*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Project Title: Data Science Workflow Manager

**Team Members:** Tomas Ortega, Serge Metellus

**Product Owner(s)**: Dr Miguel Alonso, Steven Luis

**Mentor(s)**: Dr Miguel Alonso, Steven Luis

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document provides the required information to understand the purpose of the Data Science Workflow manager 1.0 web application. The goal of the workflow manager system is to provide the airline industry with the tools needed for analyzing and exploring the large amounts of that that are produced every day in a way that is efficient and easy to use. Our focus on this first version of the project was to construct an interactive dashboard and create a system to process data downloaded from different sources allowing developers to implement machine learning models and visualize the results in the interactive dashboard in an easy and efficient way. This document will show in detail how we designed and implemented the features of the workflow manager, what technologies we used and how we used agile development to accomplish our goal of creating the first version of the workflow manager.*

**Table of Contents**

Introduction 6

Current System 6

User Stories 7

Implemented User Stories 7

Pending User Stories 14

Project Plan 19

Hardware and Software Resources 19

Sprints Plan 20

Sprint 1 20

Sprint 2 21

Sprint 3 21

Sprint 4 22

Sprint 5 23

Sprint 6 24

Sprint 7 24

System Design 26

Architectural Patterns 26

System Architecture Diagram 27

System and Subsystem Decomposition 27

Deployment Diagram 27

Design Patterns 27

Deployment Diagram 28

System Validation 29

Unit Tests 29

Integration Tests 34

Glossary 39

Appendix 40

Appendix A - UML Diagrams 40

Appendix B - User Interface Design 65

Appendix C - Sprint Review Reports 68

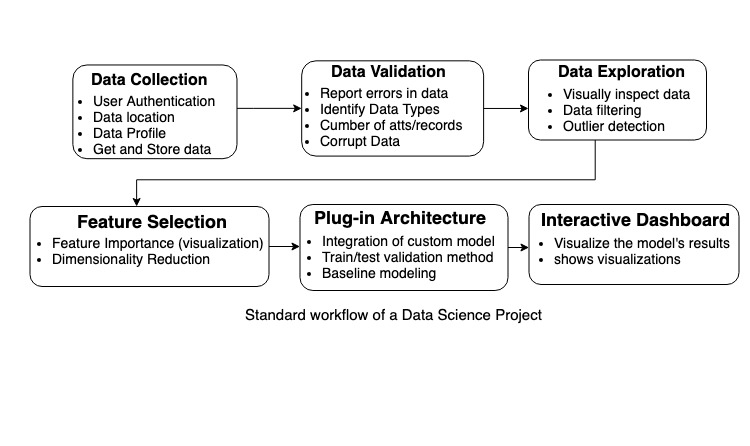
Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents 73

References 79

# Introduction

The main objective of this project was to develop a Data Science Workflow Manager (DSWM), or in other words, a set of tools that automates and provides the appropriate support for any Data Science project. For the first version of the project, we wanted to provide the user with an interactive dashboard, tools for collecting, validating, and exploring data, as well as the ability to easily introduce plugins to the standard data science workflow so that he can utilize his time more effectively evaluating the results instead of getting lost in the details of the next step in the data science workflow. To produce a viable product by the end of the semester, the team decided to focus on model integration and customized web-based visualization to allow end-users to explore model generated data interactively while using DB1B airlines fare pricing analysis model as the initial test case.

## Current System

We worked on version 1.0 of the Data Science Workflow Management System. The current system has the following functionalities. Users of the web application are able to see and interact with graphs that contain predictions from machine learning models and move single or groups of points to change the training data. After the original data have been modified, the end user can click on the train button to train the models and evaluate the new results. The user can also visualize frequency distributions involving attributes such as state of departure, market share and coupon distributions. The application is currently able to handle and process data coming from the United States Department of Transportation (DB1B and T100 databases) to generate high quality visualizations. End users are also able to see and interact with the visualizations containing data collected and processed that shows the CPI and the Crude Oil price which is really important in the airline industry and are key parameters to study if you want an accurate prediction of the fare price or any other metric related to the airline industry. The back-end follows the standard workflow of a data Science project and provides the tools required to complete some of the steps shown in the diagram below.

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of the WFMG project. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

These are all the user stories that were implemented for this project. We decided to include the Epics that we worked on during the semester to clearly show the relationship between the user stories.

**Epic - User Interface**

**User Story:** Display Distribution of Data (WFMG-2)

* **Description**:
  + As an End user I would like to be able to select attributes from the data set and display the distribution of the data so that I can see trends and compare the different graphs.
* **Acceptance Criteria:** 
  + Must work with every set of attributes.
  + The visualization must be of appropriate size.
  + It must work with a large data set.
  + Charts must be of high quality and ready to publish.

**User Story:** Filter Data by time (WFMG-3)

* **Description:**
  + As a public user, I should be able to filter time series graphs individually according to time. (look at jupyter notebook)
* **Acceptance Criteria:** 
  + User Interface should be clean and organized.
  + Must work for the specified attribute.
  + The filtering should be done quickly and preferably in the backend.

**User Story:** Organize charts into columns (WFMG-4)

* **Description:**
  + As a public user, I must be able to see the data sets in 1, 2, 3, or n columns so that I can identify trends and relationships between the charts.
* **Acceptance Criteria:** 
  + The user must be able to adjust the number of graphs per column.
  + The user must be able to select the graph in each column.
  + It must work with a large data set.
  + Charts must be of high quality and ready to publish.
  + The max amount of columns should be restricted to 5 columns.

**User Story:** Display data as line plot (WFMG-12)

* **Description:**
  + As an End user, I should be able to map available data sets and display them in the form of line plots to be able to understand trends.
* **Acceptance Criteria:** 
  + It must be integrated with the rest of the webpage.
  + The visualization must be of appropriate size.
  + It must work with a large data set.
  + Charts must be of high quality and ready to publish.

**User Story:** User Interface (WFMG-13)

* **Description:** 
  + As an End user I would like to be able to interact with the web application through my device (it must work with several sizes and ratios) so that I can download, upload, or interact with the available data sets and visualize the data from different sources as line, bar, or pie charts.

* **Acceptance Criteria:** 
  + It should have containers for the different charts and graphs.
  + It must have a cohesive design.

**User Story:** User Interface (WFMG-13)

* **Description:**
  + As a public user, I should be able to interact with the web application through my device so that I can download, upload, or interact with the available data sets.
* **Acceptance Criteria:** 
  + It should have containers for the different charts and graphs.
  + It must have a cohesive design.

**User Story:** Fare Price visualization (WFMG-19)

* **Description:**
  + As a public user, I should be able to visualize the fare price in the past and in the future over a one year period. delineating past from future (one year in the past and 6 months in the future) using multiple data sources.
* **Acceptance Criteria:** 
  + Must work with every set of attributes.
  + The visualization must be of appropriate size.
  + It must work with a large data set.
  + Charts must be of high quality and ready to publish.

**User Story:** Fix column layout (WFMG-25)

* **Description:**
  + As a public user, I must be able to see the data sets in 1, 2, 3, or n columns so that I can identify trends and relationships between the charts.
* **Acceptance Criteria:** 
  + The user must be able to adjust the number of graphs per column.
  + The user must be able to select the graphs in each column.
  + It must work with a large data set.
  + Charts must be of high quality and ready to publish.
  + The max amount of columns should be restricted to 5 columns.

**User Story:** Movable Points (WFMG-33)

* **Description:**
  + As an End User, I would like to be able to move individual data points and trigger an event to redraw the graph with the modified data so that I can visualize the modified data set and later train the model with the modified data.
* **Acceptance Criteria:** 
  + Dragging and Dropping points on the graph should update the data in the server and redraw the graph in the user interface.
  + The graph should respond quickly to the changes made by the user.
  + The software should recognize the position of the new point when they are dropped or placed in a new position.
  + It must be integrated with the rest of the components of the graph (such as the range slider).

**User Story:** Drag Multiple data points (WFMG-34)

* **Description:**
  + As an End User, I should be able to select more than one point at the same time and modify them (increase or decrease) by the same value to test the predictions of the model on different scenarios that could happen again in the feature. This should trigger an event to re-draw the graph after the user lifts the mouse to visualize the modified graph and store the new training data in the server.
* **Acceptance Criteria:** 
  + The software should recognize the position of the new point when they are dropped.
  + Dragging and Dropping points on the graph should redraw the graph.
  + The graph should respond quickly.
  + It must be integrated with the rest of the dashboard.

**User Story:** Drag Multiple data points (WFMG-35)

* **Description:**
  + As a user, I should be able to reset the graph once I have changed some data points so that I can visualize the original graph.
* **Acceptance Criteria:** 
  + The button should be of appropriate size.
  + The button should be to the right of the graph and should be clearly related to the graph that is controlling.
  + The graph should be updated quickly.
  + The files must preserve their metadata.

**User Story:** Random Forest Visualization (WFMG-42)

* **Description:**
  + As a developer, I would like to visualize the results of the Random Forest regressor model and have all the functionality of an interactive model graph. The graph should have a reset and a train button, a slider to move individual and groups of points and a graph for the error and the feature importance of the features used in the model.
* **Acceptance Criteria:** 
  + The model should be trained when points are changed or when the train button is clicked.
  + The software should recognize the new position the points when they are moved to train the model with the new data set.
  + The range of the data used to train the model should be stated near or inside the graph.
  + The graph should respond quickly.
  + It must be integrated with the rest of the dashboard.

**Epic - Implement an architecture for custom plug-ins**

**User Story:** Adjust Prediction Parameters (WFMG-22)

* **Description:**
  + As a developer, I want to be able to adjust the prediction parameters and obtain the estimated future seat price. This would allow developers to use the model wrapper class to adjust the parameters and hyper-parameters of the models used by the visualization module.
* **Acceptance Criteria:**
  + It must have be able to run the model with new parameters after the model has been initialized.
  + The feature must be available to all models that inherit from the wfmg\_model class

**User Story:** Model Wrapper (WFMG-23)

* **Description:**
  + As a model/plug-in developer, I should be able to derive a model from the base class so that a public user can visualize the results of the model and evaluate the results. The model objects should have the following functionalities: saving and loading models from pickles.
* **Acceptance Criteria:**
  + It must have functions for getting the data for the visualizations.
  + It must have functions for running the model with new parameters.
  + It must provide an interface for creating different kinds of models (univariate and multivariate)

**User Story:** Visualization Wrapper (WFMG-24)

* **Description:**
  + As a model developer, I should be able to derive a procedure class from the base class so that a public user can visualize the results of different data distributions.
* **Acceptance Criteria:**
  + It must be implemented using object-oriented programming.
  + It must have functions for getting the data for the visualizations.

**User Story:** Retrain Model on Point Movement Events (WFMG-40)

* **Description:**
  + As an End User, I would like the model to be trained with the updated data after I move a point or a group of points in the graph. The action should be executed after the user lifts the mouse and the point moves to the specified position. If the time required to train the model is unreasonable, then the user should have the option to train the model using a button or to train it after every point movement.
* **Acceptance Criteria:**
  + The software should recognize the position of the new point when they are dropped.
  + When new points are entered, the train button should send a signal to train the model and update the graph.
  + If the Train button is disabled, the points representing the model data should be updated on every point movement.
  + The graph should respond quickly if the user is using the train button or the train button is disabled.
  + It must be integrated with the rest of the dashboard.

**User Story:** Random Forest Model (WFMG-41)

* **Description:**
  + As a developer, I would like to have a plug-in for the Random Forest regressive model so that I can visualize the data in the dashboard using this wrapper class and compare the results with the historical data for the fare price.
* **Acceptance Criteria:**
  + The model should have a fit and predict method.
  + It must accept a new data set to use as training data so that the visualization module can use the new training data from the UI to retrain the model.
  + The class should implement the WFMG\_Model interface.
  + It must be integrated with the rest of the dashboard.

**User Story:** Multivariate Time Series Analysis Model (WFMG-43)

* **Description:**
  + As a developer, I would like to visualize the results of the vector autoregressive model (multivariate time series analysis model) and have all the functionality of an interactive model graph. The graph should have a reset and a train button, a slider to move individual and groups of points and a graph for the error.
* **Acceptance Criteria:**
  + The model should have a fit and predict method.
  + It must accept a new data set to use as training data so that the visualization module can use the new training data from the UI to retrain the model.
  + The model data should be visualized as a line plot.
  + The software should recognize the new position the points when they are moved.
  + The data used to train the model should be stated near or inside the graph.
  + Dragging and Dropping points on the graph should redraw the graph.
  + The graph should respond quickly.
  + It must be integrated with the rest of the dashboard.

**Epic - Data Processing/Preprocessing**

**User Story:** Data Processing/Preprocessing (WFMG-39)

* **Description:**
  + As a developer, I would like the web application to automatically clean and process raw data and store it in a separate directory so that the models can use it as training data or in some procedures.
  + (Description of data + Processing steps) The team discussed the possibility of using the make utility (a makefile) to specify the cleaning and processing scripts and the order of execution.
* **Acceptance Criteria:**
  + There should be separate scripts for every procedure
  + The developers must have a way of introducing new scripts into the processing steps.

## Pending User Stories

**User Story:** Download/upload data files. Organize and browse data sets (WFMG-1)

* **Description:**
  + As a developer, I would like to have access to the latest data and use it in the visualizations and for training models.
* **Acceptance Criteria:**
  + The system should handle csv files and text files.
  + The system should organize the data in a way that the user can easily go through it and explore it.

**User Story:** Collect necessary data from users or the web (WFMG-5)

* **Description:**
  + As a developer, I would like to be able to choose a different data set when I’m using the web application so that I can use the tools provided to me with other data sets.
* **Acceptance Criteria:**
  + The system should handle csv files and text files.
  + The system should authenticate the user if the data source needs it to download the data.

**User Story:** Validate the data (WFMG-6)

* **Description:**
  + As a developer, I would like to have a way of validating and detecting errors in the data. To validate the data the system must be able to detect missing values, check for corrupt data, Report errors in the data, show the attributes and the number of records along with the data types.
* **Acceptance Criteria:**
  + The application must detect all missing values without exceptions
  + If there are missing values or corrupt data, then the application should issue a warning and alerts the user of what’s happening.

**User Story:** Explore and visually inspect the data (WFMG-7)

* **Description:**
  + As a Developer, I would like to explore the data before using it in one of the models to inspect trends in the data and see the characteristics of this data set. The system must be able to show the data to the user so that he can visually inspect it. See the Jupyter notebook for the different kinds of visualization. The system must also be able to filter the data according to some parameters
* **Acceptance Criteria:**
  + The application must have a dedicated user interface for inspecting data sets.
  + It should work with any data set that follows the standard rules for columns and records. It must deal with different data types.

**User Story:** Detect outliers and data cleaning (WFMG-8)

* **Description:**
  + As a user, I would like to have the system detect outliers and notify me so that I can decide what to do with them. This step includes outlier detection and decides what to do with missing values. The user must decide what to do with these missing values which were detected before.
* **Acceptance Criteria:**
  + The application must trigger a warning when outliers are detected.
  + It should work with any data set that follows the standard rules for columns and records. It must deal with different data types.

**User Story:** Feature Importance (visualization) and dimensionality reduction (WFMG-9)

* **Description:**
  + As an End user, I would like to know the feature importance of models such as Vector autoregression and Random Forest. This step includes dimensionality reduction and feature importance. The feature importance must be shown along the graph that uses the features.
* **Acceptance Criteria:**
  + The feature importance must be displayed as a bar chart.
  + The feature important also should have the names of the features so that the users can see which feature is the most important in the predictions.

**User Story:** Drag and Drop (WFMG-16)

* **Description:**
  + As a Developer, I would like to be able to drag different models into the processing pipeline. The system should be able to recognize the action of the user and react accordingly.
* **Acceptance Criteria:**
  + Drag and drop capabilities are usable.
  + The software should recognize the position of the new objects when they are dropped.
  + It must be integrated with the rest of the web application.

**User Story:** Share Data (WFMG-20)

* **Description:**
  + As an End User, I should be able to share some of the available graphs in the dashboard, mainly the average fare price graph. The graph should have a timestamp and a subtle but easily recognizable watermark with the logo of the team.
* **Acceptance Criteria:**
  + The visualization must be clear and contain the same information as in the dashboard.
  + It must work with the specified graphs.
  + The site administrator should be able to decide which graphs can be shared

**User Story:** Filter data by Origin, Destination, Airlines (WFMG-21)

* **Description:**
  + As an End User and as a developer, I should be able to filter data by attributes such as Origin, Destination, and Airlines
* **Acceptance Criteria:**
  + The user must be able to select among a specified set of attributes
  + The system should give the user options for what to choose.

**User Story:** Improve normalization of data in the coupon distribution (WFMG-27)

* **Description:**
  + As an End User I would like to see the data for the coupon distribution without the columns that are 0.0% of the population.
* **Acceptance Criteria:**
  + The bars should only appear if they are above a certain limit.

**User Story:** Create Database Schema (WFMG-37)

* **Description:**
  + As a developer, I would like the data to be structured in an efficient way so that I can access it and modify it effectively.
  + The team has established that MongoDB should be used as the managing software for our NOSQL database.
* **Acceptance Criteria:**
  + The database must contain all the data that is right now in csv format.

**User Story:** Process New Data and add it to the Database (WFMG-38)

* **Description:**
  + As a Developer, I would like the web application to work with the latest data available from different sources, such as T100, DB1B, etc. The website should check the sources periodically to verify if the data is up to date. If there is new data, the application should fetch the data and add it to the database.
* **Acceptance Criteria:**
  + The application should fetch the data when it is available.
  + The system should also add it to the data base and preprocess it according to the source.

# Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint planning’s are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

* **Visual Studio** 2017 Enterprise - as the IDE
* **Flask** - as a micro web framework.
* **Dash** - as the user interface. Dash uses react and plotly.js to manage the interface using python.
* **Plotly** - as the framework for visualizations (line plots, bar plots, etc…)
* H**tml, CSS3, JavaScript** - as the front-end programming languages.
* **Heroku, gunicorn** - as the deployment platform.
* **Redis**, **flask-caching** - as a cache to store the result of computations that require a significant amount of time. Also, can be used to maintain a shared part of memory between threads (and between callbacks)
* **Bootstrap** - as part of the front-end framework
* **Bitbucket** – for repository and versioning control
* **Selenium** – to automate part of the testing process.
* **SciPy** (pandas, numpy, scikit-learn) and Dask - as an ecosystem to process csv files and create the plug-in architecture for machine learning models.

## Sprints Plan

### Sprint 1

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 11:00am

End time: 12:30pm

After discussion, the velocity of the team was estimated to be 50

Our goal for this sprint is to familiarize ourselves with the technologies that the team of graduate students used to get the data and create their data visualizations. We Also plan on creating a rough sketch of the user interface and decide on the technologies required for the implementation of the web application.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-13 User Interface
* User Story: WFMG-12 Display data as a line plot
* User Story: WFMG-2 Display Distribution of data
* User Story: WFMG-3 Filter Data by time
* User Story: WFMG-4 Organize charts into columns to identify trends

The team members indicated their willingness to work on the following user stories.

* Tomas Ortega
  + User Story: WFMG-13 User Interface
    - Create a template for organizing the graphs
    - Create a Navigation Bar
  + User Story: WFMG-12 Display data as a line plot
* Serge Metellus
  + User Story: WFMG-2 Display Distribution of data
  + User Story: WFMG-3 Filter Data by time
  + User Story: WFMG-4 Organize charts into columns to identify trends

### Sprint 2

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 11:00 AM

End time:  12:00 PM

After discussion, the velocity of the team was estimated to be 50

The goal for this sprint is to use the predictive model in the visualizations and finish the user stories from sprint 1

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-19 Fare Price Visualization
* User Story: WFMG-2 Display Distribution of data
* User Story: WFMG-3 Filter data by time
* User Story: WFMG-4 Organize charts into columns to identify trends

The team members indicated their willingness to work on the following user stories.

* Serge Metellus
  + User Story: WFMG-3 Filter data by time
  + User Story: WFMG-4 Organize charts into columns to identify trends
* Tomas Ortega
  + User Story: WFMG-19 Fare Price Visualization
  + User Story: WFMG-2 Display Distribution of data

### Sprint 3

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 11:00 AM

End time:  12:00 PM

After discussion, the velocity of the team was estimated to be 52.

The goal for this sprint is to implement the necessary classes for the inference and visualization models and to include them in the view and controller of the application.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-24 Visualization wrapper
* User Story: WFMG-23 Model wrapper
* User Story: WFMG-22 Adjust Prediction Parameters
* User Story: WFMG-25 Fix column layout
* User Story: WFMG-26 Modify the fare price visualization

The team members indicated their willingness to work on the following user stories.

* Tomas Ortega
  + User Story: WFMG-23 Model wrapper
  + User Story: WFMG-22 Adjust Prediction Parameters
  + User Story: WFMG-26 Modify the fare price visualization
* Serge Metellus
  + User Story: WFMG-24 Visualization wrapper
  + User Story: WFMG-25 Fix column layout

### Sprint 4

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 2:30 PM

End time:  3:30 PM

After discussion, the velocity of the team was estimated to be 59.

The goal for this sprint is to finish the visualization classes and start working on the data collection and being able to change the historical data by dragging the data points across the screen.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-24 Visualization wrapper
* User Story: WFMG-32 Description of data files
* User Story: WFMG-33 Draggable points
* User Story: WFMG-34 Drag multiple data points

The team members indicated their willingness to work on the following user stories.

* Tomas Ortega
  + User Story: WFMG-34 Drag multiple data points
  + User Story: WFMG-33 Draggable points
* Serge Metellus
  + User Story: WFMG-24 Visualization wrapper
  + User Story: WFMG-32 Description of data files

### Sprint 5

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 1:30 PM

End time:  2:00 PM

After discussion, the velocity of the team was estimated to be 47.

The goal for this sprint is to get the data science pipeline going. We are going to add data cleaning and pre-processing steps to the workflow manager application.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-34 Drag multiple data points
* User Story: WFMG-35 Reset graph button
* User Story: WFMG-39 Data processing/preprocessing
* User Story: WFMG-40 Retrain Model on Point Movement Events

The team members indicated their willingness to work on the following user stories.

* Tomas Ortega
  + User Story: WFMG-34 Drag multiple data points
  + User Story: WFMG-40 Retrain Model on Point Movement Events
  + User Story: WFMG-35 Reset graph button
* Serge Metellus
  + User Story: WFMG-39 Data processing/preprocessing

### Sprint 6

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 1:00 PM

End time:  2:00 PM

After discussion, the velocity of the team was estimated to be 41.

The goal for this sprint is to implement the necessary classes for the inference and visualization models and to include them in the view and controller of the application.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-41 Random Forest Model
* User Story: WFMG-42 Random Forest Visualization
* User Story: WFMG-43 Multivariate Time Series Analysis Model
* User Story: WFMG-44 Fix Layout and Styling issues
* User Story: WFMG-45 User Interface Improvements

The team members indicated their willingness to work on the following user stories.

* Tomas Ortega
  + User Story: WFMG-41 Random Forest Model
  + User Story: WFMG-42 Random Forest Visualization
  + User Story: WFMG-43 Multivariate Time Series Analysis Model
* Serge Metellus
  + User Story: WFMG-44 Fix Layout and Styling issues
  + User Story: WFMG-45 User Interface Improvements

### Sprint 7

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* AirLab Team

Start time: 1:00 PM

End time:  2:00 PM

After discussion, the velocity of the team was estimated to be 35.

The goal for this sprint is to complete all the required documentation for the final project and prepare for the presentation.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* User Story: WFMG-46 Presentation
* User Story: WFMG-47 Document
* User Story: WFMG-48 Prepare the workflow manager app for presentation demo

The team members indicated their willingness to work on the following user stories.

* Tomas Ortega
  + User Story: WFMG-47 Document
* Serge Metellus
  + User Story: WFMG-48 Prepare the workflow manager app for presentation demo
  + User Story: WFMG-46 Presentation

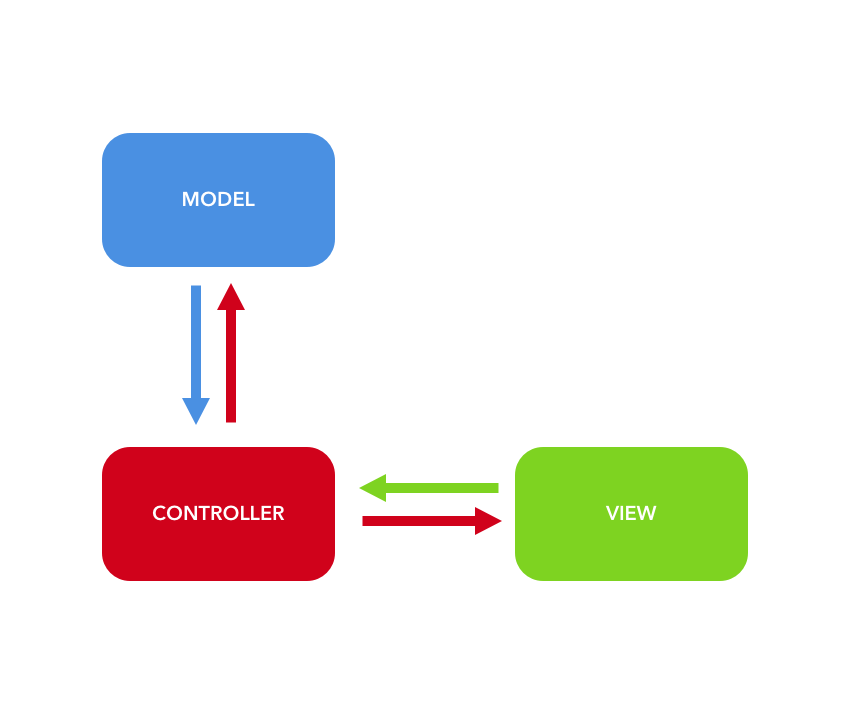
# 

# System Design

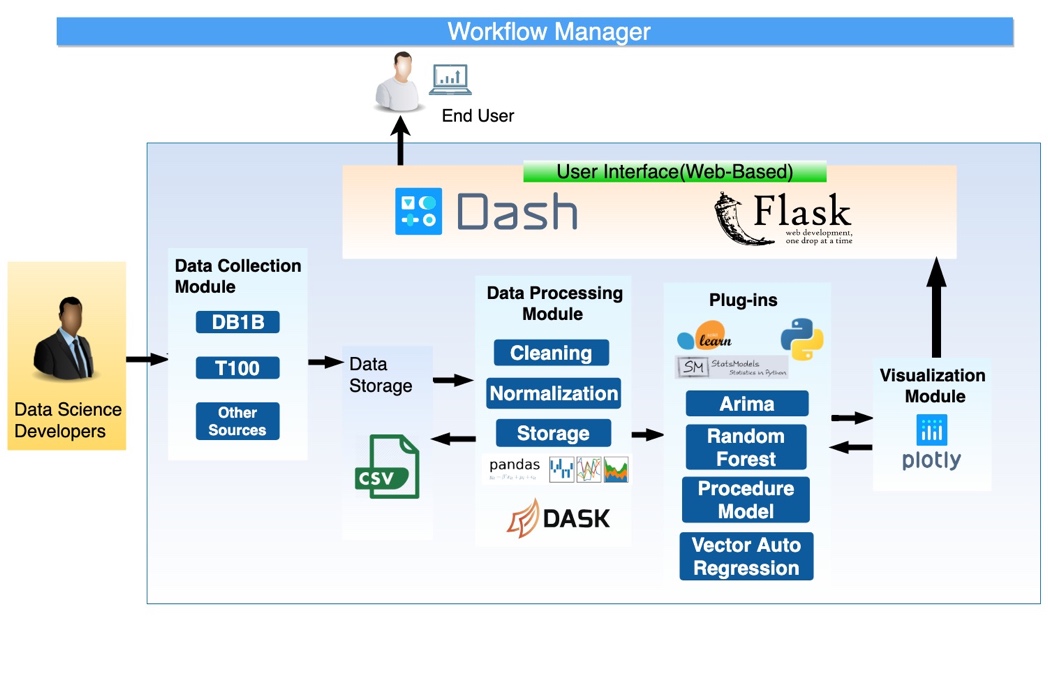
This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

The system was designed using the Model, View, Controller (MVC) Architecture to provide different levels of abstraction to the developers working on new plugins or working on the view components. The Model component corresponds to all the data-related logic that the user works with. All data used in the visualizations are either processed in the model or processed elsewhere and imported to the model. The View component is used for all the user interface logic of the application. It allows the end users to see and interact with the visualizations of the models and see the different data distributions by clicking through the page.

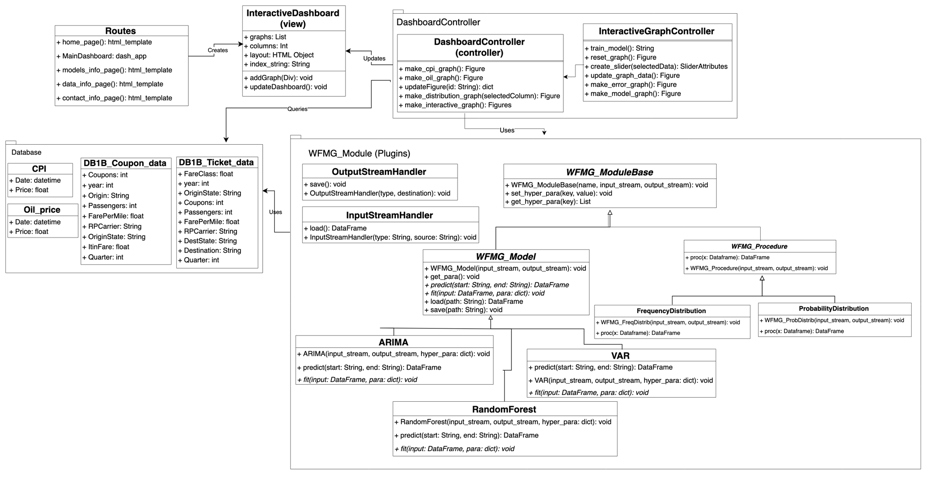
Controllers act as an interface between the Model and View components to process all the logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output and provide an interactive experience to the user. The controller receives the input, it validates the it and then performs the operation that modifies the state of the data model.

## System Architecture Diagram



## System and Subsystem Decomposition

The subsystem decomposition is illustrated in the following class diagram In



## Deployment Diagram

## Design Patterns

# Deployment Diagram

# 

# System Validation

## Unit Tests

* **Test case ID**: WFMG-2-UT
* **Description/Summary of Test**: Validate that the user is able to select different graphs for data distributions such as state of departure distributions and fare class distributions. The test was performed specifically with the following graphs: Coupon Distribution, Fare Class Distribution, State of Departure Distribution, Market Share Distribution.
* **Pre-condition:** The application must be running, and the data must be present in the server. In particular, the data must be in the folder under the directory /Data so that the view module can access the data for visualizations.
* **Expected Results:** The user should be able to see the graph for the specified data distribution in the dashboard inside of the Workflow Manager application.
* **Actual Result:** The results are as expected, the application opens with the default graph and changes with the user input.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-3-UT
* **Description/Summary of Test**: Validate that the user can select dates from the calendars and the graph updates using those dates
* **Pre-condition:** The user must be in the dashboard page
* **Expected Results:** The user should be able to see the graph for the specified date range.
* **Actual Result:** The results are as expected; the application opens with the default graph and changes with the user input.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-4-UT
* **Description/Summary of Test**: Validate that the user is able to select the number of columns and the dashboard should be displayed with those columns
* **Pre-condition:** The user must be in the dashboard page
* **Expected Results:** The user should be able to see the dashboard and graphs divided into the specified number of columns
* **Actual Result:** The results are as expected when the number of graphs is divisible by the number of columns, otherwise the graphs break to accommodate column space
* **Status (Fail/Pass):** Fail
* **Test case ID**: WFMG-12-UT
* **Description/Summary of Test**: Validate that the user is able to see the graphs in the dashboard and that they are formatted correctly in the browser. The graphs must be reactive to changes in the size of the window so we also tested the ability to change the width of the graphs.
* **Pre-condition:** The application must be running, and the data must be present in the server. The data from the different databases (T100 and DB1B) must be in the folder under the directory /Data so that the view module can access the data for visualizations.
* **Expected Results:**  The user should be able to see the graphs with lines connecting each data point, The graphs must change the width of the data displayed when the window size changes.
* **Actual Result**: The results are as expected, the dashboard opens successfully with graph showing correctly.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-13-UT
* **Description/Summary of Test**: Validate that the user is able to access the website and the dashboard when the application is locally hosted
* **Pre-condition:** The public user must have access to the url, the application must be running, and the data must be present in the server.
* **Expected Results:** The user should be able to see the home page with side navigation and must have the ability to click on the dashboard link to open it.
* **Actual Result:** The results are as expected, the application opens with the home page and allows the user to click on the dashboard.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-14-UT
* **Description/Summary of Test**: Validate that the developer is able to download and store data in specific directories.
* **Pre-condition:** The data source needs to be accessible.
* **Expected Results:** The developer should be able to download the data and store it in a corresponding directory.
* **Actual Result:** The results are as expected, the developer is able to download the data and store it in a corresponding directory.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-19-UT
* **Description/Summary of Test**: Validate that the user is able to see the graph on the dashboard and that the slider works as expected. The slider must react to clicks or to dragging the bars at both ends of the range slider.
* **Pre-condition:** The End user must have the workflow manager open in a browser window. The data for the fare price must be present in the server. The data files for the fare price must be in the folder /Data
* **Expected Results:** The user should be able to see the graph for the average fare price visualization and the data from the arima model (In this case, the data from the model is coming from a standard CSV file because the plug-in architecture has not been implemented.
* **Actual Result:** The results are as expected, the is visible from the dashboard and the user can select the range that he wants to inspect in the graph.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-22-UT
* **Description/Summary of Test**: Validate that the private user is able to create models and use the set and get hyper parameters methods to change and update the parameters for the machine learning models.
* **Pre-condition:** The developer working on a data science project (private user) must have the wfmg module in the present working directory or imported from somewhere else and have an object that inherits from the WFMG\_model class.
* **Expected Results:** The developer should be able to create new models by inheriting from the WFMG\_Model class and use the method from that class to load, save, fit and predict data from the model. The parameters used by the model should be available to the developer working with the WFMG\_model object and change them whenever he wants.
* **Actual Result:** The results are as expected, the models that inherit from the class WFMG\_Model are able to use the methods from that class, such as the set and get\_hyper\_para functions and they return the correct results.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-23-UT
* **Description/Summary of Test**: Validate that the plug-in/model developer is able to create models and use the specified methods on model objects.
* **Pre-condition:** The plug-in/model developer must have the wfmg module in the present working directory or imported from somewhere else and he should be using python 3.6 or higher.
* **Expected Results:** The private user should be able to create new models by inheriting from the WFMG\_Model class and use the method from that class to load, save, fit and predict data from the model.
* **Actual Result:** The results are as expected, the models that inherit from the class WFMG\_Model are able to use the methods from that class and they return the correct results.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-24-UT
* **Description/Summary of Test**: Validate that the developer can instantiate the distribution classes and use the specified methods.
* **Pre-condition:** The developer must have the WFMG module in the present working directory or imported from somewhere else.
* **Expected Results:** The developer should be able to instantiate the Probability and Frequency sub classes.
* **Actual Result:** The results are as expected, the models used to instantiate the distribution sub classes can use the methods from those classes and the inherited methods from the superclass.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-33-UT
* **Description/Summary of Test**: Validate that the end users are able to move individual data points using the slider on the right side of the graph and that it is updated after the user lifts the mouse from the slider. The points that have been modified should remain the same after the user moves other data points.
* **Pre-condition:** The user must have the dashboard page loaded on his/her browser and the Predictive graphs should be visible in the page.
* **Expected Results:** The end user should be able to click one of the points in the data set used to train the model and move it vertically using the slider to set a different y-value. When the user lifts the mouse, the graph should update and move the selected point to the desire position.
* **Actual Result:**  The results are as expected, the points can be moved by the user and the graph is updated when the user lets go of the slider. The changes remain after the user moves a different point. It also works when the end user changes the range using the range slider, so that the user can zoom into the graph and change data points in a precise manner.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-34-UT
* **Description/Summary of Test**:  Validate that the end users are able to move individual data points using the slider on the right side of the graph and that the system is able to react to the point movement event
* **Pre-condition:** The user must have the dashboard page loaded on his/her browser.
* **Expected Results:** The end user should be able to click one of the points in the data set used to train the model and move it vertically using the slider. When the user lifts the mouse, the graph should update and move the point to the desire position.
* **Actual Result:** The results are as expected, the points can be moved by the user and the graph is updated when the user lets go of the slider.
* **Status (Fail/Pass):** Pass
* **Test case ID**: WFMG-39-UT
* **Description/Summary of Test**: Validate that the developers are able to process raw data
* **Pre-condition:** Each data source must have its own directory and python script that specifies the processing steps
* **Expected Results:** The developer should be able to use Make to specify the scripts and order of execution
* **Actual Result:** The results are as expected, the makefile successfully runs the scripts that process the data and store it in a separate directory.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-40-UT
* **Description/Summary of Test**: Validate that the end users are able to train the machine learning models used to predict data after every point movement or after clicking on the Train button on the right side of the graph. After the user trains the model using one of the two methods, then the graph should be updated to reflected the new prediction.
* **Pre-condition:** The user must have the dashboard page loaded on his/her browser.
* **Expected Results:** The end user should be able to click on the train button after moving on or several data points to train the machine learning model. If the checkbox is not checked, then the model is going to be trained in every point movement.
* **Actual Result:** The results are as expected, the points can be moved by the user and then click on the train button to train the machine learning model and see the results. If the checkbox is not checked, then the model is trained on every point movement (it takes longer to update)
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-41-UT
* **Description/Summary of Test**: Validate that the random forest model works. this means that the functions for loading, saving, fitting and predicting/forecasting, and for getting the feature importance should work accordingly to the behavior described in the Scikitlearn library.
* **Pre-condition:** The developer working on a data science project (private user) must have the wfmg module in the present working directory or imported from somewhere else.
* **Expected Results:** The developer should be able to instantiate an object from the Random\_Forest class and predict new data points using the predict function.
* **Actual Result:** The results are as expected, the random forest model object can be initialize and the prediction and fitting functions work as described in the Scikitlearn library. It can also use the save and load function to use the model as a pickle.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-43-UT
* **Description/Summary of Test**: Validate that the end users are able to move individual data points using the slider on the right side of the graph, that the user can click on the reset button to erase any changes made to the VAR graph, that the train button works as expected, and finally that the range slider and the width and hight of the graphs are working correctly if the size of the window changes.
* **Pre-condition:** The user must have the dashboard page loaded on his/her browser.
* **Expected Results:** The end user should be able to click one of the points in the data set used to train the model and move it vertically using the slider. When the user lifts the mouse, the graph should update and move the point to the desire position. The end user should also be able to click on the train or on the reset button to change the configuration of the graph. Finally the end user should be able to change the range of the x-axis of the graph using the range slider.
* **Actual Result:** The results are as expected, the points can be moved by the user, the training and reset button works as expected and the range slider successfully changes the range of x-range of the graph.
* **Status (Fail/Pass):** Pass

## 

## Integration Tests

* **Test case ID:** WFMG-2-IT
* **Description/Summary of Test:** verify that the graph and the input components work inside of the dashboard environment in the Workflow manager application.
* **Pre-condition:**The application must be installed and running in the browser.
* **Expected Results:**The graph for the specified data distribution must be present in the dashboard. If no options is selected then the Coupon Distribution should be shown.
* **Actual Result:**the graph changes with the user input and the default options appears to be working.
* **Status (Fail/Pass):**Pass
* **Test case ID:** WFMG-3-IT
* **Description/Summary of Test:** Verify that the graph and the input works inside the dashboard environment.
* **Pre-condition:** The application must be installed and running.
* **Expected Results:** The graph with the default range along with the Datepicker must be present in the dashboard.
* **Actual Result:** the graph changes with the user input and the default options appears to be working.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-4-IT
* **Description/Summary of Test:** Verify that the dashboard responds to input.
* **Pre-condition:**The application must be installed and running.
* **Expected Results:**The dashboard should update based on specified input.
* **Actual Result:** The dashboard only changes under certain conditions when it should work for all.
* **Status (Fail/Pass):**Fail
* **Test case ID:** WFMG-12-IT
* **Description/Summary of Test:** Validate that the graphs are displayed correctly in the dashboard and that we can have at least 7 graphs showing in the browser at the same time.
* **Pre-condition:**The application must be installed and the workflow manager must be running in the browser.
* **Expected Results:**The graphs show correct as specified in the view.
* **Actual Result:**We are to see the graphs in the dashboard and they are displayed correctly.
* **Status (Fail/Pass):**Pass
* Test case ID: WFMG-13-IT
* **Description/Summary of Test:** Validate that the home page can be accessed through the dashboard and that the dashboard is accessible through the home page. Also that the graphs can be displayed in the dashboard.
* **Pre-condition:**The application must be installed and is also must be running.
* **Expected Results:**The side navigation bar works in every direction and from any page.
* **Actual Result:**We are able to navigate to the home page from the dashboard and back.
* Status (Fail/Pass): Pass
* **Test case ID:** WFMG-19-IT
* **Description/Summary of Test:** Validate that the graph is accessible from the dashboard that was previously stablished. And that all the components related to the graph (including the range slider) are working as expected in the workflow manager web application.
* **Pre-condition:**The browser must be open with the workflow manager application running.
* **Expected Results:**the graph is visible and the range slider is working.
* **Actual Result:**The graph is visible from the dashboard with the expected style and arrangement, and the slider is working perfectly.
* **Status (Fail/Pass):**Pass
* **Test case ID:** WFMG-22-IT
* **Description/Summary of Test:** Validate that the methods can modify the parameters of a machine learning models and change the configuration of the models that are going to be used by the visualization module.
* **Pre-condition:**The developer or private user should have the web application installed and have every class that's involved in the plug-in architecture present in his/her current directory.
* **Expected Results:**The configuration of the model should be updated by the developer every time that he uses the methods get\_hyper\_para and set\_hyper\_para.
* **Actual Result:**The model can update the parameters through the functions provided in the WFMG\_model class.
* **Status (Fail/Pass):**Pass
* **Test case ID:** WFMG-23-IT
* **Description/Summary of Test:** Validate that the visualizations in the graphs are able to use the models and the methods from the wfmg module.
* **Pre-condition:**The application must be installed and the workflow manager must be running in the browser.
* **Expected Results:**the graph should be the same as the graphs using data coming directly from the csv files but the data should come from a model object such as ARIMA.
* **Actual Result:**The graph is visible from the dashboard with the expected style and arrangement and the data points come from the model object.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-24-IT
* **Description/Summary of Test:** Validate that the visualizations in the graphs can use the methods from the wfmg module.
* **Pre-condition:** The application (browser) must be installed and running.
* Expected Results: The graph should use data from a dataframe return by a call to the proc method.
* **Actual Result:** The graph is visible from the dashboard with the expected style and arrangement and the data points come from a dataframe returned by the proc method.
* **Status (Fail/Pass):** Pass

* **Test case ID:** WFMG-33-IT
* **Description/Summary of Test:** Validate that the visualizations that are part of the Predictive graphs can use the slider to move the points that are part of the data set used to train the machine learning model.
* **Pre-condition:**The application must be installed and the workflow manager must be running in the browser.
* **Expected Results:**The graphs that use the slider must be visible from the dashboard.
* **Actual Result:**The graphs are visible from the dashboard with the expected style and arrangement of the data points coming from the model object and from the data set used to train the model.
* **Status (Fail/Pass):**Pass
* **Test case ID:** WFMG-34-IT
* **Description/Summary of Test:** Validate that the visualizations can use the slider to move the points in the graph that are part of the data set used to train the model.
* **Pre-condition:**The application (browser) must be running and the user must have the dashboard page loaded.
* **Expected Results:**The graphs that use the slider must be visible from the dashboard.
* **Actual Result:**The graphs are visible from the dashboard with the expected style and arrangement of the data points coming from the model object and from the data set used to train the model.
* **Status (Fail/Pass):**Pass
* **Test case ID:** WFMG-39-IT
* **Description/Summary of Test:** Validate that the model visualizations can use the data processed by the makefile and scripts
* **Pre-condition:**The application (browser) must be running and the user must have the dashboard page loaded.
* **Expected Results:**The model graphs should use the data processed by scripts
* **Actual Result:**The graphs are visible from the dashboard with the expected arrangement of the data points coming from the model object which uses the data processed by the python scripts.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-40-IT
* **Description/Summary of Test:** Validate that the visualizations for the machine learning models (Predictive Graphs) are visible in the dashboard and that the graph, including the trace for the machine learning model is updated when the user interacts with the training data.
* **Pre-condition:**The workflow manager application must be running in the browser and the user must have the dashboard page loaded.
* **Expected Results:**The Predictive graphs must be visible in the dashboard and react to user input for training the machine learning models.
* **Actual Result:**The graphs are visible from the dashboard with the expected style and arrangement of the data points coming from the machine learning model objects and from the data set used to train the model. The graph updates every time the user clicks on the train button and if the checkbox is not checked, the graph is updated on every point movement.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-41-IT
* **Description/Summary of Test:** Validate that random forest model can be used by the visualization module to get the prediction data, feature importance and that the prediction data from the model is valid and visualized in the prediction graphs in the dashboard.
* **Pre-condition:**The browser must be open with the workflow manager application running.
* **Expected Results:**The model should work with the visualization model and the prediction data coming from the model should be a dataframe with the date and the predicted fare price for each of the dates in the range provided.
* **Actual Result:**The prediction data from the model has the expected format and the visualization module can access it.
* **Status (Fail/Pass):** Pass
* **Test case ID:** WFMG-43-IT
* **Description/Summary of Test:** Validate data from the VAR model is in fact reaching the visualization module and that the training data is updated every time the user moves a single point or a group of points. We should also validate that the components that belong to the predictive graph for this model work as expected.
* **Pre-condition:**The browser must be open with the workflow manager application running.
* **Expected Results:**The graph for the VAR model should be visible in the dashboard and all the components (slider, range slider, buttons for resetting and training the model) should work as in the rest of the predictive model graphs.
* **Actual Result:**The graphs are visible from the dashboard with the expected style and arrangement. The components also work as in the other model graphs and the server is able to fetch the modified training data set.
* **Status (Fail/Pass):**Pass

# Glossary

**Agile Development:** is an approach to software development under which the requirement and solutions evolve through the collaborative efforts of self-organizing and cross-functional teams and the consumers of that product.

**Integration Testing:** Is a phase of testing in which the parts of the software are tested together to see if the integrations of the individual components will break the application.

**Unit Testing:** It is part of the software testing process. In unit testing, the developer will test the smallest testable parts of the application called units and they are scrutinized for proper operation.

**User Story:** a description of a desired functionality of the software from the point of view of the specific user that’s going to use that particular feature.

**MVC:**  is an architectural pattern used for developing user interfaces that divides the application into three interconnected parts, the Model, the View and the Controller. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

**Web Application:** a software application that runs on a remote server. In most cases, Web browsers are used to access web applications, over a network, such as the internet.

**ARIMA:** An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends.

**VAR:** Vector autoregression (VAR) is a stochastic process model used to capture the linear interdependencies among multiple time series. In the case of the WFMG project, the time series that we used are Fare Price, Oil Price, and Consumer Price Index.

**Random Forest Model:** Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

**Probability Distributions:** statistical function that describes all the possible values and likelihoods that a random variable can take within a given range.

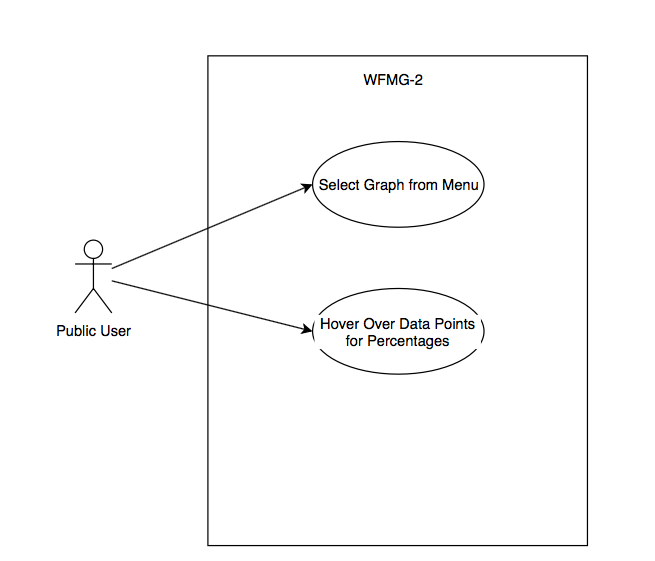
**Frequency distributions:** is a representation, either in a graphical or tabular format, that displays the number of observations within a given interval.

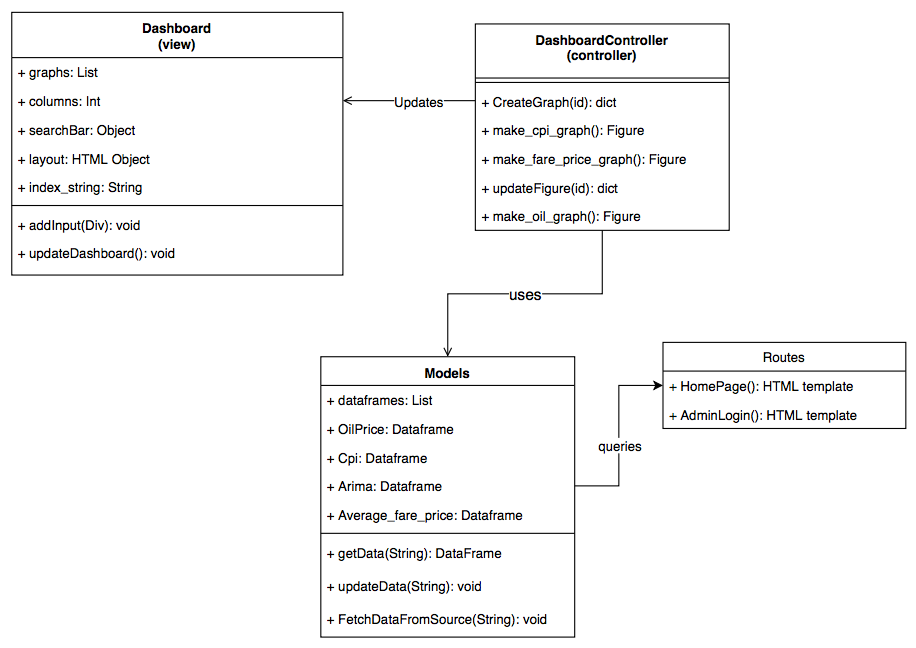
# Appendix

## Appendix A - UML Diagrams

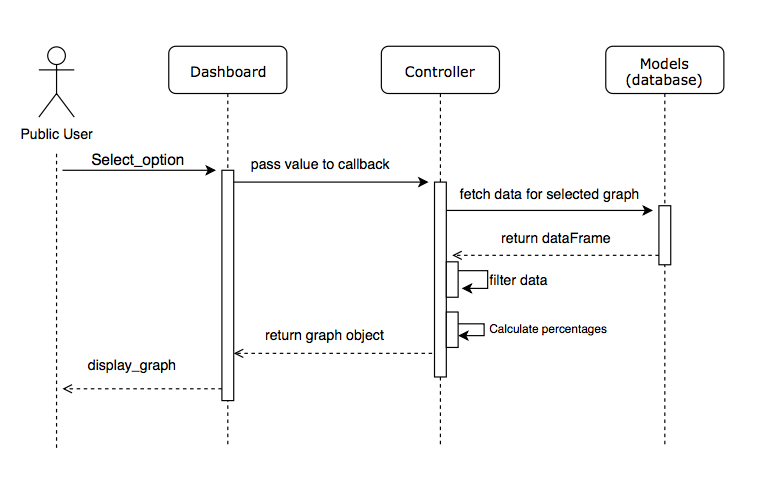
In this section we have some of the diagrams used in the WFMG project.

**WFMG-2 – Display Distribution of data:**

Use Case Diagram:

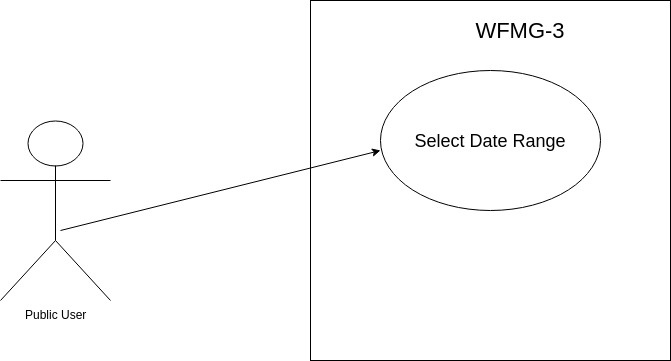
Class Diagram:

Sequence Diagram:

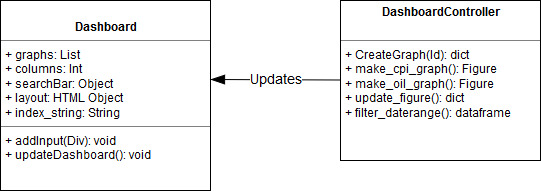


**WFMG-3 Filter Data by time**

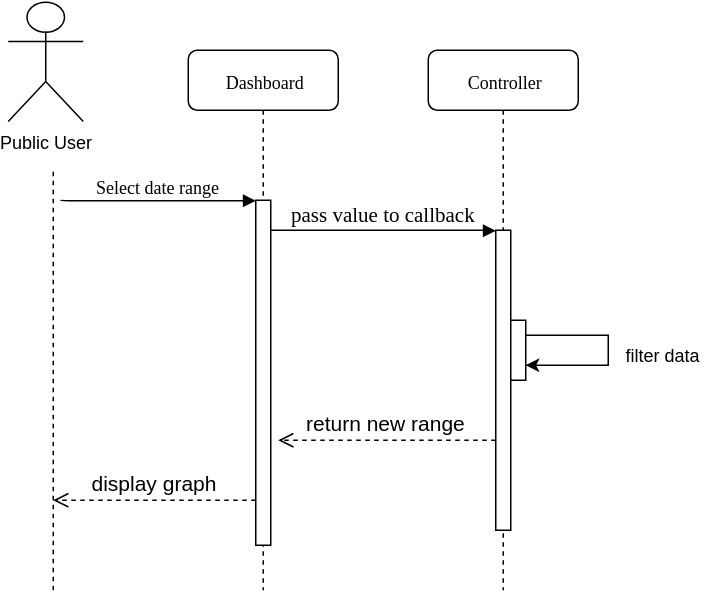
Use Case Diagram:



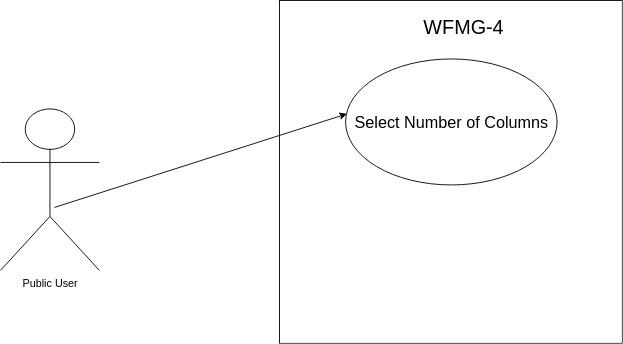
Class Diagram:

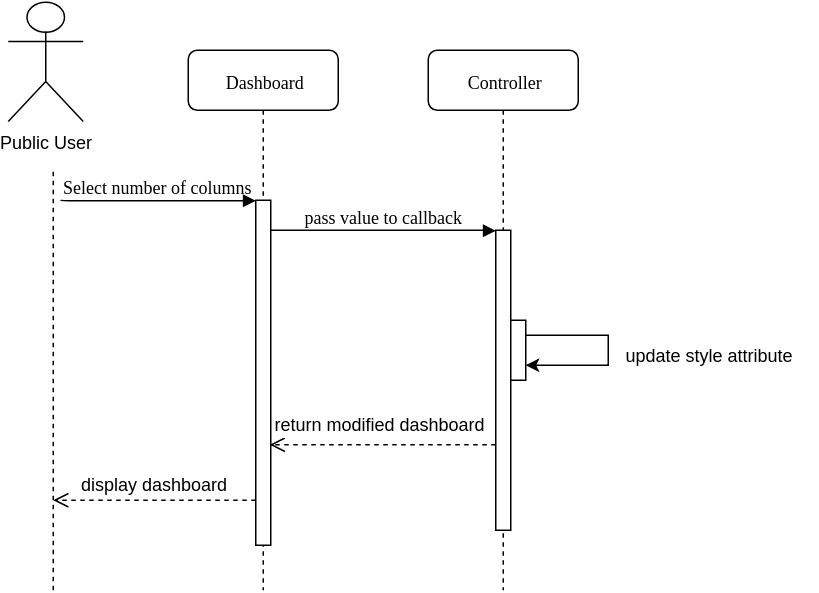


Sequence Diagram:

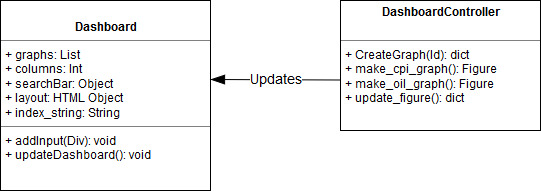


**WFMG-4 Organize charts into columns to identify trends**

Use Case Diagram:

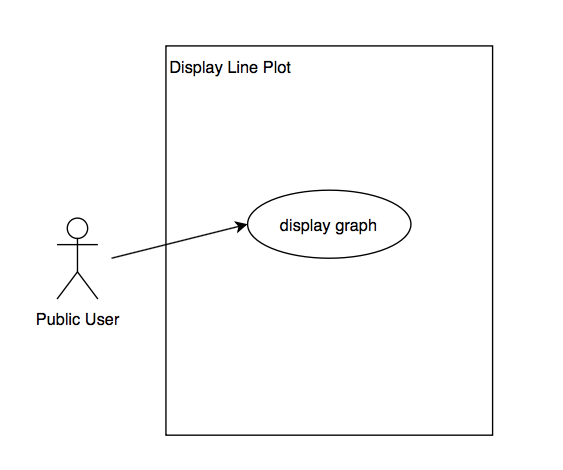
Sequence Diagram:

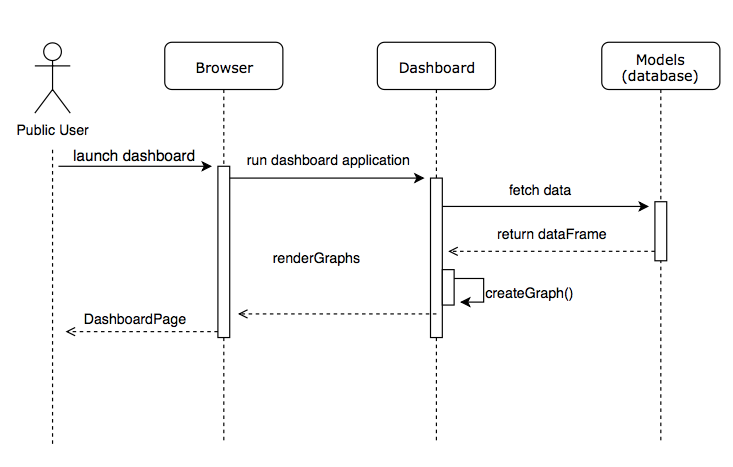
Class Diagram:



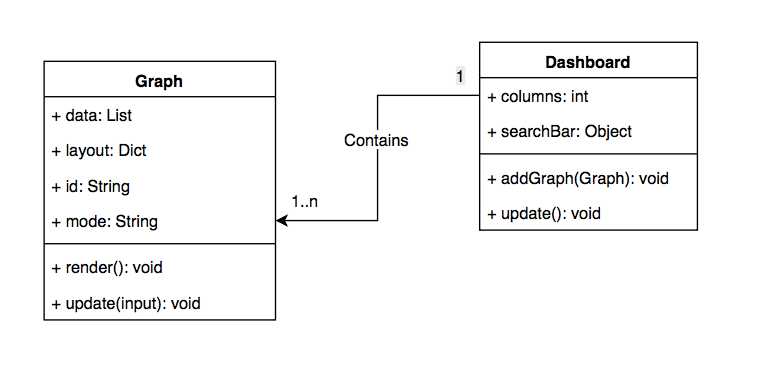
**WFMG-12 Display data as a line plot**

Use Case Diagram:

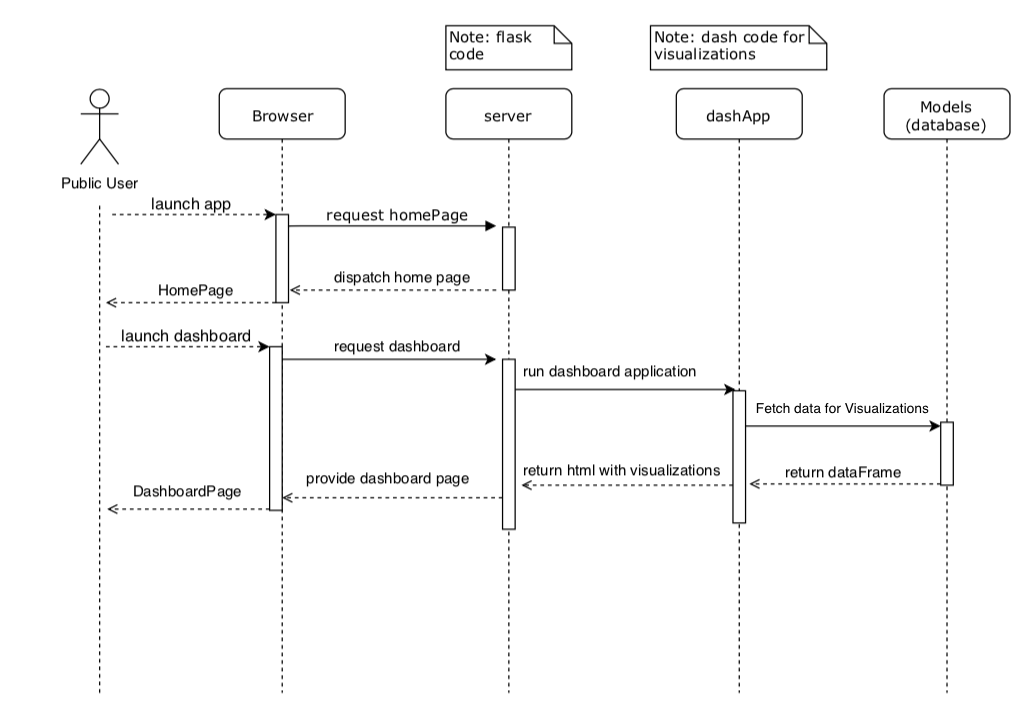
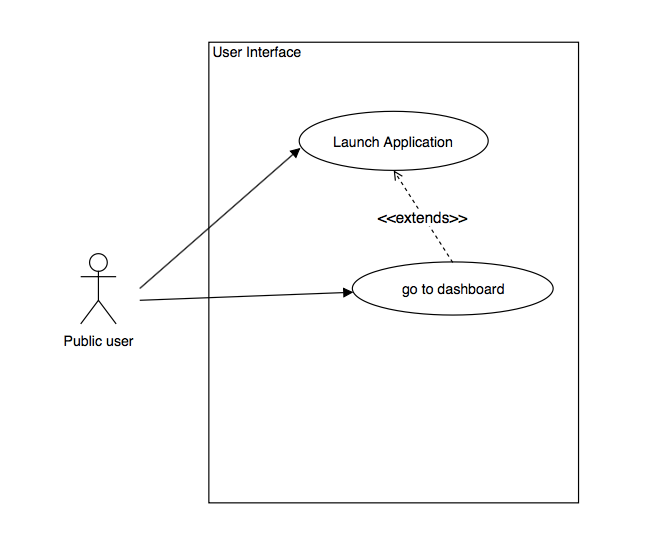


Sequence Diagram:

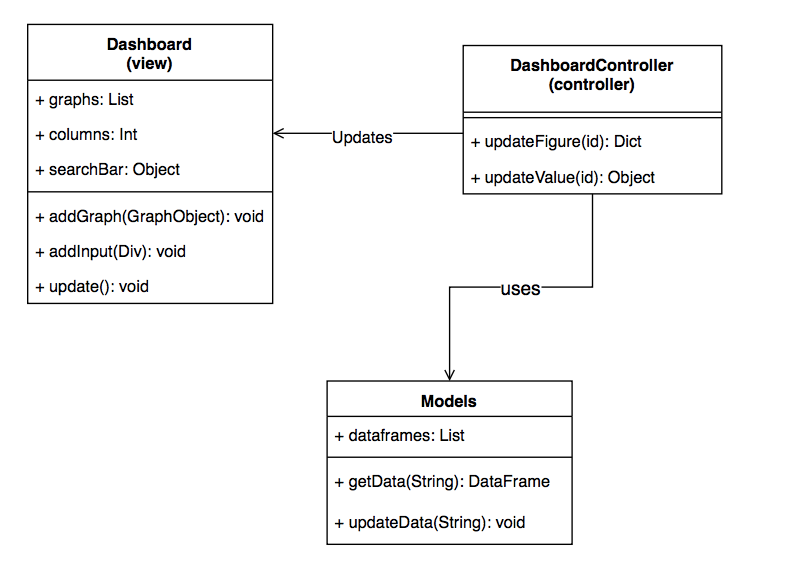
Class Diagram:



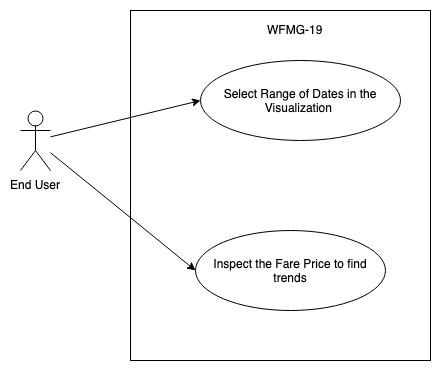
**WFMG-13 Dashboard Access**

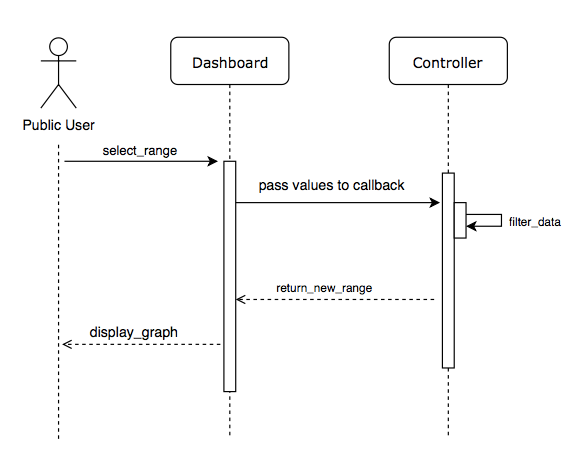
Use Case Diagram:

Sequence Diagram:

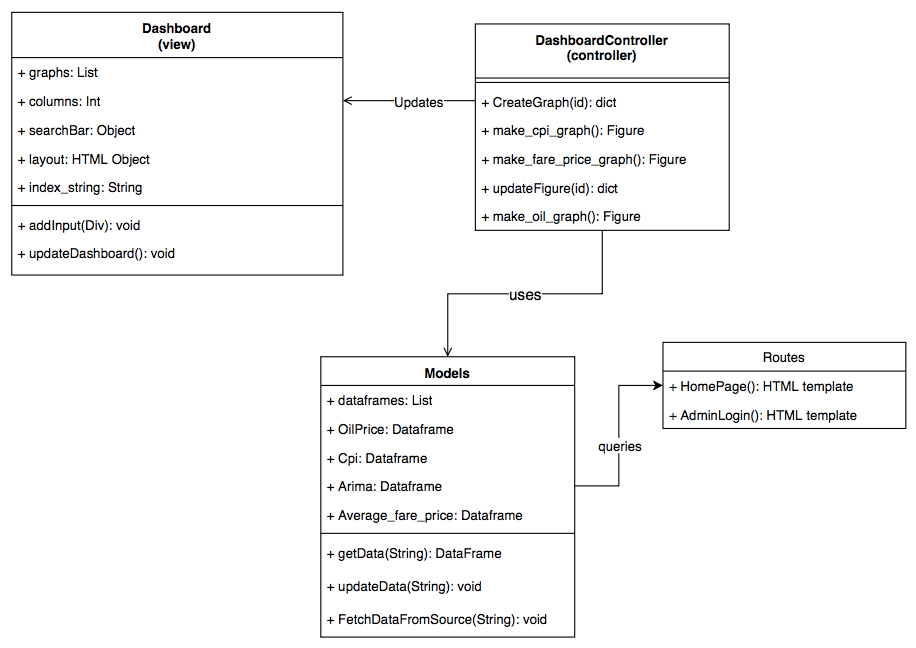
Class Diagram:

**WFMG-19 Fare Price Visualization**

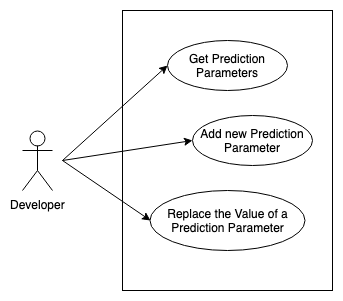
Use Case Diagram:

Sequence Diagram:

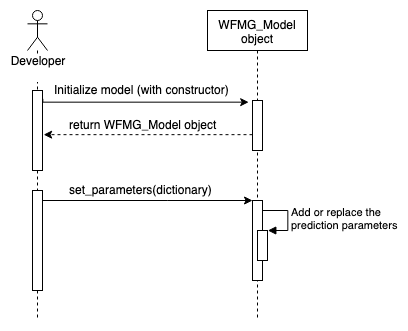
Class Diagram:

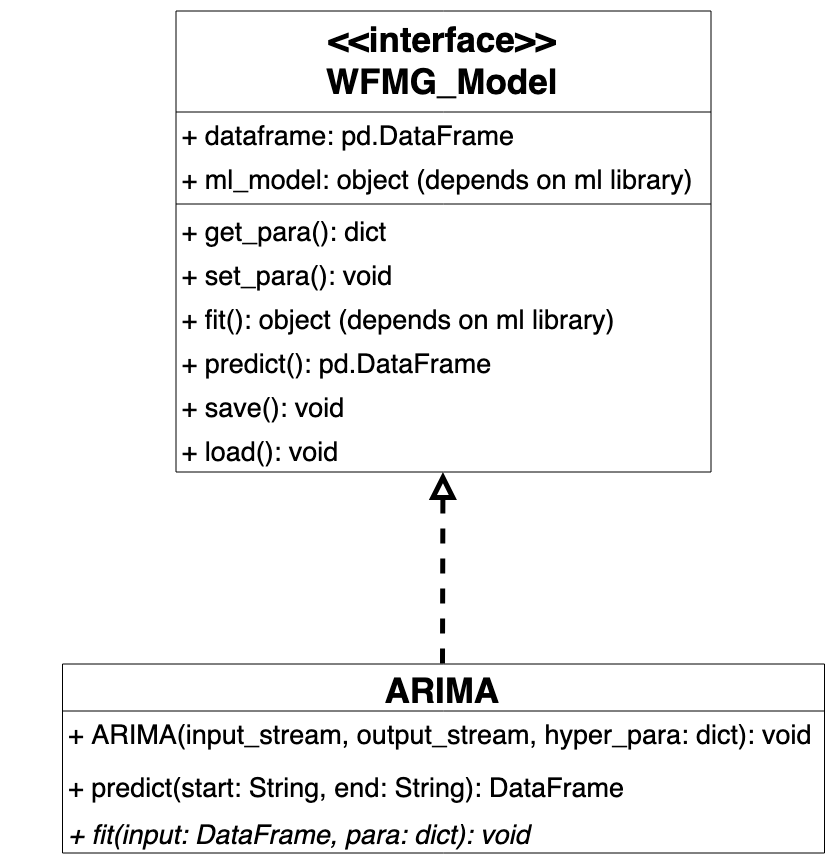


**WFMG-22 Adjust Prediction Parameters**

Use Case Diagram:

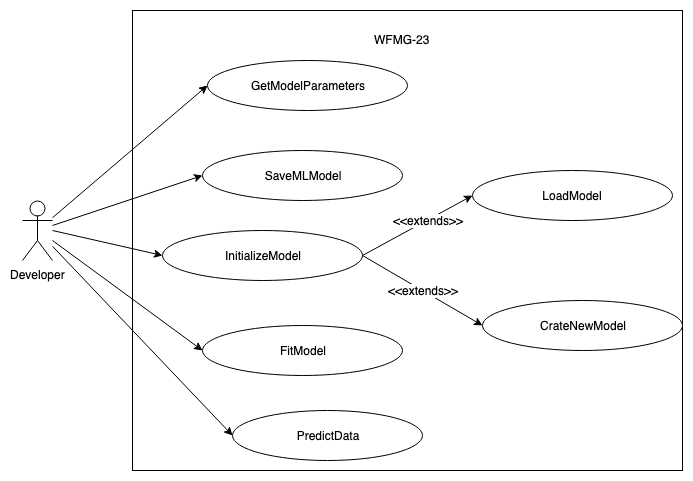
Sequence Diagram:

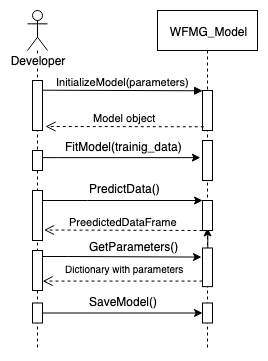


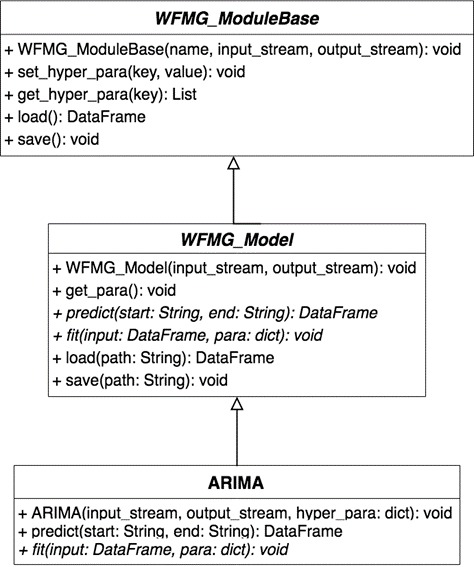
Class Diagram:

**WFMG-23 Model wrapper**

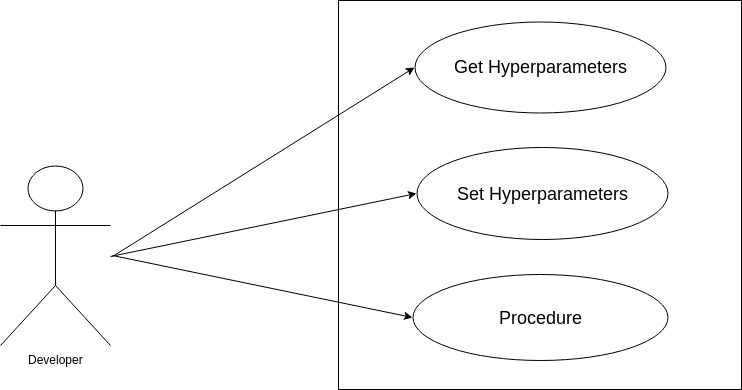
Use Case Diagram:



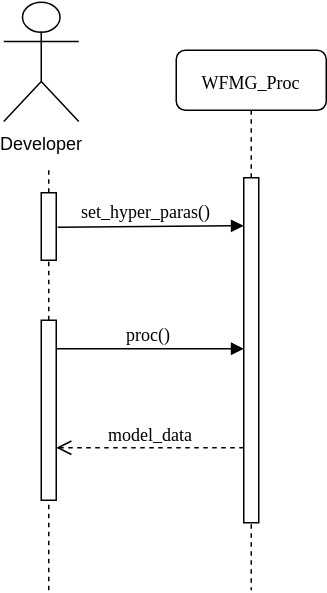
Sequence Diagram:

Class Diagram:

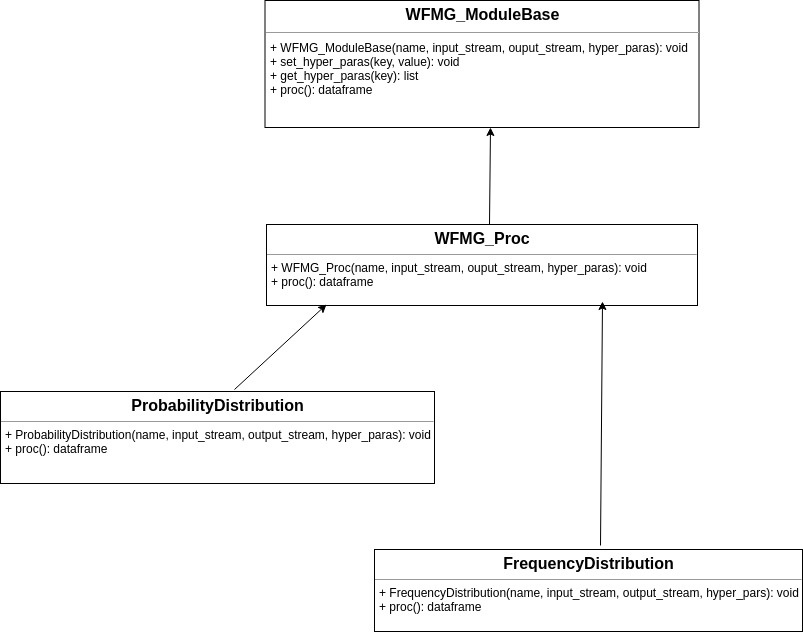
**WFMG-24 Visualization wrapper**

Use Case Diagram:

Sequence Diagram:

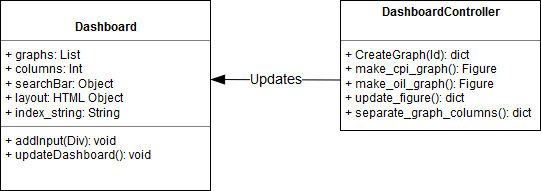


Class Diagram:



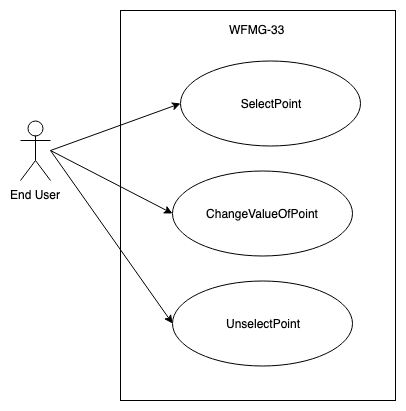
**WFMG-25 Fix column layout**

**Class Diagram:**

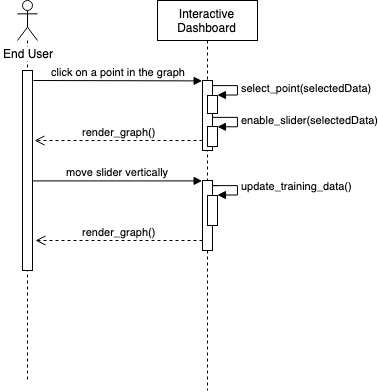


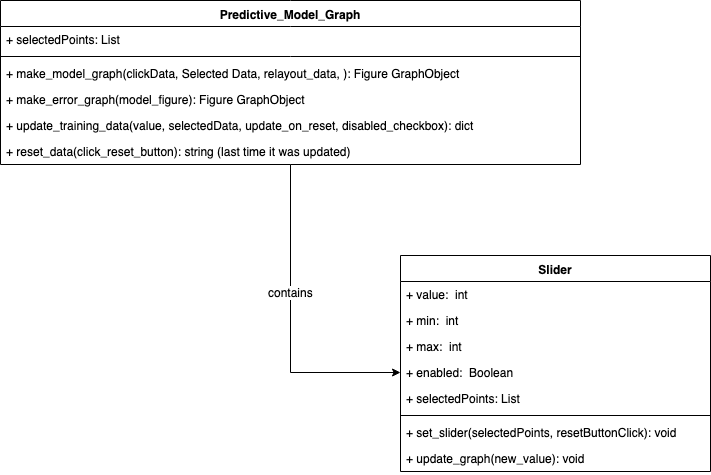
**WFMG-33 Movable Points**

Use Case Diagram:

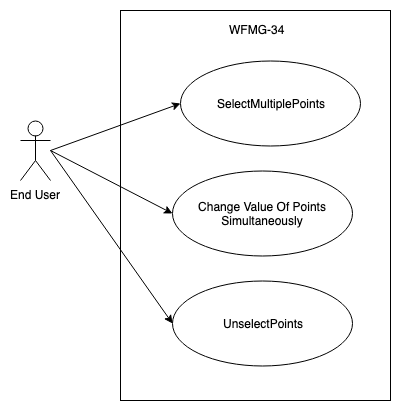


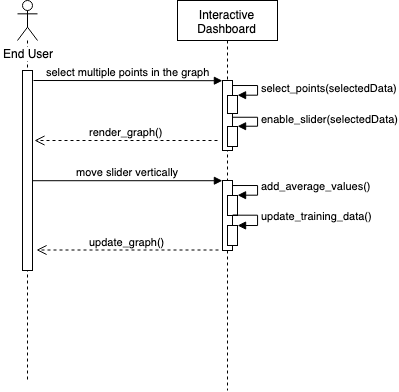
Sequence Diagram:

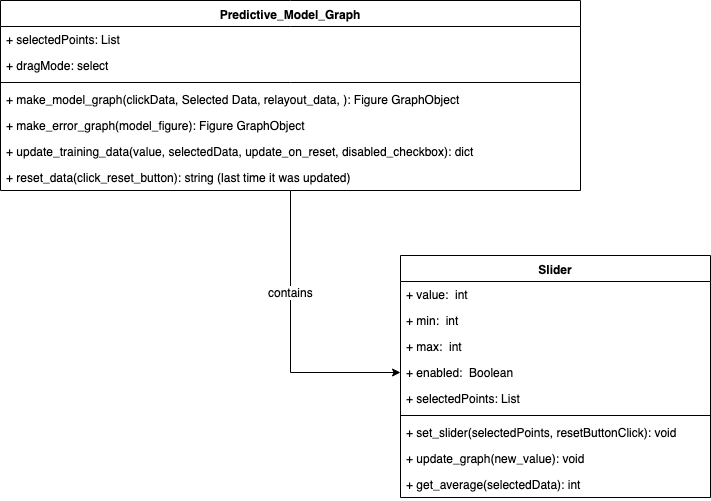


Class Diagram:

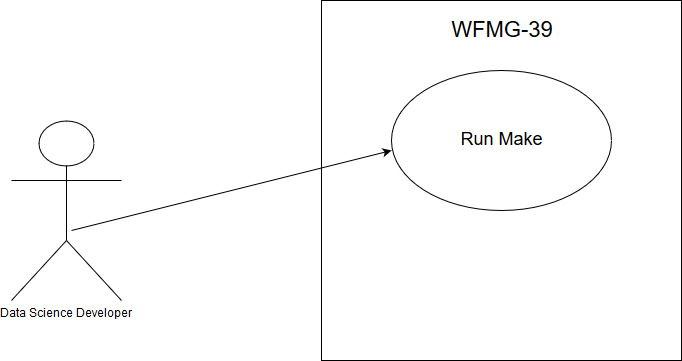
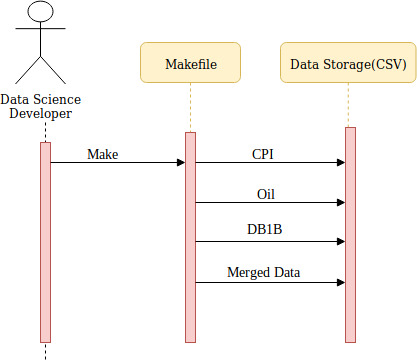
**WFMG-34 Drag multiple data points**

Use Case Diagram:

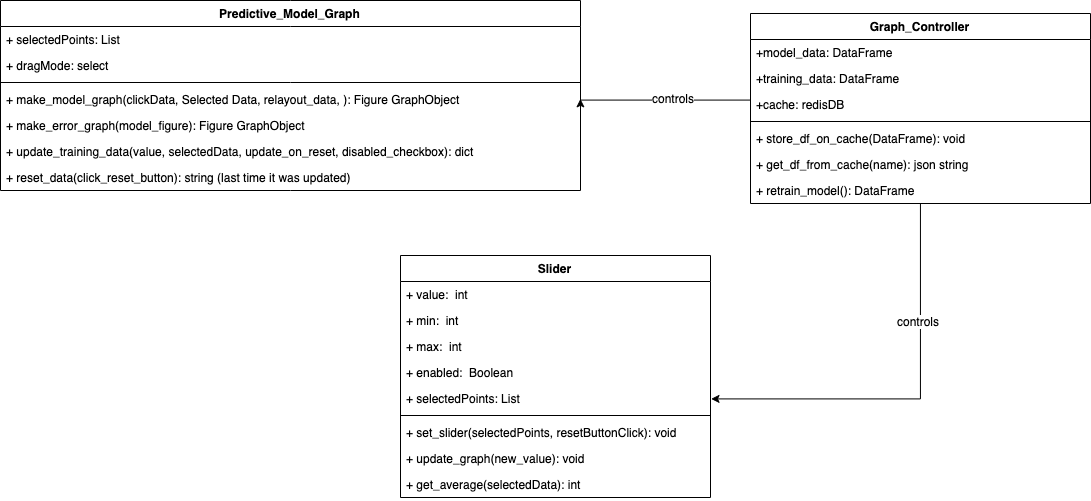
Sequence Diagram:

Class Diagram:

**WFMG-39 Data processing/preprocessing**

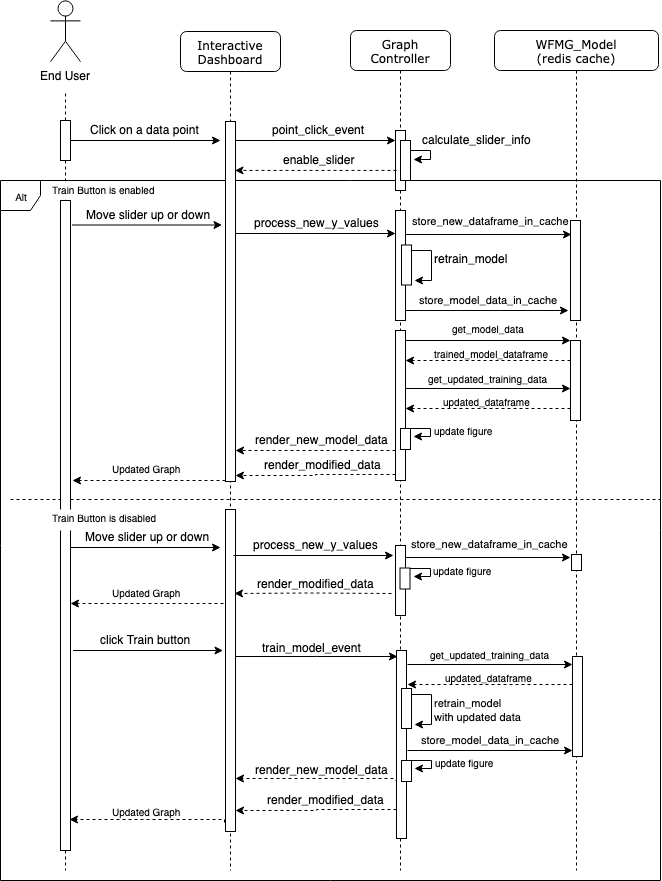
**Us****e Case Diagram:**

**Sequence Diagram:**

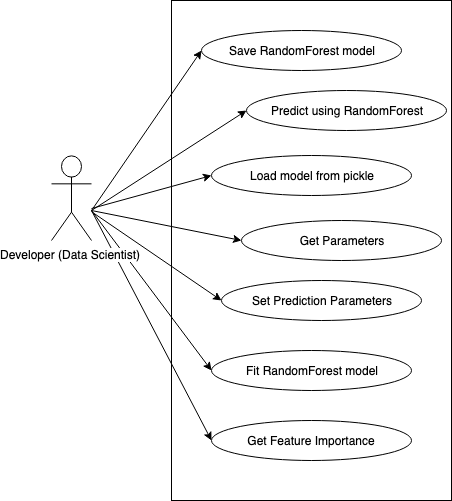
**WFMG-40 Retrain Model on Point Movement Even****ts**

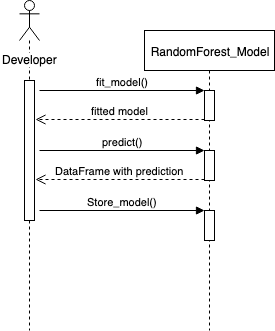
Use Case Diagram:

Class Diagram:

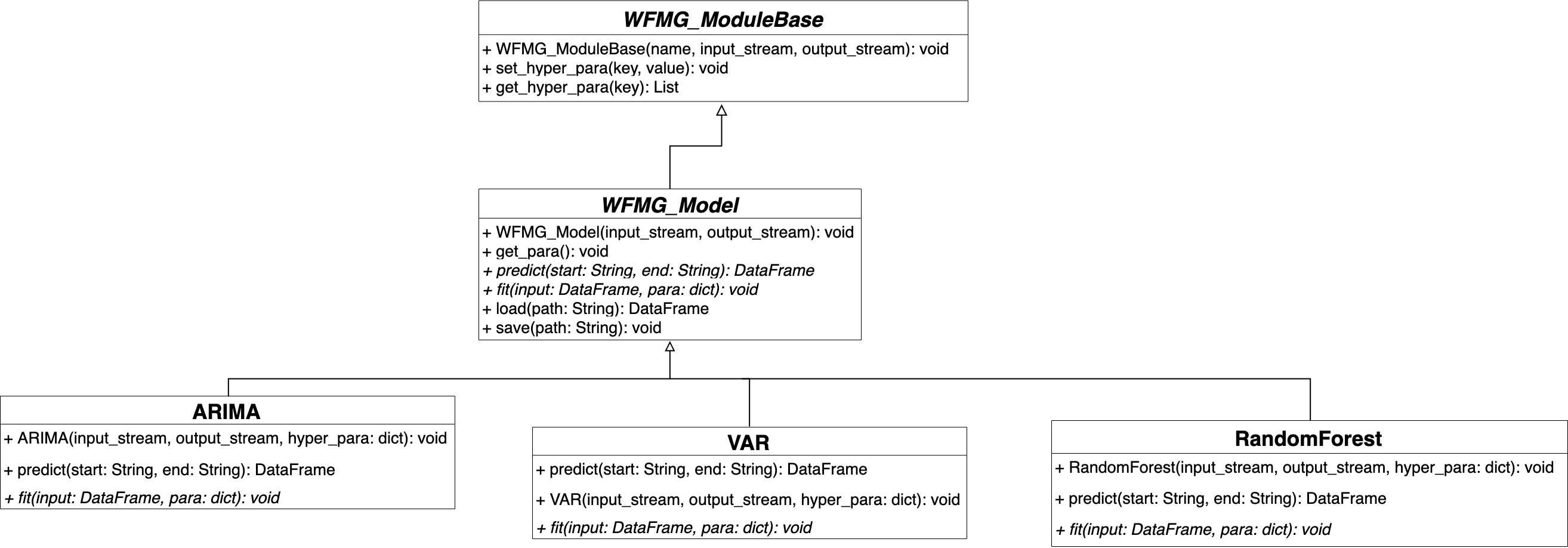
Sequence Diagram:

**WFMG-41 Random Forest Model**

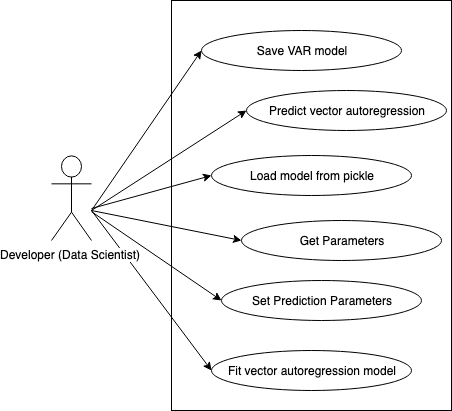
Use Case Diagram:

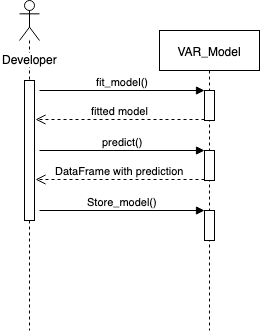
Sequence Diagram: 

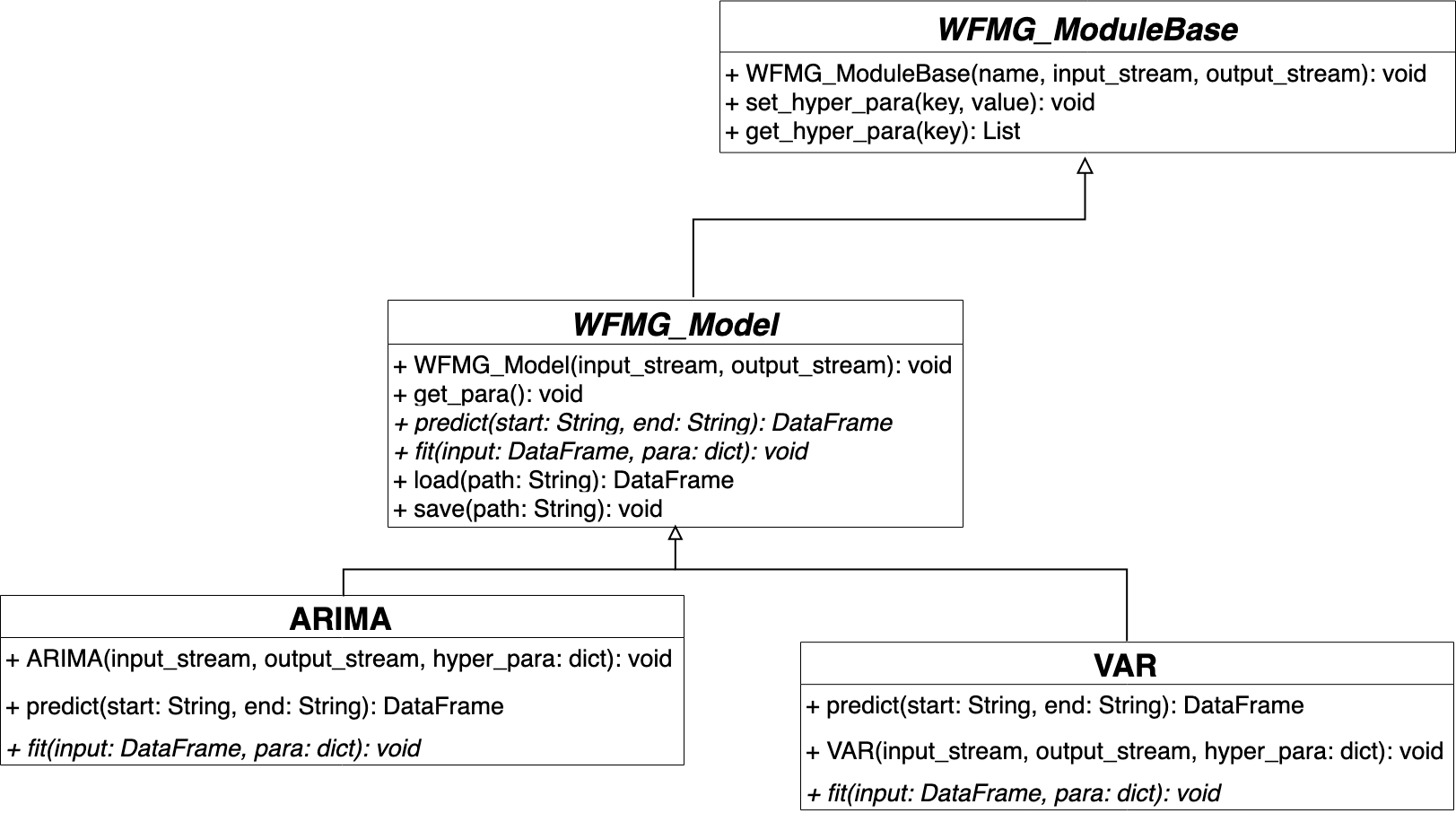
Class Diagram:



**WFMG-43 Multivariate Time Series Analysis Model**

Use Case Diagram:

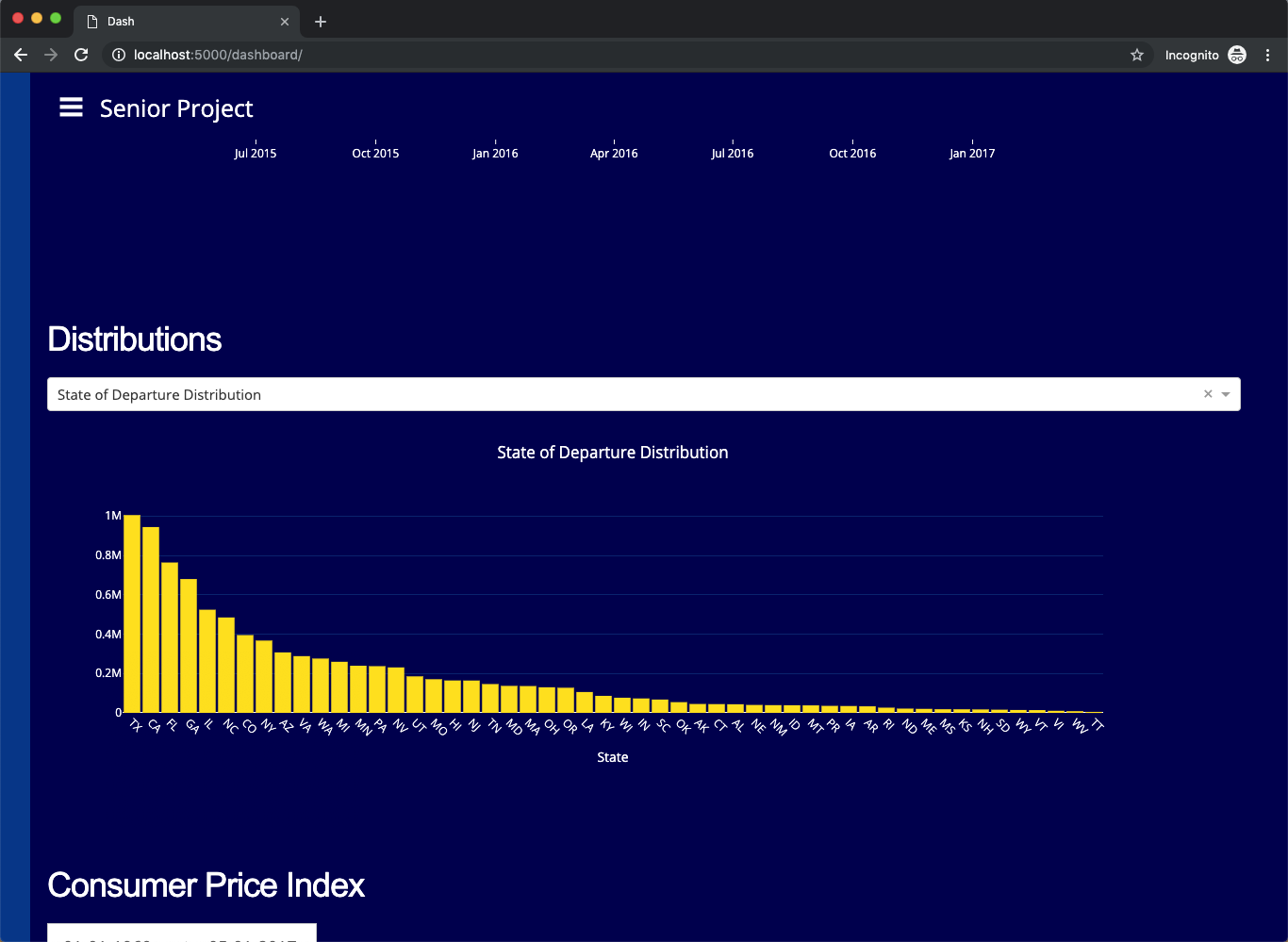
Sequence Diagram: 

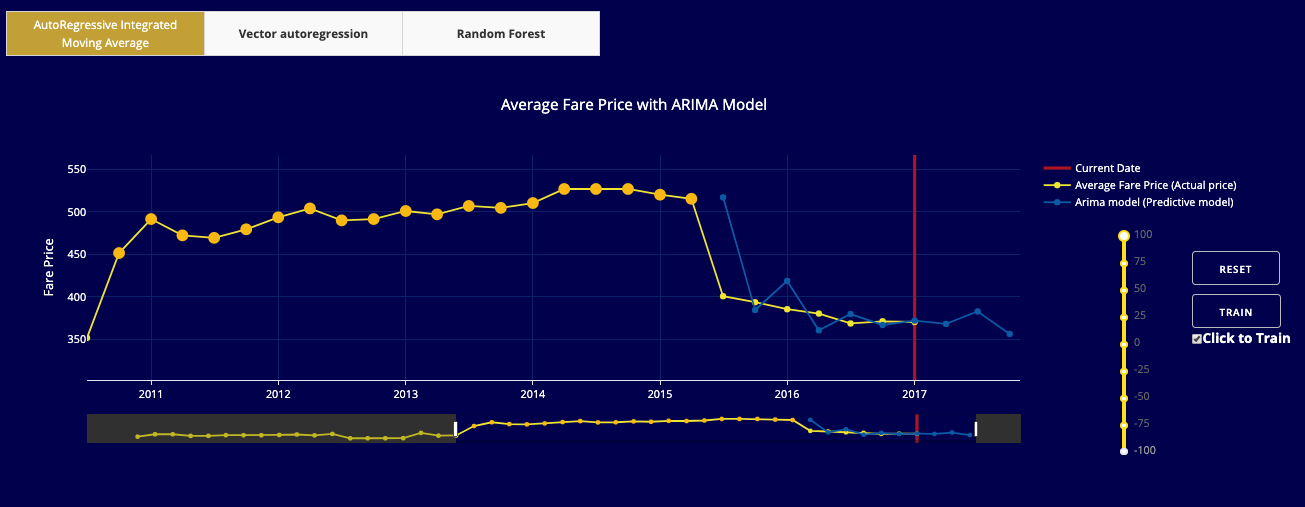
Class Diagram,:

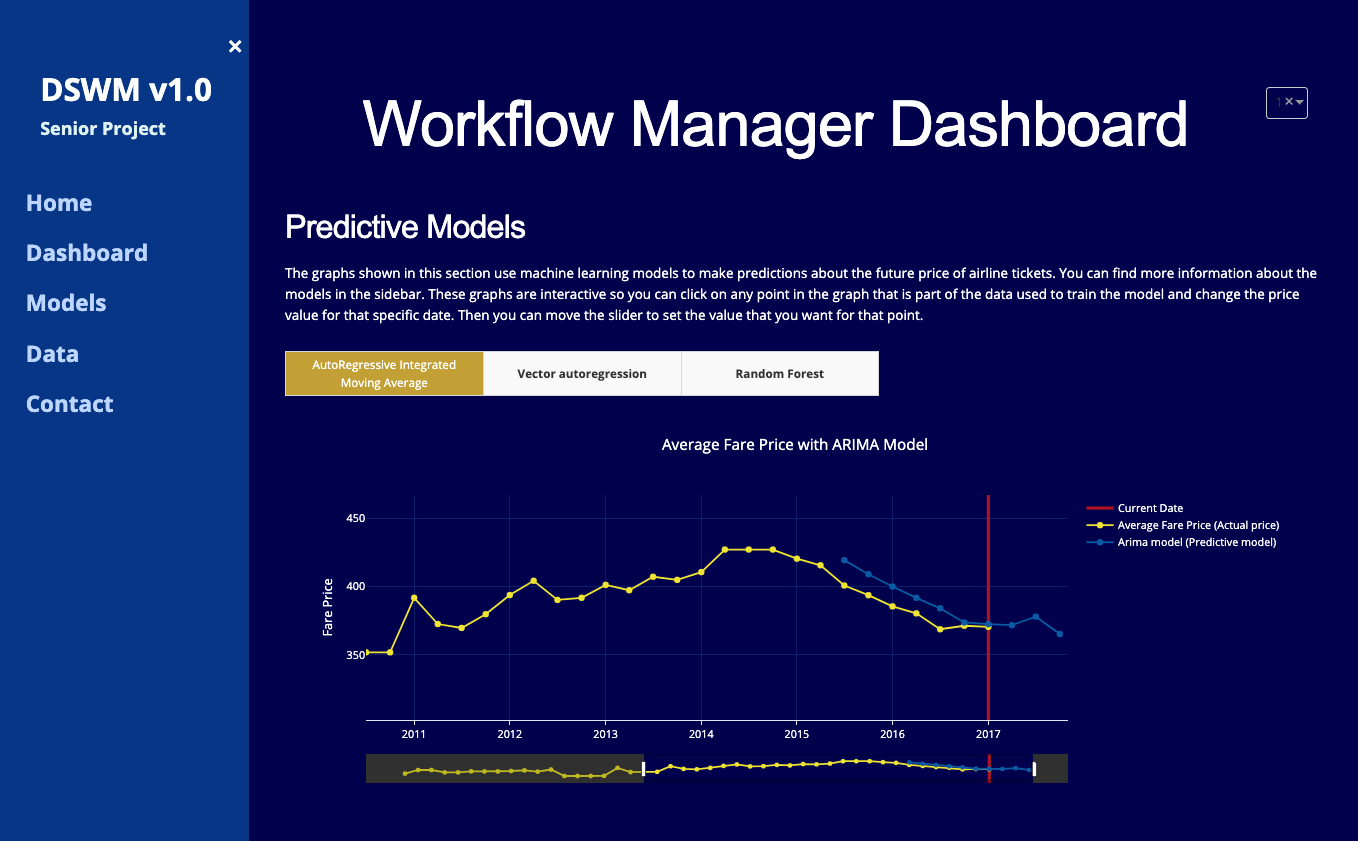
## 

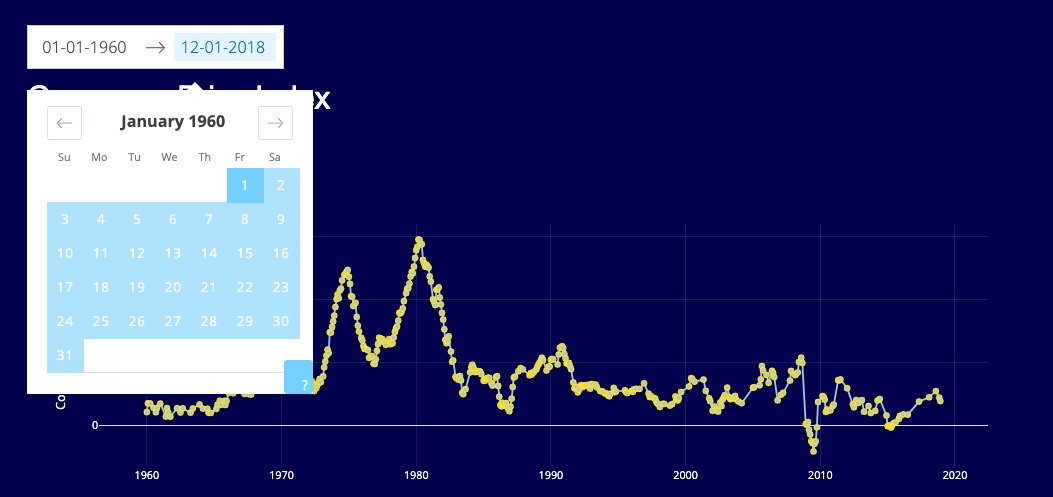
## Appendix B - User Interface Design

We used Dash to create an interactive dashboard where the user can move points in the graph or select which graph he wants to see in the dashboard using a dropdown menu.

The User can select from a collection of graph such as the coupon distribution, the state of departure distribution, market share distribution, and the fare class distribution (the last one refers to the class of the seats, economy, business etc)

In this image we can see how the machine learning model react to the user input and adapts to the new training data.

The user can also use the side navigation bar to navigate through the different pages of the website.

The user can also select the range using the range selector which was implemented for the CPI graph and the Crude Oil Price graph. Here we have a screenshot showing this functionality.

## Appendix C - Sprint Review Reports

Sprint 1 Review

20190204 - Sprint 1 Review Meeting Minutes

Attendees:

• Dev Team: Serge Metellus, Tomas Ortega

• Product Owners: Steven Luis, Miguel Alonso Jr.

• Team of graduate students

Start time: 10:00 AM

End time: 11:00 AM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

• User Story WFMG-12 - Display data as a line plot

• User Story WFMG-13 - User Interface

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

• User Story WFMG-2 - Display Distribution of data

• User Story WFMG-3 - Filter data by time.

• User Story WFMG-4 - Organize charts into columns to identify trends

Sprint 2 Review

 20190218 - Sprint 2 Review Review Meeting Minutes

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* Team of graduate students

Start time: 11:00 AM

End time: 12:15 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story   WFMG-19 - Fare price visualization
* User Story   WFMG-2 - Display Distribution of data
* User Story   WFMG-3 - Filter data by time.
* User Story   WFMG-4 - Organize charts into columns to identify trends

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* N/A  The team was able to complete all the user stories this sprint.

Sprint 3 Review

20190304 - Sprint 3 Review Meeting Minutes

Attendees:

* + Dev Team: Serge Metellus, Tomas Ortega
  + Product Owners: Steven Luis, Miguel Alonso Jr.
  + Team of graduate students

Start time: 2:00 PM

End time: 3:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* User Story   WFMG-25 - Fix Column Layout
* User Story   WFMG-22 -Adjust Prediction parameters
* User Story   WFMG-23 - Model Wrapper

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story   WFMG-24 - Visualization wrapper

Sprint 4 Review

 20190318 - Sprint 4 Review Review Meeting Minutes

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* Team of graduate students

Start time: 1:00 PM

End time: 2:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story   WFMG-24 - Visualization wrapper
* User Story   WFMG-33 - Movable points

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* User Story   WFMG-32 - Description of data files
* User Story   WFMG-34 - Drag Multiple Data Points

Sprint 5 Review

 20190401 - Sprint 5 Review Review Meeting Minutes

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* Team of graduate students

Start time: 1:00 PM

End time: 2:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story   WFMG-34 - Drag Multiple Data Points
* User Story   WFMG-35 - Reset Graph Button
* User Story   WFMG-39 - Data Processing/Preprocessing
* User Story   WFMG-40 - Retrain Model on Point Movement Events

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* N/A  The team was able to complete all the user stories this sprint.

Sprint 6 Review

 20190415 - Sprint 6 Review Review Meeting Minutes

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* Team of graduate students

Start time: 1:00 PM

End time: 2:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story   WFMG-41 - Random Forests Model
* User Story   WFMG-42 - Random Forest Visualization
* User Story   WFMG-43 - Multivariate Time Series Analysis Model
* User Story   WFMG-44 - Fix Layout and Styling Issues
* User Story   WFMG-45 - User Interface Improvements

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* N/A  The team was able to complete all the user stories this sprint.

Sprint 7 Review

 20190428 - Sprint 7 Review Review Meeting Minutes

Attendees:

* Dev Team: Serge Metellus, Tomas Ortega
* Product Owners: Steven Luis, Miguel Alonso Jr.
* Team of graduate students

Start time: 1:00 PM

End time: 2:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

* User Story   WFMG-46 - Presentation
* User Story   WFMG-47 - Document
* User Story   WFMG-48 - Prepare the Workflow Manager for presentation demo
* User Story   WFMG-49 - Prepare the final deliverable

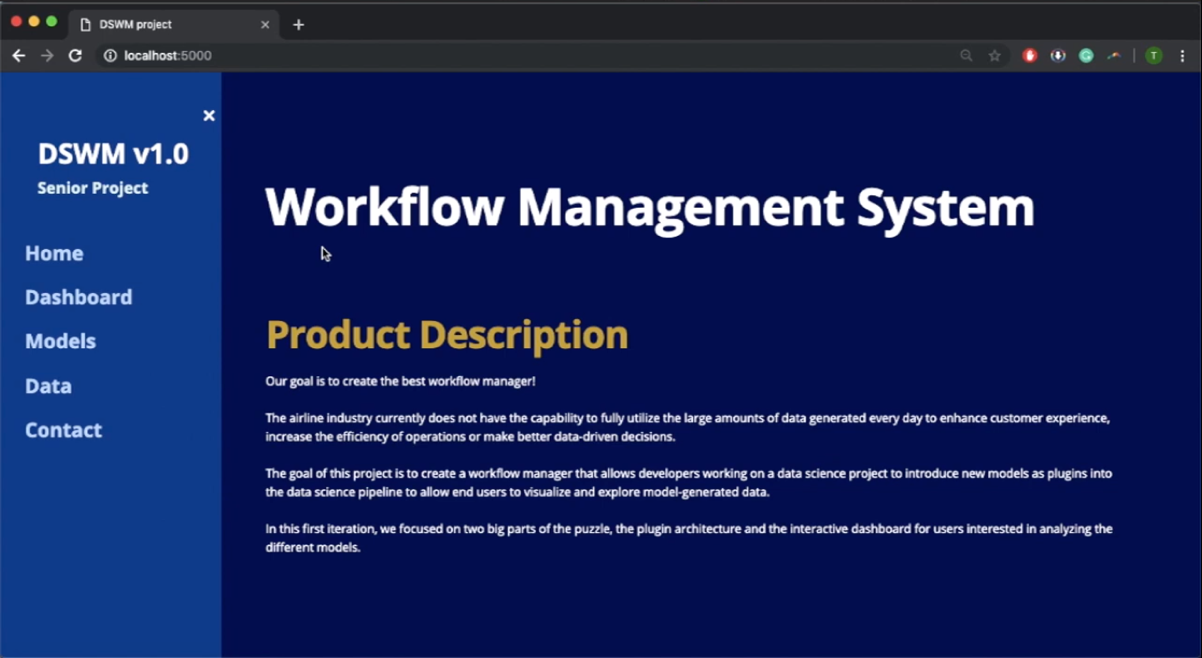
The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

* N/A  The team was able to complete all the user stories this sprint.

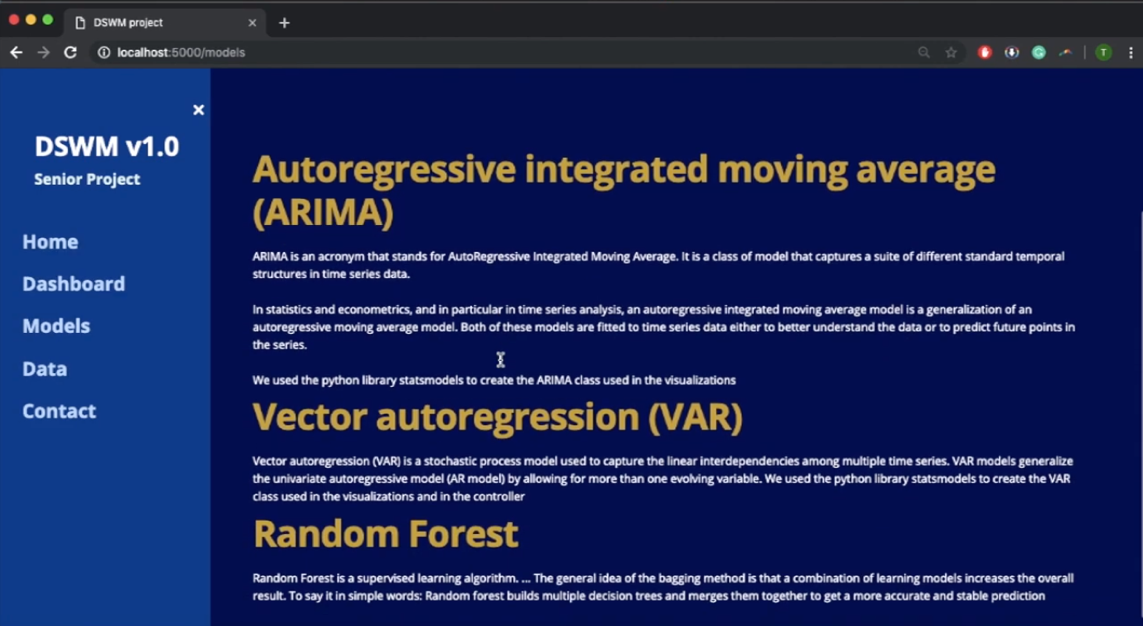
## Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

User Manual for End Users

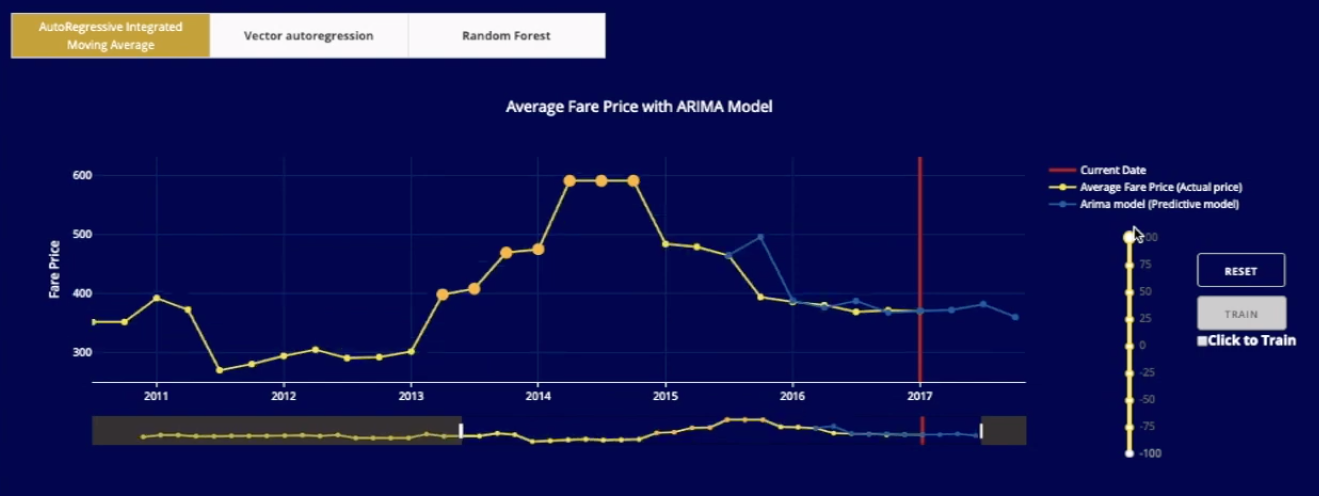
When the page loads, the user is going to be able to use the side navigation bar to move through the website.



In the main page, the user can read about the goals of the project and the new updates that will be coming soon. In the navigation bar we can see that there are several links that will direct the user to information about the different models and about the data sources that we used for training the model and for the rest of the visualizations.

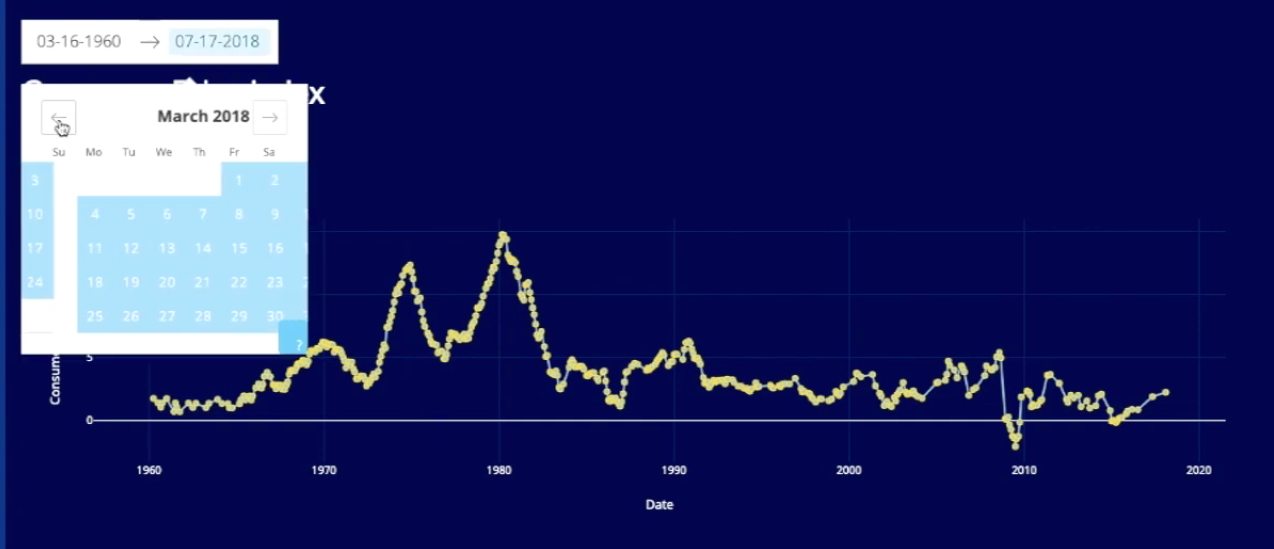


The user can go to the dashboard and click on the points that are part of the training data to move them around using the slide as shown in the picture below.



After the user has achieved a configuration that he believes is the one he was looking for, the user can click on the train button to train the machine learning model. If the train button is disabled, then every time the user moves a point up or down, the model will be train with the changes he made.

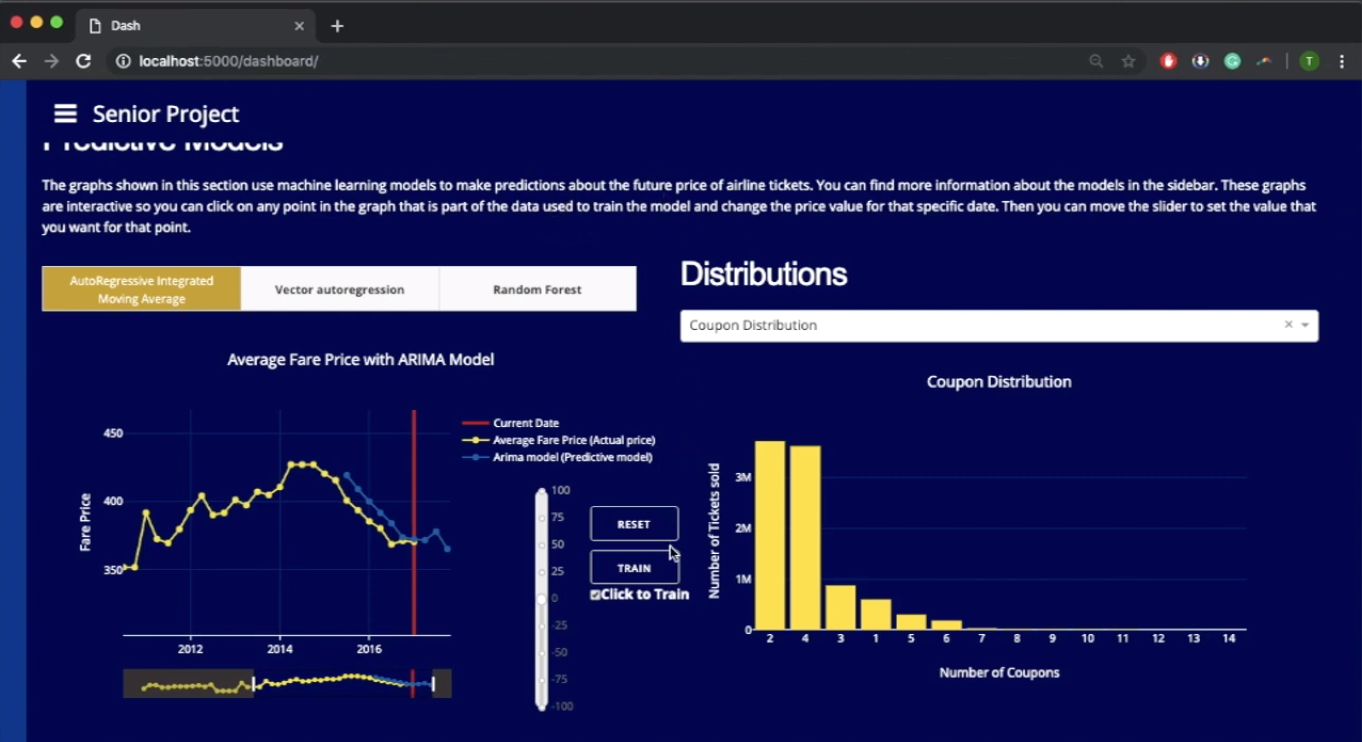
The user can also interact with the other graphs in the dashboard by picking a new range for the visualizations. Using the range selector, the user can go through the months and find exactly the moment he is looking for. Then the graph will update to show only the points that the user wants to see.



The user can also change the amount of columns that will be visible in the dashboard. The default value for this component is 1, but the user can choose to view up to 5 graphs in a single row. In the picture below we can see the component for changing the number of columns.

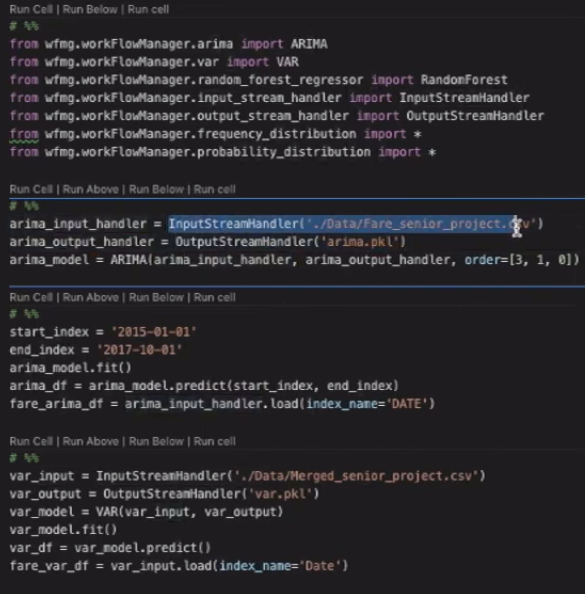


If the user decides that he prefers to view the dashboard 2 graphs at a time, the application will look like this.



User Manual for Data Science Developers

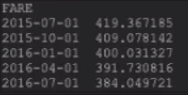
The workflow manager application allows developers to implement new models using the classes provided by the wfmg module to quickly and painlessly visualize the results in the web application.



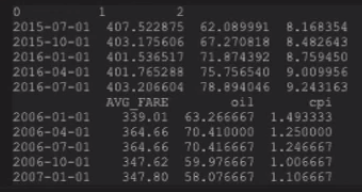
In this picture we can see that we must import the libraries for the code to work. The user then must initialize objects for the input stream handler and output stream handler for each model object that he wants to create so that the model can get access to the training data. Notice that we need to create a range for the dates that the model will predict. If we run this code, we are going to get a data frame with the predictions from the ARIMA model from the start date that we defined to the end date.

In the picture we also see that the VAR model is also available to developers. The VAR model is interesting because it uses more than one Time Series to predict the future prices.

The regular result for a univariate model such as ARIMA is shown below.

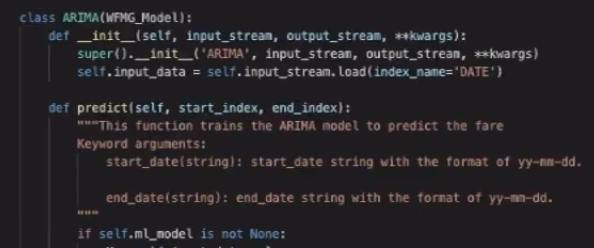


The VAR model on the other hand is capable of using several data frames to make it’s predictions like we can see in this picture.



The developers also have access to several scripts that take care of all the preprocessing of the csv files that we used such as changing the index to a dateindex, changing the name of the columns etc. Using the wfgm module, developers can also extend the functionality of our models by creating classes that inherit from the WFMG\_model class or the WFMG\_proc class.

In the project repository there are 3 exameples, the ARIMA model, the VAR model and the Random Forest Model. Here we decided to show part of the code of the ARIMA model so that you can get an Idea of how to create subclasses that can be used by the visualization module.



Installation and Maintenance.

First, you must clone the repo from the bitbucket repository or from the AirLab github repo. Once the repository is stored locally, you can start to download the dependencies. All dependencies are stored in the the requrements.txt file. To install all of them at the same time you must use pip. I suggest that you use pip3 just in case you still have the old version of pip laying around somewhere. For the development of this project we used python version 3.6.

Before you install the requirements, I would suggest that you create a virtual environment so that any new dependency that gets added to the project can be stored in the file requiremets.txt.

It is also good practice to keep the python interpreter that you are using constant throughout the development process.

You can create a virtual environment by typing this command.

python3 –m venv venv

The second venv is just the name of the folder that the utility venv is going to create in our current directory. You can change it to anything you want.

After the folder is created, you can activate the virtual environment by typing this command.

source venv/bin/activate

Notice that if you have a folder with a different name, then you should replace the venv in that last command.

Now, you are ready to install the dependencies and start coding. You can install the requirements by typing the following command in any bash terminal.

pip3 install -r requirements.txt

Now you are ready to start coding!

For more information on how to maintain the code and how to install everything that you will need, please refer to the video “FIU SCIS 2019Spring WFMG InstallMaintenanceGuide”

# 

**Shortcomings and Wishlist**

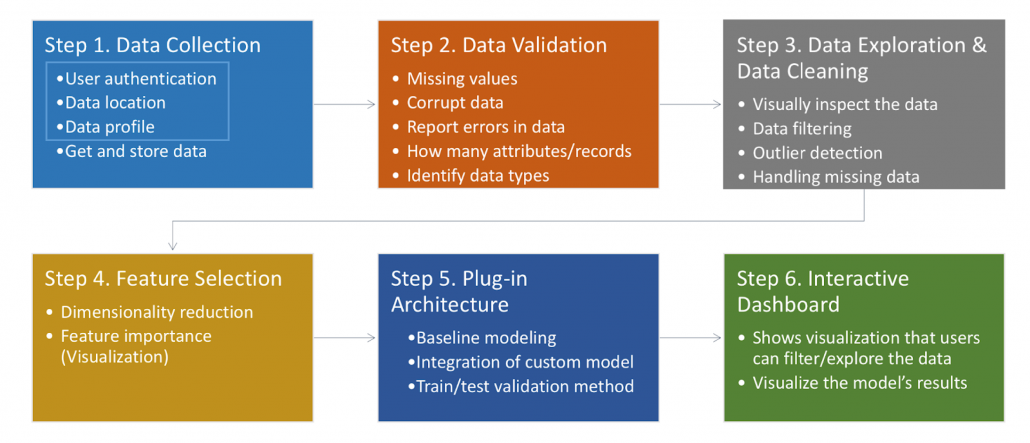
The main shortcoming in this first iteration was that we did not get to create a Database to store the data and models, in order to process the data even more efficiently and also to process new data automatically.

The main goal of the project was to implement the last few steps of the data science pipeline

as shown below and on the FIU AIRlab website. We mainly focused on the plug-in architecture and the

interactive dashboard. Future iterations should implement the remaining steps of the pipeline in

reverse order ending with Data collection.



# References

* The Plotly Reference Manual: <https://plot.ly/python/reference/>
* The Dash User Guide with tutorials and many examples: <https://dash.plot.ly/>
* Flask documentation: <http://flask.pocoo.org/docs/1.0/>