# Repository and Mining of Temporal Data

**Team Members:**

| | |
|---|---|
| Jessica Nguy | jnguy2014@my.fit.edu |
| Siomara Nieves | snieves2014@my.fit.edu |

**Faculty Sponsor and Client:**

Philip Chan pkc@fit.edu

### Meetings With the Client:

Jan 12

### Goals:

The inspiration behind our project is the fact that data collection and analysis requires multiple steps. Oftentimes users seeking to analyze their data must collect files from several separate sites before comparing their data with other data files. Our project aims to simply these issues by creating a program that analyzes data and compares it to collected data already stored in the system. Data analysis consists of: determining significant change; comparing the target variable as determined by the Data Customer and comparing it to files already in the database to determine the top variables that affected the target variable and the amount of correlation the target variable has with the top-k variables; and predicting the next timestamp based on the data. Data Providers will be able to upload and store files into the program, with a tag-system in place to keep track of files. The program will be able to recognize granularity problems and be able to handle them. Speed is an important factor to analyzing data and will be a focus for our project.

### Approach:

The project is meant to answer the following questions:

Is there a significant change in the target variable? The program analyzes the data that the user uploads, and returns a visualization of the data as it changes over time. The program also indicates areas of interest, such as large changes over a short period of time.

What are the top-k variables that affect the target? The program compares the data that the user uploads with the data already stored in the system. The user provides several search tags that allow the program to search through the system for data that correlate to the tags to determine areas of possible interest that have affected the target variable.

What is the value of the next timestamp? The culmination of all the analysis results in the final question being answered, that is, predicting what will occur in the next timestamp based on data calculated in the second question. This allows for users to see a prediction for what will occur in the future based on what had happened in the past.

The answers for these questions have been programmed with Python and the test cases that have been run are yet to be part of the website application that is being programmed using Django. The web application will consist of a landing page that will redirect to the option of uploading a file, analyzing the file, showing the results, creating an account to save the results, and search through the database to see and download files of official data. At the end of the data analysis, there will be an option to download the resulting files.

## Novel Features:

Novel features include simplifying data analysis portions by collecting, storing, and analyzing data in one singular place, allowing multiple questions to be answered. The concept allows for users to save time locating files. The program will be able to convert uploaded data into a specific format for users and the program to use. There is visualization for each question that the program answers. There is an option to download the results of the analysis when the program is done.

## Technical Challenges:

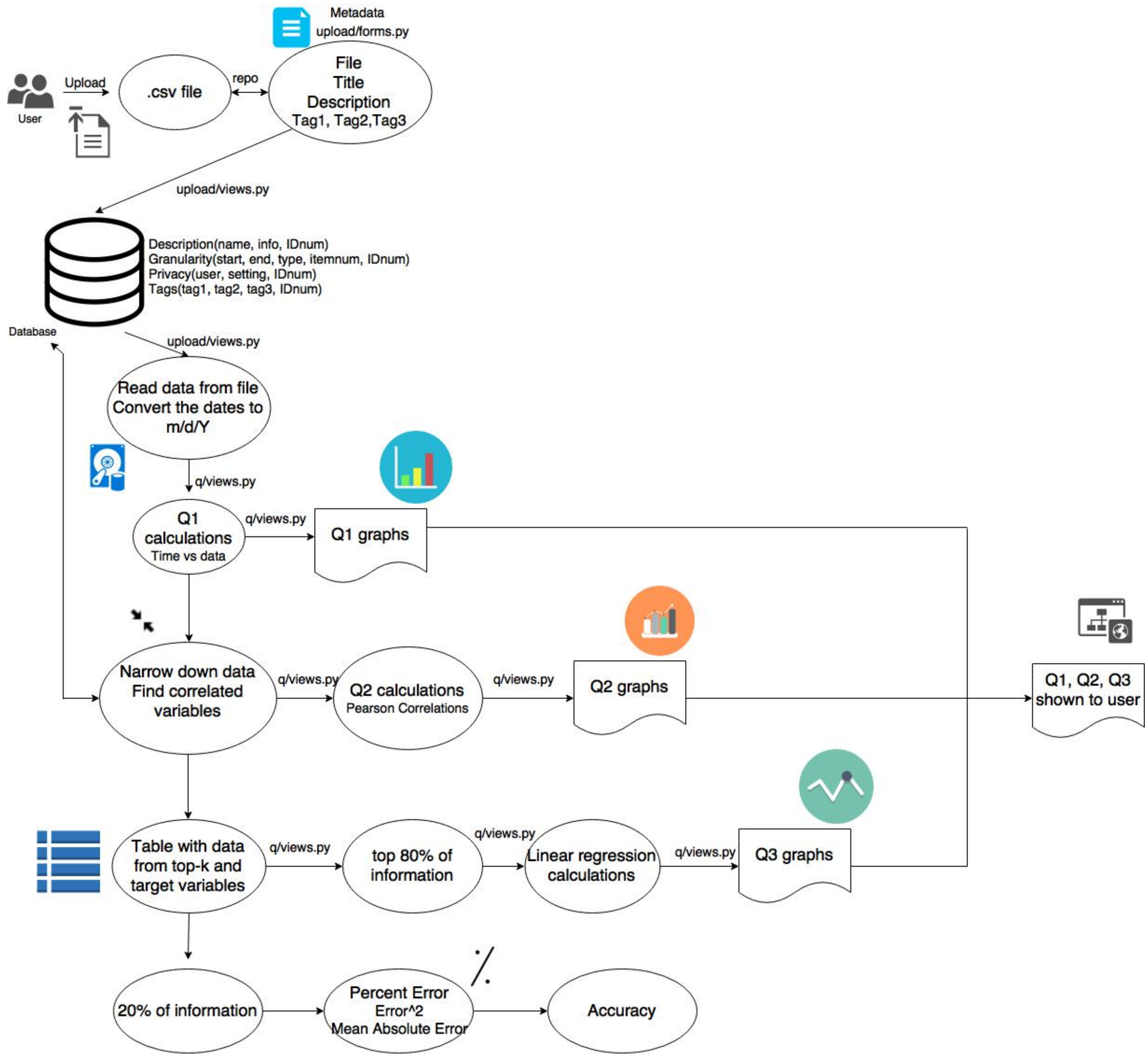
There have been problems with writing the application on Django since none of the team members have used it before and there is not enough documentation in order to achieve what the project is trying to do. The team has had to rely on different websites such as StackOverflow and other tutorials in order to create the webpage. The overall design of the website is on pause since the application is unable to detect the CSS files written for it and will have to be implemented and written in an HTML file that will make the code reading crowded.

Since everything is being saved locally, the team also has the challenge of migrating every upload into a server to show a working application. Optimization and accuracy of the results have to be checked as well as how much time it will take to show the results and narrow to k-number of variables.

The team has problems with implementing the NarrowData aspect of the code. Along with this issue, displaying the dates correctly in the visualizations has been a problem. An implementation to fix the date display is planned for the next Milestone. The team must also research tools needed to implement linear regression that is needed for Q3. Several additional classes are planned to help the program run smoothly, including a converter class to convert uploaded data.

Another problem with the implementation of the application is to gather information from the default database in Django in order to narrow the data for Q2 and Q3; the website needs to show some type of ‘form’ or ‘box’ for the user to type in the metadata and then save it on the database. After the metadata is in the database, the application needs to start narrowing the data that might affect the target variable defined, for which the algorithm will have to check for timestamps, description of data, etc.

## Design:



## Progress Summary:

| Module/feature | Completion % | To Do |
|---|---|---|
| **Main Page** | 90% | Webpage design, redirect navigation bar |
| **User Upload** | 85% | Webpage design, metadata forms for user, redirect to Q1, |

| | | |
|---|---|---|
| | | Q2, Q3 |
| **Q1** | 90% | Website application |
| **Q2** | 60% | NarrowData.py, ParseData.py, Website implementation, Database |
| **Q3** | 0% | Plan, Implementation, Website, and Testing |
| **NarrowData** | 70% | Refine method of NarrowData, research on handling different granularity, finish parsing and testing |
| **Database** | 50% | Rework Database, connect Database to Django web application |

## Milestones:

Milestone 4 (Feb. 12): itemized tasks:

- Website implementation
- Q3 research, pseudocode, implementation
- NarrowData research, pseudocode, implementation, testing
- Website pages for Q1 and Q2
- Website inputs for Metadata and file saving for uploads
- Database handling for NarrowData
- Test cases for website and code
- Add z-scores for Q2
- Evaluation Document, Presentation

Milestone 5 (Mar. 19): itemized tasks:

- Q2 with k-number of variables picked by user
- Improve Visualization
- User Registration, Public/Private tags
- Q3 Add additional forecast points
- Q3 app on website
- Search for past results
- Download uploaded files of official data
- Testing
- Evaluation Document, Presentation

Milestone 6 (Apr 16): itemized tasks:

- Final Design of website
- Save results in account (provider/user)
- Optimization
- Test/Demo Entire System
- Create User Manual
- Create Demo Video
- Evaluation Document, Presentation

**Task Matrix for Milestone 4:**

| **Task** | **Jessica** | **Siomara** |
|---|---|---|
| **Q3** | 75% | 25% |
| **NarrowData** | 85% | 15% |
| **Django Website** | 10% | 90% |
| **Save File Uploads** | 25% | 75% |
| **Database Input Handling** | 50% | 50% |
| **Improve Current Code** | 30% | 70% |
| **Evaluation Document, Presentation** | 50% | 50% |

**Discussion of Each Planned Task:**

Q3: Brainstorm and write pseudocode for Q3; research methods and tools to implement linear regression. Q3.py will use a linear regression library to determine the value in the next timestamp. The timestamp will only be one point to save on complications and errors created by using linear regression on past timestamps. Write, implement, and test code for Q3. Implement Q3 on the current Django website.

After performing Q2, the top-k variables will be put on record/array that will include the dates and data along with the dates and data of the target variable; with the top 80% of the data in the table, the program will perform linear regression mentioned above in order to predict the next value. With the remaining 20% of the data in the table, the application will perform accuracy checking, calculating the error value to see if the system has overestimated or underestimated the values. The next step will be to vary k in order to have more data to evaluate the table.

NarrowData: Research and brainstorm a method to keep all data that has the same granularity and timestamps in one current place. Data that has a start time that occurs at least 5 timestamps before the uploaded code is kept in an array; shuffle and roll functions have already been written. Checking for tags and target variables needs to be figured out and have pseudocode written out and implemented. Make sure that the code works with Q1.py, Q2.py, Q3.py, and uses calls from Input.py, Upload.py, and parseCSV.py. Create another class called PrepareData.py that sets all data that is analyzed in NarrowData.py readable for Q2.py. Test and upload onto GitHub repository.

NarrowData will first check on getting files with timestamps that happen some time before as the target's timestamps and then will search for matching tags; after performing the first two, the team plans to implement the search of similar tags, as well as private/public tags in the future. s

Django Website: Once Q3 is done, implement a new application on the Django website and show the visualizations after the provider has uploaded the file and after Q1 and Q2 have been shown as well. Create a login/logout option for the provider after they have already registered for an account and begin creating a customer view (customer is not allowed to upload files); customer's view include searching for file data with the already implemented search section on the top navigation navigation bar.

Save File Uploads: Figure out and implement how to save files that are uploaded by a user to the Django application. Search for different .csv files with different granularity and save them locally. Research about saving the files on a different interface (such as Amazon S3). Test and implement the change.

Database Input Handling: Link external database to django and/or implement it on the web application and test if database is saving the meta-data input on website for answering the project's Q2. Add column to description table that reads whether or not the data is in column format or in row format for ease of parsing. Re-create database to allow for changes. Research methods to search through database for more than one item at a time.

Improve Current Code: Change visualizations as needed if Faculty Sponsor requests it. Check existing code to ensure that there is no redundancy. Move NarrowData sections in Q2.py to NarrowData.py. Create a class called Convert.py that is called during Upload.py for both the Data Customer and Data Provider. Convert.py will convert all timestamps into a readable format for matplotlib to use, allowing for an easier time at timestamp parsing as well as displaying during visualizations. Improve timestamp parsing to different options (monthly, hourly, etc) as well as datetimes shown on visualizations. Edit Q2.py so that visualizations use the z-score rather than the actual score to improve display.

Evaluation Document, Presentation: Creation of an evaluation document to indicate what was completed in Milestone 4, future requirements in Milestone 5. Create detailed explanations of completed milestones and detailed plans of future milestones. Create a Presentation for Milestone 4. Upload complete and/or current documents/code to GitHub repository and update links on main page. Write the Milestone 4 Evaluation and create the Presentation before the Milestone 4 due date.

**Approval from Faculty Sponsor**

"I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones."

Signature: ________________________________ Date: ________