Big Data 3

PROJEKAT 3

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.

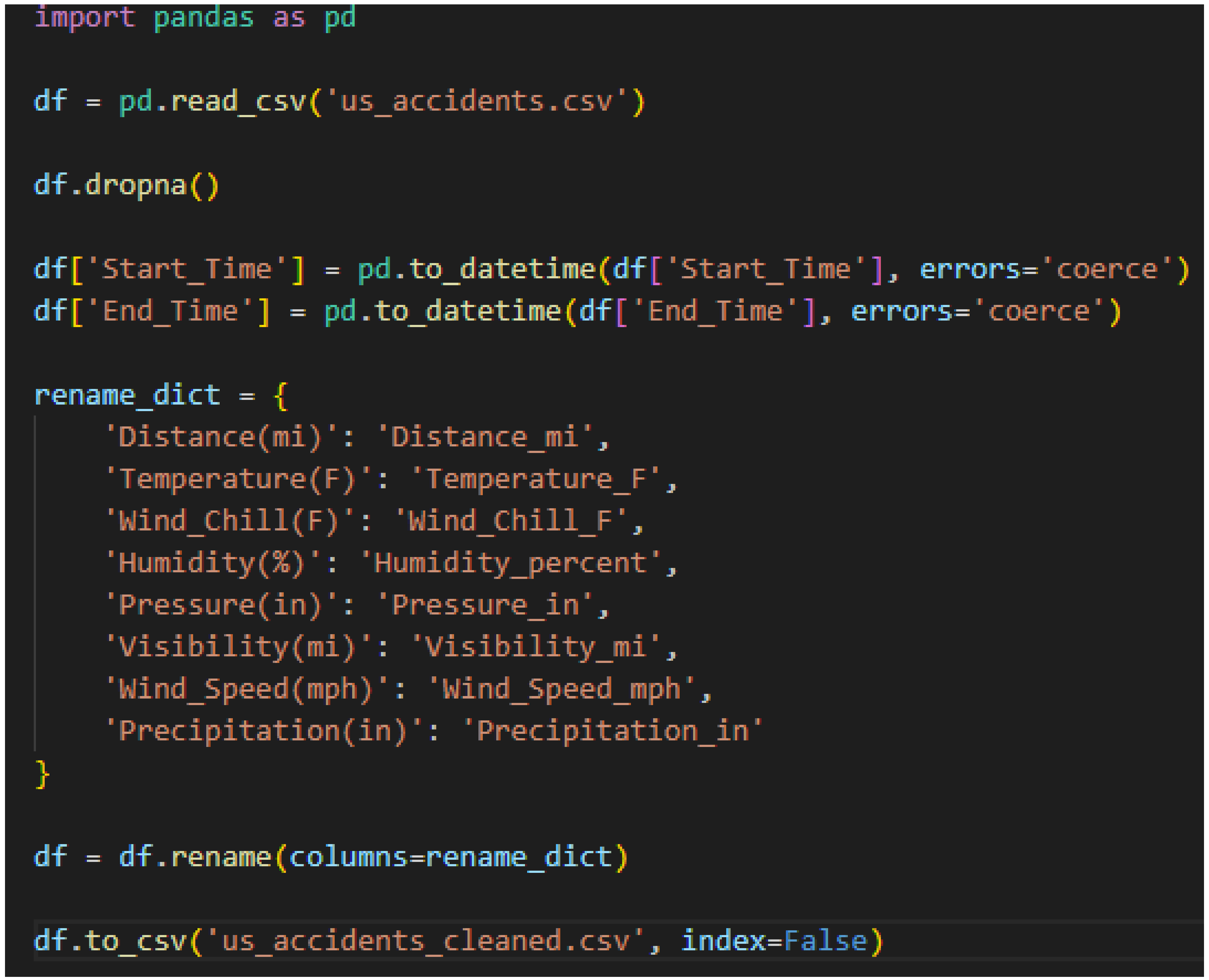
Analiza dataseta

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

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

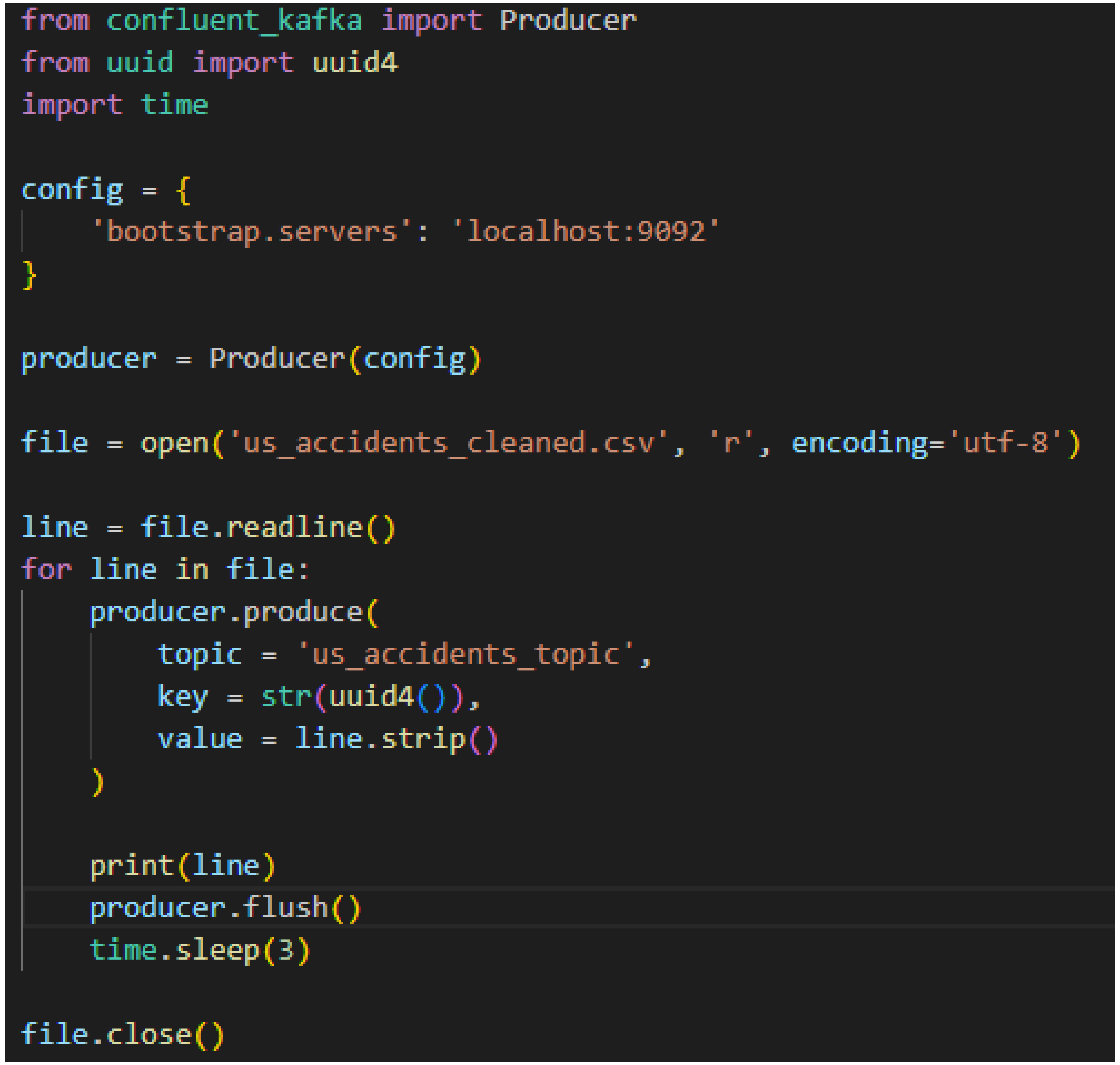
3. Pojedine kolone su preimenovane zbog lakšeg rada

preprocessor.py



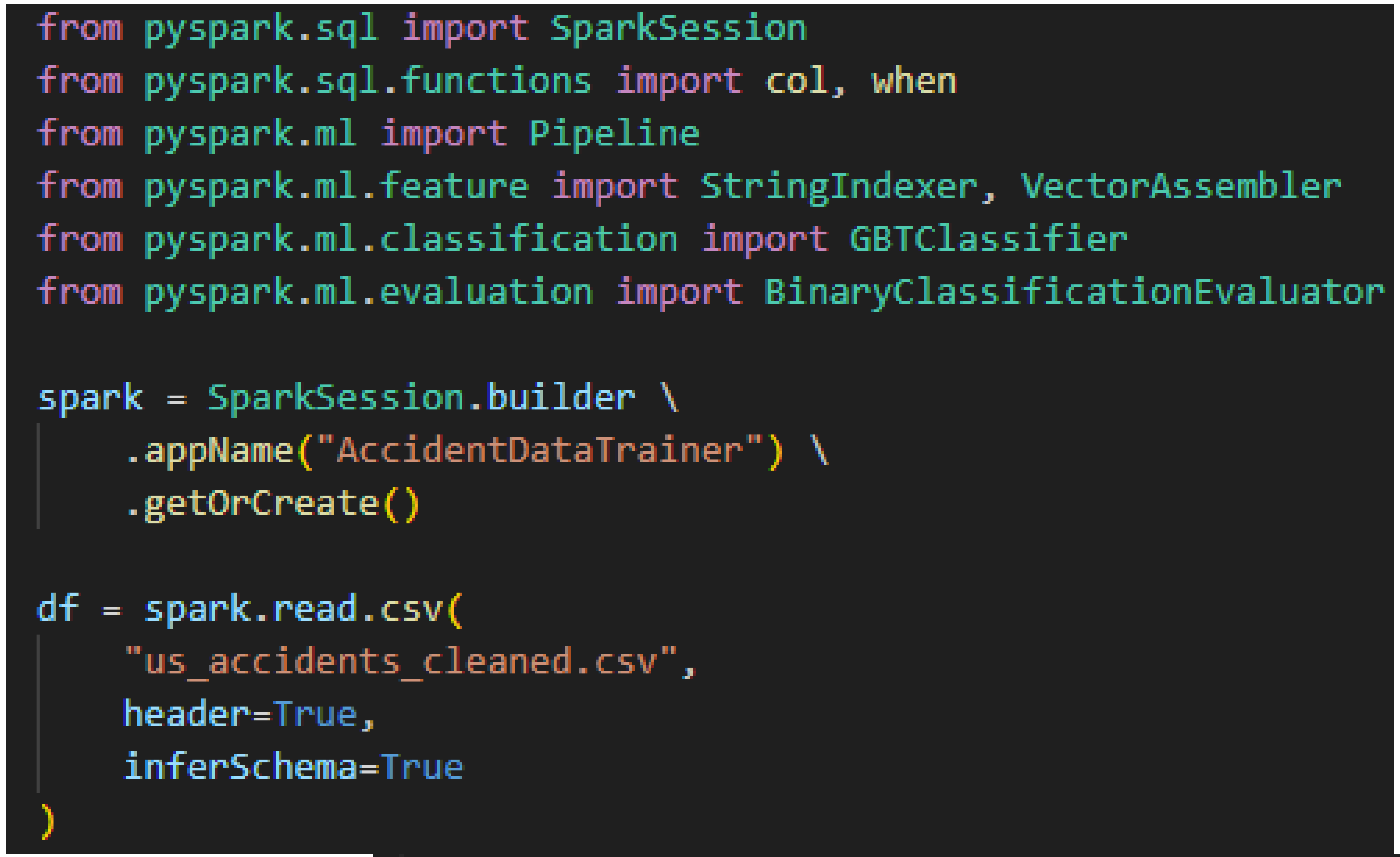
```python
import pandas as pd

df = pd.read_csv('us_accidents.csv')

df.dropna()

df['Start_Time'] = pd.to_datetime(df['Start_Time'], errors='coerce')
df['End_Time'] = pd.to_datetime(df['End_Time'], errors='coerce')

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'
}

df = df.rename(columns=rename_dict)

df.to_csv('us_accidents_cleaned.csv', index=False)
```

producer.py

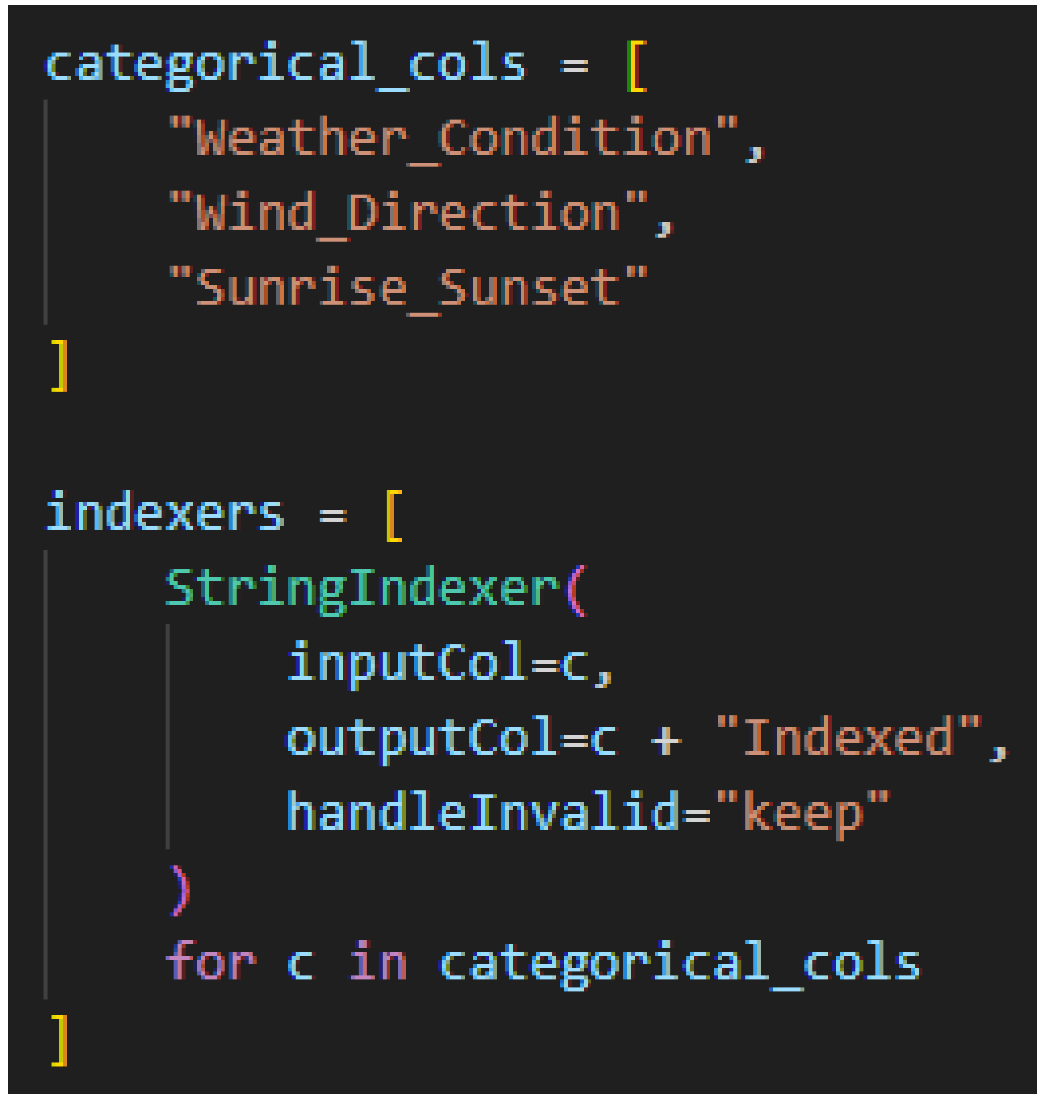


```python
from confluent_kafka import Producer
from uuid import uuid4
import time

config = {
    'bootstrap.servers': 'localhost:9092'
}

producer = Producer(config)

file = open('us_accidents_cleaned.csv', 'r', encoding='utf-8')

line = file.readline()
for line in file:
    producer.produce(
        topic = 'us_accidents_topic',
        key = str(uuid4()),
        value = line.strip()
    )

    print(line)
    producer.flush()
    time.sleep(3)

file.close()
```

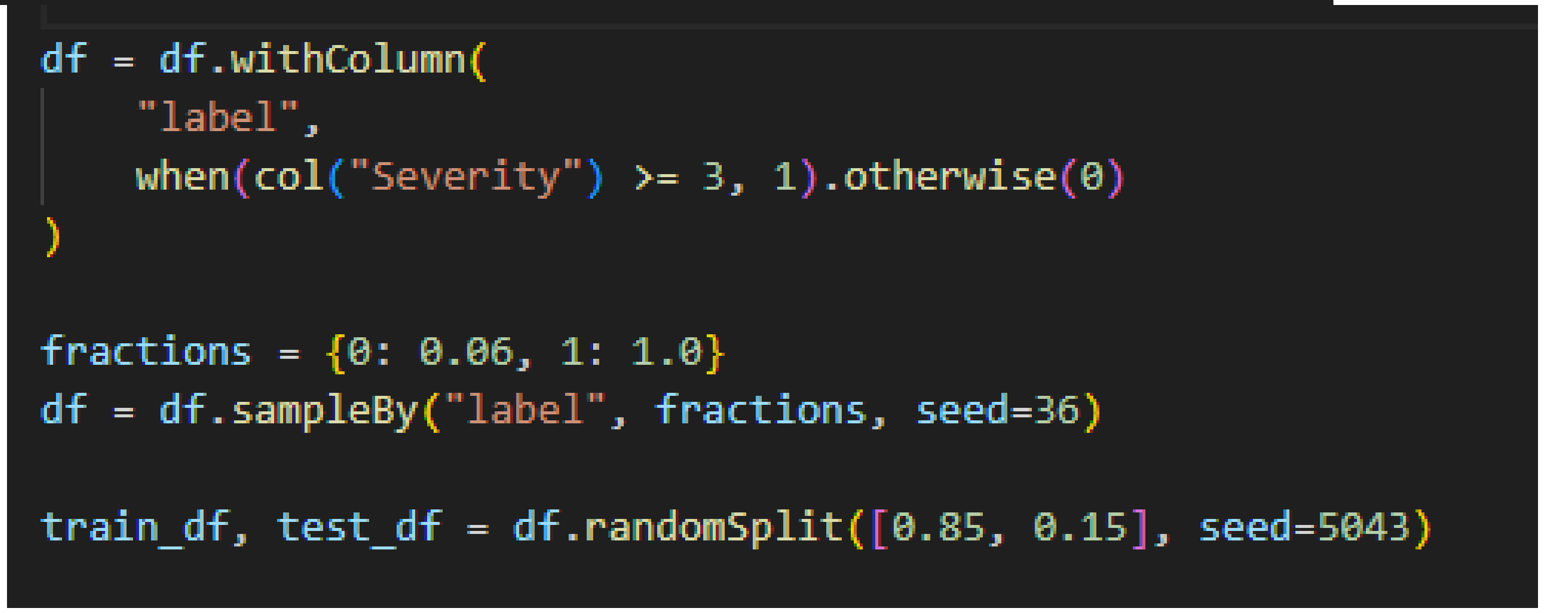
trainer.py



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator

spark = SparkSession.builder \
    .appName("AccidentDataTrainer") \
    .getOrCreate()

df = spark.read.csv(
    "us_accidents_cleaned.csv",
    header=True,
    inferSchema=True
)
```

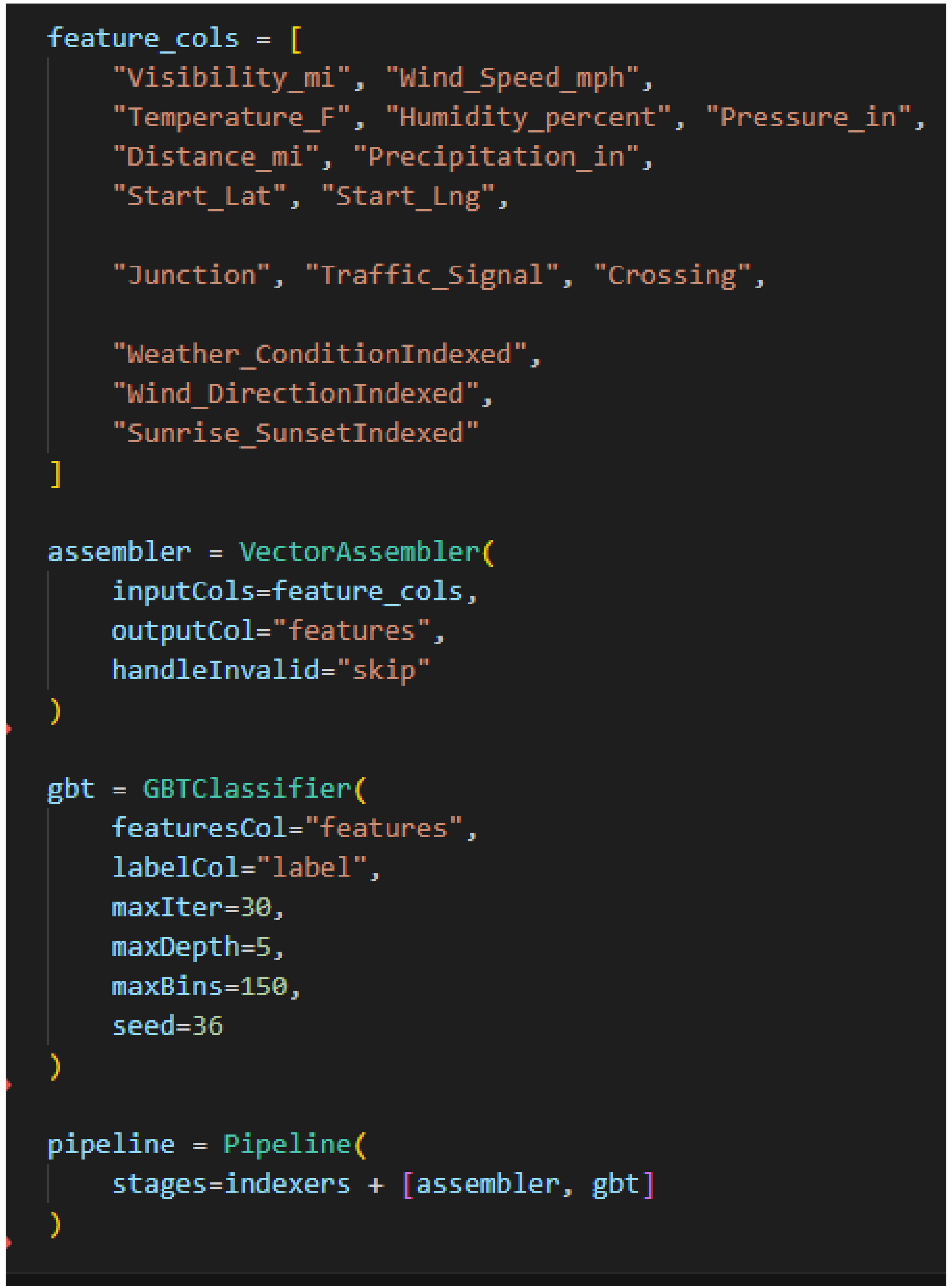


```python
categorical_cols = [
    "Weather_Condition",
    "Wind_Direction",
    "Sunrise_Sunset"
]

indexers = [
    StringIndexer(
        inputCol=c,
        outputCol=c + "Indexed",
        handleInvalid="keep"
    )
    for c in categorical_cols
]
```

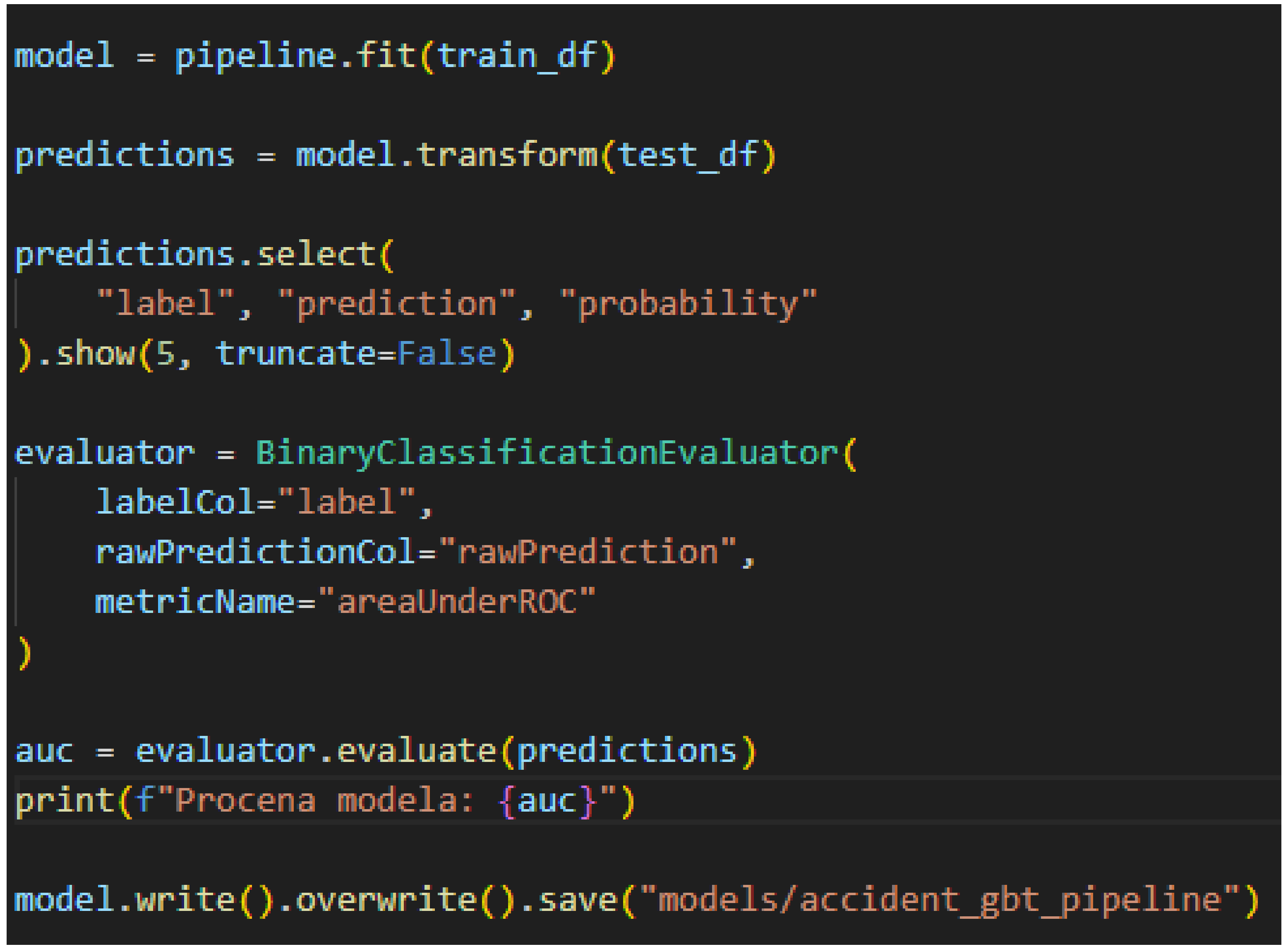


```python
df = df.withColumn(
    "label",
    when(col("Severity") >= 3, 1).otherwise(0)
)

fractions = {0: 0.06, 1: 1.0}
df = df.sampleBy("label", fractions, seed=36)

train_df, test_df = df.randomSplit([0.85, 0.15], seed=5043)
```

trainer.py



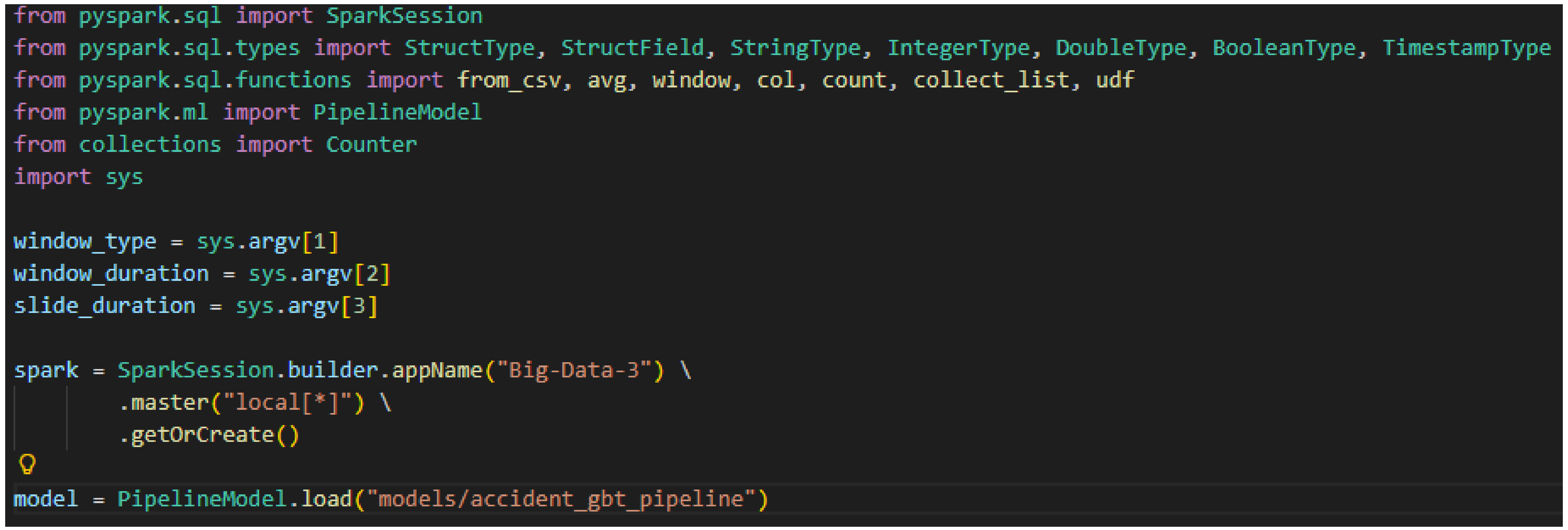
```python
feature_cols = [
    "Visibility_mi", "Wind_Speed_mph",
    "Temperature_F", "Humidity_percent", "Pressure_in",
    "Distance_mi", "Precipitation_in",
    "Start_Lat", "Start_Lng",

    "Junction", "Traffic_Signal", "Crossing",

    "Weather_ConditionIndexed",
    "Wind_DirectionIndexed",
    "Sunrise_SunsetIndexed"
]

assembler = VectorAssembler(
    inputCols=feature_cols,
    outputCol="features",
    handleInvalid="skip"
)

gbt = GBTClassifier(
    featuresCol="features",
    labelCol="label",
    maxIter=30,
    maxDepth=5,
    maxBins=150,
    seed=36
)

pipeline = Pipeline(
    stages=indexers + [assembler, gbt]
)
```



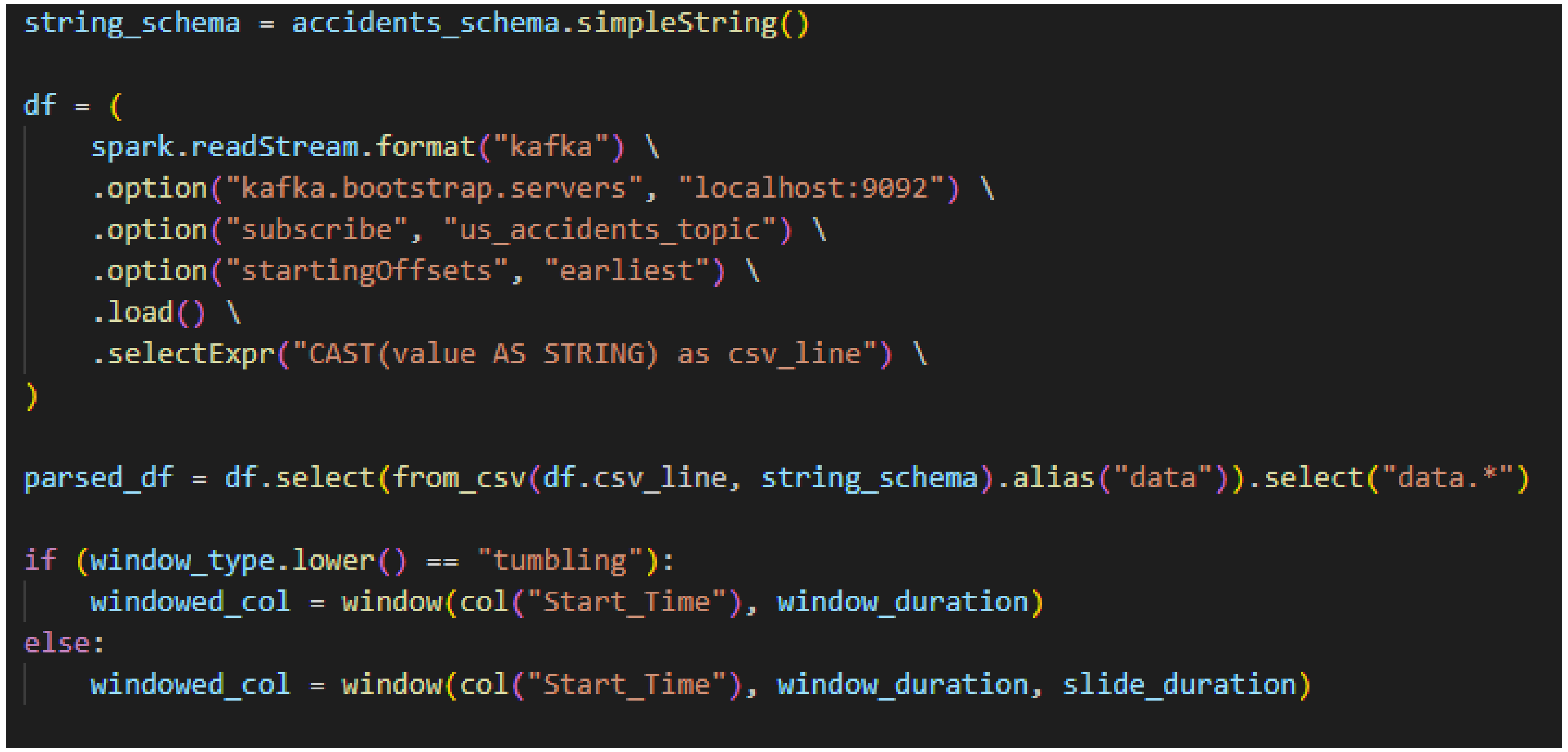
```python
model = pipeline.fit(train_df)

predictions = model.transform(test_df)

predictions.select(
    "label", "prediction", "probability"
).show(5, truncate=False)

evaluator = BinaryClassificationEvaluator(
    labelCol="label",
    rawPredictionCol="rawPrediction",
    metricName="areaUnderROC"
)

auc = evaluator.evaluate(predictions)
print(f"Procena modela: {auc}")

model.write().overwrite().save("models/accident_gbt_pipeline")
```

Procena modela:
0.8266998013514448

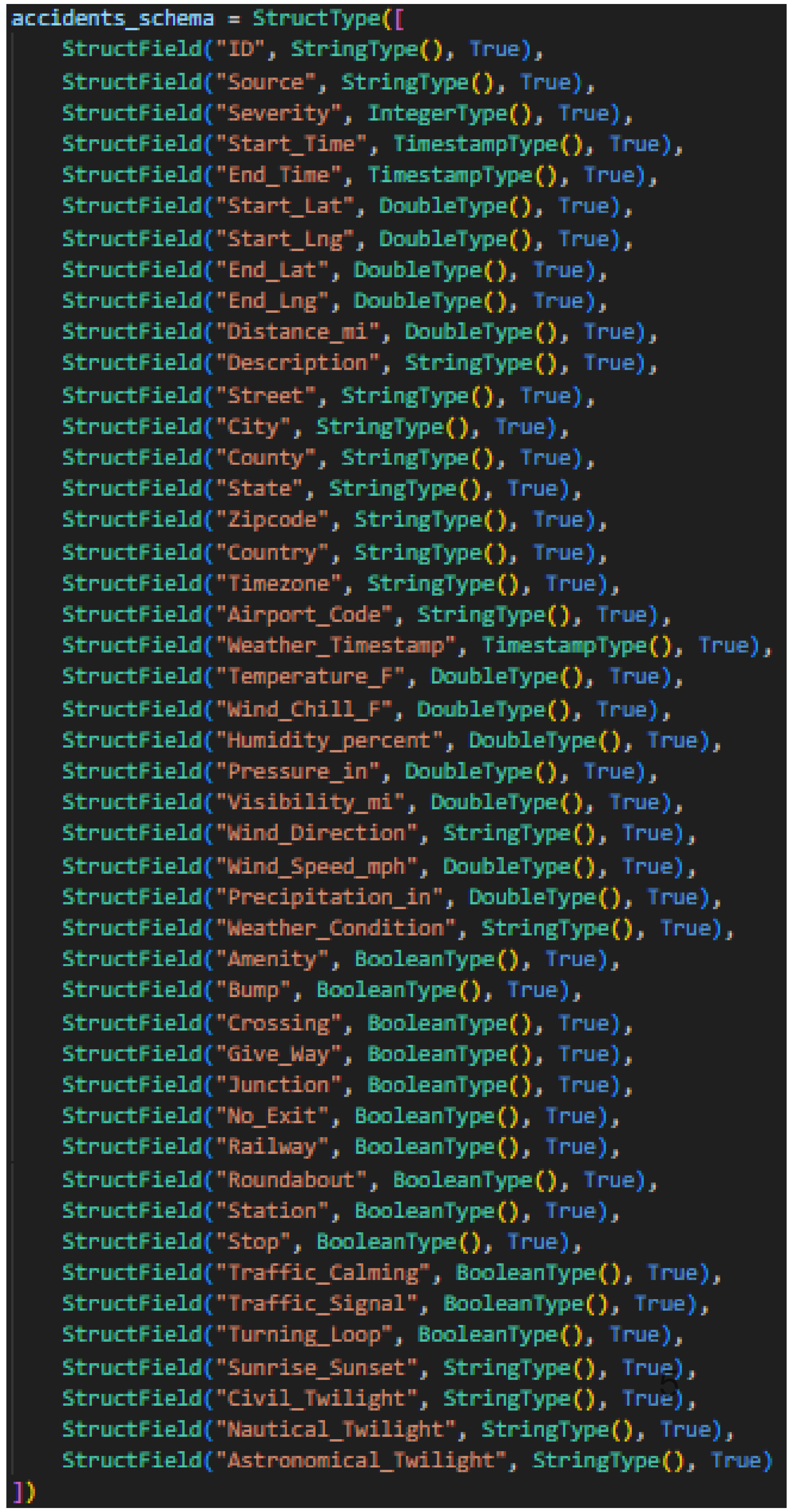
consumer.py



```python
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, BooleanType, TimestampType
from pyspark.sql.functions import from_csv, avg, window, col, count, collect_list, udf
from pyspark.ml import PipelineModel
from collections import Counter
import sys

window_type = sys.argv[1]
window_duration = sys.argv[2]
slide_duration = sys.argv[3]

spark = SparkSession.builder.appName("Big-Data-3") \
        .master("local[*]") \
        .getOrCreate()

model = PipelineModel.load("models/accident_gbt_pipeline")
```

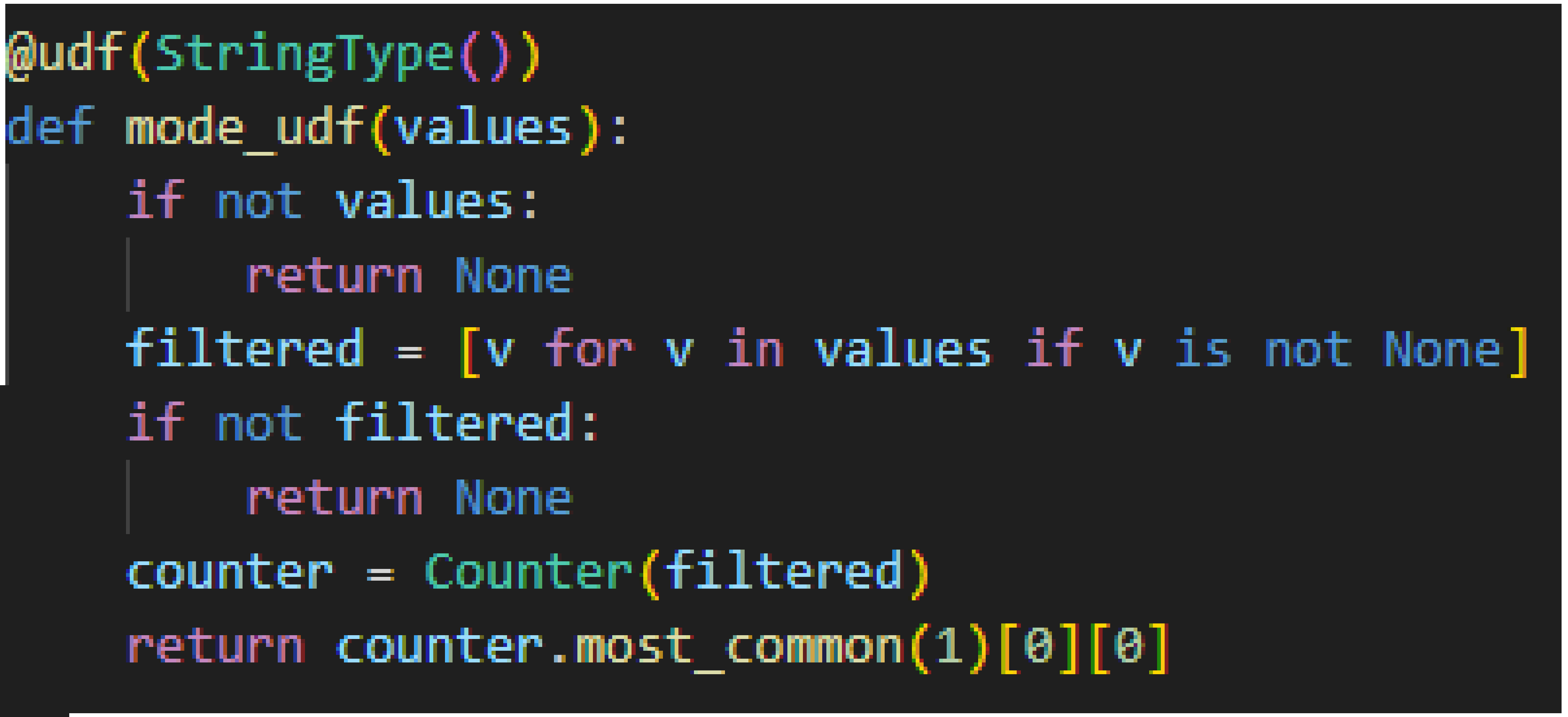


```python
string_schema = accidents_schema.simpleString()

df = (
    spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "us_accidents_topic") \
    .option("startingOffsets", "earliest") \
    .load() \
    .selectExpr("CAST(value AS STRING) as csv_line") \
)

parsed_df = df.select(from_csv(df.csv_line, string_schema).alias("data")).select("data.*")

if (window_type.lower() == "tumbling"):
    windowed_col = window(col("Start_Time"), window_duration)
else:
    windowed_col = window(col("Start_Time"), window_duration, slide_duration)
```

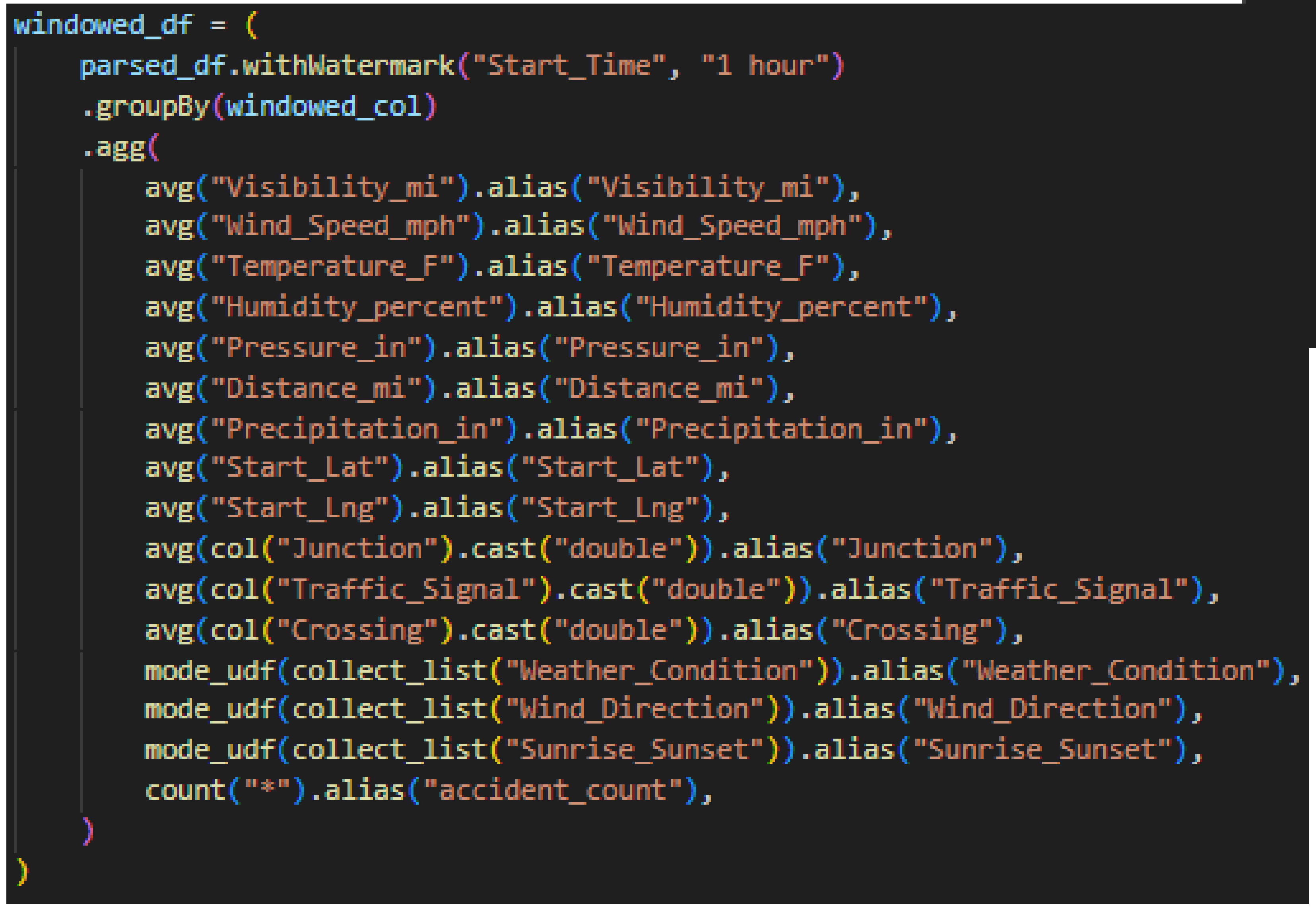


```python
accidents_schema = StructType([
    StructField("ID", StringType(), True),
    StructField("Source", StringType(), True),
    StructField("Severity", IntegerType(), True),
    StructField("Start_Time", TimestampType(), True),
    StructField("End_Time", TimestampType(), True),
    StructField("Start_Lat", DoubleType(), True),
    StructField("Start_Lng", DoubleType(), True),
    StructField("End_Lat", DoubleType(), True),
    StructField("End_Lng", DoubleType(), True),
    StructField("Distance_mi", DoubleType(), True),
    StructField("Description", StringType(), True),
    StructField("Street", StringType(), True),
    StructField("City", StringType(), True),
    StructField("County", StringType(), True),
    StructField("State", StringType(), True),
    StructField("Zipcode", StringType(), True),
    StructField("Country", StringType(), True),
    StructField("Timezone", StringType(), True),
    StructField("Airport_Code", StringType(), True),
    StructField("Weather_Timestamp", TimestampType(), True),
    StructField("Temperature_F", DoubleType(), True),
    StructField("Wind_Chill_F", DoubleType(), True),
    StructField("Humidity_percent", DoubleType(), True),
    StructField("Pressure_in", DoubleType(), True),
    StructField("Visibility_mi", DoubleType(), True),
    StructField("Wind_Direction", StringType(), True),
    StructField("Wind_Speed_mph", DoubleType(), True),
    StructField("Precipitation_in", DoubleType(), True),
    StructField("Weather_Condition", StringType(), True),
    StructField("Amenity", BooleanType(), True),
    StructField("Bump", BooleanType(), True),
    StructField("Crossing", BooleanType(), True),
    StructField("Give_Way", BooleanType(), True),
    StructField("Junction", BooleanType(), True),
    StructField("No_Exit", BooleanType(), True),
    StructField("Railway", BooleanType(), True),
    StructField("Roundabout", BooleanType(), True),
    StructField("Station", BooleanType(), True),
    StructField("Stop", BooleanType(), True),
    StructField("Traffic_Calming", BooleanType(), True),
    StructField("Traffic_Signal", BooleanType(), True),
    StructField("Turning_Loop", BooleanType(), True),
    StructField("Sunrise_Sunset", StringType(), True),
    StructField("Civil_Twilight", StringType(), True),
    StructField("Nautical_Twilight", StringType(), True),
    StructField("Astronomical_Twilight", StringType(), True)
])
```

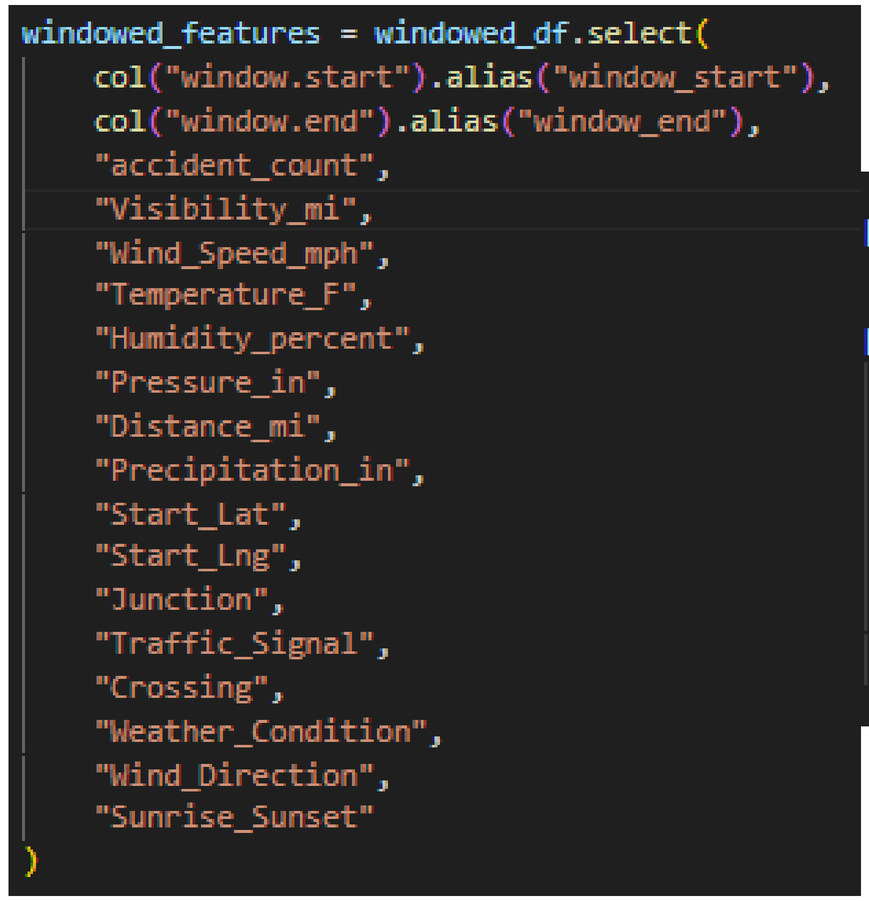
consumer.py



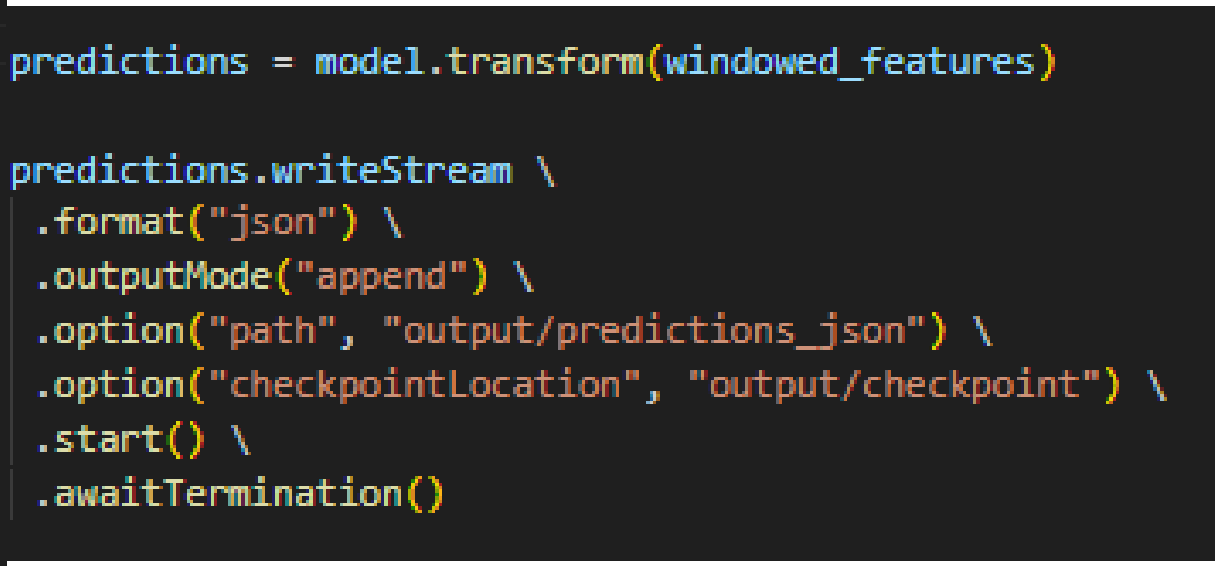
```python
@udf(StringType())
def mode_udf(values):
    if not values:
        return None
    filtered = [v for v in values if v is not None]
    if not filtered:
        return None
    counter = Counter(filtered)
    return counter.most_common(1)[0][0]
```



```python
windowed_df = (
    parsed_df.withWatermark("Start_Time", "1 hour")
    .groupBy(windowed_col)
    .agg(
        avg("Visibility_mi").alias("Visibility_mi"),
        avg("Wind_Speed_mph").alias("Wind_Speed_mph"),
        avg("Temperature_F").alias("Temperature_F"),
        avg("Humidity_percent").alias("Humidity_percent"),
        avg("Pressure_in").alias("Pressure_in"),
        avg("Distance_mi").alias("Distance_mi"),
        avg("Precipitation_in").alias("Precipitation_in"),
        avg("Start_Lat").alias("Start_Lat"),
        avg("Start_Lng").alias("Start_Lng"),
        avg(col("Junction").cast("double")).alias("Junction"),
        avg(col("Traffic_Signal").cast("double")).alias("Traffic_Signal"),
        avg(col("Crossing").cast("double")).alias("Crossing"),
        mode_udf(collect_list("Weather_Condition")).alias("Weather_Condition"),
        mode_udf(collect_list("Wind_Direction")).alias("Wind_Direction"),
        mode_udf(collect_list("Sunrise_Sunset")).alias("Sunrise_Sunset"),
        count("*").alias("accident_count"),
    )
)
```

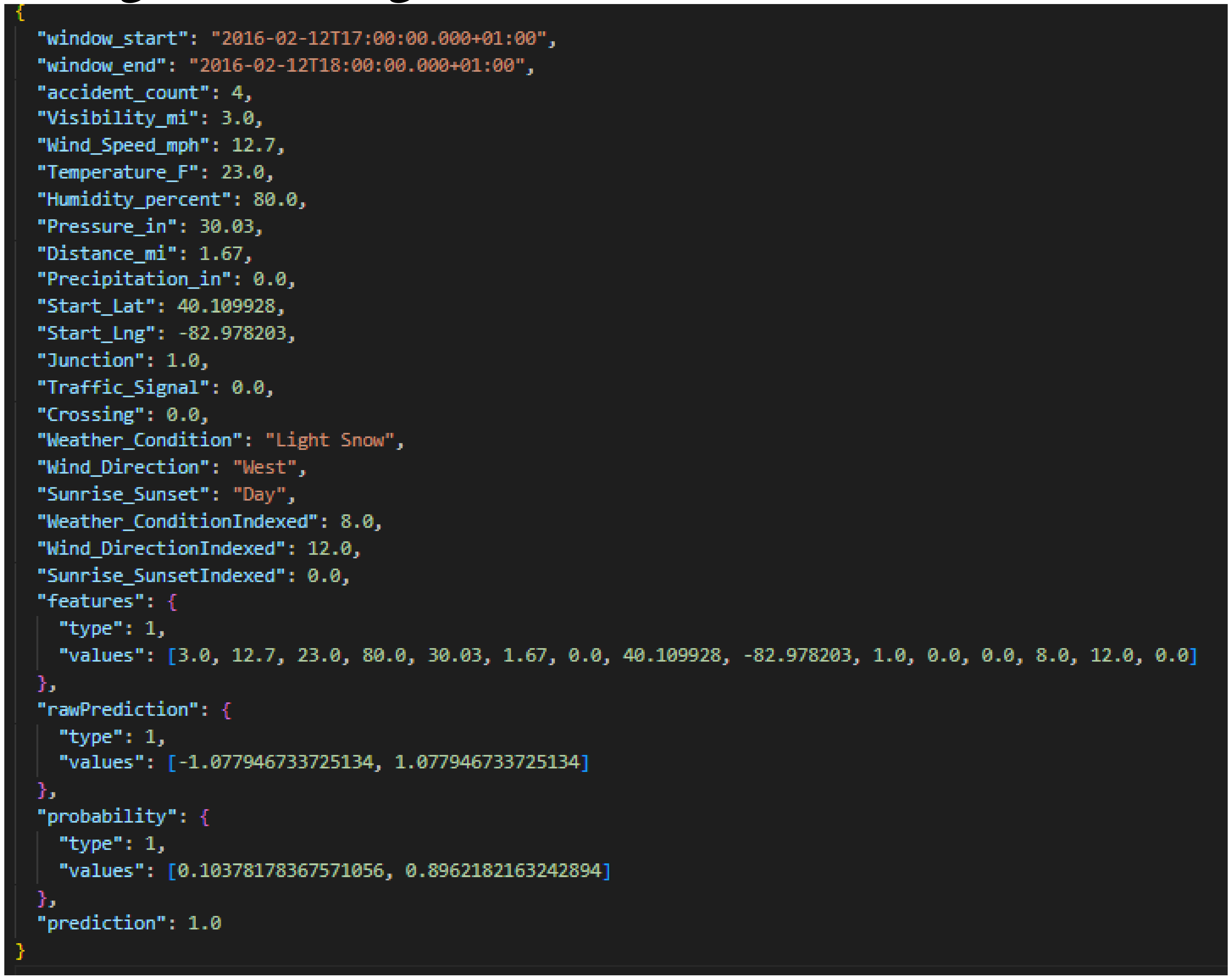


```python
windowed_features = windowed_df.select(
    col("window.start").alias("window_start"),
    col("window.end").alias("window_end"),
    "accident_count",
    "Visibility_mi",
    "Wind_Speed_mph",
    "Temperature_F",
    "Humidity_percent",
    "Pressure_in",
    "Distance_mi",
    "Precipitation_in",
    "Start_Lat",
    "Start_Lng",
    "Junction",
    "Traffic_Signal",
    "Crossing",
    "Weather_Condition",
    "Wind_Direction",
    "Sunrise_Sunset"
)
```



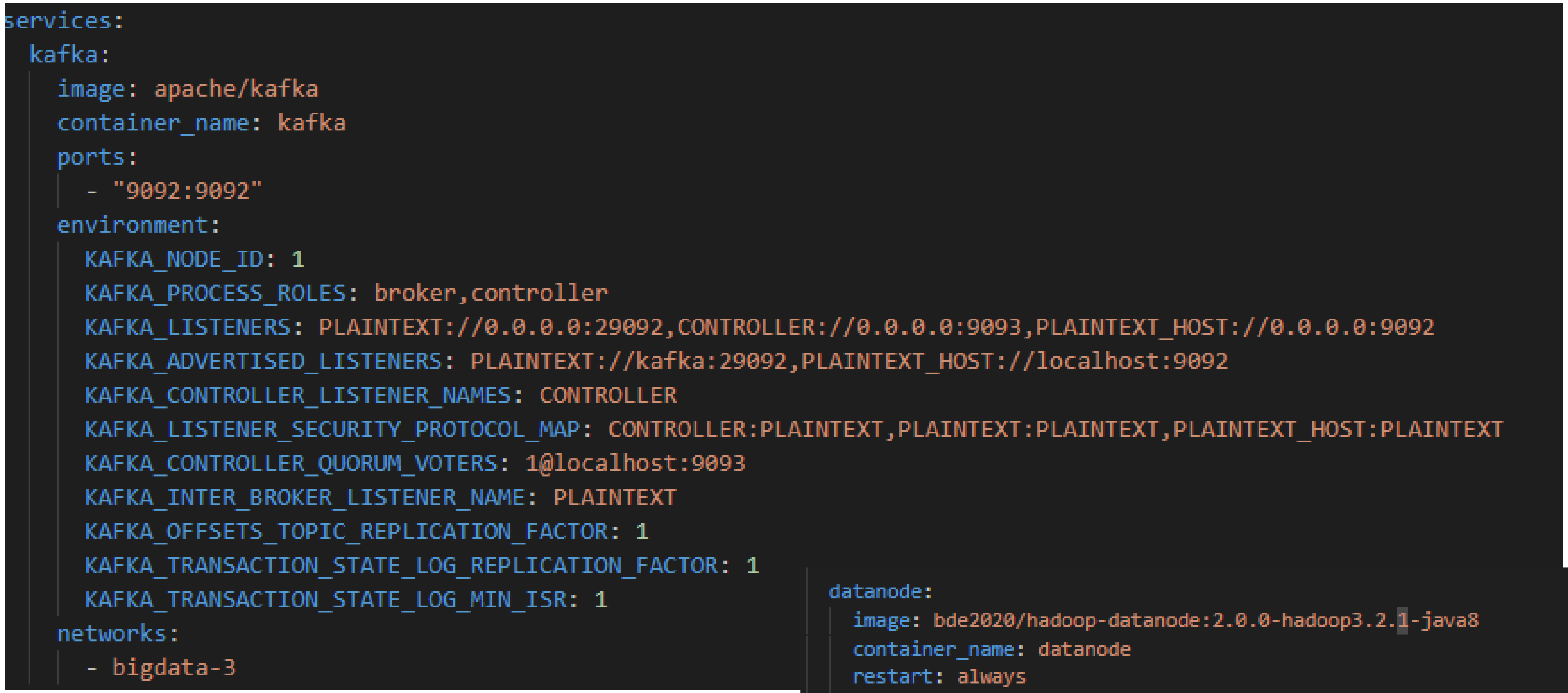
```python
predictions = model.transform(windowed_features)

predictions.writeStream \
 .format("json") \
 .outputMode("append") \
 .option("path", "output/predictions_json") \
 .option("checkpointLocation", "output/checkpoint") \
 .start() \
 .awaitTermination()
```

primer json fajla

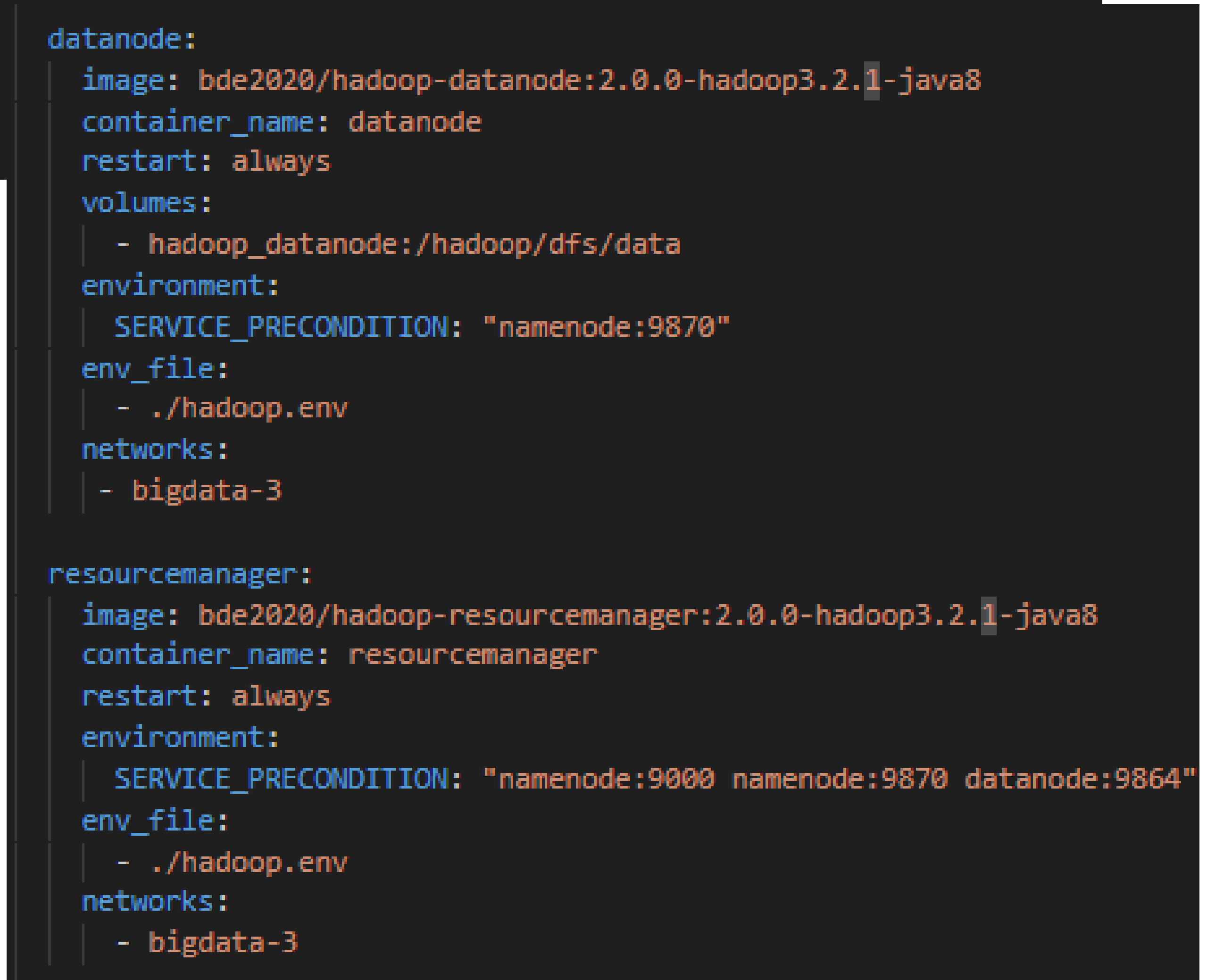


```json
{
  "window_start": "2016-02-12T17:00:00.000+01:00",
  "window_end": "2016-02-12T18:00:00.000+01:00",
  "accident_count": 4,
  "Visibility_mi": 3.0,
  "Wind_Speed_mph": 12.7,
  "Temperature_F": 23.0,
  "Humidity_percent": 80.0,
  "Pressure_in": 30.03,
  "Distance_mi": 1.67,
  "Precipitation_in": 0.0,
  "Start_Lat": 40.109928,
  "Start_Lng": -82.978203,
  "Junction": 1.0,
  "Traffic_Signal": 0.0,
  "Crossing": 0.0,
  "Weather_Condition": "Light Snow",
  "Wind_Direction": "West",
  "Sunrise_Sunset": "Day",
  "Weather_ConditionIndexed": 8.0,
  "Wind_DirectionIndexed": 12.0,
  "Sunrise_SunsetIndexed": 0.0,
  "features": {
    "type": 1,
    "values": [3.0, 12.7, 23.0, 80.0, 30.03, 1.67, 0.0, 40.109928, -82.978203, 1.0, 0.0, 0.0, 8.0, 12.0, 0.0]
  },
  "rawPrediction": {
    "type": 1,
    "values": [-1.077946733725134, 1.077946733725134]
  },
  "probability": {
    "type": 1,
    "values": [0.10378178367571056, 0.8962182163242894]
  },
  "prediction": 1.0
}
```

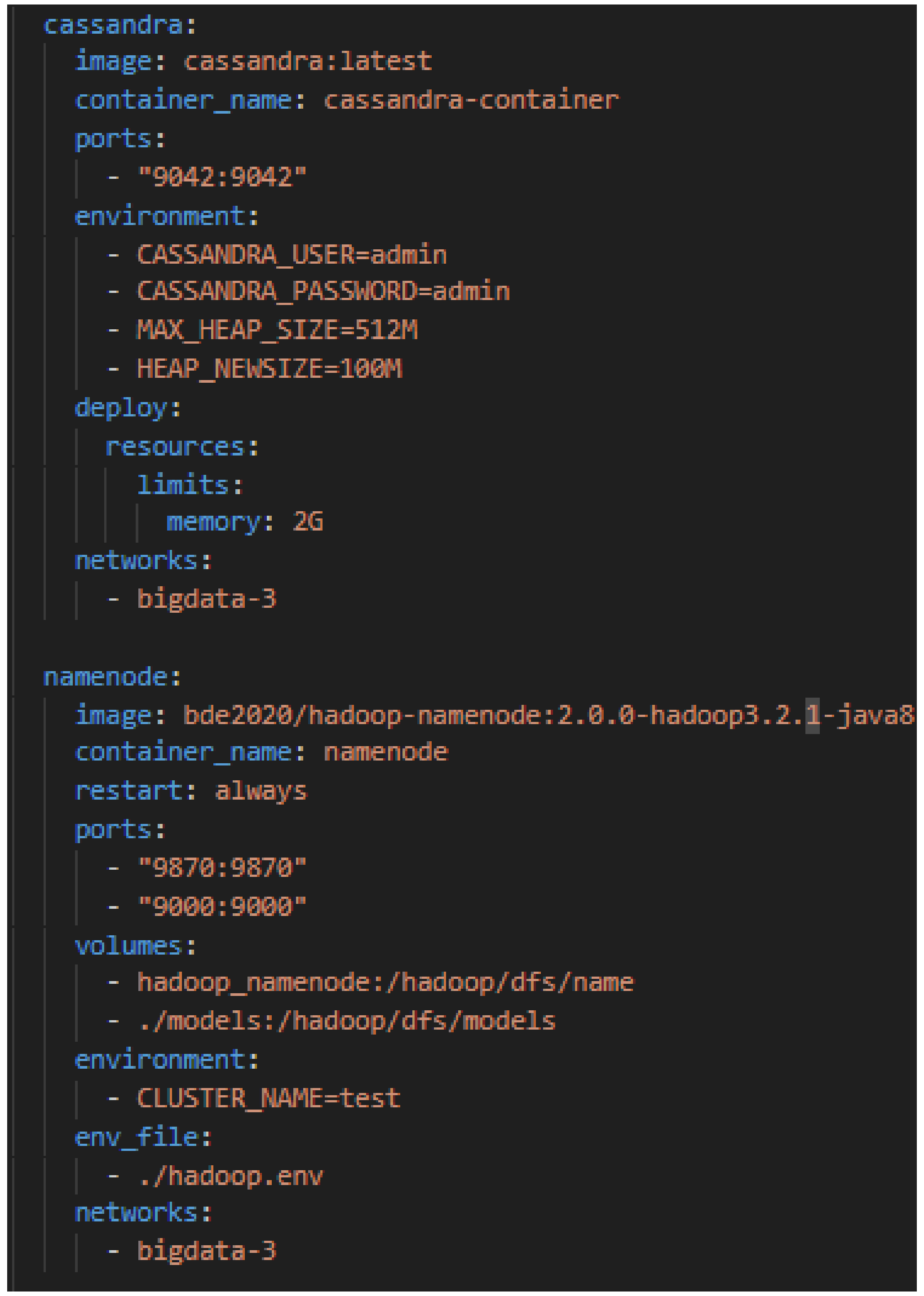
Docker compose



```yaml
services:
  kafka:
    image: apache/kafka
    container_name: kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_NODE_ID: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:29092,CONTROLLER://0.0.0.0:9093,PLAINTEXT_HOST://0.0.0.0:9092
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@localhost:9093
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
    networks:
      - bigdata-3
```

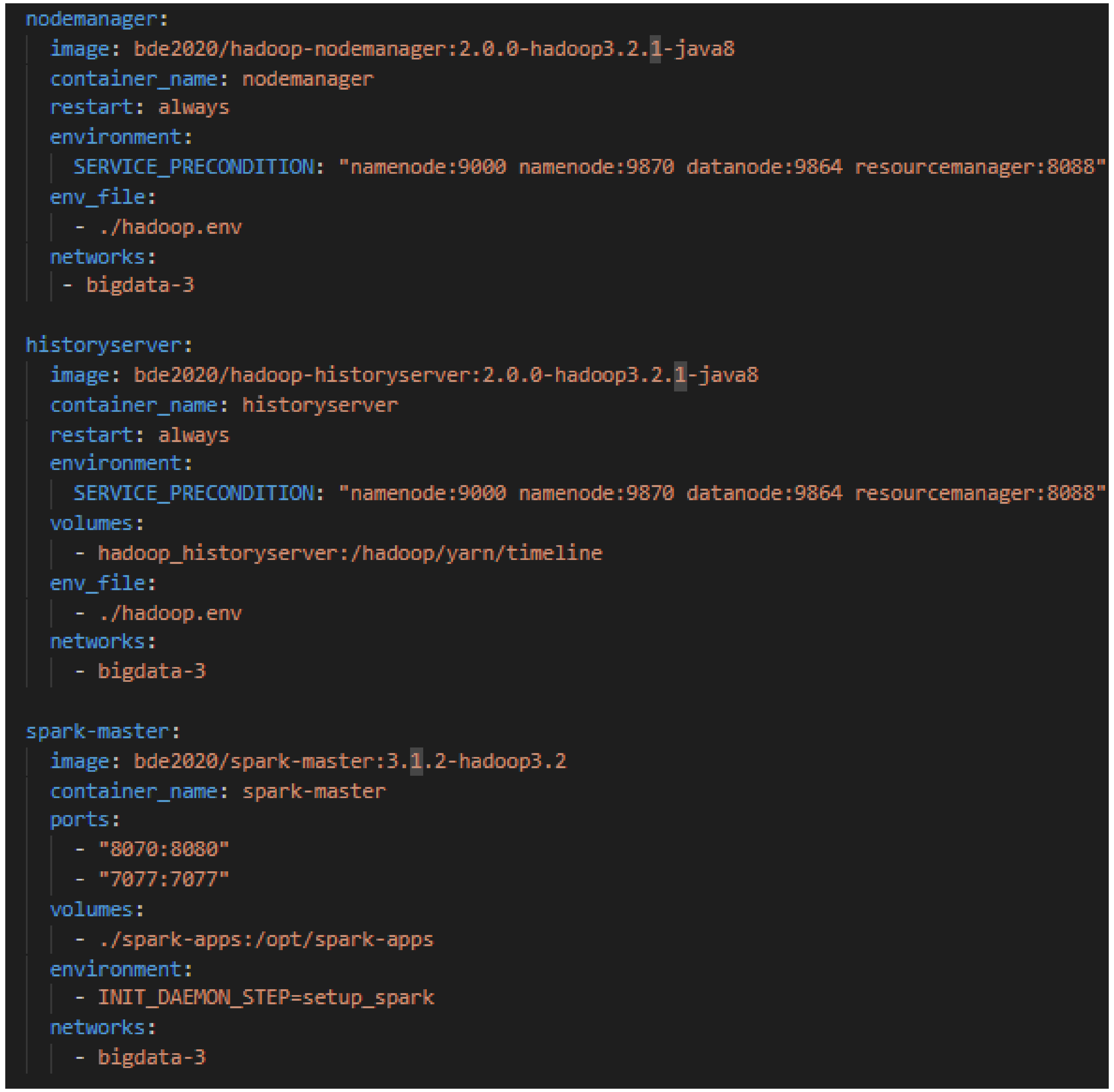


```yaml
  datanode:
    image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
    container_name: datanode
    restart: always
    volumes:
      - hadoop_datanode:/hadoop/dfs/data
    environment:
      SERVICE_PRECONDITION: "namenode:9870"
    env_file:
      - ./hadoop.env
    networks:
     - bigdata-3

  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-3
```

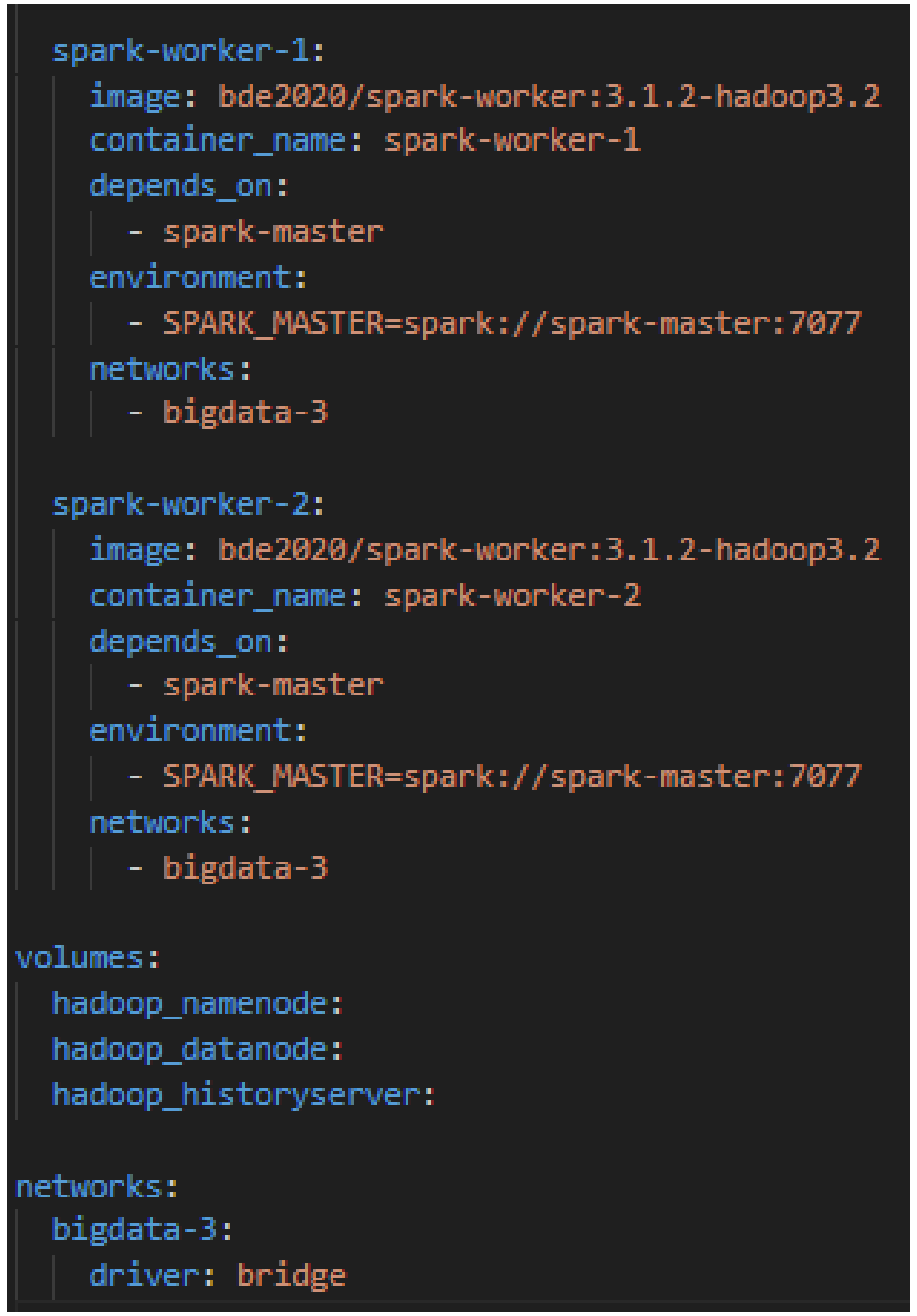


```yaml
  cassandra:
    image: cassandra:latest
    container_name: cassandra-container
    ports:
      - "9042:9042"
    environment:
      - CASSANDRA_USER=admin
      - CASSANDRA_PASSWORD=admin
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    deploy:
      resources:
        limits:
          memory: 2G
    networks:
      - bigdata-3

  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
      - ./models:/hadoop/dfs/models
    environment:
      - CLUSTER_NAME=test
    env_file:
      - ./hadoop.env
    networks:
      - bigdata-3
```

Docker compose

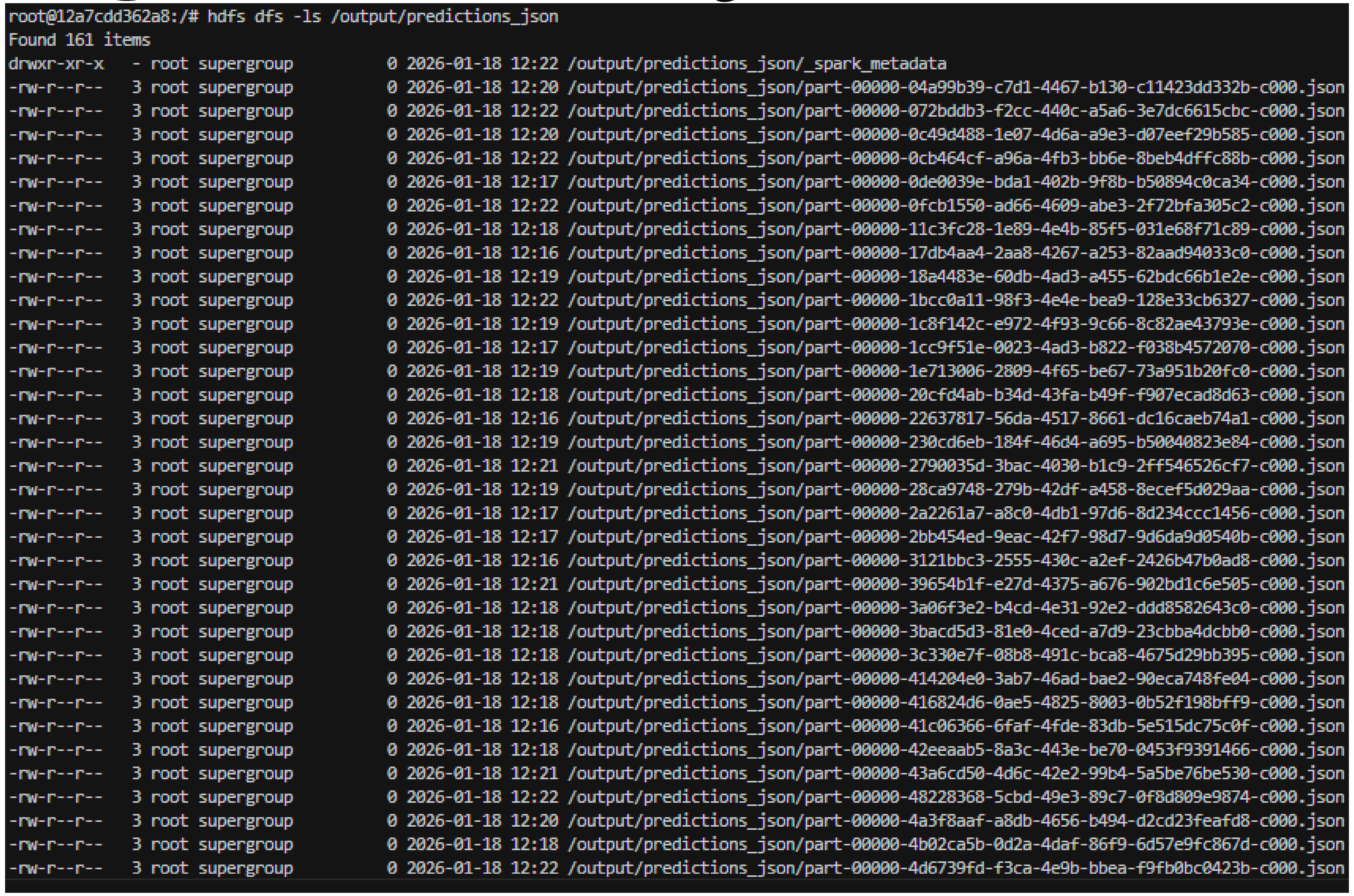


```yaml
nodemanager:
  image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8
  container_name: nodemanager
  restart: always
  environment:
    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
  env_file:
    - ./hadoop.env
  networks:
   - bigdata-3

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-3

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-3
```



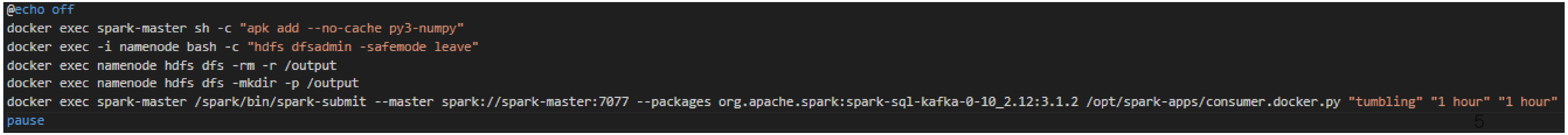
```yaml
  spark-worker-1:
    image: bde2020/spark-worker:3.1.2-hadoop3.2
    container_name: spark-worker-1
    depends_on:
      - spark-master
    environment:
      - SPARK_MASTER=spark://spark-master:7077
    networks:
      - bigdata-3

  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-3

volumes:
  hadoop_namenode:
  hadoop_datanode:
  hadoop_historyserver:

networks:
  bigdata-3:
    driver: bridge
```

Izgenerisani fajlovi



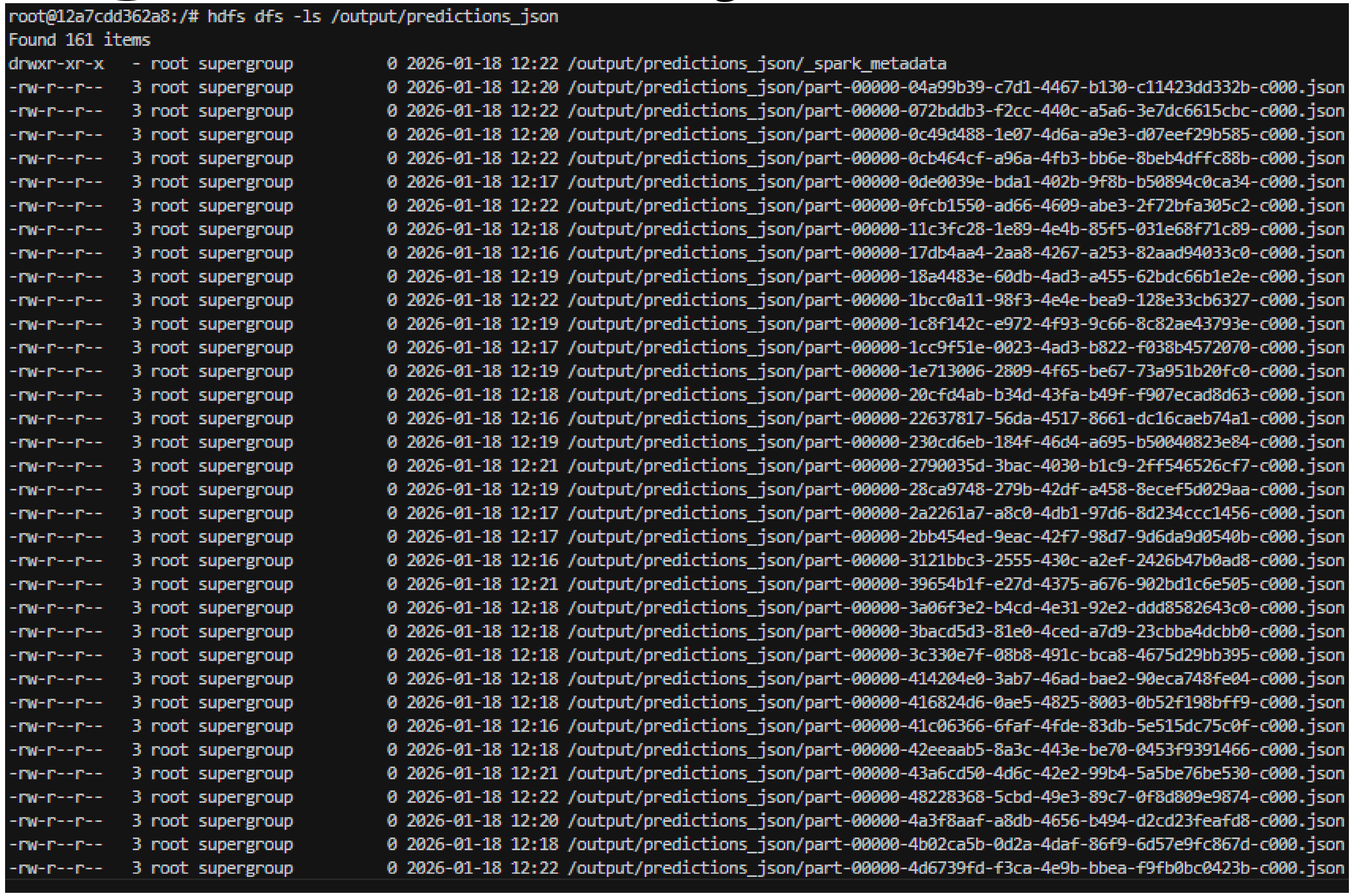
```
root@12a7cdd362a8:/# hdfs dfs -ls /output/predictions_json
Found 161 items
drwxr-xr-x   - root supergroup          0 2026-01-18 12:22 /output/predictions_json/_spark_metadata
-rw-r--r--   3 root supergroup          0 2026-01-18 12:20 /output/predictions_json/part-00000-04a99b39-c7d1-4467-b130-c11423dd332b-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-072bddb3-f2cc-440c-a5a6-3e7dc6615cbc-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:20 /output/predictions_json/part-00000-0c49d488-1e07-4d6a-a9e3-d07eef29b585-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-0cb464cf-a96a-4fb3-bb6e-8beb4dffc88b-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-0de0039e-bda1-402b-9f8b-b50894c0ca34-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-0fcb1550-ad66-4609-abe3-2f72bfa305c2-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-11c3fc28-1e89-4e4b-85f5-031e68f71c89-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-17db4aa4-2aa8-4267-a253-82aad94033c0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-18a4483e-60db-4ad3-a455-62bdc66b1e2e-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-1bcc0a11-98f3-4e4e-bea9-128e33cb6327-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-1c8f142c-e972-4f93-9c66-8c82ae43793e-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-1cc9f51e-0023-4ad3-b822-f038b4572070-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-1e713006-2809-4f65-be67-73a951b20fc0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-20cfd4ab-b34d-43fa-b49f-f907ecad8d63-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-22637817-56da-4517-8661-dc16caeb74a1-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-230cd6eb-184f-46d4-a695-b50040823e84-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:21 /output/predictions_json/part-00000-2790035d-3bac-4030-b1c9-2ff546526cf7-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-28ca9748-279b-42df-a458-8ecef5d029aa-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-2a2261a7-a8c0-4db1-97d6-8d234ccc1456-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-2bb454ed-9eac-42f7-98d7-9d6da9d0540b-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-3121bbc3-2555-430c-a2ef-2426b47b0ad8-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:21 /output/predictions_json/part-00000-39654b1f-e27d-4375-a676-902bd1c6e505-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-3a06f3e2-b4cd-4e31-92e2-ddd8582643c0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-3bacd5d3-81e0-4ced-a7d9-23cbba4dcbb0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-3c330e7f-08b8-491c-bca8-4675d29bb395-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-414204e0-3ab7-46ad-bae2-90eca748fe04-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-416824d6-0ae5-4825-8003-0b52f198bff9-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-41c06366-6faf-4fde-83db-5e515dc75c0f-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-42eeaab5-8a3c-443e-be70-0453f9391466-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:21 /output/predictions_json/part-00000-43a6cd50-4d6c-42e2-99b4-5a5be76be530-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-48228368-5cbd-49e3-89c7-0f8d809e9874-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:20 /output/predictions_json/part-00000-4a3f8aaf-a8db-4656-b494-d2cd23feafd8-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-4b02ca5b-0d2a-4daf-86f9-6d57e9fc867d-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-4d6739fd-f3ca-4e9b-bbea-f9fb0bc0423b-c000.json
```

run. bat



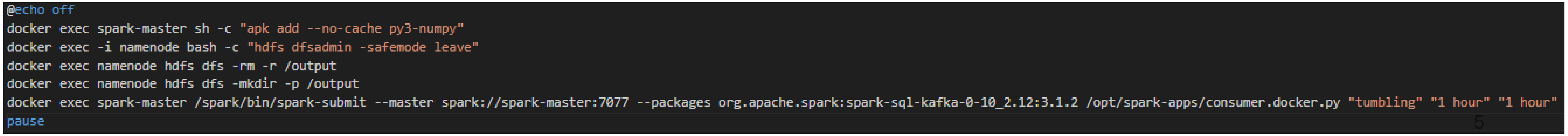
```
@echo off
docker exec spark-master sh -c "apk add --no-cache py3-numpy"
docker exec -i namenode bash -c "hdfs dfsadmin -safemode leave"
docker exec namenode hdfs dfs -rm -r /output
docker exec namenode hdfs dfs -mkdir -p /output
docker exec spark-master /spark/bin/spark-submit --master spark://spark-master:7077 --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 /opt/spark-apps/consumer.docker.py "tumbling" "1 hour" "1 hour"
pause
```

Izgenerisani fajlovi



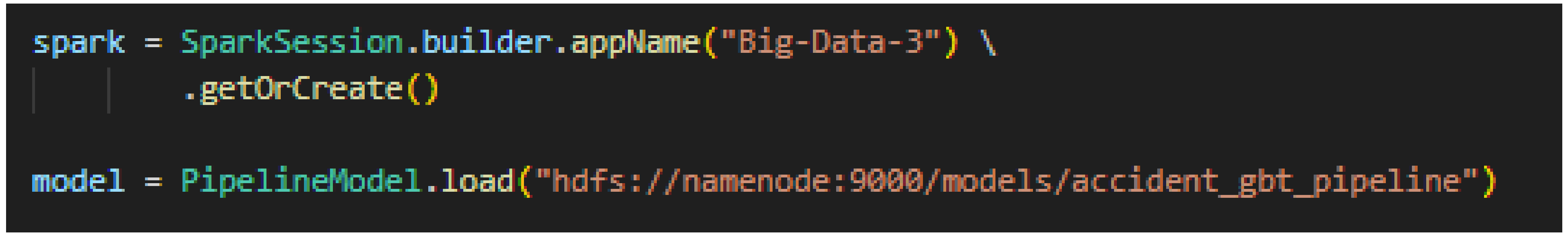
```
root@12a7cdd362a8:/# hdfs dfs -ls /output/predictions_json
Found 161 items
drwxr-xr-x   - root supergroup          0 2026-01-18 12:22 /output/predictions_json/_spark_metadata
-rw-r--r--   3 root supergroup          0 2026-01-18 12:20 /output/predictions_json/part-00000-04a99b39-c7d1-4467-b130-c11423dd332b-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-072bddb3-f2cc-440c-a5a6-3e7dc6615cbc-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:20 /output/predictions_json/part-00000-0c49d488-1e07-4d6a-a9e3-d07eef29b585-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-0cb464cf-a96a-4fb3-bb6e-8beb4dffc88b-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-0de0039e-bda1-402b-9f8b-b50894c0ca34-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-0fcb1550-ad66-4609-abe3-2f72bfa305c2-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-11c3fc28-1e89-4e4b-85f5-031e68f71c89-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-17db4aa4-2aa8-4267-a253-82aad94033c0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-18a4483e-60db-4ad3-a455-62bdc66b1e2e-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-1bcc0a11-98f3-4e4e-bea9-128e33cb6327-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-1c8f142c-e972-4f93-9c66-8c82ae43793e-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-1cc9f51e-0023-4ad3-b822-f038b4572070-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-1e713006-2809-4f65-be67-73a951b20fc0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-20cfd4ab-b34d-43fa-b49f-f907ecad8d63-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-22637817-56da-4517-8661-dc16caeb74a1-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-230cd6eb-184f-46d4-a695-b50040823e84-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:21 /output/predictions_json/part-00000-2790035d-3bac-4030-b1c9-2ff546526cf7-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:19 /output/predictions_json/part-00000-28ca9748-279b-42df-a458-8ecef5d029aa-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-2a2261a7-a8c0-4db1-97d6-8d234ccc1456-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:17 /output/predictions_json/part-00000-2bb454ed-9eac-42f7-98d7-9d6da9d0540b-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-3121bbc3-2555-430c-a2ef-2426b47b0ad8-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:21 /output/predictions_json/part-00000-39654b1f-e27d-4375-a676-902bd1c6e505-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-3a06f3e2-b4cd-4e31-92e2-ddd8582643c0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-3bacd5d3-81e0-4ced-a7d9-23cbba4dcbb0-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-3c330e7f-08b8-491c-bca8-4675d29bb395-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-414204e0-3ab7-46ad-bae2-90eca748fe04-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-416824d6-0ae5-4825-8003-0b52f198bff9-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:16 /output/predictions_json/part-00000-41c06366-6faf-4fde-83db-5e515dc75c0f-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-42eeaab5-8a3c-443e-be70-0453f9391466-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:21 /output/predictions_json/part-00000-43a6cd50-4d6c-42e2-99b4-5a5be76be530-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-48228368-5cbd-49e3-89c7-0f8d809e9874-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:20 /output/predictions_json/part-00000-4a3f8aaf-a8db-4656-b494-d2cd23feafd8-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:18 /output/predictions_json/part-00000-4b02ca5b-0d2a-4daf-86f9-6d57e9fc867d-c000.json
-rw-r--r--   3 root supergroup          0 2026-01-18 12:22 /output/predictions_json/part-00000-4d6739fd-f3ca-4e9b-bbea-f9fb0bc0423b-c000.json
```

run. bat

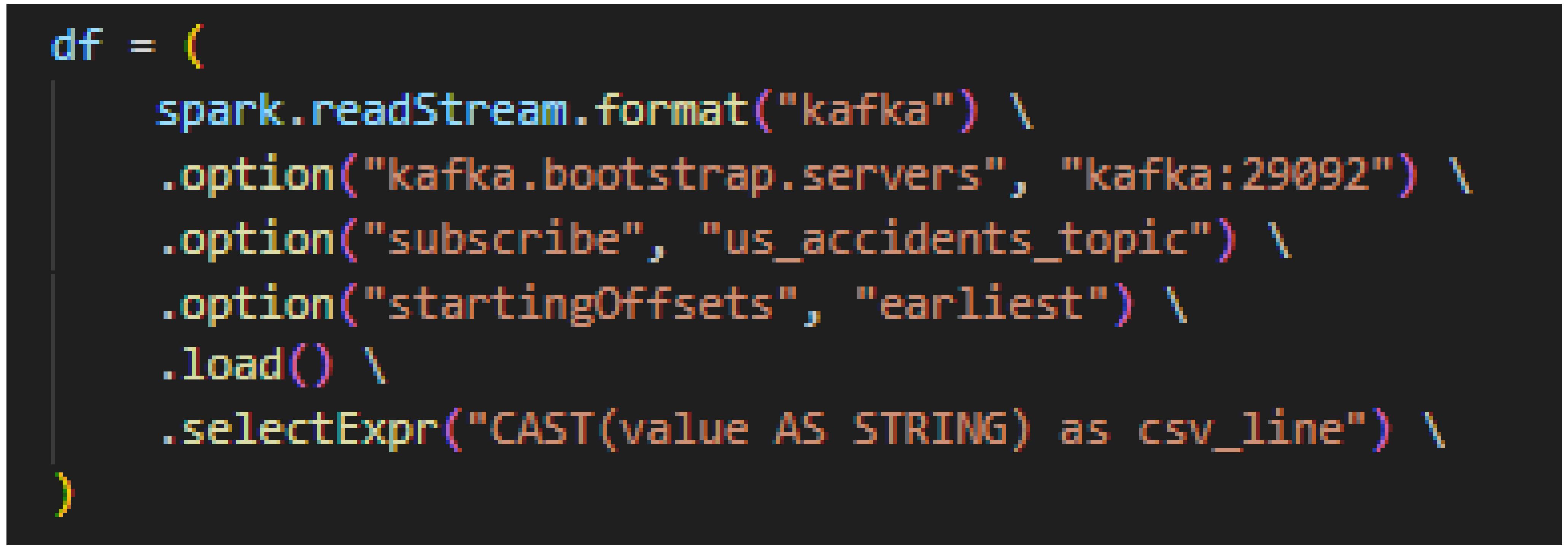


```
@echo off
docker exec spark-master sh -c "apk add --no-cache py3-numpy"
docker exec -i namenode bash -c "hdfs dfsadmin -safemode leave"
docker exec namenode hdfs dfs -rm -r /output
docker exec namenode hdfs dfs -mkdir -p /output
docker exec spark-master /spark/bin/spark-submit --master spark://spark-master:7077 --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 /opt/spark-apps/consumer.docker.py "tumbling" "1 hour" "1 hour"
pause
```

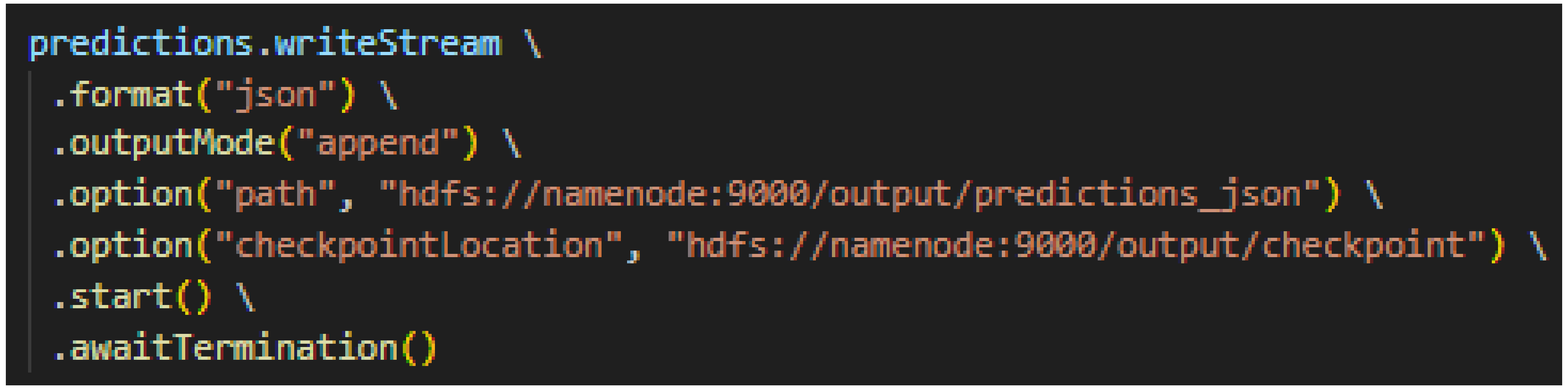
consumer.docker.py



```
spark = SparkSession.builder.appName("Big-Data-3") \
        .getOrCreate()

model = PipelineModel.load("hdfs://namenode:9000/models/accident_gbt_pipeline")
```

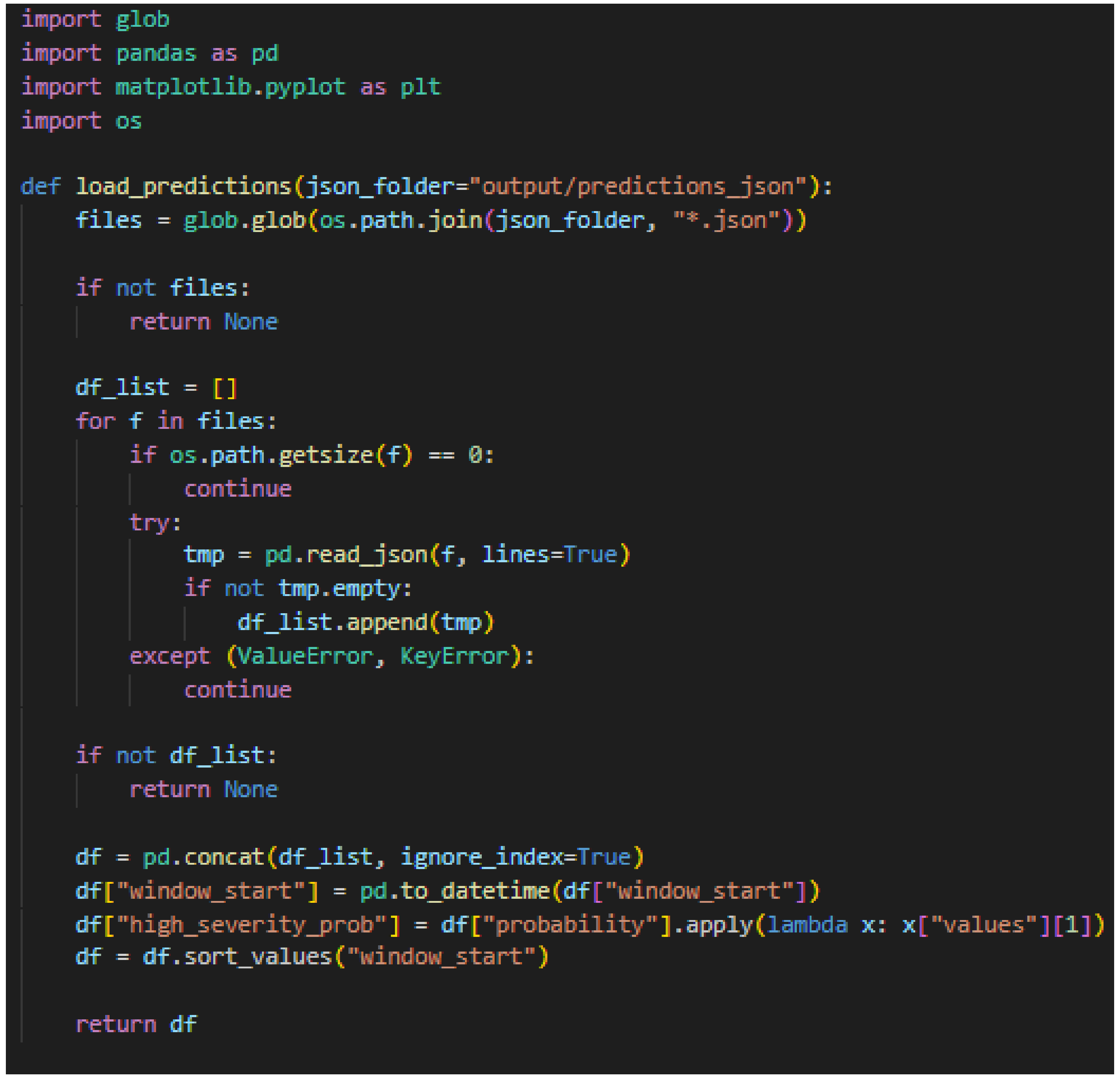


```
df = (
    spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:29092") \
    .option("subscribe", "us_accidents_topic") \
    .option("startingOffsets", "earliest") \
    .load() \
    .selectExpr("CAST(value AS STRING) as csv_line") \
)
```

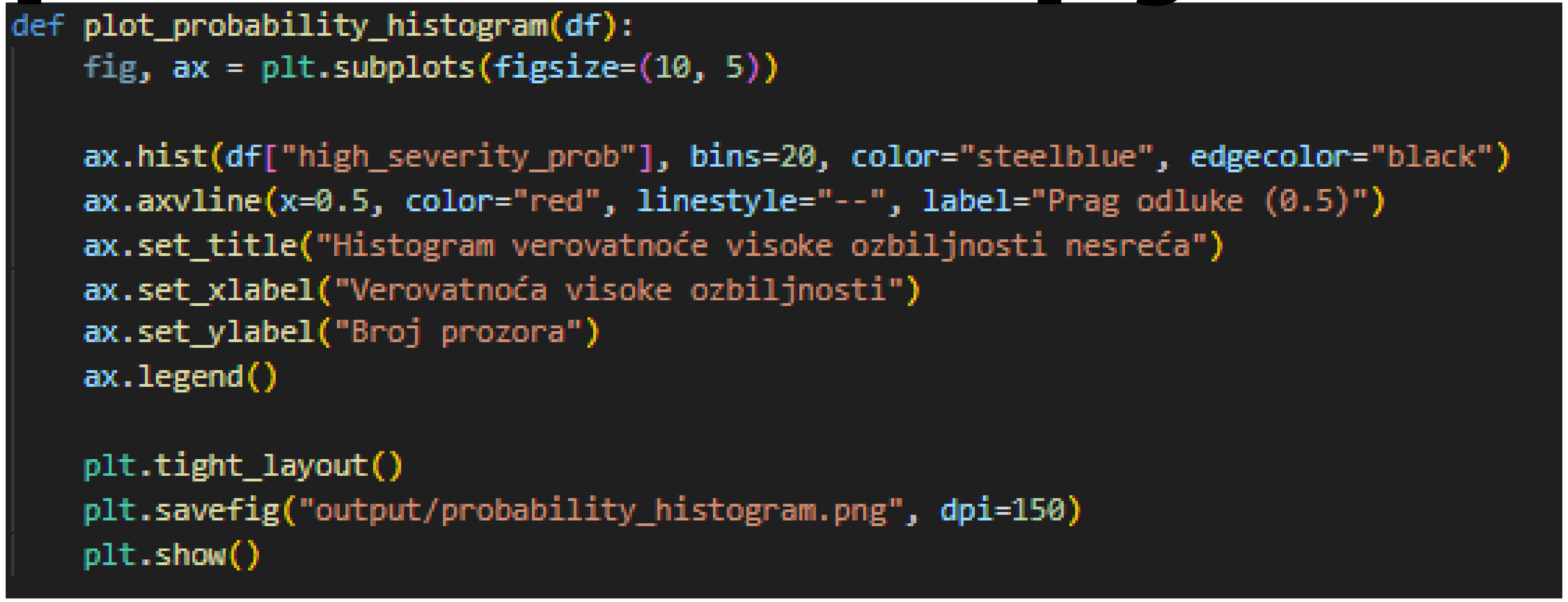


```
predictions.writeStream \
 .format("json") \
 .outputMode("append") \
 .option("path", "hdfs://namenode:9000/output/predictions_json") \
 .option("checkpointLocation", "hdfs://namenode:9000/output/checkpoint") \
 .start() \
 .awaitTermination()
```

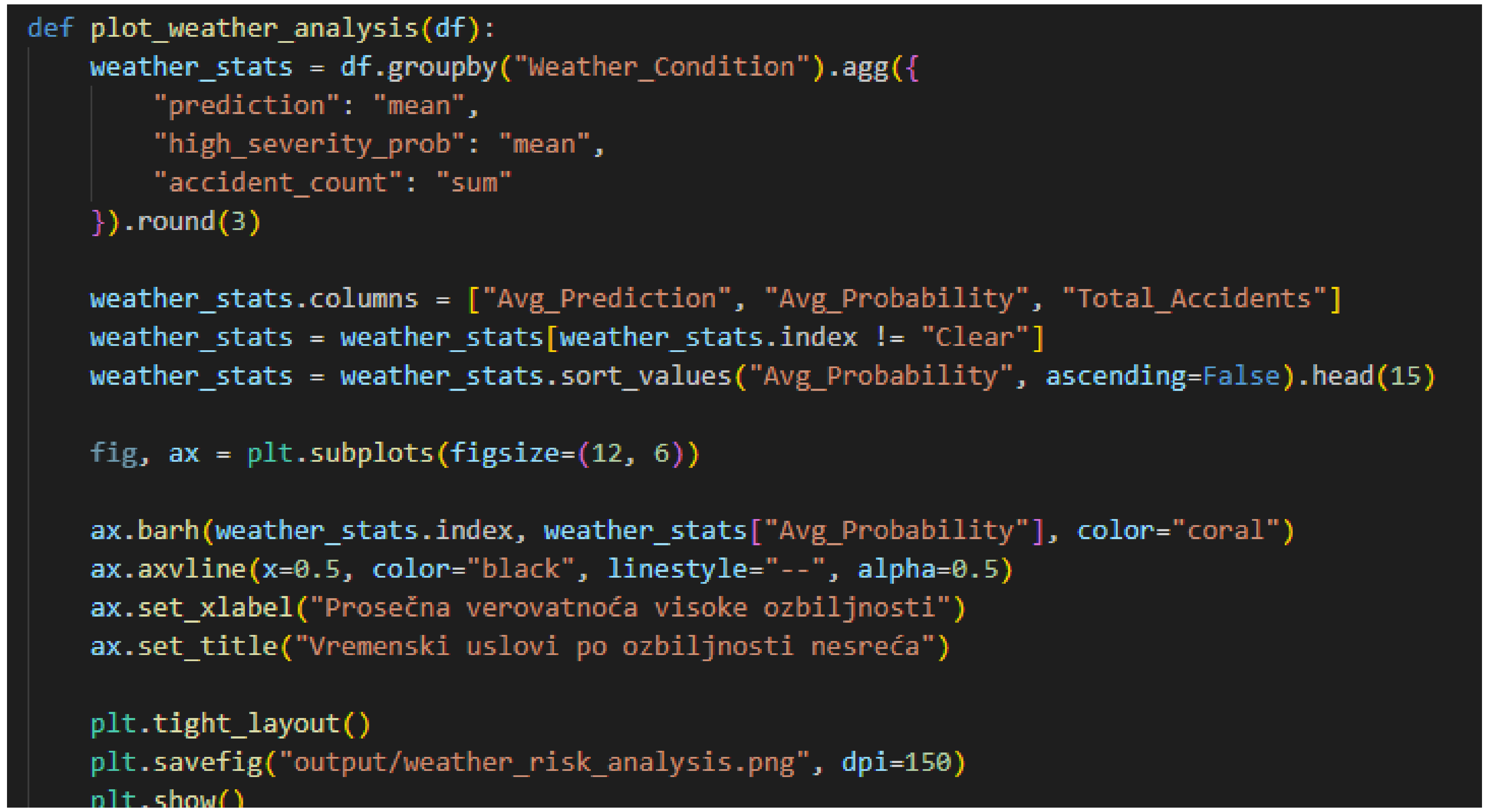
visualize_predictions.py



```python
import glob
import pandas as pd
import matplotlib.pyplot as plt
import os

def load_predictions(json_folder="output/predictions_json"):
    files = glob.glob(os.path.join(json_folder, "*.json"))

    if not files:
        return None

    df_list = []
    for f in files:
        if os.path.getsize(f) == 0:
            continue
        try:
            tmp = pd.read_json(f, lines=True)
            if not tmp.empty:
                df_list.append(tmp)
        except (ValueError, KeyError):
            continue

    if not df_list:
        return None

    df = pd.concat(df_list, ignore_index=True)
    df["window_start"] = pd.to_datetime(df["window_start"])
    df["high_severity_prob"] = df["probability"].apply(lambda x: x["values"][1])
    df = df.sort_values("window_start")

    return df
```

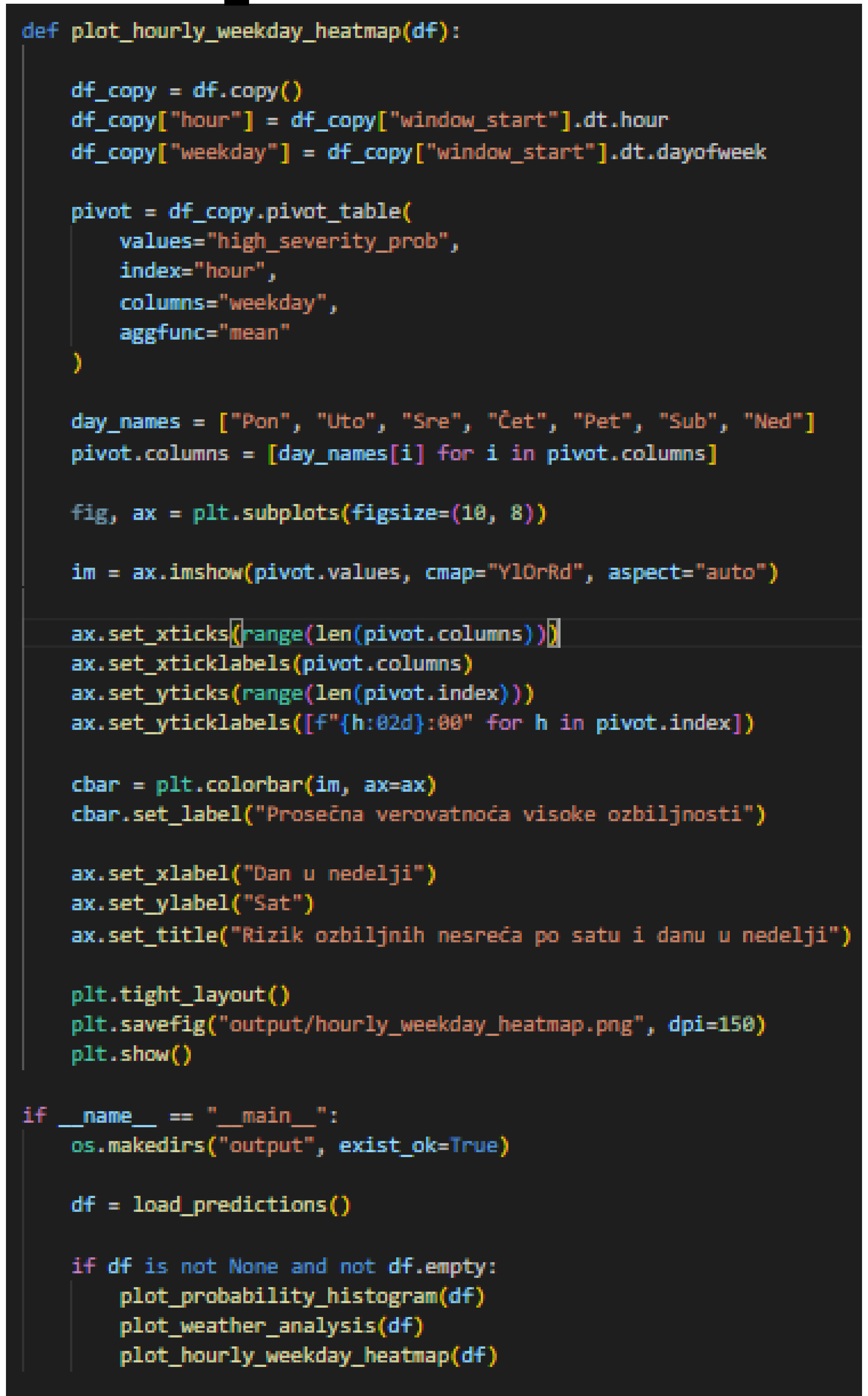


```python
def plot_probability_histogram(df):
    fig, ax = plt.subplots(figsize=(10, 5))

    ax.hist(df["high_severity_prob"], bins=20, color="steelblue", edgecolor="black")
    ax.axvline(x=0.5, color="red", linestyle="--", label="Prag odluke (0.5)")
    ax.set_title("Histogram verovatnoće visoke ozbiljnosti nesreća")
    ax.set_xlabel("Verovatnoća visoke ozbiljnosti")
    ax.set_ylabel("Broj prozora")
    ax.legend()

    plt.tight_layout()
    plt.savefig("output/probability_histogram.png", dpi=150)
    plt.show()
```



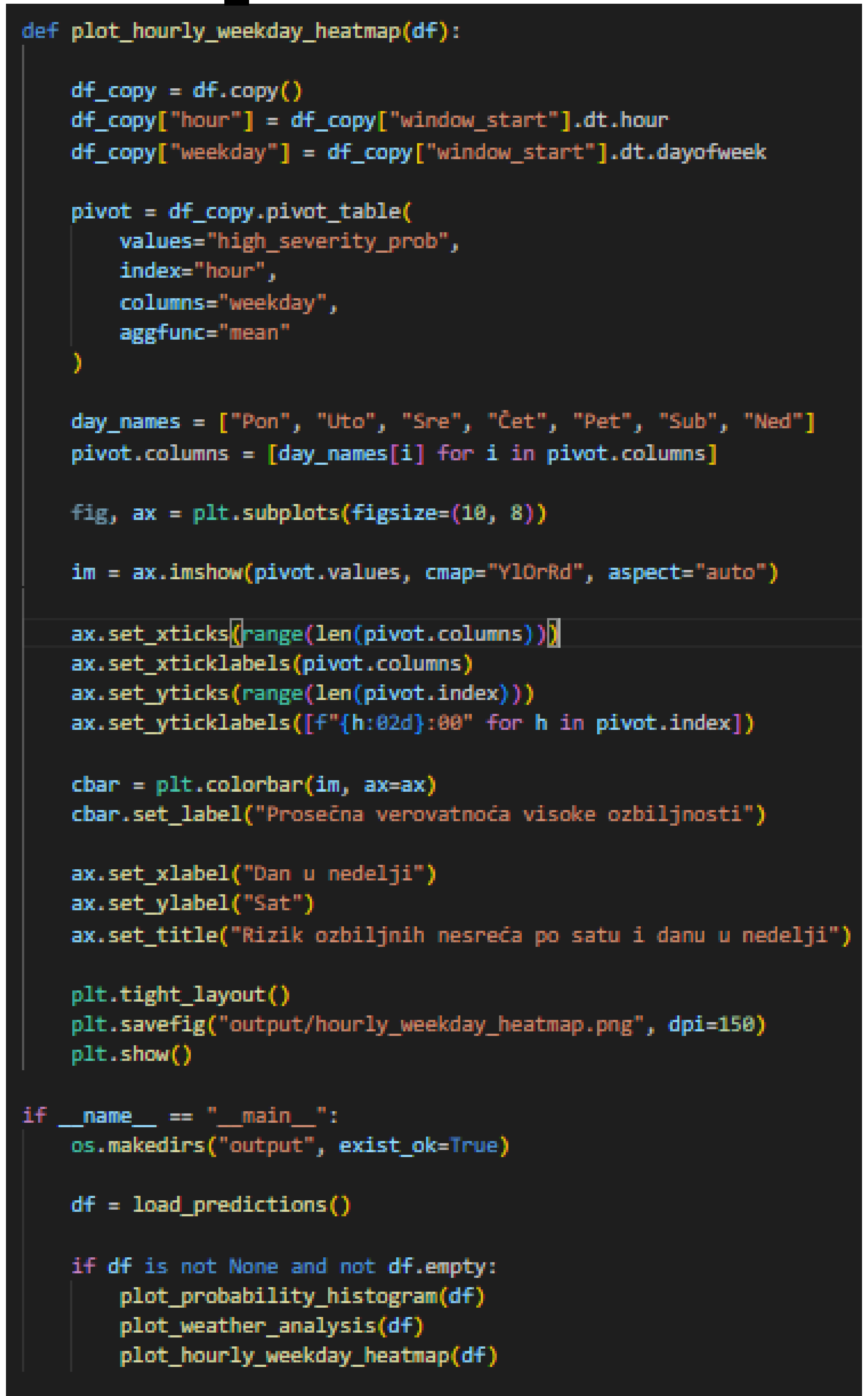
```python
def plot_weather_analysis(df):
    weather_stats = df.groupby("Weather_Condition").agg({
        "prediction": "mean",
        "high_severity_prob": "mean",
        "accident_count": "sum"
    }).round(3)

    weather_stats.columns = ["Avg_Prediction", "Avg_Probability", "Total_Accidents"]
    weather_stats = weather_stats[weather_stats.index != "Clear"]
    weather_stats = weather_stats.sort_values("Avg_Probability", ascending=False).head(15)

    fig, ax = plt.subplots(figsize=(12, 6))

    ax.barh(weather_stats.index, weather_stats["Avg_Probability"], color="coral")
    ax.axvline(x=0.5, color="black", linestyle="--", alpha=0.5)
    ax.set_xlabel("Prosečna verovatnoća visoke ozbiljnosti")
    ax.set_title("Vremenski uslovi po ozbiljnosti nesreća")

    plt.tight_layout()
    plt.savefig("output/weather_risk_analysis.png", dpi=150)
    plt.show()
```

visualize_predictions.py

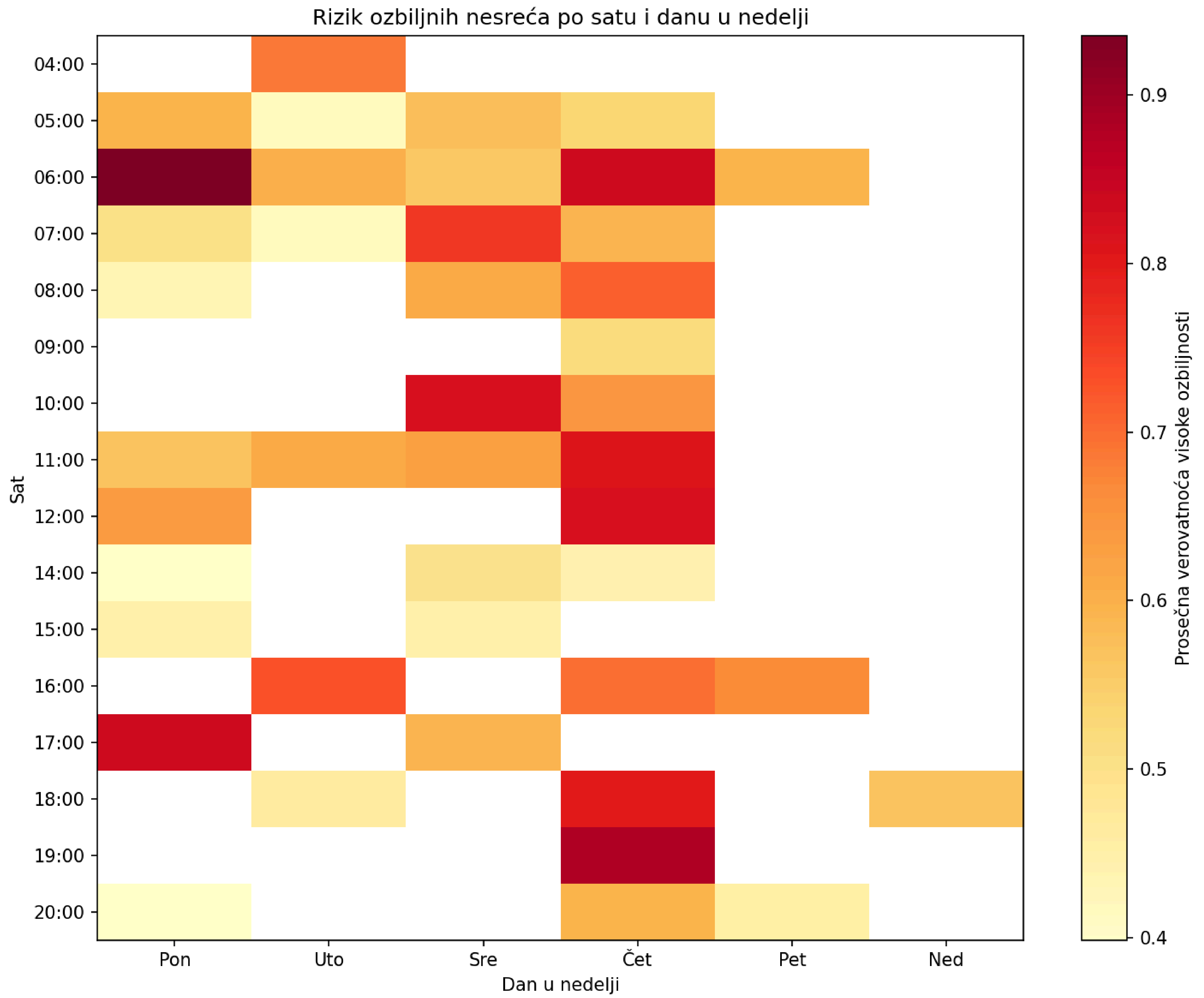


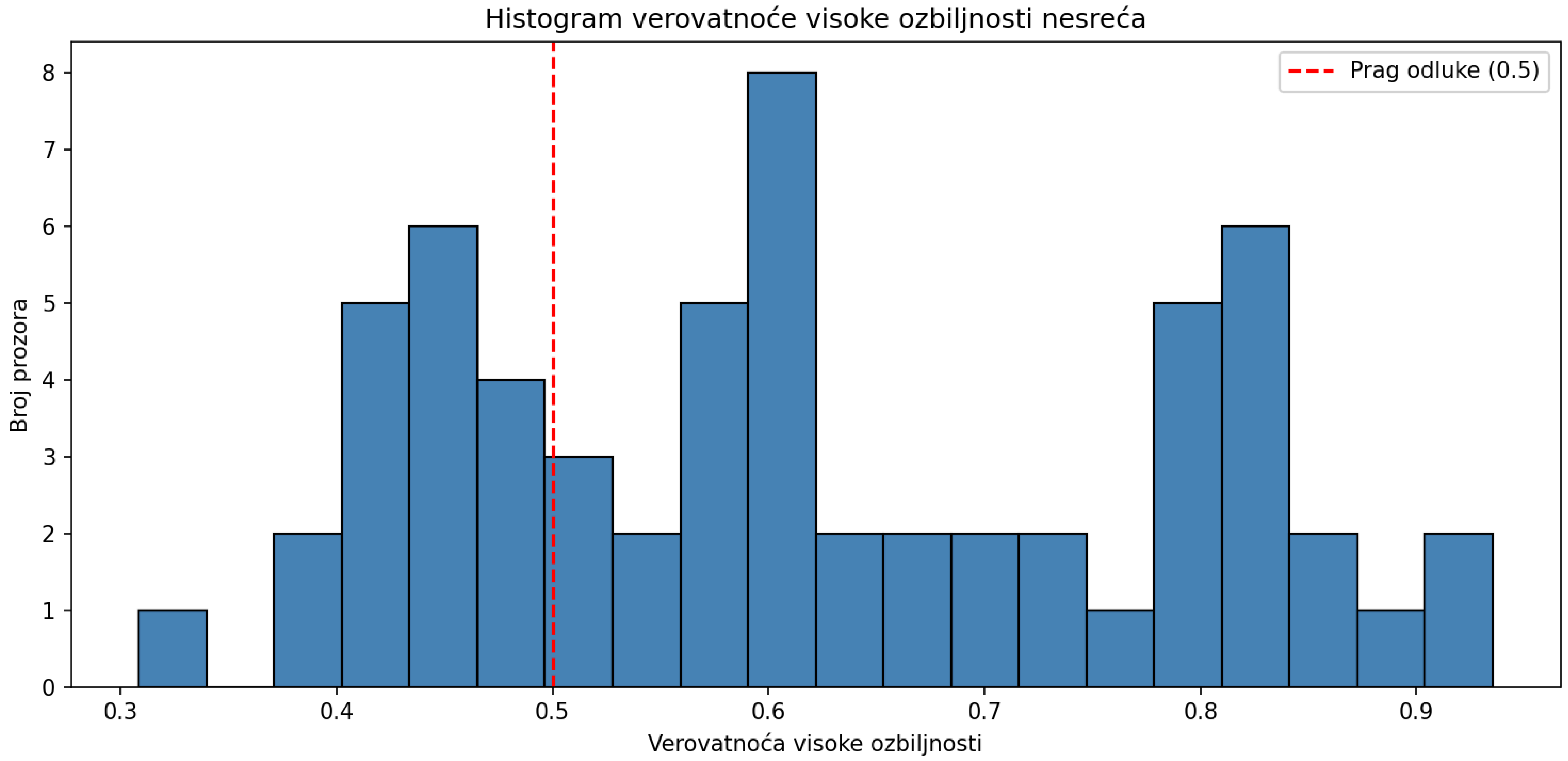
```python
def plot_hourly_weekday_heatmap(df):

    df_copy = df.copy()
    df_copy["hour"] = df_copy["window_start"].dt.hour
    df_copy["weekday"] = df_copy["window_start"].dt.dayofweek

    pivot = df_copy.pivot_table(
        values="high_severity_prob",
        index="hour",
        columns="weekday",
        aggfunc="mean"
    )

    day_names = ["Pon", "Uto", "Sre", "Čet", "Pet", "Sub", "Ned"]
    pivot.columns = [day_names[i] for i in pivot.columns]

    fig, ax = plt.subplots(figsize=(10, 8))

    im = ax.imshow(pivot.values, cmap="YlOrRd", aspect="auto")

    ax.set_xticks(range(len(pivot.columns)))
    ax.set_xticklabels(pivot.columns)
    ax.set_yticks(range(len(pivot.index)))
    ax.set_yticklabels([f"{h:02d}:00" for h in pivot.index])

    cbar = plt.colorbar(im, ax=ax)
    cbar.set_label("Prosečna verovatnoća visoke ozbiljnosti")

    ax.set_xlabel("Dan u nedelji")
    ax.set_ylabel("Sat")
    ax.set_title("Rizik ozbiljnih nesreća po satu i danu u nedelji")

    plt.tight_layout()
    plt.savefig("output/hourly_weekday_heatmap.png", dpi=150)
    plt.show()

if __name__ == "__main__":
    os.makedirs("output", exist_ok=True)

    df = load_predictions()

    if df is not None and not df.empty:
        plot_probability_histogram(df)
        plot_weather_analysis(df)
        plot_hourly_weekday_heatmap(df)
```

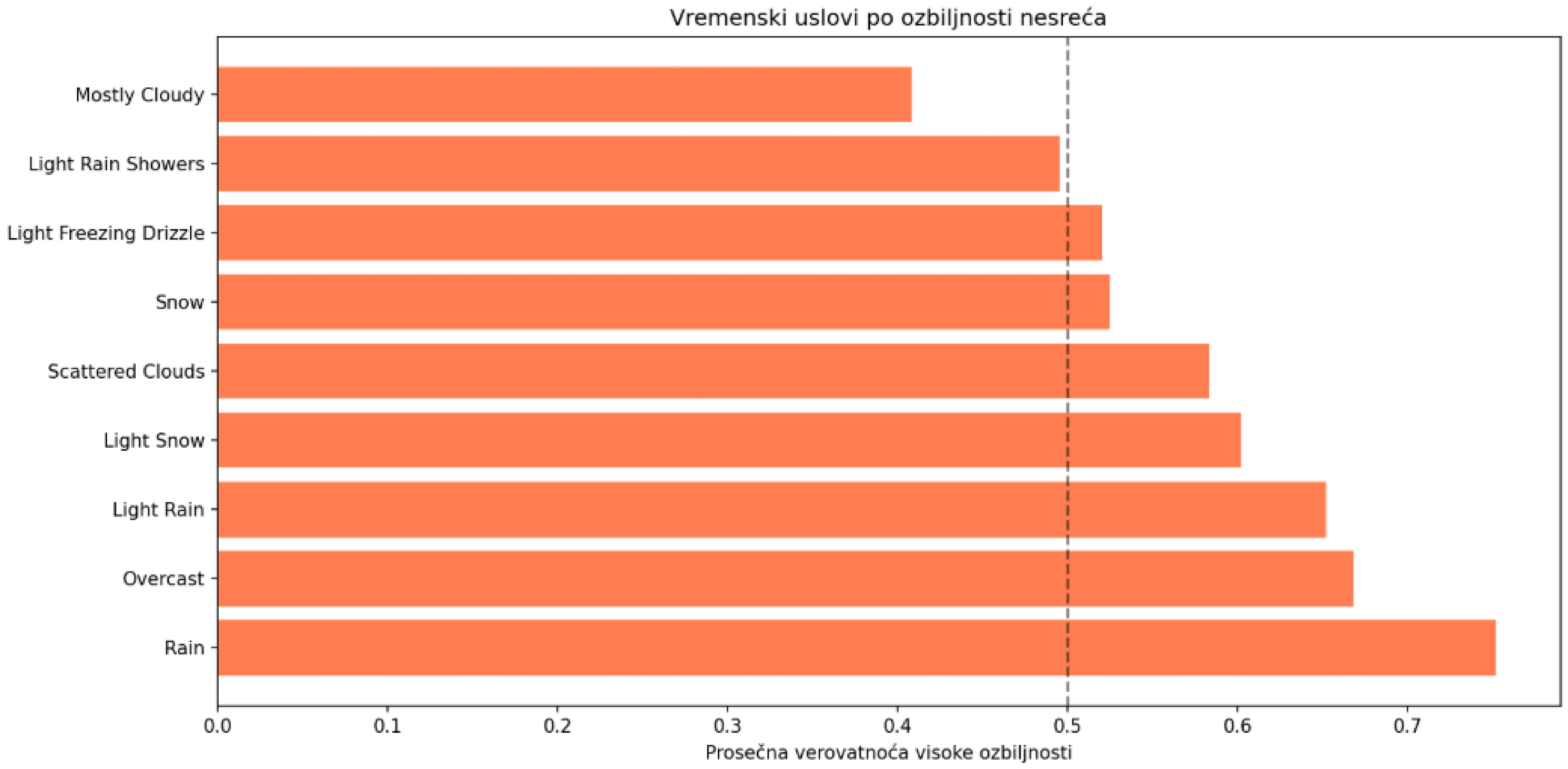
visualize_predictions.py



```python
def plot_hourly_weekday_heatmap(df):

    df_copy = df.copy()
    df_copy["hour"] = df_copy["window_start"].dt.hour
    df_copy["weekday"] = df_copy["window_start"].dt.dayofweek

    pivot = df_copy.pivot_table(
        values="high_severity_prob",
        index="hour",
        columns="weekday",
        aggfunc="mean"
    )

    day_names = ["Pon", "Uto", "Sre", "Čet", "Pet", "Sub", "Ned"]
    pivot.columns = [day_names[i] for i in pivot.columns]

    fig, ax = plt.subplots(figsize=(10, 8))

    im = ax.imshow(pivot.values, cmap="YlOrRd", aspect="auto")

    ax.set_xticks(range(len(pivot.columns)))
    ax.set_xticklabels(pivot.columns)
    ax.set_yticks(range(len(pivot.index)))
    ax.set_yticklabels([f"{h:02d}:00" for h in pivot.index])

    cbar = plt.colorbar(im, ax=ax)
    cbar.set_label("Prosečna verovatnoća visoke ozbiljnosti")

    ax.set_xlabel("Dan u nedelji")
    ax.set_ylabel("Sat")
    ax.set_title("Rizik ozbiljnih nesreća po satu i danu u nedelji")

    plt.tight_layout()
    plt.savefig("output/hourly_weekday_heatmap.png", dpi=150)
    plt.show()

if __name__ == "__main__":
    os.makedirs("output", exist_ok=True)

    df = load_predictions()

    if df is not None and not df.empty:
        plot_probability_histogram(df)
        plot_weather_analysis(df)
        plot_hourly_weekday_heatmap(df)
```


Rizik ozbiljnih nesreća po satu i danu u nedelji
Sat
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
Pon
Uto
Sre
Čet
Pet
Ned
Dan u nedelji
Prosečna verovatnoća visoke ozbiljnosti
0.9
0.8
0.7
0.6
0.5
0.4


Histogram verovatnoće visoke ozbiljnosti nesreća
Prag odluke (0.5)
Broj prozora
Verovatnoća visoke ozbiljnosti
0
1
2
3
4
5
6
7
8
0.3
0.4
0.5
0.6
0.7
0.8
0.9


Vremenski uslovi po ozbiljnosti nesreća
Mostly Cloudy
Light Rain Showers
Light Freezing Drizzle
Snow
Scattered Clouds
Light Snow
Light Rain
Overcast
Rain
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Prosečna verovatnoća visoke ozbiljnosti

Hvala na pažnji!