Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Магистерская программа Финансовые технологии и анализ данных

КУРСОВАЯ РАБОТА

На тему «**Разработка инструмента для мониторинга и анализа состояния портфеля кредитных требований банка**»

Студент группы № мФТиАД
Борисенко Глеб Витальевич

Руководитель КР, PhD
Академический руководитель
Масютин Алексей Александрович

Консультант
Руководитель проектов
Щербаков Игорь Андреевич

Москва, 2020

Реферат

Отчет 16 с., 6 рис., 6 источн.

Цель работы – спроектировать систему конструирования дашборда и разработать с её помощью инструмент для мониторинга и анализа состояния портфеля кредитных требований банка.

В процессе работы были проанализированы существующие решения данной проблемы. Была изучена выбранная платформа. Также проводилась модификация плагина и разработка сервера для системы. Затем был сконструирован дашборд основных показателей портфеля.

В результате работы были представлены работающий проект, позволяющий создать дашборд с подключением к внешнему файлу, и дашборд, визуализирующий состояние портфеля кредитных требований банка.

Ключевые слова: GRAFANA, ДАШБОРД, МОНИТОРИНГ

Содержание

Введение

Одним из этапов исследования данных на сегодняшний день является визуальный анализ. Он проводится с помощью различного рода диаграмм и графиков. Такой анализ позволяет выявлять закономерности, находить аномалии, и многое другое. Также после введения модели в эксплуатацию одним из важнейших для поддержания ее работы процессов является мониторинг показателей.

В качестве инструмента для визуального анализа и мониторинга показателей частно используется язык программирования с соответствующими библиотеками, например, язык Python с пакетом seaborn. Очевидным преимуществом такого подхода является высокая гибкость, но расплачиваться за это приходится достаточно большим количеством времени, необходимого для создания информативного графика и внесения даже минорных изменений.

Помимо описанного выше подхода, можно использовать специализированное ПО. Оно позволяет снизить порог вхождения благодаря отсутствию необходимости в знании программного API, упростить создание и изменение диаграмм с помощью графического интерфейса, а также решает большое количество неявных проблем. Но в все эти преимущества идут рука об руку с весомыми ограничениями. Например, поддержка только определенного формата, всего несколько возможных типов графиков, дополнительные ограничения, вызванные интерфейсом.

В моей работе предлагается совместить оба подхода для создания программного решения, обладающего изложенными выше плюсами и лишенного если не всех, то хотя бы нескольких недостатков, с целью создать интерактивную панель с диаграммами, позволяющую обеспечить простой и удобный мониторинг определенных показателей кредитного портфеля.

1 Постановка задачи

Целью работы является с помощью разработанного программного решения сконструировать интерактивный дашборд – панель с диаграммами. Ниже представлены требования как для ПО, так и для дашборда.

В качестве источника данных изначально должен поддерживаться формат csv как наиболее распространенный, но дальнейшее добавление новых источников должно быть простым и понятным. Должна существовать возможность разместить панель диаграмм на сервере, к которому можно подключаться с удаленного клиента.

Все диаграммы дашборда показателей должны поддерживать фильтрацию по дате. Диаграммы должны отображать следующие показатели:

- Численность договоров – количество договоров на дату с учётом всех фильтров в виде линейного графика
- Численность клиентов – количество клиентов на дату с учётом всех фильтров в виде линейного графика
- Default rate – доля тех, кто выйдет в дефолт в течение года, от тех, кто не в дефолте на дату оценки в виде линейного графика
- Уровень модельных оценок PD – среднее значение по полю вероятности дефолта с учётом всех фильтров. Исторический период – в виде линейного графика, а также барометр для самой последней даты.
- Распределение по рейтингам - столбчатая диаграмма
- Качество модели - показатель Gini.

2 Разработка платформы конструирования дашборда

2.1 Поиск и сравнение существующих решений

Одним из ключевых атрибутов системы должна быть открытость, т.е. это должно быть open source решение. Это позволит добиться той же гибкости, что дает использование только языка программирования. Из-за этого сразу отпадают лучшие приложения для исследования и визуализации данных: Qlik и Tableau.

Среди open source платформ наиболее популярная и удобная это Grafana. Она позволяет создавать интерактивные дашборды с большим набором возможных типов диаграмм и источников данных, а плагины сообщества дополнительно расширяют функционал. Главным недостатком с точки зрения данной работы является отсутствие поддержки в качестве источника данных внешнего файл – все предлагаемые источники — это различные базы данных. Но Grafana обладает достаточно понятным API, что позволяет самому создать плагин. Именно это стало основополагающим фактором в выборе платформы конструирования дашборда.

2.2 Обзор Grafana

Grafana позволяет создавать интерактивные дашборды. Пример такого дашборда [1] представлен на рисунке 1. Каждый дашборда состоит из набора панелей, каждая из которых и является диаграммой. Есть несколько видов диаграмм [2], которые платформа поддерживает прямо из коробки, но огромное количество типов доступны в качестве плагинов, которые просто ставятся через интерфейс командной строки либо обычным перемещением папок. Также среди визуализаций есть различные варианты текстового представления обычных или критических данных, вроде таблицы, списка тревог или даже графической схемы.

Ниже представлен список типов графиков, которые поддерживаются Grafana прямо из коробки:

- Линейный график
- Гистограмма
- Столбчатая диаграмма
- Барометр
- Тепловая карта
- Круговая диаграмма

Рисунок 1- Пример дашборда Grafana

У каждой панели, в зависимости от выбранного типа визуализации, есть множество настроек, позволяющее создавать диаграммы на свой вкус и цвет. Например, для линейного графика можно регулировать формат линий, осей, легенду, добавлять границы для цветного отображения, временные регионы,

ссылки и другое. Каждая панель интерактивная, ее можно переместить, трансформировать и поделиться.

Grafana работает в основном с временными рядами, т.е. датафреймами с индексом по дате, поэтому в ней сразу есть фильтрация по дате. Есть возможность работы и с табличными данными без дат, но в таком случае визуализация доступна только для агрегаций над массивами. К счастью, эта проблема решается сторонними плагинами, например, плагином Plotly [3].

Для диаграмм нужны данные, по которым их надо рисовать, а данные Grafana получает путем отправки запроса к источнику данных. Список поддерживаемых источников достаточно обширен, каждый из них представлен в виде плагина. У каждого ресурса свой формат запроса, и в большинстве это запрос на языке SQL, т.к. это самое большинство – вариации баз данных. Из обнаруженных минусов – на одной панели не может быть графиков, полученных с помощью разных источников, но при этом возможно создавать несколько запросов, таким образом отображая сразу несколько массивов. Помимо этого, платформа позволяет настроить обновление панелей путем посылки повторного запроса с заданной частотой.

Также Grafana позволяет добавлять к графикам аннотации, трансформировать полученные из запроса данные и использовать переменные нескольких типов.

Сама Grafana запускается как веб приложение, что позволяет развернуть ее один раз, настроить подключение, и просматривать и изменять любимые дашборды в режиме реального времени по сети. В добавок платформа позволяет настроить доступ и аутинтификацию.

Все описанные в этом параграфе характеристики представлены для открытого, т.е. “Open Source” решения. Также у компании есть “Enterprize” версия, более оптимизированная для командной работы, и возможность разместить свои дашборды в их облаке, но эти предложения не рассматриваются, т.к. они не актуальны для данной работы.

2.2 Архитектура программного решения

В работе в качестве источника для системы мониторинга состояния портфеля кредитный требований банка должны использоваться внешние файлы форматы csv, к тому же должен быть понятный интерфейс для подключения новых типов файлов. К сожалению, среди плагинов не нашлось какого-либо, способного обращаться к внешним файлам. Но в просторах сети был найден скрипт [4], работающий в связке с модифицированным JSON плагином и написанный на Python. Скрипт работал со старой структурой плагина, формат csv был жестко зафиксирован в коде, таким образом лишая возможности просто и быстро добавить поддержку новых типов, да и в принципе новых опций. Поэтому было решено взять за основу описанную связку, но значительно ее модифицировать.

Общая схема работы сервера Python и JSON плагина Grafana в связке представлена на рисунке 2.

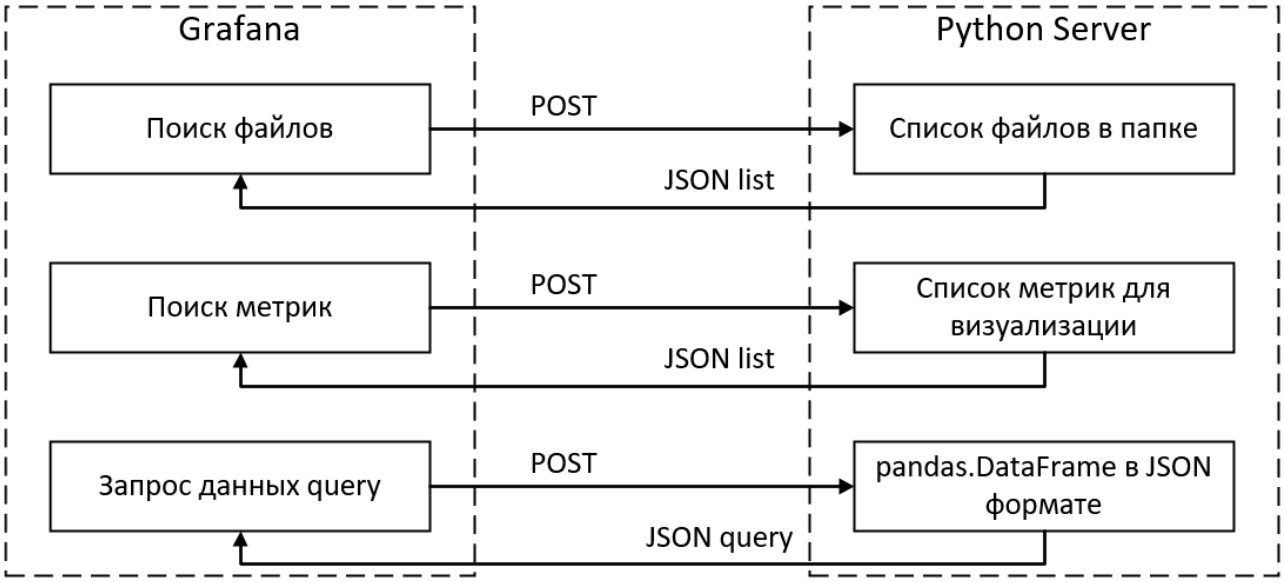

Рисунок 2- Схема работы решения

Плагин обеспечивает в поле запроса к данным четыре поля: тип ответа, файл, метрика и опциональное поле. Тип ответа – это либо временной ряд, либо таблица, он зашит в отдельное поле каждого запроса JSON к серверу.

Файл – собственно источник, откуда брать данные. При наведении на это поле Grafana предлагает список возможных выборов, которые получает с сервера ответом на соответствующий запрос. Метрика – это информация о том, что именно нужно визулизировать из выбранного файла, например, колонка таблицы. Здесь так же предлагается список возможных вариантов. Последнее поле является опциональным, оно может быть пустым либо заполнено текстом в JSON формате; в таком случае в JSON запрос добавляется поле “data”, которое и включает введённые данные. При изменении поля сразу же автоматически посылается запрос query.

Все запросы к серверу посылаются через POST метод. Для предоставления вариантов выбора плагину требуется ответ в виде списка в формате JSON. Для запроса query ответ должен быть в описанном в документации плагина формате, который зависит от выбранного типа ответа.

Таким образом, всё общение между плагином и сервером происходит через HTTP запросы с JSON содержанием. Для обеспечения работы данной связки необходимо модифицировать плагин и переписать скрипт сервера.

2.3 Модификация плагина

Проблема плагина[5] заключается в том, что он не позволяет выбрать файл, т.е. он не добавляет соответствующего поля в запрос query. Было решено слегка изменить плагин дабы добавить необходимый глазам прямоугольник.

Логика плагина написан на языке Typescript, а формат графического изображения полей представлен в виде HTML файла с соответствующими CSS-стилями, предоставляемыми Grafana.

Новое поле было добавлено такого же формата, что и поле выбора метрики, изменена только вызываемая функция на “ctrl.findSources”. Логика работы с ним так же была взята оттуда же, т.е. в конструктор класса было добавлено поле source и была реализована написанная выше функция в файле “query_ctrl.ts”, но она лишь вызывает функцию “sourceFindQuery” из файла

“datasource.ts”. Последняя функция полностью повторяет работу функции “metricFindQuery”. Для работы окна выбора метрики необходимо наличие названия выбранного файла в JSON запросе, поэтому оно также было добавлено в соответствующую структуру под названием “interpolated” дополнительным полем “source”. Итоговый формат запроса query представлен на рисунке 3.

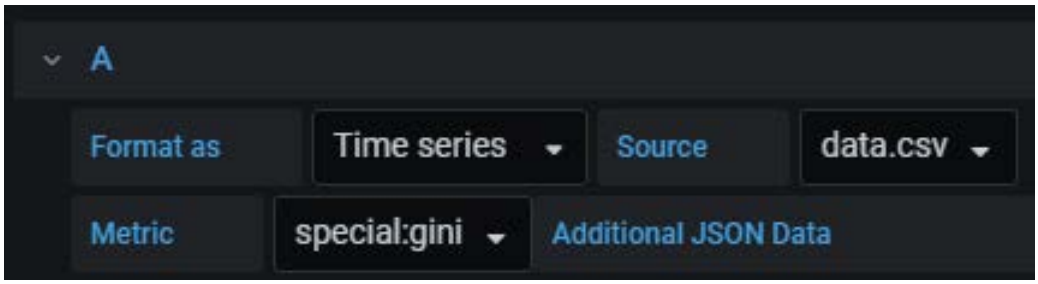

Рисунок 3- Форма запроса Query

2.4 Разработка серверной части

В Python сервере было необходимо переработать структуру найденного скрипта, чтобы появилась возможность улучшать проект в будущем и добавлять новые типы файлов.

В общем виде архитектура сервера представлена на рисунке 4.

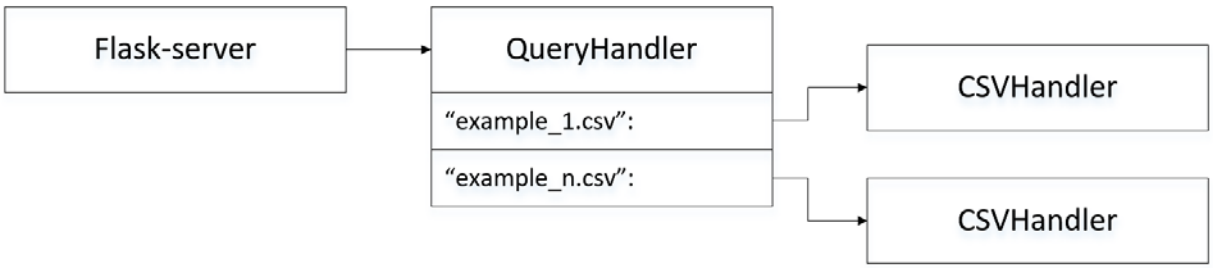

Рисунок 4- Структурная схема сервера

Сервер обращается к классу обработчика запросов QueryHandler, который должен поддерживать следующий интерфейс:

```
def get_sources(self, d_type, folder)
def get_metrics(self, d_type, source, target='')
def get_data(self, request)
```

Эти функции используются сервером для получения списка файлов, списка метрик и данных в формате JSON класса Response пакета flask. Последнее легко обеспечивается посредством выполнения команды jsonify.

Помимо описанного выше интерфейса, я так же реализовал функцию обработки специальных метрик, необходимых в работе дальнейшем, которая просто делает агрегации над pandas.DataFrame. Также есть функции, просто преобразовывающие датафрейм в необходимый плагину формат.

Класс обработчика хранит все данные как словарь, где ключом выступаем имя файла, а значением – класс обработчика данного типа файла. Для последнего также разработан интерефейс, который представлен ниже:

```
def get_columns(self, target)
def get_data_by_column(self, column, options)
```

Функции предназначены для получения списка колонок, которые используется в окне выбора метрик и для получения данных по колонке в формате pandas.DataFrame. Для временных рядов индексом результата последней функции должна быть дата в микросекундах (unix timestamp).

Для добавления нового файла достаточно разработать новый класс обработчика файла, который будет реализовывать описанный интерфейс.

3 Конструирование дашборда

Используя инструментарий платформы Grafana, был сконструирован дашборд для мониторинга состояния портфеля кредитных требований банка. Для каждого показателя создавалась панель, в которой делался специальный запрос к файлу, и настраивался график. Пример такой панели для показателя Gini с цветным порогом для значений меньше нуля представлен на рисунке 5.

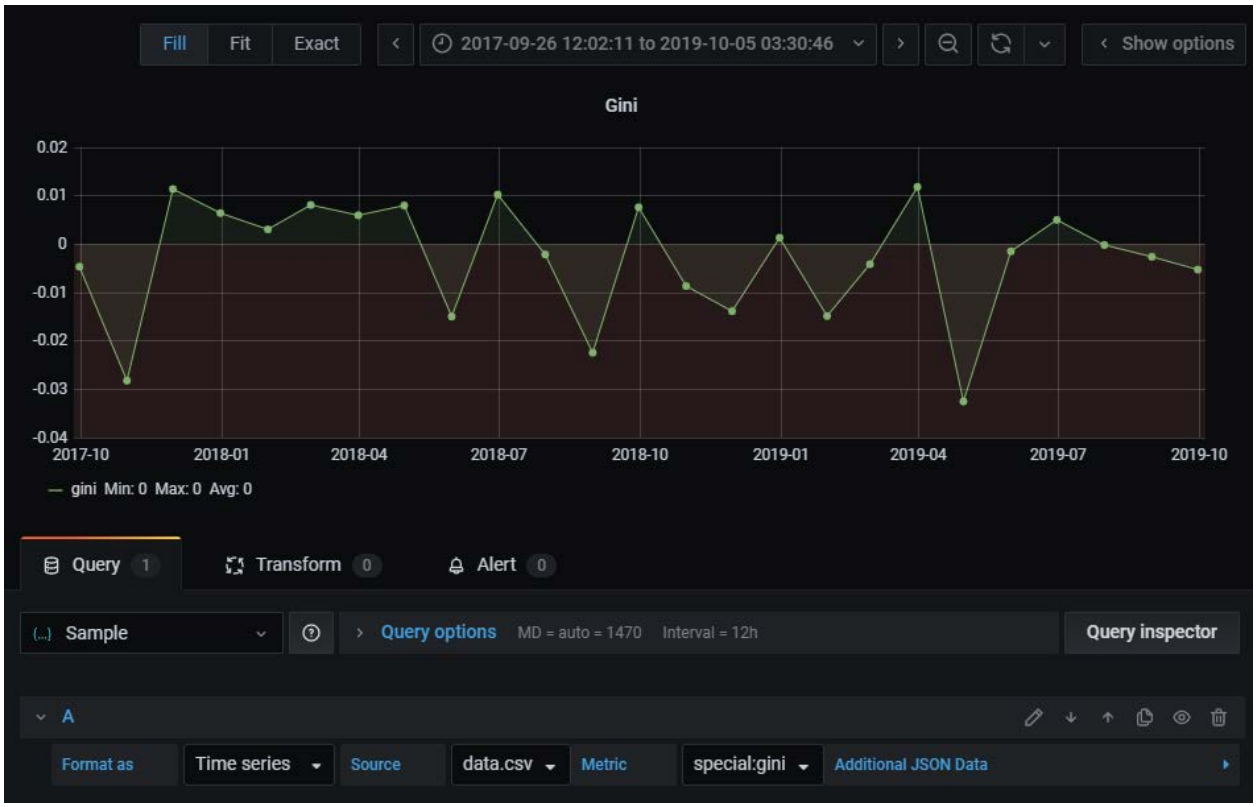

Рисунок 5- Панель показателя Gini

Для большинства метрик использовался специальный запрос, в котором просто проводилась агрегация на датафреймом. Исключением являются барометр средней PD для последней даты и гистограмма распределения рейтингов. В первом случае в настройках визуализации можно было выбрать, что именно из набора данных отображать, а во втором случае так же можно было определить тип графика. Итоговый дашборд представлен на рисунке 6.

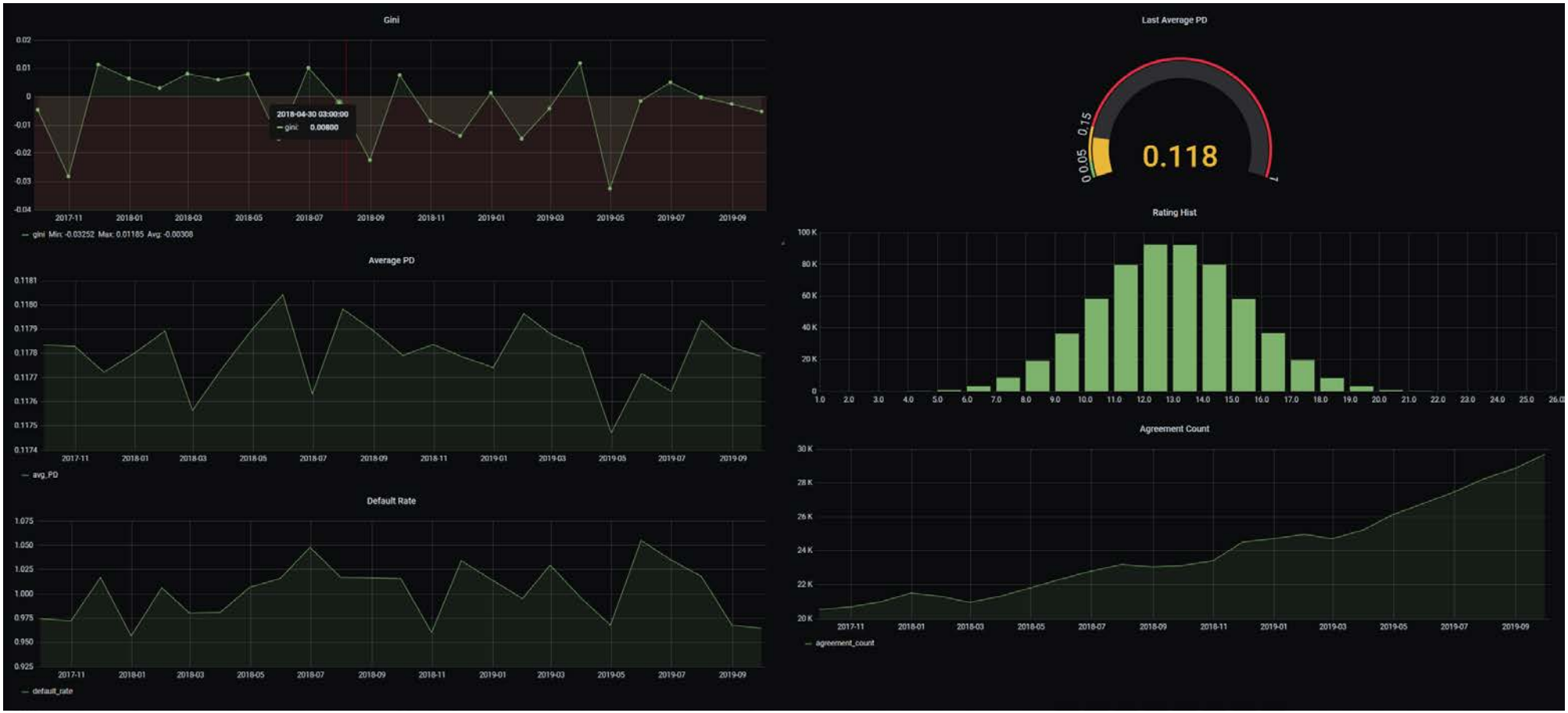

Рисунок 6- Дашборд показателей портфеля

Заключение

Таким образом в данной работе было разработано решения для создания системы наблюдения за данными с доступом к внешним файлам и был сконструирован дашборд для мониторинга состояния портфеля кредитных требований банка. Реализованный проект позволит облегчить создание систем мониторинга для малых объемов данных, а также вести визуальный анализ нескольких показателей портфеля в режиме реального времени.

Проект можно улучшать в многих направлениях. Например, добавить хорошую фильтрацию по значениям иных колонок, перенести агрегацию из pandas в поле запроса query, добавить поддержку большего количества типов, и т.п.

Созданный в процессе работы код, дашборд, а также инструкция по запуску хранятся на репозитории проекта [6].

Список использованных источников

1. Демо дашборд [Электронный ресурс]: пример дашборда Grafana. – URL: https://play.grafana.org/d/000000012/grafana-play-home?orgId=1 (дата обращения: 25.05.2020).
2. Grafana Documentation [Электронный ресурс]: документация по платформе Grafana. – URL: https://grafana.com/docs/ (дата обращения: 18.05.2020)
3. Plotly [Электронный ресурс]: плагин для Grafana. – URL: https://grafana.com/grafana/plugins/natel-plotly-panel (дата обращения: 25.05.2020)
4. Скрипт CSV источника для Grafana [Электронный ресурс]. – URL: https://github.com/SmartBlug/grafana-csv-datasource/blob/master/backend/PythonServer.py (дата обращения: 20.05.2020)
5. Оригинальный JSON плагин [Электронный ресурс]. – URL: https://grafana.com/grafana/plugins/simpod-json-datasource (дата обращения: 23.05.2020)
6. Репозиторий данной работы [Электронный ресурс]. – URL: https://github.com/illuser-maker/grafana-file-datasource