IOWA STATE UNIVERSITY

Department of Electrical and Computer Engineering

ECpE Senior Design Poster Session
December 6, 2018
sddec18-13

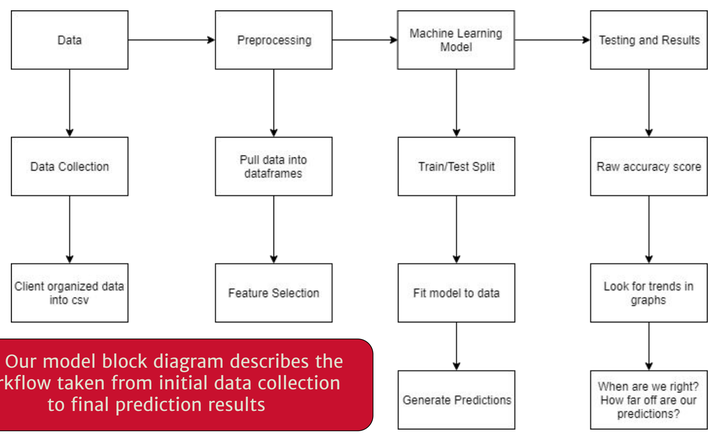
Principal

Caleb Utesch, Carter Scheve, Jack Murphy, Alex Mortimer, Nathan Hanson, and Samuel Howard with Dr. Chinmay Hegde

Asset Management - Financial Factor Discovery - "Value"

Motivation

- Investment analysis at Principal Financial Group currently relies on human calculation, using a variety of models and inputs
- Various steps of the statistical analysis process can be automated, which would remove most of the potential for human error, and reduce overhead costs by making accurate statistical modeling and prediction more accessible
- Our solution makes use of our extensive background in computational sciences to implement a software approach to multi-factor statistical analysis and risk assessment
- We have developed a Pipeline to combine and streamline our research with data aggregation methods, feature selection techniques, and machine learning models



Our model block diagram describes the workflow taken from initial data collection to final prediction results

Technical Details

Languages/Libraries
- Python 3.6
- Anaconda
- Scikit-learn, Numpy, Pandas

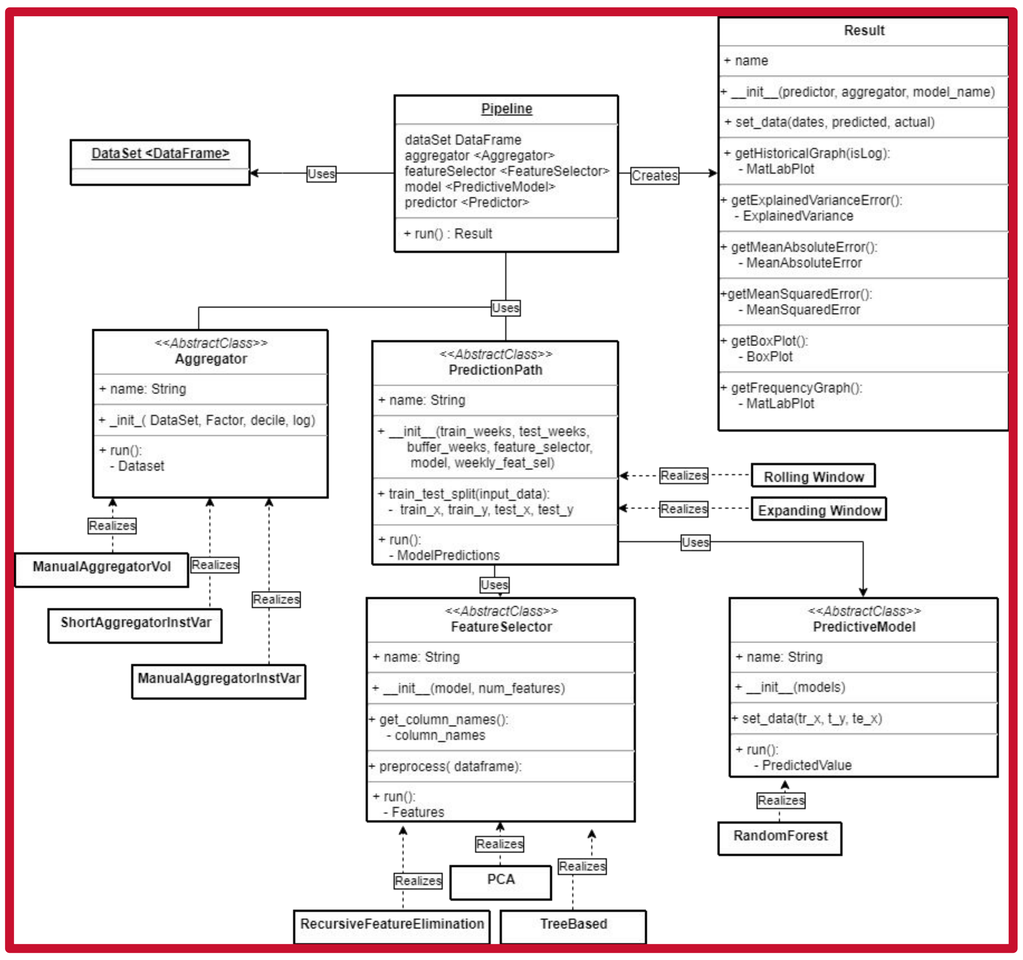
Development Tools/Environments
- Jupyter Notebooks
- Amazon AWS
- Pycharm IDE
- PostgreSQL Database Engine

Design Requirements

Non-functional
- All code will be developed in Python
  - As the most common language used for machine learning purposes, this was an easy decision for the language to use.
- Utilize machine learning techniques to generate predictions from stock data
  - This is a common strategy for obtaining predictions of stock market factor performance

Functional
- Models report processing time and accuracy
- Results are displayed in a human-readable format
- Models only use data from a certain time period to predict future behavior
  - Example: The model does not use stock market data from 2005 to make predictions on the stock market for 2002
- Summary of models gives concrete statistics for performance of each individual model, along with a comparison of each and a recommendation for which to use in similar future tasks.

Operating Environment
- Python version 3.6+, Scikit-learn, Pandas, and Numpy.
- Amazon Web Services: EC2 Instance for distributed computations hosted in the cloud



PIPELINE CLASS DIAGRAM

Pipeline Components

Pipeline: A controller which provides an interface for easy interaction with all components in the prediction pipeline sequence. The pipeline manages communication inputs and outputs between components.

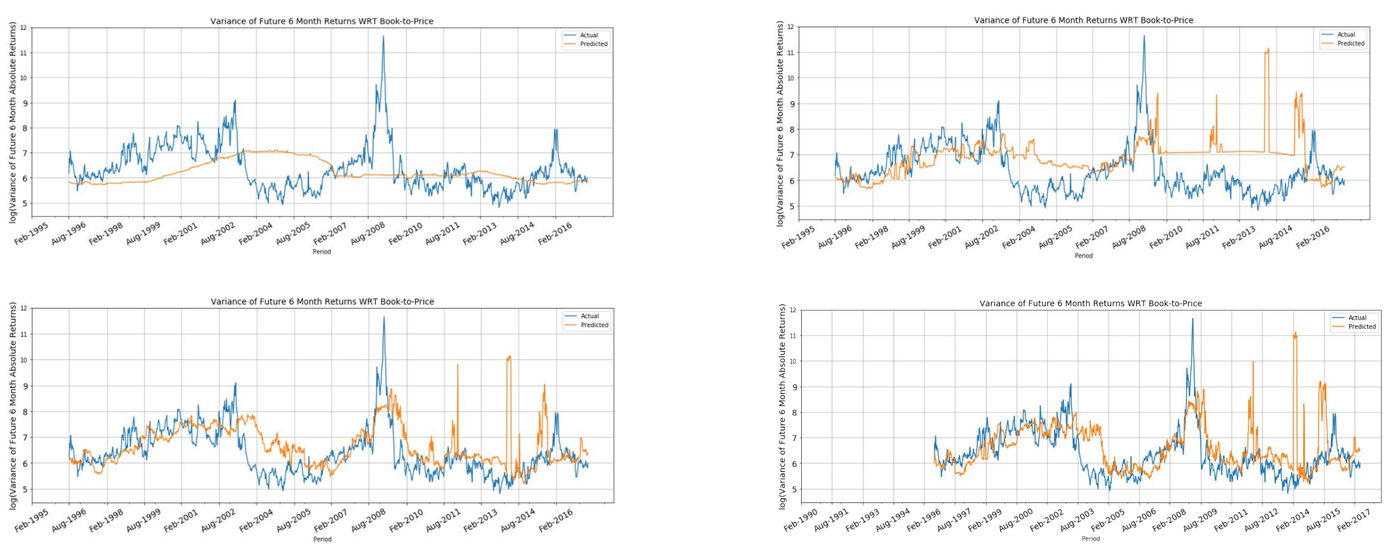
Prediction Path: This class defines the process in which the Predictive Model instance will be tested. The instantiation defines input data sizes for each window. The model is then tested in the each window. Results are then returned.

Aggregator: Encapsulates the functionality for aggregating predictor and feature columns. Each subclass contains their own method to aggregate a specific predictor. The parent abstract class implements the basic feature aggregation.

Predictive Model: Used to generate predictions for a specified factor. A subclass would utilize a machine learning model from either Scikit-learn or from a custom implementation to predict one value, using the input data.

Feature Selector: Used to implement one of the feature selection techniques developed. Recursive Feature Elimination, Tree Based, and PCA all provide a unique subset of features to be fed into a model.

PipelineResult: Used to store the results from the predictive model. Additionally contains functionality to analyze and display information about the results obtained in a human-readable format.



Top Left: Gradient boosting model with loss calculation of least absolute deviation with learning rate 0.001.
Top Right: Gradient boosting model with loss calculation of least squares and a learning rate of 0.01
Bottom Left: Random forest model with number of estimators set to 100
Bottom Right: Extra trees model with number of estimators set to 100

Constraints

- Use only past data to predict future performance
- Utilize only first decile of stock-level data
- Develop entire project on client's EC2 instance

Testing

Testing Environment
- EC2 Amazon Web Service instance
- Jupyter Notebooks
- Anaconda

Regression Testing
- Manual and informal
- Isolated components and tested with custom inputs
- Ran against multiple inputs to ensure correct output

Integration Testing
- Use Pipeline to test integration of components
- Ensure subclasses could be used interchangeably

Acknowledgements: Dr. Joseph Zambreno, Dr. Thomas Daniels