Contribution of each member:

Kalaiselvan = 50%
Ankur Shukla = 50%

We are contributing equally for both Mid submission and Final submission. We have split up the tasks equally and worked in parallel. Then collaborated and merged everything.

Top five rows of the data set at the beginning of the analysis

Event_Data

```
In [3]: df_event.head(5)
```

Out[3]:

	Device_id	Gender	Age	Group	Event_id	Event_timestamp	Longitude	Latitude
0	-1000369272589010000	F	26	F25-32	\N	\N	\N	\N
1	-1000572055892390000	F	27	F25-32	\N	\N	\N	\N
2	-1000643208750510000	M	29	M25-32	\N	\N	\N	\N
3	-1001337759327040000	M	30	M25-32	2774404	2016-05-07 09:14:24	119.61	29.7
4	-1001337759327040000	M	30	M25-32	3065018	2016-05-04 10:26:14	120.29	30.42

Non_Event_Data

```
In [24]: df_non_event.head(5)
```

Out[24]:

	Device_id	Gender	Age	Group	device_model	phone_brand
0	-7548291590301750000	M	33	M32+	è□£è€€3C	Huawei
1	-1819925713085810000	F	23	F0-24	N1 Mini	OPPO
2	3670076507269740000	M	33	M32+	menote1 2	Meizu
3	5333872006968810000	M	34	M32+	xnote	Xiaomi
4	5263633571423510000	M	27	M25-32	hu1 Plus	Huawei

App_data

```
]: app_data_1 = "capstone-data/app-data/000001_0"
df_app_data_1 = pd.read_csv(app_data_1,encoding = "ISO-8859-1",quoting=3, error_bad_lines=False,names=["event_id", "app
                                                                                                        "label_id","category"])

df_app_data_1.head()
```

]:

	event_id	app_id	is_installed	is_active	label_id	category
0	3110119	-1002609068504196793	1	0	854	Property Industry new
1	3110119	-1002609068504196793	1	0	859	pursue
2	3110119	-1002609068504196793	1	0	548	Industry tag
3	3090353	-1002609068504196793	1	0	854	Property Industry new
4	3090353	-1002609068504196793	1	0	859	pursue

```
]: app_data_2 = "capstone-data/app-data/000002_0"
```

List of data cleaning techniques applied such as missing value treatment, etc.

- There is no null value for both event-data and no-event data
- “\N” values for all columns replaced with “” and then replace with NaN values
- We will remove these after the EDA before model building
- We are dropping the app_id and label_id columns in app_data since they are just identifiers and we have already used them to join the tables. They are not needed for modelling
- We also drop those rows which do not have is_active and is_installed both as 1 as otherwise the event does not apply to the current app/category of app.
- We convert the data types for columns in event data into correct ones before merging.

Feature engineering techniques that were used along with proper reasoning to support why the technique was used

Overall Idea: Basically we need to design features that will vary based on the age or gender of the person. Only those features which have high variance in relation to the age or gender will help in predicting the same.

Feature 1: Total and average number of events/day

1. Total event count per device id - the number of events is also directly proportional to how much the device is being used by different age and gender.
2. Divide the timestamp data into date
3. Then group the event records for each date and find out the average number of events/day. This along with Total events for a device_id (point 1) will give the model very good insights into the event frequency and total count.

Out[97]:

	Device_id	Gender	Age	Group	Event_id	Event_timestamp	Longitude	Latitude	Day of Week	Hour of day	Total_events	Avg_events
0	-1001337759327040000	M	30	M25-32	2774404	2016-05-07 09:14:24	119.61	29.70	Saturday	9	109	27.25
1	-1001337759327040000	M	30	M25-32	3065018	2016-05-04 10:26:14	120.29	30.42	Wednesday	10	109	27.25
2	-1001337759327040000	M	30	M25-32	3230344	2016-05-04 10:04:42	120.30	30.41	Wednesday	10	109	27.25
3	-1001337759327040000	M	30	M25-32	2906128	2016-05-07 10:24:32	119.69	29.80	Saturday	10	109	27.25
4	-1001337759327040000	M	30	M25-32	2876843	2016-05-07 10:23:56	119.69	29.80	Saturday	10	109	27.25
...	...	...	...	...	...	...	...	...	...	...	...	...
858854	999208698621622000	M	29	M25-32	603552	2016-05-03 12:55:40	114.38	27.82	Tuesday	12	35	5.00
858855	999208698621622000	M	29	M25-32	12506	2016-05-06 11:58:46	114.39	27.82	Friday	11	35	5.00
858856	999208698621622000	M	29	M25-32	106222	2016-05-04 16:21:29	114.37	27.82	Wednesday	16	35	5.00
858857	999208698621622000	M	29	M25-32	455299	2016-05-01 16:14:24	114.29	27.90	Sunday	16	35	5.00
858858	999208698621622000	M	29	M25-32	323879	2016-05-02 23:12:18	114.38	27.82	Monday	23	35	5.00

858859 rows × 12 columns

Now we have both total number of events/device and average number of events/device

Feature 2: Most active hour of the device

Get the hour component from the timestamp and calculate the hour when the number of events is highest for a device_id. - Group by device_id and hour component of timestamp and count(event_id). - Hour having maximum count(event_id)

This will greatly contribute to identifying the gender as the active hours for male and female are different.

]:

	Device_id	Gender	Age	Group	Event_id	Event_timestamp	Longitude	Latitude	Day of Week	Hour of day	Total_events	Avg_events	Active Hour
0	-1001337759327040000	M	30	M25-32	2774404	2016-05-07 09:14:24	119.61	29.70	Saturday	9	109	27.25	10
1	-1001337759327040000	M	30	M25-32	3065018	2016-05-04 10:26:14	120.29	30.42	Wednesday	10	109	27.25	10
2	-1001337759327040000	M	30	M25-32	3230344	2016-05-04 10:04:42	120.30	30.41	Wednesday	10	109	27.25	10
3	-1001337759327040000	M	30	M25-32	2906128	2016-05-07 10:24:32	119.69	29.80	Saturday	10	109	27.25	10
4	-1001337759327040000	M	30	M25-32	2876843	2016-05-07 10:23:56	119.69	29.80	Saturday	10	109	27.25	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...
858854	999208698621622000	M	29	M25-32	603552	2016-05-03 12:55:40	114.38	27.82	Tuesday	12	35	5.00	16
858855	999208698621622000	M	29	M25-32	12506	2016-05-06 11:58:46	114.39	27.82	Friday	11	35	5.00	16
858856	999208698621622000	M	29	M25-32	106222	2016-05-04 16:21:29	114.37	27.82	Wednesday	16	35	5.00	16
858857	999208698621622000	M	29	M25-32	455299	2016-05-01 16:14:24	114.29	27.90	Sunday	16	35	5.00	16
858858	999208698621622000	M	29	M25-32	323879	2016-05-02 23:12:18	114.38	27.82	Monday	23	35	5.00	16

858859 rows × 13 columns

As you can see above we have captured the most active hour for each device

Feature 3: Total number of places visited by device

Based on the different places a person travels to, his latitude and longitude combination will have that many values.

We can find the total number of unique lat, long values for each device. That should give us the number of different places a person has gone to which will vary greatly based on age and gender.

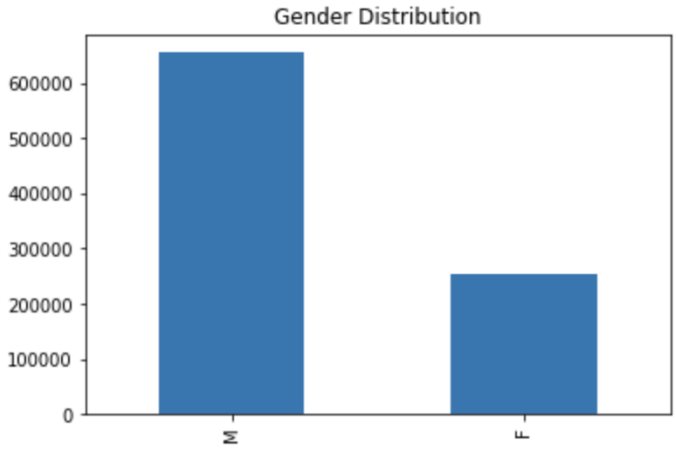
	Device_id	Gender	Age	Group	Event_id	Event_timestamp	Longitude	Latitude	Day of Week	Hour of day	Total_events	Avg_events	Active Hour	Total_places_visited
0	-1001337759327040000	M	30	M25-32	2774404	2016-05-07 09:14:24	119.61	29.70	Saturday	9	109	27.25	10	21
1	-1001337759327040000	M	30	M25-32	3065018	2016-05-04 10:26:14	120.29	30.42	Wednesday	10	109	27.25	10	21
2	-1001337759327040000	M	30	M25-32	3230344	2016-05-04 10:04:42	120.30	30.41	Wednesday	10	109	27.25	10	21
3	-1001337759327040000	M	30	M25-32	2906128	2016-05-07 10:24:32	119.69	29.80	Saturday	10	109	27.25	10	21
4	-1001337759327040000	M	30	M25-32	2876843	2016-05-07 10:23:56	119.69	29.80	Saturday	10	109	27.25	10	21
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
58854	999208698621622000	M	29	M25-32	603552	2016-05-03 12:55:40	114.38	27.82	Tuesday	12	35	5.00	16	4

Feature 4: Generalisation of category

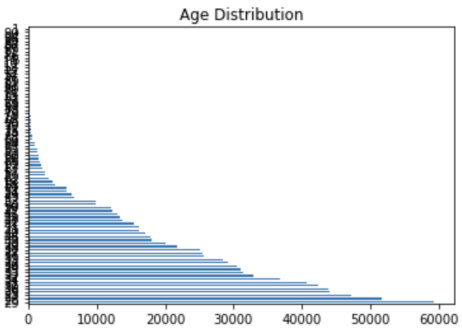
We tried reducing the category by generalising them into common parents. However even after applying only on a dataframe with just category column we got memory issues. The code has been given in the notebook.

Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation

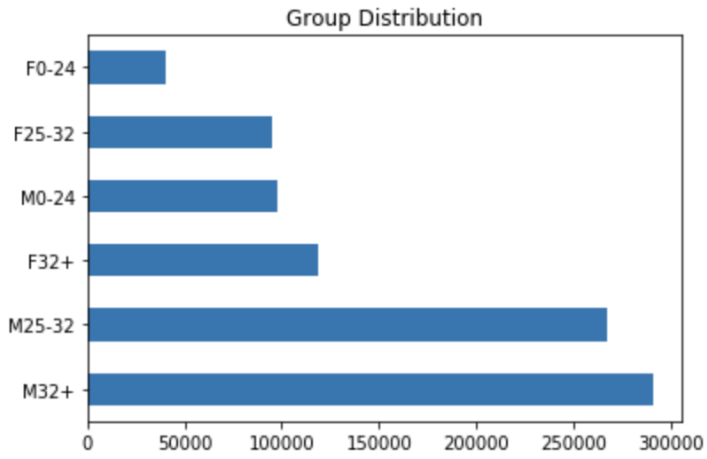
Plot appropriate graphs representing the distribution of age and gender in the data set [univariate].



Looks like the Male population is more than female population in our data

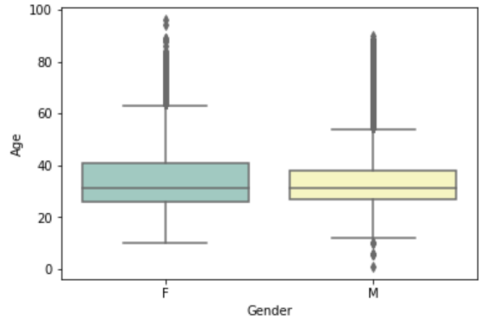


From above we can see that we have less people as age increases. There are more people with lower age which makes sense as people who are young use mobile more than people who are old



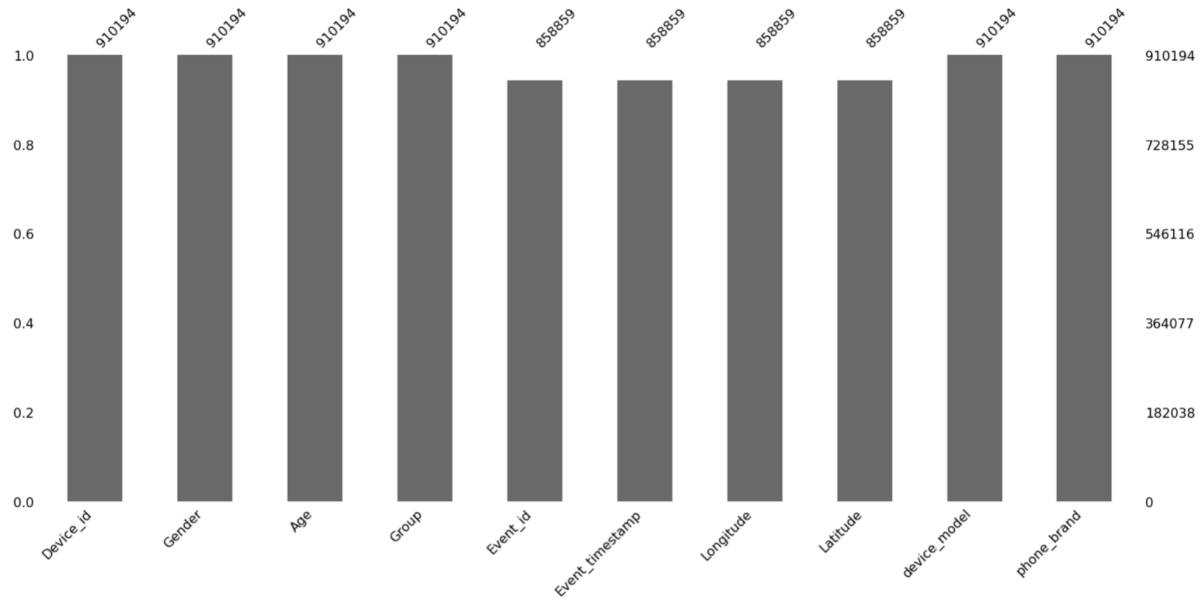
We can also see that Male with age 32 and above are the highest followed by Male between 25 and 32

Boxplot analysis for gender and age [bivariate].



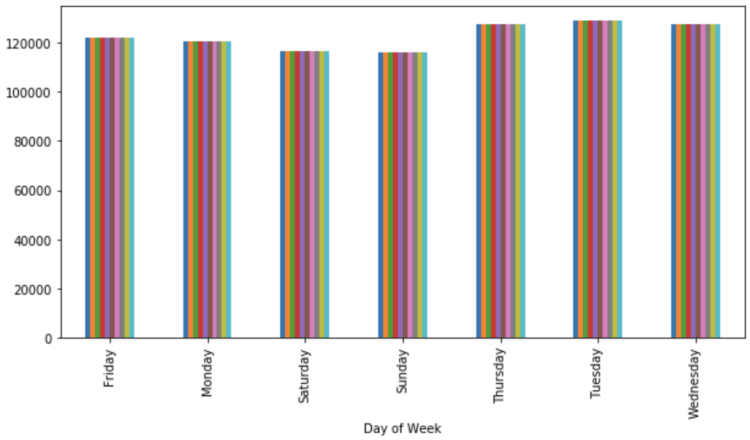
From above plot we can say that the age range for male is less whereas females are across a wider range. The mean though is same for both male and female and there is no one above 40

Plot the percentage of the device_ids with and without event data.



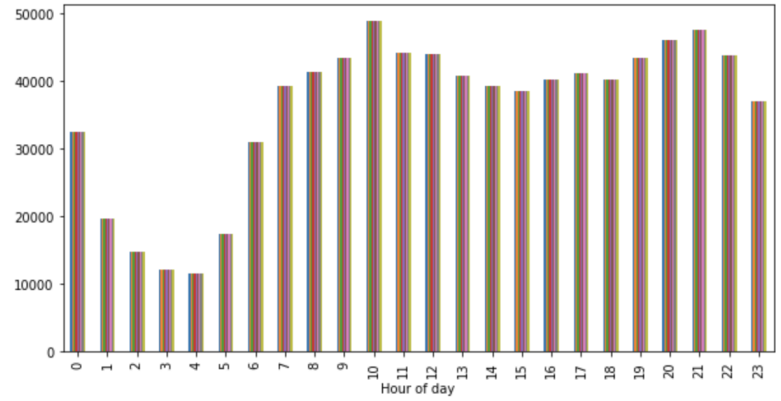
We have just reused the same result we got above for the count of NaN in event_id. 4% of the device_id does not contain events and remaining 96% contains events. You can also see missing values of all other columns here

Plot a graph representing the distribution of events over different days of a week.



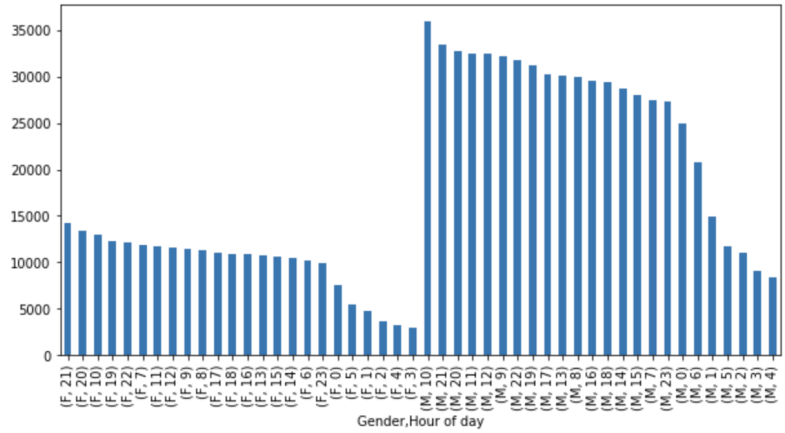
As we can see above, the events is high during mid of week Tuesday and Wednesday and low in weekends. Looks like our data has people who use the mobile more on weekdays

Plot a graph representing the distribution of events per hour [for one-week data].



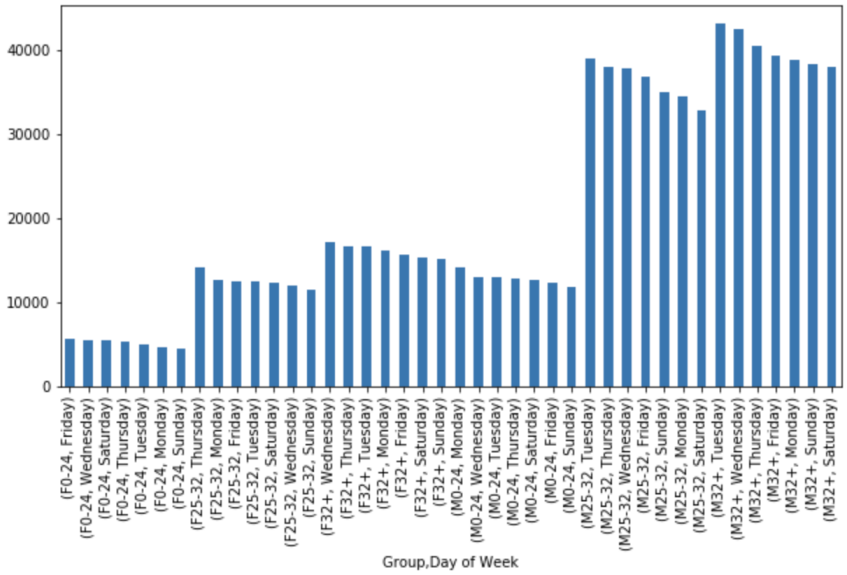
As we can see above, the events are more at highest at morning 10AM followed by night 9PM. The lowest is in early morning 4AM which makes sense that people use mobile more in start of day and end of day

The difference in the distribution of events per hour for Male and Female consumers. [Show the difference using an appropriate chart for one-week data.]



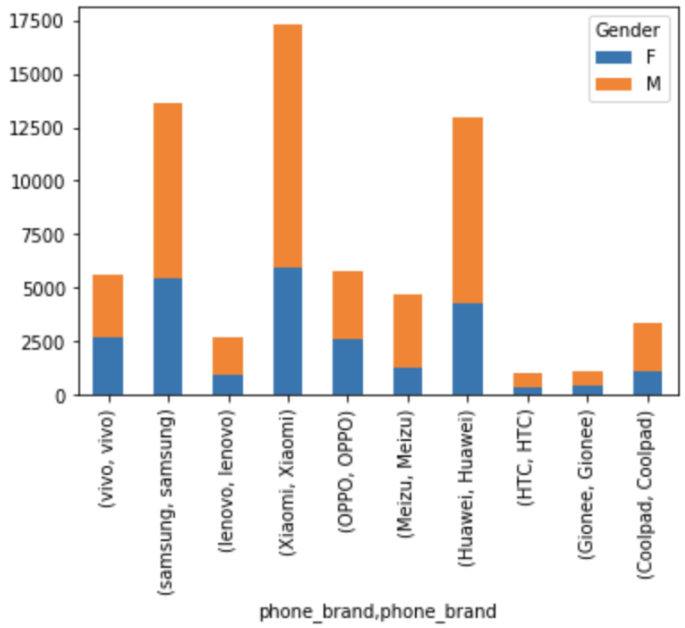
We can see how the events are distributed seggregated by male and female. HIghest for female is 9PM and highest for male is morning 10 AM followed by 9PM

Is there any difference in the distribution of Events for different Age Groups over different days of the week? [Consider the following age groups: 0–24, 25–32, 33–45, and 46+]



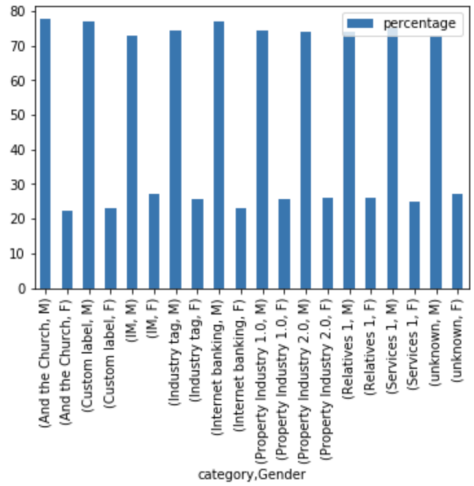
Yes. The highest for M32+ is on Tuesday whereas for M0-24 it is Monday. Similarly the day varies for female too for each age group.

Stacked bar chart for the top 10 mobile brands across male and female consumers.



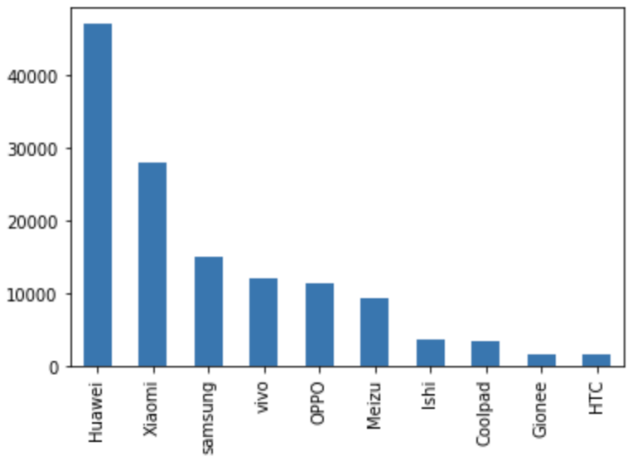
As you can see the top brands are Xiaomi, Huawei and Samsung

Prepare a chart representing the ten frequently used applications and their respective male and female percentage.

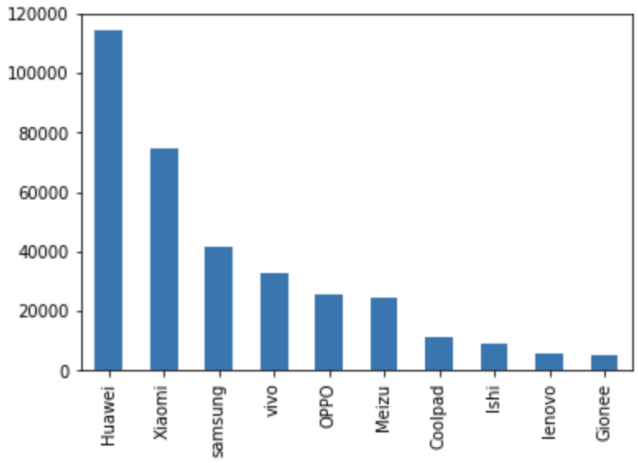


We are using categories to denote frequently used applications since we don't have app name present in the dataset. We can see the male and female percentage above

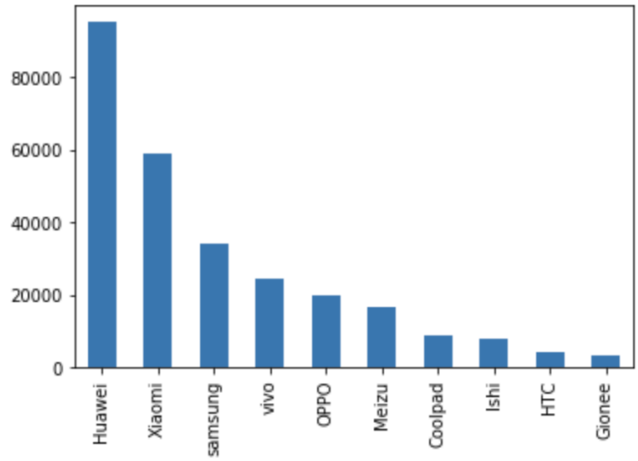
List the top 10 mobile phone brands bought by customers by age groups. [Consider the following age groups: 0–24, 25–32, 33–45, and 46+]



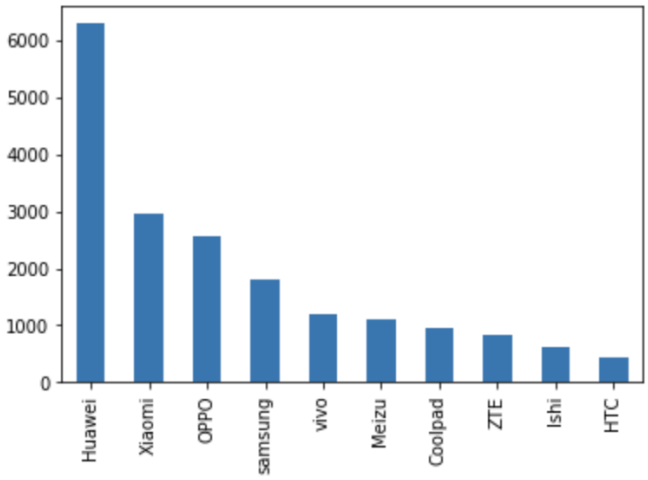
Top ten brands for people in age-group 0-24 are Huawei, Xiaomi, Samsung followed by others as seen above.



Top ten brands for people in age-group 25-32 are very similar to 0-24 age group



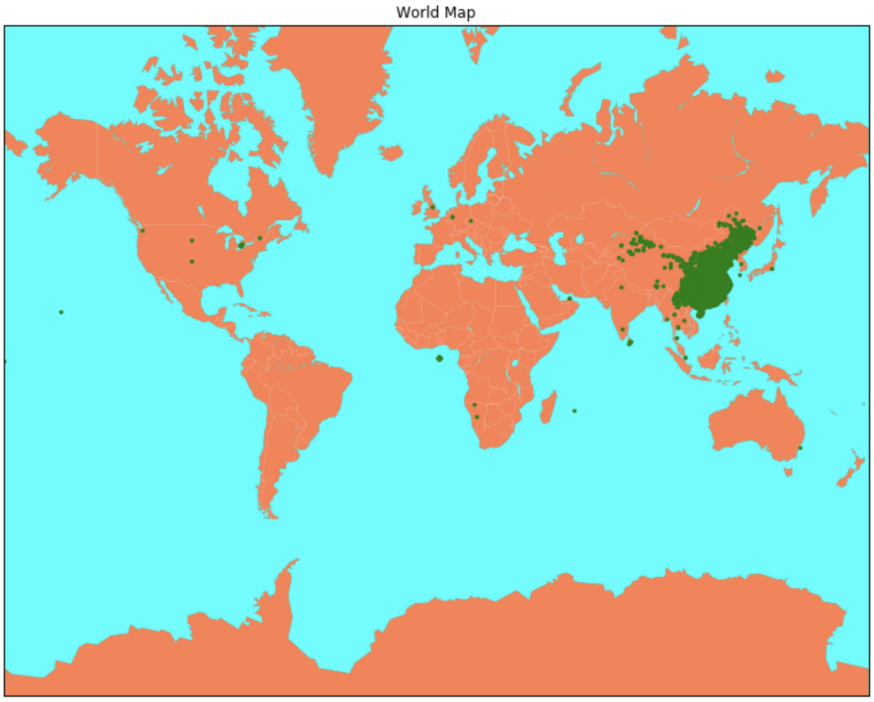
Top ten brands for people in age-group 33–45 are very similar to 0-24 age group



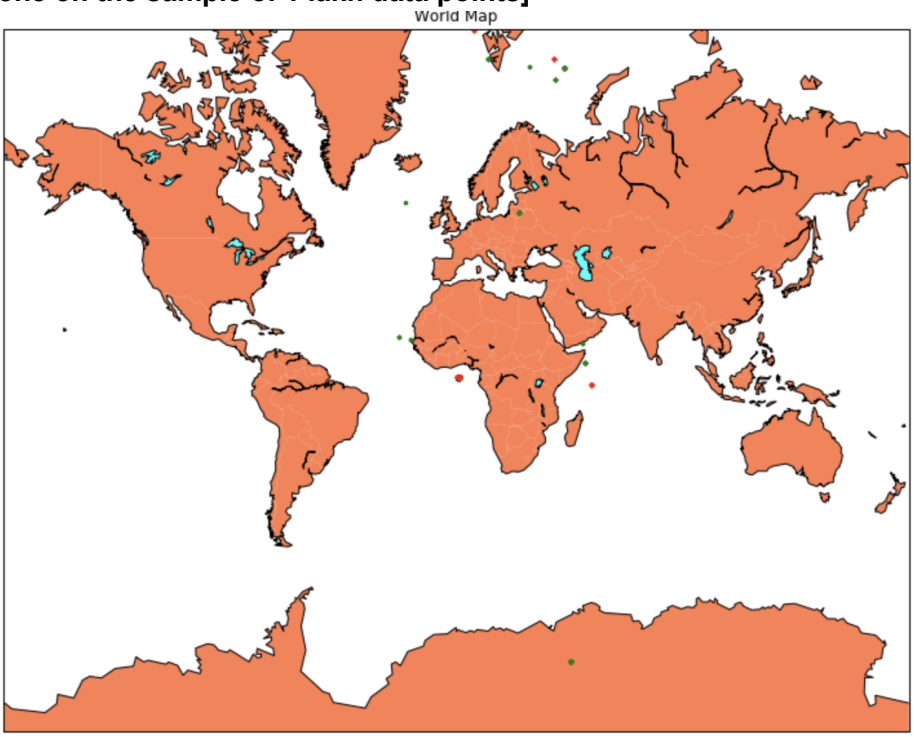
As you can see Oppo is 3rd in this group whereas it is 5th in 33-45 age group

Geospatial visualisations along with the insights gathered from this visualisation

Plot the visualisation plot for a sample of 1 lakh data points.
We initially plotted on the India map but did not get enough data points. Since the data can also be from the world population, we plotted on a world map and got more points concentrated in the Asia region as below.

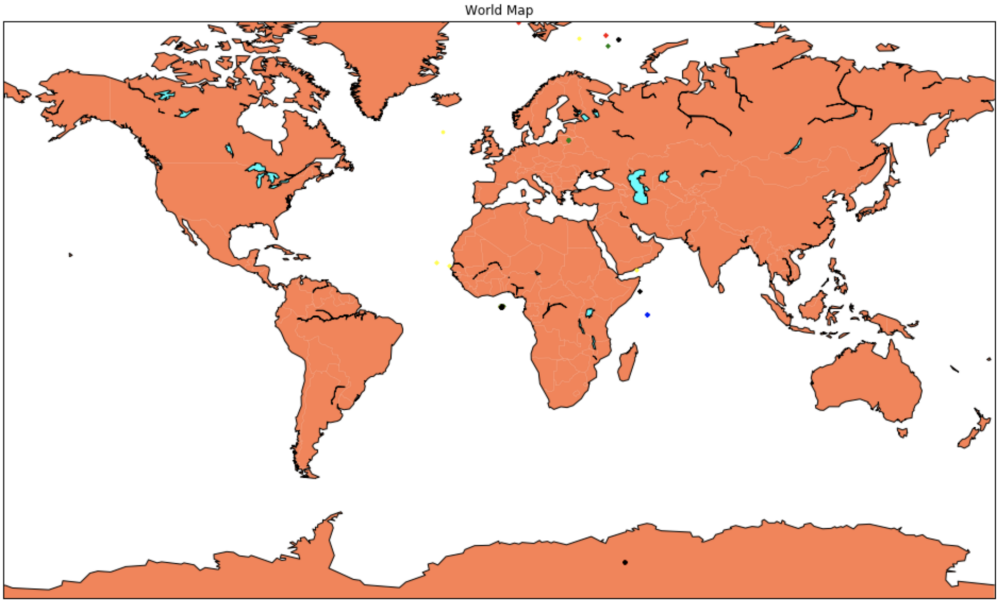


Compare the event visualisation plots based on the users' gender information. [This can be done on the sample of 1 lakh data points]



We have given green dots for one gender and red for another.

Compare the event visualisation plots based on the following age groups: [0–24,25–32, 32+]



Red, blue, green, yellow and black dots have been assigned to different age groups.

Results interpreting the clusters formed as part of DBSCAN Clustering and how the cluster information is being used

We got 8213 clusters formed using DBSCAN. We then merged the clusters using the Latitude and Longitude values to the original table and then dropped the actual longitude and latitude columns.

We then used these clusters in model building.

A brief summary of any additional subtask that was performed and may have improved the data cleaning and feature generation step

We dropped duplicates at various points of the dataframe which helped in forming the clusters in DBSCAN. Without dropping the duplicates, it was taking a long time to form the clusters.

All the data preparation steps that were used before applying the ML algorithm

1. We did encoding for all categorical columns.
2. We used GroupShuffleSplit class to split the data based on the train_test_flag which was given in the CSV.
3. Since the dataset is very huge, in order to save memory and time we used label encoding mostly. However in real time we should use one hot encoding for variables which does not have any ordinal value in them.

Documentation of all the machine learning models that were built along with the respective parameters that were used (e.g., DBSCAN, XGBoost, Random Forest, GridSearchCV, etc.)

Logistic/Linear Regression:

1. We first used Logistic/Linear regression as a base model to set the benchmark on the minimum accuracy and various metrics that are required.
2. We used the default parameters of Logistic/Linear Regression.

XGBoost:

1. We tried XGBoost with a lot of parameters but they caused either memory error or ran for a very long time and did not complete.
2. Below are reduced set of parameters we then used and got the model running.
   params = {
   'gamma': [0.5, 1, 1.5, 2, 5],
   'max_depth': [3, 4, 5]
   }

We used kfold as 2 for cross validation..

Stacking:

We finally used stacking for all the scenarios with Logistic/Linear regression as applicable and Random forest with 10 trees to avoid memory issues. Then we used an XGBClassifier/Regressor on top of these models.

The reason for using regression or classification for age prediction

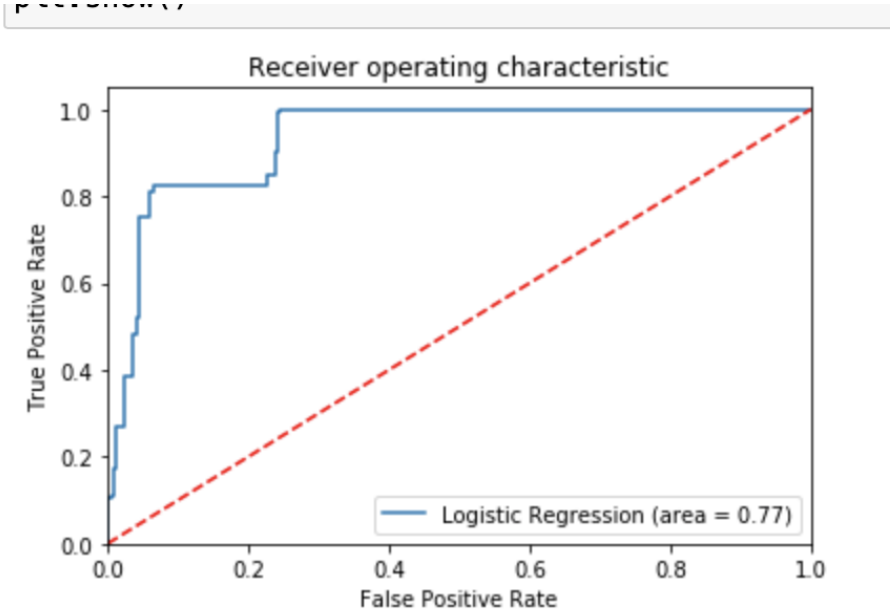
Since age is a continuous variable rather than discrete variable regression is the best method for data that is continuous. Though we need the age to be in bins later, we can put them into corresponding bins.

The outcomes of the evaluation metrics (results for both Scenario 1 and Scenario 2 must be shown separately).

Below are the inferences from each model evaluation metrics.

Scenario 1 - Gender Prediction:

1. The area under the ROC curve came to 0.77 which means the model performs pretty decently with base model logistic regression. Also precision, recall and f1 score are also pretty decent above 0.8 in all cases and with f1 score of 0.88 for Gender Female and 0.69 for Gender male



2. The optimal threshold probability came to **0.379930** with KS statistic for logistic regression

:

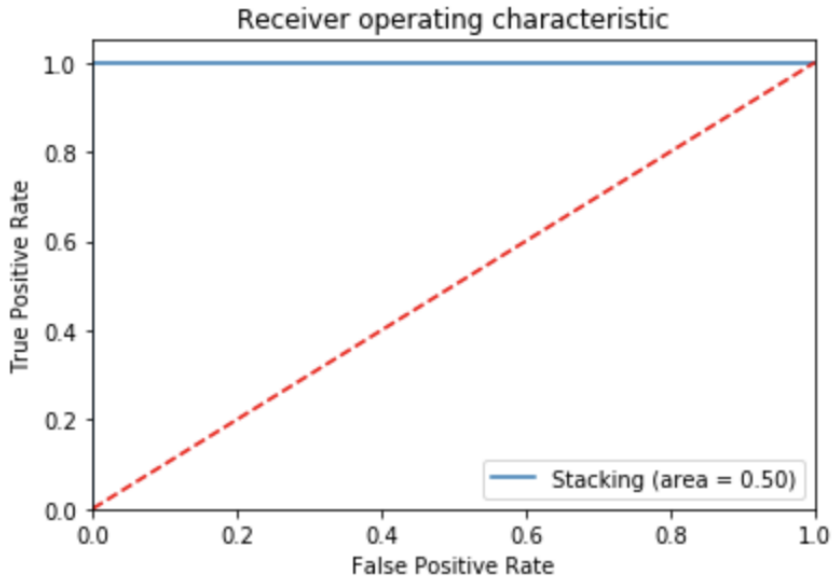
	min_scr	max_scr	bads	goods	total	odds	bad_rate	ks	max_ks
0	0.379930	0.404284	18235.0	1209.0	19444.0	0.07	93.78%	4.37	<----
1	0.404920	0.417056	18795.0	6763.0	25558.0	0.36	73.54%	-3.98	
2	0.417608	0.443658	0.0	0.0	0.0	nan	nan%	-3.98	
3	0.447019	0.477709	0.0	0.0	0.0	nan	nan%	-3.98	
4	0.479421	0.487558	22504.0	2245.0	24749.0	0.10	90.93%	-0.33	
5	0.488201	0.509897	7142.0	1046.0	8188.0	0.15	87.23%	0.05	
6	0.510375	0.524429	23233.0	8059.0	31292.0	0.35	74.25%	-9.57	
7	0.525423	0.541283	22940.0	1528.0	24468.0	0.07	93.76%	-4.09	
8	0.541992	0.578595	16453.0	4303.0	20756.0	0.26	79.27%	-7.63	
9	0.580484	0.617210	21922.0	4492.0	26414.0	0.20	82.99%	-9.45	

3. XGBoost performed much better than the logistic base model giving 1.00 precision for females and 1.00 recall for male. The precision for male and recall for female was less though. However we got 1.00 for the AUC score. We might be getting 100% here for AUC since the data taken is not full data. If we run with

entire 92L data then we would get more realistic metrics.However we can conclude the model still performs good with this data.

```
              precision    recall  f1-score   support

           0       1.00      0.24      0.39    100514
           1       0.41      1.00      0.58     52974

    accuracy                           0.50    153488
   macro avg       0.71      0.62      0.49    153488
weighted avg       0.80      0.50      0.46    153488


accuracy_score 0.5044107682685292

roc-auc score for the class 1, from target 1.0

elapsed time in seconds:  297.3593578338623
-----
Best parameters {'gamma': 0.5, 'max_depth': 4}
Mean cross-validated accuracy score of the best_estimator: 1.000
```

4. When we used stacking finally we got 1 in the area under the ROC curve and accuracy was 100%.



5. Hence we can conclude that for Scenario 1 - Gender prediction Stacking is the best model

Scenario 1 - Age Prediction:

We used Linear regression for age prediction Scenario 1 and below are metrics.

1. We got pretty low R2 score of 0.18 and RMSE was high as 43.29613212721642 which might be because linear regression is a basic model

```
Slope: [-5.45025415e-19 -2.54369020e-09 -8.97898275e-08  6.03626190e-07
  8.35958261e-08  2.58628869e-10  6.75272214e-09  0.00000000e+00
  0.00000000e+00 -1.00640026e-08 -1.28526756e-05]
Intercept: 31.226399997768965
Root mean squared error:  43.29613212721642
R2 score:  0.18435999435850503
```

2. However the percentage of difference between actual and predicted was pretty less as -1%

```
18]: Perc = (diff/df_out["A
```

```
19]: Perc.head(25)
```

```
19]: 0    -1.043479
     1    -1.043480
     2    -1.043478
     3    -1.043476
     4    -1.043480
     5    -1.043483
     6    -1.043479
     7    -1.043478
     8    -1.043479
     9    -1.043481
     10   -1.043477
     11   -1.043479
     12   -1.043478
     13   -1.043478
     14   -1.043473
     15   -1.043479
     16   -1.043476
```

3. When we used XGBoost, we found that the model is overfitting as it gave low error for train and high error for test.

```
[20:10:05] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/objective/regress
1: reg:linear is now deprecated in favor of reg:squarederror.
RMSE for test :  11.168503
RMSE for train :  0.867478
```

Since RMSE for test is greater than RMSE of train data for the XGBoost, the model is over-fitting.

click to expand output; double click to hide output

4. With Stacking however the RMSE and R2 score was very high compared to linear and XGBoost as shown below. Hence we use stacking as best model for deployment

RMSE for train : 0.867478 in XGBoost is more than the RMSE for train in Stacking -RMSE Value 0.0

R2 score: 0.18435999435850503 for Linear Regression is less than the R2 score for Stacking R2 Score 1.0

Stacking for Age Prediction is performing better than all other model as inferences can be seen above. Hence the stacking model will be saved for Model Deployment

Scenario 2 - Age Prediction:

1. When using linear regression for scenario 2 age prediction we got low R2 score and high RMSE from which we can infer the model did not perform well.

```
Slope: [-2.55955529e-21  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00]
Intercept: 31.45813427838783
Root mean squared error:  97.06887475779143
R2 score:  1.92088912842614e-06
```

2. The percentage population distribution were also as high as -36%

```
]: Perc.head()

]: 0     4.641340
   1   -36.676159
   2     4.666680
   3     7.538228
   4   -16.437189
   dtype: float64
```

3. But when we used XGBoost the RMSE decreased to great extent to 5.5

```
[21:02:26] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/objective/regression_obj.cu:17
1: reg:linear is now deprecated in favor of reg:squarederror.
RMSE :  5.546937
```

4. Stacking gave a much better CV Regressor score of 0.727.

```
StackingCVRegressor SCORE :  0.7273490974803062
```

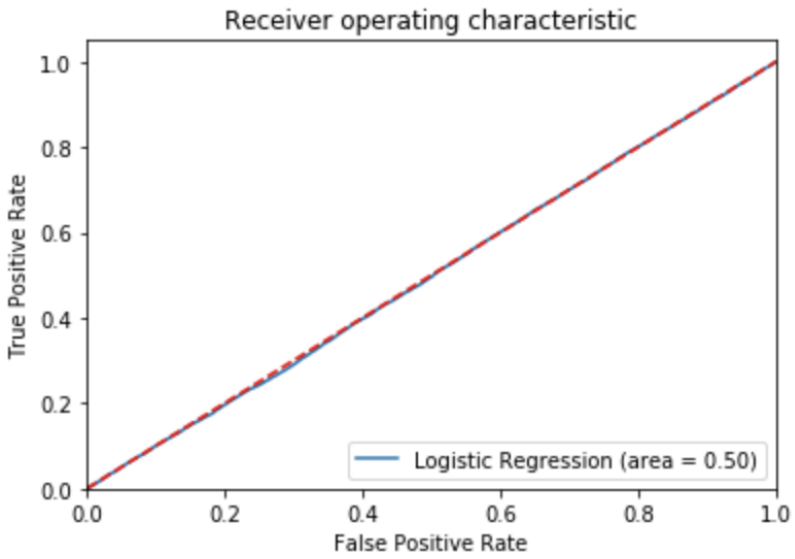
Hence stacking still performs better than other models.

Scenario 2 - Gender Prediction:

1. The basic logistic regression model gave poor recall, precision and F1 score as given below and also AUC was 0.5 only.

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.36      0.50      0.42     21002
           1       0.64      0.50      0.56     37548

    accuracy                           0.50     58550
   macro avg       0.50      0.50      0.49     58550
weighted avg       0.54      0.50      0.51     58550
```



2. XGBoost however gave a very good score of 1.0 and obviously better than logistic regression

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     21002
           1       1.00      1.00      1.00     37548

    accuracy                           1.00     58550
   macro avg       1.00      1.00      1.00     58550
weighted avg       1.00      1.00      1.00     58550


accuracy_score 1.0

roc-auc score for the class 1, from target 1.0

elapsed time in seconds:  80.18289995193481
-----
Best parameters {'gamma': 0.5, 'max_depth': 3}
Mean cross-validated accuracy score of the best_estimator: 1.000
```

3. When using stacking too we got full accuracy.

```
Accuracy: 1.00 (+/- 0.00) [lr]
Accuracy: 1.00 (+/- 0.00) [Random Forest]
[21:28:12] WARNING: C:/Users/Administrator/workspace/xgboost-v
GBoost 1.3.0, the default evaluation metric used with the obje
logloss'. Explicitly set eval_metric if you'd like to restore
[21:28:12] WARNING: C:/Users/Administrator/workspace/xgboost-v
GBoost 1.3.0, the default evaluation metric used with the obje
logloss'. Explicitly set eval_metric if you'd like to restore
[21:28:13] WARNING: C:/Users/Administrator/workspace/xgboost-v
GBoost 1.3.0, the default evaluation metric used with the obje
logloss'. Explicitly set eval_metric if you'd like to restore
Accuracy: 1.00 (+/- 0.00) [StackingClassifier]
```

Final Summary:
We recommend stacking here too since it is a combination of different models and provides variation in the output and will perform much more efficiently on large data than other models.

Using stacking makes sure the model is as generalised as possible as it the output is not dependent on any single model but a combination of models.