Big Data 1

PROJEKAT 1

Veljko Veljović 1937

ELEKTRONSKI FAKULTET NIŠ

Dataset - US Accidents (2016 - 2023)

1. Skup podataka o saobraćajnim nezgodama u 49 država u SAD.

2. Podaci su prikupljeni od februara 2016. do marta 2023. godine.

3. Dataset sadrži oko 7 miliona zapisa i veličine je 3.06 GB. Postoji i umanjena verzija sa oko 500 000 zapisa i veličine 188 MB. Dataset ima 46 kolona.

Analiza dataseta

1. Dataset sadrži veliki broj null vrednosti u različitim kolonama.

2. Takođe pojedine vrednosti kolona koje se odnose na vreme nisu standardizovane.

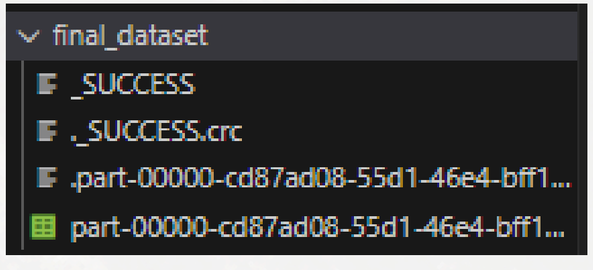
3. Pojedine kolone su preimenovane zbog lakšeg rada

dataset_cleaner.py

parsiranje ulaznih argumenata

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_timestamp
import argparse

parser = argparse.ArgumentParser()

parser.add_argument("--input_file", required=True, help="Path to the input CSV file")
parser.add_argument("--output_dir", required=True, help="Path to the output directory")

args = parser.parse_args()

input_file = args.input_file
output_dir = args.output_dir
```

```
:\Users\veljk\OneDrive\Desktop\Big-Data-1\Big Data - 1> spark-submit dataset_analysis.py --input_file us_accidents_sample.csv --output_dir final_dataset
```



sređivanje fajla

```python
spark = SparkSession.builder \
    .appName("Big Data") \
    .master("local[*]") \
    .getOrCreate()

df = spark.read.csv(input_file,
    header=True,
    inferSchema=True)

df = df.dropna()

df = df.withColumn(
    "Start_Time",
    to_timestamp(col("Start_Time"), "yyyy-MM-dd HH:mm:ss")
).withColumn(
    "End_Time",
    to_timestamp(col("End_Time"), "yyyy-MM-dd HH:mm:ss")
)

rename_dict = {
    'Distance(mi)': 'Distance_mi',
    'Temperature(F)': 'Temperature_F',
    'Wind_Chill(F)': 'Wind_Chill_F',
    'Humidity(%)': 'Humidity_percent',
    'Pressure(in)': 'Pressure_in',
    'Visibility(mi)': 'Visibility_mi',
    'Wind_Speed(mph)': 'Wind_Speed_mph',
    'Precipitation(in)': 'Precipitation_in'
}

for old_name, new_name in rename_dict.items():
    df = df.withColumnRenamed(old_name, new_name)

df.coalesce(1).write \
  .mode("overwrite") \
  .option("header", True) \
  .csv(output_dir)
```

Spark aplikacija

Parsiranje ulaznih argumenata i import

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_timestamp, lit, avg, count, stddev,min, max, variance, sum
import argparse
import time

start_time = time.perf_counter()

parser = argparse.ArgumentParser()
parser.add_argument("group_by_column", help="Column name to group by")
parser.add_argument("metric_column", help="Column name used for statistical calculations")
parser.add_argument("sum_column", help="Column name to sum over")
parser.add_argument("start_date", help="Start date")
parser.add_argument("end_date", help="End date")
parser.add_argument("input_file", help="Path to the input CSV file")

args = parser.parse_args()

group_by_column = args.group_by_column
metric_column = args.metric_column
sum_column = args.sum_column
start_date = args.start_date
end_date = args.end_date
local_path = args.input_file
```

Otvaranje spark sesije i čitanje fajla

```
spark = SparkSession.builder \
    .appName("Big Data") \
    .master("local[*]") \
    .getOrCreate()

df = spark.read \
    .format("csv") \
    .option("header", "true") \
    .load(local_path)
```

Spark aplikacija

```
df_filtered = df.filter(
    (col("Start_Time") >= to_timestamp(lit(start_date))) &
    (col("End_Time") <= to_timestamp(lit(end_date)))
).groupBy(group_by_column) \
 .agg(count("*").alias("broj_nezgoda"))\
 .coalesce(1).write \
 .mode("overwrite") \
 .option("header", "true") \
 .csv("output/filtered_count_by_parameters")

df_summed = (
    df.filter(
        (col("Start_Time") >= start_date) &
        (col("End_Time") <= end_date)
    )
    .groupBy(group_by_column)
    .agg(sum(col(sum_column)).alias(f"{sum_column}_sum"))
).coalesce(1).write \
 .mode("overwrite") \
 .option("header", "true") \
 .csv("output/sum_by_parameters")


df = df.groupBy(group_by_column).agg(
    min(col(metric_column)).alias(f"{metric_column}_min"),
    max(col(metric_column)).alias(f"{metric_column}_max"),
    avg(col(metric_column)).alias(f"{metric_column}_avg"),
    stddev(col(metric_column)).alias(f"{metric_column}_stddev"),
    variance(col(metric_column)).alias(f"{metric_column}_variance")
).coalesce(1).write \
 .mode("overwrite") \
 .option("header", "true") \
 .csv("output/grouped_stats")

spark.stop()

end_time = time.perf_counter()
elapsed_time = end_time - start_time

print(f"Elapsed time: {elapsed_time:.4f} seconds")
```

```
PS C:\Users\veljk\OneDrive\Desktop\Big-Data-1\Big Data - 1> spark-submit spark.py City Temperature_F Distance_mi 2016-01-01 2025-01-31 us_accidents_sample_cleaned.csv
```

Broj nesreca po gradovima

```
City,broj_nezgoda
Tyler,261
Worcester,12
Aitkin,18
Azalea,13
Osteen,9
Sugar City,1
Hanover,92
Prattville,18
Harleysville,29
Santa Paula,65
Hanceville,8
Johnsonburg,5
Lismore,2
Saint George,40
Blythewood,44
Deerwood,8
Springfield,682
```

Ukupna dužina puta u saobraćajnim nesrećama po gradovima

```
City,Distance_mi_sum
Tyler,38.650999999999996
Worcester,9.153
Aitkin,6.888000000000001
Azalea,22.893
Osteen,6.255
Sugar City,0.8640000000000001
Hanover,45.36300000000001
Prattville,15.533000000000001
Harleysville,5.987
Santa Paula,35.605999999999995
Hanceville,17.315999999999995
Johnsonburg,12.014999999999999
Lismore,1.2289999999999999
Saint George,56.654
Blythewood,13.947000000000001
Deerwood,2.432
Springfield,444.799
```

Spark aplikacija

Nalaženje statističkih informacija grupisano po atributu

```
City,Temperature_F_min,Temperature_F_max,Temperature_F_avg,Temperature_F_stddev,Temperature_F_variance
Abbottstown,42.0,73.0,60.285714285714285,13.634689651941548,185.90476190476193
Abingdon,19.0,93.0,57.54545454545455,17.883178710958543,319.80808080808083
Abington,20.0,94.0,56.14,18.241027987501592,332.73510204081634
Absarokee,49.0,94.0,71.5,31.81980515339464,1012.5
Absecon,53.0,82.0,64.41666666666667,9.15977702520728,83.90151515151513
Accokeek,32.0,80.0,58.666666666666664,16.46667893509542,271.1515151515152
Accomac,37.0,75.0,56.0,26.870057685088806,722.0
Acme,66.0,79.0,72.5,9.192388155425117,84.5
Acra,66.0,66.0,66.0,,
Acworth,28.0,81.0,55.40909090909091,13.164378671280533,173.30086580086578
Adair,16.0,77.0,38.25,26.663020584072363,710.9166666666666
Adairsville,29.0,75.0,59.0,20.591260281974,424.0
Adams Center,76.0,80.0,78.0,2.8284271247461903,8.0
Adel,17.0,88.0,41.78181818181818,20.079357710360313,403.1806060606061
Adelanto,37.0,95.0,69.15384615384616,19.038759386429543,362.47435897435906
Advance,41.0,86.0,51.888888888888886,16.019085838808376,256.6111111111111
Afton,14.0,92.0,46.51724137931034,21.82335035437169,476.25862068965506
```

Docker compose

```yaml
version: "3.8"

services:

  namenode:
    image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
    container_name: namenode
    restart: always
    ports:
      - "9870:9870"
      - "9000:9000"
    volumes:
      - hadoop_namenode:/hadoop/dfs/name
      - ./final_dataset:/data
    environment:
      - CLUSTER_NAME=test
    env_file:
      - ./hadoop.env
    networks:
      - bigdata-network

  datanode:
    image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
    container_name: datanode
    restart: always
    volumes:
      - hadoop_datanode:/hadoop/dfs/data
    environment:
      SERVICE_PRECONDITION: "namenode:9870"
    env_file:
      - ./hadoop.env
    networks:
      - bigdata-network

  resourcemanager:
    image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8
    container_name: resourcemanager
    restart: always
    environment:
      SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"
    env_file:
      - ./hadoop.env
    networks:
      - bigdata-network
```

```yaml
nodemanager:
  image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8
  container_name: nodemanager
  restart: always
  environment:
    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
  env_file:
    - ./hadoop.env
  networks:
    - bigdata-network

historyserver:
  image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8
  container_name: historyserver
  restart: always
  environment:
    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
  volumes:
    - hadoop_historyserver:/hadoop/yarn/timeline
  env_file:
    - ./hadoop.env
  networks:
    - bigdata-network

spark-master:
  image: bde2020/spark-master:3.1.2-hadoop3.2
  container_name: spark-master
  ports:
    - "8070:8080"
    - "7077:7077"
  volumes:
    - ./spark-apps:/opt/spark-apps
  environment:
    - INIT_DAEMON_STEP=setup_spark
  networks:
    - bigdata-network

spark-worker-1:
  image: bde2020/spark-worker:3.1.2-hadoop3.2
  container_name: spark-worker-1
  depends_on:
    - spark-master
  environment:
    - SPARK_MASTER=spark://spark-master:7077
  networks:
    - bigdata-network
```

```yaml
  spark-worker-2:
    image: bde2020/spark-worker:3.1.2-hadoop3.2
    container_name: spark-worker-2
    depends_on:
      - spark-master
    environment:
      - SPARK_MASTER=spark://spark-master:7077
    networks:
      - bigdata-network

volumes:
  hadoop_namenode:
  hadoop_datanode:
  hadoop_historyserver:

networks:
  bigdata-network:
    driver: bridge
```

Pokretanje i postavljanje podataka na HDFS

```
PS C:\Users\veljk\OneDrive\Desktop\Big-Data-1\Big Data - 1> docker compose up -d
time="2026-01-12T20:20:32+01:00" level=warning msg="C:\\Users\\veljk\\OneDrive\\Desktop\\Big-Data-1\\Big Data - 1\\docker-compose.yaml: the attribute `version` is obsolet
ion"
[+] Running 9/9
 ✓ Network bigdata-1_bigdata-network  Created
 ✓ Container spark-master             Started
 ✓ Container historyserver            Started
 ✓ Container namenode                 Started
 ✓ Container datanode                 Started
 ✓ Container resourcemanager          Started
 ✓ Container nodemanager              Started
 ✓ Container spark-worker-2           Started
 ✓ Container spark-worker-1           Started
PS C:\Users\veljk\OneDrive\Desktop\Big-Data-1\Big Data - 1> docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED         STATUS                            PORTS
   NAMES
2c9a9d637ac3   bde2020/spark-worker:3.1.2-hadoop3.2           "/bin/bash /worker.sh"   5 seconds ago   Up 3 seconds                      8081/tcp
   spark-worker-1
af52edcc2de3   bde2020/spark-worker:3.1.2-hadoop3.2           "/bin/bash /worker.sh"   5 seconds ago   Up 3 seconds                      8081/tcp
   spark-worker-2
d78f017890be   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"   5 seconds ago   Up 4 seconds (health: starting)   0.0.0.0:9000->9000/tcp
   namenode
2b69ec9182a1   bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   5 seconds ago   Up 4 seconds (health: starting)   8042/tcp
   nodemanager
e89fb06ffab0   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"   5 seconds ago   Up 4 seconds (health: starting)   9864/tcp
   datanode
836fe9b6502f   bde2020/spark-master:3.1.2-hadoop3.2           "/bin/bash /master.sh"   5 seconds ago   Up 4 seconds                      0.0.0.0:7077->7077/tcp
   spark-master
10b9cfefd26b   bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8   "/entrypoint.sh /run…"   5 seconds ago   Up 4 seconds (health: starting)   8088/tcp
   resourcemanager
817410d394e6   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8     "/entrypoint.sh /run…"   5 seconds ago   Up 4 seconds (health: starting)   8188/tcp
```

run.bat

```
@echo off
docker exec -i namenode hdfs dfs -test -d /input || \
docker exec -i namenode hdfs dfs -mkdir /input

docker exec -i namenode hdfs dfs -test -f /input/us_accidents_cleaned.csv || \
docker exec -i namenode hdfs dfs -put /data/us_accidents_cleaned.csv /input/
docker exec -i spark-master /spark/bin/spark-submit ^
  --master spark://spark-master:7077 ^
  --name big-data-app ^
  /opt/spark-apps/cluster-app.py ^
  City Temperature_F Distance_mi ^
  2016-01-01 2025-12-31
pause
```

cluster-app.py

```python
spark = SparkSession.builder \
    .appName("Big Data") \
    .getOrCreate()

spark.sparkContext.setLogLevel("ERROR")


df = spark.read \
    .format("csv") \
    .option("header", "true") \
    .load("hdfs://namenode:9000/input/us_accidents_cleaned.csv")

df_filtered = df.filter(
    (col("Start_Time") >= to_timestamp(lit(start_date))) &
    (col("End_Time") <= to_timestamp(lit(end_date)))
).groupBy(group_by_column) \
 .agg(count("*").alias("broj_nezgoda"))\
 .coalesce(1).write \
 .mode("overwrite") \
 .option("header", "true") \
 .csv("hdfs://namenode:9000/output/filtered_count_by_parameters")

df_summed = (
    df.filter(
        (col("Start_Time") >= start_date) &
        (col("End_Time") <= end_date)
    )
    .groupBy(group_by_column)
    .agg(sum(col(sum_column)).alias(f"{sum_column}_sum"))
).coalesce(1).write \
 .mode("overwrite") \
 .option("header", "true") \
 .csv("hdfs://namenode:9000/output/sum_by_parameters")


df = df.groupBy(group_by_column).agg(
    min(col(metric_column)).alias(f"{metric_column}_min"),
    max(col(metric_column)).alias(f"{metric_column}_max"),
    avg(col(metric_column)).alias(f"{metric_column}_avg"),
    stddev(col(metric_column)).alias(f"{metric_column}_stddev"),
    variance(col(metric_column)).alias(f"{metric_column}_variance")
).coalesce(1).write \
 .mode("overwrite") \
 .option("header", "true") \
 .csv("hdfs://namenode:9000/output/grouped_stats")
```

Upisuje i čita podatke sa hdfs-a

Evaluacija

Lokalno izvršenje

```
26/01/13 11:48:57 INFO SparkContext: Successfully stopped SparkContext
Elapsed time: 20.3273 seconds
```

Na klasteru

```
Elapsed time: 39.5985 seconds
Press any key to continue . . .
```

Na rezultate u izvršenju aplikacija utiču i veličine fajlova koje se obrađuju, kao i overhead u komunikaciji kada se aplikacija pokreće na klasteru računara.

Hvala na pažnji!