

西南大学

课程设计报告

课程名称 大规模分布式系统综合实践

开课学期 2024 至 2025 学年 第 1 学期

年 级 2022

专业班级 计算机科学与技术专业(中外合作办学) 4 班

姓 名 张乐之

学 号 222022321102120

任课教师 邱开金

一、课程设计任务与目的

1.1 任务概述

本课程设计要求使用 Spark 框架进行数据预处理、统计分析、机器学习任务与数据可视化。通过对给定数据集的深入分析和处理，学习如何在分布式系统中应用 Spark 进行大规模数据处理与分析。

1.2 设计目的

1. 掌握 Spark 框架的数据预处理和分析方法。
2. 理解机器学习算法在大数据中的应用，并进行有效的模型训练与预测。
3. 使用 Spark 进行数据的可视化分析，帮助理解分析结果和模式。

二、课程设计完成过程

2.1 数据预处理

2.1.1 数据字段分析与中文描述

ID

- 含义: 站点 ID，11 个字符的站点识别码。
- 中文描述: 站点标识符，用于唯一标识气象观测站。

YEAR/MONTH/DAY

- 含义: 日期，8 个字符的年月日，格式为 YYYYMMDD，例如 19860529 表示 1986 年 5 月 29 日。
- 中文描述: 观测日期，格式为“年月日”。

ELEMENT

- 含义: 要素类型，4 个字符表示观测的气象要素类型（如 TMAX、TMIN、PRCP 等）。
- 中文描述: 气象观测要素类型，如最高温度、最低温度或降水量。

DATA VALUE

- 含义: 数据值，5 个字符，表示对应气象要素的观测值（单位需要参考具体 ELEMENT 说明）。
- 中文描述: 观测值，表示气象要素的具体测量值。

M-FLAG

- **含义:** 测量标记, 1 个字符, 用于指示观测值的测量标记 (具体含义需参考 M-FLAG 的定义)。

- **中文描述:** 测量标记, 用于描述观测值的测量状态。

Q-FLAG

- **含义:** 质量标记, 1 个字符, 用于指示观测值的质量状态 (具体含义需参考 Q-FLAG 的定义)。

- **中文描述:** 数据质量标记, 标识观测值是否可能存在问题。

S-FLAG

- **含义:** 数据来源标记, 1 个字符, 用于指示数据的来源 (具体含义需参考 S-FLAG 的定义)。

- **中文描述:** 数据来源标记, 指明数据的来源或处理方式。

OBS-TIME

- **含义:** 观测时间, 4 个字符, 表示观测发生的具体时间 (格式为小时分钟, 如 0700 表示早上 7:00)。

- **中文描述:** 观测时间, 表示气象观测的具体时间点。

2.1.2 数据文件关联分析

基础元数据:

- **ghcnd-stations.txt:** 包含所有气象站的元数据信息 (如站点 ID、经纬度、海拔等), 是数据集的核心站点信息表。

- **ghcnd-countries.txt:** 描述气象站所属的国家, 使用 FIPS 国家代码关联。

- **ghcnd-states.txt:** 描述气象站所在的州或省份, 使用邮政代码关联。

气象数据文件:

- **.dly 文件:** 每个 .dly 文件对应一个气象站, 记录了一个月的每日气象观测数据, 包括日期、要素 (如温度、降水量等) 及标记 (测量标记、质量标记、来源标记等)。

- **by_year 子目录:** 按年份分割的 .dly 文件版本, 便于按年度进行分析。

- **grid** 子目录: 包含 GHCN-Daily 的网格化数据集 (HadGHCND)。

数据参考文档:

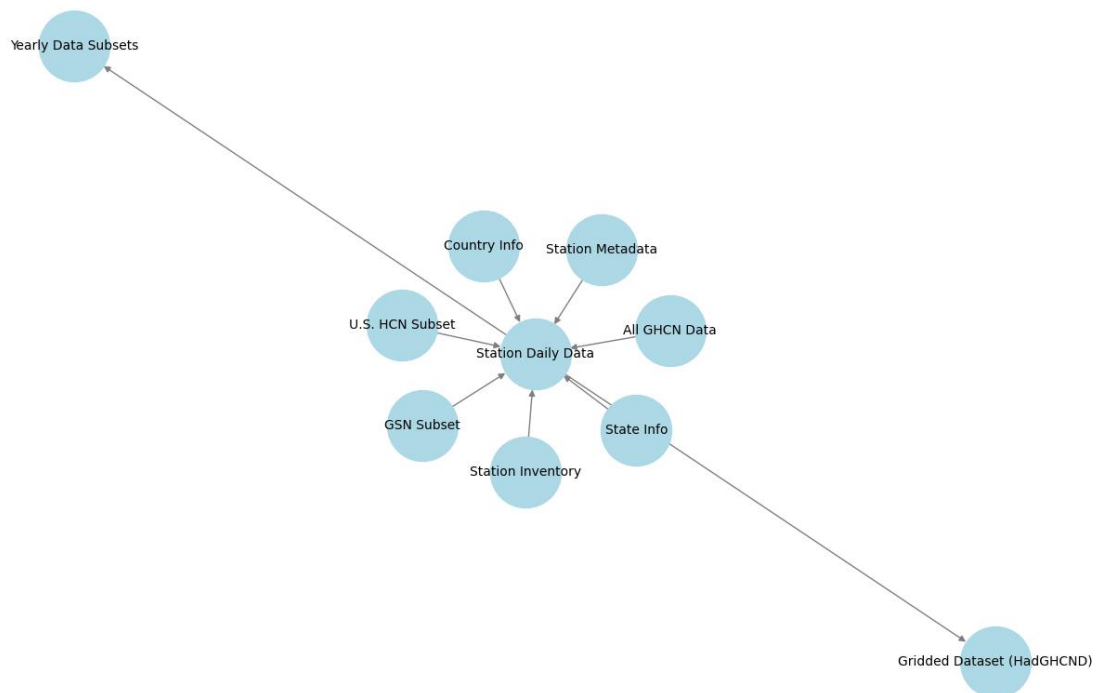
- **papers** 子目录: 包含与 GHCN-Daily 数据集相关的学术文章, 提供背景信息。

- **figures** 子目录: 包含对站点记录的库存和处理过程的图形总结。

其他辅助文件:

- **ghcnd-inventory.txt**: 提供气象站中观测数据的详细记录 (如时间跨度、要素类型等)。
- **mingle-list.txt**: 包含某些额外的数据变量。

详细关系图如下:



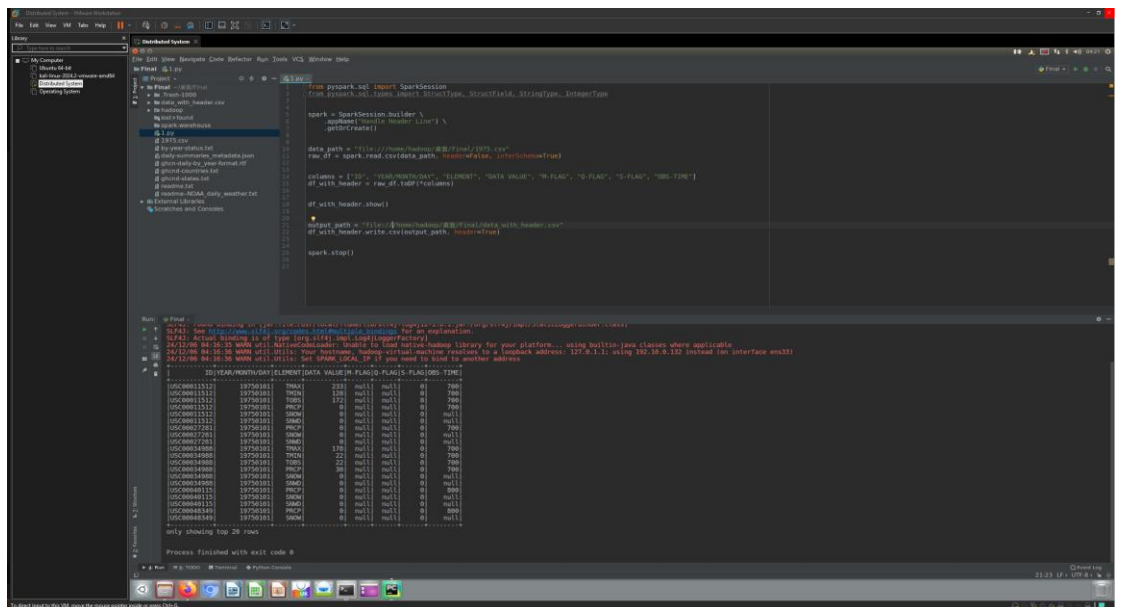
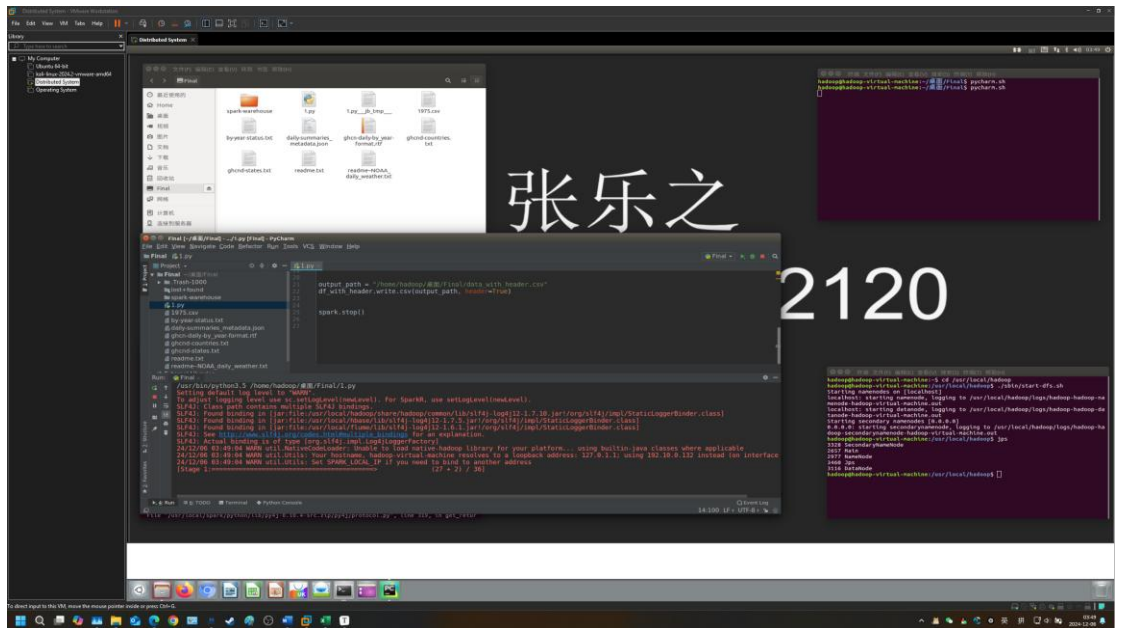
2.1.3 标题行处理

使用 `pyspark` 处理数据集中的标题行:

```
raw_df = spark.read.csv(data_path, header=False, inferSchema=True)
```

```
columns = ["ID", "YEAR/MONTH/DAY", "ELEMENT", "DATA VALUE", "M-FLAG", "Q-FLAG",  
"S-FLAG", "OBS-TIME"]
```

```
df_with_header = raw_df.toDF(*columns)
```



2.1.4 缺失值处理

检查数据集中的空值，并清理空值。

用特定值填充缺失值

```
df_filled = df_with_header.na.fill({
```

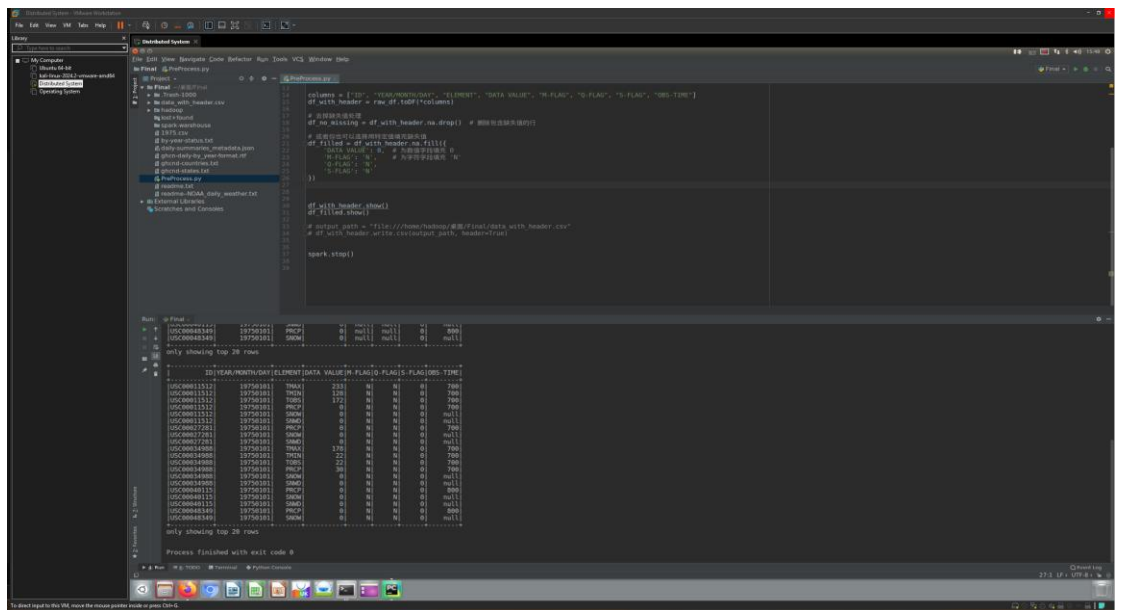
```
    'DATA VALUE': 0, # 为数值字段填充 0
```

```
    'M-FLAG': 'N', # 为字符字段填充 'N'
```

```
    'Q-FLAG': 'N',
```

```
    'S-FLAG': 'N'
```

```
})
```



2.1.5 异常值检查

查找并处理数据中的异常值（如负值、非数值类型等）。

筛选异常值

```
df_invalid = df_with_header.filter((df_with_header["DATA VALUE"] < -200) |
```

```
(df_with_header["DATA VALUE"] > 300))
```

查看异常值

```
print("异常值记录：")
```

```
df_invalid.show()
```

删除包含异常值的行

```
df_cleaned = df_with_header.filter((df_with_header["DATA VALUE"] >= -200) &
```

```
(df_with_header["DATA VALUE"] <= 300))
```

或者处理异常值，例如将异常值替换为固定值

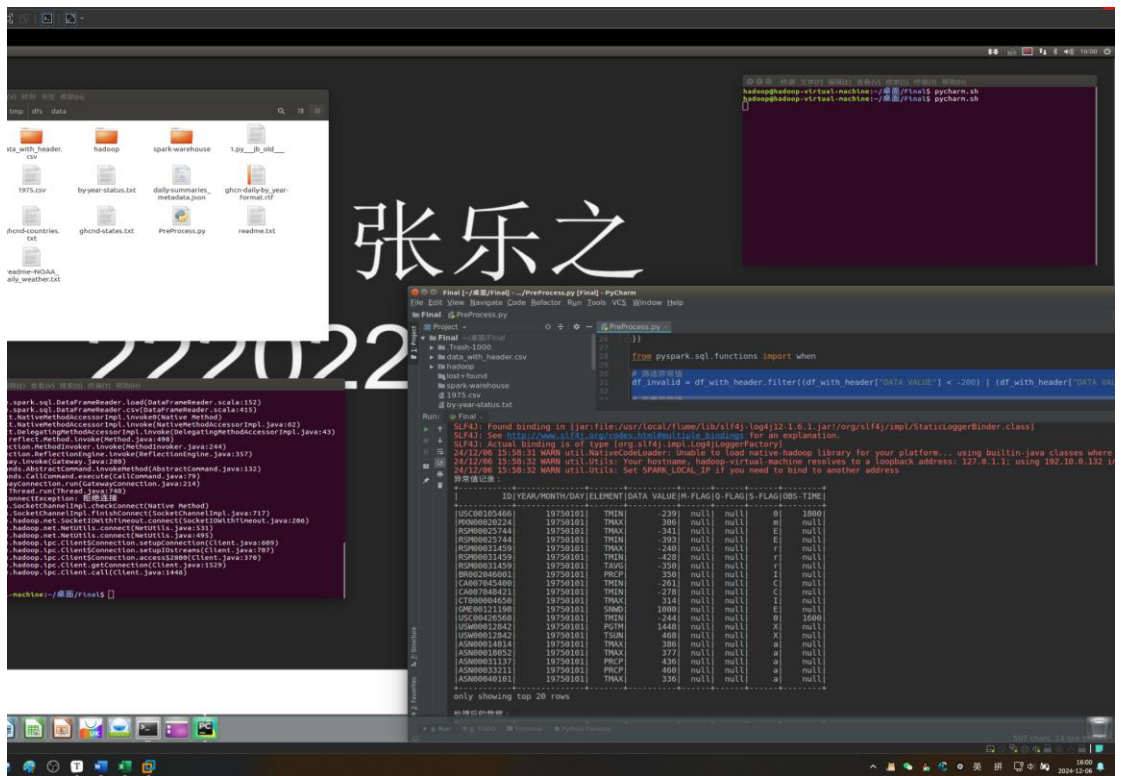
```
df_corrected = df_with_header.withColumn(
```

```
"DATA VALUE",
```

```
when((df_with_header["DATA VALUE"] < -200) | (df_with_header["DATA VALUE"] >
```

```
300), 0).otherwise(df_with_header["DATA VALUE"])
```

```
)
```



如图所示我们提取出了所有的数据极值。

2.1.6 添加自定义字段

根据需求拆分或添加字段，例如时间拆分为年、月、日等。

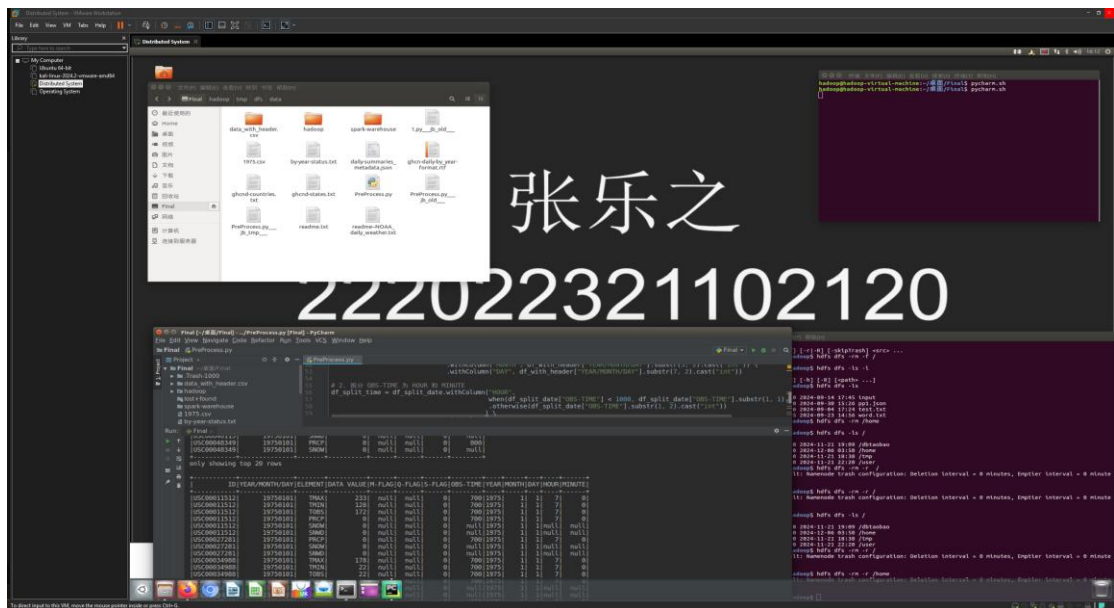
```
# 1. 拆分 YEAR/MONTH/DAY 或 YEAR_MONTH 和 DAY
df_split_date = df_with_header.withColumn("YEAR", df_with_header["YEAR/MONTH/DAY"].substr(1, 4).cast("int")) \
    .withColumn("MONTH", df_with_header["YEAR/MONTH/DAY"].substr(5, 2).cast("int")) \
    .withColumn("DAY", df_with_header["YEAR/MONTH/DAY"].substr(7, 2).cast("int"))

# 2. 拆分 OBS-TIME 为 HOUR 和 MINUTE
df_split_time = df_split_date.withColumn("HOUR",
    when(df_split_date["OBS-TIME"] < 1000, df_split_date["OBS-TIME"].substr(1, 1).cast("int"))
    .otherwise(df_split_date["OBS-TIME"].substr(1, 2).cast("int"))
    ) \
    .withColumn("MINUTE",
    when(df_split_date["OBS-TIME"] < 1000, df_split_date["OBS-TIME"].substr(2, 2).cast("int"))
    .otherwise(df_split_date["OBS-TIME"].substr(3, 2).cast("int"))
    )
```

only showing top 20 rows

ID	YEAR/MONTH/DAY	ELEMENT	DATA	VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME	YEAR	MONTH	DAY	HOUR	MINUTE
USC00011512	19750101	TMAX	233	null	null	0	700	1975	1	1	7	0	
USC00011512	19750101	TMIN	128	null	null	0	700	1975	1	1	7	0	
USC00011512	19750101	TOBS	172	null	null	0	700	1975	1	1	7	0	
USC00011512	19750101	PRCP	0	null	null	0	700	1975	1	1	7	0	
USC00011512	19750101	SNOW	0	null	null	0	null	1975	1	1	null	null	
USC00011512	19750101	SNWD	0	null	null	0	null	1975	1	1	null	null	
USC00027281	19750101	PRCP	0	null	null	0	700	1975	1	1	7	0	
USC00027281	19750101	SNOW	0	null	null	0	null	1975	1	1	null	null	
USC00027281	19750101	SNWD	0	null	null	0	null	1975	1	1	null	null	
USC00034988	19750101	TMAX	178	null	null	0	700	1975	1	1	7	0	
USC00034988	19750101	TMIN	22	null	null	0	700	1975	1	1	7	0	
USC00034988	19750101	TOBS	22	null	null	0	700	1975	1	1	7	0	
USC00034988	19750101	PRCP	30	null	null	0	700	1975	1	1	7	0	
USC00034988	19750101	SNOW	0	null	null	0	null	1975	1	1	null	null	
USC00034988	19750101	SNWD	0	null	null	0	null	1975	1	1	null	null	
USC00040115	19750101	PRCP	0	null	null	0	800	1975	1	1	8	0	
USC00040115	19750101	SNOW	0	null	null	0	null	1975	1	1	null	null	
USC00040115	19750101	SNWD	0	null	null	0	null	1975	1	1	null	null	
USC00048349	19750101	PRCP	0	null	null	0	800	1975	1	1	8	0	
USC00048349	19750101	SNOW	0	null	null	0	null	1975	1	1	null	null	

only showing top 20 rows



2.1.7 其他预处理项

标准化 (Standardization):

- 数据通过减去均值并除以标准差，使数据具有均值为 0 和标准差为 1。

$$z = \frac{x - \mu}{\sigma}$$

- 公式:

归一化 (Normalization):

- 数据通过缩放到特定区间 (例如 [0, 1]), 使所有数据点的数值范围一致。

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- 公式:

计算总和和总数 (用于计算均值)

```
sum_count = df_split_time.rdd.map(lambda row: (row["DATA VALUE"], 1)) \
    .reduce(lambda x, y: (x[0] + y[0], x[1] + y[1]))
```

mean = sum_count[0] / sum_count[1] # 均值

计算标准差


```

variance = df_split_time.rdd.map(lambda row: (row["DATA VALUE"] - mean)
** 2) \
                                .reduce(lambda x, y: x + y) / sum_count[1]
stddev = variance ** 0.5  # 标准差

# 计算最大值和最小值
min_max = df_split_time.rdd.map(lambda row: (row["DATA VALUE"],
row["DATA VALUE"])) \
                                .reduce(lambda x, y: (min(x[0], y[0]),
max(x[1], y[1])))
min_value = min_max[0]  # 最小值
max_value = min_max[1]  # 最大值

# 打印计算结果
print("Mean: {}, Standard Deviation: {}, Min: {}, Max: {}".format(mean,
stddev, min_value, max_value))

# 添加标准化和归一化列
df_standardized = df_split_time.withColumn(
    "DATA VALUE (Standardized)",
    (df_split_time["DATA VALUE"] - mean) / stddev
)

df_normalized = df_standardized.withColumn(
    "DATA VALUE (Normalized)",
    (df_standardized["DATA VALUE"] - min_value) / (max_value -
min_value)
)

```

如图所示，这些代码将归一化之后的值放入了新的列，以及每行对应的标准插。

ID	YEAR/MONTH/DAY	ELEMENT	DATA VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME	YEAR	MONTH	DAY	HOUR	MINUTE	DATA VALUE (Standardized)	DATA VALUE (Normalized)
USC00011512	19750101	TMX	233	null	null	0	700 1975	1	1	7	0	0	0.9840586891810396	0.056536492781963536
USC00011512	19750101	TMIN	128	null	null	0	700 1975	1	1	7	0	0	0.35784847008351	0.05049749813678737
USC00011512	19750101	TOBS	172	null	null	0	700 1975	1	1	7	0	0	0.6202582665854777	0.05302812446880405
USC00011512	19750101	PRCP	0	null	null	0	700 1975	1	1	7	0	0	-0.40553964630692646	0.04313567607982976
USC00011512	19750101	SNOW	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00011512	19750101	SNWD	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00027281	19750101	PRCP	0	null	null	0	700 1975	1	1	7	0	0	-0.40553964630692646	0.04313567607982976
USC00027281	19750101	SNOW	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00027281	19750101	SNWD	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00034988	19750101	TMX	178	null	null	0	700 1975	1	1	7	0	0	0.6560419147096312	0.05337320986044269
USC00034988	19750101	TMIN	22	null	null	0	700 1975	1	1	7	0	0	-0.27433293651836316	0.044400989244838096
USC00034988	19750101	TOBS	22	null	null	0	700 1975	1	1	7	0	0	-0.27433293651836316	0.044400989244838096
USC00034988	19750101	PRCP	30	null	null	0	700 1975	1	1	7	0	0	-0.22662140588015832	0.044861183123022945
USC00034988	19750101	SNOW	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00034988	19750101	SNWD	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00040115	19750101	PRCP	0	null	null	0	800 1975	1	1	8	0	0	-0.40553964630692646	0.04313567607982976
USC00040115	19750101	SNOW	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976
USC00048349	19750101	PRCP	0	null	null	0	800 1975	1	1	8	0	0	-0.40553964630692646	0.04313567607982976
USC00048349	19750101	SNOW	0	null	null	0	null 1975	1	1	null	null	0	-0.40553964630692646	0.04313567607982976

only showing top 20 rows

2.2 数据统计分析

2.2.1 数据集行数

统计数据集的总行数。

```
24/12/06 18:17:48 WARN util.NativeCodeLoader: Unable to load native code library: /lib64/libjvarkit.so, falling back to the pure Java path. This can lead to reduced performance.  
24/12/06 18:17:47 WARN util.Utils: Your system has 16 GB memory.  
24/12/06 18:17:47 WARN util.Utils: Set the JVM's heap size to 1 GB.  
数据集总行数: 33775698  
[Stage 4:>
```

显示每个重要字段的不重复数据:

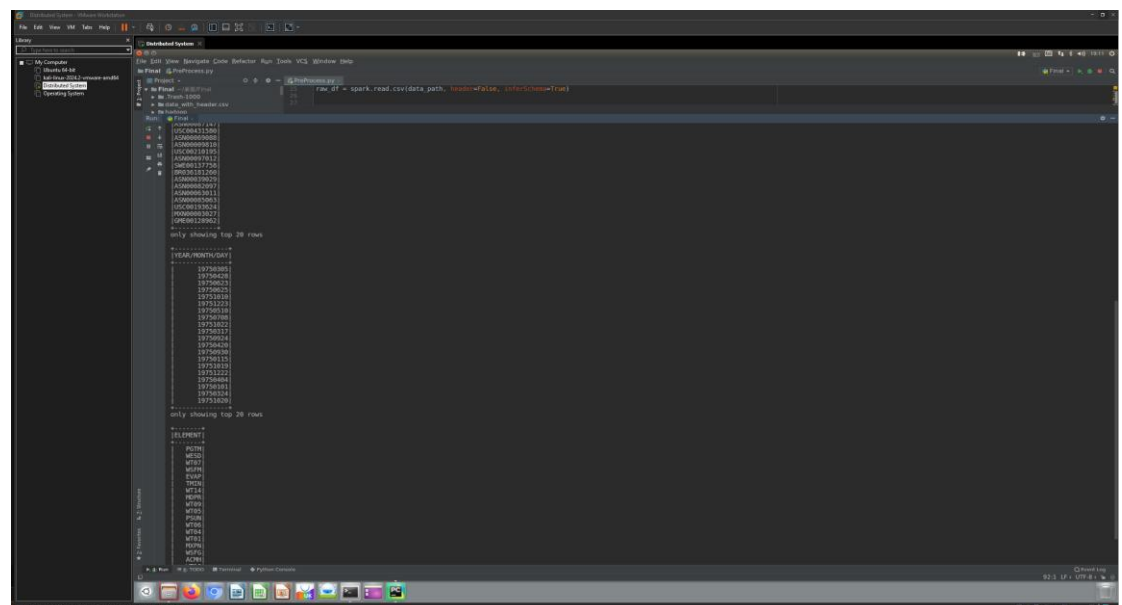
```
df_with_header.select("YEAR/MONTH/DAY").distinct().show()
```

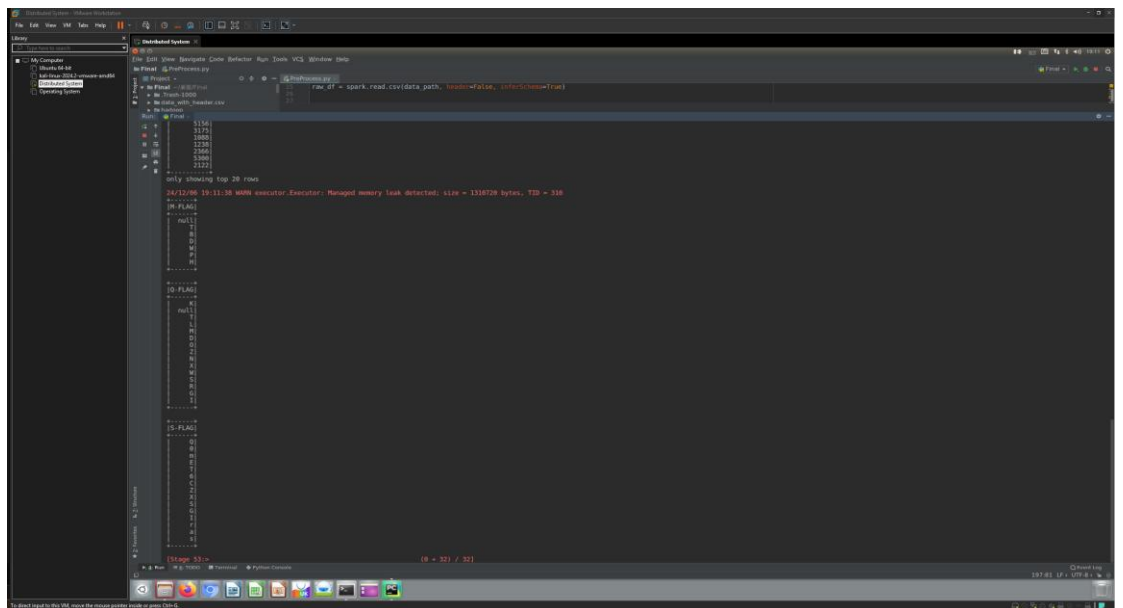
```
df_with_header.select("DATA VALUE").distinct().show()
```

```
df_with_header.select("Q-FLAG").distinct().show()
```

```
df_with_header.select("OBS-TIME").distinct().show()
```

借助这些代码，我们可以拉取重要字段的不重复的数据：





2.3 字段统计信息

统计每个数值字段的最大值、最小值、平均值等信息。

from pyspark.sql import functions as F

stats_df = df_with_header.groupBy('ELEMENT', 'ID').agg(

 F.avg('DATA VALUE').alias('AVG_DATA_VALUE'),

 F.max('DATA VALUE').alias('MAX_DATA_VALUE'),

 F.min('DATA VALUE').alias('MIN_DATA_VALUE')

)

stats_df.show()

ELEMENT	ID	AVG_DATA_VALUE	MAX_DATA_VALUE	MIN_DATA_VALUE
TOBS	USC00309189	85.61917808219178	250	-178
TAVG	ARM00087016	235.46905537459284	350	78
SNWD	RQC00660061	0.0	0	0
TOBS	USC00411810	148.73888888888888	261	-72
TMIN	MXN00013049	85.96153846153847	144	-11
SNWD	USC00227701	0.0	0	0
TMIN	USC00331781	56.37391304347826	222	-183
TMIN	CA007058560	2.408219178082192	228	-283
PRCP	ASN00091077	28.64804469273743	360	0
PRCP	ASN00003019	18.742465753424657	900	0
TMIN	USC00256018	13.482093663911845	233	-306
TOBS	USC00210643	22.626373626373628	278	-311
SNWD	SWE00138588	15.161290322580646	120	0
SNWD	USC00302574	55.62739726027397	610	0
TMAX	AFM00040948	190.16949152542372	350	-10
SNWD	USC00141699	6.328767123287672	178	0
SNWD	USC00414563	0.0	0	0
PRCP	ASN00035178	19.821727019498606	712	0
PRCP	ASN00050034	11.271232876712329	546	0
TMIN	UKE00105875	29.101369863013698	151	-114

only showing top 20 rows

2.2.4 特定查询

查询全年最大温度、最小温度、最大降雨、最小降雨的日期。

```
# 转换 DataFrame 为 RDD
```

```
rdd = df_with_header.rdd
```

```
# 筛选出与温度和降雨相关的数据
```

```
filtered_rdd = rdd.filter(lambda row: row["ELEMENT"] in ["TMAX", "TMIN", "PRCP"])
```

```
# 找到每种 ELEMENT 的最大值及日期
```

```
max_rdd = (
    filtered_rdd
    .map(lambda row: (row["ELEMENT"], (row["DATA VALUE"],
row["YEAR/MONTH/DAY"])))
    .reduceByKey(lambda a, b: a if a[0] > b[0] else b) # 按最大值比较
)
```

```
# 找到每种 ELEMENT 的最小值及日期
```

```
min_rdd = (
```

```

        filtered_rdd

        .map(lambda row: (row["ELEMENT"], (row["DATA VALUE"],
row["YEAR/MONTH/DAY"])))

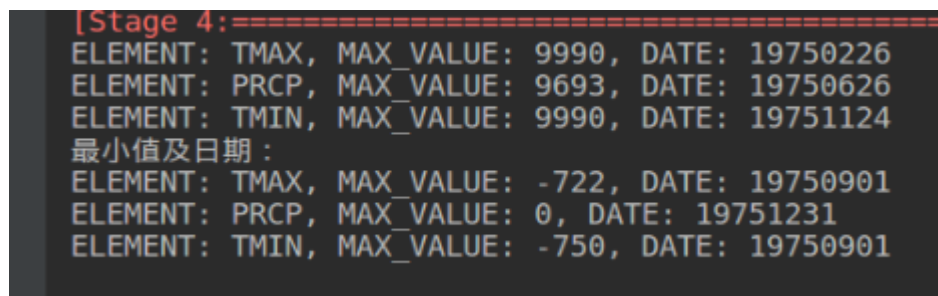
        .reduceByKey(lambda a, b: a if a[0] < b[0] else b) # 按最小值比较
    )

# 将结果收集到本地
max_results = max_rdd.collect()
min_results = min_rdd.collect()

# 打印结果
print("最大值及日期: ")
for element, (value, date) in max_results:
    print("ELEMENT: {}, MAX_VALUE: {}, DATE: {}".format(element, value, date))

print("最小值及日期: ")
for element, (value, date) in min_results:
    print("ELEMENT: {}, MAX_VALUE: {}, DATE: {}".format(element, value, date))

```



```

[Stage 4:=====]
ELEMENT: TMAX, MAX_VALUE: 9990, DATE: 19750226
ELEMENT: PRCP, MAX_VALUE: 9693, DATE: 19750626
ELEMENT: TMIN, MAX_VALUE: 9990, DATE: 19751124
最小值及日期:
ELEMENT: TMAX, MAX_VALUE: -722, DATE: 19750901
ELEMENT: PRCP, MAX_VALUE: 0, DATE: 19751231
ELEMENT: TMIN, MAX_VALUE: -750, DATE: 19750901

```

2.2.5 重庆站点数据查询

查询数据集是否包含重庆站点的数据，并列出具体的信息。

首先我们寻找重庆的站点。根据重庆经纬度：

中国重庆市的纬度是 29.5647398，经度是 106.5478767
度分秒格式的纬度是 29°33'53.1"N，经度是 106°32'52.4"E

城市或地点	经纬度（纬度在前，经度在后）	度分秒格式（纬度在前，经度在后）
中国 重庆市	29.5647398, 106.5478767	29°33'53.1"N, 106°32'52.4"E
中国 重庆市	29.5647398, 106.5478767	29°33'53.1"N, 106°32'52.4"E
Chongqing, China	29.5647398, 106.5478767	29°33'53.1"N, 106°32'52.4"E

以及提供的 stations.txt 数据中的经纬度，我们寻找重庆市区经纬度误差在 1 度以内的站点：

```
stations_path = "file:///home/hadoop/桌面/Final/ghcnd-stations.txt"
```

```
stations_schema = ["id", "lat", "lng", "altitude", "city", "unknown1", "unknownid"]
```

```
# 定义解析函数
```

```
def parse_fixed_width(row):
```

```
    return (
```

```
        row[0:11].strip(),    # ID
```

```
        float(row[12:20].strip()),    # LATITUDE
```

```
        float(row[21:30].strip()),    # LONGITUDE
```

```
        float(row[31:37].strip()) if row[31:37].strip() else None,    # ELEVATION
```

```
        row[38:40].strip(),    # STATE
```

```
        row[41:71].strip(),    # NAME
```

```
        row[72:75].strip(),    # GSN FLAG
```

```
        row[76:79].strip()    # HCN/CRN FLAG
```

```
    )
```

```
# 加载文件并跳过 header
```

```
stations_rdd = spark.read.text(stations_path).rdd.zipWithIndex().filter(lambda x: x[1] >
```

```
0).map(lambda x: x[0])
```

```

# 应用解析函数

parsed_rdd = stations_rdd.map(lambda row: parse_fixed_width(row.value))

# 转换为 DataFrame

stations_df = parsed_rdd.toDF(["ID", "LATITUDE", "LONGITUDE", "ELEVATION", "STATE",
"NAME", "GSN_FLAG", "HCN_CRN_FLAG"])

stations_df.show(5, truncate=False)

# 定义重庆的经纬度和筛选范围

chongqing_lat = 29.5647398

chongqing_lng = 106.5478767

delta = 1 # 容差范围，单位为度

# 筛选范围内的站点

chongqing_stations = stations_df.filter(
    (stations_df["LATITUDE"] >= chongqing_lat - delta) &
    (stations_df["LATITUDE"] <= chongqing_lat + delta) &
    (stations_df["LONGITUDE"] >= chongqing_lng - delta) &
    (stations_df["LONGITUDE"] <= chongqing_lng + delta)
)

随后我们将提取出来的站点和我们已有的数据文件 join:

# 假设气象数据 DataFrame 为 weather_df, 包含 ID 字段

chongqing_weather = chongqing_stations.join(df_with_header, on="ID", how="inner")

chongqing_weather.show(truncate=False)

```

ID	LATITUDE	LONGITUDE	ELEVATION	STATE	NAME	GSN_FLAG	HCN_CRN_FLAG
ACW00011604	17.1167	-61.7833	10.1		ST JOHNS COOLIDGE FLD		
ACW00011647	17.1333	-61.7833	19.2		ST JOHNS		
AE000041196	25.333	55.517	34.0		SHARJAH INTER. AIRP	GSN	
AEM00041194	25.255	55.364	10.4		DUBAI INTL		
AEM00041217	24.433	54.651	26.8		ABU DHABI INTL		

only showing top 5 rows

ID	LATITUDE	LONGITUDE	ELEVATION	STATE	NAME	GSN_FLAG	HCN_CRN_FLAG	YEAR/MONTH/DAY	ELEMENT	DATA VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	TMAX	87	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	TMIN	71	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	PRCP	0	T	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	TAVG	78	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	TMAX	81	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	TMIN	66	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	PRCP	0	T	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	TAVG	75	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	TMAX	78	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	TMIN	56	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	PRCP	18	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	TAVG	67	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	TMAX	107	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	TMIN	51	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	PRCP	0	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	TAVG	73	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	TMAX	84	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	TMIN	67	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	PRCP	0	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	TAVG	77	null	null	s	null

only showing top 20 rows

Process finished with exit code 0

2.2.6 每月降雨量统计

统计每月的降雨量，并找出降雨量最大月份（以重庆为例）：

```
rainfall_df = chongqing_weather.filter("ELEMENT = 'PRCP'")
```

提取年份和月份，使用 selectExpr 提取子字符串

```
rainfall_df = rainfall_df.selectExpr(
    "*",
    "substring(`YEAR/MONTH/DAY`, 1, 4) as YEAR",
    "substring(`YEAR/MONTH/DAY`, 5, 2) as MONTH",
    "CAST(`DATA VALUE` AS FLOAT) as DATA_VALUE"
)
```

按年份和月份汇总降雨量

```
monthly_rainfall = rainfall_df.groupBy("YEAR", "MONTH") \
    .agg({"DATA_VALUE": "sum"}) \
    .withColumnRenamed("sum(DATA_VALUE)", "TOTAL_RAINFALL")
```

找到降雨量最大的月份

```
max_rainfall = monthly_rainfall.orderBy("TOTAL_RAINFALL", ascending=False).limit(1)
```

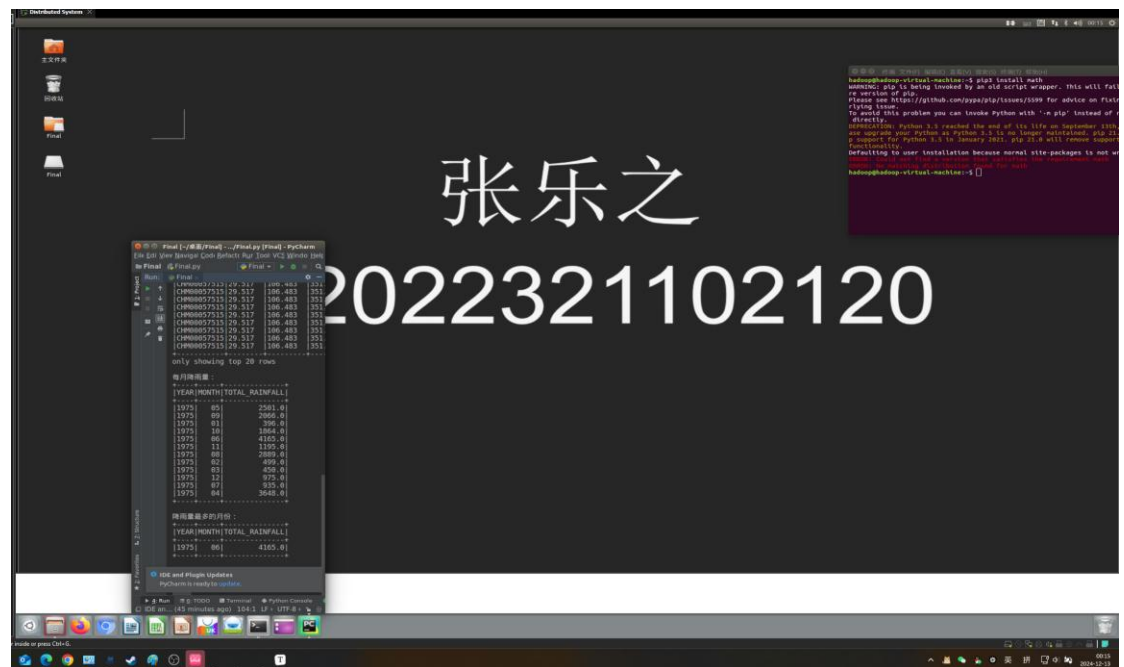
打印结果


```
print("每月降雨量：")

monthly_rainfall.show()

print("降雨量最多的月份：")

max_rainfall.show()
```



重庆每年降雨量最大的月份：

每月降雨量：

YEAR	MONTH	TOTAL_RAINFALL
1975	05	2501.0
1975	09	2066.0
1975	01	396.0
1975	10	1864.0
1975	06	4165.0
1975	11	1195.0
1975	08	2889.0
1975	02	499.0
1975	03	450.0
1975	12	975.0
1975	07	935.0
1975	04	3648.0

降雨量最多的月份：

YEAR	MONTH	TOTAL_RAINFALL
1975	06	4165.0

2.2.7 12 月天气统计

统计 12 月份的平均温度、最低温度、最高温度。

```
# 过滤温度数据 (假设温度相关的 ELEMENT 为 'TMAX', 'TMIN', 或 'TAVG')
temperature_df = df_with_header.filter("ELEMENT IN ('TMAX', 'TMIN', 'TAVG')")

# 提取年份、月份, 并转换 DATA VALUE 为数值类型
temperature_df = temperature_df.selectExpr(
    "*",
    "substring(`YEAR/MONTH/DAY`, 1, 4) as YEAR",
    "substring(`YEAR/MONTH/DAY`, 5, 2) as MONTH",
    "CAST(`DATA VALUE` AS FLOAT) as DATA_VALUE"
)

# 筛选 12 月的数据
december_df = temperature_df.filter("MONTH = '12'")

# 按照 ELEMENT 分类, 并统计 12 月份的平均温度、最低温度和最高温度
december_stats = december_df.groupBy("ELEMENT") \
    .agg(
        {"DATA_VALUE": "avg", "DATA_VALUE": "min", "DATA_VALUE": "max"}
    ) \
    .withColumnRenamed("avg(DATA_VALUE)", "AVERAGE_TEMPERATURE") \
    .withColumnRenamed("min(DATA_VALUE)", "MIN_TEMPERATURE") \
    .withColumnRenamed("max(DATA_VALUE)", "MAX_TEMPERATURE")

# 打印结果
print("12 月份的温度统计信息: ")
december_stats.show()
```

```

12 月份的温度统计信息：
+-----+-----+
|ELEMENT|MAX_TEMPERATURE|
+-----+-----+
|   TMIN|          4683.0|
|   TMAX|          1050.0|
|   TAVG|           383.0|
+-----+-----+

Process finished with exit code 0

```

2.3 利用 Spark 机器学习库进行深入分析

2.3.1 数据分析与预测任务

选择机器学习算法（例如回归模型）进行温度和降雨的预测。

训练模型并对模型的表现进行评估。

以重庆为例：

```

import math

from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType

# 定义自定义 UDF
def sin_udf(value):
    return math.sin(2 * math.pi * value / 12)

def cos_udf(value):
    return math.cos(2 * math.pi * value / 12)

sin_udf = udf(sin_udf, DoubleType())
cos_udf = udf(cos_udf, DoubleType())

# 添加周期性特征
def add_periodic_features(df):
    df = df.withColumn("MONTH_SIN", sin_udf(df["MONTH"].cast(DoubleType())))
    df = df.withColumn("MONTH_COS", cos_udf(df["MONTH"].cast(DoubleType())))

```

```

        return df

# 假设 temp_df 和 rain_df 是包含温度和降雨数据的 DataFrame
temp_df = add_periodic_features(data.filter("ELEMENT IN ('TMAX', 'TMIN', 'TAVG')"))
rain_df = add_periodic_features(data.filter("ELEMENT = 'PRCP'"))

# 构造特征向量
from pyspark.ml.feature import VectorAssembler

assembler_temp = VectorAssembler(inputCols=["MONTH_SIN", "MONTH_COS"],
outputCol="features")

assembler_rain = VectorAssembler(inputCols=["MONTH_SIN", "MONTH_COS"],
outputCol="features")

temp_data = assembler_temp.transform(temp_df).select("features", "DATA_VALUE")
rain_data = assembler_rain.transform(rain_df).select("features", "DATA_VALUE")

# 拆分训练集和测试集
temp_train, temp_test = temp_data.randomSplit([0.8, 0.2], seed=42)
rain_train, rain_test = rain_data.randomSplit([0.8, 0.2], seed=42)

# 初始化线性回归模型
from pyspark.ml.regression import LinearRegression

lr_temp = LinearRegression(featuresCol="features", labelCol="DATA_VALUE",
maxIter=100)

lr_rain = LinearRegression(featuresCol="features", labelCol="DATA_VALUE",
maxIter=100)

# 训练温度预测模型

```

```

temp_model = lr_temp.fit(temp_train)

rain_model = lr_rain.fit(rain_train)

# 对测试集进行预测

temp_predictions = temp_model.transform(temp_test)

rain_predictions = rain_model.transform(rain_test)

# 模型评估

from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(labelCol="DATA_VALUE", predictionCol="prediction",
metricName="rmse")

temp_rmse = evaluator.evaluate(temp_predictions)

rain_rmse = evaluator.evaluate(rain_predictions)

print("Temperature Prediction RMSE: {}".format(temp_rmse))

print("Rainfall Prediction RMSE: {}".format(rain_rmse))

```

ID	LATITUDE	LONGITUDE	ELEVATION	STATE	NAME	GSN_FLAG	HCN_CRN_FLAG
CHM00057515	29.517	106.483	351.0	CHONG-QING			
CHM00057516	29.583	106.467	416.0	CHONGQING			

ID	LATITUDE	LONGITUDE	ELEVATION	STATE	NAME	GSN_FLAG	HCN_CRN_FLAG	YEAR/MONTH/DAY	ELEMENT	DATA_VALUE	M-FLAG	Q-FLAG	S-FLAG	OBS-TIME
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	TMAX	87	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	TMIN	71	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	PRCP	0	T	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750101	TAVG	78	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	TMAX	81	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	TMIN	66	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	PRCP	0	T	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750102	TAVG	75	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	TMAX	78	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	TMIN	56	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	PRCP	18	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750103	TAVG	67	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	TMAX	107	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	TMIN	51	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	PRCP	0	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750104	TAVG	73	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	TMAX	84	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	TMIN	67	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	PRCP	0	null	null	s	null
CHM00057515	29.517	106.483	351.0		CHONG-QING			19750105	TAVG	77	null	null	s	null

only showing top 20 rows

24/12/13 02:24:41 WARN optim.WeightedLeastSquares: regParam is zero, which might cause numerical instability and overfitting.
[Stage 32:=====] (69 + 39) / 208]24/12/13 02:26:08 WARN netlib.BLAS: Failed to load implementation
24/12/13 02:26:08 WARN netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
24/12/13 02:26:08 WARN netlib.LAPACK: Failed to load implementation from: com.github.fommil.netlib.NativeSystemLAPACK
24/12/13 02:31:29 WARN optim.WeightedLeastSquares: regParam is zero, which might cause numerical instability and overfitting.
Temperature Prediction RMSE: 44.15889720979118
Rainfall Prediction RMSE: 50.43697744730628

2.3.2 预测方法

选择适当的算法（如线性回归、随机森林回归等），并简要说明所选方法的原

理和应用场景。

所选方法：线性回归

为了预测温度和降雨量，我选择了 **线性回归（Linear Regression）**，并结合了 **周期性特征**，即利用正弦（sine）和余弦（cosine）函数，将时间（月份）转换为可以捕捉一年内周期变化的特征。这些特征可以有效地表达全年温度和降雨量的趋势。

原理：

线性回归是一种经典的回归分析方法，用于通过最小化误差平方和拟合一条线性函数（即直线）。公式为：

$$y=w_1x_1+w_2x_2+\cdots+w_nx_n+by = w_1x_1 + w_2x_2 + \cdots + w_nx_n + by=w_1x_1+w_2x_2+\cdots+w_nx_n+b$$

其中：

- y 是预测值（标签）。
- x_1,x_2,\dots,x_n 是输入特征。
- w_1,w_2,\dots,w_n 是对应的回归系数（权重）。
- b 是截距项。

线性回归假设因变量和自变量之间存在线性关系，通过训练找到最优的回归系数 w 和 b 。

周期性特征

气候数据具有明显的周期性。例如，温度通常随季节周期性变化，降雨量在不同季节也可能表现出周期模式。为了捕捉这种规律，我们将时间特征（月份）转换为正弦和余弦值：

$$\text{MONTH_SIN} = \sin\left(\frac{2\pi \cdot \text{MONTH}}{12}\right), \quad \text{MONTH_COS} = \cos\left(\frac{2\pi \cdot \text{MONTH}}{12}\right)$$

正弦和余弦值的好处：

- 保持月份的周期关系，例如 12 月与 1 月的距离很小。
- 可以将时间特征转化为连续的、周期性变化的特征，有利于模型捕捉一年内的趋势。

数据预处理

为了让模型更好地拟合，我们对数据进行了以下预处理：

- 提取并转换时间特征为周期性特征。

- 分别为温度和降雨量构建特征向量（MONTH_SIN 和 MONTH_COS）。
- 按照 80%-20% 比例将数据分为训练集和测试集。

方法应用场景

1. 应用于温度预测

温度数据通常随季节呈周期性波动，使用正弦和余弦作为特征可以有效捕捉这种变化。

线性回归可以通过简单的参数拟合，快速预测全年温度趋势，适合短期预测或明确的周期性模式。

2. 应用于降雨量预测

降雨量可能也呈现周期性规律（例如雨季与旱季）。正弦和余弦特征可以帮助捕捉这些规律。

线性回归在降雨量预测中是一个基线模型，可用作更复杂模型（如时间序列分析）的对比。

3. 优点

简单高效：线性回归是非常高效的算法，计算复杂度较低。

可解释性强：模型权重直接反映每个特征对目标值的影响。

适合小规模数据：对我们当前的气候数据，线性回归的表现可以满足需求。

4. 局限性

对非线性数据表现有限：如果温度或降雨量的变化包含更复杂的非线性趋势，线性回归可能无法很好地拟合。

对异常值敏感：线性回归容易受极端数据点的影响。

无法捕捉多层复杂关系：如果气候数据受多种复杂因素影响，可能需要非线性模型（如决策树、随机森林或深度学习模型）。

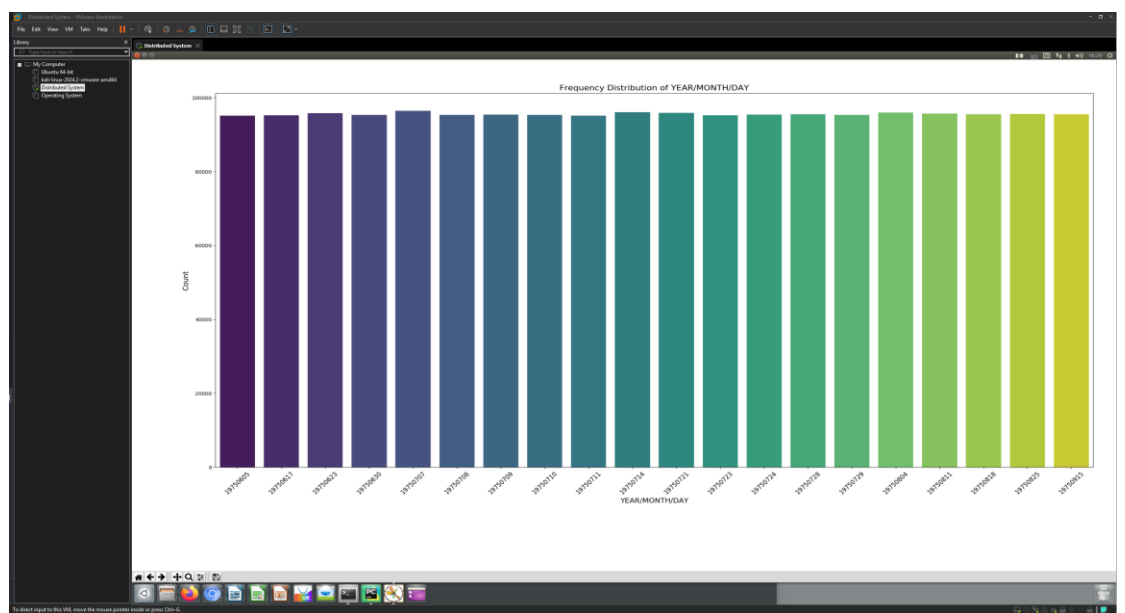
2.4 数据可视化

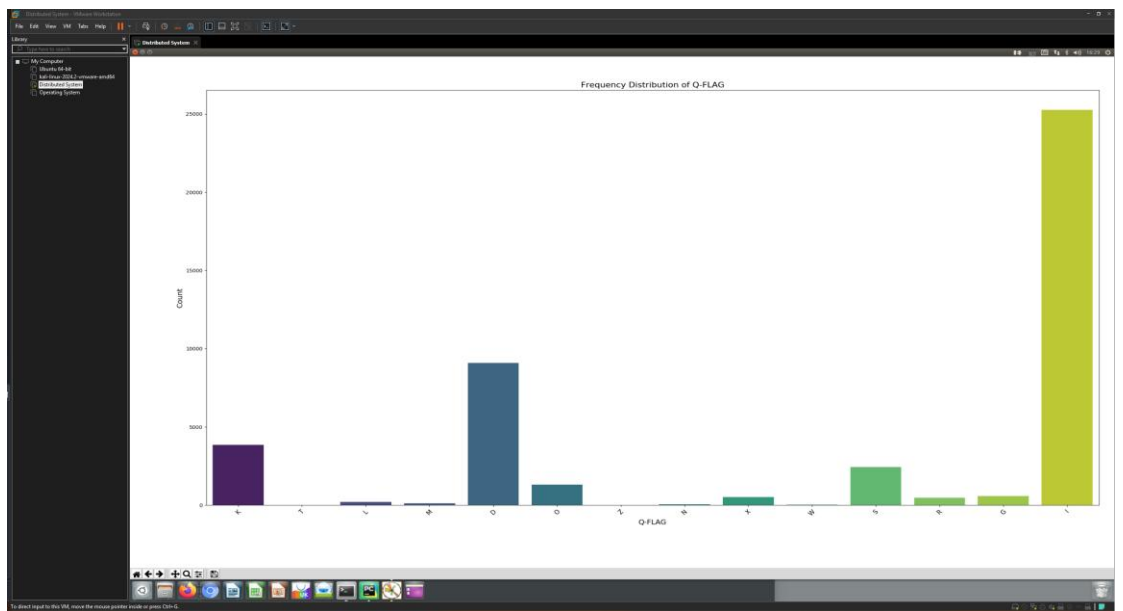
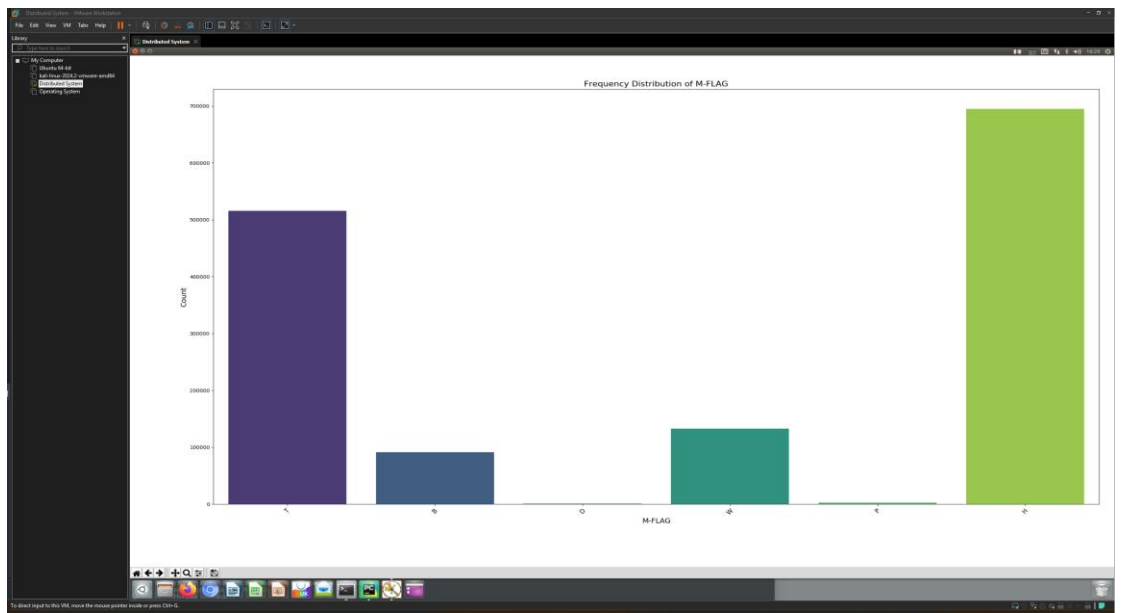
2.4.1 可视化任务

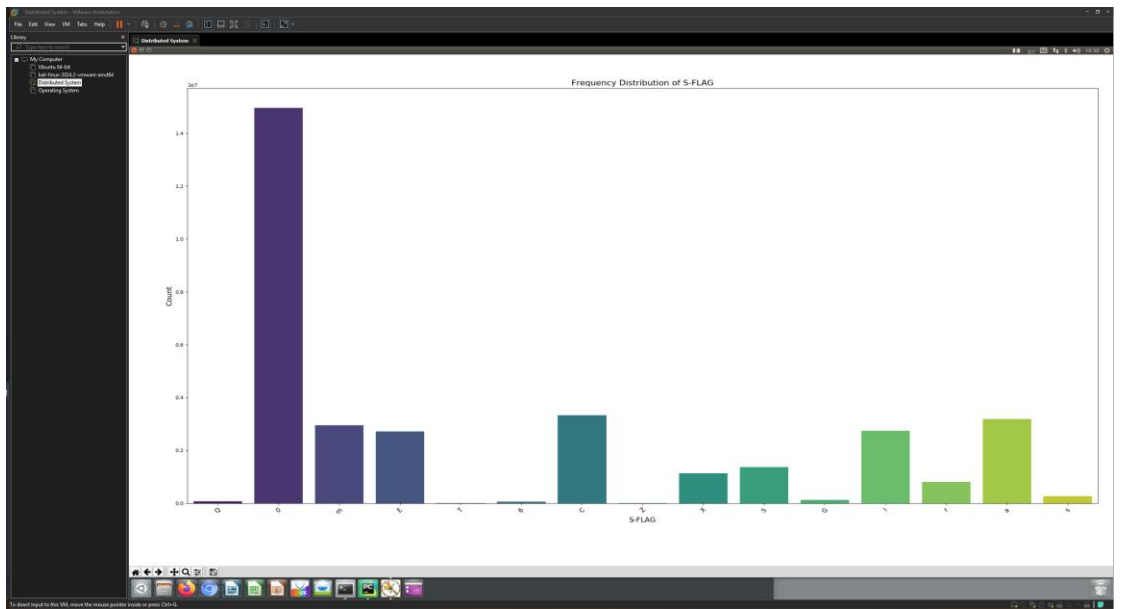
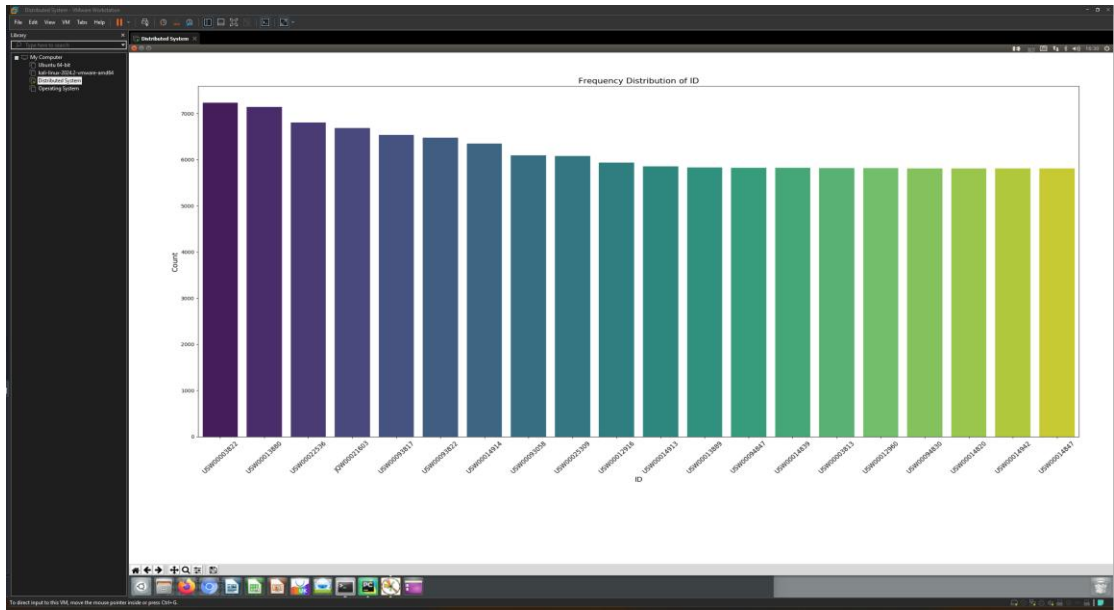
使用可视化工具（如 Matplotlib、Seaborn 等）展示数据分析结果。

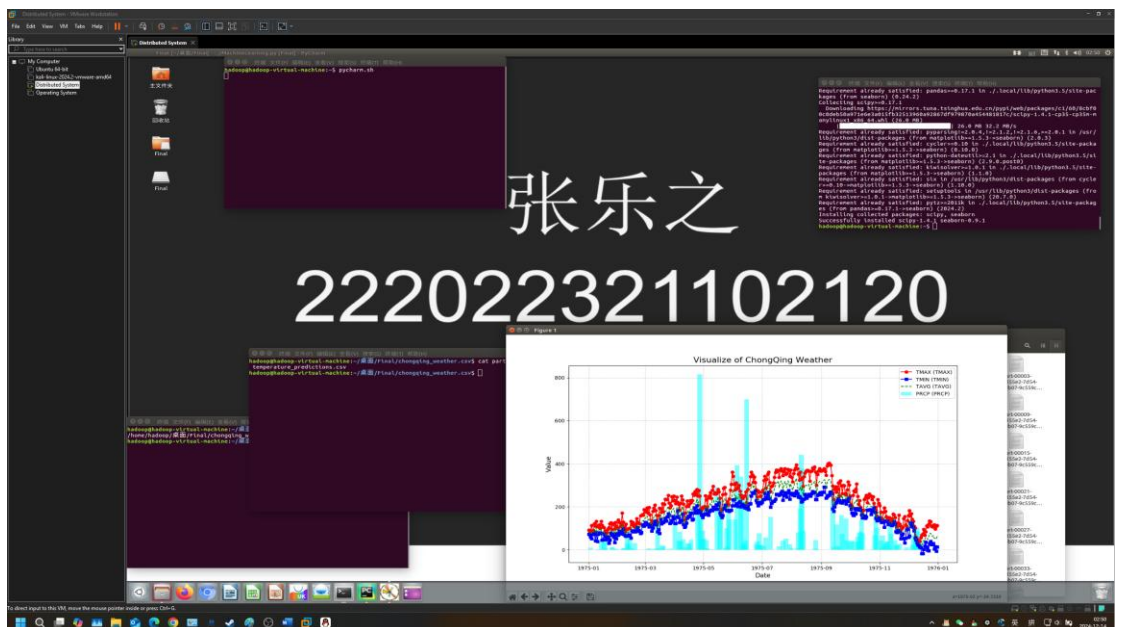
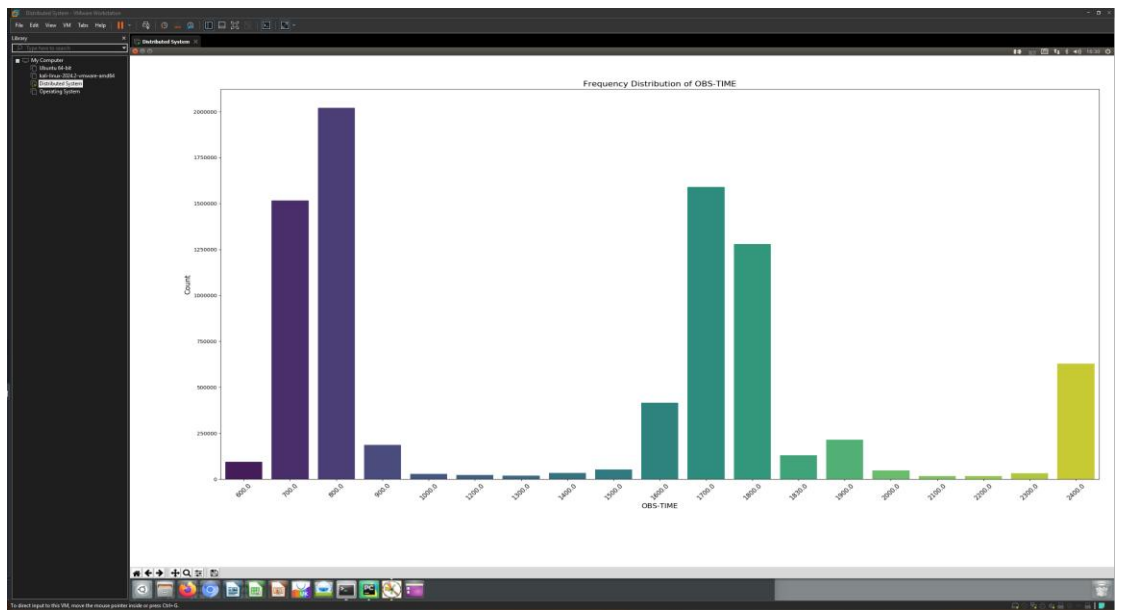
展示每月降雨量、最大降雨量月份、全年温度变化等统计结果。

每个重要统计字段的不重复数据有哪些:

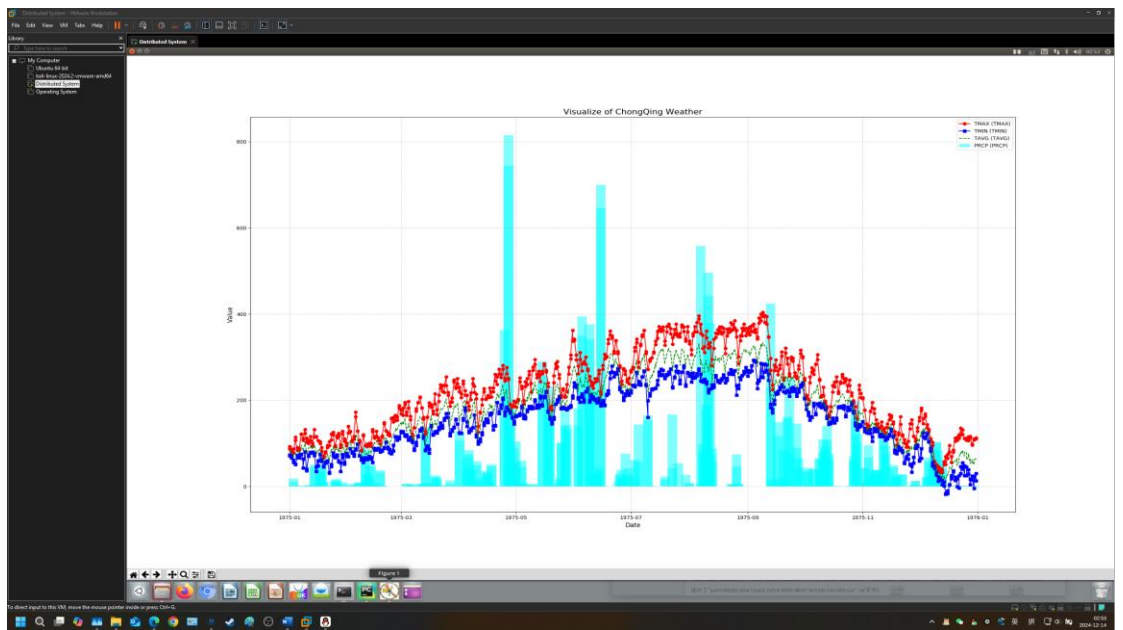




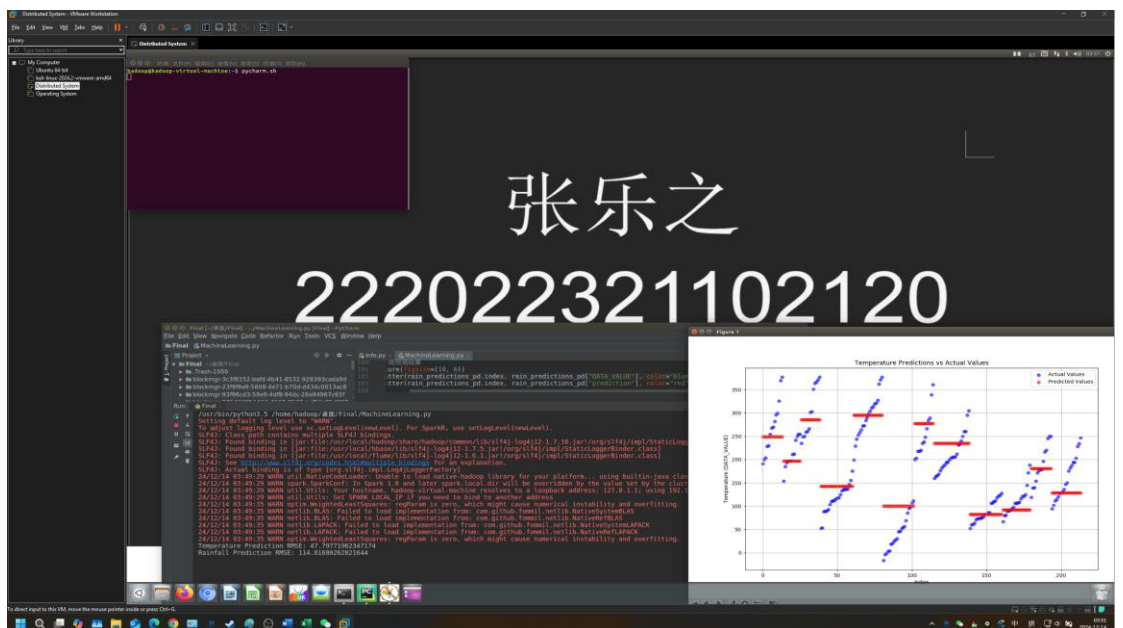
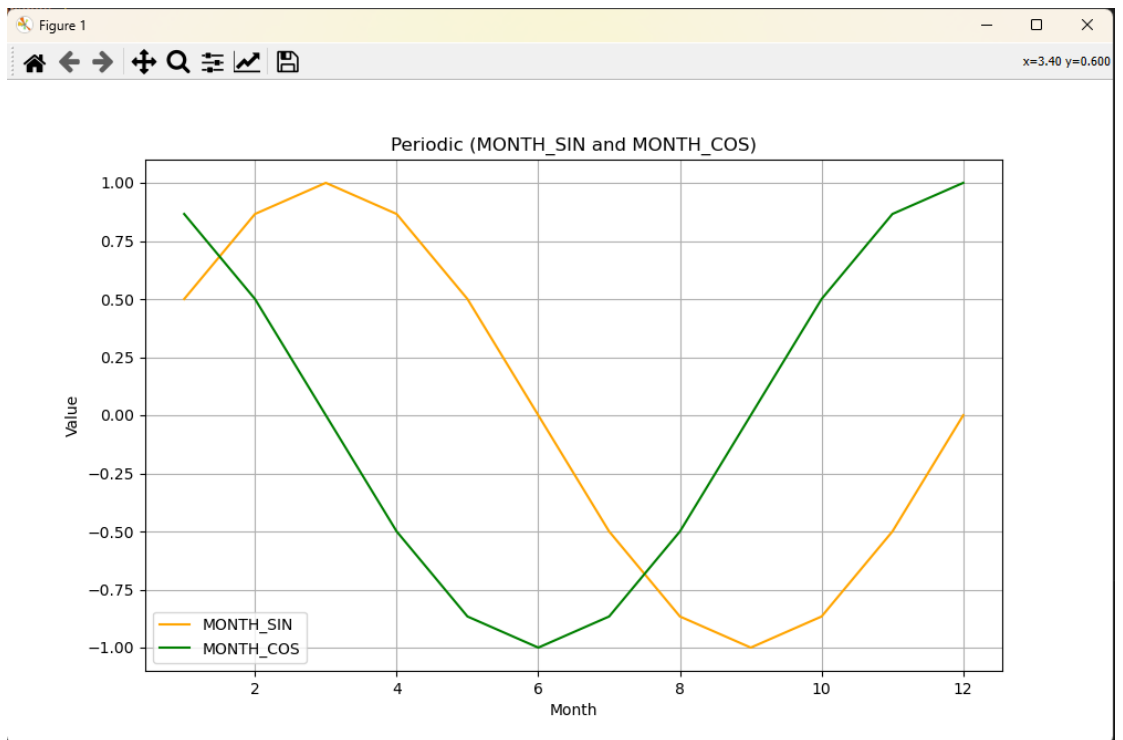




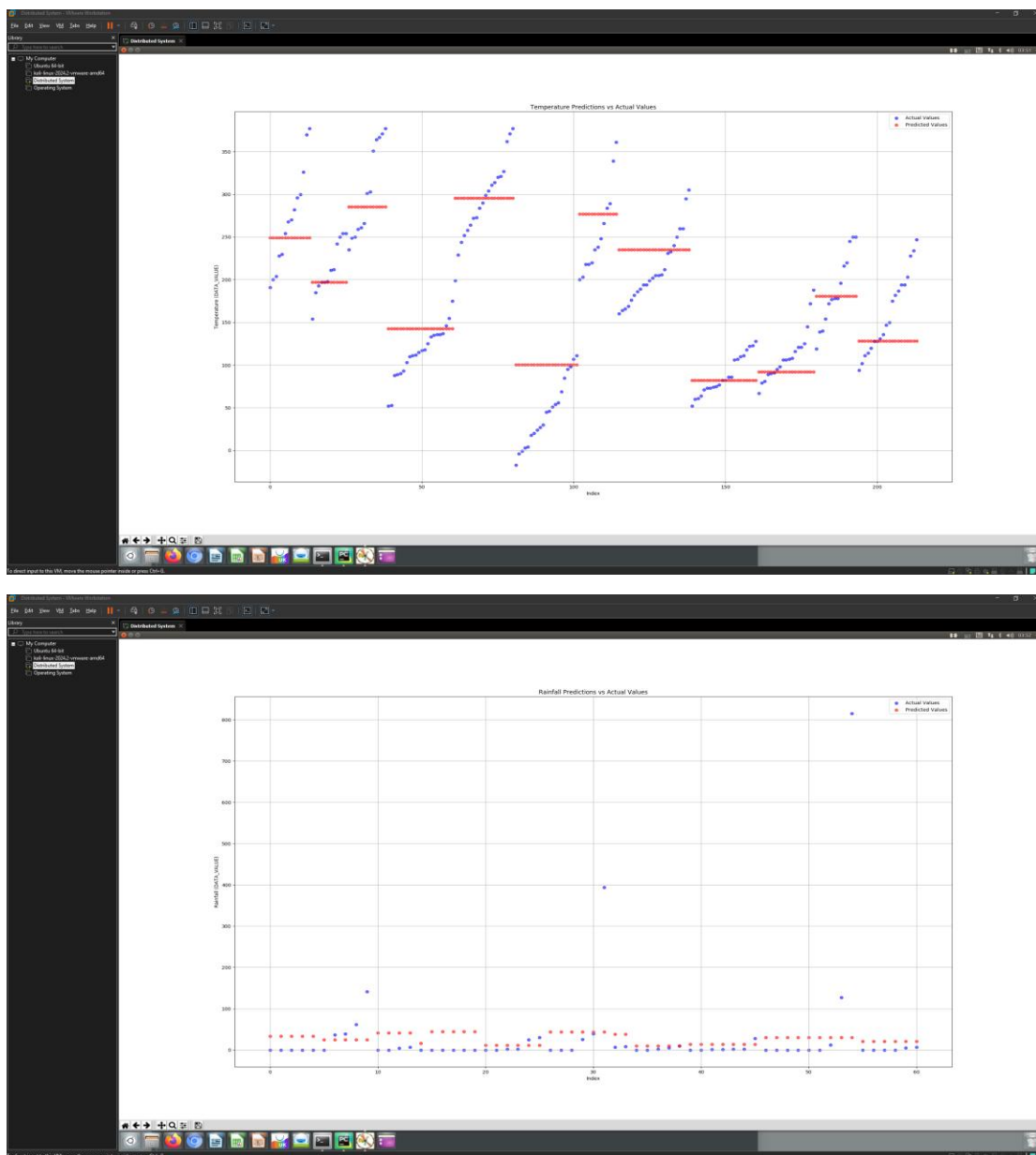
这幅图以可视化的方式展现了最高温，最低温，降水量的信息。考虑到源数据集较为庞大，包含多地区信息，此处仅展示重庆地区的天气信息。



这幅图可以直观告诉我们机器学习中添加的两个参数的作用范围：



这幅图则展示了我们的模型根据月份预测出来的平均气温。可以看到红色是预测值，蓝色是实际值。我们的模型很好地捕捉了气温的周期变化。



如图所示，这是我们的模型预测的降水量。可以看到实际值中有一些偏离平均值的极值，我们很难预测他们，但是大多数情况下我们的预测值还是很接近实际值的。

三、心得体会

在本次实验中我学会了很多东西。

以前在实验室打下手，经常写一点简单的程序来分析实验数据。当时自己写了一个简单的 `python` 程序对比图像差异，我看着任务管理器里 1 个满载的核心和 17 个空闲的核心在想，我能不能写一个程序利用这些空闲核心呢。现在我学习了操作系统和计算机组成原理，我知道当时的幼稚想法是可以实现的。后来我又想，能不能让自己的笔记本和宿舍的台式机同时进行运

算呢？

现在我知道了分布式计算，Hadoop 和 Spark 可以帮我做到这些事。

在本课程设计的过程中，我深入学习了如何使用 Spark 框架进行大数据处理与分析。通过对数据预处理、统计分析以及机器学习的实际操作，我不仅巩固了对 Spark 的理解，还提升了使用 Spark 进行大规模数据分析的能力。此外，在机器学习模型的应用和调整方面，我也掌握了多种常见的算法，并理解了它们的应用场景。

还有一件非常重要的事情，那就是如果使用 12 代以后 Intel 的 CPU，务必要使用管理员权限启动 VMWare，不然虚拟机会在小核心上运行。过去的两年里我都一直在用小核心跑虚拟机，非常卡，我以为这是由于虚拟机性能折损导致的。后来进行大数据计算的时候我打开任务管理器却发现大核心都在摸鱼，查阅资料才得知这件事情。

四、参考文献

Spark 官方文档: <https://spark.apache.org/docs/latest/>

<https://m.bilibili.com/opus/801343238473515027>

五、附：完整代码

请见压缩包中的完整代码。

课程设计 A

一、课程设计目的

在 Ubuntu 16.04(版本不限),Hadoop ,spark2.X 或 3.X 环境下，巩固和加深对课程知识的理解，并能够综合利用所学知识，实现对数据的处理，尤其是解决在大规模分布式环境下的数据处理问题。

二、数据来源

使用老师给定的数据完成课程设计。

给定的数据集 noaa-global-historical-climatology-network-daily 介绍：

Noaa-global-historical-climatology-network-daily 是全球历史气候网络-每日数据集，给定的数据是这个数据集的子集，涵盖了从 1962 年到 2016 年的气象数据。本次课程设计要求每位同选择一个年份（同班同学之间不能重

复)的数据开展数据处理。

三、课程设计

以下统计分析、机器学习任务均应在 Spark 框架中完成,不利用 Spark 框架完成的项目作品将视为不符合题目要求(可视化除外)。

1、数据预处理

- 1) 分析数据集中每个字段含义,并给出中文描述
- 2) 分析数据集中不同文件之间的关联,给出描述(图形和文字)
- 3) 处理标题行(如果没有则需要添加标题行,不可以直接对原始数据进行编辑,只能在 Spark 中操作)
- 4) 检查是否有空值的项,并清理
- 5) 检查数据中是否有异常值(负值,非数值类型等)
- 6) 根据自己需要添加某些字段(比如时间可以拆分成年月日,方便后续统计)
- 7) 其它自定义可预处理项

2、数据统计分析

- 1) 显示数据集行数
- 2) 每个重要统计字段的不重复数据有哪些
- 3) 每个字段的最大、最小、平均值等信息(可统计字段)
- 4) 查询全年最大温度、最小温度、最大降雨、最小降雨的日期
- 5) 查询所用数据集是否有属于重庆的站点的数据,如果有则列出详细信息
- 6) 统计每月降雨量、最大的降雨量月份
- 7) 统计 12 月份的平均温度,最低温度,最高温度

3、利用 Spark 机器学习库深入分析

尽可能对数据进行仔细分析,利用所学的 Spark 机器学习库进行深入分析,利用机器学习方法发现规律。

利用机器学习算法对温度和降雨进行预测,可使用临近的年份为验证

数据，预测方法可自行选择（但需要是 Spark 中的），因天气本身也很难预测，因此重点不在于考查大家的预测准确率，而在于对工具的使用，以及对使用的方法的解释。

4、数据可视化

根据各自完成情况，将第 2 步“数据统计分析”中的 2)、4)、6) 以及第 3 步“利用 Spark 机器学习库深入分析”中得到的结果（可参考天预报的界面），利用所学的可视化方法进行可视化。

四、提交课程设计包括的内容

- 1、课程设计任务与目的；
- 2、课程设计完成过程（含关键程序源代码，及系统截图）；
- 3、心得体会；
- 4、参考文献；
- 5、附完整代码（放压缩包内）；
- 6、演示视频

评分参考标准

课程设计成绩分为本表成绩评定分优秀、良好、中等、及格、不及格五个等级。具体评分参考标准如下：

成绩等级	评估标准
优	对数据的预处理全面充分，能全部清理除数据中的异常数据。使用 Spark SQL 或 DataFrame 算子全面统计数据 的内在特性，能对输入的数据有较深的洞察并能在程序和文档中体现，能熟练使用 Spark 机器学习库完成课程设计，数据可视化结果具有多样性（截图与代码证明），

	<p>能充分展示数据分布的内在特性（并有结果解释）。</p> <p>提交的代码与截图结果相符合，完全满足课程设计题目要求。文档撰写格式规范，对课程设计过程描述详实，每步骤均有对应详细描述，课程总结全面且能全面真实反应课程设计过程中的问题及解决过程，并能充分体现出个人对问题及解决的思考过程。实现课程设计过程中能积极主动查阅相应文档（尤其是官方文档）。</p>
良	<p>能使用 RDD、DataFrame 对数据进行预处理，能使用 Spark SQL 或 DataFram 完成统计数据分析，能使用 Spark 机器学习库完成机器学习分析任务，能够实现数据可视化。</p> <p>提交的代码与截图结果相符合，满足课程设计题目要求。文档撰写格式规范，对课程设计过程描述较详实，课程总结较充分反应课程设计过程中的问题及解决过程。实现课程设计过程中能主动查阅相应文档，能够较充分理解算法基本原理，并完成课程设计。能够完成数据可视化。</p>
中	<p>基本能使用 RDD、DataFrame 对数据进行预处理，存在部分异常未能清理的情况(或无对应清理代码)，基本能使用 Spark SQL 或 DataFram 完成统计数据分析，统计分析覆盖面不全，基本能使用 Spark 机器学习库完成机器学习分析任务，但并不能充分满足课程设计要求，文档内容编写基本符合课程设计要求，能够进行数据可视化，但可视化不太全面。</p>
及格	<p>存在数据预处理、Spark SQL 或 DataFram 统计分析，Spark 机器学习库、数据可视化四者中一项未完成的情况，但有提交文档且文档撰写基本符合课程设计要求，提交的代码可执行。</p>
不及格	<p>未能完成课程设计要求的内容（存在数据预处理、Spark SQL 或 DataFram 统计分析，Spark 机器学习库、数据可视化四者中 2 项及以上未完成的情况），文档撰写不符合要求。</p> <p>或者有下列情况之一者，课程设计成绩不及格：</p>

	<ul style="list-style-type: none">1)、未按要求完成课程设计；2)、抄袭、剽窃他人结果。3)、未提交相应代码
--	---