1zwidzrpr

May 8, 2025

[1]:
```
#Install Pyspark and circlify
!pip install pyspark
!pip install circlify
!pip install plotly
!pip install numpy
!pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pyspark in /usr/lib/spark/python (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in
/opt/conda/miniconda3/lib/python3.11/site-packages (from pyspark) (0.10.9.7)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: circlify in
/home/sp8201_nyu_edu/.local/lib/python3.11/site-packages (0.15.0)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: plotly in
/home/sp8201_nyu_edu/.local/lib/python3.11/site-packages (6.0.1)
Requirement already satisfied: narwhals>=1.15.1 in
/home/sp8201_nyu_edu/.local/lib/python3.11/site-packages (from plotly) (1.38.0)
Requirement already satisfied: packaging in
/opt/conda/miniconda3/lib/python3.11/site-packages (from plotly) (23.1)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in
/opt/conda/miniconda3/lib/python3.11/site-packages (1.26.4)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in
/opt/conda/miniconda3/lib/python3.11/site-packages (2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in
/opt/conda/miniconda3/lib/python3.11/site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/miniconda3/lib/python3.11/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/opt/conda/miniconda3/lib/python3.11/site-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in
/opt/conda/miniconda3/lib/python3.11/site-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in
/opt/conda/miniconda3/lib/python3.11/site-packages (from python-
dateutil>=2.8.2->pandas) (1.17.0)
```

[2]:
```
import pyspark
from pyspark.sql.functions import *
from pyspark.sql.types import StructType, StructField, StringType, FloatType,␣
 ↪IntegerType
from pyspark.sql.window import Window
from pyspark.sql.functions import element_at, split, col
import pandas as pd
import os
import plotly.express as px
```

[3]:
```
# from pyspark.sql import SparkSession

# spark = SparkSession.builder \
#     .appName("NotebookSession") \
#     .config("spark.sql.repl.eagerEval.enabled", True) \
#     .getOrCreate()

# sc = spark.sparkContext  # Now you can access the SparkContext if needed
```

[3]:
```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

spark = SparkSession.builder.appName(" Retail_Behaviour_Analysis") \
    .config("spark.submit.deployMode", "client") \
    .config("spark.driver.memory", "8g") \
    .config("spark.executor.memory", "16g") \
    .config("spark.executor.instances", "4") \
    .getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
25/05/08 05:54:29 INFO SparkEnv: Registering MapOutputTracker
25/05/08 05:54:29 INFO SparkEnv: Registering BlockManagerMaster
25/05/08 05:54:30 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
25/05/08 05:54:30 INFO SparkEnv: Registering OutputCommitCoordinator
```

0.0.1 Initialize Spark Configuration and Context

This cell sets up the PySpark environment: - Enables eager evaluation of Spark SQL queries for easier debugging and inline result viewing. - Creates a `SparkContext` using the defined configuration. - Initializes a `SQLContext` to enable use of Spark SQL functionalities.

[4]:
```
# for Dataproc / Hadoop cluster
path_file_2019 = "hdfs:///user/sp8201_nyu_edu/retail_data_2024/2019-Nov-3MM.csv"
output_show = True  # it makes .show() on/off
small_data_testing = False  # make it false when testing on whole dataset
```

```
if small_data_testing:
    # For small data testing, you can copy a small sample locally and upload to␣
 ↪HDFS if needed
    import pandas as pd
    raw_data = pd.read_csv("/home/sp8201_nyu_edu/Retail_Behaviour_Analysis/
 ↪2019-Nov-3MM.csv", nrows=5000)
    raw_data.head(2)
    raw_data.to_csv("/home/sp8201_nyu_edu/Retail_Behaviour_Analysis/
 ↪small_nov_2019.csv", index=False)
    # (Optional) Upload to HDFS:
    # !hdfs dfs -put -f /home/sp8201_nyu_edu/Retail_Behaviour_Analysis/
 ↪small_nov_2019.csv /user/sp8201_nyu_edu/retail_data_2024/
    path_file_2019 = "hdfs:///user/sp8201_nyu_edu/retail_data_2024/
 ↪small_nov_2019.csv"

print("the file chosen is ", path_file_2019)
print("small data testing is ", small_data_testing)
```

```
the file chosen is
hdfs:///user/sp8201_nyu_edu/retail_data_2024/2019-Nov-3MM.csv
small data testing is  False
```

0.0.2 Configure File Path and Testing Mode

This cell sets up paths and flags for loading the dataset: - `path_file_2019` points to the full dataset file. - `output_show` toggles whether DataFrame results will be displayed with `.show()`. - `small_data_testing` flag allows switching to a smaller sample of the dataset for faster debugging. - If `True`, it loads only 5,000 rows, previews them, and saves the small sample as a new CSV file. - Prints the selected file path and current testing mode.

[5]:
```
# For final we will take the full dataset
original_df = spark.read \
    .option("inferSchema", "true") \
    .option("header", "true") \
    .format("csv") \
    .load(path_file_2019)

if output_show:
    original_df.show(5)
```

```
+-------------------+----------+----------+-------------------+----------------
---+-------+-------+---------+--------------------+
|         event_time|event_type|product_id|        category_id|
category_code|  brand|  price|  user_id|        user_session|
+-------------------+----------+----------+-------------------+----------------
---+-------+-------+---------+--------------------+
```

```
|2019-11-07 04:15:14|      view|
1005160|2053013555631882655|electronics.smart…| xiaomi|
210.53|513169571|f5d3f5b0-9da8-4cd…|
|2019-11-05 12:06:16|      view|   1004159|2053013555631882655|electronics.smart
…|samsung|1253.06|516325949|6711f48b-6255-454…|
|2019-11-17 07:24:15|      view|  15900033|2053013558190408249|
NULL|  tefal|   27.0|572468337|c66cddf2-e8bd-479…|
|2019-11-17 14:16:50|      view|  34500013|2061717948468297988|
NULL|   NULL|   6.18|524042389|87dd26f0-8a1a-479…|
|2019-11-16 09:33:21|      view|
8700375|2053013563097744201|appliances.person…|philips|
47.34|525988370|c3fee0d5-03db-478…|
+-------------------+----------+----------+-------------------+----------------
---+-------+-------+---------+--------------------+
only showing top 5 rows
```

0.1 Total No of Rows and Cols in given data set **

[6]:
```
# Number of rows
num_rows = original_df.count()

# Number of columns
num_columns = len(original_df.columns)

print(f"Number of rows: {num_rows}")
print(f"Number of columns: {num_columns}")
```

```
[Stage 3:=============================================>             (3 + 1) / 4]

Number of rows: 3000000
Number of columns: 9
```

0.1.1 Load Full Dataset into Spark DataFrame

This cell reads the complete CSV dataset using Spark: - Schema inference is enabled to automatically detect column data types. - Header option is set to `true` to treat the first row as column names. - Loads the file specified by `path_file_2019` into a Spark DataFrame called `original_df`.

[7]:
```
preprocessed_df = original_df
```

Dataset Overview

[8]:
```
preprocessed_df.printSchema()
```

```
root
 |-- event_time: timestamp (nullable = true)
 |-- event_type: string (nullable = true)
```

```
 |-- product_id: integer (nullable = true)
 |-- category_id: long (nullable = true)
 |-- category_code: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- price: double (nullable = true)
 |-- user_id: integer (nullable = true)
 |-- user_session: string (nullable = true)
```

0.1.2 Dataset Feature Descriptions

- **event_time**: Denotes the date and time of the user session.
- **event_type**: Represents the type of user interaction — can be one of three events: `view`, `cart`, or `purchase`.
- **product_id**: Identifies the specific product involved in the event.
- **category_id**: Identifies the category to which the product belongs.
- **user_id**: Uniquely identifies each user in the dataset.
- **user_session**: Represents a session ID; a single user can have multiple sessions capturing different event types (e.g., view, cart, purchase).
- **brand**: Indicates the brand associated with the product.
- **category_code**: A nested string (e.g., `electronics.smartphone.android`) representing hierarchical categorization of the product.

Data Summary and Pre-processing

[9]:
```
preprocessed_df = preprocessed_df.dropna(subset=["event_type"])
```

Data Imputation

Extracting Catgeory and Product

[10]:
```
preprocessed_df = preprocessed_df.na.fill(value = "empty", subset =␣
↪["category_code","brand"])
```

[11]:
```
if output_show:
  preprocessed_df.show(5)
```

```
+-------------------+----------+----------+-------------------+----------------
---+-------+-------+---------+--------------------+
|         event_time|event_type|product_id|        category_id|
category_code|  brand|  price|  user_id|        user_session|
+-------------------+----------+----------+-------------------+----------------
---+-------+-------+---------+--------------------+
|2019-11-07 04:15:14|      view|
1005160|2053013555631882655|electronics.smart…| xiaomi|
210.53|513169571|f5d3f5b0-9da8-4cd…|
|2019-11-05 12:06:16|      view|   1004159|2053013555631882655|electronics.smart
…|samsung|1253.06|516325949|6711f48b-6255-454…|
|2019-11-17 07:24:15|      view|  15900033|2053013558190408249|
empty|  tefal|   27.0|572468337|c66cddf2-e8bd-479…|
```

```
|2019-11-17 14:16:50|      view|  34500013|2061717948468297988|
empty|  empty|   6.18|524042389|87dd26f0-8a1a-479…|
|2019-11-16 09:33:21|      view|
8700375|2053013563097744201|appliances.person…|philips|
47.34|525988370|c3fee0d5-03db-478…|
+-------------------+----------+----------+-------------------+----------------
---+-------+-------+---------+--------------------+
only showing top 5 rows
```

[12]:
```
#Deriving category and product names from the category_code column using UDF.
@udf
def extract_category(category, brand):
    newlist = str(category).split('.')
    if newlist[0]=="empty":
      if brand == "empty":
        return "unknown"
      return brand
    return newlist[0]

@udf
def extract_product(category, brand):
    newlist = str(category).split('.')
    if newlist[-1] == "empty":
      if brand == "empty":
        return "unknown"
      return brand
    return newlist[-1]
```

0.1.3 Define UDFs to Extract Category and Product Names

This cell defines two User Defined Functions (UDFs) to process the `category_code` column: - `extract_category`: - Extracts the main category from the first segment of `category_code`. - Falls back to `brand` if the category is `"empty"`, and returns `"unknown"` if both are `"empty"`. - `extract_product`: - Extracts the specific product type from the last segment of `category_code`. - Similar fallback logic as above applies when the segment is `"empty"`. These UDFs help enrich the dataset by generating more interpretable `category` and `product` labels.

[13]:
```
df_category_product_extracted = preprocessed_df.select("*",␣
 ↪extract_category("category_code", "brand"), ␣
 ↪extract_product("category_code", "brand"))
df_category_product_extracted = df_category_product_extracted.
 ↪withColumnRenamed("extract_category(category_code, brand)","category").
 ↪withColumnRenamed("extract_product(category_code, brand)", "product")
df_category_product_extracted = df_category_product_extracted.
 ↪drop("category_code")
if output_show:
```

```
df_category_product_extracted.show(5)
```

```
[Stage 7:>                                                          (0 + 1) / 1]
+-------------------+----------+----------+-------------------+-------+-------+-
--------+-------------------+-----------+-----------+
|         event_time|event_type|product_id|        category_id|  brand|  price|
user_id|        user_session|   category|    product|
+-------------------+----------+----------+-------------------+-------+-------+-
--------+-------------------+-----------+-----------+
|2019-11-07 04:15:14|      view|   1005160|2053013555631882655| xiaomi|
210.53|513169571|f5d3f5b0-9da8-4cd…|electronics| smartphone|
|2019-11-05 12:06:16|      view|   1004159|2053013555631882655|samsung|1253.06|5
16325949|6711f48b-6255-454…|electronics| smartphone|
|2019-11-17 07:24:15|      view|  15900033|2053013558190408249|  tefal|
27.0|572468337|c66cddf2-e8bd-479…|      tefal|      tefal|
|2019-11-17 14:16:50|      view|  34500013|2061717948468297988|  empty|
6.18|524042389|87dd26f0-8a1a-479…|    unknown|    unknown|
|2019-11-16 09:33:21|      view|   8700375|2053013563097744201|philips|
47.34|525988370|c3fee0d5-03db-478…| appliances|hair_cutter|
+-------------------+----------+----------+-------------------+-------+-------+-
--------+-------------------+-----------+-----------+
only showing top 5 rows
```

0.1.4 Apply UDFs and Refine DataFrame

This cell applies the previously defined UDFs to enrich the DataFrame: - Adds two new columns: `category` and `product`, derived from `category_code` and `brand`. - Renames the resulting columns to meaningful names (`category`, `product`). - Drops the original `category_code` column as it's no longer needed. - Displays the first 5 rows if `output_show` is set to `True`.

[14]:
```
df_time = df_category_product_extracted.withColumn('Date',␣
 ↪split(preprocessed_df['event_time'], ' ').getItem(0)).withColumn('Time',␣
 ↪split(preprocessed_df['event_time'], ' ').getItem(1))
df_time = df_time.withColumn('Day', split(df_time['Date'], '-').getItem(2)).
 ↪withColumn('Hour', split(df_time['Time'], ':').getItem(0))
df_time = df_time.drop("Date")
if output_show:
  df_time.show(5)
```

```
+-------------------+----------+----------+-------------------+-------+-------+-
--------+-------------------+-----------+-----------+--------+---+----+
|         event_time|event_type|product_id|        category_id|  brand|  price|
user_id|        user_session|   category|    product|    Time|Day|Hour|
+-------------------+----------+----------+-------------------+-------+-------+-
--------+-------------------+-----------+-----------+--------+---+----+
```

```
|2019-11-07 04:15:14|      view|   1005160|2053013555631882655| xiaomi|
210.53|513169571|f5d3f5b0-9da8-4cd…|electronics| smartphone|04:15:14| 07|  04|
|2019-11-05 12:06:16|      view|   1004159|2053013555631882655|samsung|1253.06|5
16325949|6711f48b-6255-454…|electronics| smartphone|12:06:16| 05|  12|
|2019-11-17 07:24:15|      view|  15900033|2053013558190408249|  tefal|
27.0|572468337|c66cddf2-e8bd-479…|      tefal|      tefal|07:24:15| 17|  07|
|2019-11-17 14:16:50|      view|  34500013|2061717948468297988|  empty|
6.18|524042389|87dd26f0-8a1a-479…|    unknown|    unknown|14:16:50| 17|  14|
|2019-11-16 09:33:21|      view|   8700375|2053013563097744201|philips|
47.34|525988370|c3fee0d5-03db-478…| appliances|hair_cutter|09:33:21| 16|  09|
+-------------------+----------+----------+-------------------+-------+-------+-
--------+--------------------+-----------+-----------+--------+---+----+
only showing top 5 rows
```

0.1.5 Extract Day and Hour from Event Timestamp

This cell enriches the DataFrame with temporal features: - Splits the `event_time` into `Date` and `Time` components. - Further extracts the `Day` from the `Date` and the `Hour` from the `Time`. - Drops the intermediate `Date` column to reduce redundancy. - Displays the first 5 rows if `output_show` is `True`.

[15]:
```
df = df_time
```

Unique Visitors in November

[16]:
```
unique_visitors = df.select(countDistinct("user_id"))
print(unique_visitors.head())
```

```
[Stage 11:=============================================>              (3 + 1) / 4]

Row(count(DISTINCT user_id)=1240125)
```

Analysis The e-commerce site has a footfall of **X unique visitors** in the month of October.

0.1.6 Future Scope

If geographic coordinates (latitude/longitude) are captured in future data collection, demographic insights such as regional trends and customer localization can be derived to enhance targeted marketing and business strategy.

Journey of a user in one session

[17]:
```
df.select("user_session", "event_time").show(5, truncate=False)
```

```
+------------------------------------+-------------------+
|user_session                        |event_time         |
+------------------------------------+-------------------+
|f5d3f5b0-9da8-4cd0-8682-fd360fa20cee|2019-11-07 04:15:14|
```

```
|6711f48b-6255-4544-aa92-54796163870c|2019-11-05 12:06:16|
|c66cddf2-e8bd-4793-9fee-7f0aad2847a8|2019-11-17 07:24:15|
|87dd26f0-8a1a-4791-a465-b75992774f89|2019-11-17 14:16:50|
|c3fee0d5-03db-478a-bb55-1e47a382b908|2019-11-16 09:33:21|
+------------------------------------+-------------------+
only showing top 5 rows
```

[20]:
```
df.filter(df.user_session=='f5d3f5b0-9da8-4cd0-8682-fd360fa20cee').
↪orderBy("event_time").toPandas()
```

[20]:

	event_time	event_type	product_id	category_id	brand	price	user_id	user_session	category	product	Time	Day	Hour
0	2019-11-07 04:15:14	view	1005160	2053013555631882655	xiaomi	210.53	513169571	f5d3f5b0-9da8-4cd0-8682-fd360fa20cee	electronics	smartphone	04:15:14	07	04

0.1.7 Inspect a Specific User Session

This cell filters the dataset to focus on a specific session (`user_session == '4d3b30da-a5e4-49df-b1a8-ba5943f1dd33'`): - Retrieves all events associated with that session. - Orders the events chronologically by `event_time`. - Converts the Spark DataFrame to a pandas DataFrame for easier inspection and display.

Analysis of User Behaviour on the e-commerce site

[21]:
```
df_view = df[df['event_type'] == "view"]
df_cart = df[df['event_type'] == "cart"]
df_purchase = df[df['event_type'] == "purchase"]
```

[22]:
```
data = dict(
    count=[df_view.count(), df_cart.count(), df_purchase.count()],
    event_type=["View", "Cart", "Purchase"])
fig = px.funnel(data, x='event_type', y='count')
fig.show()
```

```
/home/sp8201_nyu_edu/.local/lib/python3.11/site-
packages/plotly/io/_renderers.py:55: UserWarning:

Plotly version >= 6 requires Jupyter Notebook >= 7 but you have 6.5.7 installed.
 To upgrade Jupyter Notebook, please run `pip install notebook --upgrade`.
```

0.1.8 Visualize Event Funnel: View → Cart → Purchase

This cell creates a funnel chart to visualize user drop-off through the e-commerce journey: - `count` values are collected for each event type: View, Cart, and Purchase. - A funnel plot is generated using Plotly Express to illustrate how users progress (or drop off) from viewing a product to making a purchase.

0.1.9 Part 1: Determine Best Performing Categories Based on Purchases

Top 10 Categories Purchased This analysis highlights the product categories with the highest number of successful purchases, indicating strong customer demand and conversion efficiency.

Top 10 Categories Browsed This section reveals the categories that attracted the most customer attention (via views or cart additions), offering insight into popular interest even if not all led to purchases.

[23]:
```
df_cat_browsed = df.select("*").filter("event_type == 'view' OR event_type ==␣
↪'cart'")
df_cat_purchased = df.select("*").filter("event_type == 'purchase'")

df_cat_browsed = df_cat_browsed.select("*").filter("category!='unknown'")
df_cat_browsed = df_cat_browsed.select("*").filter("product!='unknown'")

df_cat_purchased = df_cat_purchased.select("*").filter("category!='unknown'")
df_cat_purchased = df_cat_purchased.select("*").filter("product!='unknown'")
```

0.1.10 Filter Browsed and Purchased Items by Known Categories and Products

This cell prepares two filtered DataFrames: - `df_cat_browsed`: Contains only 'view' and 'cart' events where both `category` and `product` are known (i.e., not `"unknown"`). - `df_cat_purchased`: Contains only 'purchase' events with known `category` and `product`.

These filtered datasets are used to analyze meaningful customer behavior across known product categories.

[24]:
```
df_cat_browsed_count = df_cat_browsed.groupBy("category").count().
↪orderBy(desc("count")).limit(10)
df_cat_browsed_count = df_cat_browsed_count.withColumnRenamed("count",␣
↪"category_browsed_count")
```

[25]:
```
if output_show:
  df_cat_browsed_count.show(5)
```

```
[Stage 33:============================>                            (2 + 2) / 4]

+-----------+----------------------+
|   category|category_browsed_count|
+-----------+----------------------+
|electronics|               1043473|
| appliances|                373208|
```

```
|  computers|                184793|
|    apparel|                133068|
|  furniture|                 93787|
+-----------+---------------------+
only showing top 5 rows
```

[26]:
```
browsed_category = [val.category for val in df_cat_browsed_count.
 ↪select('category').collect()]
browsed_count = [val.category_browsed_count for val in df_cat_browsed_count.
 ↪select('category_browsed_count').collect()]
d = {'browsed_category': browsed_category, 'browsed_count': browsed_count}
fig = px.bar(d, x="browsed_category", y="browsed_count", title="Top 10␣
 ↪Categories Browsed", text_auto='.2s')
fig.show()
```

```
/home/sp8201_nyu_edu/.local/lib/python3.11/site-
packages/plotly/io/_renderers.py:55: UserWarning:

Plotly version >= 6 requires Jupyter Notebook >= 7 but you have 6.5.7 installed.
 To upgrade Jupyter Notebook, please run `pip install notebook --upgrade`.
```

0.1.11 Bar Chart of Top 10 Categories Browsed

This cell visualizes user interest across product categories: - Extracts `category` names and their corresponding browse counts from `df_cat_browsed_count`. - Constructs a dictionary `d` to hold the data. - Uses Plotly Express to generate a bar chart showing the top 10 most browsed categories. - Displays the count values directly on the bars using `text_auto`.

[27]:
```
df_cat_purchased_count = df_cat_purchased.groupBy("category").count().
 ↪orderBy(desc("count")).limit(10)
df_cat_purchased_count = df_cat_purchased_count.withColumnRenamed("count",␣
 ↪"category_purchase_count")
```

0.1.12 Top 10 Purchased Categories

This cell identifies the most purchased product categories: - Groups the `df_cat_purchased` DataFrame by `category` and counts the number of purchase events. - Orders the categories by descending count and limits the result to the top 10. - Renames the count column to `category_purchase_count` for clarity.

[28]:
```
if output_show:
  df_cat_purchased_count.show()
```

```
[Stage 42:=============================================>             (3 + 1) / 4]
```

```
+------------+----------------------+
|    category|category_purchase_count|
+------------+----------------------+
| electronics|                 22054|
|  appliances|                  4358|
|   computers|                  1545|
|    cordiant|                   724|
|     apparel|                   668|
|     lucente|                   643|
|        auto|                   520|
|      xiaomi|                   505|
|   furniture|                   487|
|construction|                   377|
+------------+----------------------+
```

[29]:
```
purchase_category = [val.category for val in df_cat_purchased_count.
 ↪select('category').collect()]
purchase_count = [val.category_purchase_count for val in df_cat_purchased_count.
 ↪select('category_purchase_count').collect()]
d = {'purchase_category': purchase_category, 'purchase_count': purchase_count}
fig = px.bar(d, x="purchase_category", y="purchase_count", title="Top 10␣
 ↪Categories Purchased", text_auto='.2s')
fig.show()
```

```
/home/sp8201_nyu_edu/.local/lib/python3.11/site-
packages/plotly/io/_renderers.py:55: UserWarning:

Plotly version >= 6 requires Jupyter Notebook >= 7 but you have 6.5.7 installed.
 To upgrade Jupyter Notebook, please run `pip install notebook --upgrade`.
```

0.1.13 Bar Chart of Top 10 Categories Purchased

This cell visualizes the most frequently purchased product categories: - Extracts `category` names and their corresponding purchase counts from `df_cat_purchased_count`. - Constructs a dictionary `d` with purchase data. - Uses Plotly Express to plot a bar chart showing the top 10 categories by purchase volume. - Displays count values on the bars using `text_auto` formatting.

Carted vs Purchased - Top Performing Products

[30]:
```
df_cat_carted = df.select("*").filter("event_type == 'cart'")
df_cp_cart_count = df_cat_carted.groupBy("category","product").count().
 ↪orderBy(desc("count"))
df_cp_cart_count = df_cp_cart_count.withColumnRenamed("count", "cart_count")
if output_show:
  df_cp_cart_count.show(5)
```

```
[Stage 51:=============================================>             (3 + 1) / 4]
```

```
+----------+---------+----------+
|  category|  product|cart_count|
+----------+---------+----------+
|electronics|smartphone|     51553|
|   unknown|  unknown|      7715|
|electronics| headphone|      5855|
|electronics|       tv|      4752|
| appliances|   washer|      3117|
+----------+---------+----------+
only showing top 5 rows
```

0.1.14 Count of Carted Items by Category and Product

This cell analyzes products added to cart: - Filters the dataset to include only 'cart' events. - Groups the data by `category` and `product`, then counts how many times each combination was added to cart. - Renames the resulting `count` column to `cart_count` for clarity. - Displays the top 5 results if `output_show` is `True`.

[31]:
```
df_cp_purchase_count = df_cat_purchased.groupBy("category","product").count().
  ↪orderBy(desc("count"))
df_cp_purchase_count = df_cp_purchase_count.withColumnRenamed("count",␣
  ↪"purchase_count")
if output_show:
  df_cp_purchase_count.show()
```

```
[Stage 54:=============================================>             (3 + 1) / 4]
```

```
+----------+-------------+--------------+
|  category|      product|purchase_count|
+----------+-------------+--------------+
|electronics|   smartphone|         17104|
|electronics|    headphone|          1795|
|electronics|           tv|          1290|
|electronics|       clocks|          1052|
| appliances|       washer|           867|
| appliances|       vacuum|           820|
|  computers|     notebook|           816|
|   cordiant|     cordiant|           724|
|    lucente|      lucente|           643|
| appliances|refrigerators|           572|
|     xiaomi|       xiaomi|           505|
|    apparel|        shoes|           492|
|electronics|       tablet|           320|
|       sony|         sony|           317|
|     nokian|       nokian|           269|
```

```
|     viatti|       viatti|           247|
| appliances|    microwave|           233|
|   triangle|     triangle|           221|
|electronics|    telephone|           210|
| appliances|         iron|           197|
+-----------+-------------+--------------+
only showing top 20 rows
```

[32]:
```
df_product = df_cp_cart_count.join(df_cp_purchase_count, ["category",␣
  ↪"product"], "left").orderBy(desc("purchase_count")).limit(20)
if output_show:
  df_product.show()
```

```
+-----------+-------------+----------+--------------+
|   category|      product|cart_count|purchase_count|
+-----------+-------------+----------+--------------+
|electronics|   smartphone|     51553|         17104|
|electronics|    headphone|      5855|          1795|
|electronics|           tv|      4752|          1290|
|electronics|       clocks|      3083|          1052|
| appliances|       washer|      3117|           867|
| appliances|       vacuum|      2883|           820|
|  computers|     notebook|      2631|           816|
|   cordiant|     cordiant|      2778|           724|
|    lucente|      lucente|      1698|           643|
| appliances|refrigerators|      2267|           572|
|     xiaomi|       xiaomi|      1643|           505|
|    apparel|        shoes|      1847|           492|
|electronics|       tablet|      1025|           320|
|       sony|         sony|      1052|           317|
|     nokian|       nokian|       986|           269|
|     viatti|       viatti|       914|           247|
| appliances|    microwave|       675|           233|
|   triangle|     triangle|       677|           221|
|electronics|    telephone|       640|           210|
| appliances|         iron|       648|           197|
+-----------+-------------+----------+--------------+
```

[33]:
```
#Getting values for x-axis and y-axis
product = [val.product for val in df_product.select('product').collect()]
cart_count = [val.cart_count for val in df_product.select('cart_count').
  ↪collect()]
```

```
purchase_count = [val.purchase_count for val in df_product.
 ↪select('purchase_count').collect()]
```

0.1.15 Extract Data for Product-Level Cart and Purchase Analysis

This cell prepares data for plotting or further analysis: - Retrieves lists of product names, corresponding cart counts, and purchase counts from the `df_product` DataFrame. - These lists are typically used as x-axis and y-axis values for visual comparisons (e.g., bar charts or scatter plots).

[34]:
```
import plotly.graph_objects as go

fig = go.Figure(data=[
    go.Bar(name='Cart Count', x=product, y=cart_count),
    go.Bar(name='Purchase Count', x=product, y=purchase_count)
])
# Change the bar mode
fig.update_layout(barmode='group', title_text='Carted vs Purchase Count - Top␣
 ↪Performing Products', xaxis_title = "Products", yaxis_title = 'Count')
fig.show()
```

0.1.16 Grouped Bar Chart: Carted vs Purchased Products

This cell visualizes the relationship between cart activity and actual purchases: - Uses Plotly's `go.Figure` to create grouped bar charts. - Each product is shown along the x-axis with its respective cart and purchase counts as separate bars. - The chart helps compare how frequently products are carted versus purchased, providing insights into conversion behavior.

Top 5 Products Purchased in each Category

[35]:
```
from pyspark.sql.window import Window
window = Window.partitionBy(df_cp_purchase_count['category']).
 ↪orderBy(df_cp_purchase_count['purchase_count'].desc())
ranked =  df_cp_purchase_count.select("*", rank().over(window).alias("rank")).
 ↪filter(col('rank')<=5)
if output_show:
  ranked.orderBy(desc("count")).show(20)
```

```
+-----------+-------------+--------------+----+
|   category|      product|purchase_count|rank|
+-----------+-------------+--------------+----+
|electronics|   smartphone|         17104|   1|
|electronics|    headphone|          1795|   2|
|electronics|           tv|          1290|   3|
|electronics|       clocks|          1052|   4|
| appliances|       washer|           867|   1|
```

```
| appliances|       vacuum|           820|   2|
|  computers|     notebook|           816|   1|
|   cordiant|     cordiant|           724|   1|
|    lucente|      lucente|           643|   1|
| appliances|refrigerators|           572|   3|
|     xiaomi|       xiaomi|           505|   1|
|    apparel|        shoes|           492|   1|
|electronics|       tablet|           320|   5|
|       sony|         sony|           317|   1|
|     nokian|       nokian|           269|   1|
|     viatti|       viatti|           247|   1|
| appliances|    microwave|           233|   4|
|   triangle|     triangle|           221|   1|
| appliances|         iron|           197|   5|
|       auto|       player|           185|   1|
+-----------+-------------+--------------+----+
only showing top 20 rows
```

Analysis: We can see that within electronics, smartphones are the most purchased products

Top 5 Brands Browsed within each category

[36]:
```
df_cb_browsed_count = df_cat_browsed.groupBy("category","brand").count().
  ↪orderBy(desc("count"))
if output_show:
  df_cb_browsed_count.show(5)
```

```
[Stage 93:=============================================>              (3 + 1) / 4]
+-----------+-------+------+
|   category|  brand| count|
+-----------+-------+------+
|electronics|samsung|285565|
|electronics|  apple|259019|
|electronics| xiaomi|168822|
|electronics| huawei| 60328|
|    lucente|lucente| 51683|
+-----------+-------+------+
only showing top 5 rows
```

[37]:
```
window = Window.partitionBy(df_cb_browsed_count['category']).
  ↪orderBy(df_cb_browsed_count['count'].desc())
ranked =  df_cb_browsed_count.select("*", rank().over(window).alias("rank")).
  ↪filter(col('rank')<=5)
```

```
if output_show:
  ranked.orderBy(desc("count")).show(5)
```

```
+-----------+-------+------+----+
|   category|  brand| count|rank|
+-----------+-------+------+----+
|electronics|samsung|285565|   1|
|electronics|  apple|259019|   2|
|electronics| xiaomi|168822|   3|
|electronics| huawei| 60328|   4|
|    lucente|lucente| 51683|   1|
+-----------+-------+------+----+
only showing top 5 rows
```

Analysis: We can see that within electronics, samsung is most browsed brand

Top 5 brands within each category Purchased

[38]:
```
df_cb_purchased_count = df_cat_purchased.groupBy("category","brand").count().
  ↪orderBy(desc("count"))
if output_show:
  df_cb_purchased_count.show(5)
```

```
[Stage 108:=============================================>             (3 + 1) / 4]
```

```
+-----------+--------+-----+
|   category|   brand|count|
+-----------+--------+-----+
|electronics| samsung| 8143|
|electronics|   apple| 7305|
|electronics|  xiaomi| 2543|
|electronics|  huawei| 1122|
|   cordiant|cordiant|  724|
+-----------+--------+-----+
only showing top 5 rows
```

[39]:
```
window = Window.partitionBy(df_cb_purchased_count['category']).
  ↪orderBy(df_cb_purchased_count['count'].desc())
ranked =  df_cb_purchased_count.select("*", rank().over(window).alias("rank")).
  ↪filter(col('rank')<=5)
if output_show:
  ranked.orderBy(desc("count")).show(5)
```

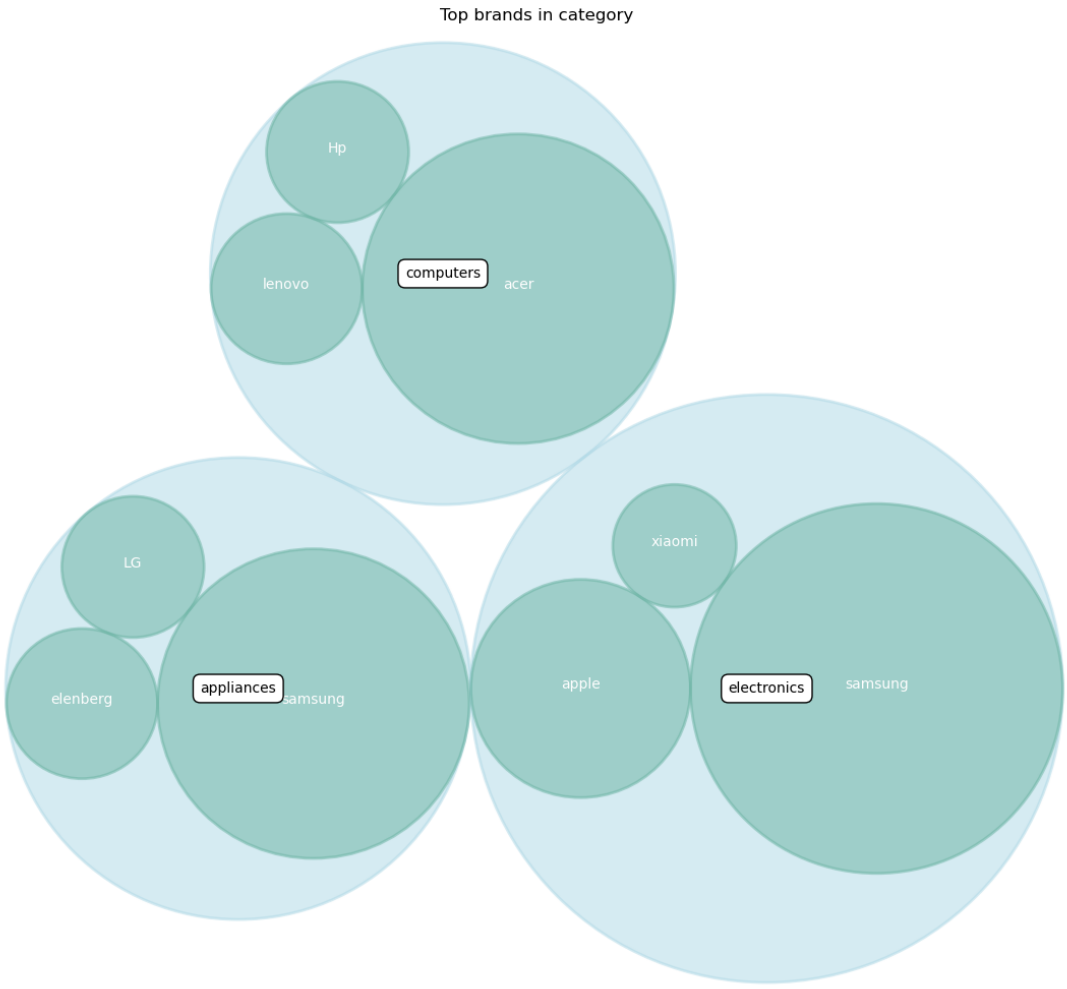
```
+----------+--------+-----+----+
|   category|   brand|count|rank|
+----------+--------+-----+----+
|electronics| samsung| 8143|   1|
|electronics|   apple| 7305|   2|
|electronics|  xiaomi| 2543|   3|
|electronics|  huawei| 1122|   4|
|   cordiant|cordiant|  724|   1|
+----------+--------+-----+----+
only showing top 5 rows
```

[40]:
```
import circlify
data = [{'id': 'World', 'datum': 6964195249, 'children' : [
              {'id' : "electronics", 'datum': 450448697,
                   'children' : [
                     {'id' : "samsung", 'datum' : 308865000},
                     {'id' : "apple", 'datum' : 107550697},
                     {'id' : "xiaomi", 'datum' : 34033000}
                   ]},
              {'id' : "appliances", 'datum' : 278095425,
                   'children' : [
                     {'id' : "samsung", 'datum' : 192612000},
                     {'id' : "elenberg", 'datum' : 45349000},
                     {'id' : "LG", 'datum' : 40134425}
                   ]},
              {'id' : "computers", 'datum' : 278095425,
              'children' : [
                {'id' : "acer", 'datum' : 192612000},
                {'id' : "lenovo", 'datum' : 45349000},
                {'id' : "Hp", 'datum' : 40134425}
              ]},
    ]}]
```

[41]:
```
# Compute circle positions thanks to the circlify() function
circles = circlify.circlify(
    data,
    show_enclosure=False,
    target_enclosure=circlify.Circle(x=0, y=0, r=1)
)
```

[42]:
```
# import libraries
import circlify
import matplotlib.pyplot as plt
import numpy as np
# Create just a figure and only one subplot
fig, ax = plt.subplots(figsize=(14,14))
```

```
# Title
ax.set_title('Top brands in category')

# Remove axes
ax.axis('off')

# Find axis boundaries
lim = -1
for circle in circles:
  k = np.max([np.abs(circle.x) + circle.r, np.abs(circle.y) + circle.r])
  lim = np.max([lim, k])

plt.xlim(-lim, lim)
plt.ylim(-lim, lim)

# Print circle the highest level (continents):
for circle in circles:
    if circle.level != 2:
      continue
    x, y, r = circle
    ax.add_patch( plt.Circle((x, y), r, alpha=0.5, linewidth=2,␣
 ↪color="lightblue"))

# Print circle and labels for the highest level:
for circle in circles:
    if circle.level != 3:
      continue
    x, y, r = circle
    label = circle.ex["id"]
    ax.add_patch( plt.Circle((x, y), r, alpha=0.5, linewidth=2,␣
 ↪color="#69b3a2"))
    plt.annotate(label, (x,y ), ha='center', color="white")

# Print labels for the continents
for circle in circles:
    if circle.level != 2:
      continue
    x, y, r = circle
    label = circle.ex["id"]
    plt.annotate(label, (x,y ) ,va='center', ha='center',␣
 ↪bbox=dict(facecolor='white', edgecolor='black', boxstyle='round', pad=.5))
```



Part 2: Evaluate Add to Cart and Cart Abandonment Rate

Cart Abandonment Rate

[43]:
```
df_only_cart = df.filter(df.event_type=="cart")
df_only_purchase = df.filter(df.event_type=="purchase")
```

CART by Category

[44]:
```
#Distinct category in cart
df_only_cart_cat_dis = df_only_cart.groupBy("category").count()
df_only_cart_cat_dis = df_only_cart_cat_dis.withColumnRenamed("count",␣
 ↪"cart_count")
```

```
#Distinct category in purchase
df_only_purchase_cat_dis = df_only_purchase.groupBy("category").count()
df_only_purchase_cat_dis = df_only_purchase_cat_dis.withColumnRenamed("count",␣
 ↪"purchase_count")

#Performing a left join
columns_ = df.columns
cart_purchase = df_only_cart_cat_dis.alias("cart_cat").
 ↪join(df_only_purchase_cat_dis.alias("purchase_cat"),

 ↪df_only_cart_cat_dis["category"]==df_only_purchase_cat_dis["category"]).
 ↪select("cart_cat.category", "cart_cat.cart_count", "purchase_cat.
 ↪purchase_count")
if output_show:
  cart_purchase.show(5)
```

```
+----------+----------+--------------+
|  category|cart_count|purchase_count|
+----------+----------+--------------+
|   matador|       462|           120|
|marcomenti|         6|             4|
|  imperial|        28|            13|
| remington|        76|            16|
|    ritmix|        15|             9|
+----------+----------+--------------+
only showing top 5 rows
```

[45]:
```
#Creating a UDF for calculating Cart Abandonment Rate
@udf(returnType=FloatType())
def cart_miss_rate_udf(cart_count, purchase_count):
  rate = (1 - purchase_count/cart_count)*100
  #rate_string = "{:.2f}".format(rate)
  return rate
```

[46]:
```
df_cart_miss_rate = cart_purchase.select("*", cart_miss_rate_udf("cart_count",␣
 ↪"purchase_count"))
df_cart_miss_rate = df_cart_miss_rate.
 ↪withColumnRenamed("cart_miss_rate_udf(cart_count, purchase_count)",␣
 ↪"cart_miss_rate")
df_cart_miss_rate = df_cart_miss_rate.filter("cart_miss_rate>0")
df_cart_miss_rate = df_cart_miss_rate.filter("cart_count>5000")
df_cart_miss_rate = df_cart_miss_rate.orderBy(desc("cart_miss_rate")).limit(10)
if output_show:
  df_cart_miss_rate.show(5)
```

```
+-----------+----------+--------------+--------------+
|   category|cart_count|purchase_count|cart_miss_rate|
+-----------+----------+--------------+--------------+
| appliances|     15603|          4358|      72.06947|
|    unknown|      7715|          2216|      71.27673|
|  computers|      5282|          1545|      70.74972|
|electronics|     67842|         22054|      67.49211|
+-----------+----------+--------------+--------------+
```

[47]:
```
category = [val.category for val in df_cart_miss_rate.select('category').
 ↪collect()]
cart_miss_rate_list = [val.cart_miss_rate for val in df_cart_miss_rate.
 ↪select('cart_miss_rate').collect()]

d = {'category':category, 'cart_miss_rate': cart_miss_rate_list}
fig = px.pie(d, values='cart_miss_rate', names='category', title="Cart␣
 ↪Abandonment Rate for category")
fig.show()
```

```
/home/sp8201_nyu_edu/.local/lib/python3.11/site-
packages/plotly/io/_renderers.py:55: UserWarning:

Plotly version >= 6 requires Jupyter Notebook >= 7 but you have 6.5.7 installed.
 To upgrade Jupyter Notebook, please run `pip install notebook --upgrade`.
```

Analysis: We can see that X category had the most cart abandonment rate with X%

CAR by Brands

[48]:
```
# distinct brands in cart

df_only_cart_brand_dis = df_only_cart.filter("category=='electronics'").
 ↪groupBy("brand").count()
df_only_cart_brand_dis = df_only_cart_brand_dis.withColumnRenamed("count",␣
 ↪"cart_count")

# distinct brands in purchase
df_only_purchase_brand_dis = df_only_purchase.filter("category=='electronics'").
 ↪groupBy("brand").count()
df_only_purchase_brand_dis = df_only_purchase_brand_dis.
 ↪withColumnRenamed("count", "purchase_count")

#Performing a left join
cart_purchase_brand = df_only_cart_brand_dis.alias("cart_cat").
 ↪join(df_only_purchase_brand_dis.alias("purchase_cat"),
```

```
 ↪df_only_cart_brand_dis["brand"]==df_only_purchase_brand_dis["brand"]).
 ↪select("cart_cat.brand", "cart_cat.cart_count", "purchase_cat.
 ↪purchase_count")
if output_show:
  cart_purchase_brand.show(5)
```

```
+---------+----------+--------------+
|    brand|cart_count|purchase_count|
+---------+----------+--------------+
|   ritmix|        26|             5|
|panasonic|        99|            24|
|      tcl|       189|            62|
|   armani|        29|            10|
| rockdale|         3|             4|
+---------+----------+--------------+
only showing top 5 rows
```

[49]:
```
df_cart_miss_rate_brand = cart_purchase_brand.select("*",
 ↪cart_miss_rate_udf("cart_count", "purchase_count"))
df_cart_miss_rate_brand = df_cart_miss_rate_brand.
 ↪withColumnRenamed("cart_miss_rate_udf(cart_count, purchase_count)",
 ↪"cart_miss_rate_brand")
df_cart_miss_rate_brand = df_cart_miss_rate_brand.
 ↪filter("cart_miss_rate_brand>0")
df_cart_miss_rate_brand = df_cart_miss_rate_brand.filter("cart_count>5000")
df_cart_miss_rate_brand = df_cart_miss_rate_brand.
 ↪orderBy(desc("cart_miss_rate_brand")).limit(10)
df_cart_miss_rate_brand.toPandas()
```

[49]:

	brand	cart_count	purchase_count	cart_miss_rate_brand
0	xiaomi	9555	2543	73.385658
1	apple	21438	7305	65.924995
2	samsung	23782	8143	65.759819

[50]:
```
brand = [val.brand for val in df_cart_miss_rate_brand.select('brand').collect()]
cart_miss_rate_brand_list = [val.cart_miss_rate_brand for val in
 ↪df_cart_miss_rate_brand.select('cart_miss_rate_brand').collect()]

d = {'brand':brand, 'cart_miss_rate': cart_miss_rate_brand_list}
fig = px.pie(d, values='cart_miss_rate', names='brand', title="Cart Abandonment
 ↪Rate for brands")
```

```
#fig.update_traces(textinfo='value')
fig.show()
```

```
/home/sp8201_nyu_edu/.local/lib/python3.11/site-
packages/plotly/io/_renderers.py:55: UserWarning:

Plotly version >= 6 requires Jupyter Notebook >= 7 but you have 6.5.7 installed.
 To upgrade Jupyter Notebook, please run `pip install notebook --upgrade`.
```

Analysis: Show brands with highest abandonment rate. Business users need to talk to brand people to work on their campaign.

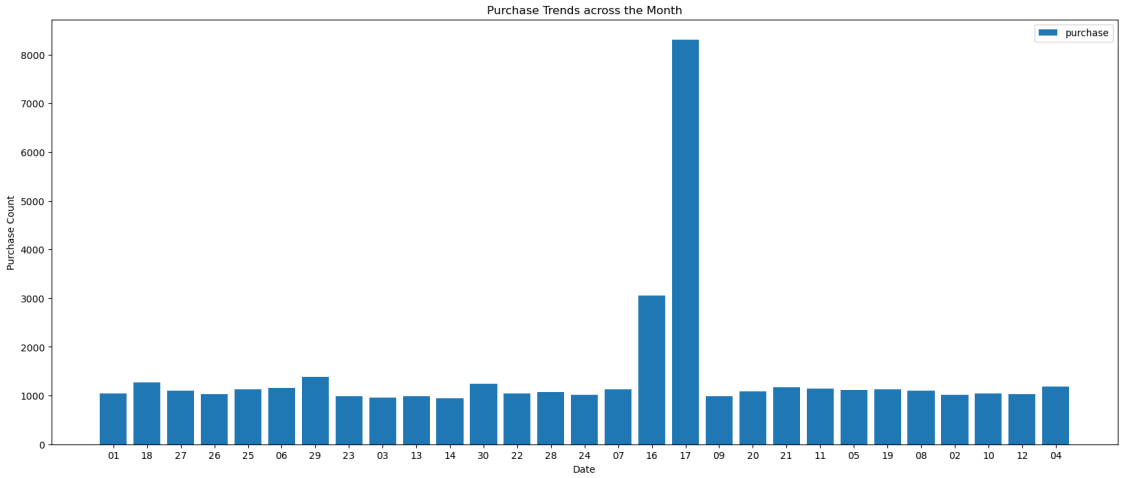
Part 3: Effect of day-time on purchase trends

Purchase trends across the month

[51]:
```
df_view = df[df['event_type'] == "view"]
df_purchase = df[df['event_type'] == "purchase"]
df_cart = df[df['event_type'] == "cart"]
```

[52]:
```
df_purchase_date_count = df_purchase.groupby("Day").count()
```

[53]:
```
import pandas as pd
import matplotlib.pyplot as plt

count = [val[0] for val in df_purchase_date_count.select('count').collect()]
date = [val.Day for val in df_purchase_date_count.select('Day').collect()]

plt.figure(figsize=(20, 8))
plt.bar(date, count)

plt.ylabel('Purchase Count')
plt.xlabel('Date')
plt.title('Purchase Trends across the Month')
plt.legend(['purchase'], loc='upper right')

plt.show()
```



Analysis: User's buying interest is highest in the middle of the month on day 17, therefore, to increase the sales we can offer mid-month sale/discount from day 11 until 17

E-Commerce Prime Time

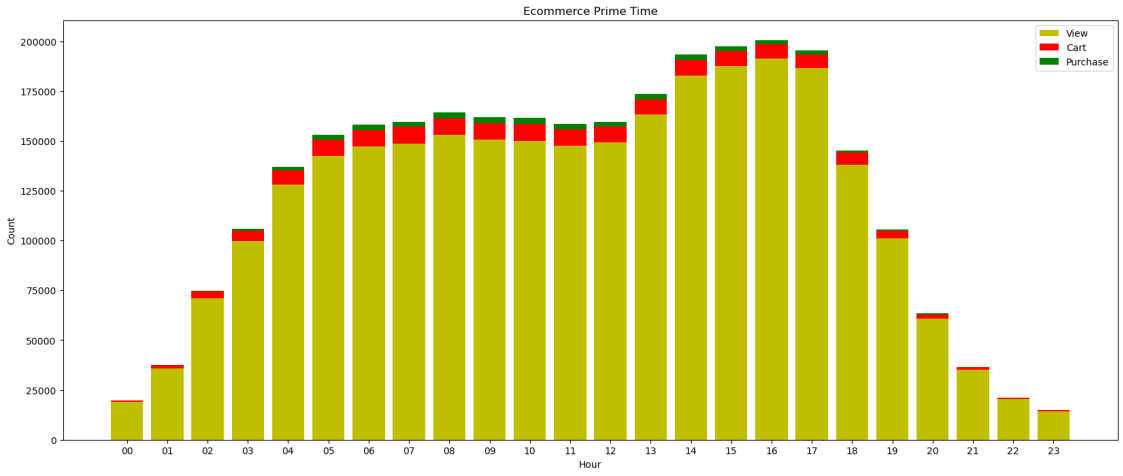
[57]:
```
df_view_hour_count = df_view.groupby("Hour").count().
 ↪withColumnRenamed("count","view_count")
df_cart_hour_count = df_cart.groupby("Hour").count().
 ↪withColumnRenamed("count","cart_count")
df_purchase_hour_count = df_purchase.groupby("Hour").count().
 ↪withColumnRenamed("count","purchase_count")
```

[58]:
```
df_combined_type_hour = df_view_hour_count.join(df_cart_hour_count,␣
 ↪["Hour"],"left")
df_combined_type_hour = df_combined_type_hour.join(df_purchase_hour_count,␣
 ↪["Hour"], "left")
df_combined_type_hour = df_combined_type_hour.na.fill(value=0).orderBy('Hour')
```

[59]:
```
#purchase_count
hour = [val.Hour for val in df_combined_type_hour.select('Hour').collect()]
view_count = [val.view_count for val in df_combined_type_hour.
 ↪select('view_count').collect()]
cart_count = [val.cart_count for val in df_combined_type_hour.
 ↪select('cart_count').collect()]
purchase_count = [val.purchase_count for val in df_combined_type_hour.
 ↪select('purchase_count').collect()]
```

[60]:
```
#Data visualization of Ecommerce Prime Time
import pandas as pd
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 8))
plt.bar(hour, view_count, color='y')
plt.bar(hour, cart_count, bottom=view_count, color='r')
plt.bar(hour, purchase_count, bottom=[view + cart for view, cart in␣
 ↪zip(view_count, cart_count)], color='g')
plt.xlabel("Hour")
plt.ylabel("Count")
plt.legend(["View", "Cart", "Purchase"])
plt.title("Ecommerce Prime Time")
plt.show()
```



Analysis: We can see from the graph that X number of users have already accessed our Ecommerce at 3:00 In the morning, it is increasing significantly in the afternoon and reached peak time at 16:00. Hence, a flash sale from 13:00 until 16:00 will help in increasing the impulsivity of the user for buying items

Part 4: Predict whether the product added to cart is actually purchased?

Data Processing

[61]:
```
df.groupBy("event_type").count()
```

[61]:
```
DataFrame[event_type: string, count: bigint]
```

[62]:
```
#Unique number of people who have added products to the cart or purchase
cart_purchase_users = df.filter("event_type == 'cart' OR event_type ==␣
 ↪'purchase' ")
distinct_cart_purchase = cart_purchase_users.
 ↪drop_duplicates(subset=['event_type', 'product_id','price',␣
 ↪'user_id','user_session'])
```

```
distinct_cart_purchase.groupby("event_type").count()
```

[62]: DataFrame[event_type: string, count: bigint]

[63]:
```
#All user activity for adding product to cart or purchased
columns_ = df.columns
cart_purchase_users_all_activity = df.alias("d").join(cart_purchase_users.
 ↪alias("c"), df["user_id"]==cart_purchase_users["user_id"]).select("d.
 ↪user_id", "d.event_time", "d.event_type", "d.product_id", "d.category_id",␣
 ↪"d.brand", "d.price", "d.user_session", "d.category", "d.product")
cart_purchase_users_all_activity.groupBy("event_type").count()
```

[63]: DataFrame[event_type: string, count: bigint]

High Value Customers

[64]:
```
activity_in_session = cart_purchase_users_all_activity.
 ↪groupby(['user_session']).count()
if output_show:
  activity_in_session.show(5)
```

```
[Stage 258:=============================================>             (3 + 1) / 4]

+--------------------+-----+
|        user_session|count|
+--------------------+-----+
|6963623a-59bb-4ba…|    9|
|8555afbf-dd0b-45b…|    1|
|cd77611e-fe24-4c1…|    4|
|ac5594a5-baba-48f…|    2|
|86e213c0-80ce-4cb…|    3|
+--------------------+-----+
only showing top 5 rows
```

Label Encoding Target Variable

[65]:
```
@udf(returnType=IntegerType())
def is_purchased_label(purchase):
  if purchase == "purchase":
    return 1
  return 0
```

[66]:
```
df_targets = distinct_cart_purchase.select("*",␣
 ↪is_purchased_label("event_type"))
df_targets = df_targets.
 ↪withColumnRenamed("is_purchased_label(event_type)","is_purchased")
```

[67]:
```
df_targets = df_targets.join(activity_in_session, on="user_session",how="left")
```

[68]:
```
from datetime import datetime
@udf(returnType=IntegerType())
def week(s):
  return datetime.strptime(str(s)[0:10], "%Y-%m-%d").weekday()
```

[69]:
```
df_targets_week = df_targets.select("*", week("event_time"))
df_targets_week = df_targets_week.withColumnRenamed("week(event_time)", "week")
```

[70]:
```
df_targets_week.printSchema()
```

```
root
 |-- user_session: string (nullable = true)
 |-- event_time: timestamp (nullable = true)
 |-- event_type: string (nullable = true)
 |-- product_id: integer (nullable = true)
 |-- category_id: long (nullable = true)
 |-- brand: string (nullable = false)
 |-- price: double (nullable = true)
 |-- user_id: integer (nullable = true)
 |-- category: string (nullable = true)
 |-- product: string (nullable = true)
 |-- Time: string (nullable = true)
 |-- Day: string (nullable = true)
 |-- Hour: string (nullable = true)
 |-- is_purchased: integer (nullable = true)
 |-- count: long (nullable = true)
 |-- week: integer (nullable = true)
```

[71]:
```
df_targets_week = df_targets_week.dropDuplicates(["user_session"])
```

Feature Selection

[72]:
```
features = df_targets_week.select("event_type", "brand", "price",␣
 ↪"count","week","category", "product","is_purchased")
features.printSchema()
```

```
root
 |-- event_type: string (nullable = true)
 |-- brand: string (nullable = false)
 |-- price: double (nullable = true)
 |-- count: long (nullable = true)
 |-- week: integer (nullable = true)
 |-- category: string (nullable = true)
 |-- product: string (nullable = true)
 |-- is_purchased: integer (nullable = true)
```

[73]:
```
features.count()
```

[73]: 165389

Prediction Model - SparkML

[74]:
```
from pyspark.ml.feature import StringIndexer, OneHotEncoder
from pyspark.ml.feature import VectorAssembler
from pyspark.ml import Pipeline

categotyIdxer = StringIndexer(inputCol='category',outputCol='category_idx')
event_typeIdxer =␣
 ↪StringIndexer(inputCol='event_type',outputCol='event_type_idx')
brandIdxer = StringIndexer(inputCol='brand',outputCol='brand_idx')
productIdxer = StringIndexer(inputCol='product',outputCol='product_idx')
labelIndexer = StringIndexer(inputCol="is_purchased", outputCol="label")

one_hot_encoder_category = OneHotEncoder(inputCol="category_idx",␣
 ↪outputCol="category_vec")
one_hot_encoder_product = OneHotEncoder(inputCol="product_idx",␣
 ↪outputCol="product_vec")
one_hot_encoder_brand = OneHotEncoder(inputCol="brand_idx",␣
 ↪outputCol="brand_vec")
one_hot_encoder_event_type = OneHotEncoder(inputCol="event_type_idx",␣
 ↪outputCol="event_type_vec")

stages_indexer = [categotyIdxer,
          event_typeIdxer,
          brandIdxer,
          productIdxer,
          labelIndexer]
stages_one_hot = [
          one_hot_encoder_category,
          one_hot_encoder_event_type,
          one_hot_encoder_brand,
          one_hot_encoder_product]


assembler_cat = VectorAssembler(inputCols=[encoder.getOutputCol() for encoder␣
 ↪in stages_one_hot], outputCol="features_cat")
num_cols = ["count", "week", "price"]
assemblerNum = VectorAssembler(inputCols = num_cols, outputCol = "features_num")
```

```
final_assembler = VectorAssembler(inputCols = ["features_cat", "features_num"],␣
 ↪outputCol = "features")
pipeline = Pipeline(stages = stages_indexer + stages_one_hot + [assembler_cat]␣
 ↪+ [assemblerNum]+ [final_assembler])
```

[75]:
```
features = features.na.drop()
```

[88]:
```
df_transformed = pipeline.fit(features).transform(features)
# if output_show:
#   df_transformed.show(2)
```

[89]:
```
final_data = df_transformed.select("features", "label")
final_data = final_data.na.drop()
final_data.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- label: double (nullable = false)
```

[90]:
```
(trainingData, testData) = final_data.randomSplit([0.7, 0.3])
```

[96]:
```
from pyspark.ml.classification import DecisionTreeClassifier,␣
 ↪RandomForestClassifier, GBTClassifier, LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import lit
```

0.2 Random Forest

[100]:
```
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier,␣
 ↪DecisionTreeClassifier, GBTClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql import SparkSession
import pandas as pd

# Initialize Spark session (if not already done)
spark = SparkSession.builder.appName("ClassifierComparison").getOrCreate()

# Define StringIndexer with handleInvalid="keep" to handle unseen labels
categotyIdxer = StringIndexer(inputCol='category', outputCol='category_idx',␣
 ↪handleInvalid='keep')
event_typeIdxer = StringIndexer(inputCol='event_type',␣
 ↪outputCol='event_type_idx', handleInvalid='keep')
```

```
brandIdxer = StringIndexer(inputCol='brand', outputCol='brand_idx',␣
 ↪handleInvalid='keep')
productIdxer = StringIndexer(inputCol='product', outputCol='product_idx',␣
 ↪handleInvalid='keep')
labelIndexer = StringIndexer(inputCol="is_purchased", outputCol="label",␣
 ↪handleInvalid='keep')

# Define OneHotEncoder
one_hot_encoder_category = OneHotEncoder(inputCol="category_idx",␣
 ↪outputCol="category_vec")
one_hot_encoder_product = OneHotEncoder(inputCol="product_idx",␣
 ↪outputCol="product_vec")
one_hot_encoder_brand = OneHotEncoder(inputCol="brand_idx",␣
 ↪outputCol="brand_vec")
one_hot_encoder_event_type = OneHotEncoder(inputCol="event_type_idx",␣
 ↪outputCol="event_type_vec")

# Define stages
stages_indexer = [categotyIdxer, event_typeIdxer, brandIdxer, productIdxer,␣
 ↪labelIndexer]
stages_one_hot = [one_hot_encoder_category, one_hot_encoder_event_type,␣
 ↪one_hot_encoder_brand, one_hot_encoder_product]

# VectorAssembler for categorical features
assembler_cat = VectorAssembler(inputCols=[encoder.getOutputCol() for encoder␣
 ↪in stages_one_hot], outputCol="features_cat")

# Numerical columns
num_cols = ["count", "week", "price"]
assemblerNum = VectorAssembler(inputCols=num_cols, outputCol="features_num")

# Final assembler for all features
final_assembler = VectorAssembler(inputCols=["features_cat", "features_num"],␣
 ↪outputCol="features")

# Pipeline
pipeline = Pipeline(stages=stages_indexer + stages_one_hot + [assembler_cat,␣
 ↪assemblerNum, final_assembler])

# Assuming 'features' is your input DataFrame
features = features.na.drop()
df_transformed = pipeline.fit(features).transform(features)
final_data = df_transformed.select("features", "label")
final_data = final_data.na.drop()

# Split data into training and test sets
```

```
(trainingData, testData) = final_data.randomSplit([0.7, 0.3], seed=42)

# Define evaluators
evaluator = MulticlassClassificationEvaluator(labelCol="label",␣
 ↪predictionCol="prediction", metricName="accuracy")

# Initialize classifiers
rf = RandomForestClassifier(labelCol="label", featuresCol="features",␣
 ↪numTrees=100, seed=42)
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features", seed=42)
gbt = GBTClassifier(labelCol="label", featuresCol="features", maxIter=100,␣
 ↪seed=42)

# Dictionary to store models
models = {
    "RandomForest": rf,
    "DecisionTree": dt,
    "GradientBoostedTrees": gbt
}

# List to store results
results = []

# Train and evaluate each model
for model_name, model in models.items():
    # Train the model
    fitted_model = model.fit(trainingData)

    # Predictions on training data
    train_predictions = fitted_model.transform(trainingData)
    train_accuracy = evaluator.evaluate(train_predictions)

    # Predictions on test data
    test_predictions = fitted_model.transform(testData)
    test_accuracy = evaluator.evaluate(test_predictions)

    # Append results
    results.append({
        "Model": model_name,
        "Train_Accuracy": train_accuracy,
        "Test_Accuracy": test_accuracy
    })

# Convert results to Spark DataFrame
results_df = spark.createDataFrame(pd.DataFrame(results))

# Show the results
```

```
results_df.show(truncate=False)

# Optionally, convert to pandas for better formatting
results_pandas = results_df.toPandas()
print(results_pandas)

# Stop Spark session (optional)
spark.stop()
```

```
25/05/08 06:29:03 WARN DAGScheduler: Broadcasting large task binary with size
1167.9 KiB
25/05/08 06:29:40 WARN DAGScheduler: Broadcasting large task binary with size
1219.4 KiB
25/05/08 06:30:21 WARN DAGScheduler: Broadcasting large task binary with size
1256.4 KiB
25/05/08 06:31:10 WARN DAGScheduler: Broadcasting large task binary with size
1291.1 KiB
25/05/08 06:31:47 WARN DAGScheduler: Broadcasting large task binary with size
1326.2 KiB
25/05/08 06:32:51 WARN DAGScheduler: Broadcasting large task binary with size
1263.5 KiB
25/05/08 06:33:06 WARN DAGScheduler: Broadcasting large task binary with size
1263.5 KiB
25/05/08 06:33:35 WARN DAGScheduler: Broadcasting large task binary with size
1134.4 KiB
25/05/08 06:34:50 WARN DAGScheduler: Broadcasting large task binary with size
1089.9 KiB
25/05/08 06:34:53 WARN DAGScheduler: Broadcasting large task binary with size
1096.9 KiB
25/05/08 06:34:54 WARN DAGScheduler: Broadcasting large task binary with size
1097.4 KiB
25/05/08 06:34:55 WARN DAGScheduler: Broadcasting large task binary with size
1098.0 KiB
25/05/08 06:34:55 WARN DAGScheduler: Broadcasting large task binary with size
1098.7 KiB
25/05/08 06:34:56 WARN DAGScheduler: Broadcasting large task binary with size
1099.3 KiB
25/05/08 06:34:57 WARN DAGScheduler: Broadcasting large task binary with size
1100.9 KiB
25/05/08 06:34:58 WARN DAGScheduler: Broadcasting large task binary with size
1101.4 KiB
25/05/08 06:34:58 WARN DAGScheduler: Broadcasting large task binary with size
1101.8 KiB
25/05/08 06:34:59 WARN DAGScheduler: Broadcasting large task binary with size
1102.4 KiB
25/05/08 06:34:59 WARN DAGScheduler: Broadcasting large task binary with size
1103.7 KiB
```

```
25/05/08 06:35:00 WARN DAGScheduler: Broadcasting large task binary with size
1105.4 KiB
25/05/08 06:35:01 WARN DAGScheduler: Broadcasting large task binary with size
1105.9 KiB
25/05/08 06:35:02 WARN DAGScheduler: Broadcasting large task binary with size
1106.5 KiB
25/05/08 06:35:02 WARN DAGScheduler: Broadcasting large task binary with size
1107.7 KiB
25/05/08 06:35:03 WARN DAGScheduler: Broadcasting large task binary with size
1108.3 KiB
25/05/08 06:35:04 WARN DAGScheduler: Broadcasting large task binary with size
1109.4 KiB
25/05/08 06:35:05 WARN DAGScheduler: Broadcasting large task binary with size
1109.9 KiB
25/05/08 06:35:05 WARN DAGScheduler: Broadcasting large task binary with size
1110.5 KiB
25/05/08 06:35:06 WARN DAGScheduler: Broadcasting large task binary with size
1111.4 KiB
25/05/08 06:35:07 WARN DAGScheduler: Broadcasting large task binary with size
1112.3 KiB
25/05/08 06:35:07 WARN DAGScheduler: Broadcasting large task binary with size
1113.2 KiB
25/05/08 06:35:08 WARN DAGScheduler: Broadcasting large task binary with size
1113.7 KiB
25/05/08 06:35:09 WARN DAGScheduler: Broadcasting large task binary with size
1114.4 KiB
25/05/08 06:35:09 WARN DAGScheduler: Broadcasting large task binary with size
1115.3 KiB
25/05/08 06:35:10 WARN DAGScheduler: Broadcasting large task binary with size
1115.9 KiB
25/05/08 06:35:11 WARN DAGScheduler: Broadcasting large task binary with size
1117.0 KiB
25/05/08 06:35:11 WARN DAGScheduler: Broadcasting large task binary with size
1117.5 KiB
25/05/08 06:35:12 WARN DAGScheduler: Broadcasting large task binary with size
1118.1 KiB
25/05/08 06:35:13 WARN DAGScheduler: Broadcasting large task binary with size
1119.7 KiB
25/05/08 06:35:14 WARN DAGScheduler: Broadcasting large task binary with size
1120.2 KiB
25/05/08 06:35:15 WARN DAGScheduler: Broadcasting large task binary with size
1120.8 KiB
25/05/08 06:35:15 WARN DAGScheduler: Broadcasting large task binary with size
1121.5 KiB
25/05/08 06:35:16 WARN DAGScheduler: Broadcasting large task binary with size
1122.1 KiB
25/05/08 06:35:17 WARN DAGScheduler: Broadcasting large task binary with size
1123.3 KiB
```

```
25/05/08 06:35:17 WARN DAGScheduler: Broadcasting large task binary with size
1123.9 KiB
25/05/08 06:35:18 WARN DAGScheduler: Broadcasting large task binary with size
1124.5 KiB
25/05/08 06:35:19 WARN DAGScheduler: Broadcasting large task binary with size
1125.5 KiB
25/05/08 06:35:20 WARN DAGScheduler: Broadcasting large task binary with size
1126.4 KiB
25/05/08 06:35:20 WARN DAGScheduler: Broadcasting large task binary with size
1127.4 KiB
25/05/08 06:35:21 WARN DAGScheduler: Broadcasting large task binary with size
1127.9 KiB
25/05/08 06:35:22 WARN DAGScheduler: Broadcasting large task binary with size
1128.5 KiB
25/05/08 06:35:23 WARN DAGScheduler: Broadcasting large task binary with size
1129.7 KiB
25/05/08 06:35:23 WARN DAGScheduler: Broadcasting large task binary with size
1130.3 KiB
25/05/08 06:35:24 WARN DAGScheduler: Broadcasting large task binary with size
1131.2 KiB
25/05/08 06:35:25 WARN DAGScheduler: Broadcasting large task binary with size
1131.7 KiB
25/05/08 06:35:25 WARN DAGScheduler: Broadcasting large task binary with size
1132.3 KiB
25/05/08 06:35:26 WARN DAGScheduler: Broadcasting large task binary with size
1133.0 KiB
25/05/08 06:35:27 WARN DAGScheduler: Broadcasting large task binary with size
1133.6 KiB
25/05/08 06:35:27 WARN DAGScheduler: Broadcasting large task binary with size
1134.9 KiB
25/05/08 06:35:28 WARN DAGScheduler: Broadcasting large task binary with size
1135.4 KiB
25/05/08 06:35:29 WARN DAGScheduler: Broadcasting large task binary with size
1136.0 KiB
25/05/08 06:35:30 WARN DAGScheduler: Broadcasting large task binary with size
1136.6 KiB
25/05/08 06:35:30 WARN DAGScheduler: Broadcasting large task binary with size
1137.8 KiB
25/05/08 06:35:31 WARN DAGScheduler: Broadcasting large task binary with size
1139.5 KiB
25/05/08 06:35:32 WARN DAGScheduler: Broadcasting large task binary with size
1140.1 KiB
25/05/08 06:35:33 WARN DAGScheduler: Broadcasting large task binary with size
1140.7 KiB
25/05/08 06:35:33 WARN DAGScheduler: Broadcasting large task binary with size
1142.2 KiB
25/05/08 06:35:34 WARN DAGScheduler: Broadcasting large task binary with size
1142.7 KiB
```

```
25/05/08 06:35:35 WARN DAGScheduler: Broadcasting large task binary with size
1143.1 KiB
25/05/08 06:35:36 WARN DAGScheduler: Broadcasting large task binary with size
1144.7 KiB
25/05/08 06:35:36 WARN DAGScheduler: Broadcasting large task binary with size
1145.2 KiB
25/05/08 06:35:37 WARN DAGScheduler: Broadcasting large task binary with size
1145.8 KiB
25/05/08 06:35:38 WARN DAGScheduler: Broadcasting large task binary with size
1146.8 KiB
25/05/08 06:35:38 WARN DAGScheduler: Broadcasting large task binary with size
1148.1 KiB
25/05/08 06:35:39 WARN DAGScheduler: Broadcasting large task binary with size
1149.1 KiB
25/05/08 06:35:40 WARN DAGScheduler: Broadcasting large task binary with size
1149.6 KiB
25/05/08 06:35:40 WARN DAGScheduler: Broadcasting large task binary with size
1150.2 KiB
25/05/08 06:35:41 WARN DAGScheduler: Broadcasting large task binary with size
1151.1 KiB
25/05/08 06:35:42 WARN DAGScheduler: Broadcasting large task binary with size
1151.7 KiB
25/05/08 06:35:42 WARN DAGScheduler: Broadcasting large task binary with size
1152.9 KiB
25/05/08 06:35:43 WARN DAGScheduler: Broadcasting large task binary with size
1153.4 KiB
25/05/08 06:35:44 WARN DAGScheduler: Broadcasting large task binary with size
1154.0 KiB
25/05/08 06:35:44 WARN DAGScheduler: Broadcasting large task binary with size
1154.4 KiB
25/05/08 06:35:45 WARN DAGScheduler: Broadcasting large task binary with size
1154.7 KiB
25/05/08 06:35:45 WARN DAGScheduler: Broadcasting large task binary with size
1155.9 KiB
25/05/08 06:35:46 WARN DAGScheduler: Broadcasting large task binary with size
1156.5 KiB
25/05/08 06:35:47 WARN DAGScheduler: Broadcasting large task binary with size
1157.1 KiB
25/05/08 06:35:47 WARN DAGScheduler: Broadcasting large task binary with size
1158.3 KiB
25/05/08 06:35:48 WARN DAGScheduler: Broadcasting large task binary with size
1159.8 KiB
25/05/08 06:35:48 WARN DAGScheduler: Broadcasting large task binary with size
1160.8 KiB
25/05/08 06:35:49 WARN DAGScheduler: Broadcasting large task binary with size
1161.3 KiB
25/05/08 06:35:50 WARN DAGScheduler: Broadcasting large task binary with size
1161.9 KiB
```

```
25/05/08 06:35:51 WARN DAGScheduler: Broadcasting large task binary with size
1162.5 KiB
25/05/08 06:35:51 WARN DAGScheduler: Broadcasting large task binary with size
1162.9 KiB
25/05/08 06:35:51 WARN DAGScheduler: Broadcasting large task binary with size
1164.1 KiB
25/05/08 06:35:52 WARN DAGScheduler: Broadcasting large task binary with size
1164.6 KiB
25/05/08 06:35:53 WARN DAGScheduler: Broadcasting large task binary with size
1165.2 KiB
25/05/08 06:35:54 WARN DAGScheduler: Broadcasting large task binary with size
1166.1 KiB
25/05/08 06:35:54 WARN DAGScheduler: Broadcasting large task binary with size
1166.7 KiB
25/05/08 06:35:54 WARN DAGScheduler: Broadcasting large task binary with size
1168.1 KiB
25/05/08 06:35:55 WARN DAGScheduler: Broadcasting large task binary with size
1168.6 KiB
25/05/08 06:35:56 WARN DAGScheduler: Broadcasting large task binary with size
1169.1 KiB
25/05/08 06:35:57 WARN DAGScheduler: Broadcasting large task binary with size
1170.1 KiB
25/05/08 06:35:57 WARN DAGScheduler: Broadcasting large task binary with size
1171.4 KiB
25/05/08 06:35:58 WARN DAGScheduler: Broadcasting large task binary with size
1172.7 KiB
25/05/08 06:35:58 WARN DAGScheduler: Broadcasting large task binary with size
1173.2 KiB
25/05/08 06:35:59 WARN DAGScheduler: Broadcasting large task binary with size
1173.8 KiB
25/05/08 06:36:00 WARN DAGScheduler: Broadcasting large task binary with size
1175.4 KiB
25/05/08 06:36:01 WARN DAGScheduler: Broadcasting large task binary with size
1175.9 KiB
25/05/08 06:36:01 WARN DAGScheduler: Broadcasting large task binary with size
1176.5 KiB
25/05/08 06:36:02 WARN DAGScheduler: Broadcasting large task binary with size
1177.1 KiB
25/05/08 06:36:03 WARN DAGScheduler: Broadcasting large task binary with size
1177.8 KiB
25/05/08 06:36:03 WARN DAGScheduler: Broadcasting large task binary with size
1179.0 KiB
25/05/08 06:36:04 WARN DAGScheduler: Broadcasting large task binary with size
1179.5 KiB
25/05/08 06:36:05 WARN DAGScheduler: Broadcasting large task binary with size
1179.8 KiB
25/05/08 06:36:05 WARN DAGScheduler: Broadcasting large task binary with size
1180.5 KiB
```

```
25/05/08 06:36:06 WARN DAGScheduler: Broadcasting large task binary with size
1180.9 KiB
25/05/08 06:36:06 WARN DAGScheduler: Broadcasting large task binary with size
1182.1 KiB
25/05/08 06:36:07 WARN DAGScheduler: Broadcasting large task binary with size
1182.6 KiB
25/05/08 06:36:08 WARN DAGScheduler: Broadcasting large task binary with size
1183.2 KiB
25/05/08 06:36:09 WARN DAGScheduler: Broadcasting large task binary with size
1183.5 KiB
25/05/08 06:36:09 WARN DAGScheduler: Broadcasting large task binary with size
1183.8 KiB
25/05/08 06:36:09 WARN DAGScheduler: Broadcasting large task binary with size
1185.4 KiB
25/05/08 06:36:10 WARN DAGScheduler: Broadcasting large task binary with size
1185.9 KiB
25/05/08 06:36:11 WARN DAGScheduler: Broadcasting large task binary with size
1186.5 KiB
25/05/08 06:36:12 WARN DAGScheduler: Broadcasting large task binary with size
1187.4 KiB
25/05/08 06:36:13 WARN DAGScheduler: Broadcasting large task binary with size
1188.1 KiB
25/05/08 06:36:13 WARN DAGScheduler: Broadcasting large task binary with size
1189.0 KiB
25/05/08 06:36:14 WARN DAGScheduler: Broadcasting large task binary with size
1189.6 KiB
25/05/08 06:36:15 WARN DAGScheduler: Broadcasting large task binary with size
1190.2 KiB
25/05/08 06:36:15 WARN DAGScheduler: Broadcasting large task binary with size
1190.8 KiB
25/05/08 06:36:16 WARN DAGScheduler: Broadcasting large task binary with size
1191.4 KiB
25/05/08 06:36:16 WARN DAGScheduler: Broadcasting large task binary with size
1192.7 KiB
25/05/08 06:36:17 WARN DAGScheduler: Broadcasting large task binary with size
1193.2 KiB
25/05/08 06:36:18 WARN DAGScheduler: Broadcasting large task binary with size
1193.5 KiB
25/05/08 06:36:18 WARN DAGScheduler: Broadcasting large task binary with size
1195.2 KiB
25/05/08 06:36:19 WARN DAGScheduler: Broadcasting large task binary with size
1195.7 KiB
25/05/08 06:36:20 WARN DAGScheduler: Broadcasting large task binary with size
1196.3 KiB
25/05/08 06:36:20 WARN DAGScheduler: Broadcasting large task binary with size
1197.3 KiB
25/05/08 06:36:21 WARN DAGScheduler: Broadcasting large task binary with size
1197.6 KiB
```

```
25/05/08 06:36:21 WARN DAGScheduler: Broadcasting large task binary with size
1198.9 KiB
25/05/08 06:36:22 WARN DAGScheduler: Broadcasting large task binary with size
1199.4 KiB
25/05/08 06:36:23 WARN DAGScheduler: Broadcasting large task binary with size
1200.0 KiB
25/05/08 06:36:23 WARN DAGScheduler: Broadcasting large task binary with size
1200.3 KiB
25/05/08 06:36:24 WARN DAGScheduler: Broadcasting large task binary with size
1201.7 KiB
25/05/08 06:36:25 WARN DAGScheduler: Broadcasting large task binary with size
1202.3 KiB
25/05/08 06:36:25 WARN DAGScheduler: Broadcasting large task binary with size
1202.8 KiB
25/05/08 06:36:26 WARN DAGScheduler: Broadcasting large task binary with size
1203.1 KiB
25/05/08 06:36:26 WARN DAGScheduler: Broadcasting large task binary with size
1204.5 KiB
25/05/08 06:36:27 WARN DAGScheduler: Broadcasting large task binary with size
1205.0 KiB
25/05/08 06:36:28 WARN DAGScheduler: Broadcasting large task binary with size
1205.3 KiB
25/05/08 06:36:29 WARN DAGScheduler: Broadcasting large task binary with size
1205.6 KiB
25/05/08 06:36:30 WARN DAGScheduler: Broadcasting large task binary with size
1205.9 KiB
25/05/08 06:36:30 WARN DAGScheduler: Broadcasting large task binary with size
1207.6 KiB
25/05/08 06:36:31 WARN DAGScheduler: Broadcasting large task binary with size
1208.1 KiB
25/05/08 06:36:32 WARN DAGScheduler: Broadcasting large task binary with size
1208.7 KiB
25/05/08 06:36:33 WARN DAGScheduler: Broadcasting large task binary with size
1209.0 KiB
25/05/08 06:36:33 WARN DAGScheduler: Broadcasting large task binary with size
1210.4 KiB
25/05/08 06:36:34 WARN DAGScheduler: Broadcasting large task binary with size
1211.0 KiB
25/05/08 06:36:35 WARN DAGScheduler: Broadcasting large task binary with size
1211.5 KiB
25/05/08 06:36:36 WARN DAGScheduler: Broadcasting large task binary with size
1212.1 KiB
25/05/08 06:36:36 WARN DAGScheduler: Broadcasting large task binary with size
1212.8 KiB
25/05/08 06:36:37 WARN DAGScheduler: Broadcasting large task binary with size
1214.1 KiB
25/05/08 06:36:38 WARN DAGScheduler: Broadcasting large task binary with size
1214.6 KiB
```

```
25/05/08 06:36:38 WARN DAGScheduler: Broadcasting large task binary with size
1215.2 KiB
25/05/08 06:36:39 WARN DAGScheduler: Broadcasting large task binary with size
1215.8 KiB
25/05/08 06:36:40 WARN DAGScheduler: Broadcasting large task binary with size
1216.8 KiB
25/05/08 06:36:40 WARN DAGScheduler: Broadcasting large task binary with size
1218.5 KiB
25/05/08 06:36:41 WARN DAGScheduler: Broadcasting large task binary with size
1219.0 KiB
25/05/08 06:36:42 WARN DAGScheduler: Broadcasting large task binary with size
1219.6 KiB
25/05/08 06:36:43 WARN DAGScheduler: Broadcasting large task binary with size
1220.6 KiB
25/05/08 06:36:43 WARN DAGScheduler: Broadcasting large task binary with size
1221.2 KiB
25/05/08 06:36:44 WARN DAGScheduler: Broadcasting large task binary with size
1222.3 KiB
25/05/08 06:36:44 WARN DAGScheduler: Broadcasting large task binary with size
1222.9 KiB
25/05/08 06:36:45 WARN DAGScheduler: Broadcasting large task binary with size
1223.5 KiB
25/05/08 06:36:46 WARN DAGScheduler: Broadcasting large task binary with size
1224.4 KiB
25/05/08 06:36:47 WARN DAGScheduler: Broadcasting large task binary with size
1225.0 KiB
25/05/08 06:36:47 WARN DAGScheduler: Broadcasting large task binary with size
1226.4 KiB
25/05/08 06:36:48 WARN DAGScheduler: Broadcasting large task binary with size
1226.9 KiB
25/05/08 06:36:49 WARN DAGScheduler: Broadcasting large task binary with size
1227.5 KiB
25/05/08 06:36:49 WARN DAGScheduler: Broadcasting large task binary with size
1229.1 KiB
25/05/08 06:36:50 WARN DAGScheduler: Broadcasting large task binary with size
1229.6 KiB
25/05/08 06:36:51 WARN DAGScheduler: Broadcasting large task binary with size
1230.3 KiB
25/05/08 06:36:51 WARN DAGScheduler: Broadcasting large task binary with size
1230.9 KiB
25/05/08 06:36:52 WARN DAGScheduler: Broadcasting large task binary with size
1231.6 KiB
25/05/08 06:36:53 WARN DAGScheduler: Broadcasting large task binary with size
1232.8 KiB
25/05/08 06:36:53 WARN DAGScheduler: Broadcasting large task binary with size
1233.3 KiB
25/05/08 06:36:54 WARN DAGScheduler: Broadcasting large task binary with size
1235.1 KiB
```

```
25/05/08 06:36:55 WARN DAGScheduler: Broadcasting large task binary with size
1235.6 KiB
25/05/08 06:36:55 WARN DAGScheduler: Broadcasting large task binary with size
1236.2 KiB
25/05/08 06:36:56 WARN DAGScheduler: Broadcasting large task binary with size
1236.5 KiB
25/05/08 06:36:56 WARN DAGScheduler: Broadcasting large task binary with size
1236.9 KiB
25/05/08 06:36:57 WARN DAGScheduler: Broadcasting large task binary with size
1238.3 KiB
25/05/08 06:36:57 WARN DAGScheduler: Broadcasting large task binary with size
1238.8 KiB
25/05/08 06:36:58 WARN DAGScheduler: Broadcasting large task binary with size
1240.8 KiB
25/05/08 06:36:59 WARN DAGScheduler: Broadcasting large task binary with size
1241.3 KiB
25/05/08 06:37:00 WARN DAGScheduler: Broadcasting large task binary with size
1243.4 KiB
25/05/08 06:37:00 WARN DAGScheduler: Broadcasting large task binary with size
1243.9 KiB
25/05/08 06:37:01 WARN DAGScheduler: Broadcasting large task binary with size
1244.3 KiB
25/05/08 06:37:02 WARN DAGScheduler: Broadcasting large task binary with size
1245.9 KiB
25/05/08 06:37:02 WARN DAGScheduler: Broadcasting large task binary with size
1246.4 KiB
25/05/08 06:37:03 WARN DAGScheduler: Broadcasting large task binary with size
1246.8 KiB
25/05/08 06:37:04 WARN DAGScheduler: Broadcasting large task binary with size
1247.4 KiB
25/05/08 06:37:04 WARN DAGScheduler: Broadcasting large task binary with size
1247.8 KiB
25/05/08 06:37:05 WARN DAGScheduler: Broadcasting large task binary with size
1249.2 KiB
25/05/08 06:37:05 WARN DAGScheduler: Broadcasting large task binary with size
1249.7 KiB
25/05/08 06:37:06 WARN DAGScheduler: Broadcasting large task binary with size
1251.8 KiB
25/05/08 06:37:07 WARN DAGScheduler: Broadcasting large task binary with size
1252.3 KiB
25/05/08 06:37:08 WARN DAGScheduler: Broadcasting large task binary with size
1253.0 KiB
25/05/08 06:37:08 WARN DAGScheduler: Broadcasting large task binary with size
1253.9 KiB
25/05/08 06:37:09 WARN DAGScheduler: Broadcasting large task binary with size
1254.3 KiB
25/05/08 06:37:10 WARN DAGScheduler: Broadcasting large task binary with size
1255.3 KiB
```

```
25/05/08 06:37:10 WARN DAGScheduler: Broadcasting large task binary with size
1255.8 KiB
25/05/08 06:37:11 WARN DAGScheduler: Broadcasting large task binary with size
1256.4 KiB
25/05/08 06:37:12 WARN DAGScheduler: Broadcasting large task binary with size
1256.8 KiB
25/05/08 06:37:12 WARN DAGScheduler: Broadcasting large task binary with size
1257.4 KiB
25/05/08 06:37:12 WARN DAGScheduler: Broadcasting large task binary with size
1259.0 KiB
25/05/08 06:37:13 WARN DAGScheduler: Broadcasting large task binary with size
1259.5 KiB
25/05/08 06:37:14 WARN DAGScheduler: Broadcasting large task binary with size
1260.0 KiB
25/05/08 06:37:15 WARN DAGScheduler: Broadcasting large task binary with size
1260.3 KiB
25/05/08 06:37:15 WARN DAGScheduler: Broadcasting large task binary with size
1260.7 KiB
25/05/08 06:37:15 WARN DAGScheduler: Broadcasting large task binary with size
1262.2 KiB
25/05/08 06:37:16 WARN DAGScheduler: Broadcasting large task binary with size
1262.7 KiB
25/05/08 06:37:17 WARN DAGScheduler: Broadcasting large task binary with size
1263.3 KiB
25/05/08 06:37:18 WARN DAGScheduler: Broadcasting large task binary with size
1264.0 KiB
25/05/08 06:37:18 WARN DAGScheduler: Broadcasting large task binary with size
1264.3 KiB
25/05/08 06:37:19 WARN DAGScheduler: Broadcasting large task binary with size
1265.5 KiB
25/05/08 06:37:19 WARN DAGScheduler: Broadcasting large task binary with size
1266.0 KiB
25/05/08 06:37:20 WARN DAGScheduler: Broadcasting large task binary with size
1266.6 KiB
25/05/08 06:37:21 WARN DAGScheduler: Broadcasting large task binary with size
1268.1 KiB
25/05/08 06:37:21 WARN DAGScheduler: Broadcasting large task binary with size
1268.7 KiB
25/05/08 06:37:22 WARN DAGScheduler: Broadcasting large task binary with size
1269.2 KiB
25/05/08 06:37:23 WARN DAGScheduler: Broadcasting large task binary with size
1269.8 KiB
25/05/08 06:37:23 WARN DAGScheduler: Broadcasting large task binary with size
1270.8 KiB
25/05/08 06:37:24 WARN DAGScheduler: Broadcasting large task binary with size
1272.0 KiB
25/05/08 06:37:25 WARN DAGScheduler: Broadcasting large task binary with size
1272.5 KiB
```

```
25/05/08 06:37:25 WARN DAGScheduler: Broadcasting large task binary with size
1273.0 KiB
25/05/08 06:37:26 WARN DAGScheduler: Broadcasting large task binary with size
1273.3 KiB
25/05/08 06:37:27 WARN DAGScheduler: Broadcasting large task binary with size
1273.7 KiB
25/05/08 06:37:27 WARN DAGScheduler: Broadcasting large task binary with size
1275.2 KiB
25/05/08 06:37:28 WARN DAGScheduler: Broadcasting large task binary with size
1275.7 KiB
25/05/08 06:37:29 WARN DAGScheduler: Broadcasting large task binary with size
1276.3 KiB
25/05/08 06:37:29 WARN DAGScheduler: Broadcasting large task binary with size
1277.9 KiB
25/05/08 06:37:30 WARN DAGScheduler: Broadcasting large task binary with size
1278.4 KiB
25/05/08 06:37:31 WARN DAGScheduler: Broadcasting large task binary with size
1279.0 KiB
25/05/08 06:37:32 WARN DAGScheduler: Broadcasting large task binary with size
1279.7 KiB
25/05/08 06:37:32 WARN DAGScheduler: Broadcasting large task binary with size
1281.0 KiB
25/05/08 06:37:33 WARN DAGScheduler: Broadcasting large task binary with size
1281.5 KiB
25/05/08 06:37:34 WARN DAGScheduler: Broadcasting large task binary with size
1282.1 KiB
25/05/08 06:37:35 WARN DAGScheduler: Broadcasting large task binary with size
1282.7 KiB
25/05/08 06:37:36 WARN DAGScheduler: Broadcasting large task binary with size
1283.1 KiB
25/05/08 06:37:36 WARN DAGScheduler: Broadcasting large task binary with size
1284.5 KiB
25/05/08 06:37:37 WARN DAGScheduler: Broadcasting large task binary with size
1285.0 KiB
25/05/08 06:37:38 WARN DAGScheduler: Broadcasting large task binary with size
1285.5 KiB
25/05/08 06:37:38 WARN DAGScheduler: Broadcasting large task binary with size
1285.9 KiB
25/05/08 06:37:39 WARN DAGScheduler: Broadcasting large task binary with size
1286.2 KiB
25/05/08 06:37:39 WARN DAGScheduler: Broadcasting large task binary with size
1287.7 KiB
25/05/08 06:37:40 WARN DAGScheduler: Broadcasting large task binary with size
1288.2 KiB
25/05/08 06:37:41 WARN DAGScheduler: Broadcasting large task binary with size
1288.8 KiB
25/05/08 06:37:42 WARN DAGScheduler: Broadcasting large task binary with size
1289.1 KiB
```

```
25/05/08 06:37:42 WARN DAGScheduler: Broadcasting large task binary with size
1289.4 KiB
25/05/08 06:37:43 WARN DAGScheduler: Broadcasting large task binary with size
1291.0 KiB
25/05/08 06:37:44 WARN DAGScheduler: Broadcasting large task binary with size
1291.5 KiB
25/05/08 06:37:44 WARN DAGScheduler: Broadcasting large task binary with size
1292.0 KiB
25/05/08 06:37:45 WARN DAGScheduler: Broadcasting large task binary with size
1293.6 KiB
25/05/08 06:37:46 WARN DAGScheduler: Broadcasting large task binary with size
1294.1 KiB
25/05/08 06:37:47 WARN DAGScheduler: Broadcasting large task binary with size
1294.4 KiB
25/05/08 06:37:47 WARN DAGScheduler: Broadcasting large task binary with size
1294.8 KiB
25/05/08 06:37:48 WARN DAGScheduler: Broadcasting large task binary with size
1295.1 KiB
25/05/08 06:37:49 WARN DAGScheduler: Broadcasting large task binary with size
1296.7 KiB
25/05/08 06:37:49 WARN DAGScheduler: Broadcasting large task binary with size
1297.2 KiB
25/05/08 06:37:50 WARN DAGScheduler: Broadcasting large task binary with size
1297.8 KiB
25/05/08 06:37:51 WARN DAGScheduler: Broadcasting large task binary with size
1298.1 KiB
25/05/08 06:37:51 WARN DAGScheduler: Broadcasting large task binary with size
1298.4 KiB
25/05/08 06:37:52 WARN DAGScheduler: Broadcasting large task binary with size
1300.1 KiB
25/05/08 06:37:52 WARN DAGScheduler: Broadcasting large task binary with size
1300.6 KiB
25/05/08 06:37:53 WARN DAGScheduler: Broadcasting large task binary with size
1301.2 KiB
25/05/08 06:37:54 WARN DAGScheduler: Broadcasting large task binary with size
1301.6 KiB
25/05/08 06:37:54 WARN DAGScheduler: Broadcasting large task binary with size
1301.9 KiB
25/05/08 06:37:54 WARN DAGScheduler: Broadcasting large task binary with size
1303.4 KiB
25/05/08 06:37:55 WARN DAGScheduler: Broadcasting large task binary with size
1303.9 KiB
25/05/08 06:37:56 WARN DAGScheduler: Broadcasting large task binary with size
1304.5 KiB
25/05/08 06:37:56 WARN DAGScheduler: Broadcasting large task binary with size
1304.8 KiB
25/05/08 06:37:56 WARN DAGScheduler: Broadcasting large task binary with size
1305.1 KiB
```

```
25/05/08 06:37:57 WARN DAGScheduler: Broadcasting large task binary with size
1306.5 KiB
25/05/08 06:37:57 WARN DAGScheduler: Broadcasting large task binary with size
1307.0 KiB
25/05/08 06:37:58 WARN DAGScheduler: Broadcasting large task binary with size
1307.5 KiB
25/05/08 06:37:58 WARN DAGScheduler: Broadcasting large task binary with size
1307.8 KiB
25/05/08 06:37:59 WARN DAGScheduler: Broadcasting large task binary with size
1309.3 KiB
25/05/08 06:38:00 WARN DAGScheduler: Broadcasting large task binary with size
1309.8 KiB
25/05/08 06:38:00 WARN DAGScheduler: Broadcasting large task binary with size
1310.1 KiB
25/05/08 06:38:00 WARN DAGScheduler: Broadcasting large task binary with size
1310.7 KiB
25/05/08 06:38:01 WARN DAGScheduler: Broadcasting large task binary with size
1312.2 KiB
25/05/08 06:38:01 WARN DAGScheduler: Broadcasting large task binary with size
1312.7 KiB
25/05/08 06:38:02 WARN DAGScheduler: Broadcasting large task binary with size
1313.1 KiB
25/05/08 06:38:03 WARN DAGScheduler: Broadcasting large task binary with size
1314.7 KiB
25/05/08 06:38:03 WARN DAGScheduler: Broadcasting large task binary with size
1315.2 KiB
25/05/08 06:38:04 WARN DAGScheduler: Broadcasting large task binary with size
1315.8 KiB
25/05/08 06:38:05 WARN DAGScheduler: Broadcasting large task binary with size
1316.4 KiB
25/05/08 06:38:05 WARN DAGScheduler: Broadcasting large task binary with size
1317.1 KiB
25/05/08 06:38:06 WARN DAGScheduler: Broadcasting large task binary with size
1318.7 KiB
25/05/08 06:38:07 WARN DAGScheduler: Broadcasting large task binary with size
1319.2 KiB
25/05/08 06:38:08 WARN DAGScheduler: Broadcasting large task binary with size
1319.5 KiB
25/05/08 06:38:08 WARN DAGScheduler: Broadcasting large task binary with size
1319.8 KiB
25/05/08 06:38:09 WARN DAGScheduler: Broadcasting large task binary with size
1320.4 KiB
25/05/08 06:38:09 WARN DAGScheduler: Broadcasting large task binary with size
1321.9 KiB
25/05/08 06:38:10 WARN DAGScheduler: Broadcasting large task binary with size
1322.4 KiB
25/05/08 06:38:10 WARN DAGScheduler: Broadcasting large task binary with size
1322.7 KiB
```

```
25/05/08 06:38:11 WARN DAGScheduler: Broadcasting large task binary with size
1324.4 KiB
25/05/08 06:38:12 WARN DAGScheduler: Broadcasting large task binary with size
1324.9 KiB
25/05/08 06:38:12 WARN DAGScheduler: Broadcasting large task binary with size
1326.8 KiB
25/05/08 06:38:13 WARN DAGScheduler: Broadcasting large task binary with size
1327.4 KiB
25/05/08 06:38:14 WARN DAGScheduler: Broadcasting large task binary with size
1328.0 KiB
25/05/08 06:38:14 WARN DAGScheduler: Broadcasting large task binary with size
1328.6 KiB
25/05/08 06:38:15 WARN DAGScheduler: Broadcasting large task binary with size
1329.2 KiB
25/05/08 06:38:16 WARN DAGScheduler: Broadcasting large task binary with size
1330.7 KiB
25/05/08 06:38:17 WARN DAGScheduler: Broadcasting large task binary with size
1331.2 KiB
25/05/08 06:38:17 WARN DAGScheduler: Broadcasting large task binary with size
1331.9 KiB
25/05/08 06:38:18 WARN DAGScheduler: Broadcasting large task binary with size
1332.8 KiB
25/05/08 06:38:19 WARN DAGScheduler: Broadcasting large task binary with size
1334.0 KiB
25/05/08 06:38:20 WARN DAGScheduler: Broadcasting large task binary with size
1334.5 KiB
25/05/08 06:38:21 WARN DAGScheduler: Broadcasting large task binary with size
1335.1 KiB
25/05/08 06:38:21 WARN DAGScheduler: Broadcasting large task binary with size
1335.7 KiB
25/05/08 06:38:22 WARN DAGScheduler: Broadcasting large task binary with size
1336.1 KiB
25/05/08 06:38:22 WARN DAGScheduler: Broadcasting large task binary with size
1337.4 KiB
25/05/08 06:38:23 WARN DAGScheduler: Broadcasting large task binary with size
1338.0 KiB
25/05/08 06:38:24 WARN DAGScheduler: Broadcasting large task binary with size
1338.6 KiB
25/05/08 06:38:24 WARN DAGScheduler: Broadcasting large task binary with size
1339.6 KiB
25/05/08 06:38:25 WARN DAGScheduler: Broadcasting large task binary with size
1340.2 KiB
25/05/08 06:38:25 WARN DAGScheduler: Broadcasting large task binary with size
1341.1 KiB
25/05/08 06:38:26 WARN DAGScheduler: Broadcasting large task binary with size
1341.6 KiB
25/05/08 06:38:27 WARN DAGScheduler: Broadcasting large task binary with size
1342.2 KiB
```

```
25/05/08 06:38:27 WARN DAGScheduler: Broadcasting large task binary with size
1343.7 KiB
25/05/08 06:38:28 WARN DAGScheduler: Broadcasting large task binary with size
1344.2 KiB
25/05/08 06:38:29 WARN DAGScheduler: Broadcasting large task binary with size
1346.2 KiB
25/05/08 06:38:30 WARN DAGScheduler: Broadcasting large task binary with size
1346.7 KiB
25/05/08 06:38:30 WARN DAGScheduler: Broadcasting large task binary with size
1347.3 KiB
25/05/08 06:38:31 WARN DAGScheduler: Broadcasting large task binary with size
1347.7 KiB
25/05/08 06:38:31 WARN DAGScheduler: Broadcasting large task binary with size
1349.1 KiB
25/05/08 06:38:32 WARN DAGScheduler: Broadcasting large task binary with size
1349.6 KiB
25/05/08 06:38:33 WARN DAGScheduler: Broadcasting large task binary with size
1351.4 KiB
25/05/08 06:38:34 WARN DAGScheduler: Broadcasting large task binary with size
1351.9 KiB
25/05/08 06:38:34 WARN DAGScheduler: Broadcasting large task binary with size
1352.5 KiB
25/05/08 06:38:35 WARN DAGScheduler: Broadcasting large task binary with size
1354.1 KiB
25/05/08 06:38:36 WARN DAGScheduler: Broadcasting large task binary with size
1354.6 KiB
25/05/08 06:38:37 WARN DAGScheduler: Broadcasting large task binary with size
1355.1 KiB
25/05/08 06:38:37 WARN DAGScheduler: Broadcasting large task binary with size
1355.4 KiB
25/05/08 06:38:37 WARN DAGScheduler: Broadcasting large task binary with size
1356.0 KiB
25/05/08 06:38:38 WARN DAGScheduler: Broadcasting large task binary with size
1357.5 KiB
25/05/08 06:38:38 WARN DAGScheduler: Broadcasting large task binary with size
1358.0 KiB
25/05/08 06:38:39 WARN DAGScheduler: Broadcasting large task binary with size
1358.6 KiB
25/05/08 06:38:39 WARN DAGScheduler: Broadcasting large task binary with size
1360.2 KiB
25/05/08 06:38:40 WARN DAGScheduler: Broadcasting large task binary with size
1360.7 KiB
25/05/08 06:38:41 WARN DAGScheduler: Broadcasting large task binary with size
1361.3 KiB
25/05/08 06:38:42 WARN DAGScheduler: Broadcasting large task binary with size
1362.9 KiB
25/05/08 06:38:42 WARN DAGScheduler: Broadcasting large task binary with size
1363.4 KiB
```

```
25/05/08 06:38:43 WARN DAGScheduler: Broadcasting large task binary with size
1363.7 KiB
25/05/08 06:38:43 WARN DAGScheduler: Broadcasting large task binary with size
1364.3 KiB
25/05/08 06:38:44 WARN DAGScheduler: Broadcasting large task binary with size
1365.0 KiB
25/05/08 06:38:45 WARN DAGScheduler: Broadcasting large task binary with size
1366.5 KiB
25/05/08 06:38:45 WARN DAGScheduler: Broadcasting large task binary with size
1367.0 KiB
25/05/08 06:38:46 WARN DAGScheduler: Broadcasting large task binary with size
1367.7 KiB
25/05/08 06:38:47 WARN DAGScheduler: Broadcasting large task binary with size
1368.0 KiB
25/05/08 06:38:47 WARN DAGScheduler: Broadcasting large task binary with size
1368.4 KiB
25/05/08 06:38:48 WARN DAGScheduler: Broadcasting large task binary with size
1369.8 KiB
25/05/08 06:38:49 WARN DAGScheduler: Broadcasting large task binary with size
1370.3 KiB
25/05/08 06:38:50 WARN DAGScheduler: Broadcasting large task binary with size
1370.9 KiB
25/05/08 06:38:50 WARN DAGScheduler: Broadcasting large task binary with size
1371.2 KiB
25/05/08 06:38:51 WARN DAGScheduler: Broadcasting large task binary with size
1372.7 KiB
25/05/08 06:38:51 WARN DAGScheduler: Broadcasting large task binary with size
1373.2 KiB
25/05/08 06:38:52 WARN DAGScheduler: Broadcasting large task binary with size
1373.8 KiB
25/05/08 06:38:53 WARN DAGScheduler: Broadcasting large task binary with size
1374.1 KiB
25/05/08 06:38:53 WARN DAGScheduler: Broadcasting large task binary with size
1375.6 KiB
25/05/08 06:38:54 WARN DAGScheduler: Broadcasting large task binary with size
1376.1 KiB
25/05/08 06:38:55 WARN DAGScheduler: Broadcasting large task binary with size
1376.6 KiB
25/05/08 06:38:56 WARN DAGScheduler: Broadcasting large task binary with size
1378.2 KiB
25/05/08 06:38:57 WARN DAGScheduler: Broadcasting large task binary with size
1378.7 KiB
25/05/08 06:38:58 WARN DAGScheduler: Broadcasting large task binary with size
1379.3 KiB
25/05/08 06:38:58 WARN DAGScheduler: Broadcasting large task binary with size
1379.6 KiB
25/05/08 06:38:59 WARN DAGScheduler: Broadcasting large task binary with size
1380.0 KiB
```

```
25/05/08 06:39:00 WARN DAGScheduler: Broadcasting large task binary with size
1381.3 KiB
25/05/08 06:39:00 WARN DAGScheduler: Broadcasting large task binary with size
1381.8 KiB
25/05/08 06:39:01 WARN DAGScheduler: Broadcasting large task binary with size
1382.4 KiB
25/05/08 06:39:02 WARN DAGScheduler: Broadcasting large task binary with size
1383.3 KiB
25/05/08 06:39:02 WARN DAGScheduler: Broadcasting large task binary with size
1384.5 KiB
25/05/08 06:39:03 WARN DAGScheduler: Broadcasting large task binary with size
1385.9 KiB
25/05/08 06:39:04 WARN DAGScheduler: Broadcasting large task binary with size
1386.4 KiB
25/05/08 06:39:05 WARN DAGScheduler: Broadcasting large task binary with size
1387.0 KiB
25/05/08 06:39:05 WARN DAGScheduler: Broadcasting large task binary with size
1387.3 KiB
25/05/08 06:39:06 WARN DAGScheduler: Broadcasting large task binary with size
1389.0 KiB
25/05/08 06:39:06 WARN DAGScheduler: Broadcasting large task binary with size
1389.5 KiB
25/05/08 06:39:07 WARN DAGScheduler: Broadcasting large task binary with size
1389.8 KiB
25/05/08 06:39:08 WARN DAGScheduler: Broadcasting large task binary with size
1390.4 KiB
25/05/08 06:39:08 WARN DAGScheduler: Broadcasting large task binary with size
1391.0 KiB
25/05/08 06:39:08 WARN DAGScheduler: Broadcasting large task binary with size
1392.4 KiB
25/05/08 06:39:09 WARN DAGScheduler: Broadcasting large task binary with size
1392.9 KiB
25/05/08 06:39:10 WARN DAGScheduler: Broadcasting large task binary with size
1393.2 KiB
25/05/08 06:39:10 WARN DAGScheduler: Broadcasting large task binary with size
1394.9 KiB
25/05/08 06:39:11 WARN DAGScheduler: Broadcasting large task binary with size
1395.4 KiB
25/05/08 06:39:12 WARN DAGScheduler: Broadcasting large task binary with size
1396.0 KiB
25/05/08 06:39:12 WARN DAGScheduler: Broadcasting large task binary with size
1396.4 KiB
25/05/08 06:39:13 WARN DAGScheduler: Broadcasting large task binary with size
1396.7 KiB
25/05/08 06:39:14 WARN DAGScheduler: Broadcasting large task binary with size
1398.0 KiB
25/05/08 06:39:14 WARN DAGScheduler: Broadcasting large task binary with size
1398.5 KiB
```

```
25/05/08 06:39:15 WARN DAGScheduler: Broadcasting large task binary with size
1398.8 KiB
25/05/08 06:39:16 WARN DAGScheduler: Broadcasting large task binary with size
1399.2 KiB
25/05/08 06:39:16 WARN DAGScheduler: Broadcasting large task binary with size
1400.9 KiB
25/05/08 06:39:17 WARN DAGScheduler: Broadcasting large task binary with size
1401.4 KiB
25/05/08 06:39:17 WARN DAGScheduler: Broadcasting large task binary with size
1401.7 KiB
25/05/08 06:39:18 WARN DAGScheduler: Broadcasting large task binary with size
1402.3 KiB
25/05/08 06:39:19 WARN DAGScheduler: Broadcasting large task binary with size
1402.9 KiB
25/05/08 06:39:19 WARN DAGScheduler: Broadcasting large task binary with size
1404.5 KiB
25/05/08 06:39:20 WARN DAGScheduler: Broadcasting large task binary with size
1405.0 KiB
25/05/08 06:39:20 WARN DAGScheduler: Broadcasting large task binary with size
1405.3 KiB
25/05/08 06:39:21 WARN DAGScheduler: Broadcasting large task binary with size
1405.6 KiB
25/05/08 06:39:21 WARN DAGScheduler: Broadcasting large task binary with size
1406.2 KiB
25/05/08 06:39:22 WARN DAGScheduler: Broadcasting large task binary with size
1408.3 KiB
25/05/08 06:39:23 WARN DAGScheduler: Broadcasting large task binary with size
1408.8 KiB
25/05/08 06:39:23 WARN DAGScheduler: Broadcasting large task binary with size
1409.4 KiB
25/05/08 06:39:24 WARN DAGScheduler: Broadcasting large task binary with size
1411.0 KiB
25/05/08 06:39:25 WARN DAGScheduler: Broadcasting large task binary with size
1411.5 KiB
25/05/08 06:39:25 WARN DAGScheduler: Broadcasting large task binary with size
1411.8 KiB
25/05/08 06:39:26 WARN DAGScheduler: Broadcasting large task binary with size
1412.4 KiB
25/05/08 06:39:26 WARN DAGScheduler: Broadcasting large task binary with size
1413.0 KiB
25/05/08 06:39:27 WARN DAGScheduler: Broadcasting large task binary with size
1414.3 KiB
25/05/08 06:39:28 WARN DAGScheduler: Broadcasting large task binary with size
1414.8 KiB
25/05/08 06:39:29 WARN DAGScheduler: Broadcasting large task binary with size
1415.3 KiB
25/05/08 06:39:29 WARN DAGScheduler: Broadcasting large task binary with size
1415.6 KiB
```

```
25/05/08 06:39:29 WARN DAGScheduler: Broadcasting large task binary with size
1415.9 KiB
25/05/08 06:39:29 WARN DAGScheduler: Broadcasting large task binary with size
1417.5 KiB
25/05/08 06:39:30 WARN DAGScheduler: Broadcasting large task binary with size
1418.0 KiB
25/05/08 06:39:31 WARN DAGScheduler: Broadcasting large task binary with size
1418.3 KiB
25/05/08 06:39:44 WARN DAGScheduler: Broadcasting large task binary with size
1279.5 KiB
25/05/08 06:39:59 WARN DAGScheduler: Broadcasting large task binary with size
1279.5 KiB
```

```
+--------------------+------------------+-----------------+
|Model               |Train_Accuracy    |Test_Accuracy    |
+--------------------+------------------+-----------------+
|RandomForest        |0.7671222246173744|0.764540413647954|
|DecisionTree        |1.0               |1.0              |
|GradientBoostedTrees|1.0               |1.0              |
+--------------------+------------------+-----------------+
```

```
                  Model  Train_Accuracy  Test_Accuracy
0          RandomForest        0.767122        0.76454
1          DecisionTree        1.000000        1.00000
2  GradientBoostedTrees        1.000000        1.00000
```

0.3 Model Performance Analysis

Model	Train Accuracy	Test Accuracy
Random Forest	0.7671	0.7645
Decision Tree	1.0000	1.0000
Gradient Boosted Trees	1.0000	1.0000

0.3.1 Conclusions

1. **Random Forest**
   - Moderate accuracy on both training and test sets.
   - Very small gap between training and test accuracy → **Good generalization**.
   - Likely a **realistically performing model** with **no overfitting**.
2. **Decision Tree**
   - Perfect accuracy on both train and test sets.
   - This is **suspicious** and may indicate **overfitting** or **data leakage**.
   - Should be examined further, especially if the dataset is large or complex.
3. **Gradient Boosted Trees**
   - Also shows perfect scores.
   - Similar concerns as Decision Tree: may suggest **overfitting** or **data leakage**.
   - Needs **validation of training/testing process**.

0.3.2 Overall Summary

- **Random Forest** is the most trustworthy model in this comparison.
- **Decision Tree** and **Gradient Boosted Trees** require careful investigation to rule out data leakage or overly simplistic data