دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پروژه‌ی درس کلان‌داده

عنوان تکلیف

گزارش فردی پروژه‌ی آنالیز داده‌های تاکسی اینترنتی

زلفا شفرئی ۴۰۰۳۶۴۴۰۰۵

تیر ۱۴۰۱

مقدمه

این گزارش یه گزارش فردی می‌باشد که تنها بخش‌هایی از پروژه تحت مسئولیت بنده بوده است در آن توضیح داده شده است. لینک گیت‌هاب پروژه «https://github.com/zolfaShefreie/Online-Taxi-Service» است که برای دسترسی به کدها می‌توان از آن استفاده کرد.

اسکلت و ساختار پروژه

در ابتدا کدهای مربوط به فایل ستینگ و هندل کردن فایل env. در نظر گرفته و پیاده‌سازی شده است.

طبق صورت پروژه این پروژه شامل ۷ بلاک هستند که از داده‌های استریمی بر روی کافکا به صورت پایپ لاین استفاده می‌کنند که اسکلت پروژه بر طبق این موضوع زده شده است. برای مدیریت پایپ لاین این موضوع در نظر گرفته شده است هر بلاک از یک topic کافکا می‌خواند و به خروجی ای را به یک topic دیگر می‌ریزد. بلاک اول از روی فایل csv پروژه را می‌خواند و به یک تاپیکی با نام FileDataTopic میریزد. پس از آن به ترتیب هر بلاک در تاپیک‌ها بعدی خروجی را می‌نویسند و از تاپیک قبلی خود اطلاعات را خواهند گرفت.

```
TOPICS = ['FileDataTopic', 'ClusterTopic', 'ElasticTopic', 'CassandraTopic', 'RedisTopic',
          'PredictorTopic', 'PageRankTopic']
```

یک کلاس مدیریت کافکا برای مدیریت اعمال کافکا و بلاک ها زده شده است که به ترتیب عملیات‌ها زیر را انجام می‌دهد.

۱. پاک کردن تمامی تاپیک‌ها (برای از بین بردن نوشته‌های قبلی) و ساخت دوباره‌ی تاپیک‌ها
۲. ساخت آبجکت مربوط به هر بلاک و انتقال پارامترهای مربوطه مانند اسم دو تاپیک که از آن استفاده می‌کند و سشن اسپارک و آدرس bootsrap-server کافکا
۳. ران کردن آن بلاک در یک ترد جداگانه

به دلیل اینکه هر بلاک در واقع KafkaProducer وKafkaConsumer هم هست که باید همواره از تاپیکی بخواند و به تاپیکی بریزد کلاس مدیریت کافکا به صورت مالتی ترد در نظر گرفته شده است و این فرآیند را کنترل خواهد کرد. جزئیات این کلاس در فایل kafka_management.py قابل مشاهده می‌باشد.

برای بلاک‌ها یک ساختار ثابت در نظر گرفته شده است. این ساختار در کلاسی به نام BaseBlock پیاده سازی شده است و بلاک‌ها از این کلاس ارث‌بری خواهند. خواندن و نوشتن از روی کافکا می‌تواند با استفاده از کتاب‌خانه‌ی پایتون مربوط به کافکا و یا با استفاده از اسپارک صورت بگیرد به همین علت دو تایپ متفاوت نرمال و اسپارک برای آن در نظر گرفته شده است که هر کدام تابع‌های مختلفی را ران خواهند کرد. موارد در نظر گرفته شده در این کلاس:

۱. ساخت KafkaConsumer اگر تاپیک مربوط به خواندن در اختیار کلاس قرار داده بشه (برای هندل خواندن بلاک یک از فایل)
۲. ساخت KafkaProducer
۳. تابعی مربوط به نوشتن در تاپیک + سریالایز کردن ولیو
۴. تابعی برای ولیدیت کردن پارامتر کلاس

۵. تابعی برای خواندن پیام‌های تاپیک توسط اسپارک و تبدیل به فرمت قابل قبول (خروجی یک دیتافریم اسپارک می باشد که ستون‌هایی را طبق فرمت یکسان جیسون نوشته شده در کافکا درست و ذخیره می‌کند)
۶. تابعی برای پک کردن مجموعه ای از ستون ها و تبدیل به یک پیام با فرمت دیکشنری و نوشتن بر روی تاپیک با استفاده از اسپارک
۷. دیگر توابعی که به عنوان توابع پیاده سازی نشده ثبت شده اند تا هنگام ارث بری، کلاس فرزند کدهای مربوط به بلاک را در همین ساختار توابع ثبت کند.

جزئیات پیاده سازی و کدها در فایل base_classes.py در پوشه‌ی blocks قابل مشاهده است.

پس از زدن اسکلت کدهای مربوط به فاز اول به همین ساختار تبدیل شده است.

کلاسترینگ آنلاین (گام سوم)

این گام به دو صورت ممکن زده شده است:

۱. استفاده از اسپارک برای خواندن و نوشتن برروی کافکا و استفاده از یک مدل از قبل آموزش داده شده برای کلاستر کردن آن‌ها
۲. خواندن و نوشتن به صورت نرمال و تبدیل به داده‌ساختارهای اسپارک و استفاده از مدل‌های کلاسترینگ و آموزش مدل با استفاده از داده‌هایی که تاکنون از تاپیک کافکا خوانده شده است.

به دلیل سرعت بسیار پایین مورد دوم، مورد اول در ران کردن کد کلی درنظر گرفته شده است اما پیاده سازی آن‌ها به‌صورتی انجام شده است که به‌راحتی امکان جایگذاری مورد دوم به جای مورد اول انجام می‌گیرد.

بقیه‌ی حالت‌های پیاده‌سازی به دلیل مشکلات مربوط داده‌ساختارهای اسپارک و مدل streamingKMeans و نسخه‌ی بازنویسی شده‌ی آن توسط بنده و عدم سازگاری این موارد با یکدیگر و یا استفاده از نسخه‌ی اسپارک ۲ برای هندل کردن ماجرا و عدم امکان نوشتن خروجی در تاپیک مربوطه، قابل اجرا نبوده و تمامی موارد تست شده است.

داشبورد کیبانا

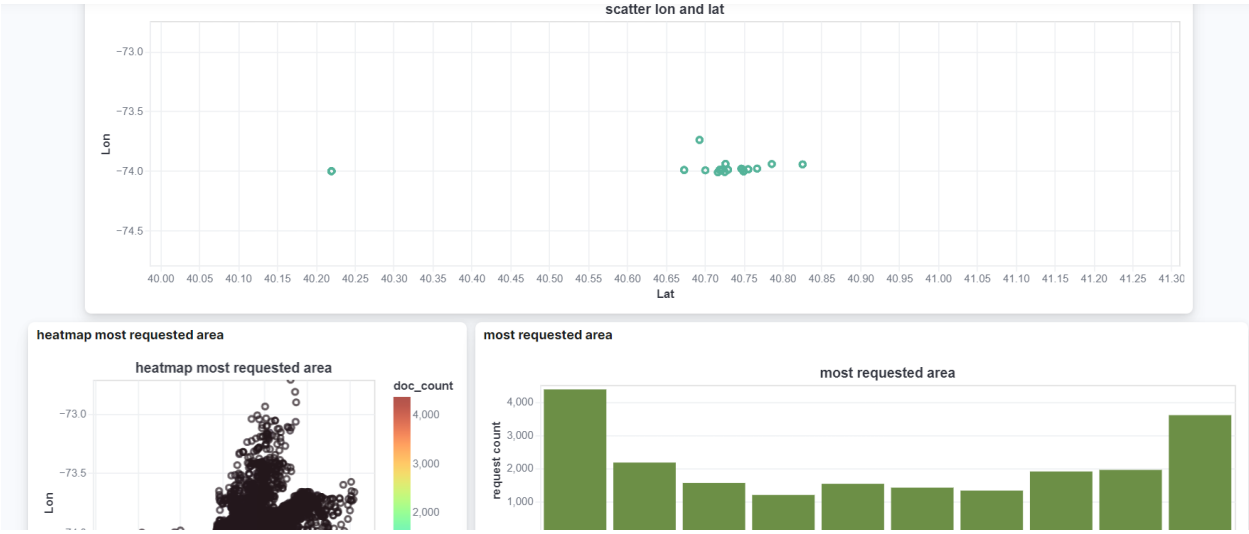
تمامی پلات‌های کیبانا در یک داشبورد مربوطه پیاده سازی شده است. خروجی تمامی پلات‌ها و پکت کامل داشبورد در پوشه‌ی elastic_kinaba_utils/kibana_exports ذخیره شده است که می‌توان با import کردن در کیبانا به آن دسترسی کامل داشت.

تمامی پلات با استفاده از vega-lite پیاده سازی شده‌اند که در توانایی کوئری زدن موجود می‌باشد. از خروجی کوئری پلات رسم می‌شود و عملیات‌هایی چون تبدیل های مورد نیاز، تعیین نوع پلات، استفاده از پارامترهایی برای زوم کردن و اکشن‌های مربوطه هنگام کلیک کردن و ...، تعیین پارامترهای پلات (برای مثال x و y) و شکل ظاهری آن در هر پلات پیاده سازی شده است.

۱. کوئری اول به صورت پلات اسکتر رسم شده است
۲. کوئری دوم با استفاده از بار پلات رسم شده است

۳. کوئری سوم با استفاده از اسکتر رسم شده است که رنگ‌ها نشان‌دهنده‌ی تراکم در هر ناحیه می‌باشد که با زوم کردن بر روی پلات می‌توان با دقت بهتری برای دید داشته باشید.

نمایی از داشبورد:



پیش‌بینی کننده (گام هفتم)

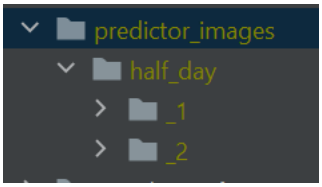
با وارد شدن هر داده‌ی استریمی داده‌ها از سه جدول کاساندرا با استفاده از اسپارک خوانده می‌شود. و عملیاتی برای گرفتن تعداد درخواست ها در هفته و ۱۲ ساعت و یک ماه انجام می‌شود. آموزش طبق یک محدودیت مقدار داده‌ی آموزشی انجام می‌شود.

```
MIN_NUMBER_START_TRAIN = {'half_day': 5, 'week': 5, 'month': 4}
TRAIN_WAIT_STEP = {'half_day': 5, 'week': 1, 'month': 1}
TEST_SPLIT = {'half_day': 1/5, 'week': 1/5, 'month': 1/3}
```

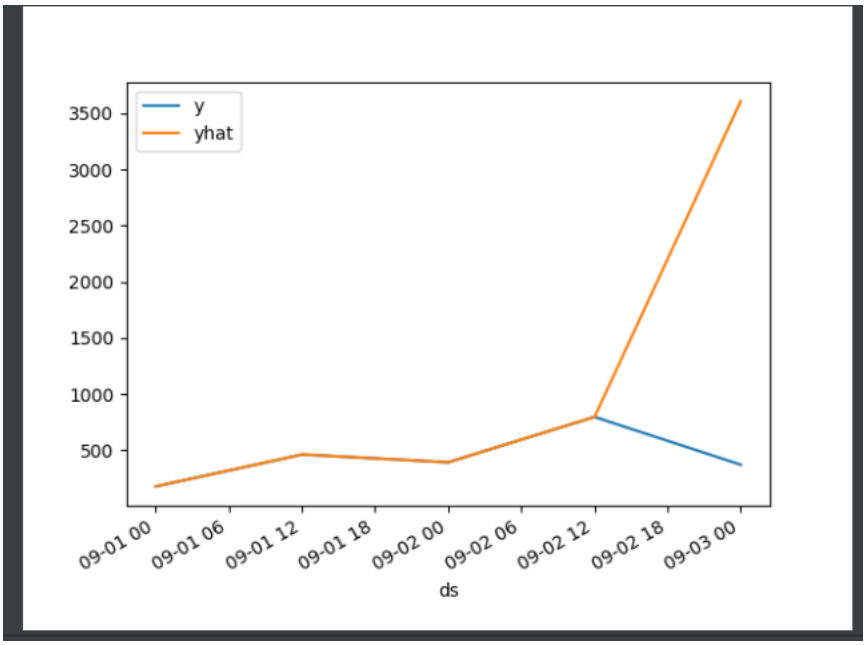
```
def _get_permission(self, df_count, kind="week") -> bool:
    """
    check can start new fit or not
    :param df_count:
    :param kind: half_day, week, month
    :return:
    """
    limit = (self.last_train_index[kind] * self.TRAIN_WAIT_STEP[kind]) + self.MIN_NUMBER_START_TRAIN[kind]
    return df_count >= limit
```

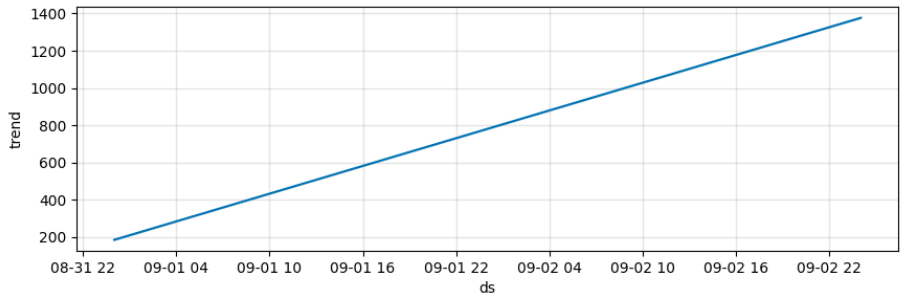
مدل پیش‌بینی کننده prophet فیسبوک برای پیش‌بینی استفاده می‌شود که مدلی برای پیش‌بینی داده‌های زمانی است. پیش پردازش‌های مربوط به این مدل باید انجام بگیرد (ورودی برای آموزش مدل باید دارای دو ستون ds (زمان‌ها) و y (برای پیش‌بینی) باشد).

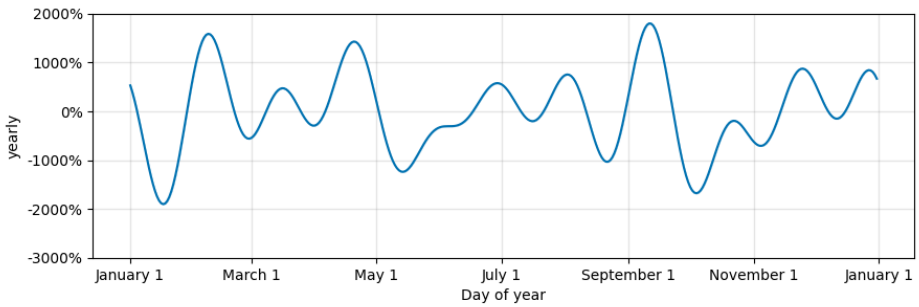
سه نوع پلات در مسیر predictor_images ذخیره می‌شود (هر بار آموزش پلات جدیدی را در پوشه‌ی جدید در همین پوشه تولید می‌کند).

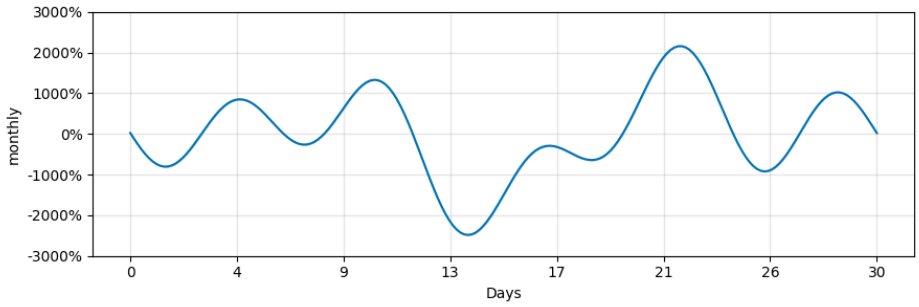


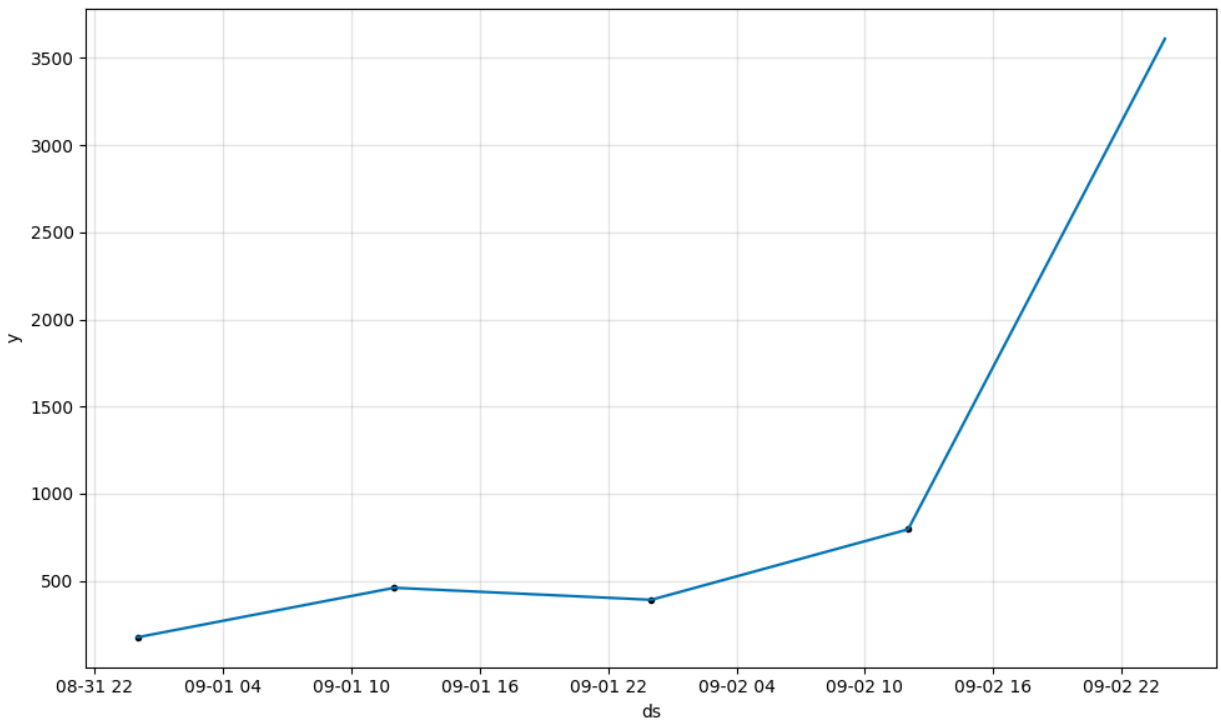
نمونه‌ای از پلات‌ها (دو پلات آخر توسط خود مدل کشیده می‌شود)




trend
1400
1200
1000
800
600
400
200
08-31 22
09-01 04
09-01 10
09-01 16
09-01 22
09-02 04
09-02 10
09-02 16
09-02 22
ds


yearly
2000%
1000%
0%
-1000%
-2000%
-3000%
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


monthly
3000%
2000%
1000%
0%
-1000%
-2000%
-3000%
0
4
9
13
17
21
26
30
Days


y
3500
3000
2500
2000
1500
1000
500
08-31 22
09-01 04
09-01 10
09-01 16
09-01 22
09-02 04
09-02 10
09-02 16
09-02 22
ds

پیج رنک (گام نهم)

در این گام گره‌ها نقاط و (نقطه‌ی استارت، نقطه‌ی end) لبه‌ها هستند.

با خواندن هر داده‌ی استریمی داده‌های در یک دیتافریم اسپارک و یک دیتافریم مربوط به گره‌ها (در آن نقاط ذخیره میشود که یک آیدی بر اساس نقطه ساخته در یک ستون آیدی ذخیره می‌شود) ذخیره می‌شوند اگر تعداد داده‌ها زوج باشند به ترتیب کارهای زیر انجام می‌شوند

۱. داده‌ها تقسیم بر ۲ میشوند و در مقابل یکدیگر قرار میگیرند (جوین می‌شوند که قسمت اول برای شروع سفر و قسمت دوم برای پایان سفر)
۲. ساخت دیتا فریم مربوط به لبه‌ها
۳. ساخت گراف بر اساس گره و لبه
۴. اجرای الگوریتم پیج رنک
۵. سورت بر اساس امتیاز ها و نوشتن ۳ تا نقاط پر امتیاز بر روی آخرین تاپیک