Task 1 - Data Exploration:

1. 5 most popular products sold by the e-commerce company in the month

```
+----------+-----+
|product_id|count|
+----------+-----+
|   1004856|28944|
|   1004767|21806|
|   1004833|12697|
|   1005115|12543|
|   4804056|12381|
+----------+-----+
```

   Above are the 5 most popular products sold by the company in the month. Here we filter if the event is “purchase” and group the data based on product ID and then sort the count by descending.

2. 5 most popular Brands on the platform

```
+-------+-------+
|  brand|  count|
+-------+-------+
|   null|6113008|
|samsung|5282775|
|  apple|4122554|
| xiaomi|3083763|
| huawei|1111205|
|lucente| 655861|
+-------+-------+
only showing top 6 rows
```

   Above are the 5 most popular brands. Here since all event types count, each row denotes an action by a user. Hence we group them by brand.

   In the above result we see there are a lot of records without any brand.

3. 5 most popular product categories

```
+----------------------------+--------+
|category_code               |count   |
+----------------------------+--------+
|null                        |13515609|
|electronics.smartphone      |11507231|
|electronics.clocks          |1311033 |
|computers.notebook          |1137623 |
|electronics.video.tv        |1113750 |
|electronics.audio.headphone|1100188 |
+----------------------------+--------+
only showing top 6 rows
```

As expected smartphones are the most popular product categories, followed by clocks and notebooks (laptops).

4. Number of unique users on the platform
   There are 30,22,290 (30 Lakh 22 thousand) unique users in the platform

   Here we fetch the distinct user ID count.

5. Most active user on the platform

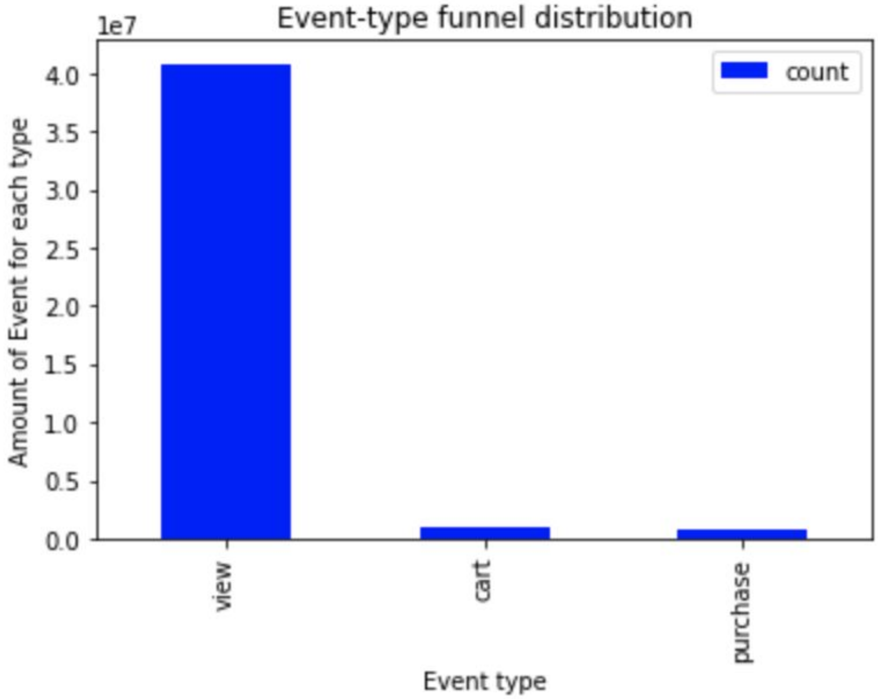
```
+---------+-----+
|  user_id|count|
+---------+-----+
|512475445| 7436|
+---------+-----+
only showing top 1 row
```

   The most active user is the one who does the most number of activities like different events. Hence each row is an activity and hence grouping based on user_id and sorting gives us the most active user_id. From this we can get user details.

6. Average and Maximum price for smartphones purchased by the customers
   The average price of smartphones purchased is 464.62 and maximum is 2110.45.

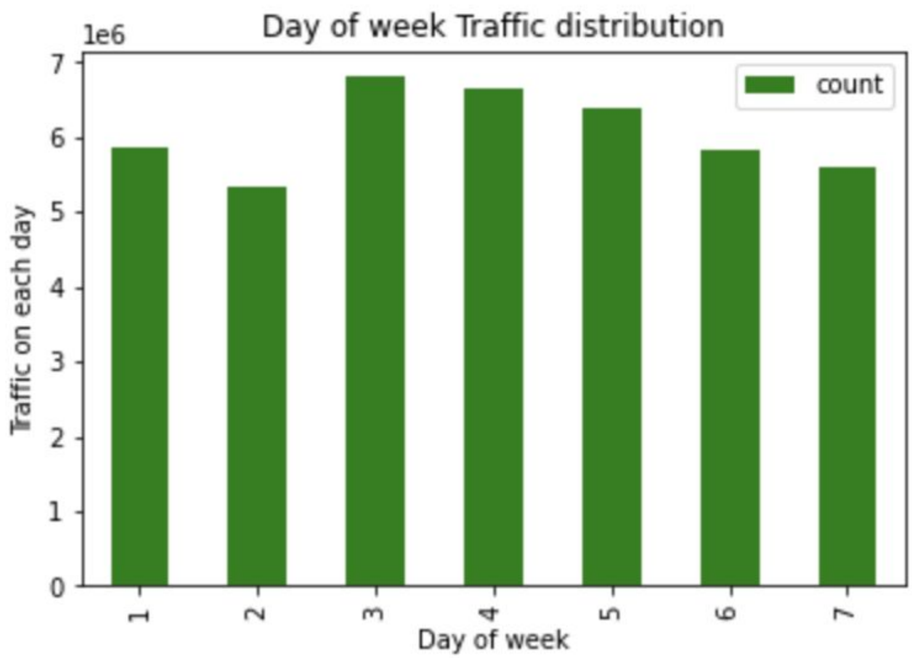
   Here we have to filter based on 2 conditions, one to check if the category is smartphone and another to check if the event was purchase.

7. Event-type funnel distribution in e-commerce shopping journey



As we can see above the distribution of event type funnel is not the same and is very less for cart and purchase. This means a lot of people view the product but do not add to cart and purchase them.

8. Traffic on different days of the week



We can see that the traffic is high during the mid of the week and low during the starting and ending of the week.

Task 2 - Feature Engineering:

- Handle missing values (provide justification for approach)

  We have not removed any missing values as the proportion is large. We still need those rows to track other activities of user, let us impute the null rows with 'NA' value since the columns which has Null values are string

  Duplicates are also dropped.

  Generate the category code at 2 levels (Split into 2 columns)
- Example: electronics.video.tv - electronics, video
  The escape character is used to split '.'. The Null values are inserted with 'NA' for new category columns too.

```
+--------------------+-----------+-----------+
|       category_code|     cat_l1|     cat_l2|
+--------------------+-----------+-----------+
|       apparel.shoes|    apparel|      shoes|
|                  NA|         NA|         NA|
|electronics.smart...|electronics| smartphone|
|                  NA|         NA|         NA|
|                  NA|         NA|         NA|
|appliances.person...| appliances|   personal|
|       apparel.shoes|    apparel|      shoes|
|appliances.kitche...| appliances|    kitchen|
|  electronics.clocks|electronics|     clocks|
|electronics.smart...|electronics| smartphone|
|computers.periphe...|  computers|peripherals|
|  accessories.wallet|accessories|     wallet|
|       kids.carriage|       kids|   carriage|
|electronics.smart...|electronics| smartphone|
|auto.accessories....|       auto|accessories|
|  apparel.shoes.keds|    apparel|      shoes|
|                  NA|         NA|         NA|
|                  NA|         NA|         NA|
|                  NA|         NA|         NA|
|                  NA|         NA|         NA|
+--------------------+-----------+-----------+
only showing top 20 rows
```

- Total activities (view/cart/etc.) in the session
  Here we use the window function for each session ID of the user and then take the count of the event_type row which gives us the total activity in that session. Since each session ID is unique this will work.
  We need to drop the duplicates at the end to get a clear output as show below.

user_id	user_session	activity_count
564555327	00019495-9f33-48fa-ae79-c94c951aba40	1
543073137	0002854a-13ef-490a-a838-b3be082eedd4	17
513193974	0002a642-8f4a-48cb-89b6-260cc37073e8	1
513257116	0002af2d-cbff-4557-b6fd-ee00a86de4c1	1
553177004	00084f25-74e3-4df7-be8c-5504588f1f45	1
519287061	00088378-6d3f-40cf-ad96-6e3a5aa24d6b	1
554129220	000a2754-1167-47ce-88c7-92fa7eae9d6d	7
513649894	000a378f-37b9-4f8c-b315-679003e41053	9
547170917	000b2a8a-194b-40be-b4ba-e3f01b6a895c	6
393914239	000d8e70-ed4a-4d51-b7a9-035168d79efd	1
555374095	000d99dc-b6e9-4305-938e-3a4bc45a2052	5
555623672	000e9abc-3a8b-4bc8-9139-2cf954973750	6
523101797	000eef2b-65e5-4eaa-8768-f1689e9e5c34	1
551354237	000f63ed-a15f-4c08-a5ca-ca3cc0e0d10f	1
561407991	0010ac52-cc78-4062-aa55-4ba087a0d1d2	3
512437916	0010c8e5-44fd-40b4-ab8d-135bfda49879	2
540528317	0010f624-b475-4253-bc94-b0d09a9e67be	2
519250415	0012b8f1-5965-45df-b8a7-98541b8aeeaf	1
520061567	00131c36-c6b4-4ebc-95b2-59101cafe1e4	2
559713985	00135fcf-bd50-46af-88fa-5e2293488b5c	3
556059985	00137b31-7cbe-400a-bdce-f1c82ed976a6	1
548323545	0016ef79-082c-4d7c-ba13-3cdd7b7af123	1
530830066	00170aeb-bee0-44bc-a756-3eddab08ac26	3
565066597	001aec60-3725-44c0-9635-337cfb8c9a1a	11
561438023	001c083e-1790-498f-953b-39120eea51fb	5
540149843	001d6bea-536e-44f0-a01d-6f8dc9f5a481	1
518675792	001df4c7-5ce3-481c-aa90-cf5e09cf3cde	1
513683876	001e637c-818d-4ccd-8c25-152f6c124de6	1
554232041	002086fa-cc20-43c2-aa32-9fd378b8fdb7	2
519131275	0022ecd6-9306-4e80-ab33-ed2b6147ad50	2

only showing top 30 rows

- Affinity towards a particular product (Product count for user)
  It is very similar to above except we filter only view event types. Here we can sort the view_count column descending to identify the products for which the user has high affinity. I have not done that here so that you can see different view count values

user_id	product_id	view_count
240522111	5100565	27
240522111	4804056	27
240522111	5100566	27
240522111	5100854	27
240522111	4804055	27
240522111	4802036	27
240522111	18500054	27
240522111	5100567	27
240522111	4803976	27
277067319	3701151	5
277067319	3700412	5
277067319	17200971	5
277067319	3701429	5
277067319	3701101	5
303418896	13900421	5
303418896	36600080	5
303418896	36600091	5
303418896	36600096	5
303418896	36600028	5
356463487	1003315	3
356463487	1004238	3
357446328	2600940	18
357446328	2600574	18
357446328	2600652	18
357446328	2601430	18
357446328	2601340	18
357446328	2600446	18
357446328	2602135	18
357446328	2601122	18
357446328	2601106	18

- Affinity towards a category (Secondary category count for user)
  This is also very similar to above but we also need to filter Null categories (NA)

user_id	cat_l2	view_count
240522111	clocks	27
240522111	audio	27
240522111	tablet	27
277067319	environment	5
277067319	living_room	5
303418896	components	1
356463487	smartphone	3
366968564	clocks	6
366968564	carriage	6
389518481	environment	4
391260478	fmcg	1
406827257	smartphone	1
410271694	smartphone	2

  We can get the categories to which every user has high affinity by sorting based on view_count

- Average shopping expense for a product category (secondary)

cat_l2	avg_price
clocks	267.6873304091836
desktop	346.98866493842996
peripherals	149.08874292643486
shoes	79.2413671817497
shirt	51.64783783783781
smartphone	464.1542272617024
jumper	30.63
bedroom	154.22857561793043
fmcg	13.71985155195684
trainer	302.36889830508477
wallet	48.96971530249106
shorts	16.09
cartrige	15.267538461538452
ski	244.89875
umbrella	25.480000000000008
tablet	277.41042915136035
lawn_mower	148.05837209302322
skates	286.8628310502271
kitchen	212.4026546304877
dress	58.087333333333326
jeans	44.02307142857143
iron	61.28573277074591
belt	58.209523809523816
living_room	299.32971859588224
audio	115.86345935268335
bathroom	83.90974358974363
ebooks	162.4964184397166
underwear	21.8390322580645
accessories	119.2177050645484
cultivator	327.8992592592592

only showing top 30 rows

Here we partition by category level 2 then calculate the average of price column

- Number of user sessions
  Here we create a partition for each user_id and count the user_session column.

```
+---------+-------------+
|user_id  |session_count|
+---------+-------------+
|240522111|27           |
|277067319|5            |
|303418896|5            |
|356463487|3            |
|357446328|18           |
|366968564|6            |
|369454898|1            |
|370633539|1            |
|389518481|5            |
|391260478|7            |
|406827257|1            |
|410271694|2            |
|420155938|1            |
|420412138|3            |
|424456347|1            |
|425909320|2            |
```

- Impact of time: Day and Hour (Binning hours into 4 buckets)

```
+----+-----------+
|hour|hour_bucket|
+----+-----------+
|   0|        0.0|
|   0|        0.0|
|   0|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
|   2|        0.0|
```

We use Bucketizer class of pyspark ML for this and split into midnight and early morning (0-8), morning (8-12), noon (12-16), evening and night (16-23)

- Reduction in brands for analysis: Top 20 + ‘others’

  First finding the top 20 brands

```
+--------+-------+
|brand   |count  |
+--------+-------+
|NA      |4542823|
|samsung |3580281|
|apple   |3058048|
|xiaomi  |1940090|
|huawei  |736368 |
|lucente |498799 |
|bosch   |364026 |
|lg      |360396 |
|oppo    |327184 |
|sony    |307580 |
|acer    |275298 |
|cordiant|255957 |
|respect |225769 |
|lenovo  |220647 |
|artel   |206830 |
|hp      |199882 |
|indesit |193595 |
|casio   |187350 |
|dauscher|186141 |
|philips |183704 |
|stels   |182506 |
+--------+-------+
```

We can leave the first one as it is Null value. Then for rest we do the mapping and hence it gets transformed as below.

```
+---------+---------+
|    brand|brand_red|
+---------+---------+
|   fassen|   others|
|  redmond|   others|
|  samsung|  samsung|
|      bts|   others|
|  matador|   others|
|  philips|  philips|
|       NA|   others|
|     beko|   others|
|   orient|   others|
|    apple|    apple|
|       lg|       lg|
|    karya|   others|
|wingoffly|   others|
|    apple|    apple|
|   alpine|   others|
|    escan|   others|
| nestogen|   others|
|    lider|   others|
|  navitel|   others|
|legeartis|   others|
+---------+---------+
only showing top 20 rows
```

- Target variable generation: is_purchased

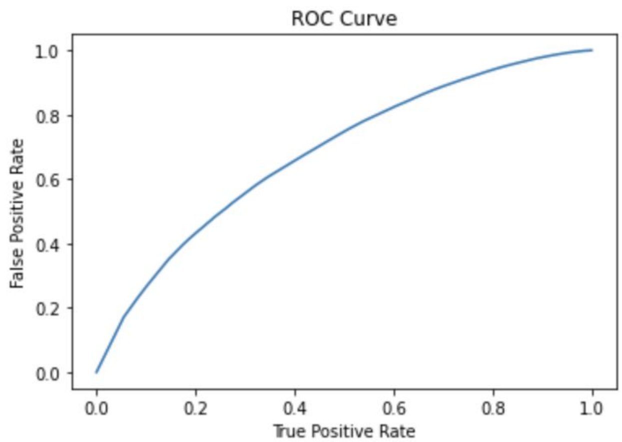
  Here if event_type is ‘purchase’ we set is_purchased to 1 otherwise 0.

Task 3 - Model Selection:
Logistic Regression:

First, we build a model with the default threshold which is 0.5. But we get a recall as 0 for the positive class where the customer bought the product.

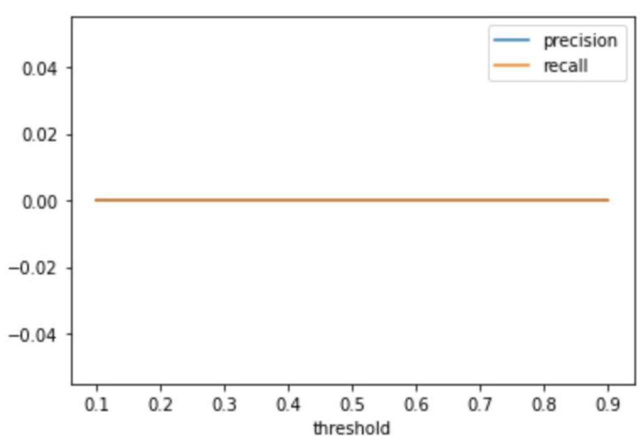
We need to get good recall in order to be able to predict most of the customer purchases.

Area under ROC curve:




```
TestSet areaUnderROC: 0.6815226653139731
```

Since it is very poor, we test with different thresholds and find out the point where precision and recall coincides.



Even with different thresholds, recall does not improve for the positive class and hence we can conclude logistic regression is not able to perform well on this model. Might be the data we have is not linear data.

Decision Trees:

With below paramGrid tried finding out the best model with 5 folds in cross validation.

```
dtparamGrid = (ParamGridBuilder()
            .addGrid(dt.maxDepth, [2, 5, 10, 20, 30])
            .addGrid(dt.maxBins, [10, 20, 40, 80])
            .addGrid(dt.impurity, ['gini','entropy'])
            .build())
```

However since it took more than 3 hours (and still running), let us try to reduce the combinations of values as below and folds to 3.

```
dtparamGrid = (ParamGridBuilder()
            .addGrid(dt.maxDepth, [2, 10, 30])
            .addGrid(dt.maxBins, [10, 30, 50])
            .addGrid(dt.impurity, ['gini','entropy'])
            .build())
```

After running for 6 hours with above hyperparameter tuning we got the best decision tree with 30 as depth. However the area under ROC 0.5565098111611424 is still lower than what logistic regression gave which is 0.68.

The area under PR curve is also low 0.03185758626995479

We can infer that from this the depth of the tree needs to be increased further and also the folds so that the model can learn from different subsets. Only if we increase we will get a better area under the ROC curve.

I am not increasing the depth and folds further since it is taking longer time. However we can conclude that decision trees will perform better on such huge data provided we train it enough on different hyper parameters and folds.

Random Forests:

Let us try random forests with similar sets of Depths as above.

When running a random forest with below param grid, it ran for almost 15 hours and then the EC2 server crashed.

```
rfparamGrid = (ParamGridBuilder()
            .addGrid(rf.maxDepth, [2, 10, 30])
            .addGrid(rf.numTrees, [10, 30, 50])
            .addGrid(rf.impurity, ['gini','entropy'])
            .build())
```

In order to run these kinds of hyper parameter tuning with random forests a good instance is needed and also it will take 2-3 days. Hence let us try a single random forest with 30 decision trees and see how it performs on this data.

We achieved a decent accuracy, f1score and recall with a random forest of 30 decision trees. However this can be further tuned based on the business requirement.

```
# Check accuracy
evaluator.evaluate(rfpredictions, {evaluator.metricName: "accuracy"})
```

```
0.9758568121453868
```

As we can see we get a pretty decent accuracy with random forests

```
# Check F1 score
evaluator.evaluate(rfpredictions, {evaluator.metricName: "f1"})
```

```
0.963932722206274
```

```
# Check weighted recall
evaluator.evaluate(rfpredictions, {evaluator.metricName: "weightedRecall"})
```

```
0.9758568121453868
```

Task 4 - Model Inference:

Below is the feature importance detail given by random forest.

```
+---+--------------------+-----+
|idx|                name|score|
+---+--------------------+-----+
|  0|               price|  0.0|
| 64|       cat_l2_en_ski|  0.0|
| 74| brand_red_en_xiaomi|  0.0|
| 73|  brand_red_en_apple|  0.0|
| 72|brand_red_en_samsung|  0.0|
| 71| brand_red_en_others|  0.0|
| 70|    cat_l2_en_shorts|  0.0|
| 69|     cat_l2_en_skirt|  0.0|
| 68| cat_l2_en_furniture|  0.0|
| 67|      cat_l2_en_belt|  0.0|
| 66|      cat_l2_en_sock|  0.0|
| 65|     cat_l2_en_scarf|  0.0|
| 63|    cat_l2_en_tennis|  0.0|
| 51|      cat_l2_en_fmcg|  0.0|
| 62|    cat_l2_en_jumper|  0.0|
| 61| cat_l2_en_snowboard|  0.0|
| 60|  cat_l2_en_umbrella|  0.0|
| 59|cat_l2_en_lawn_mower|  0.0|
| 58|cat_l2_en_cultivator|  0.0|
| 57|  cat_l2_en_cartrige|  0.0|
+---+--------------------+-----+
only showing top 20 rows
```

We can infer the below things from the above table.

1. Price influences whether a customer shall buy a product or not to a great extent. So Price needs to be tweaked based on the average purchase price of the customer as we found in Task 2.
2. Then it is influenced whether a product belongs to a particular level2 category *ski.* This belongs to the sports category. Looks like more products in this category are bought.
3. Products in brands Xiaomi, Apple, Samsung are brought more than other products
4. Products in category shorts, skirts, furniture, belt,socks and scarf are also bought more

Hence we can conclude that random forests perform well on this large data and give pretty decent metrics than Logistic regression or decision trees.