04-27
create participant page	Aiden & Erin & Team	2023-04-13	2023-04-27
present statistical comparisons (and graphs?) to par	Aiden & Erin & Team	2023-04-20	2023-05-01
create page for contributor to upload TS data & fore	Aiden & Erin & Team	2023-04-20	2023-05-01
create page for participant to download forecasting	Aiden & Erin & Team	2023-04-20	2023-05-01
create page for participant to upload solution data t	Aiden & Erin & Team	2023-04-20	2023-05-01
create admin page	Aiden & Erin & Team	2023-04-20	2023-05-01
Desgin			
SRS Document	Zane & Team	2023-04-05	2023-04-24
SDS Document	Team	2023-04-05	2023-04-24
design SW architecture	Team	2023-04-05	2023-04-24
Organization & Outputs			
distribute tasks	Team	2023-04-05	2023-04-17
set up git repository	Zane	2023-04-05	2023-04-05
set up trello	Aiden	2023-04-05	2023-04-05
set up google drive	Aiden	2023-04-05	2023-04-05
set up discord	Erin	2023-04-05	2023-04-05
Team Meetings	Team	recurring	
Progress Meetings	Team	recurring	
Project presentations	Team	2023-05-03 to 2023-05-05	
project due	Team	3-05-05 to 2023-05-09 ??	



5 Building Plan

The first steps in our project involved figuring out the project requirements and deciding on a software environment to use. Once we had an idea of the use cases and had decided on using Django to develop our app, we were able to break down our software into modules and assign those modules to members of the team to work on.

The second stage of our development was focused on implementing the modules. We often worked as pairs so that we would always have two people working on the code. This would turn out to be helpful because it would make learning Django much easier, as no one on our team had any experience with it. We would mostly track our progress using Discord, updating the other team members as to what we were working on and how much we had progressed.

Lastly, we planned on integrating the frontend and the backend and fixing any bugs or missing functionality. This was one of the major faults in our plan. We had read that Django made integrating the frontend and backend easy, so we decided to have Erin and Aiden develop the UI almost completely separately from the work that Zach, Zane, and Kareem were doing on the server and database side. This would result in Erin having to rewrite almost all the frontend functionality so that it would work with Django.

6 Monitoring and Reporting

To monitor our work, we would communicate frequently on Discord, as well as giving more formal updates during our twice-a-week meetings. We used the project timeline to keep each other accountable for the work that we were all doing and did our best to update the timeline to accurately keep track of our work.

We created a Trello board, but its workflow didn't work very well for our group, so we abandoned it.

7 Rationale

Our approach was very similar to an Agile methodology. Our meetings were akin to sprint reviews, and the time between meetings was like a mini sprint. This method works for

8 Meeting Outcomes

These meeting outcomes do not include our progress meetings with Dr. Flores.

8.1 2023-04-10

8.1.1 Goals

- Have the timeline spreadsheet ready.
- Come with a general idea of making a web app, Do research on designs, languages, and infrastructures.
- Research Flask and MongoDB
- Decide what to use for each part of the web app.
- Start the SRS to get an understanding of the requirements.

8.1.2 Questions

- More details on what each user would be able to do and what it would look like?
- Do we need to host our web app?
- Do we need to do any modeling of our data?

8.1.3 Outcomes

- Broke down the general idea of the app.
- Created a new time to meet.
- Established previous experiences and strengths to try and align people to Teams.
- Initial team organization
    - Backend: Zach, Zane, Kareem
    - Frontend: Aiden, Erin
- Create the SRS document.

8.2 2023-04-13

8.2.1 Goals

- Have skeletal implementation of perhaps Django web app.
- Finish SRS by the next meeting.
- Start work on the SDS.

8.2.2 Questions

No questions raised.

8.2.3 Outcomes

- Decided on using Django as a framework for making the web app.
- Created the SDS document.

8.3 2023-04-17

8.3.1 Goals

- Create more pages and models for user choose and additional time series and sets options.
- Create a working database.

8.3.2 Questions

No questions raised.

8.3.3 Outcomes

- Created a working Django upload document frontend page model.
- Established additional time to lay out getting a working database setup.
- SRS almost complete (only a couple sections remain).

8.4 2023-04-20

8.4.1 Goals

- Create more front-end features for user and design elements.
- Enable document reading for backend processing and collecting data/

8.4.2 Questions

No questions raised.

8.4.3 Outcomes

- Database Django interface understanding achieved.
- Created more html templates and website art.
- SRS completed.

8.5 2023-04-24

8.5.1 Goals

- Establish a more successful handshake between file upload/download with backend.
- Create more database processing scripts and models.

8.5.2 Questions

No questions raised.

8.5.3 Outcomes

- Integrated userchoose and file upload html templates into working Django frontend pages.

8.6 023-04-27

8.6.1 Goals

- Fine tune document processing from frontend to backend to maintain a more successful manipulation of time series sets.
- Finish backend models and scripts for all the time series metadata.

8.6.2 Questions

No questions raised.

8.6.3 Outcomes

- Established database handling and with python scripts to arrange series by different values.
- Hosted website (Frontend only!!! No backend functionality) to https://pages.uoregon.edu/estone3/webapp/userchoose.html.
- Created more entries for endpoint user to configure uploaded data.

8.7 2023-05-01

8.7.1 Goals

- Have functional frontend web app working by the deadline.
- Improve file processing and handshake between frontend and backend.
- Create presentation and have it ready by class on 2023-05-03.

8.7.2 Questions

No questions raised.

8.7.3 Outcomes

- Created presentation slides document.