CIS 422 Project #1

Prof. Juan Flores

Team Tue2 Project Plan

Problem

There exists a need among data scientists for a system which has the following functionality:

- Create and execute transformation pipelines
- Process time series data
- Feed data to machine learning models
- Test machine learning performance

Solution

Create a python library for creating transformation trees which process time series data, train and optimize machine learning models, and execute transformation pipelines.

The Team

- Riley Matthews
- Lannin Nakai
- Evan Podrabsky
- Justin Spidell
- Theodore Yun

Organization

This systems development team is made up of computer science students with different interests and disciplines. In the interest of promoting equality of opportunity for each member to lead a specific module, the team will be structured as a "flat" democracy. Communication will be held online in an open forum through Discord (a text and voice chat application). All team members' opinions will hold equal weight and decisions will be made through majority consensus.

Risks and Risk Mitigation

This project will function as a learning environment for students of the CIS 422 course: Software Methodologies 1. As such, team members are expected to be generally inexperienced with working on large scale programs and systems in a group environment. The main risk this presents is a lack of knowledge on certain aspects of a system's development cycle. Gaps in this knowledge are expected to be filled as team members attend class lectures and engage with the course textbook. Another risk, this time presented by the project plan, is the possibility of the "big bang" effect towards the end

development where a large portion of modules and functionalities are completed at the same time. Mitigation for this risk is planned to be weekly functionality tests and comparisons with the expected final functionality. If the team can maintain a clear idea of the end product, the "bang" at the end when everything comes together will be minimized.

Final Work Allocation

- Riley Matthews
    - Preprocessing (preprocessing.py)
        - Data standardization
        - Smoothing
        - Impute outliers/missing data
    - Statistics and Visualization (stastics_and_visualization.py)
        - All graphical output
- Lannin Nakai
    - Tree Module (tree.py)
        - Node class
        - Serialize/Deserialize Tree
        - Tree-to-pipeline
    - Pipeline Module (pipelines.py)
        - Pipeline class
        - Save
        - Execute pipeline
- Justin Spidell
    - Tree Module (tree.py)
        - Tree class outline
        - Traversal (search, preorder)
        - Testing
    - Documentation
        - README
- Evan Podrabsky
    - File IO (fileIO.py)
        - Read/Write file
    - Forecasting
        - Abstract estimator class (estimator.py)
        - MLP Model class (mlp_model.py)
        - MLP scripting
- Theodore Yun
    - Statistics (statistics_and_visualization.py)
        - Error checking (MAPE, SMAPE, etc.)
    - Documentation
        - Use-cases
        - Installation instructions
    - Packaging/Installation

- Docker

Work/Meeting Schedule

Main meeting times for the team will occur every week on Tuesday at 2pm (thus, the name Tue2). Meeting duration will be dependent on the amount of work to get done, but the general goal will be to work for the whole hour. Prototypes and incremental modules will be presented to the team and reviewed to determine the next milestone to work towards. Along with the weekly meetings, team members have agreed to allot a small amount of time directly after class lectures on Wednesdays and Fridays in case an impromptu meeting to discuss short-term plans becomes necessary. Below is a log of prototypes, notes, and plans discussed during each weekly meeting -- including after-lecture meetings if, and when, they occurred.

Tuesday January 12, 2021 (Week 2) 14:00-15:00 PST
Main Topics/Milestones: ● BitBucket repository created ● SRS rough draft created ○ Basic definition of the problem and solution ○ A few use-cases ● Went over project specifications ● Decide team organization/work dispersal Current Plans: ● Research libraries to be used ○ pandas ■ Time series object ■ DataFrame object ■ read_csv ○ scikit-learn ■ MLPClassifier ■ PreProcessing ○ matplotlib ○ tkinter ● Team members will report to group by Wednesday after class what they are interested in working on Current Work Allocation: ● Riley: Research/Prepare ● Lannin: Research/Prepare ● Evan: Research/Prepare ● Justin: Research/Prepare ● Theodore: Research/Prepare
Prototype to work on: (Practice not established at this period in development)

Wednesday January 13, 2021 (Week 2) 16:00-16:30 PST

Topics:

- Assign work
    - Riley
        - Preprocessing
        - Stats and visualization
    - Lannin
        - Tree/pipeline class
    - Evan
        - File input/output
        - mlp model
    - Justin
        - Tree/pipeline class
        - UI (command line)
    - Theodore
        - Documentation
        - Tree/pipeline class
- Team name: Tue2 (keep it simple)

Current Work Allocation:

- Riley: Research appropriate libraries
- Lannin: Start building tree prototype
- Evan: Research appropriate libraries
- Justin: Start building tree prototype
- Theodore: Start building tree prototype

Monday January 18, 2021 (Week 3) 14:45-16:00 PST

Main Topics/Milestones:

- Prepare for first client meeting on Thursday
- Showcase early working tree prototype
- Created rough Gantt chart
    - Some portions are unclear - refine tomorrow
- Contact Prof. Flores to see if we need to have a rough SRS/SDS ready for client meeting
- Report current progress on library research

Current Plans:

- Get a basic tree model ready to present in first meeting
    - Could reuse tree classes from CIS 313
    - Function pointers held in tree nodes
        - Use small test functions for now with hard-coded returns

- Nodes hold info about incoming arguments and outputs
- A pipeline will be a single traversal: a straight unary tree (a list)

- Revisit Gantt chart during tomorrow's meeting

Tuesday January 19, 2021 (Week 3) 14:00-15:30 PST

Main Topics/Milestones:

- Finish first version of Gantt chart
  - Still subject to change as more is learned about actual coding workload
- Professor response: SRS/SDS not necessary for first client meeting
- Show and test slightly more refined tree class (ready for thursday!)
- First few phases planned out in Gantt chart
- Basic software architecture discussed
  - Some central module
  - Pass data structure between modules (pandas.DataFrame)

Current Plans:

- First phase to be completed by 1/25
  - Have at least one function from each category ready and be able to run a full pipeline (with or without the tree)

Current Work Allocation:

- Riley: preprocessing function
- Lannin: refine tree class
- Evan: file input
- Justin: refine tree class
- Theodore: error checking (stats and visualization)

Prototype to work on: Have prototype function from each category

Wednesday January 20, 2021 (Week 3) 13:45-14:05 PST

Topic:

- Final preparations and checks for Thursday client meeting
- Review Gantt chart, phases, deliverables

Saturday January 23, 2021 (Week 3) 13:00-14:30 PST

Topics:

- File input, preprocessing, stats and visualization are ready
- Tree is coming along well, early version of search, delete, traverse
- First phase deliverable likely won't be met

    - Workload and research depth of mlp model doesn't fit in with other class schedules
- New goal for Monday: create a pipeline up until the mlp model
    - Still test stats and visualization function
- What is design_matrix()?
- What do the parameters of split_data() mean?

Current Plans:

- Be able to run current functions through tree
    - Error checking between function input/output is not ready
    - Revisit this on Tuesday
- Start on mlp model
    - Will not be finished by Tuesday
- Continue knocking out individual preprocessing functions

Tuesday January 26, 2021 (Week 4) 14:00-15:00 PST

Main Topics/Milestones:

- Tree can execute individual nodes
    - Pipelines to be worked on during the end of this week
- How will pipelines work with the current tree structure?
- Group tested individual functions and ran data through file input, preprocessing, and visualization
    - Everything is very basic at the moment, but works and can very easily be executed in sequence

Current Plans:

- Write basic function documentation for current versions of presented functions
    - Docstrings, comments, etc. (docstrings can be copied over to documentation)
- Begin major research/work push on mlp model
    - sklearn MLPClassifier?
    - Should we also do random forest? -- if time allows and implementation is simple
- Create pipelines module which can interact with tree module
    - Parameter checking here or in tree?
- Fill out more functions in preprocessing, stats/visualization

Current Work Allocation:

- Riley: preprocessing and stats/visualization
- Lannin: Pipelines/tree
- Evan: MLP model
- Justin: Tree
- Theodore: Documentation, statistics

Prototype to work on: Deliver a fully working pipeline (w/ mlp model) by next meeting

Friday January 29, 2021 (Week 4) 14:00-15:00 PST

Topics:

- Progress check
- MLPClassifier will not work for this implementation
    - MLPRegressor will be main mlp implementation
    - Model will be held in a new class
- Pipeline class currently in the works
- New refined version of tree class is almost ready
- What is everyone working on over the weekend?

Current Plans:

- Work on pipeline and tree class
- Work on mlp regressor class
- Continue with preprocessing and stats/visualization

Tuesday February 2, 2021 (Week 5) 14:00-15:30 PST

Main Topics/Milestones:

- MLP model class and scripting functions are ready
    - Tests show very poor predictions, but grading is not based on model performance
- Pipelines class has just one more piece of functionality
    - Can initialize pipeline by itself, but not yet ready to transfer from tree to pipeline
    - Can execute string of functions
    - Tested and worked well with other available module
- Full tree functionality is finished outside of interaction with pipelines class
- More preprocessing functions finished
    - A few straggling functions, but enough to present with
- Error statistics finished
- Presentation slides created
    - Rough start for a few slides completed during meeting

Current Plans:

- Prepare for presentation on Friday
    - Prepare demonstration program
    - Finish presentation slides by Thursday evening
    - Practice spoken parts on Friday morning
- Test MLP to see if there's a way to improve estimations
- Finish pipelines, show group on Thursday

- Meeting in the evening on Thursday

Current Work Allocation:

- Riley: Finish preprocessing slides, work on some straggling functions in preprocessing and stats/visualization
- Lannin: Finish pipelines and slides for original tree diagrams/plans
- Evan: Finish original plan and mlp slides, finish file output
- Justin: Finish tree slides
- Theodore: Finish reflection slides, begin compiling information for Docker

No more prototypes. We are now in the home stretch for the final modules.

Thursday February 4, 2021 (Week 5) 19:30-21:00 PST

Topics:

- Worked on slides (will finalize during meeting tomorrow morning)
- Tree class can now export traversal to pipeline class
- MLP will stay as-is
    - Estimator abstract class added
- Lannin met with Prof. Flores and confirmed functionality of design matrix
- Meet tomorrow morning 09:30

Friday February 5, 2021 (Week 5) 09:30-10:30 PST

Topics:

- Finalize slides
- Rehearse presentation and demo

Sunday February 7, 2021 (Week 6) 13:00-14:30 PST

Main Topics/Milestones:

- Version of design_matrix() ready, needs to be tested an reworked
- Large-scale documentation begins today
- We have rough draft of SRS/SDS from earlier in the term, add onto this

Current Tasks:

- Complete final project plan document
- Complete final SRS document
- Complete final SDS document
- Complete final README.md document
- Beautify code and standardize docstrings
- Finalize Docker and complete installation instructions

- Finish a few straggling functions
    - design_matrix()
    - assign_time()
    - ts2db()

Current Work Allocation:

- Riley: Finalize preprocessing.py, work on SRS/SDS
- Lannin: Finalize pipelines.py, finish design_matrix(), word on SRS/SDS
- Evan: Make unit test cases for mlp_model.py, work on project plan
- Justin: Beautify/organize repository, work on README.md
- Theodore: Finish Docker/documentation, work on SRS/SDS

Tuesday February 9, 2021 (Week 6) 14:00-23:00 PST

Topics:

- Long work session where team members can come and go as necessary
- Big push for finalization of files and repository
    - Close communication is necessary as pushes are made to the repository
- SRS, SDS, and Project Plan are nearly finished
    - Meeting at 16:00 tomorrow for finalization

Current Plans:

- Continue work allocation as-is

Wednesday February 10, 2021 (Week 6) 16:00-20:00

Topics:

- Finalize all documentation
- Finalize all docstrings, comments, and code format
- Finalize all modules and test suites
- Finalize Docker and installation instructions
- Turn in final product

Reflection

Looking back, the group collectively agrees that the original plan was followed relatively well. Meetings were held regularly and communication was effective whenever the group was collaborating. The definition of "prototype" warped from week to week, but the general idea of incremental development was still upheld. All team members consistently updated the rest of the group with their current work and any issues they were having. Individuals were very quick to jump on to help solve coding problems.

Having sufficient documentation to begin development with at would have been extremely beneficial. Though the initial idea was to mitigate the risk of a "big bang" of functionality towards the end of development, the effect still occurred to a certain degree. The group was able to test by manually pipelining transformation functions but the full pipelining functionality wasn't ready until late in development. This made extensive testing difficult with everyone's schedule being relatively full around the end of the project.

The main aspects of the project our group would have changed are:

- Have a better understanding of the project specifications and technical requirements in the beginning
- Be able to accurately judge the workload of each functionality implementation
- Schedule more client meetings to get feedback

Gantt Chart

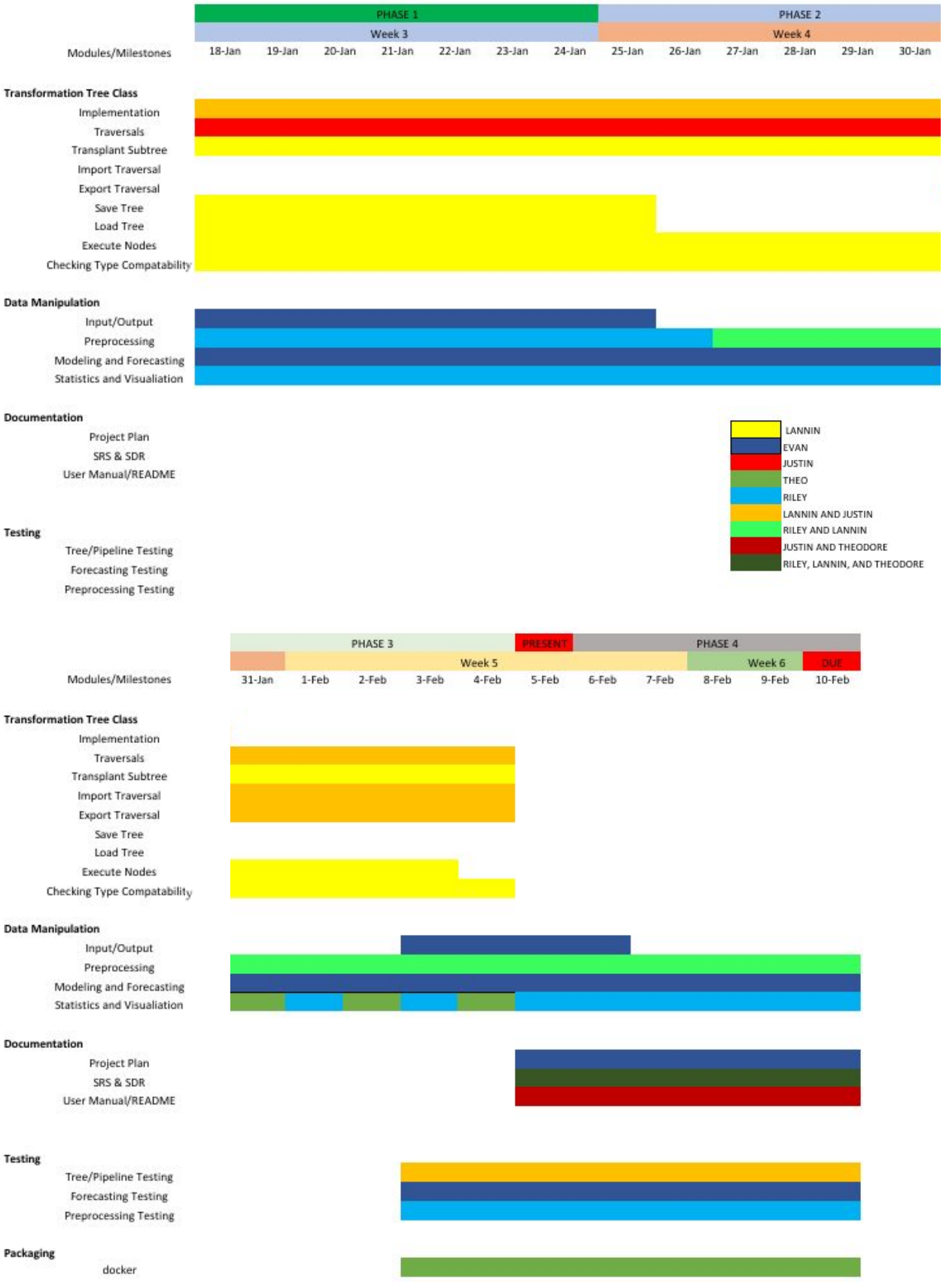
Below is a refined version of the original Gantt chart. Sections of more than one color denote tasks which were completed by more than one person. There is a legend in the top right corner of the chart for who each color represents. The top of the chart shows phases 1-4 of development. The phases are as follows:

Phase 1. Have a working tree class which can hold and execute functions as its nodes.

Phase 2. Complete at least one transformation function from each category and be able to plug them into the tree.

Phase 3. Be able to construct a working pipeline which can execute a line of transformation functions.

Phase 4. Complete packaging, testing suites, and documentation for finished products.


PHASE 1
PHASE 2
Week 3
Week 4
Modules/Milestones
18-Jan
19-Jan
20-Jan
21-Jan
22-Jan
23-Jan
24-Jan
25-Jan
26-Jan
27-Jan
28-Jan
29-Jan
30-Jan
Transformation Tree Class
Implementation
Traversals
Transplant Subtree
Import Traversal
Export Traversal
Save Tree
Load Tree
Execute Nodes
Checking Type Compatability
Data Manipulation
Input/Output
Preprocessing
Modeling and Forecasting
Statistics and Visualiation
Documentation
Project Plan
SRS & SDR
User Manual/README
Testing
Tree/Pipeline Testing
Forecasting Testing
Preprocessing Testing
LANNIN
EVAN
JUSTIN
THEO
RILEY
LANNIN AND JUSTIN
RILEY AND LANNIN
JUSTIN AND THEODORE
RILEY, LANNIN, AND THEODORE
PHASE 3
PRESENT
PHASE 4
Week 5
Week 6
DUE
Modules/Milestones
31-Jan
1-Feb
2-Feb
3-Feb
4-Feb
5-Feb
6-Feb
7-Feb
8-Feb
9-Feb
10-Feb
Transformation Tree Class
Implementation
Traversals
Transplant Subtree
Import Traversal
Export Traversal
Save Tree
Load Tree
Execute Nodes
Checking Type Compatability
Data Manipulation
Input/Output
Preprocessing
Modeling and Forecasting
Statistics and Visualiation
Documentation
Project Plan
SRS & SDR
User Manual/README
Testing
Tree/Pipeline Testing
Forecasting Testing
Preprocessing Testing
Packaging
docker

Work Allocation Tracking

Below is the list of functions used to track progress of individual components and whether or not they have been pushed to the BitBucket repository.

Transformation Tree

- Node class – Lannin: completed and pushed to repo
- Tree class
    - __init__(), __repr__() - Justin: completed and pushed to repo
    - search(target) – Justin: completed and pushed to repo
    - traverse() – Justin: completed and pushed to repo
    - delete(node) – Justin: completed and pushed to repo
    - replace(node, func) – Justin: completed and pushed to repo
    - insert(func, io_type, func_args, parent) – Lannin: completed and pushed to repo
    - match(parent, child) – Lannin: completed and pushed to repo
    - get_args(parent, child) – Lannin: completed and pushed to repo
    - serialize(node) – Lannin: completed and pushed to repo
    - deserialize(node, saved_string) – Lannin: completed and pushed to repo
    - save_tree() – Lannin: completed and pushed to repo
    - restore_tree(state_number) – Lannin: completed and pushed to repo

Pipelines

- Pipeline class
    - save_pipeline() – Lannin: completed and pushed to repo
    - make_pipeline(tree, route) – Lannin: completed and pushed to repo
    - add_to_pipeline(func) – Lannin: completed and pushed to repo
    - functions_connect(func_out, func_in) – Lannin: completed and pushed to repo
    - run_pipeline(data) – Lannin: completed and pushed to repo

Input/Output

- read_from_file(input_file_name) – Evan: completed and pushed to repo
- write_to_file(output_file_name) – Evan: complete and pushed to repo

Preprocessing

- denoise(ts) – Riley: completed and pushed to repo
- imput_missing_data(ts) – Riley: completed and pushed to repo
- imput_outliers(ts) – Riley: completed and pushed to repo
- longest_continuous_run(ts) – Riley: completed and pushed to repo
- clip(ts, starting_date, final_date) – Riley: completed and pushed to repo
- assign_time – Riley: completed and pushed to repo
- difference(ts) – Riley: completed and pushed to repo
- scaling(ts) – Riley: completed and pushed to repo

- standardize(ts) – Riley: completed and pushed to repo
- split_data(ts, perc_training, perc_valid) – Evan: completed and pushed to repo
- design_matrix(ts, input_index, output_index) – Lannin: completed and pushed to repo
- design_matrix(ts, mi, ti, mo, to) – Lannin: completed and pushed to repo
- ts2db(input_filename, perc_training, perc_valid, perc_test, input_index, output_index, output_index, output_file_name) – Evan: completed and pushed to repo

Modeling and Forecasting

- Estimator abstract class – Evan: completed and pushed to repo
- MLPModel class – Evan: completed and pushed to repo
- mlp_model(input_dimension, output_dimension [, layers]) – Evan: completed and pushed to repo
- mlp.fit(x_train, y_train) – Evan: completed and pushed to repo
- mlp.forecast(x) – Evan: completed and pushed to repo
- mlp.score(x_valid, y_valid) – Evan: completed and pushed to repo
- train_new_mlp_model(train_data, input_dimension, output_dimension [, layers]) – Evan: completed and pushed to repo
- write_mlp_predictions_to_file(mlp, test_data, file_name) – Evan: completed and pushed to repo

Statistics and Visualization

- plot(ts|ts_list) – Riley: completed and pushed to repo
- histogram(ts) – Riley: completed and pushed to repo
- box_plot(ts) – Riley: completed and pushed to repo
- normality_test(ts) – Riley: completed and pushed to repo
- mse(y_test, y_forecast) – Theodore: completed and pushed to repo
- mape(y_test, y_forecast) – Theodore: completed and pushed to repo
- smape(y_test, y_forecast) – Theodore: completed and pushed to repo