Table of Contents

Load Libraries

In [1]:

```
# Import PySpark related modules
import pyspark
from pyspark.rdd import RDD
from pyspark.sql import Row
from pyspark.sql import DataFrame
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql import functions
from pyspark.sql.functions import lit, desc, col, size, array_contains, isnan, u
```

```python
from pyspark.sql.types import *
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.regression import GBTRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.feature import Imputer
from pyspark.ml.regression import GeneralizedLinearRegression
from pyspark.ml.regression import DecisionTreeRegressor
from pyspark.ml.regression import RandomForestRegressor
from pyspark.sql.functions import *

# Import other modules not related to PySpark
import os
import sys
import pandas as pd
from pandas import DataFrame
import numpy as np
%matplotlib inline
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import matplotlib
#from mpl_toolkits.mplot3d import Axes3D
import math
from IPython.core.interactiveshell import InteractiveShell
from datetime import *
import statistics as stats

# This helps auto print out the items without explixitly using 'print'
InteractiveShell.ast_node_interactivity = "all"

import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
```

Initialize pyspark framework

In [2]:

```python
conf = pyspark.SparkConf().setAll([('spark.master', 'local[4]'),
                                   ('spark.app.name', 'Python Spark SQL Demo')])
spark = SparkSession.builder.config(conf=conf).getOrCreate()
```

Load data

In [3]:

```python
!pwd
```

```
/home/work/ecommerce
```

In [4]:

```python
!ls
```

```
'E-commerce EDA.ipynb'
'E-commerce Sales Forecast - time series.ipynb'
'E-commerce Sales Forecast.ipynb'
 customer_reviews_dataset.csv
 customers_dataset.csv
 geolocation_dataset.csv
 launch.sh
 order_items_dataset.csv
 order_payments_dataset.csv
 orders_dataset.csv
 product_category_name_translation.csv
 products_dataset.csv
 sellers_dataset.csv
```

In [5]:

```
!hadoop fs -mkdir /data
```

```
mkdir: `/data': File exists
```

In [6]:

```
!hadoop fs -copyFromLocal products_dataset.csv /data
```

```
copyFromLocal: `/data/products_dataset.csv': File exists
```

In [7]:

```
!hadoop fs -copyFromLocal product_category_name_translation.csv /data
```

```
copyFromLocal: `/data/product_category_name_translation.csv': File exists
```

In [8]:

```
!hadoop fs -copyFromLocal customers_dataset.csv /data
```

```
copyFromLocal: `/data/customers_dataset.csv': File exists
```

In [9]:

```
!hadoop fs -copyFromLocal sellers_dataset.csv /data
```

```
copyFromLocal: `/data/sellers_dataset.csv': File exists
```

In [10]:

```
!hadoop fs -copyFromLocal orders_dataset.csv /data
```

```
copyFromLocal: `/data/orders_dataset.csv': File exists
```

In [11]:

```
!hadoop fs -copyFromLocal order_payments_dataset.csv /data
```

```
copyFromLocal: `/data/order_payments_dataset.csv': File exists
```

In [12]:

```
!hadoop fs -copyFromLocal order_items_dataset.csv /data
```

```
copyFromLocal: `/data/order_items_dataset.csv': File exists
```

In [13]:

```
!hadoop fs -copyFromLocal geolocation_dataset.csv /data
```

```
copyFromLocal: `/data/geolocation_dataset.csv': File exists
```

In [14]:

```
!hadoop fs -copyFromLocal customer_reviews_dataset.csv /data
```

```
copyFromLocal: `/data/customer_reviews_dataset.csv': File exists
```

In [15]:

```
DATA_PATH="hdfs:///data/"
```

```
products_dataset = spark.read.csv(DATA_PATH+"products_dataset.csv", header=True,
product_category_name_translation = spark.read.csv(DATA_PATH+"product_category_n
customers_dataset = spark.read.csv(DATA_PATH+"customers_dataset.csv", header=Tru
sellers_dataset = spark.read.csv(DATA_PATH+"sellers_dataset.csv", header=True, i
orders_dataset = spark.read.csv(DATA_PATH+"orders_dataset.csv", header=True, inf
order_payments_dataset = spark.read.csv(DATA_PATH+"order_payments_dataset.csv",
order_items_dataset = spark.read.csv(DATA_PATH+"order_items_dataset.csv", header
geolocation_dataset = spark.read.csv(DATA_PATH+"geolocation_dataset.csv", header
customer_reviews_dataset = spark.read.csv(DATA_PATH+"customer_reviews_dataset.cs
```

In [16]:

```
# Show sample data in each dataset
products_dataset.show(3)
product_category_name_translation.show(3)
customers_dataset.show(3)
sellers_dataset.show(3)
orders_dataset.show(3)
order_payments_dataset.show(3)
order_items_dataset.show(3)
geolocation_dataset.show(3)
customer_reviews_dataset.show(3)
```

```
+--------------------+---------------------+-------------------+----------------
----------+------------------+----------------+-----------------+---------------
--+-----------------+
|          product_id|product_category_name|product_name_lenght|product_descript
ion_lenght|product_photos_qty|product_weight_g|product_length_cm|product_height_
cm|product_width_cm|
+--------------------+---------------------+-------------------+----------------
----------+------------------+----------------+-----------------+---------------
--+-----------------+
|1e9e8ef04dbcff454...|           perfumaria|                 40|
287|                 1|             225|               16|                10|
14|
|3aa071139cb16b67c...|                artes|                 44|
276|                 1|            1000|               30|                18|
20|
|96bd76ec8810374ed...|        esporte_lazer|                 46|
250|                 1|             154|               18|                 9|
15|
+--------------------+---------------------+-------------------+----------------
----------+------------------+----------------+-----------------+---------------
--+-----------------+
only showing top 3 rows

+---------------------+-----------------------------+
|product_category_name|product_category_name_english|
+---------------------+-----------------------------+
|         beleza_saude|                health_beauty|
| informatica_acess...|         computers_accesso...|
|           automotivo|                         auto|
+---------------------+-----------------------------+
only showing top 3 rows

+--------------------+--------------------+------------------------+------------
--------+--------------+
|         customer_id|  customer_unique_id|customer_zip_code_prefix|       custo
mer_city|customer_state|
+--------------------+--------------------+------------------------+------------
--------+--------------+
|06b8999e2fba1a1fb...|861eff4711a542e4b...|                   14409|
franca|            SP|
|18955e83d337fd6b2...|290c77bc529b7ac93...|                    9790|sao bernardo
```

```
do c...|            SP|
|4e7b3e00288586ebd...|060e732b5b29e8181...|                    1151|           s
ao paulo|            SP|
+--------------------+--------------------+------------------------+------------
--------+--------------+
only showing top 3 rows

+--------------------+----------------------+--------------+------------+
|           seller_id|seller_zip_code_prefix|   seller_city|seller_state|
+--------------------+----------------------+--------------+------------+
|3442f8959a84dea7e...|                 13023|      campinas|          SP|
|d1b65fc7debc3361e...|                 13844|    mogi guacu|          SP|
|ce3ad9de960102d06...|                 20031|rio de janeiro|          RJ|
+--------------------+----------------------+--------------+------------+
only showing top 3 rows

+--------------------+--------------------+------------+------------------------
+-------------------+---------------------------+----------------------------+--
---------------------------+
|            order_id|         customer_id|order_status|order_purchase_timestamp
|  order_approved_at|order_carrier_delivery_date|order_customer_delivery_date|or
der_estimated_delivery_date|
+--------------------+--------------------+------------+------------------------
+-------------------+---------------------------+----------------------------+--
---------------------------+
|e481f51cbdc54678b...|9ef432eb625129730...|   delivered|     2017-10-02 10:56:33
|2017-10-02 11:07:15|        2017-10-04 19:55:00|         2017-10-10 21:25:13|
2017-10-18 00:00:00|
|53cdb2fc8bc7dce0b...|b0830fb4747a6c6d2...|   delivered|     2018-07-24 20:41:37
|2018-07-26 03:24:27|        2018-07-26 14:31:00|         2018-08-07 15:27:45|
2018-08-13 00:00:00|
|47770eb9100c2d0c4...|41ce2a54c0b03bf34...|   delivered|     2018-08-08 08:38:49
|2018-08-08 08:55:23|        2018-08-08 13:50:00|         2018-08-17 18:06:29|
2018-09-04 00:00:00|
+--------------------+--------------------+------------+------------------------
+-------------------+---------------------------+----------------------------+--
---------------------------+
only showing top 3 rows

+--------------------+------------------+------------+--------------------+-----
--------+
|            order_id|payment_sequential|payment_type|payment_installments|payme
nt_value|
+--------------------+------------------+------------+--------------------+-----
--------+
|b81ef226f3fe1789b...|                 1| credit_card|                   8|
99.33|
|a9810da82917af2d9...|                 1| credit_card|                   1|
24.39|
|25e8ea4e93396b6fa...|                 1| credit_card|                   1|
65.71|
+--------------------+------------------+------------+--------------------+-----
--------+
only showing top 3 rows

+--------------------+-------------+--------------------+--------------------+--
-----------------+-----+-------------+
|            order_id|order_item_id|          product_id|           seller_id|sh
ipping_limit_date|price|freight_value|
+--------------------+-------------+--------------------+--------------------+--
-----------------+-----+-------------+
|00010242fe8c5a6d1...|            1|4244733e06e7ecb49...|48436dade18ac8b2b...|20
17-09-19 09:45:35| 58.9|        13.29|
|00018f77f2f0320c5...|            1|e5f2d52b802189ee6...|dd7ddc04e1b6c2c61...|20
17-05-03 11:05:13|239.9|        19.93|
```

```
|000229ec398224ef6...|           1|c777355d18b72b67a...|5b51032eddd242adc...|20
18-01-18 14:48:30|199.0|        17.87|
+--------------------+-------------+--------------------+--------------------+--
-----------------+-----+-------------+
only showing top 3 rows

+------------------+-------------------+-------------------+---------+---------+
|geo_zip_code_prefix|            geo_lat|            geo_lng| geo_city|geo_state|
+------------------+-------------------+-------------------+---------+---------+
|              1037| -23.54562128115268|-46.63929204800168|sao paulo|       SP|
|              1046|-23.546081127035535|-46.64482029837157|sao paulo|       SP|
|              1046| -23.54612896641469|-46.64295148361138|sao paulo|       SP|
+------------------+-------------------+-------------------+---------+---------+
only showing top 3 rows

+--------------------+--------------------+------------+--------------------+----
-----------------+--------------------+----------------------+
|           review_id|            order_id|survey_score|survey_review_title|surv
ey_review_content|   survey_send_date|survey_completion_date|
+--------------------+--------------------+------------+--------------------+----
-----------------+--------------------+----------------------+
|7bc2406110b926393...|73fc7af87114b3971...|           4|               null|
null|2018-01-18 00:00:00|   2018-01-18 21:46:59|
|80e641a11e56f04c1...|a548910a1c6147796...|           5|               null|
null|2018-03-10 00:00:00|   2018-03-11 03:05:13|
|228ce5500dc1d8e02...|f9e4b658b201a9f2e...|           5|               null|
null|2018-02-17 00:00:00|   2018-02-18 14:36:24|
+--------------------+--------------------+------------+--------------------+----
-----------------+--------------------+----------------------+
only showing top 3 rows
```

Create functions

In [17]:

```python
def fill_na(df, strategy):
    imputer = Imputer(
        strategy=strategy,
        inputCols=df.columns,
        outputCols=["{}_imputed".format(c) for c in df.columns]
    )

    new_df = imputer.fit(df).transform(df)

    # Select the newly created columns with all filled values
    new_df = new_df.select([c for c in new_df.columns if "imputed" in c])

    for col in new_df.columns:
        new_df = new_df.withColumnRenamed(col, col.split("_imputed")[0])

    return new_df

def plot_function(predictions):
    figure(figsize = (20, 20), dpi = 80)

    plt.title('Results on test data')
    plt.xlabel('row number')
    plt.ylabel('labels and predictions')

    num = 290
    lr_p_1 = predictions.select('label').toPandas()
    lr_p_2 = predictions.select('prediction').toPandas()
```

```
    x = np.arange(num)
    y_list_1 = []
    y_list_2 = []
    y_list_1 = lr_p_1[:290]
    y_list_2 = lr_p_2[:290]

    plt.scatter(x, y_list_1, color = 'blue', marker = '*', label = 'labels', alp
    plt.scatter(x, y_list_2, color = 'red', marker = 'o', label = 'predictions',

    plt.xlim(-10, 300)
    plt.ylim(-10, 1000)

    plt.legend()
    plt.grid()
    plt.show()

def fit_predict_plot_function(featureIndexer, model, trainingData, testData):

    # Chain indexer and GBT in a Pipeline
    pipeline = Pipeline(stages=[featureIndexer, model])

    # Train model.  This also runs the indexer.
    model = pipeline.fit(trainingData)

    # Make predictions.
    predictions = model.transform(testData)

    # Select example rows to display.
    predictions.select("prediction", "label", "features").show(5)

    # Select (prediction, true label) and compute test error
    evaluator = RegressionEvaluator(labelCol="label", predictionCol="prediction"
    rmse = evaluator.evaluate(predictions)
    print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)

    model_= model.stages[1]
    print(model_)  # summary only

    plot_function(predictions)

    return rmse
```

Data preparation process

Merge datasets

In [18]:

```
# Merge these four dataframes together: orders_dataset, order_payments_dataset,
df_merge_1 = orders_dataset.join(order_payments_dataset, on=['order_id'], how='i
df_merge_2 = df_merge_1.join(order_items_dataset, on=['order_id'], how='inner')
df_merge_3 = df_merge_2.join(customer_reviews_dataset, on=['order_id'], how='inn

# Select 2500 data to speed up running the notebook. You can remove this line fo
# But that will consume quite a lot of time and may cause kernel crashes.
df_merge_3 = df_merge_3.limit(2500)
df_merge_3.printSchema()
df_merge_3.show(3)
```

```
print("Data count = {}".format(df_merge_3.count()))
```

```
root
 |-- order_id: string (nullable = true)
 |-- customer_id: string (nullable = true)
 |-- order_status: string (nullable = true)
 |-- order_purchase_timestamp: string (nullable = true)
 |-- order_approved_at: string (nullable = true)
 |-- order_carrier_delivery_date: string (nullable = true)
 |-- order_customer_delivery_date: string (nullable = true)
 |-- order_estimated_delivery_date: string (nullable = true)
 |-- payment_sequential: integer (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- payment_installments: integer (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- order_item_id: integer (nullable = true)
 |-- product_id: string (nullable = true)
 |-- seller_id: string (nullable = true)
 |-- shipping_limit_date: string (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- review_id: string (nullable = true)
 |-- survey_score: string (nullable = true)
 |-- survey_review_title: string (nullable = true)
 |-- survey_review_content: string (nullable = true)
 |-- survey_send_date: string (nullable = true)
 |-- survey_completion_date: string (nullable = true)

+--------------------+--------------------+------------+------------------------
+-------------------+----------------------------+----------------------------+--
---------------------------+------------------+------------+--------------------
+-------------+-------------+--------------------+--------------------+---------
----------+-----+-------------+--------------------+------------+---------------
----+--------------------+-------------------+----------------------+
|            order_id|         customer_id|order_status|order_purchase_timestamp
|  order_approved_at|order_carrier_delivery_date|order_customer_delivery_date|or
der_estimated_delivery_date|payment_sequential|payment_type|payment_installments
|payment_value|order_item_id|          product_id|           seller_id|shipping_
limit_date|price|freight_value|           review_id|survey_score|survey_review_t
itle|survey_review_content|   survey_send_date|survey_completion_date|
+--------------------+--------------------+------------+------------------------
+-------------------+----------------------------+----------------------------+--
---------------------------+------------------+------------+--------------------
+-------------+-------------+--------------------+--------------------+---------
----------+-----+-------------+--------------------+------------+---------------
----+--------------------+-------------------+----------------------+
|014405982914c2cde...|2de342d6e5905a5a8...|   delivered|     2017-07-26 17:38:47
|2017-07-26 17:50:17|        2017-07-27 19:39:52|         2017-07-31 15:53:33|
2017-08-17 00:00:00|                 1| credit_card|                   7|
78.43|            1|6782d593f63105318...|325f3178fb58e2a97...|2017-08-01 17:50:1
7| 27.9|         3.81|ba5e6d78da2b2c3f5...|           5|               null|
null|2017-08-01 00:00:00|   2017-08-04 19:34:54|
|014405982914c2cde...|2de342d6e5905a5a8...|   delivered|     2017-07-26 17:38:47
|2017-07-26 17:50:17|        2017-07-27 19:39:52|         2017-07-31 15:53:33|
2017-08-17 00:00:00|                 1| credit_card|                   7|
78.43|            2|e95ee6822b66ac605...|a17f621c590ea0fab...|2017-08-01 17:50:1
7|21.33|        25.39|ba5e6d78da2b2c3f5...|           5|               null|
null|2017-08-01 00:00:00|   2017-08-04 19:34:54|
|019886de8f385a39b...|8cf88d7ba142365ef...|   delivered|     2018-02-10 12:52:51
|2018-02-10 13:08:12|        2018-02-14 15:28:51|         2018-02-23 02:03:03|
2018-03-14 00:00:00|                 1| credit_card|                   2|
188.4|            1|e9a69340883a438c3...|1b4c3a6f53068f0b6...|2018-02-15 13:08:1
2|159.9|         28.5|3a6ebeb7f45583720...|           5|               null| Pro
duto foi entre...|2018-02-24 00:00:00|   2018-02-27 15:10:57|
```

```
+--------------------+--------------------+------------+------------------------
+-------------------+--------------------------+----------------------------+--
--------------------------+------------------+------------+-------------------
+-------------+-------------+--------------------+-------------------+---------
----------+-----+-------------+--------------------+------------+--------------
----+---------------------+-------------------+----------------------+
only showing top 3 rows

Data count = 2500
```

Data cleaning

Drop unused features

In [19]:

```
df_merge = df_merge_3.drop('customer_id', 'order_status', \
                           'order_approved_at', 'order_carrier_delivery_date', '
                           'order_estimated_delivery_date', 'payment_sequential'
                           'order_item_id', 'product_id', 'seller_id', 'shipping
                           'review_id', 'survey_review_title', 'survey_review_co
df_merge.printSchema()
df_merge.show(3)

print("Data count = {}".format(df_merge.count()))
```

```
root
 |-- order_id: string (nullable = true)
 |-- order_purchase_timestamp: string (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- survey_score: string (nullable = true)

+--------------------+------------------------+-------------+-----+-------------
+------------+
|            order_id|order_purchase_timestamp|payment_value|price|freight_value
|survey_score|
+--------------------+------------------------+-------------+-----+-------------
+------------+
|e481f51cbdc54678b...|     2017-10-02 10:56:33|        18.59|29.99|         8.72
|           4|
|e481f51cbdc54678b...|     2017-10-02 10:56:33|          2.0|29.99|         8.72
|           4|
|e481f51cbdc54678b...|     2017-10-02 10:56:33|        18.12|29.99|         8.72
|           4|
+--------------------+------------------------+-------------+-----+-------------
+------------+
only showing top 3 rows

Data count = 2500
```

Drop duplicated values

In [20]:

```
df_merge = df_merge.drop_duplicates(['order_id'])

df_merge.printSchema()
df_merge.show(3)

print("Data count = {}".format(df_merge.count()))
```

```
root
```

```
 |-- order_id: string (nullable = true)
 |-- order_purchase_timestamp: string (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- survey_score: string (nullable = true)

+--------------------+------------------------+-------------+-----+-------------
+------------+
|            order_id|order_purchase_timestamp|payment_value|price|freight_value
|survey_score|
+--------------------+------------------------+-------------+-----+-------------
+------------+
|00125cb692d048878...|     2017-03-23 12:21:17|       135.41|109.9|        25.51
|           5|
|00571ded73b3c0619...|     2017-05-18 20:59:24|       389.82|179.9|        15.01
|           5|
|006dd93155bc2abd8...|     2017-12-02 01:20:28|        57.68| 49.9|         7.78
|           5|
+--------------------+------------------------+-------------+-----+-------------
+------------+
only showing top 3 rows

Data count = 2088
```

Drop nullable values

In [21]:

```
df_merge = df_merge.dropna()

df_merge.printSchema()
df_merge.show(3)

print("Data count = {}".format(df_merge.count()))
```

```
root
 |-- order_id: string (nullable = true)
 |-- order_purchase_timestamp: string (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- survey_score: string (nullable = true)

+--------------------+------------------------+-------------+-----+-------------
+------------+
|            order_id|order_purchase_timestamp|payment_value|price|freight_value
|survey_score|
+--------------------+------------------------+-------------+-----+-------------
+------------+
|00125cb692d048878...|     2017-03-23 12:21:17|       135.41|109.9|        25.51
|           5|
|00571ded73b3c0619...|     2017-05-18 20:59:24|       389.82|179.9|        15.01
|           5|
|006dd93155bc2abd8...|     2017-12-02 01:20:28|        57.68| 49.9|         7.78
|           5|
+--------------------+------------------------+-------------+-----+-------------
+------------+
only showing top 3 rows

Data count = 2088
```

Convert numeric to double

In [22]:

```
df_merge = df_merge.select('order_id', 'order_purchase_timestamp', 'price', 'fre

df_merge.printSchema()
df_merge.show(3)

print("Data count = {}".format(df_merge.count()))
```

```
root
 |-- order_id: string (nullable = true)
 |-- order_purchase_timestamp: string (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- survey_score: double (nullable = true)

+--------------------+------------------------+-----+-------------+-------------
+------------+
|            order_id|order_purchase_timestamp|price|freight_value|payment_value
|survey_score|
+--------------------+------------------------+-----+-------------+-------------
+------------+
|00125cb692d048878...|     2017-03-23 12:21:17|109.9|        25.51|       135.41
|         5.0|
|00571ded73b3c0619...|     2017-05-18 20:59:24|179.9|        15.01|       389.82
|         5.0|
|006dd93155bc2abd8...|     2017-12-02 01:20:28| 49.9|         7.78|        57.68
|         5.0|
+--------------------+------------------------+-----+-------------+-------------
+------------+
only showing top 3 rows

Data count = 2088
```

Convert datetime to date

In [23]:

```
df_date = df_merge.select(date_format(col('order_purchase_timestamp'),"yyyy-MM-d

df_date.printSchema()
df_date.show(3)

print("Data count = {}".format(df_date.count()))
```

```
root
 |-- order_purchase_date: date (nullable = true)

+-------------------+
|order_purchase_date|
+-------------------+
|         2017-03-23|
|         2017-05-18|
|         2017-12-02|
+-------------------+
only showing top 3 rows

Data count = 2088
```

Convert date to quarter

In [24]:

```
df_quarter = df_date.select(quarter('order_purchase_date').alias('quarter'))

df_quarter.printSchema()
```

```
df_quarter.show(3)

print("Data count = {}".format(df_quarter.count()))
```

```
root
 |-- quarter: integer (nullable = true)

+-------+
|quarter|
+-------+
|      1|
|      2|
|      4|
+-------+
only showing top 3 rows

Data count = 2088
```

Add id column

In [25]:

```
df_quarter = df_quarter.withColumn("id", monotonically_increasing_id())
df_merge = df_merge.withColumn("id", monotonically_increasing_id())

df_quarter.printSchema()
df_quarter.show(3)

print("Data count = {}".format(df_quarter.count()))

df_merge.printSchema()
df_merge.show(3)

print("Data count = {}".format(df_merge.count()))
```

```
root
 |-- quarter: integer (nullable = true)
 |-- id: long (nullable = false)

+-------+---+
|quarter| id|
+-------+---+
|      1|  0|
|      2|  1|
|      4|  2|
+-------+---+
only showing top 3 rows

Data count = 2088
root
 |-- order_id: string (nullable = true)
 |-- order_purchase_timestamp: string (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- survey_score: double (nullable = true)
 |-- id: long (nullable = false)

+--------------------+------------------------+-----+-------------+-------------
+------------+---+
|            order_id|order_purchase_timestamp|price|freight_value|payment_value
|survey_score| id|
+--------------------+------------------------+-----+-------------+-------------
+------------+---+
```

```
|00125cb692d048878...|     2017-03-23 12:21:17|109.9|        25.51|       135.41
|           5.0|  0|
|00571ded73b3c0619...|     2017-05-18 20:59:24|179.9|        15.01|       389.82
|           5.0|  1|
|006dd93155bc2abd8...|     2017-12-02 01:20:28| 49.9|         7.78|        57.68
|           5.0|  2|
+--------------------+------------------------+-----+-------------+-------------
+------------+---+
only showing top 3 rows

Data count = 2088
```

In [26]:

```
# Merge df_merge and quarter_df together and drop unused id column and order_pur
df = df_merge.join(df_quarter, on=["id"], how="left").drop("id", "order_purchase

# Drop nullable values
df = df.dropna()

df.printSchema()
df.show(3)

print("Data count = {}".format(df.count()))
```

```
root
 |-- order_id: string (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- survey_score: double (nullable = true)
 |-- quarter: integer (nullable = true)

+--------------------+-----+-------------+-------------+------------+-------+
|            order_id|price|freight_value|payment_value|survey_score|quarter|
+--------------------+-----+-------------+-------------+------------+-------+
|00125cb692d048878...|109.9|        25.51|       135.41|         5.0|      1|
|00571ded73b3c0619...|179.9|        15.01|       389.82|         5.0|      2|
|006dd93155bc2abd8...| 49.9|         7.78|        57.68|         5.0|      4|
+--------------------+-----+-------------+-------------+------------+-------+
only showing top 3 rows

Data count = 2088
```

Feature engineering

In [27]:

```
# Select features for ml models.
df = df.select('price', 'freight_value', 'payment_value', 'survey_score', 'quart

df.printSchema()
df.show(10)

print("Data count = {}".format(df.count()))
```

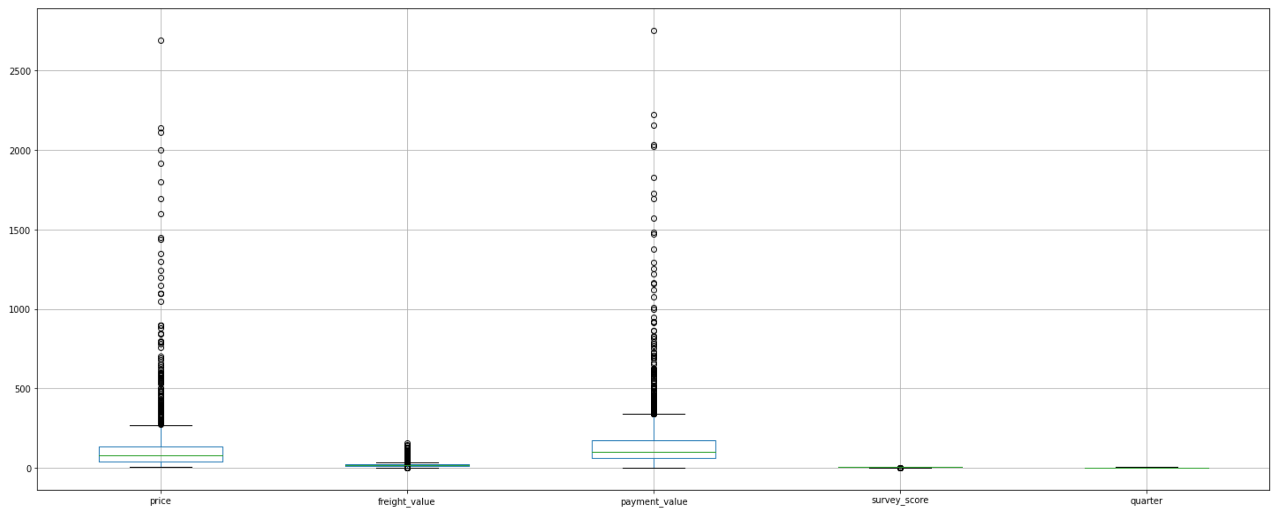
```
root
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- survey_score: double (nullable = true)
 |-- quarter: integer (nullable = true)

+------+-------------+-------------+------------+-------+
| price|freight_value|payment_value|survey_score|quarter|
```

```
+------+-------------+-------------+------------+-------+
| 109.9|        25.51|       135.41|         5.0|      1|
| 179.9|        15.01|       389.82|         5.0|      2|
|  49.9|         7.78|        57.68|         5.0|      4|
|  99.9|        14.35|       232.72|         4.0|      4|
| 149.0|        45.12|       194.12|         5.0|      1|
| 118.9|        18.93|       137.83|         5.0|      2|
|243.37|        53.83|        297.2|         5.0|      1|
|  89.9|        17.88|       215.56|         5.0|      4|
| 56.99|        15.15|        72.14|         1.0|      2|
| 39.49|         8.27|        95.52|         5.0|      4|
+------+-------------+-------------+------------+-------+
only showing top 10 rows

Data count = 2088
```

Feature visualization

In [28]:

```python
plot = df.toPandas().boxplot(figsize = (25,10))
```

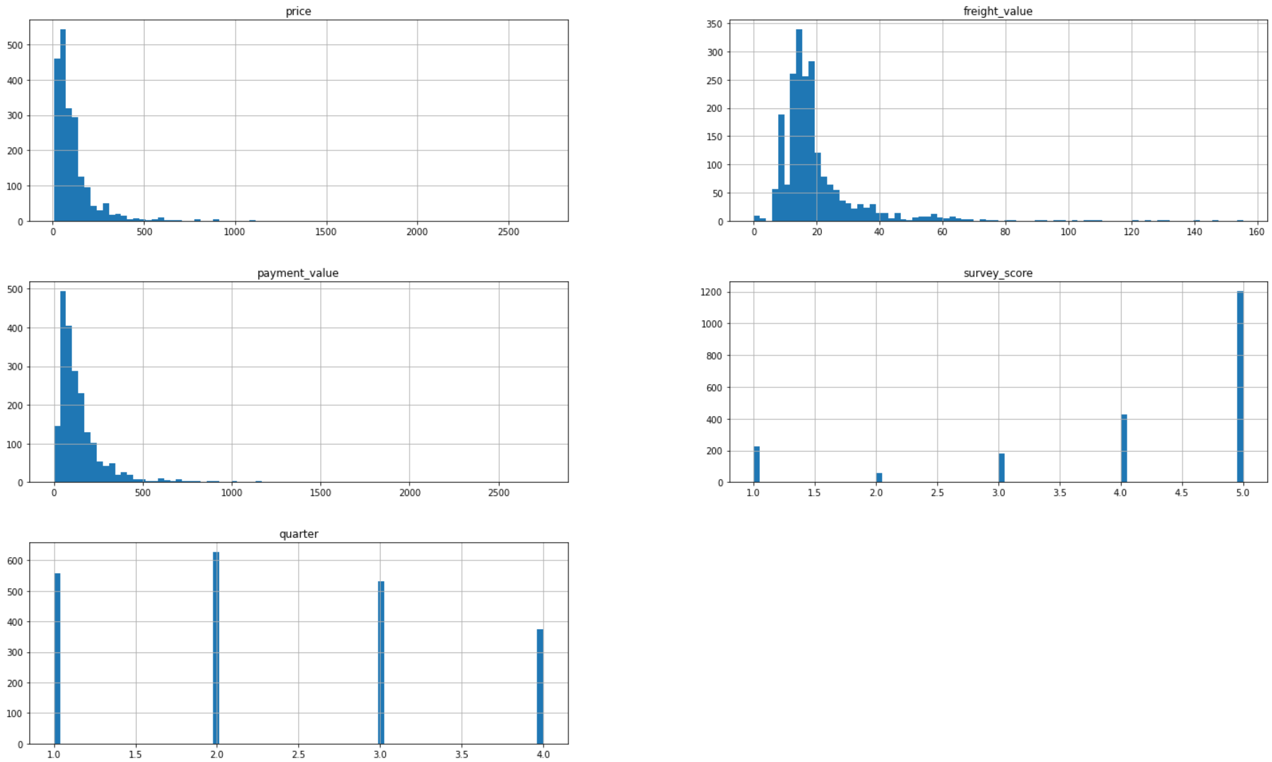


In [29]:

```python
df.toPandas().hist(figsize = (25,15), bins = 80)
```

Out[29]:

```
array([[<AxesSubplot:title={'center':'price'}>,
        <AxesSubplot:title={'center':'freight_value'}>],
       [<AxesSubplot:title={'center':'payment_value'}>,
        <AxesSubplot:title={'center':'survey_score'}>],
       [<AxesSubplot:title={'center':'quarter'}>, <AxesSubplot:>]],
      dtype=object)
```

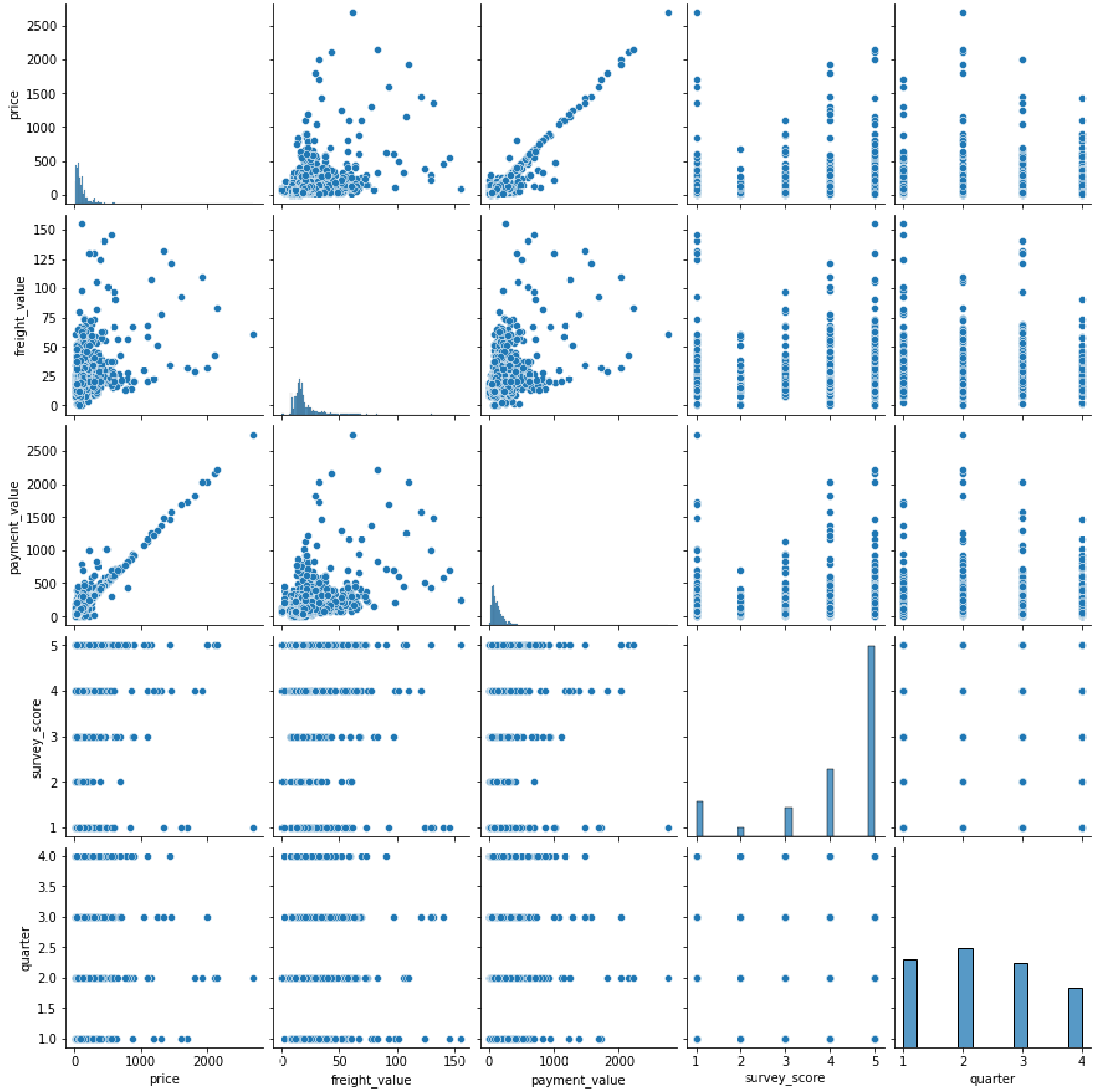


In [30]:

```python
sns.pairplot(df.toPandas())
plt.show()
```

Out[30]:

```
<seaborn.axisgrid.PairGrid at 0x7f5560ab3e80>
```



In [31]:
```
display(df.toPandas().describe())
```

	price	freight_value	payment_value	survey_score	quarter
count	2088.000000	2088.000000	2088.000000	2088.000000	2088.000000
mean	123.269090	19.822749	153.551015	4.115900	2.343870
std	182.136296	13.999481	193.904080	1.312275	1.057318
min	5.990000	0.000000	0.000000	1.000000	1.000000
25%	41.200000	13.142500	61.087500	4.000000	1.000000
50%	78.000000	16.235000	102.995000	5.000000	2.000000
75%	132.750000	20.990000	172.140000	5.000000	3.000000
max	2690.000000	155.390000	2751.240000	5.000000	4.000000

Create feature vector

In [32]:

```
vectorAssembler = VectorAssembler(inputCols = ['quarter', 'price','freight_value
v_df = vectorAssembler.transform(df)
v_df = v_df.select(['features', 'payment_value'])
v_df = v_df.withColumnRenamed('payment_value', 'label')

v_df.printSchema()
v_df.show(10, False)

print("Data count = {}".format(v_df.count()))
```

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

+----------------------+------+
|features              |label |
+----------------------+------+
|[1.0,109.9,25.51,5.0] |135.41|
|[2.0,179.9,15.01,5.0] |389.82|
|[4.0,49.9,7.78,5.0]   |57.68 |
|[4.0,99.9,14.35,4.0]  |232.72|
|[1.0,149.0,45.12,5.0] |194.12|
|[2.0,118.9,18.93,5.0] |137.83|
|[1.0,243.37,53.83,5.0]|297.2 |
|[4.0,89.9,17.88,5.0]  |215.56|
|[2.0,56.99,15.15,1.0] |72.14 |
|[4.0,39.49,8.27,5.0]  |95.52 |
+----------------------+------+
only showing top 10 rows

Data count = 2088
```

Split datasets

In [33]:

```
train_df, test_df = v_df.randomSplit([0.7, 0.3], seed = 42)

train_df.show(10, False)
print('number of rows in train dataframe:', train_df.count())

test_df.show(10, False)
print('number of rows in test dataframe:',test_df.count())
```

```
+---------------------+-----+
|features             |label|
+---------------------+-----+
|[1.0,6.0,8.72,3.0]   |14.72|
|[1.0,7.8,10.96,4.0]  |18.76|
|[1.0,9.9,8.72,5.0]   |18.62|
|[1.0,9.9,8.72,5.0]   |18.62|
|[1.0,10.99,14.52,5.0]|25.51|
|[1.0,11.5,10.96,5.0] |22.46|
|[1.0,12.5,14.1,4.0]  |53.2 |
|[1.0,12.9,17.63,5.0] |30.53|
|[1.0,12.98,15.1,5.0] |28.08|
|[1.0,14.67,17.63,5.0]|32.3 |
+---------------------+-----+
only showing top 10 rows
```

```
number of rows in train dataframe: 1513
+----------------------+-----+
|features              |label|
+----------------------+-----+
|[1.0,9.5,7.78,3.0]    |17.28|
|[1.0,10.99,18.23,4.0]|29.22|
|[1.0,11.87,7.39,3.0]  |25.0 |
|[1.0,12.5,11.85,5.0]  |20.0 |
|[1.0,13.9,10.96,5.0]  |24.86|
|[1.0,14.5,10.96,3.0]  |25.46|
|[1.0,14.6,7.78,5.0]   |22.38|
|[1.0,15.0,15.1,5.0]   |30.1 |
|[1.0,15.9,7.78,5.0]   |23.68|
|[1.0,16.5,12.48,5.0]  |28.98|
+----------------------+-----+
only showing top 10 rows

number of rows in test dataframe: 575
```

Standardize data

In [34]:

```python
scaler = StandardScaler(inputCol = "features", outputCol = "features_scaled", wi
scalerModel = scaler.fit(train_df)
```

In [35]:

```python
train_df_s = scalerModel.transform(train_df)
test_df_s  = scalerModel.transform(test_df)
train_df_s.show(10, False)
test_df_s.show(10, False)
```

```
+----------------------+-----+-------------------------------------------------------
---------------------------+
|features              |label|features_scaled
|
+----------------------+-----+-------------------------------------------------------
---------------------------+
|[1.0,6.0,8.72,3.0]    |14.72|[0.9443902735590791,0.03600162908663947,0.641141896
6613659,2.264242293977733] |
|[1.0,7.8,10.96,4.0]   |18.76|[0.9443902735590791,0.0468021178126313,0.8058388976
385975,3.018989725303644]  |
|[1.0,9.9,8.72,5.0]    |18.62|[0.9443902735590791,0.05940268799295512,0.641141896
6613659,3.773737156629552]|
|[1.0,9.9,8.72,5.0]    |18.62|[0.9443902735590791,0.05940268799295512,0.641141896
6613659,3.773737156629552]|
|[1.0,10.99,14.52,5.0]|25.51|[0.9443902735590791,0.06594298394369462,1.067589488
477412,3.773737156629552] |
|[1.0,11.5,10.96,5.0]  |22.46|[0.9443902735590791,0.06900312241605897,0.805838897
6385975,3.773737156629552]|
|[1.0,12.5,14.1,4.0]   |53.2 |[0.9443902735590791,0.07500339393049889,1.036708800
794181,3.018989725303644]  |
|[1.0,12.9,17.63,5.0]  |30.53|[0.9443902735590791,0.07740350253627484,1.296253628
2270503,3.773737156629552]|
|[1.0,12.98,15.1,5.0]  |28.08|[0.9443902735590791,0.07788352425743005,1.110234247
6590166,3.773737156629552]|
|[1.0,14.67,17.63,5.0]|32.3 |[0.9443902735590791,0.08802398311683349,1.296253628
2270503,3.773737156629552]|
+----------------------+-----+-------------------------------------------------------
---------------------------+
only showing top 10 rows

+----------------------+-----+-------------------------------------------------------
---------------------------+
```

```
|features             |label|features_scaled
|
+--------------------+-----+---------------------------------------------------
---------------------------+
|[1.0,9.5,7.78,3.0]   |17.28|[0.9443902735590791,0.05700257938717915,0.572027976
6084204,2.264242293977733] |
|[1.0,10.99,18.23,4.0]|29.22|[0.9443902735590791,0.06594298394369462,1.340368896
3459517,3.018989725303644] |
|[1.0,11.87,7.39,3.0] |25.0 |[0.9443902735590791,0.07122322287640173,0.543353052
3311345,2.264242293977733] |
|[1.0,12.5,11.85,5.0] |20.0 |[0.9443902735590791,0.07500339393049889,0.871276545
348301,3.773737156629552]  |
|[1.0,13.9,10.96,5.0] |24.86|[0.9443902735590791,0.08340377405071475,0.805838897
6385975,3.773737156629552]|
|[1.0,14.5,10.96,3.0] |25.46|[0.9443902735590791,0.0870039369593787,0.8058388976
385975,2.264242293977733]  |
|[1.0,14.6,7.78,5.0]  |22.38|[0.9443902735590791,0.08760396411082269,0.572027976
6084204,3.773737156629552]|
|[1.0,15.0,15.1,5.0]  |30.1 |[0.9443902735590791,0.09000407271659866,1.110234247
6590166,3.773737156629552]|
|[1.0,15.9,7.78,5.0]  |23.68|[0.9443902735590791,0.09540431707959458,0.572027976
6084204,3.773737156629552]|
|[1.0,16.5,12.48,5.0] |28.98|[0.9443902735590791,0.09900447998825852,0.917597576
8731475,3.773737156629552]|
+--------------------+-----+---------------------------------------------------
---------------------------+
only showing top 10 rows
```

In [36]:

```python
print(scalerModel.mean, '\n')
print(scalerModel.std)
```

```
[2.3238598810310647,119.5182088565764,19.679484467944473,4.105089226701915]

[1.058884264268571,166.65912494017266,13.600733387426205,1.3249465430352492]
```

In [37]:

```python
lr_train_df = train_df_s.drop("features")
lr_test_df = test_df_s.drop("features")
```

Modeling

Ridge regression

In [38]:

```python
rmse_list = []
```

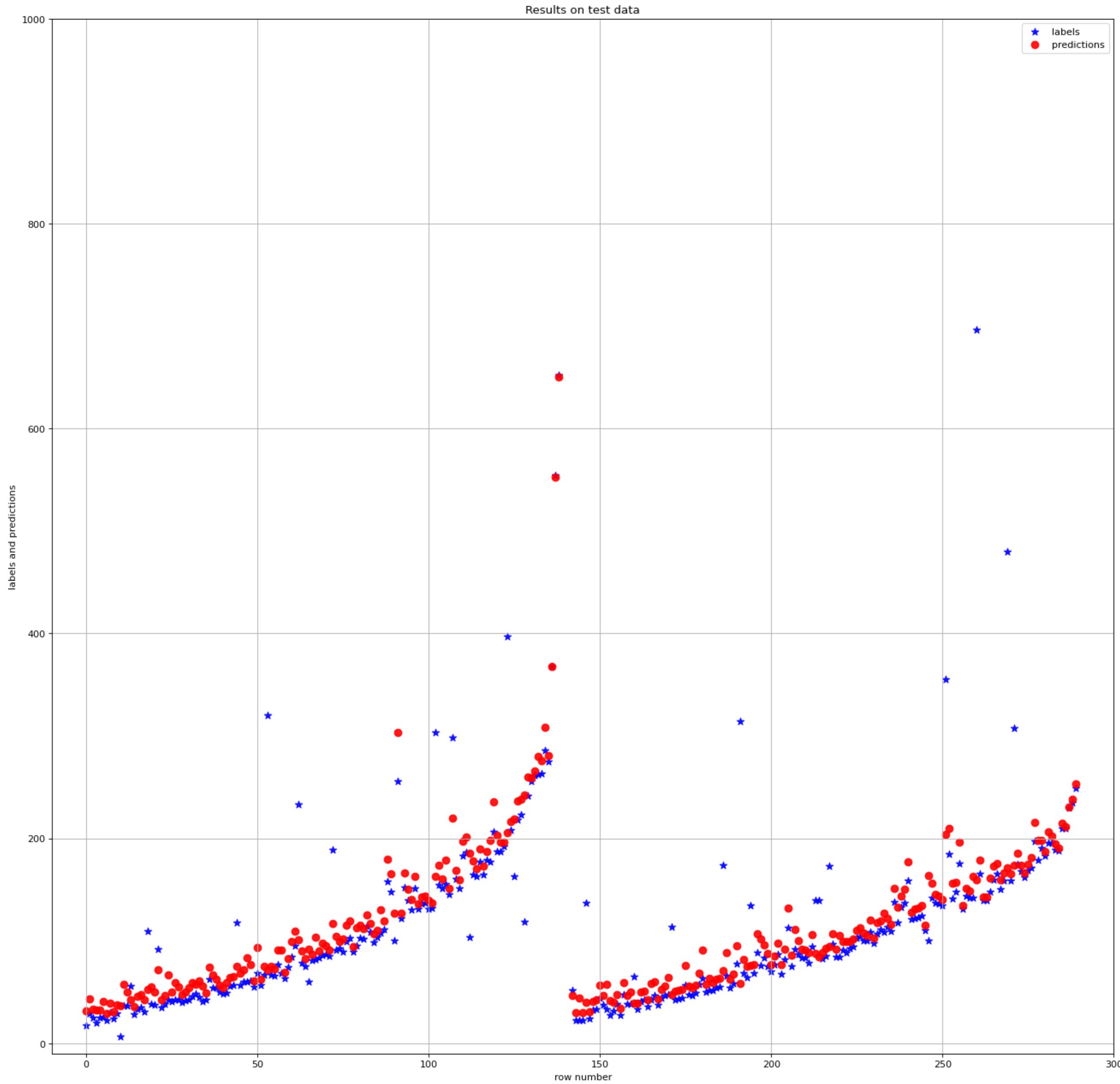
In [39]:

```python
lr_1 = LinearRegression(featuresCol = 'features_scaled', labelCol = 'label', max
lr_model_1 = lr_1.fit(lr_train_df)
lr_predictions = lr_model_1.transform(lr_test_df)
lr_predictions.select("prediction", "label", "features_scaled")
lr_evaluator = RegressionEvaluator(predictionCol = "prediction", labelCol = "lab
test_result = lr_model_1.evaluate(lr_test_df)
rmse = test_result.rootMeanSquaredError

print("RMSE: %f" % rmse)

plot_function(lr_predictions)
```

```
rmse_list.append(rmse)
```

Out[39]: DataFrame[prediction: double, label: double, features_scaled: vector]

RMSE: 51.941823



Lasso regression

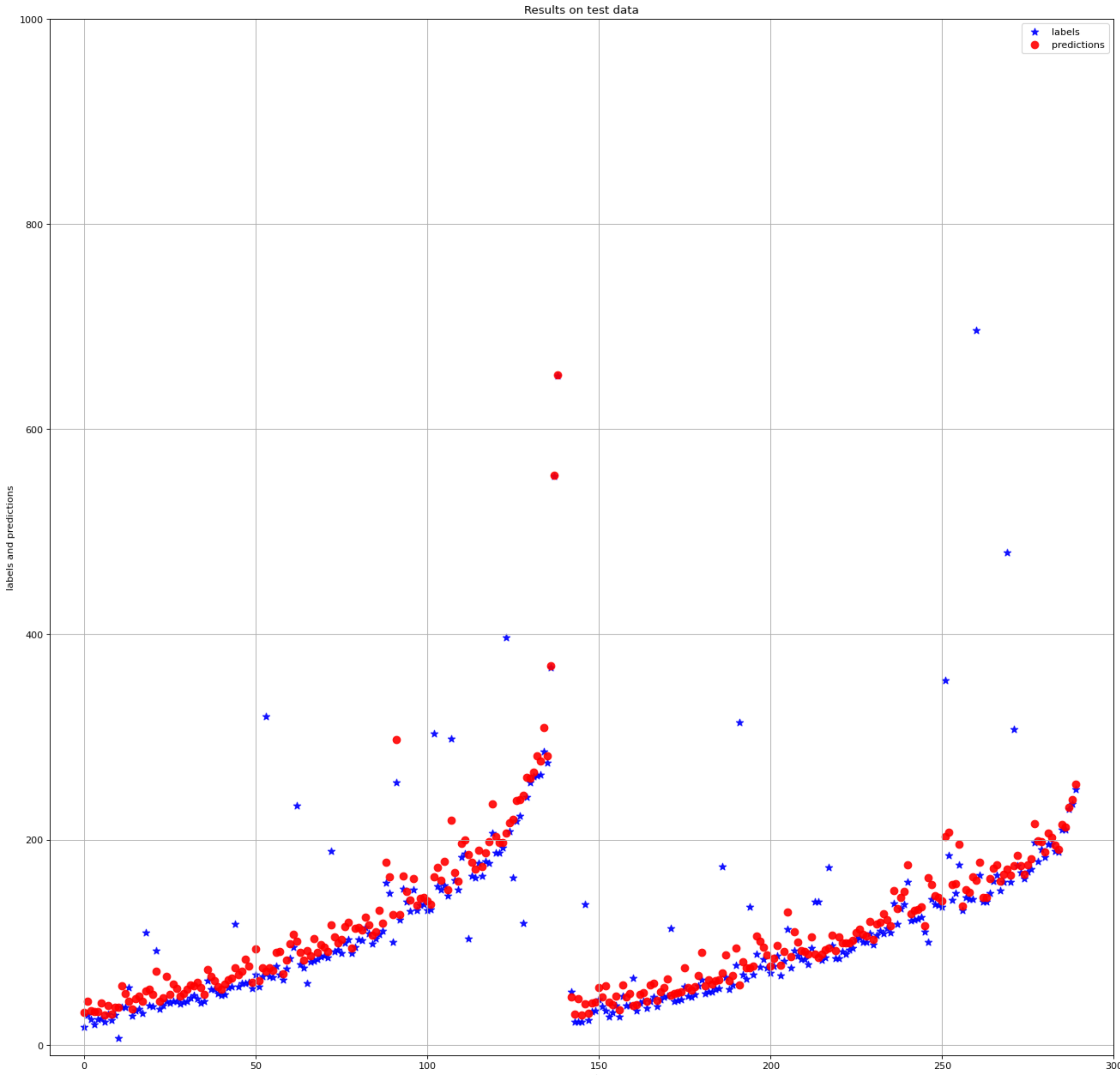
In [40]:

```
lr_train_df = train_df_s.drop("features")
lr_test_df = test_df_s.drop("features")

lr_2 = LinearRegression(featuresCol = 'features_scaled', labelCol = 'label', max
lr_model_2 = lr_2.fit(lr_train_df)
lr_predictions = lr_model_2.transform(lr_test_df)
lr_predictions.select("prediction", "label", "features_scaled")
lr_evaluator = RegressionEvaluator(predictionCol = "prediction", labelCol = "lab
test_result = lr_model_2.evaluate(lr_test_df)
rmse = test_result.rootMeanSquaredError

print("RMSE: %f" % rmse)
```

```
plot_function(lr_predictions)

rmse_list.append(rmse)
```

Out[40]: DataFrame[prediction: double, label: double, features_scaled: vector]

RMSE: 51.895901



Hyperparameter tuning

In [41]:

```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

lr_3 = LinearRegression(featuresCol = 'features_scaled', labelCol = 'label')

# Create ParamGrid for Cross Validation
lrparamGrid = (ParamGridBuilder()
             .addGrid(lr_3.regParam, [0.1, 0.5, 1.0, 2.0])
             .addGrid(lr_3.elasticNetParam, [0.0, 0.5, 0.7, 1.0])
             .addGrid(lr_3.maxIter, [1, 5, 10, 15])
             .build())
```

```
lrevaluator = RegressionEvaluator(predictionCol = "prediction", labelCol = "labe

# Create 5-fold CrossValidator
lrcv = CrossValidator(estimator = lr_3, estimatorParamMaps = lrparamGrid, evalua

# Run cross validations
lrcvModel = lrcv.fit(lr_test_df)
best_model = lrcvModel.bestModel
```

In [42]:

```
best_reg_param = best_model._java_obj.getRegParam()
best_elasticnet_param = best_model._java_obj.getElasticNetParam()
best_max_Iter = best_model._java_obj.getMaxIter()
print('best regParam:', best_reg_param, '\nbest elasticNetParam:', best_elasticn
```

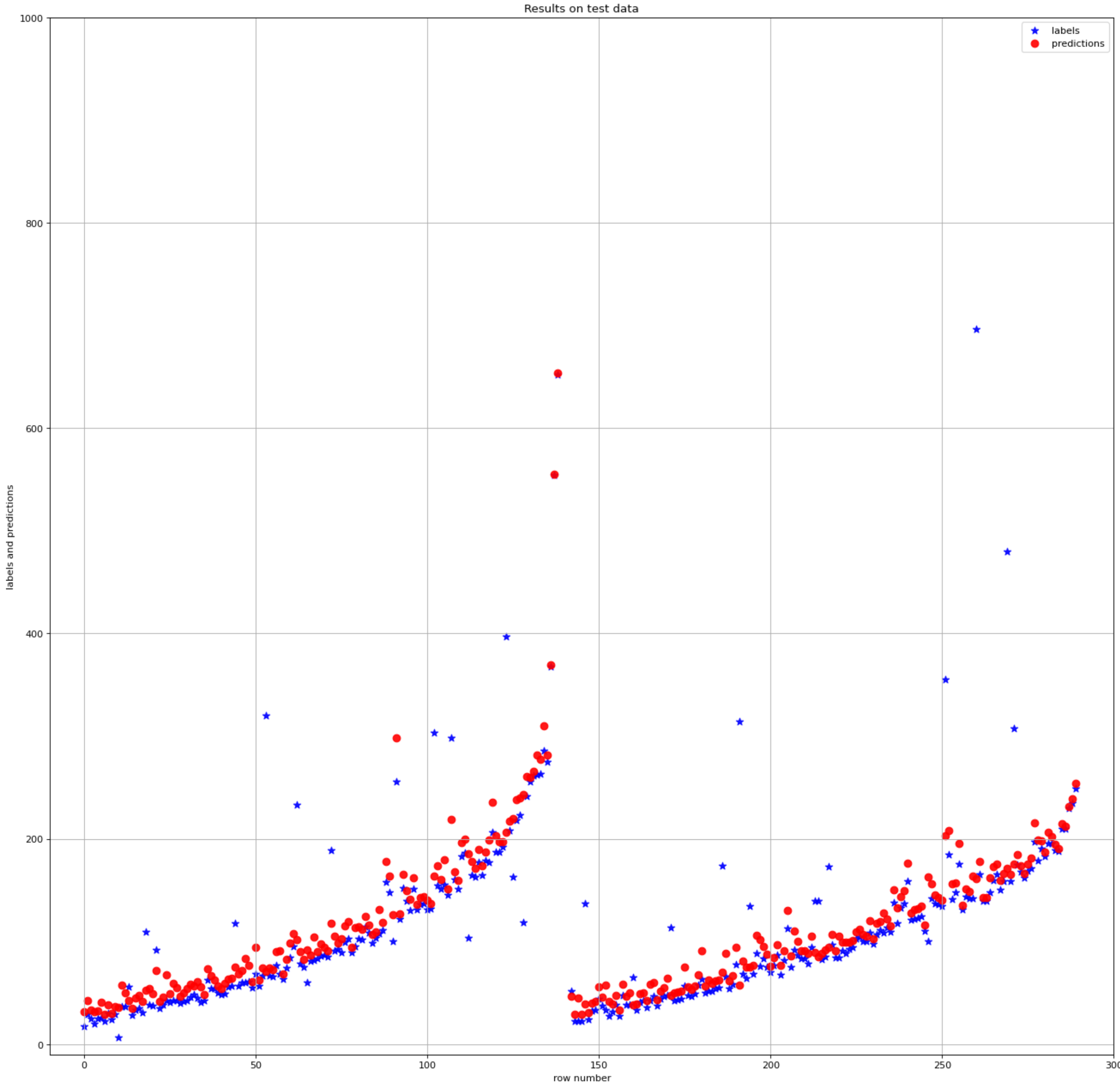
```
best regParam: 0.1
best elasticNetParam: 1.0
best max iter: 5
```

In [43]:

```
lr_3 = LinearRegression(featuresCol = 'features_scaled', labelCol = 'label', max
lr_model_3 = lr_3.fit(lr_train_df)
lr_predictions = lr_model_3.transform(lr_test_df)
lr_predictions.select("prediction", "label", "features_scaled")
lr_evaluator = RegressionEvaluator(predictionCol = "prediction", labelCol = "lab
test_result = lr_model_3.evaluate(lr_test_df)
rmse = test_result.rootMeanSquaredError

print("RMSE: %f" % rmse)

plot_function(lr_predictions)

rmse_list.append(rmse)
```

Out[43]:

```
DataFrame[prediction: double, label: double, features_scaled: vector]

RMSE: 51.879311
```



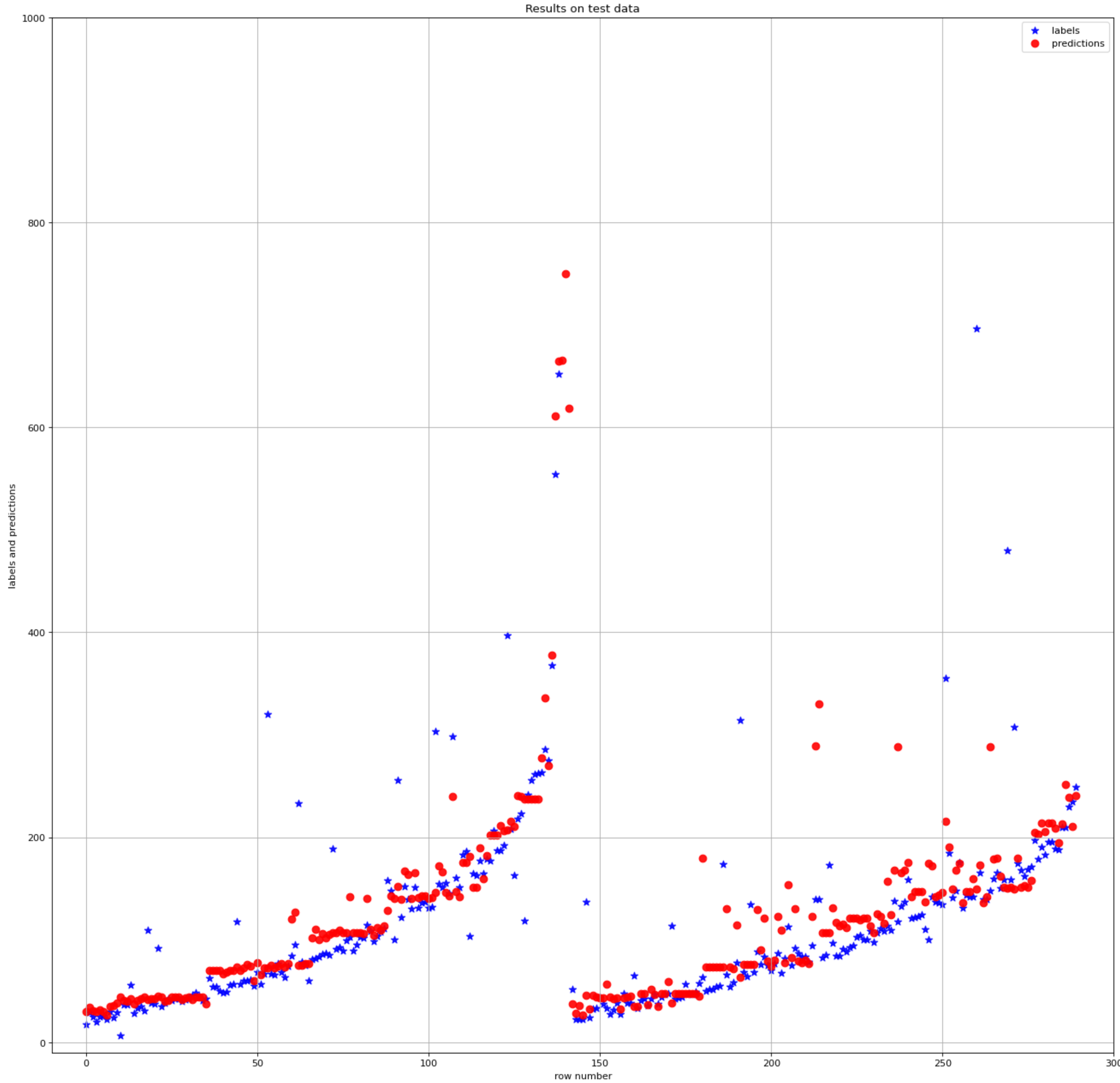
Gradient-boosted tree regression

In [44]:

```
# Identify categorical features, and index them.
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures",

# Train a GBT model.
gbt = GBTRegressor(featuresCol="indexedFeatures", maxIter=10)

rmse = fit_predict_plot_function(featureIndexer, gbt, train_df, test_df)

rmse_list.append(rmse)
```

```
+------------------+-----+--------------------+
|        prediction|label|            features|
+------------------+-----+--------------------+
| 29.68978303982267|17.28|  [1.0,9.5,7.78,3.0]|
| 34.26632314541024|29.22|[1.0,10.99,18.23,...|
| 31.15213527015517| 25.0|[1.0,11.87,7.39,3.0]|
| 30.36432902162655| 20.0|[1.0,12.5,11.85,5.0]|
|31.397649145209886|24.86|[1.0,13.9,10.96,5.0]|
```

```
+------------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 140.471
GBTRegressionModel: uid=GBTRegressor_dbf1bb3e4591, numTrees=10, numFeatures=4
```



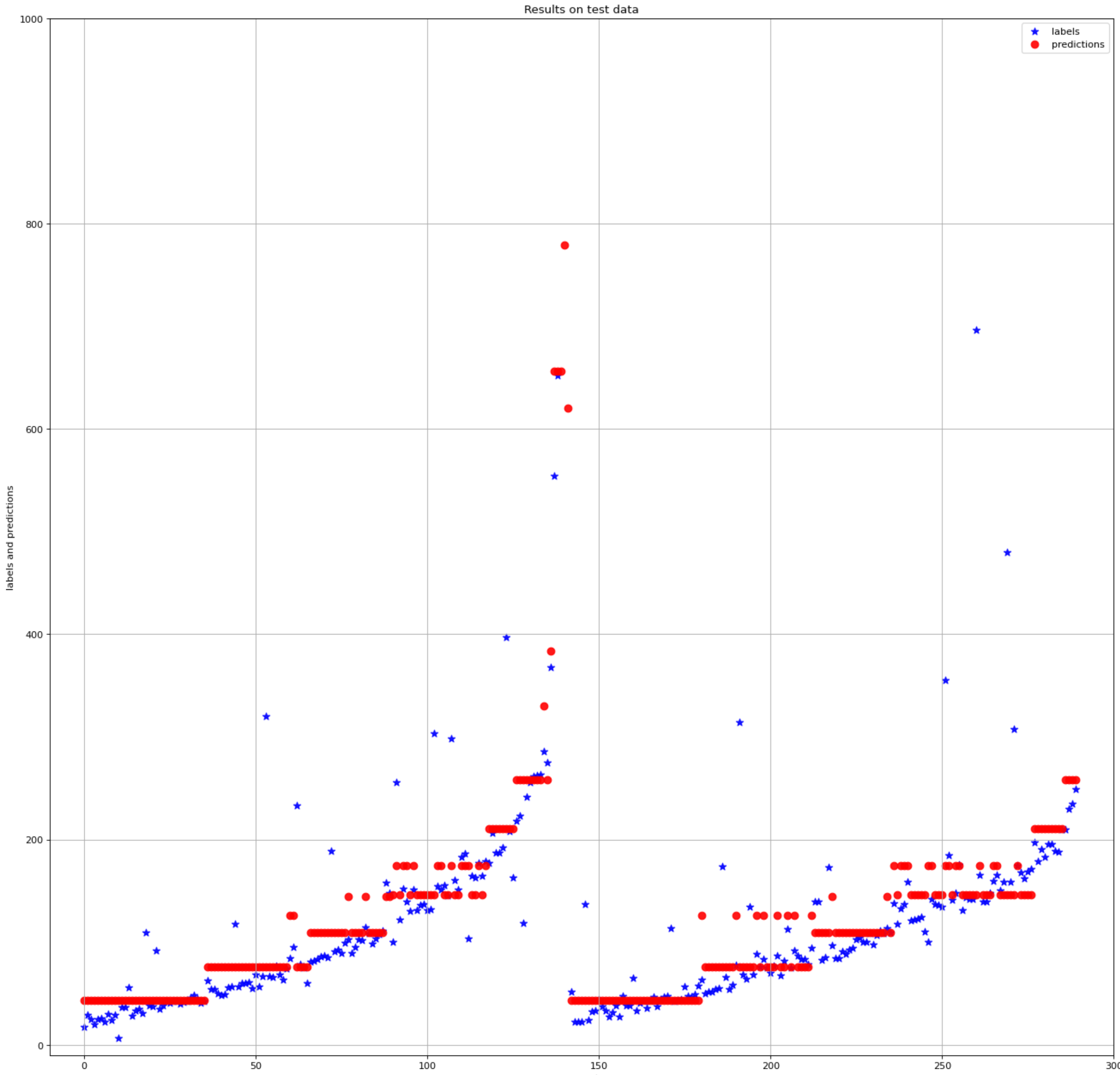
Decision tree regression

In [45]:

```
# Identify categorical features, and index them.
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures",

# Train a DecisionTree model.
dt = DecisionTreeRegressor(featuresCol="indexedFeatures")

rmse = fit_predict_plot_function(featureIndexer,dt , train_df, test_df)

rmse_list.append(rmse)
```

```
+----------------+-----+--------------------+
|      prediction|label|            features|
+----------------+-----+--------------------+
```

```
|42.9671794871795|17.28|  [1.0,9.5,7.78,3.0]|
|42.9671794871795|29.22|[1.0,10.99,18.23,...|
|42.9671794871795| 25.0|[1.0,11.87,7.39,3.0]|
|42.9671794871795| 20.0|[1.0,12.5,11.85,5.0]|
|42.9671794871795|24.86|[1.0,13.9,10.96,5.0]|
+----------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 140.684
DecisionTreeRegressionModel: uid=DecisionTreeRegressor_dd042cef3ef2, depth=5, nu
mNodes=61, numFeatures=4
```



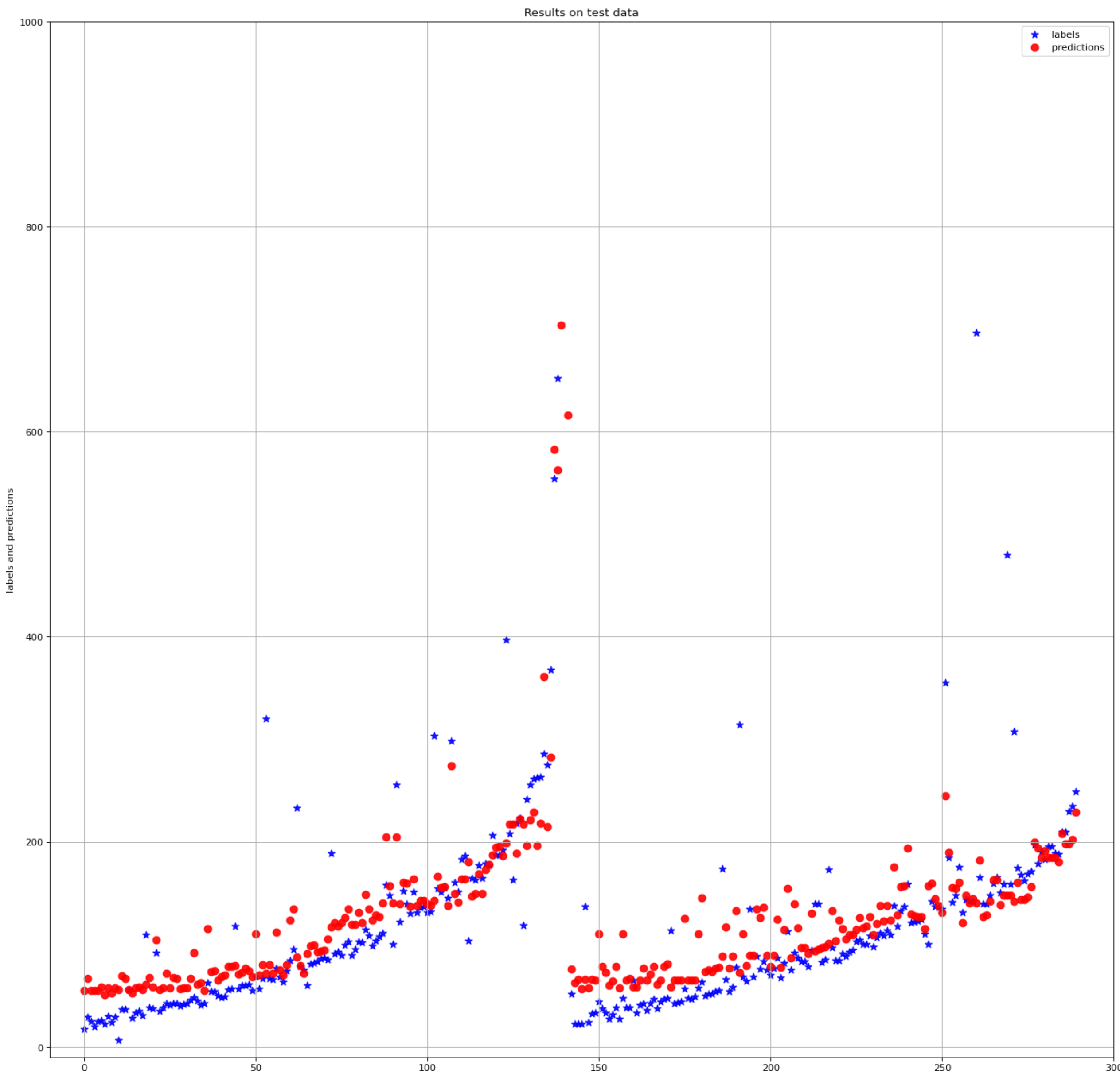
Random forest regression

In [46]:

```
# Identify categorical features, and index them.
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures",

# Train a DecisionTree model.
rf = RandomForestRegressor(featuresCol="indexedFeatures")

rmse = fit_predict_plot_function(featureIndexer, rf, train_df, test_df)
```

```
rmse_list.append(rmse)
```

```
+-----------------+-----+--------------------+
|       prediction|label|            features|
+-----------------+-----+--------------------+
| 54.6541450871861|17.28|  [1.0,9.5,7.78,3.0]|
|66.86202518590429|29.22|[1.0,10.99,18.23,...|
| 54.6541450871861| 25.0|[1.0,11.87,7.39,3.0]|
| 54.8361410220902| 20.0|[1.0,12.5,11.85,5.0]|
| 54.8361410220902|24.86|[1.0,13.9,10.96,5.0]|
+-----------------+-----+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 135.09
RandomForestRegressionModel: uid=RandomForestRegressor_4e748c4fb93e, numTrees=2
0, numFeatures=4
```



Results

In [47]:

```
# Show the RMSE results
```

```
rmse_list
```

Out[47]:
```
[51.94182282480001,
 51.895900687515734,
 51.87931101256097,
 140.47056799019543,
 140.68440960751286,
 135.08950056945932]
```

In [48]:
```
figure(figsize = (18, 10), dpi = 80)

plt.title('Compare results')
plt.ylabel('RMSE')
x = ['Ridge regression', 'Lasso regression', 'Hyperparameter Tuning']
y = rmse_list[:3]

plt.plot(x, y, color = 'blue', marker = '*', label = 'RMSE', alpha = 0.9)

y_max = np.max(y)+0.1
y_min = np.min(y)-0.1

plt.xlim(-0.5, 2.5)
plt.ylim(y_min, y_max)

plt.legend()
plt.grid()
plt.show()
```

Out[48]:
```
<Figure size 1440x800 with 0 Axes>
```

Out[48]:
```
Text(0.5, 1.0, 'Compare results')
```

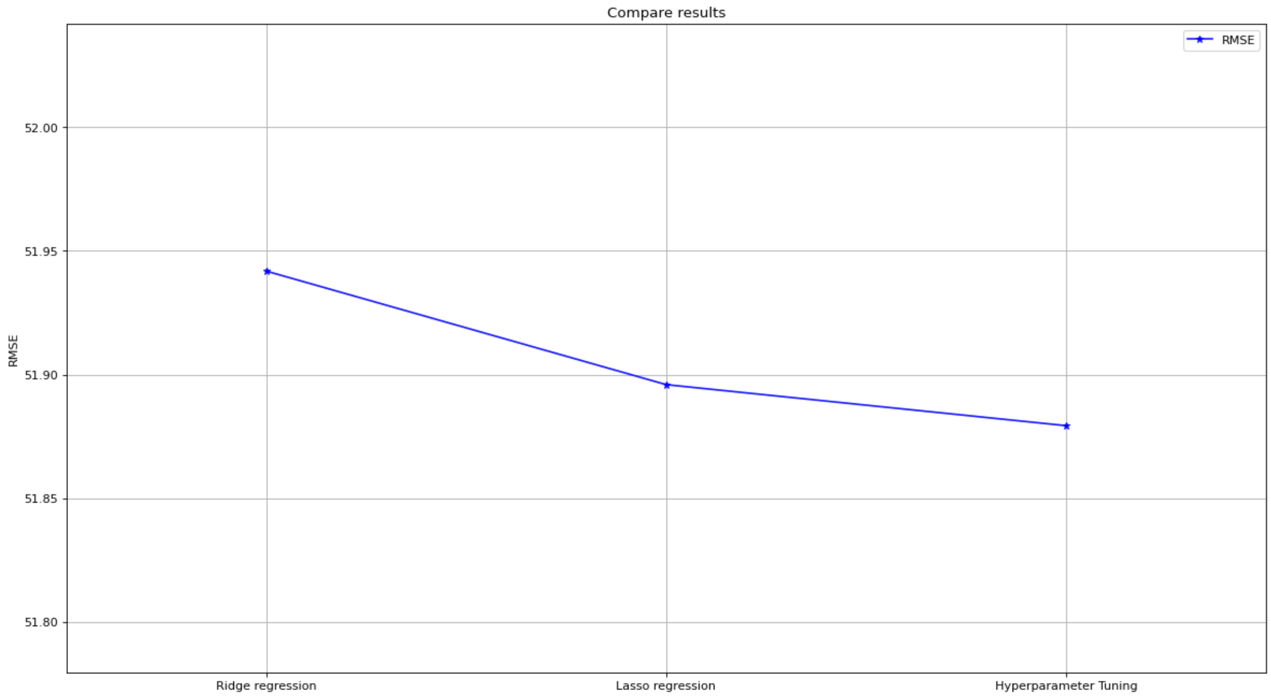
Out[48]:
```
Text(0, 0.5, 'RMSE')
```

Out[48]:
```
[<matplotlib.lines.Line2D at 0x7f550c81c550>]
```

Out[48]:
```
(-0.5, 2.5)
```

Out[48]:
```
(51.77931101256097, 52.041822824800015)
```

Out[48]:
```
<matplotlib.legend.Legend at 0x7f554d687220>
```



In [49]:

```
figure(figsize = (18, 10), dpi = 80)

plt.title('Compare results')
plt.ylabel('RMSE')
x = ['Gradient-boosted tree regression', 'Decision tree regression', 'Random for
y = rmse_list[3:]

plt.plot(x, y, color = 'blue', marker = '*', label = 'RMSE', alpha = 0.9)

y_max = np.max(y)+0.1
y_min = np.min(y)-0.1

plt.xlim(-0.5, 2.5)
plt.ylim(y_min, y_max)

plt.legend()
plt.grid()
plt.show()
```

Out[49]: <Figure size 1440x800 with 0 Axes>

Out[49]: Text(0.5, 1.0, 'Compare results')

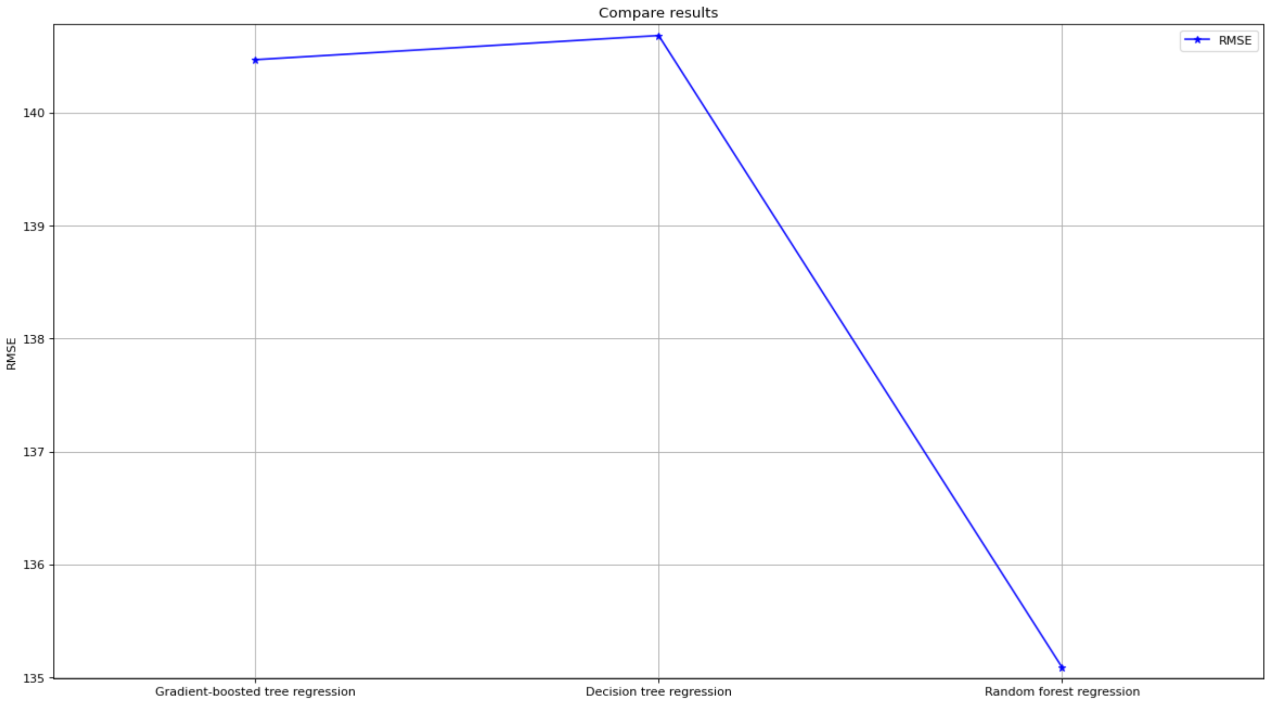
Out[49]: Text(0, 0.5, 'RMSE')

Out[49]: [<matplotlib.lines.Line2D at 0x7f550c7f2b80>]

Out[49]: (-0.5, 2.5)

Out[49]: (134.98950056945932, 140.78440960751286)

Out[49]: <matplotlib.legend.Legend at 0x7f550ca092b0>



Conclusion

As we know, RMSE is a vary good measure of how accurately different models predict the future, and it is the most important criterion for fit if the main purpose of the model is prediction. Since lower values of RMSE indicate better fit, we can see from the above models that the results for Linear regression after hyperparameter tuning is much better than Gradient-boosted tree regression, Decision tree regression, and Random forest regression. Hence, for sales forecast of e-commerce data, linear regression is preferred.

Stop the spark session

In [50]:

```
spark.stop()
```

In [ ]:

```

```