



# Forecasting Retail Sales for Walmart at scale

IDS 561 – Analytics for Big Data

1

# *Problem Setting*

*Context & reason for our Analysis*

# Problem Setting

*The Competitive Landscape & Thin Margins in the Retail Segment*

**Fierce Competition**

**Expansion of Stores**

**Retain Loyal Customers**

**Solution: Leverage Data**

PySpark

# 2 *Data Description*

*Details about the Dataset*

# Data Description

- We have used publicly available sales data for 45 Walmart stores across the United States. (Available on Kaggle)
- 420K+ rows for training and 100K+ rows of validation data.
- The primary data consists of weekly sales per store per department. Additionally, we have information on indicators like CPI, fuel prices, unemployment, temperature, holiday, etc.
- The entire dataset is split across five files, which is merged into a single dataset as a part of the data cleaning process.

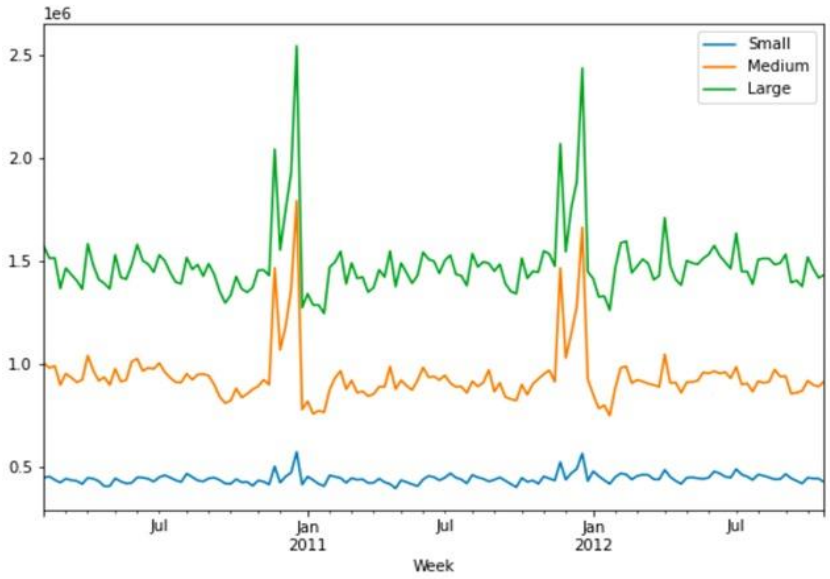
```
[ ]  1  merged_data.show()
```

| Dept | Weekly_Sales | Store | Date | Temperature | Fuel_Price | MarkDown1 | MarkDown2 | MarkDown3 | MarkDown4 | MarkDown5 | CPI | Unemployment | IsHoliday | Type | Size | Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 24924.5 | 1 | 2010-02-05 | 42.31 | 2.572 | NA | NA | NA | NA | NA | 211.0963582 | 8.106 | FALSE | A | 151315 | Large |
| 1 | 46039.49 | 1 | 2010-02-12 | 38.51 | 2.548 | NA | NA | NA | NA | NA | 211.2421698 | 8.106 | TRUE | A | 151315 | Large |
| 1 | 41595.55 | 1 | 2010-02-19 | 39.93 | 2.514 | NA | NA | NA | NA | NA | 211.2891429 | 8.106 | FALSE | A | 151315 | Large |
| 1 | 19403.54 | 1 | 2010-02-26 | 46.63 | 2.561 | NA | NA | NA | NA | NA | 211.3196429 | 8.106 | FALSE | A | 151315 | Large |
| 1 | 21827.9 | 1 | 2010-03-05 | 46.5 | 2.625 | NA | NA | NA | NA | NA | 211.3501429 | 8.106 | FALSE | A | 151315 | Large |
| 1 | 21043.39 | 1 | 2010-03-12 | 57.79 | 2.667 | NA | NA | NA | NA | NA | 211.3806429 | 8.106 | FALSE | A | 151315 | Large |

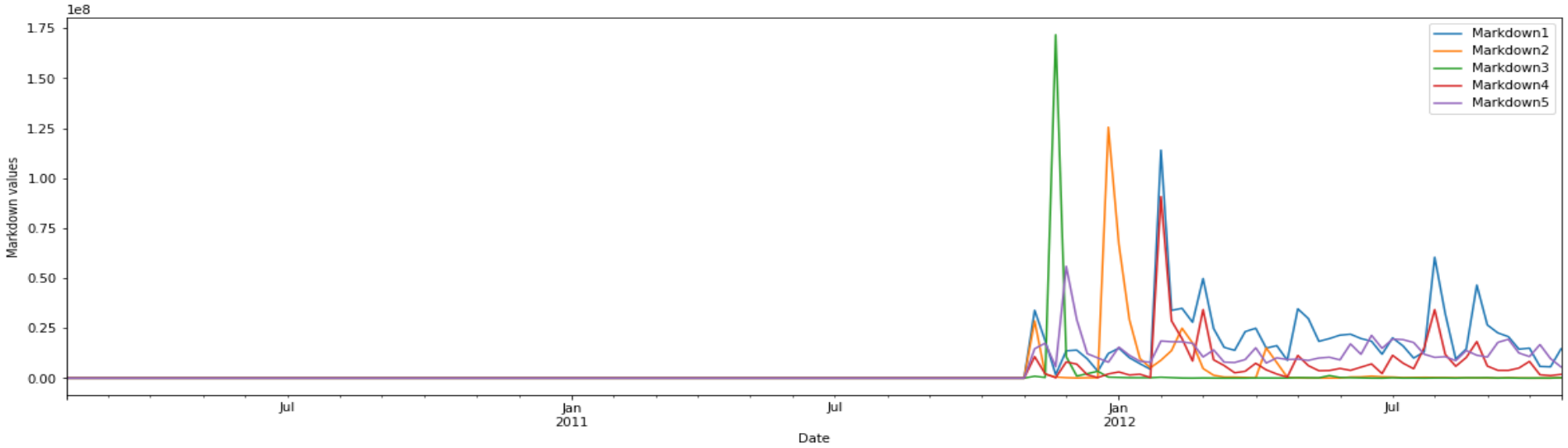
# Data Description

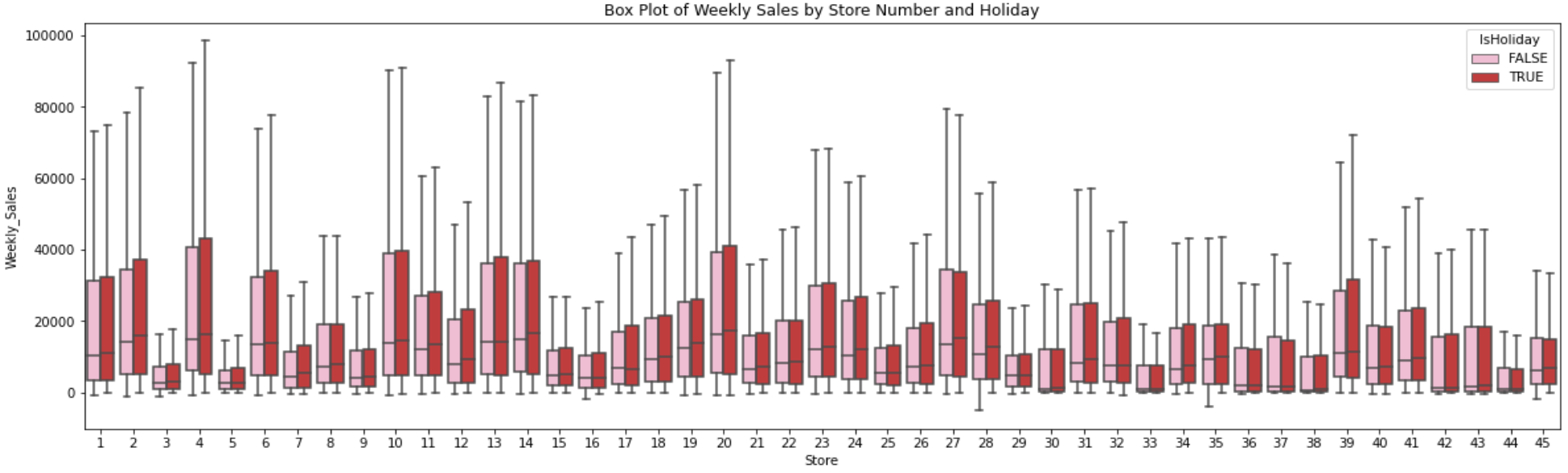
# Data Analysis





# Data Analysis



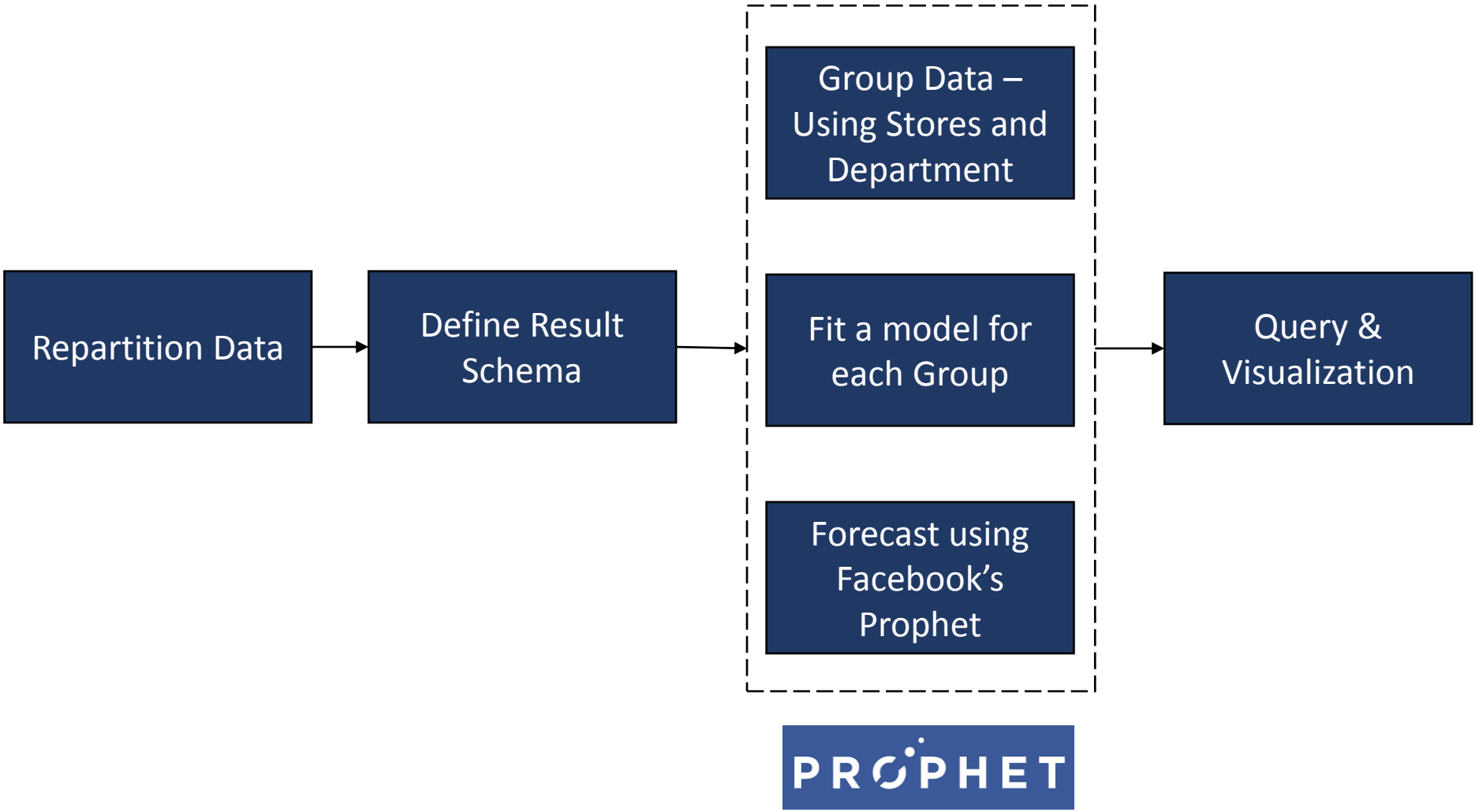



3

# *Methodology*

*Our Approach to Understand and Predict Sales*

# Multiple Time Series Forecasting



# Multiple Time Series Forecasting

```
[ ]  1 train.explain()
```

```
== Physical Plan ==
FileScan csv [Store#244,Dept#245,Date#246,Weekly_Sales#247] Batched: false, DataFilters: [], Format: CSV, Location: InMemoryFileIndex(1 paths)
```

```
1 sql_statement = '''
2   SELECT
3     Store as store,
4     Dept as dept,
5     CAST(Date as date) as ds,
6     SUM(Weekly_Sales) as y
7   FROM train
8   GROUP BY store, dept, ds
9   ORDER BY store, dept, ds
10  '''
11
12 train = (
13   spark
14     .sql( sql_statement )
15     .repartition(sc.defaultParallelism, ['store', 'dept'])
16   ).cache()
```

```
[ ]  1 train.explain()
```

```
== Physical Plan ==
InMemoryTableScan [store#279, dept#280, ds#281, y#282]
   +- InMemoryRelation [store#279, dept#280, ds#281, y#282], StorageLevel(disk, memory, deserialized, 1 replicas)
         +- Exchange hashpartitioning(store#279, dept#280, 2), REPARTITION_BY_NUM, [id=#225]
            +- *(3) Sort [store#279 ASC NULLS FIRST, dept#280 ASC NULLS FIRST, ds#281 ASC NULLS FIRST], true, 0
               +- Exchange rangepartitioning(store#279 ASC NULLS FIRST, dept#280 ASC NULLS FIRST, ds#281 ASC NULL
                  +- *(2) HashAggregate(keys=[store#244, dept#245, _groupingexpression#288], functions=[sum(Weekly
                     +- Exchange hashpartitioning(store#244, dept#245, _groupingexpression#288, 200), ENSURE_REQU
                        +- *(1) HashAggregate(keys=[store#244, dept#245, _groupingexpression#288], functions=[par
                           +- *(1) Project [Store#244, Dept#245, Weekly_Sales#247, cast(Date#246 as date) AS _gro
                              +- FileScan csv [Store#244,Dept#245,Date#246,Weekly_Sales#247] Batched: false, Datal
```

# Multiple Time Series Forecasting

```
result_schema = StructType([
  StructField('ds',DateType()),
  StructField('store',IntegerType()),
  StructField('dept',IntegerType()),
  StructField('y',FloatType()),
  StructField('yhat',FloatType()),
  StructField('yhat_upper',FloatType()),
  StructField('yhat_lower',FloatType())
  ])
```

```
def forecast_store_dept( history_pd: pd.DataFrame ) -> pd.DataFrame:

  model = Prophet(
    interval_width=0.95,
    daily_seasonality=False,
    weekly_seasonality=True,
    yearly_seasonality=True,
    holidays = holidays
    )

  model.fit( history_pd )

  future_pd = model.make_future_dataframe(
    periods=52,
    freq='w',
    include_history=True
    )

  forecast_pd = model.predict( future_pd )
```

```
from pyspark.sql.functions import current_date

results = (
  train
    .groupBy('store', 'dept')
      .applyInPandas(forecast_store_dept, schema=result_schema)
    .withColumn('training_date', current_date() )
    )

results.createOrReplaceTempView('forecasts')
```

```
test_sql = '''
SELECT
ds,
y
yhat,
yhat_upper,
yhat_lower
FROM forecasts
WHERE store = 1 AND dept = 1
'''

test_result = spark.sql(test_sql)
```

```
results.coalesce(1)
```

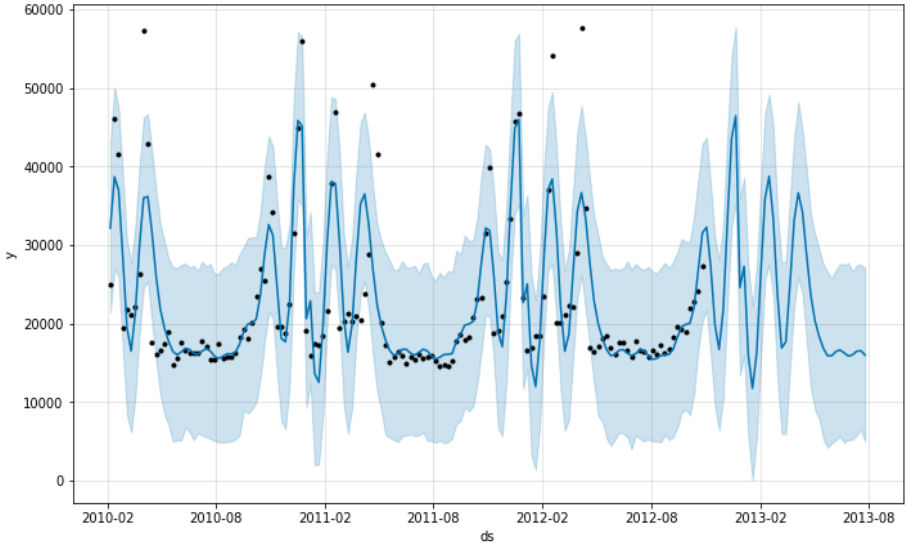
```
DataFrame[ds: date, store: int, dept: int, y: float, yhat: float, yhat_upp
```

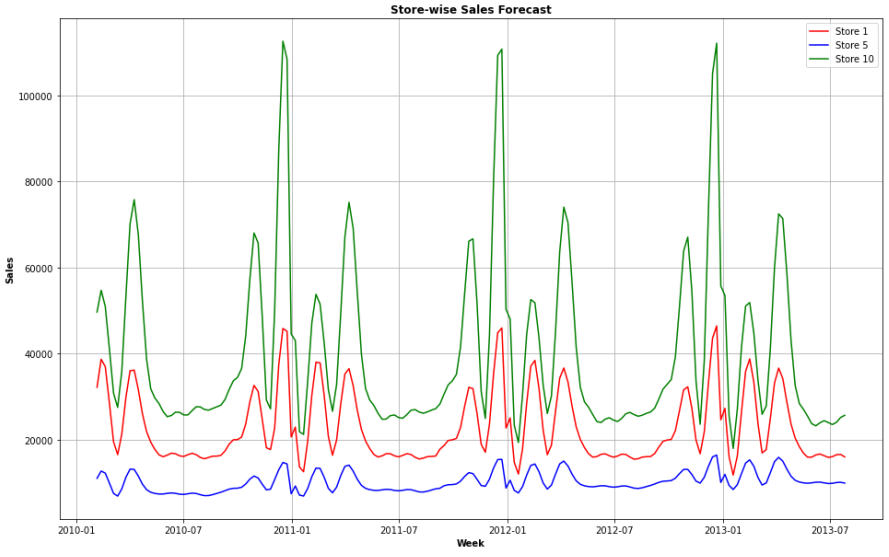
4

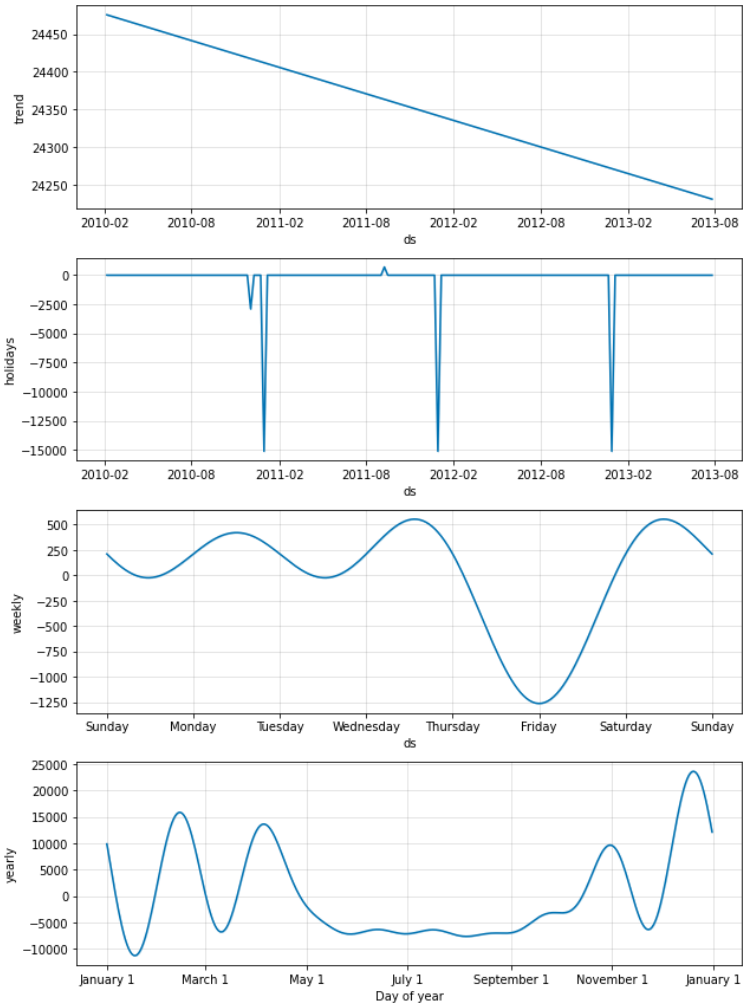
# *Results*

*Insights derived from the analysis*

# Results


y
ds
2010-02
2010-08
2011-02
2011-08
2012-02
2012-08
2013-02
2013-08


Store-wise Sales Forecast
Store 1
Store 5
Store 10
Sales
Week


trend
holidays
weekly
yearly
ds
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of year
January 1
March 1
May 1
July 1
September 1
November 1

# Thank you

Made by – Anirudha Balkrishna, Sanjay Madesha & Shubham Khode