sdMay19-06- Project Plan

Doh Yun Kim
Drake Mossman
Gabriel Klein
Josiah Anderson
Nathan Schaffer
Jacob Richards

1 Introductory Material

1.1 Acknowledgement

Team 06’s Faculty advisor: Srikanta Tirthapura
Team 06’s client: Principal Financial, primarily Benjamin Harlander and Vishnu Vemuru
Interviewed Data Scientists: Josh Zimmerman, …
We would like to thank Srikanta Tirthapura for being our advisor and helping us out.
P

1.2 Problem Statement

Principal Financial recently created a team of data scientists to work within their Global Investments Department to analyze equity data and help the company make better financial decisions. This team does not have a unified or consistent way of aggregating data and running models on data. This causes losses in time and efficiency.

Our task is to research effective tools and procedures for the data science team, and create a unified application that can be used by all members of the team to run prediction models on stock equity data.

1.3 Operating Environment

The intended operating environment for this application will be the personal work computers used by the data scientists at Principal. The software environment for this application is something that is still flexible and needs to be established.

1.4 Intended Users and Uses

The intended users of this project are the 11 data scientists already employed at Principal Financial and any future data scientists that will work alongside them. This means that this tool must be easy to grasp and must be intuitive for those experienced in data science. The uses of this product should be similar for all users: to take stock equity data and use it to model the most likely trajectory of those stocks.

1.5 Assumptions and Limitations

Assumptions:

1. All data will come from either Factset or Bloomberg
2. Users will have moderate experience with either Python or R

Limitations:

1. Most data scientists only know one of Python or R

1.6 Expected End Product and Deliverables

The final tool will be an application that meets a few specific criteria:

1. It will be modular in design
2. It will be compatible with Python and R
3. It will efficiently run the common prediction models used by Principal data scientists
4. It will have an intuitive User Interface
5. We expect to provide a functional product that is ready for deployment by May of 2019.

1.7 Related Work / Market Survey / Literature Review

We will also be using some Python and R packages that have already been developed to implement our product. Our team has not found any similar tools being used at companies like Principal.

2. Specifications and Analysis

2.1 Proposed Approach

2.1.1 Functional Requirements

The project has the following five main functional requirements:

1. It should be able to run a selectable number of iterations of rolling window calculations automatically with a user-defined prediction model

2. It will be able to import/export results
3. It will be possible to augment the model with new data if needed (no need to rerun the model with new data)
4. It should be able to fit various machine learning models, including linear regression
5. It will be able to use data from multiple sources

2.1.2 Non-Functional Requirements

1. (Performance) Should not lengthen runtime of a single iteration of rolling window calculations
2. (Performance) Should be able to handle up to several GB of data
3. (Performance) Should be able to handle millions of observations and hundreds of columns
4. (Usability) Faster to use than their existing method of manually finding data and inputting it into models

2.1.3 Constraints Considerations

The program languages Principal primary uses is R and Python. Our program will be need to be either R and Python because that is what Principal's data scientist team is most familiar with.

2.1.4 Technology Considerations

There are no technology considerations of yet.

2.1.5 Security Considerations

There are no significant security considerations for our project. Our chosen approach doesn't require any communication with the internet or other computers, remaining entirely local. Additionally, our approach doesn't persist any sensitive information on the computer after running. Thus any and all information involved is discarded after runtime. For these reasons we do not have any further considerations for the security of the project.

2.1.6 Safety Considerations

There are also no significant safety considerations for our project. Our approach doesn't contain any sort of physical component, meaning there is no danger of physical harm to a user. Additionally our project is meant to handle financial information and not any sort of safety-critical data, so there is no danger of the results bringing about harm either. The worst that could happen is a loss of money due to a poorly created model which we will of course be taking precautions against using our testing plan in section 3. In any case, we have no further considerations for injury in the use of our project.

2.1.7 Standards

There are very few standards the code we are creating must comply with. The code must be able to communicate with SQL servers in order to gather and aggregate data. The code must also be able to be accessed by programmers for their implementation.

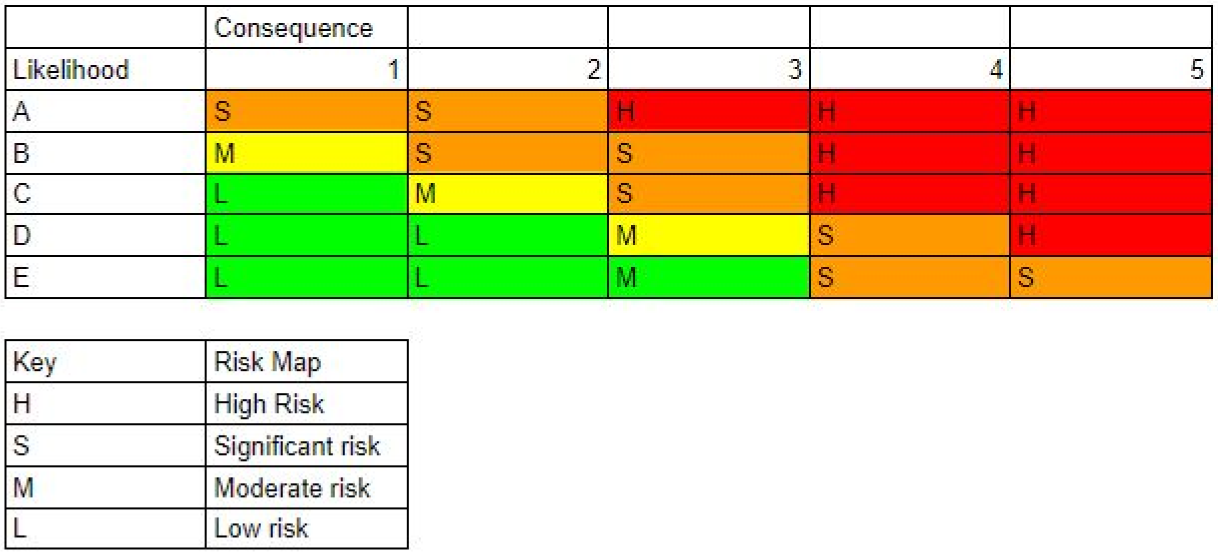
2.1.8 Possible Risks and Risk Management

Risk Map	Consequence	Likelihood	Risk Description	Project Impact	Risk Area	Symptons	Risk Response	Response Strategy
S	3	B	Contact with Principal will come late	Crucial information the team currently needs might delay the current process and in the worst case no work can proceed	Schedule	Principal conatcts us beyond the promised deadline or is not easily reachable	Mitigation	Try to have multiple tasks running at the same time so when I part of the project gets delayed, another part of the project may be worked on
H	3	A	A team member gets busy with other course work	A team member who is busy with other work might not be able to finish their taks on time or hand in sub-standard work	Organizational, Schedule	A team member is unavailable at work at designated timres or are missing deadlines	Mitigation	Redeistrube the work the team member's work to the other people
S	4	E	Principal does not like our current progress we have made in our progress	Time and effort must be spent to redesign a new project	Organizational, Schedule, Budget	Principal expresses negetive feedback during weekly meetings	Mitigatoin	Try and implement changes if Principal expresses negetive feedback during meetings. Get in contact more to try and make chagnes as soon as possible
S	2	B	A team member gets stuck on a part of their task and can't get work progressed	The person who is stuck cannot get work progressed, pontentially risking dealys	Schedule, Organizational	A team member is not getting any sufficent work done a task even after working on it for a (set amount of time)	Avoidence	If a team member believe they are stuck or is about to get stuck, notify the other team members so work can be done together, or work is assigned to someone else
S	4	D	The first prototype is not recieved well	Considerable time from the risk buffer must be taken out to make changes to the prototype	Organizational, Schedule	During the presentation of the first prototype Principal does not like it	Mitigatoin	Make rapid chagnes to the prototype with all team members thriving to finish the redesign as quickly as possible
H	4	B	A major bug is found during the testing period	Unforseenable changes might have to be made to the project in the last minute	Organizational, Schedule, Budget	Bugs found during testing are not easily resolved	Avoidence	During program, adhere to good practices and perform small tests on the way to not encounter major bugs

F

Level	Descriptor	Description
A	Almost certain	Almost certain Expected to occur in most circumstances
B	Likely	Will probably occur in most circumstances
C	Moderate	Should occur at some time
D	Unlikely	Could occur at some time
E	Rare	may occur only in exceptional circumstances

Level	Descriptor
1	Insignificant
2	Minor
3	Moderate
4	Major
5	Catastrophic



	Consequence				
Likelihood	1	2	3	4	5
A	S	S	H	H	H
B	M	S	S	H	H
C	L	M	S	H	H
D	L	L	M	S	H
E	L	L	M	S	S

Key	Risk Map
H	High Risk
S	Significant risk
M	Moderate risk
L	Low risk

Figure 1: Our Risk Management Chart and Keys

2.1.9 Proposed Milestones and Evaluation Criteria

- Investigate
    - Interview different data scientists that work at Principal to gather information about their coding practices.
    - Research example code and find commonalities between them.
- Prototype
    - Taking information we receive from the interview process and create a prototype.
- Test
    - Run prototyped code with real simulation code.
    - Test for bugs and errors in code.
    - Fix and adjust any problems with code.
- Final Project
    - Present final product to Principal executives.

2.1.10 Project Tracking Procedures

The project management systems we chose to utilize are Gitlab issues and GroupMe. Using Gitlab issues allows us to track progress of specific tasks by allowing us to assign tasks and establish deadlines to each individual. Gitlab Issues allow us to also review tasks at our bi-weekly meetings and help keep all team members on the same page. We also chose to use GroupMe for instant messaging which allows for quicker communication between group members.

2.2 Statement of Work

2.2.1 Task Objective

Our objective for this project is to take data manipulation and prediction processes currently used by our client and automate them. This includes the splitting of training and testing data from a chronological set, normalization, non-linear feature engineering, cross validation, model creation, and of course prediction. The training data must be kept safe from contamination due to the temporal nature of the desired predictions. Our project will allow data scientists to quickly and easily create and test models using a variety of parameters and without needing to write boilerplate code or worry about errors in the data manipulation processes.

2.2.2 Task Approach

After discussing several alternatives, the approach we've decided on is to write a Python library to meet our objective. The library will consist of high-level functions and objects with a variety of parameters able to perform the necessary data manipulations. This enables the user to create and test models using very few lines of code.

Strengths:

1. Python is a common general purpose language
2. Python is already used by some of the client's employees
3. Flexible functionality using object members and function parameters
4. Easily extendable by adding new calls
5. Easy to install and distribute

Weaknesses:

1. High level of parameterization needed could require verbose function calls or object initialization
2. Requires users to learn a new set of functions to interface with

Overall we feel that the strengths of this approach outweigh the weaknesses, and that other approaches aren't significant improvements (See Section 2.2.3). While users will have to learn the set of functions we provide, this is true for really any solution we come up with for automating processes. Similarly, there may be significant explicit configuration required resulting in verbose code, but this is again true for any approach that wants to provide that level of customization.

On the other hand, Python is a powerful general purpose language with significant popularity in data science. It’s faster than R, meaning the overall runtime of scripts using the library will be reduced. Additionally the library will be easily integratable into the client’s current workflow, as it simply replaces large blocks of code with a single line that calls the library to do the same manipulations.

The library will provide an object type that can be loaded with data through a method call. Once initialized as such, other functions can then be called with the data object to perform manipulations and transformations. Due to the enormous amount of customization required for the various data manipulations, some generalized functions would require tens or hundreds of arguments to specify. To cut down on this, we’ll offer various more specific calls for different classes of manipulations and keep the number of arguments on a small scale.

The internals of the library will make heavy use of the other libraries our client is already making use of including numpy, pandas, sklearn, and matplotlib. Each of these already implements a lot of very useful functionality, so our library will mainly be focused on combining them to perform larger tasks.

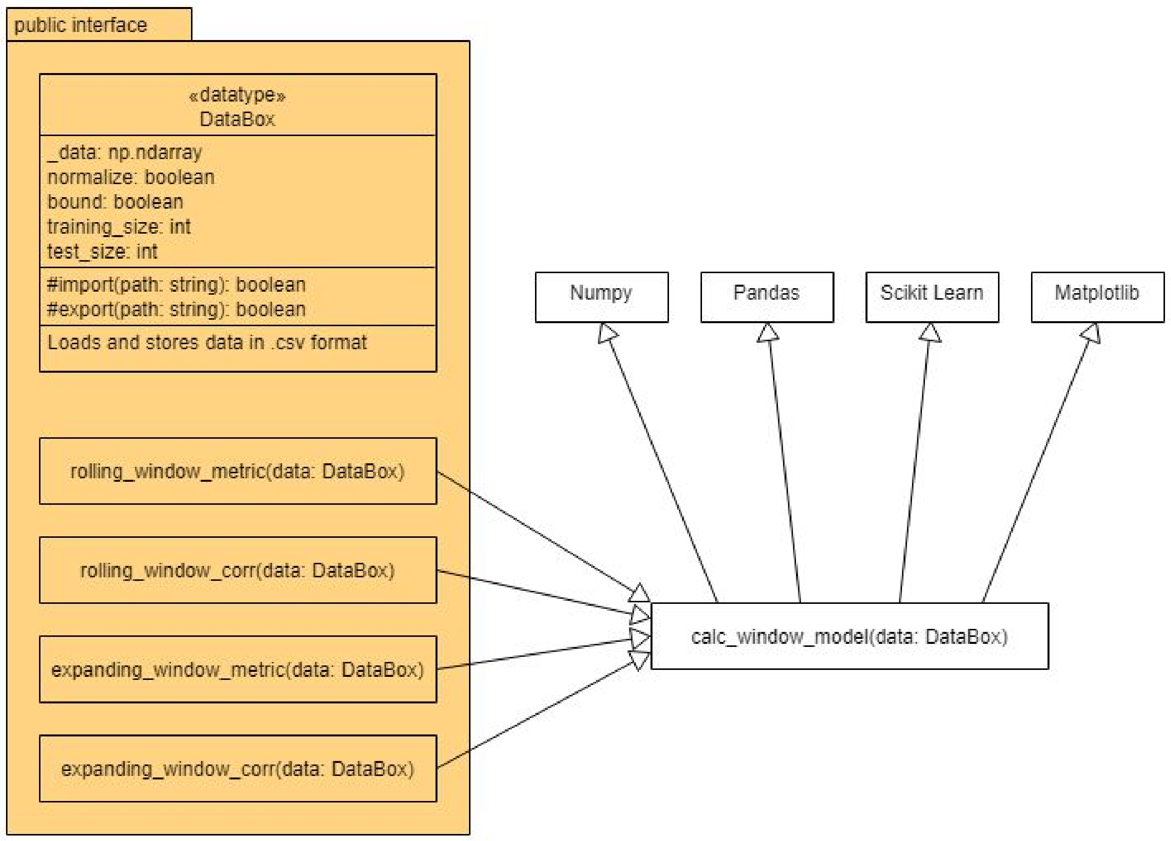

public interface
«datatype»
DataBox
_data: np.ndarray
normalize: boolean
bound: boolean
training_size: int
test_size: int
#import(path: string): boolean
#export(path: string): boolean
Loads and stores data in .csv format
rolling_window_metric(data: DataBox)
rolling_window_corr(data: DataBox)
expanding_window_metric(data: DataBox)
expanding_window_corr(data: DataBox)
calc_window_model(data: DataBox)
Numpy
Pandas
Scikit Learn
Matplotlib

Figure 2: Example of a small subset of function calls and their relationship to the internals

2.2.3 Alternative Approaches

The following consist of several approaches we considered but eventually rejected in favor of our current plan.

R Library

We considered meeting our objective by writing a library of data manipulation functionality implemented in R. The library would consist of high-level functions and objects with a variety of parameters able to perform the necessary processes using single lines of code.

Strengths:

1. R is a very common language used by data scientists
2. R is already used by some of the client's employees
3. Flexible functionality using object members and function parameters
4. Easily extendable by adding new calls
5. Easy to install and distribute

Weaknesses:

1. High level of parameterization needed could require verbose function calls or object initialization
2. R is relatively slow at performing the required processes
3. Requires users to learn a new set of functions to interface with

We decided not to go with this solution mostly due to R's lackluster performance in data manipulations. Many of the processes required take a significant amount of time to complete depending on the amount of data to process and the type of model being tested. The unfortunate lack of speed when using R would result in much longer runtime, which is counter to the project's goal of saving time through automation. Additionally we found that using Python would be a similar yet faster solution with many of the same strengths.

Standalone Application

Another way we considered meeting our objective was by creating a standalone application that could run various data manipulation processes through a UI. The user would feed data to the application and choose the processes to be run on it. The UI would also allow for a significant amount of custom configuration for users to adjust as needed.

Strengths:

1. Doesn't require any coding to process data
2. Able to keep track of user preferences and state between sessions
3. Intuitive to use

4. Can easily save and reload models and results
5. Doesn't require installation of dependencies
6. Doesn't require user to use a particular language for other data manipulation

Weaknesses:

1. Difficult to feed results back into code
2. Unfamiliar concept for client's employees
3. Additional functionality requires building more UI
4. Less platform independent
5. Doesn't update along with packages automatically

We didn't choose this approach mainly because our client wants the end product to be available to many users easily and without a high learning curve. This application would require users to learn an entirely new interface that doesn't even mesh well with their existing workflows. Normally data is passed almost exclusively through code, so adding an application that has to import and export the data into the mix is somewhat awkward.

Browser Application

The final approach we considered was a browser application that could run all of the data manipulation processes remotely. The user could upload data and choose processes to be run on it. The application would provide a UI that could configure the processes as necessary. These processes would then be run on a server and the results sent back to the user.

Strengths:

1. All processes can be run on a server with above average processing power and memory
2. Results can be saved remotely and shared with other users
3. Doesn't require the user's computer to be available while running
4. Doesn't require coding to process data
5. Able to keep track of user preferences and state between sessions
6. Intuitive to use
7. Doesn't require installation of dependencies
8. Doesn't require user to use a particular language for other data manipulation

Weaknesses:

1. User must be online initially and to get the results
2. Big data must be uploaded and downloaded often
3. Difficult to feed results back into code
4. Unfamiliar concept for client's employees
5. Additional functionality requires building more UI
6. Requires a server to be accessed from

Similar to the previous approach, the browser application was not chosen because of the learning curve and adjustment to workflow required. With the added complication of uploading and downloading the data, this approach could serious disrupt the flow of data without significant benefits.

2.2.4 Expected Results

Once the project is complete, we expect to have a functioning Python library that allows users to make one line function calls to perform significant data manipulations such as engineering nonlinear features, cross validation, or backtesting. This will be possible through a variety of function calls made available to the user. By stringing a few of these calls together, we expect the user to be able to create complex models able to predict responses based on the features given.

The library will be portable and easy to distribute. It will simply need to be added using a package manager and then imported into the code to be used. Users can expect to be able to use all the functionality immediately and with little hassle. The code will also be written consistently and with maintainability in mind, allowing for new functionality to be added as necessary in the future.

3 Testing and Implementation

There are primarily two parts to our plan to test our product: requirement testing and user testing. The first will test each of the functional and non-functional requirements we have defined for our deliverables, ensuring correctness of the algorithms, and adequate functionality within its use context. The second will test our design and documentation for our users' perceptions, ensuring that its use is intuitive and self-explanatory. Outcomes from either of these test types could lead to reworking our product and retesting.

3.1 Requirements Testing

The primary goal of requirements testing is to ensure that our automation of tasks remains correct (does not change the expected outputs for each simulation) and performs well within its context.

3.1.1 Functional Testing

Examples include unit, integration, system, acceptance testing

1. Compare 10+ runs of rolling window calculations between our solution and the previous method, varying selectable parameters, including, but not limited to:
    a. Window train size
    b. Window test size
    c. Buffer size
    d. Predictive model / algorithm
    e. Each run should produce the same result from legacy code to new solution

2. The user can observe the output of the tests run in at least one manner.

3. Augment varying amounts of additional data to a partially-constructed model and reverify the model's functionality. The model must still functions with 5+ trials.

4. Run simulations with linear regression and at least two other models, verifying the correctness of their results

5. Run 5+ simulations with data from each of FactSet and Bloomberg, verifying their results

3.1.2 Non-Functional Testing

Testing for performance, security, usability, compatibility

1. Benchmark the runtimes of rolling window iterations in both old scripts and new scripts over tests for functional requirements and ensure that the runtime for the new version isn't longer (within a threshold)
2. Run a simulation with a data set of 2-5 Gb and verify it finishes.
3. Run a simulation over data consisting of > 100 features and > 1,000,000 observations, verifying that it completes.
4. Test with 6+ data scientists, getting feedback to ensure the product does save them time.

3.2 Usability Testing

Usability testing is paramount to our project because if the product we produce isn't intuitive and convenient to use, it will defeat one of its primary purposes of saving our users time.

3.2.1 Process

To test our library's usability, we will be putting it in the hands of data scientists, especially the members of Principal's team whom we interviewed at the onset of the project. We will ask them both to use it in their typical tasks and, if necessary, to try it in contexts and tasks that would more thoroughly test its functionality. After a fixed length of time testing our product, we will meet with them to elicit their feedback.

3.6 Results

No testing has been done in our project so far.

4 Estimated Resources and Timeline

4.1 Estimated Resources

4.1.1 Personnel Effort Requirements

Each team member is expected to put in around 7 hours a week for this project. Each hour of work needs to be filled with hard work and focus in order for this project to succeed.

4.1.2 Other Resource Requirements

Other resources needed for this project are: previous code from the team before us, examples of the different tests they run, and examples of the data format.

4.1.3 Financial Requirements

Since this project is a software only project, there is no cost needed for hardware or any special equipment. The data needed for the project is provided by Principal at no cost.

4.2 Project Timeline

Task Number	TASK TITLE	START DATE	DUE DATE	DURATION	10-01 to 10-05	10-08 to 10-12	10-15 to 10-19	10-22 to 10-26
					M T W R F	M T W R F	M T W R F	M T W R F
1	Interview	10/1/18	10/5/18	4	🟦🟦🟦🟦🟦			
2	Plan out Designs	10/8/18	10/19/18	11		🟦🟦🟦🟦🟦	🟦🟦🟦🟦🟦	
3	Research Most Used Functions	10/8/18	10/12/18	4		🟧🟧🟧🟧🟧		
4	Create Generialized Functions	10/15/18	10/19/18	4			🟧🟧🟧🟧🟧	
5	Consult Design Plan	10/22/2018	10/26/18	4				🟦🟦🟦🟦🟦
6	Refine Design Plan	10/29/18	11/2/18	3				
7	Plan out Prototype #1	11/5/18	11/9/18	4				
8	Create Library Functions	11/12/18	11/30/18	18				
9	Show/Refine Prototype #1	12/3/18	12/7/18	4				
10	Continue Refining Prototype #1	1/14/19	1/18/19	4				
11	Plan out Prototype #2 (based off #1)	1/21/19	1/25/19	4				
12	Create Prototype #2	1/28/19	2/8/19	10				
13	Refine Prototype #2	2/11/19	2/15/19	4				
14	Create Final Design	2/18/19	3/8/19	20				
15	Testing	3/11/19	4/5/19	24				
16	Functional Testing	3/11/19	3/15/19	4				
17	Non-functional Testing	3/25/19	3/29/19	4				
18	Usability Testing	4/1/19	4/5/19	4				
19	Documentatoin	4/8/19	4/12/19	4				
	Risk Buffer			0				
				0				

Task Number	10-29 to 11-02	11-05 to 11-09	11-12 to 11-16	11-19 to 11-23	11-26 to 11-30	12-03 to 12-07	12-10 to 12-14	01-14 to 01-18
	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F	M T W R F
				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
1				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
2				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
3				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
4				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
5				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
6	🟦🟦🟦🟦🟦			🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
7		🟦🟦🟦🟦🟦		🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
8			🟦🟦🟦🟦🟦	🟥🟥🟥🟥🟥	🟦🟦🟦🟦🟦		🟥🟥🟥🟥🟥	
9				🟥🟥🟥🟥🟥		🟦🟦🟦🟦🟦	🟥🟥🟥🟥🟥	
10				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	🟦🟦🟦🟦🟦
				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
11				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
12				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
13				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
14				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
15				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
16				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
17				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
18				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
19				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	
				🟥🟥🟥🟥🟥			🟥🟥🟥🟥🟥	

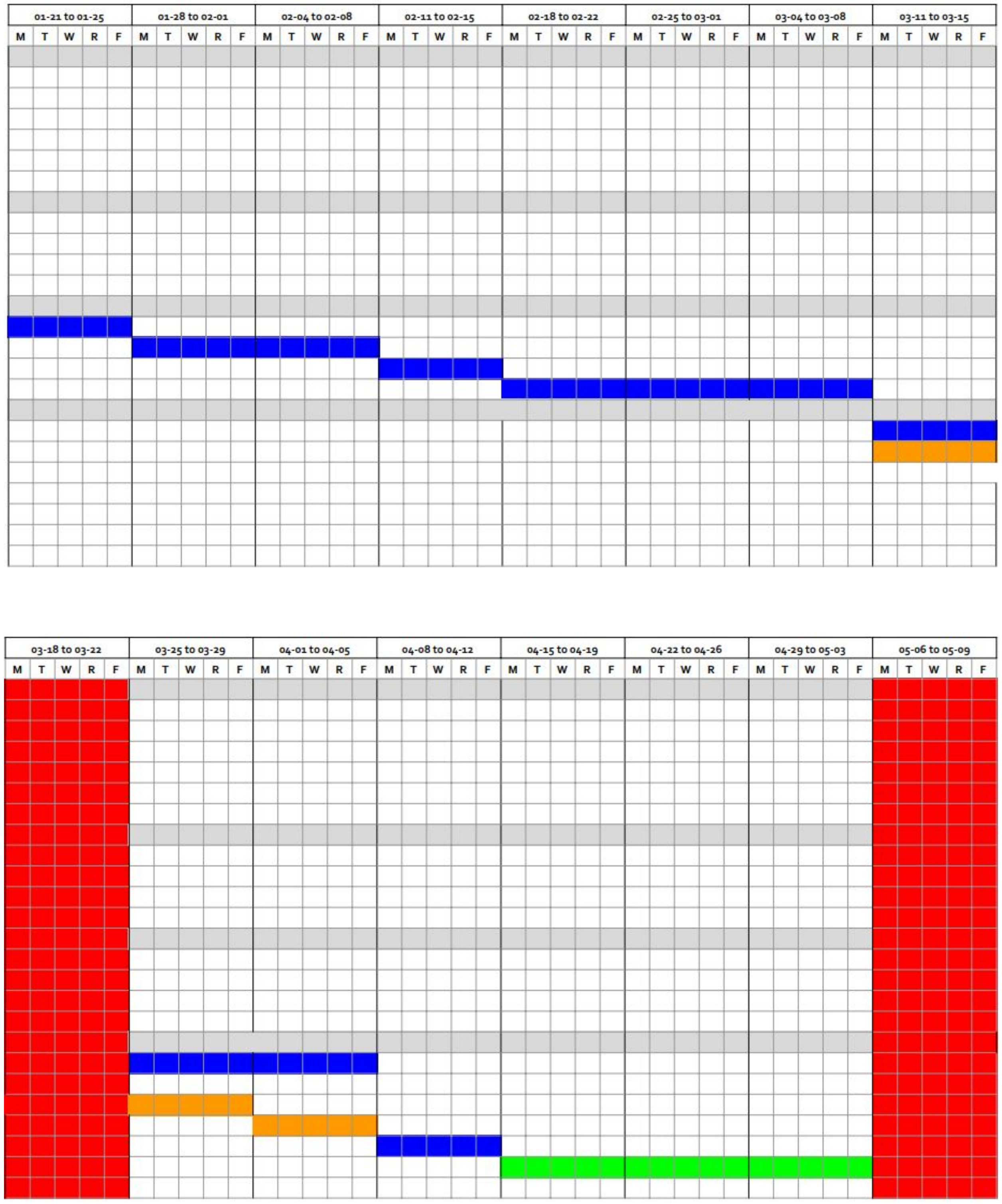


Figure 3: The gantt chart for our project

5 Closing Material

5.1 Conclusion

So to summarize, our team will create a Python library for Principal to use. This library will allow Principal to save time on their redundant tasks, and allows them to further streamline their work. This will allow for consistency and less mistakes in the process of creating the models.

5.2 References

No references of now.

5.3 Appendices

No appendices of now.