

Извештај за Домашна Задача 3

Систем за предвидување на дијабетес во реално време со Apache Spark и Kafka

1. Опис на проектот

Целта на овој проект е да се изгради комплетен систем за обработка на податоци (Data Pipeline) кој се состои од две фази: **Offline фаза** за тренирање на модел за машинско учење и **Online фаза** за предвидување на дијабетес во реално време користејќи стриминг податоци.

2. Подготовка на податоците

Податоците се преземени од *Diabetes Health Indicators Dataset*. За потребите на задачата, оригиналното множество е поделено на две датотеки:

- offline.csv**: Се користи за тренирање и евалуација на моделите.
- online.csv**: Се користи за симулација на стриминг податоци во реално време преку Kafka.

3. Offline фаза (Машинско учење)

Во оваа фаза, во рамките на Apache Spark апликација, се извршени следните чекори:

- Трансформации**: Дефиниран е метод за обработка кој користи `VectorAssembler` за спојување на карактеристиките во вектор и `StandardScaler` за нормализација на податоците.
- Тренирање**: Тестирани се 3 класификациски модели со користење на `CrossValidator` и `ParamGridBuilder`:
 - Logistic Regression**
 - Random Forest Classifier**
 - Decision Tree Classifier**
- Евалуација**: Моделите беа евалуирани според **F1 метриката**.
- Серијализација**: Најдобриот модел е зачуван локално во директориумот `saved_models/best_diabetes_model` како `PipelineModel` за да може да се вчита во online фазата со веќе дефинираните трансформации.

4. Архитектура на системот (Online фаза)

Online фазата е имплементирана преку микросервисна архитектура користејќи Docker контејнери.

- **Apache Kafka & Zookeeper:** Служат како пораки-брокер (message broker). Користени се два топици:
 - `health_data`: За сурови податоци од пациентите.
 - `health_data_predicted`: За збогатени податоци со предвидувањето од моделот.
 - **Kafka Producer (`producer.py`):** Скрипта која го чита `online.csv` ред по ред и ги испраќа податоците (без класата `Diabetes_012`) во JSON формат до Kafka.
 - **Spark Structured Streaming:** Апликација која го вчитува зачуваниот модел, се претплатува на `health_data`, врши предвидувања во реално време и ги испраќа назад во вториот топик.
 - **Kafka Consumer (`consumer.py`):** Скрипта која ги презема предвидувањата од `health_data_predicted` и ги зачувува во `predictions_log.txt`.
-

5. Технички предизвици и решенија (macOS)

За време на имплементацијата на macOS со Apple Silicon, беа надминати следните проблеми:

- **Bus Error / Java Gateway Error:** Решено со зголемување на ресурсите во Docker Desktop (8GB RAM) и користење на готов `jupyter/pyspark-notebook` имиџ наместо сопствен Dockerfile.
 - **Kafka Listeners:** Конфигурирани се две порти за Kafka:
 - `localhost:9092`: За надворешна комуникација (Mac терминал).
 - `kafka:29092`: За внатрешна комуникација (Spark во Docker).
-

6. Стартување на апликацијата

Процесот на извршување се одвива по следниот редослед:

1. **Подигнување на околината:** `docker-compose up -d`
2. **Креирање topics:**

```
docker exec -it kafka kafka-topics --create --topic health_data --bootstrap-server localhost:9092
```

```
docker exec -it kafka kafka-topics --create --topic health_data_predicted --bootstrap-server localhost:9092
```

3. **Стартување на Consumer:** `python3 consumer.py` (запишува во `predictions_log.txt`)
4. **Стартување на Spark Streaming:** Извршување на Notebook-от во Jupyter со претходно вчитани Kafka пакети.
5. **Стартување на Producer:** `python3 producer.py`

```
(myenv310) davidhristov@Davids-MacBook-Pro RNMP_homework3 % python consumer.py
Listening for predictions on topic: health_data_predicted
Received prediction: {'HighBP': 0.0, 'HighChol': 1.0, 'CholCheck': 1.0, 'BMI': 20.0, 'Smoker': 1.0, 'Stroke': 0.0, 'HeartDiseaseorAttack': 0.0, 'PhysActivity': 1.0, 'Fruits': 1.0, 'Veggies': 1.0, 'HvyAlcoholConsump': 0.0, 'AnyHealthcare': 1.0, 'NoDocbcCost': 0.0, 'GenHlth': 2.0, 'MentHlth': 0.0, 'PhysHlth': 0.0, 'DiffWalk': 0.0, 'Sex': 1.0, 'Age': 12.0, 'Education': 6.0, 'Income': 8.0, 'features': {'type': 1, 'values': [0.0, 1.0, 1.0, 20.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, 12.0, 6.0, 8.0]}, 'scaledFeatures': {'type': 1, 'values': [-0.8653874310579519, 1.1663430764329745, 0.1967049684043948, -1.2660887112687882, 1.1216118286574952, -0.20565426404167997, -0.3225555005613735, 0.5664463322923406, 0.7591267884944116, 0.48154049041807184, -0.2439704685462388, 0.22692690780505867, -0.30303478146747603, -0.47913481259850665, -0.42990064298079955, -0.487195521133236, -0.4498437064662096, 1.127172571598338, 1.2993978627805, 0.9621713158103339, 0.939704660930107]}, 'rawPrediction': {'type': 1, 'values': [2.2771327088825064, -1.922351171377567, -0.35478153750493974]}, 'probability': {'type': 1, 'values': [0.9200106521767623, 0.013803212682783627, 0.06618613514045432]}, 'prediction': 0.0}
```

7. Резултати

Системот успешно ги процесира податоците во реално време. Крајниот резултат во `predictions_log.txt` содржи информации за пациентот заедно со предвидената вредност (0.0 за здрави, 1.0/2.0 за дијабетес).

Примерок од излез:

Plaintext

```
Patient - Age: 12.0, BMI: 20.0, Prediction: 0.0
Patient - Age: 8.0, BMI: 34.0, Prediction: 0.0
Patient - Age: 12.0, BMI: 24.0, Prediction: 1.0
```

8. Заклучок

Успешно е демонстрирана интеграција помеѓу Spark MLlib и Spark Structured Streaming. Системот е скалабилен и овозможува брза дијагностика базирана на веќе тренирани историски податоци, што е од големо значење во медицинската информатика.

Изработил: Давид Христов **Датум:** Јануари, 2026