# IIT- M  Advanced Certificate Program in Machine Learning and Cloud – up Grad Capstone Project

*User Demographics Prediction using Telecom dataset*

*Task3_Module*

*Authors :*

*Mukul Pahawa*

*Mitesh*

Contribution

Mukul Pahawa= 60%
Mitesh = 40%.

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

Non Events Data

```
dfnev.head(5)
```

[18]:

| | device_id | phone_brand | device_model | gender | age | group_train |
|---|---|---|---|---|---|---|
| **0** | 1845358998536310000 | meitu | 2 | F | 25 | F25-32 |
| **1** | 3126957642374570000 | meitu | 2 | M | 27 | M25-32 |
| **2** | 6005031767544890000 | meitu | 2 | F | 30 | F25-32 |
| **3** | 7862170554164260000 | meitu | 2 | F | 22 | F0-24 |
| **4** | -1463646610464190000 | meitu | 2 | F | 24 | F0-24 |

## Events

```
#
dfev.head(5)
```

[29]:

| | device_id | event_id | event_time | latitude | longitude | gender | age | group_train |
|---|---|---|---|---|---|---|---|---|
| **0** | 4676982795249940000 | 2958924 | 2016-05-01 22:03:46 | 113.24 | 22.85 | M | 52 | M32+ |
| **1** | 4782582047729160000 | 2958931 | 2016-05-01 22:24:09 | 114.47 | 38.03 | M | 36 | M32+ |
| **2** | 2181284491650730000 | 2958933 | 2016-05-01 22:24:44 | 125.66 | 43.02 | M | 28 | M25-32 |
| **3** | 4221762657972680000 | 2958935 | 2016-05-01 22:40:47 | 115.19 | 24.07 | F | 19 | F0-24 |
| **4** | -2990318111507320000 | 2958938 | 2016-05-01 21:59:08 | 0.00 | 0.00 | F | 32 | F25-32 |

## Apps Data

```
            )
```

[37]:
```
# display first 5 records
#
dfapp.head(5)
```

[37]:

| | event_id | app_id | is_installed | is_active | label_id | category |
|---|---|---|---|---|---|---|
| **0** | 2960967 | 8557198901083791098 | 1.0 | 1.0 | 548.0 | Industry tag |
| **1** | 2960967 | 8557198901083791098 | 1.0 | 1.0 | 204.0 | sports and gym |
| **2** | 2960967 | 8557198901083791098 | 1.0 | 1.0 | 172.0 | IM |
| **3** | 2960967 | 8557198901083791098 | 1.0 | 1.0 | 223.0 | convenience services |
| **4** | 2960967 | 8557198901083791098 | 1.0 | 1.0 | 206.0 | Medical |

[38]:
```
total_rows = dfapp.shape[0]
```

## TASK 2 - Cleaning Data

- Null checks
- Imputations
- Handling Categorical, Binary and Numerical columns
- Check for high cardinality columns
- Handling categorical columns having large values

TASK 3 - Basic EDA and Visualization, Feature Engineering Ideas

- Plot appropriate graphs which represent the distribution of Age and gender in the Dataset [Univariate]
- Boxplot analysis for gender and Age [Bivariate]
- Plot percentage of device_ids with and without event data
- Graph representing the distribution of events on different days of a week
- Graph representing the distribution of events per hour [ For one-week data]
- The difference in the distribution of events per hour for Male and Females? [Show the difference using an appropriate chart for one-week data]
- Is there any difference in the distribution of Events for different Age Groups over different days of the week? [Consider the age groups as 0-24, 25-32, 33-45, 46+]
- Stacked bar chart for top 10 mobile brands across male and female consumers
- Chart representing ten frequent applications and their respective male and female percentage
- Top 10 Mobile Phone Brand by age groups [Consider the age groups as 0-24, 25-32, 33-45, 46+]

**Feature Engineering**

Feature Engineering Non-event dataset

Grouping the existing categories to create a new super category that will establish a significance in predicting the outcome variable

significance in predicting the outcome variable

In [6]:
```
#function to calculage the age_group regardless of gender
def getAgeGroup(row):
    if (row['group_train'] == 'M0-24') or (row['group_train'] == 'F0-24') :
        return '0-24'
    elif (row['group_train'] == 'M25-32') or (row['group_train'] == 'F25-32') :
        return '25-32'
    else:
        return '32+'
```

In [7]:
```
#create supercategory for age group
dfnev['age_group'] = dfnev.apply(getAgeGroup, axis=1)
```

In [8]:
```
dfnev.head(5)
```

Out[8]:

| | device_id | phone_brand | device_model | gender | age | group_train | age_group |
|---|---|---|---|---|---|---|---|
| 0 | -7548291590301750000 | Huawei | 3C | M | 33 | M32+ | 32+ |
| 1 | 6943568600617760000 | Xiaomi | xnote | M | 37 | M32+ | 32+ |
| 2 | 5441349705980020000 | OPPO | R7s | M | 40 | M32+ | 32+ |
| 3 | -5393876656119450000 | Xiaomi | MI 4 | M | 33 | M32+ | 32+ |

Grouping the existing categories to create a new supercategory that will establish a significance in predicting the outcome variable

Feature Engineering event dataset 4. You may also think of grouping the existing categories to create a new supercategory that will establish a significance in predicting the outcome variable

In [12]:
```
#create supercategory for age group
dfev['age_group'] = dfev.apply(getAgeGroup, axis=1)
```

In [14]:
```
dfev.head(5)
```

Out[14]:

| | device_id | event_id | event_time | latitude | longitude | gender | age | group_train | age_group |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4676982795249940000 | 2958924 | 2016-05-01 22:03:46 | 113.24 | 22.85 | M | 52 | M32+ | 32+ |
| 1 | 4782582047729160000 | 2958931 | 2016-05-01 22:24:09 | 114.47 | 38.03 | M | 36 | M32+ | 32+ |
| 2 | 2181284491650730000 | 2958933 | 2016-05-01 22:24:44 | 125.66 | 43.02 | M | 28 | M25-32 | 25-32 |
| 3 | 4221762657972680000 | 2958935 | 2016-05-01 22:40:47 | 115.19 | 24.07 | F | 19 | F0-24 | 0-24 |
| 4 | -6242501228649110000 | 2958939 | 2016-05-01 21:59:40 | 111.21 | 27.85 | M | 20 | M0-24 | 0-24 |

In [15]:
```
#save non-event dataframe to csv
dfnev.to_csv(r'non_events_init2.csv', index = False)
```

In [16]:
```
#save event dataframe to csv
dfev.to_csv(r'events_init2.csv', index = False)
```

A feature called "total_events" can be added here based on the total number of events for each device id

**A feature called "total_events" can be added here based on the total number of events for each device id**

In [53]:
```
event.groupby(["device_id"])["event_id"].count()
```

Out[53]:
```
device_id
-9222956879900150000     52
-9221026417907250000    132
-9220061629197650000     39
-9218769147970100000     17
-9215352913819630000     18
                       ...
 9215085115859650000     17
 9216925254504440000     73
 9219164468944550000    407
 9219842210460030000      4
 9220914901466450000      3
Name: event_id, Length: 11949, dtype: int64
```

In [54]:
```
df=pd.DataFrame(event.groupby(["device_id"])["event_id"].count())
df.rename(columns={"event_id":"total_events"},inplace=True)
df.head()
```

Out[54]:

| device_id | total_events |
|---|---|
| -9222956879900150000 | 52 |
| -9221026417907250000 | 132 |
| -9220061629197650000 | 39 |
| -9218769147970100000 | 17 |
| -9215352913819630000 | 18 |

## Output of EDA and Visualization

Univariate Analysis

### 3.1 Plot appropriate graphs which represent the distribution of Age and gender in the Dataset [Univariate]

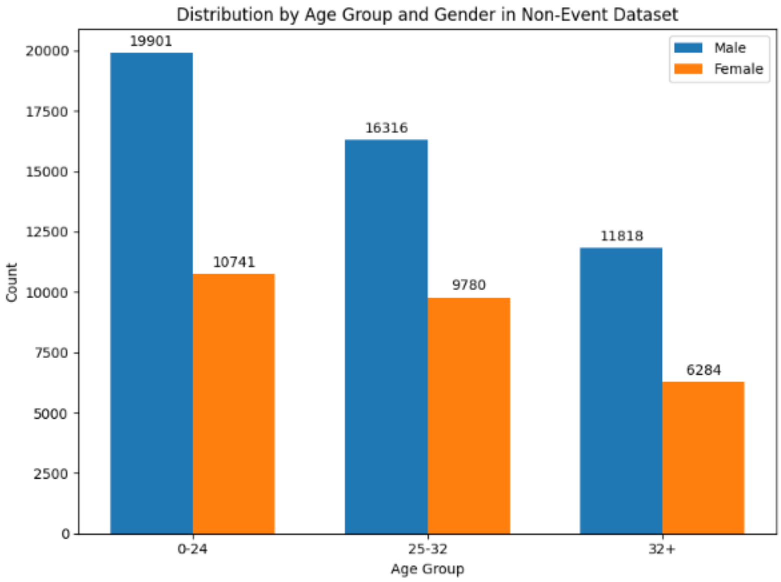
In [49]:
```
# Get the count of males and females in the event and non-event data set grouped by the age group (0-24,25-32, 32+)
#
nev_males = dfnev[dfnev['group_train'].isin(['M0-24','M25-32','M32+'])]['group_train'].value_counts()
nev_females= dfnev[dfnev['group_train'].isin(['F0-24','F25-32','F32+'])]['group_train'].value_counts()
ev_males = dfev[dfev['group_train'].isin(['M0-24','M25-32','M32+'])]['group_train'].value_counts()
ev_females= dfev[dfev['group_train'].isin(['F0-24','F25-32','F32+'])]['group_train'].value_counts()
```

In [50]:
```
print("non event data set - males")
print(nev_males)
print(" ")
print("non event data set - females")
print(nev_females)
print(" ")
print("event data set - males")
print(ev_males)
print(" ")
print("event data set - females")
print(ev_females)
```

```
non event data set - males
group_train
M25-32    19901
M32+      16316
M0-24     11818
Name: count, dtype: int64

non event data set - females
group_train
F25-32    10741
F32+       9780
F0-24      6284
```

#### 3.1.1 Non-Event Dataset

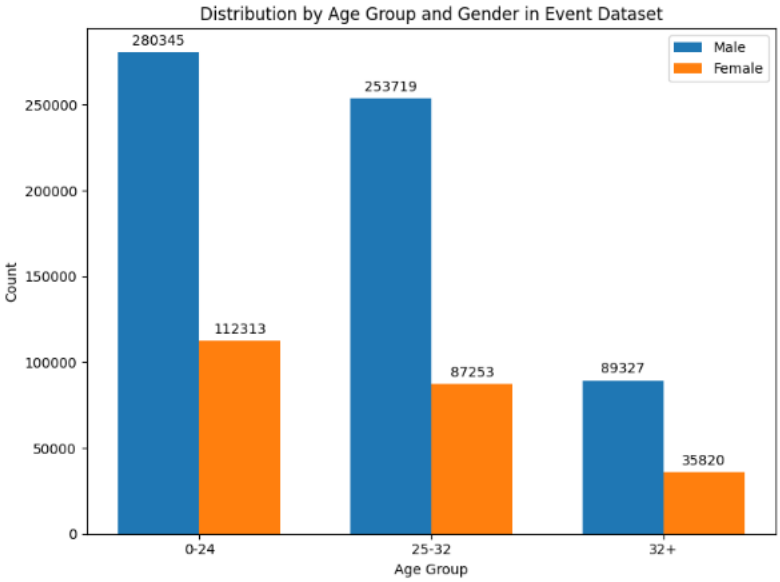
In [52]:
```
labels = ['0-24', '25-32', '32+']
nev_males = nev_males.tolist()
nev_femals = nev_females.tolist()
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots(figsize=(8,6))
rects1 = ax.bar(x - width/2, nev_males, width, label='Male')
rects2 = ax.bar(x + width/2, nev_females, width, label='Female')
ax.set_xlabel('Age Group')
ax.set_ylabel('Count')
ax.set_title('Distribution by Age Group and Gender in Non-Event Dataset')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
autolabel(rects1)
autolabel(rects2)
fig.tight_layout()
plt.show()
```




#### 3.1.2 Event dataset

#### 3.1.2 Event dataset

In [53]:
```
labels = ['0-24', '25-32', '32+']
ev_males = ev_males.tolist()
ev_femals = ev_females.tolist()
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots(figsize=(8,6))
rects1 = ax.bar(x - width/2, ev_males, width, label='Male')
rects2 = ax.bar(x + width/2, ev_females, width, label='Female')
ax.set_xlabel('Age Group')
ax.set_ylabel('Count')
ax.set_title('Distribution by Age Group and Gender in Event Dataset')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
autolabel(rects1)
autolabel(rects2)
fig.tight_layout()
plt.show()
```



Boxplot analysis for gender and Age [Bivariate]

#### 4.2 Boxplot analysis for gender and Age [Bivariate]

##### 4.2.1 Non-Event dataset

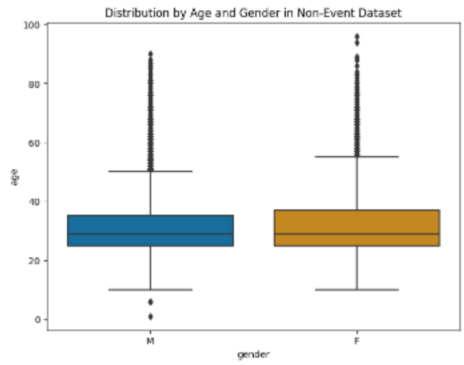
In [54]:
```
# Printing the statistics for non-event data set
# with Gender == Female
#
df_tmp1 =  dfnev.loc[dfnev.gender=='F']
df_tmp1['age'].describe()
```

Out[54]:
```
count    26805.000000
mean        32.051483
std         10.539238
min         10.000000
25%         25.000000
50%         29.000000
75%         37.000000
max         96.000000
Name: age, dtype: float64
```

In [55]:
```
# Printing the statistics for non-event data set
# with Gender == Male
#
df_tmp2 =  dfnev.loc[dfnev.gender=='M']
df_tmp2['age'].describe()
```

Out[55]:
```
count    48035.000000
mean        31.048798
std          9.452689
min          1.000000
25%         25.000000
50%         29.000000
75%         35.000000
max         90.000000
Name: age, dtype: float64
```

In [56]:
```
plt.figure(figsize=(8,6))
sns.boxplot(y='age', x='gender',
                data=dfnev,
                palette='colorblind'
            )
plt.title('Distribution by Age and Gender in Non-Event Dataset')
plt.show()
```



#### 4.2.2 Event Dataset

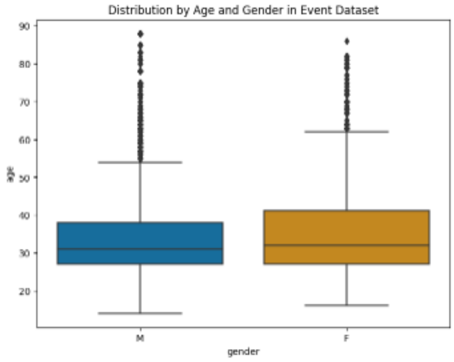
In [57]:
```
# Printing the statistics for event data set
# with Gender == Female
#
df_tmp3 =  dfev.loc[dfev.gender=='F']
df_tmp3['age'].describe()
```

Out[57]:
```
count    235386.000000
mean         34.362940
std          10.604955
min          16.000000
25%          27.000000
50%          32.000000
75%          41.000000
max          86.000000
Name: age, dtype: float64
```

In [58]:
```
# Printing the statistics for event data set
# with Gender == Male
#
df_tmp4 =  dfev.loc[dfev.gender=='M']
df_tmp4['age'].describe()
```

Out[58]:
```
count    623391.000000
mean         33.337809
std           9.393527
min          14.000000
25%          27.000000
50%          31.000000
75%          38.000000
max          88.000000
Name: age, dtype: float64
```

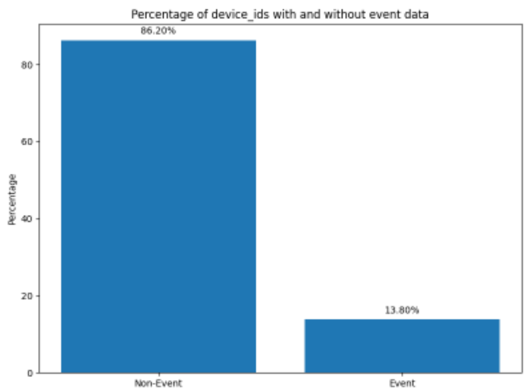
In [59]:
```
plt.figure(figsize=(8,6))
sns.boxplot(y='age', x='gender',
                data=dfev,
                palette='colorblind'
            )
plt.title('Distribution by Age and Gender in Event Dataset')
plt.show()
```




Plot percentage of device_ids with and without event data


#### 4.3. Plot percentage of device_ids with and without event data

In [60]:
```
nev_deviceid_count = dfnev['device_id'].unique().size
ev_deviceid_count = dfev['device_id'].unique().size
total_devices_cnt = nev_deviceid_count + ev_deviceid_count
fig, ax = plt.subplots(figsize=(8,6))
labels = ['Non-Event', 'Event']
device_perc = [(nev_deviceid_count/total_devices_cnt)*100, (ev_deviceid_count/total_devices_cnt)*100]
ax.bar(labels,device_perc)
plt.title('Percentage of device_ids with and without event data')
plt.ylabel('Percentage')

plt.annotate('{:.2f}'.format(device_perc[0])+'%', # this is the text
             (0,device_perc[0]), # this is the point to label
             textcoords="offset points", # how to position the text
             xytext=(0,10), # distance from text to points (x,y)
             ha='center', # horizontal alignment
             va = 'center') # vertica alignment
plt.annotate('{:.2f}'.format(device_perc[1])+'%', # this is the text
             (1,device_perc[1]), # this is the point to label
             textcoords="offset points", # how to position the text
             xytext=(0,10), # distance from text to points (x,y)
             ha='center', # horizontal alignment
             va = 'center') # vertica alignment
fig.tight_layout()
plt.show()
```



Graph representing the distribution of events on different days of a week

4.4 Graph representing the distribution of events on different days of a week

In [61]:
```
dfev.head(5)
```

Out[61]:

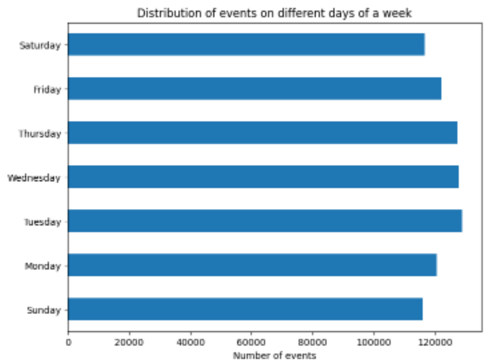
| | device_id | event_id | event_time | latitude | longitude | gender | age | group_train |
|---|---|---|---|---|---|---|---|---|
| 0 | -7548291590301750000 | 2279193 | 2016-05-03 03:10:01 | 116.79 | 33.98 | M | 33 | M32+ |
| 1 | -7548291590301750000 | 1121009 | 2016-05-03 15:37:40 | 116.79 | 33.98 | M | 33 | M32+ |
| 2 | -7548291590301750000 | 1121005 | 2016-05-03 15:33:51 | 116.79 | 33.98 | M | 33 | M32+ |
| 3 | -7548291590301750000 | 1113814 | 2016-05-01 10:27:52 | 116.79 | 33.98 | M | 33 | M32+ |
| 4 | -7548291590301750000 | 1113205 | 2016-05-04 09:10:43 | 116.79 | 33.98 | M | 33 | M32+ |

In [62]:
```
dfev['event_time']= pd.to_datetime(dfev['event_time'])
```

In [63]:
```
days_of_week = pd.to_datetime(dfev['event_time'], format='%Y-%m-%d').dt.day_name()
pd.Categorical(days_of_week, categories=['Sunday', 'Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'],
    ordered=True).value_counts()
```

Out[63]:
```
Sunday       115984
Monday       120617
Tuesday      128733
Wednesday    127601
Thursday     127299
Friday       122047
Saturday     116496
Name: count, dtype: int64
```

In [64]:
```
# Week days are categorised using pd.Categorical then sorted
#
plt.figure(figsize=(8,6))
days_of_week = pd.to_datetime(dfev['event_time'], format='%Y-%m-%d').dt.day_name()
pd.Categorical(days_of_week, categories=['Sunday', 'Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'],
    ordered=True).value_counts().plot(kind='barh')
plt.title('Distribution of events on different days of a week')
plt.xlabel('Number of events')
plt.show()
```

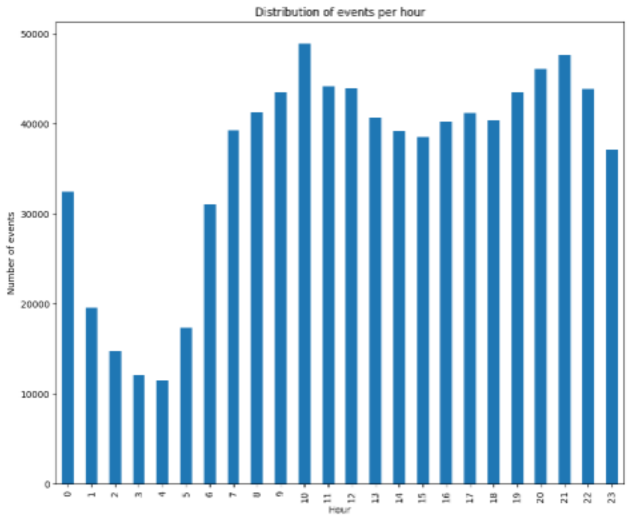



Graph representing the distribution of events per hour [ For one-week data]

Number of events

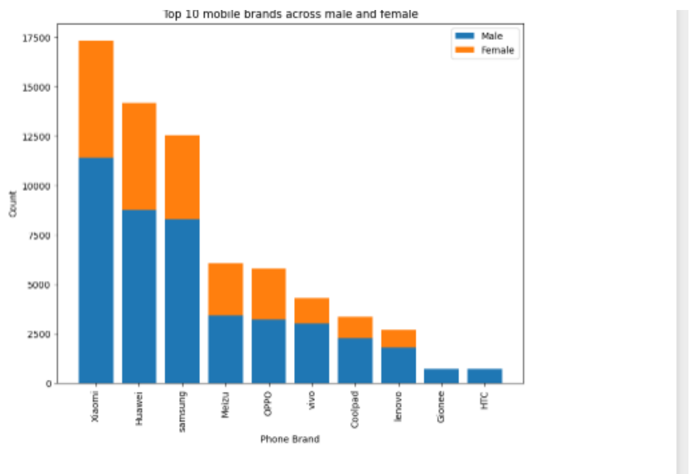
4.6 Graph representing the distribution of events per hour [ For one-week data]

[65]:
```
#whole dataset is covered, not just one-week data
plt.figure(figsize=(11,9))
event_hours = pd.to_datetime(dfev['event_time']).dt.hour
event_hours.value_counts().sort_index().plot(kind='bar')
plt.title('Distribution of events per hour')
plt.ylabel('Number of events')
plt.xlabel('Hour')
plt.show()
```

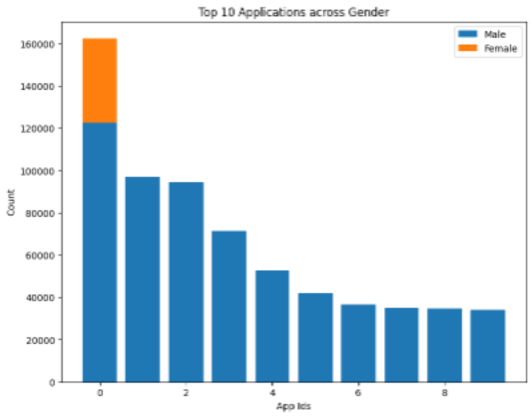



Is there any difference in the distribution of Events for different Age Groups over different days of the week? [Consider the age groups as 0-24, 25-32, 33-45, 46+]

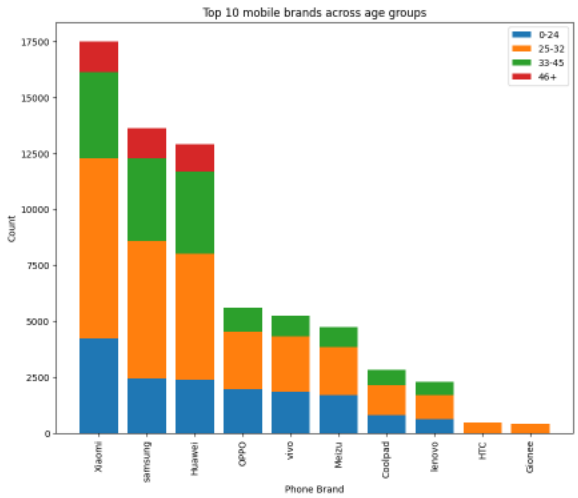

Stacked bar chart for top 10 mobile brands across male and female consumers




Stacked bar chart for top 10 application across male and female consumers




Stacked bar chart for top 10 mobile brands across age groups

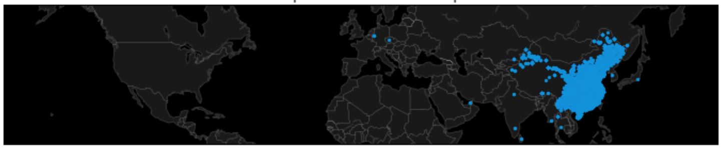



Geospatial visualisations along with the insights gathered from this visualisation

Plot the visualization plot for a sample of 1 lakh data points

In [9]:
```
fig = plt.figure(figsize=(12, 12))

# Mercator of World
m = Basemap(projection='merc',
             llcrnrlat=5,  #latitude of lower left hand corner of the desired map domain
             urcrnrlat=60, #latitude of upper right hand corner of the desired map domain
             llcrnrlon=-180, #longitude of lower left hand corner of the desired map domain
             urcrnrlon=180, #longitude of upper right hand corner of the desired map domain
             lat_ts=0, #latitude of true scale
             resolution='c') #resolution of boundary dataset being used - c for crude


m.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m.drawmapboundary(fill_color='#000000')                # black background
m.drawcoastlines(color='#333333')
m.drawcountries(linewidth=0.15, color="w")              # thin white line for country borders

# Plot the data
mxy = m(dfevsub['latitude'].tolist(), dfevsub['longitude'].tolist())
m.scatter(mxy[0], mxy[1], s=5, c="#1292db", zorder=2)    # zorder for the points

plt.title('Geospatial Events dataset Map')
plt.show()
```

Geospatial Events dataset Map

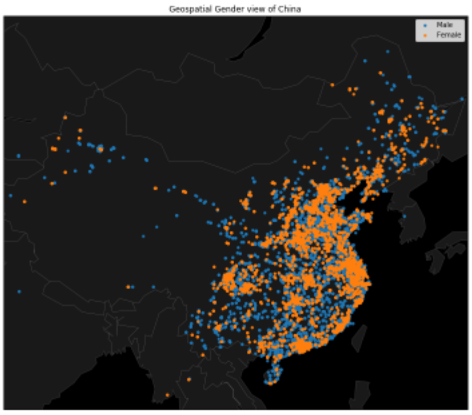


# Compare the event visualization plots based on the gender information [Can be done on the sample of 1 lakh points]

2. Compare the event visualization plots based on the gender information [Can be done on the sample of 1 lakh points]

In [10]:
```
#From the above mercator plot we observed most the plot is in china.  So lets plot only china demographic
#China latitude, longitude
#llcrnrlon, urcrnrlon = 75, 135
#llcrnrlat, urcrnrlat = 15, 55
fig, ax = plt.subplots(figsize=(10,10))


# Mercator of China
m = Basemap(projection='merc',
             llcrnrlat=15,  #latitude of lower left hand corner of the desired map domain
             urcrnrlat=55, #latitude of upper right hand corner of the desired map domain
             llcrnrlon=75, #longitude of lower left hand corner of the desired map domain
             urcrnrlon=135, #longitude of upper right hand corner of the desired map domain
             lat_ts=0, #latitude of true scale
             resolution='c',ax=ax) #resolution of boundary dataset being used - c for crude


m.fillcontinents(color='#191919',lake_color='#000000',ax=ax) # dark grey land, black lakes
m.drawmapboundary(fill_color='#000000',ax=ax)                # black background
m.drawcoastlines(color='#333333',ax=ax)
m.drawcountries(linewidth=0.15, color="w",ax=ax)              # thin white line for country borders

# Plot the data
#dataset for male
mxy = m(dfevsub[dfevsub['gender']=='M']['latitude'].tolist(), dfevsub[dfevsub['gender']=='M']['longitude'].tolist())
#dataset for female
fxy = m(dfevsub[dfevsub['gender']=='F']['latitude'].tolist(), dfevsub[dfevsub['gender']=='F']['longitude'].tolist())
#plot for male
ax.scatter(mxy[0], mxy[1], s=20, lw = 0, label='Male', cmap='viridis',  zorder=3)
#plot for female
ax.scatter(fxy[0], fxy[1], s=20, lw = 0, label='Female', cmap='viridis',  zorder=3)
ax.set_title('Geospatial Gender view of China')
ax.legend()
fig.tight_layout()
plt.show()
```

Geospatial Gender view of China

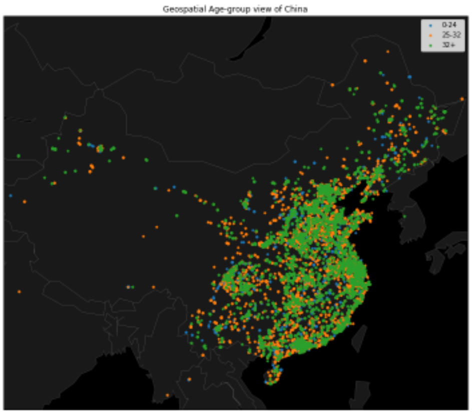


## Compare the events visualization plots based on the different age bands

3. Compare the events visualization plots based on the different age bands

In [11]:
```
fig, ax = plt.subplots(figsize=(10,10))

# Mercator of China
m = Basemap(projection='merc',
            llcrnrlat=15,  #Latitude of lower left hand corner of the desired map domain
            urcrnrlat=55, #Latitude of upper right hand corner of the desired map domain
            llcrnrlon=75, #Longitude of lower left hand corner of the desired map domain
            urcrnrlon=135, #Longitude of upper right hand corner of the desired map domain
            lat_ts=0, #Latitude of true scale
            resolution='c',ax=ax) #resolution of boundary dataset being used - c for crude


m.fillcontinents(color='#191919',lake_color='#000000',ax=ax) # dark grey land, black lakes
m.drawmapboundary(fill_color='#000000',ax=ax)                # black background
m.drawcoastlines(color='#333333',ax=ax)
m.drawcountries(linewidth=0.15, color="w",ax=ax)              # thin white line for country borders

#Dataset for Age group 0-24
g1xy = m(dfevsub[dfevsub['age_group']=='0-24']['latitude'].tolist(), dfevsub[dfevsub['age_group']=='0-24']['longitude'].tolist()
#Dataset for Age group 25-32
g2xy = m(dfevsub[dfevsub['age_group']=='25-32']['latitude'].tolist(), dfevsub[dfevsub['age_group']=='25-32']['longitude'].tolist
#Dataset for Age group 32+
g3xy = m(dfevsub[dfevsub['age_group']=='32+']['latitude'].tolist(), dfevsub[dfevsub['age_group']=='32+']['longitude'].tolist())

#plot for Age group 0-24
ax.scatter(g1xy[0], g1xy[1], s=15, lw = 0, label='0-24', cmap='viridis', zorder=3)
#plot for Age group 25-32
ax.scatter(g2xy[0], g2xy[1], s=15, lw = 0, label='25-32', cmap='viridis', zorder=3)
#plot for Age group 32+
ax.scatter(g3xy[0], g3xy[1], s=15, lw = 0, label='32+', cmap='viridis', zorder=3)
ax.set_title('Geospatial Age-group view of China')
ax.legend()
fig.tight_layout()
plt.show()
```

Geospatial Age-group view of China




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

Plot the distribution of latitude, longitude of event data

Load Event CSVs into Pandas Dataframe

In [2]:
```
#show the full content of the column
pd.set_option('display.max_colwidth', None)
```

In [3]:
```
#read latest events dataset into pandas dataframe
dfev = pd.read_csv('events_init2.csv', encoding='utf-8')
```

In [4]:
```
dfev.shape
```

Out[4]: (858777, 9)

In [5]:
```
#print 5 rows from dataframe
dfev.head(5)
```

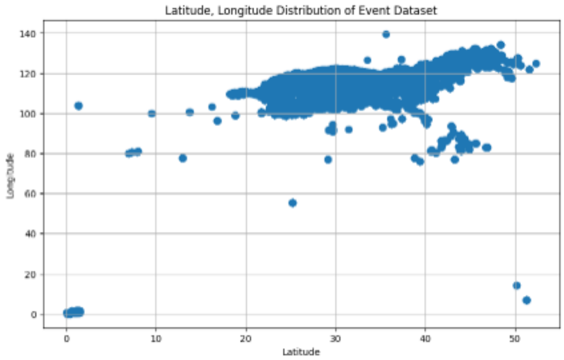
Out[5]:

| | device_id | event_id | event_time | latitude | longitude | gender | age | group_train | age_group |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -7548291590301750000 | 2279193 | 2016-05-03 03:10:01 | 116.79 | 33.98 | M | 33 | M32+ | 32+ |
| 1 | -7548291590301750000 | 1121009 | 2016-05-03 15:37:40 | 116.79 | 33.98 | M | 33 | M32+ | 32+ |
| 2 | -7548291590301750000 | 1121005 | 2016-05-03 15:33:51 | 116.79 | 33.98 | M | 33 | M32+ | 32+ |
| 3 | -7548291590301750000 | 1113814 | 2016-05-01 10:27:52 | 116.79 | 33.98 | M | 33 | M32+ | 32+ |
| 4 | -7548291590301750000 | 1113205 | 2016-05-04 09:10:43 | 116.79 | 33.98 | M | 33 | M32+ | 32+ |

Observation: From the below graph, the cluster formation will be within the range of

```
Latitude - 20 to 50
Longitude - 80 to 120
```

In [6]:
```
#plot the distribution of latitude, longitude of event data
plt.figure(figsize=(10, 6))
plt.scatter(dfev['longitude'],dfev['latitude'], s = 50)
plt.grid(True)
plt.title('Latitude, Longitude Distribution of Event Dataset')
plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.show()
```




# Plot the outlier cluster

```
2        13
Name: count, dtype: int64
```

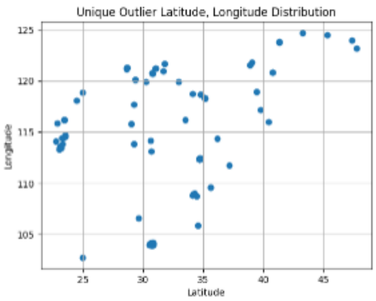
In [17]:
```
#cluster id -1 means outlier.  Check for outlier existance.
dfsub[dfsub['cluster_id'] == -1]
```

Out[17]:

| | longitude | latitude | cluster_id |
|---|---|---|---|
| 3718 | 30.85 | 104.19 | -1 |
| 3726 | 30.58 | 104.13 | -1 |
| 3739 | 30.87 | 103.98 | -1 |
| 3740 | 30.84 | 104.12 | -1 |
| 3825 | 30.88 | 104.02 | -1 |
| ... | ... | ... | ... |
| 858736 | 30.25 | 119.94 | -1 |
| 858748 | 47.77 | 123.19 | -1 |
| 858752 | 34.78 | 118.70 | -1 |
| 858762 | 30.79 | 120.69 | -1 |
| 858767 | 30.73 | 120.75 | -1 |

95 rows × 3 columns

In [18]:
```
#plot the outlier cluster
#.plot(kind='scatter', dfsub['Longitude'], dfsub['Latitude'])
x = dfsub[dfsub['cluster_id'] == -1]['longitude']
y = dfsub[dfsub['cluster_id'] == -1]['latitude']
plt.scatter(x, y)
plt.grid(True)
plt.title('Unique Outlier Latitude, Longitude Distribution')
plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.show()
```



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

Reduce app data

```
In [1]: import pandas as pd
```

```
In [2]: dfapp = pd.read_csv('appdata_init.csv', encoding='utf-8')
        dfapp.head(5)
```

Out[2]:

| | event_id | app_id | is_installed | is_active | label_id | category |
|---|---|---|---|---|---|---|
| 0 | 400185 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 1 | 2941538 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 2 | 34316 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 3 | 488229 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 4 | 1151851 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |

```
In [3]: dfapp.shape

Out[3]: (89825317, 6)
```

```
In [4]: dfapp.head(5)
```

Out[4]:

| | event_id | app_id | is_installed | is_active | label_id | category |
|---|---|---|---|---|---|---|
| 0 | 400185 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 1 | 2941538 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 2 | 34316 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 3 | 488229 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |
| 4 | 1151851 | 7324884708820027918 | 1.0 | 0.0 | 251.0 | Finance |

```
In [5]: #filter only installed & active app records
        dfapp = dfapp.loc[((dfapp['is_installed'] == 1) & (dfapp['is_active'] == 1))]
        dfapp.shape

Out[5]: (33917018, 6)
```

```
In [6]: #save reduced appdata dataframe to csv
        dfapp.to_csv(r'appdata_init2.csv', index = False)
```

```
In [ ]: #upload appdata_init2 csv to S3
        !aws s3 cp appdata_init2.csv s3://capstone-mm/final_submission/appdata_init2.csv

        Completed 8.0 MiB/1.8 GiB (30.4 MiB/s) with 1 file(s) remaining
```

### A brief summary of any additional subtask that was performed and may have improved the data cleaning and feature generation step & All the data preparation steps that were used before applying the ML algorithm

## Data Preparation

```
In [11]: # Merging the tables event and non-event for Scenario 1 (all information)
         df = pd.merge(event,non_event,on="device_id",how="left")
```

```
In [12]: df.head()
```

Out[12]:

| | device_id | event_id | event_time | latitude | longitude | gender_x | age_x | group_train_x | age_group_x | cluster_id | phone_brand | device_model | gend |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -7548291590301750000 | 2279193 | 2016-05-03 03:10:01 | 116.79 | 33.98 | M | 33 | M32+ | 32+ | 0 | Huawei | 3C | |
| 1 | -7548291590301750000 | 1121009 | 2016-05-03 15:37:40 | 116.79 | 33.98 | M | 33 | M32+ | 32+ | 0 | Huawei | 3C | |
| 2 | -7548291590301750000 | 1121005 | 2016-05-03 15:33:51 | 116.79 | 33.98 | M | 33 | M32+ | 32+ | 0 | Huawei | 3C | |
| 3 | -7548291590301750000 | 1113814 | 2016-05-01 10:27:52 | 116.79 | 33.98 | M | 33 | M32+ | 32+ | 0 | Huawei | 3C | |
| 4 | -7548291590301750000 | 1113205 | 2016-05-04 09:10:43 | 116.79 | 33.98 | M | 33 | M32+ | 32+ | 0 | Huawei | 3C | |

In [16]:
```python
# Merging the tables df and train_test
df = pd.merge(df,train_test,on="device_id",how="left")
```

In [17]:
```python
df.head()
```

Out[17]:

| | device_id | event_id | event_time | latitude | longitude | gender_x | age_x | group_train_x | age_group_x | cluster_id | phone_brand | device_model | gend |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4676982795249940000 | 2958924 | 2016-05-01 22:03:46 | 113.24 | 22.85 | M | 52 | M32+ | 32+ | 0.0 | Huawei | Ascend P8 | |
| 1 | 4782582047729160000 | 2958931 | 2016-05-01 22:24:09 | 114.47 | 38.03 | M | 36 | M32+ | 32+ | 0.0 | HTC | One | |
| 2 | 2181284491650730000 | 2958933 | 2016-05-01 22:24:44 | 125.66 | 43.02 | M | 28 | M25-32 | 25-32 | 0.0 | vivo | X6 D | |
| 3 | 4221762657972680000 | 2958935 | 2016-05-01 22:40:47 | 115.19 | 24.07 | F | 19 | F0-24 | 0-24 | 0.0 | OPPO | R7 | |
| 4 | -6242501228649110000 | 2958939 | 2016-05-01 21:59:40 | 111.21 | 27.85 | M | 20 | M0-24 | 0-24 | 0.0 | OPPO | R3 | |

In [18]:
```python
df.columns
```

Out[18]:
```
Index(['device_id', 'event_id', 'event_time', 'latitude', 'longitude',
       'gender_x', 'age_x', 'group_train_x', 'age_group_x', 'cluster_id',
       'phone_brand', 'device_model', 'gender_y', 'age_y', 'group_train_y',
       'age_group_y', 'gender', 'age', 'group', 'train_test_flag'],
      dtype='object')
```

In [19]:
```python
# Dropping irrelevant columns
df=df.drop(columns=["gender_y","age_y","age_x","age_group_y","group_train_y","group_train_x","gender","age","group"],axis=1)
```

In [20]:
```python
# Renaming columns gender, age and group_train
df.rename(columns={"gender_x":"gender","age_group_x":"age_group"},inplace=True)
```

In [21]:
```python
df.head()
```

Out[21]:

| | device_id | event_id | event_time | latitude | longitude | gender | age_group | cluster_id | phone_brand | device_model | train_test_flag |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4676982795249940000 | 2958924 | 2016-05-01 22:03:46 | 113.24 | 22.85 | M | 32+ | 0.0 | Huawei | Ascend P8 | train |
| 1 | 4782582047729160000 | 2958931 | 2016-05-01 22:24:09 | 114.47 | 38.03 | M | 32+ | 0.0 | HTC | One | test |
| 2 | 2181284491650730000 | 2958933 | 2016-05-01 22:24:44 | 125.66 | 43.02 | M | 25-32 | 0.0 | vivo | X6 D | test |
| 3 | 4221762657972680000 | 2958935 | 2016-05-01 22:40:47 | 115.19 | 24.07 | F | 0-24 | 0.0 | OPPO | R7 | test |
| 4 | -6242501228649110000 | 2958939 | 2016-05-01 21:59:40 | 111.21 | 27.85 | M | 0-24 | 0.0 | OPPO | R3 | train |

In [22]:
```python
df.count()
```

Out[22]:
```
device_id          862364
event_id           862364
event_time         862364
latitude           862364
longitude          862364
gender             862364
age_group          862364
cluster_id         862364
phone_brand        862364
device_model       862364
train_test_flag    862364
dtype: int64
```

In [23]:
```
# Data cleaning part already took care of null values
df.isnull().sum()
```

Out[23]:
```
device_id          0
event_id           0
event_time         0
latitude           0
longitude          0
gender             0
age_group          0
cluster_id         0
phone_brand        0
device_model       0
train_test_flag    0
dtype: int64
```

In [24]:
```
# Merging app data and labels data
app_label = pd.merge(app,label,on=("label_id"))
app_label.head()
```

Out[24]:

| | event_id | app_id | is_installed | is_active | label_id | category_x | category_y |
|---|---|---|---|---|---|---|---|
| 0 | 2960967 | 8557198901083791098 | 1.0 | 1.0 | 548.0 | Industry tag | Industry tag |
| 1 | 2960967 | 5300107820801348133 | 1.0 | 1.0 | 548.0 | Industry tag | Industry tag |
| 2 | 2960967 | 4348659952760821294 | 1.0 | 1.0 | 548.0 | Industry tag | Industry tag |
| 3 | 2960967 | -6793861127573349654 | 1.0 | 1.0 | 548.0 | Industry tag | Industry tag |
| 4 | 2960969 | -5472633337921616096 | 1.0 | 1.0 | 548.0 | Industry tag | Industry tag |

In [25]:
```
# Removing extra category column
app_label = app_label.drop(columns=["category_y"],axis=1)
app_label.rename(columns={"category_x":"category"},inplace=True)
```

In [27]:
```
# Dropping any duplicates
app_label = app_label.drop_duplicates()
```

In [28]:
```
app_label.count()
```

Out[28]:
```
event_id        18771672
app_id          18771672
is_installed    18771672
is_active       18771672
label_id        18771672
category        18771672
dtype: int64
```

In [29]:
```
app_label[app_label.duplicated(["app_id"])]
```

Out[29]:

| | event_id | app_id | is_installed | is_active | label_id | category |
|---|---|---|---|---|---|---|
| 17 | 2960977 | 5927333115845830913 | 1.0 | 1.0 | 548.0 | Industry tag |
| 19 | 2960977 | -5472633337921616096 | 1.0 | 1.0 | 548.0 | Industry tag |
| 20 | 2960977 | 8693964245073640147 | 1.0 | 1.0 | 548.0 | Industry tag |
| 24 | 2960977 | 4348659952760821294 | 1.0 | 1.0 | 548.0 | Industry tag |
| 29 | 2960978 | 5927333115845830913 | 1.0 | 1.0 | 548.0 | Industry tag |
| ... | ... | ... | ... | ... | ... | ... |
| 18916650 | 190501 | 6808240730269550030 | 1.0 | 1.0 | 944.0 | Southeast Asia (Travel) |
| 18916651 | 452329 | 6808240730269550030 | 1.0 | 1.0 | 944.0 | Southeast Asia (Travel) |
| 18916652 | 548328 | 6808240730269550030 | 1.0 | 1.0 | 944.0 | Southeast Asia (Travel) |
| 18916654 | 264311 | -7995837913973148847 | 1.0 | 1.0 | 762.0 | Trust |
| 18916655 | 264311 | -7995837913973148847 | 1.0 | 1.0 | 763.0 | Trust funds |

18763342 rows × 6 columns

```
In [32]: # Removing the digits from categories for feature engineering
import re
def digits_elimination(string):
    pattern = r'[0-9]'
    new_string = re.sub(pattern, '', string)
    pattern=r'[^\x00-\x7F]+'
    new_string = re.sub(pattern, '', new_string)
    pattern ="\."
    new_string = re.sub(pattern, '', new_string)
    return new_string
```

```
In [33]: app_label["category"]=app_label["category"].apply(digits_elimination)
app_label["category"].value_counts()
```

```
Out[33]: Property Industry     3424440
Industry tag          2609857
unknown               1226010
Services               820260
Custom label           619480
                        ...
game-Rowing                 1
Trust                       1
Trust funds                 1
reality show                1
Educational games           1
Name: category, Length: 428, dtype: int64
```

## Feature Engineering

### A feature called "total_events" can be added here based on the total number of events for each device id

```
In [53]: event.groupby(["device_id"])["event_id"].count()
```

```
Out[53]: device_id
-9222956879900150000     52
-9221026417907250000    132
-9220061629197650000     39
-9218769147970100000     17
-9215352913819630000     18
                       ...
 9215085115859650000     17
 9216925254504440000     73
 9219164468944550000    407
 9219842210460030000      4
 9220914901466450000      3
Name: event_id, Length: 11949, dtype: int64
```

```
In [54]: df=pd.DataFrame(event.groupby(["device_id"])["event_id"].count())
df.rename(columns={"event_id":"total_events"},inplace=True)
df.head()
```

Out[54]:

| device_id | total_events |
|---|---|
| -9222956879900150000 | 52 |
| -9221026417907250000 | 132 |
| -9220061629197650000 | 39 |
| -9218769147970100000 | 17 |
| -9215352913819630000 | 18 |

### Adding a feature latitude median and longitude median

In [62]:
```
# Taking the median of the latitudes and longitudes corresponding to the different device ids
dff = final_df.groupby(["device_id"]).aggregate({"longitude":['median'],"latitude":['median']}).droplevel(1,axis=1).reset_index()
```

In [63]:
```
# Renaming median values of long and lat and removing the irrelevant columns
dff.rename(columns={"longitude":"longitude_median","latitude":"latitude_median"},inplace=True)
final_df=pd.merge(final_df,dff,on=("device_id"),how="left")
```

In [64]:
```
final_df.head()
```

Out[64]:

| | device_id | event_id | event_time | latitude | longitude | gender | cluster_id | phone_brand | train_test_flag | label_id | category | age_group | total_events |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -2582852392726770000 | 2961008 | 2016-05-03 23:09:55 | 104.07 | 30.65 | F | 1.0 | samsung | train | 548.0 | Industry tag | 32+ | 303 |
| 1 | -2582852392726770000 | 2961008 | 2016-05-03 23:09:55 | 104.07 | 30.65 | F | 1.0 | samsung | train | 262.0 | Pay | 32+ | 303 |
| 2 | -2582852392726770000 | 2961008 | 2016-05-03 23:09:55 | 104.07 | 30.65 | F | 1.0 | samsung | train | 251.0 | Finance | 25-32 | 303 |
| 3 | -2582852392726770000 | 2961008 | 2016-05-03 23:09:55 | 104.07 | 30.65 | F | 1.0 | samsung | train | 1008.0 | Pay | 0-24 | 303 |
| 4 | -2582852392726770000 | 2961008 | 2016-05-03 23:09:55 | 104.07 | 30.65 | F | 1.0 | samsung | train | 786.0 | Low risk | 0-24 | 303 |

### Adding the day of the week

In [65]:
```
import datetime as dt
final_df["event_time"] = pd.to_datetime(final_df["event_time"])
final_df.dtypes
```

Out[65]:
```
device_id                    int64
event_id                     int64
event_time          datetime64[ns]
latitude                   float64
longitude                  float64
gender                      object
cluster_id                 float64
phone_brand                 object
train_test_flag             object
label_id                   float64
category                    object
age_group                   object
total_events                 int64
longitude_median           float64
latitude_median            float64
dtype: object
```

In [66]:
```
final_df["week_day"]=final_df["event_time"].dt.dayofweek
final_df["week_day"].value_counts()
```

Out[66]:
```
1    402895
3    388843
4    381691
2    381201
0    364967
5    356250
6    352728
Name: week_day, dtype: int64
```

```
Name: week_day, dtype: int64
```

**Adding a feature called hour from the event_time**

In [67]:
```python
final_df["hour"]=final_df["event_time"].dt.hour
final_df["event_time"].dt.week.value_counts()
```

```
/home/ec2-user/.local/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: Series.dt.weekofyear and Series.dt.we
ek have been deprecated.  Please use Series.dt.isocalendar().week instead.
```

Out[67]:
```
18    2275209
17     353366
Name: event_time, dtype: int64
```

In [68]:
```python
final_df.dtypes
```

Out[68]:
```
device_id                    int64
event_id                     int64
event_time          datetime64[ns]
latitude                   float64
longitude                  float64
gender                      object
cluster_id                 float64
phone_brand                 object
train_test_flag             object
label_id                   float64
category                    object
age_group                   object
total_events                 int64
longitude_median           float64
latitude_median            float64
week_day                     int64
hour                         int64
dtype: object
```

In [69]:
```python
final_df["hour"].value_counts().sort_values()
```

## Dropping irrelevant columns

In [71]:
```python
#Dropping irrelevant columns
final_df.drop(columns=["event_id","event_time","latitude","longitude"],inplace=True)
final_df.head()
```

Out[71]:

| | device_id | gender | cluster_id | phone_brand | train_test_flag | label_id | category | age_group | total_events | longitude_median | latitude_median | week |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -2582852392726770000 | F | 1.0 | samsung | train | 548.0 | Industry tag | 32+ | 303 | 30.65 | 104.08 | |
| 1 | -2582852392726770000 | F | 1.0 | samsung | train | 262.0 | Pay | 32+ | 303 | 30.65 | 104.08 | |
| 2 | -2582852392726770000 | F | 1.0 | samsung | train | 251.0 | Finance | 25-32 | 303 | 30.65 | 104.08 | |
| 3 | -2582852392726770000 | F | 1.0 | samsung | train | 1008.0 | Pay | 0-24 | 303 | 30.65 | 104.08 | |
| 4 | -2582852392726770000 | F | 1.0 | samsung | train | 786.0 | Low risk | 0-24 | 303 | 30.65 | 104.08 | |

In [72]:
```python
final_df.count()
```

Out[72]:
```
device_id           2628575
gender              2628575
cluster_id          2628575
phone_brand         2628575
train_test_flag     2628575
label_id            2628575
category            2628575
age_group            862364
total_events        2628575
longitude_median    2628575
latitude_median     2628575
week_day            2628575
hour                2628575
dtype: int64
```

**Reducing the number of categories to final categories**

## Reducing the number of categories to final categories

### Reducing the number of categories to final categories

In [73]:
```
final_df['category']
```

Out[73]:
```
0                     Industry tag
1                              Pay
2                          Finance
3                              Pay
4                         Low risk
                        ...
2628570    Personal Effectiveness 1
2628571                    unknown
2628572                    unknown
2628573               Industry tag
2628574       Property Industry 2.0
Name: category, Length: 2628575, dtype: object
```

In [74]:
```
# Finding differences or “diff” between contents of files or other hashable Python objects, which is category in this case
import difflib
```

In [75]:
```
def final_category(word):
    seq=[]
    Categories_of_Apps=["Industry","property","finance","games","health","church","Entertainment","Custom label","tencent","Buy",
    for a in Categories_of_Apps:
        temp = difflib.SequenceMatcher(None,a,word)
        d = temp.ratio()*100
        seq.append(d)
    word1 = Categories_of_Apps[seq.index(max(seq))]
    return word1
```

In [77]:
```
final_df["final_category"]=final_df["category"].apply(final_category)
```

In [78]:
```
final_df.head(15)
```

Out[78]:

| | device_id | gender | cluster_id | phone_brand | train_test_flag | label_id | category | age_group | total_events | longitude_median | latitude_median | we |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -2582852392726770000 | F | 1.0 | samsung | train | 548.0 | Industry tag | 32+ | 303 | 30.65 | 104.08 | |
| 1 | -2582852392726770000 | F | 1.0 | samsung | train | 262.0 | Pay | 32+ | 303 | 30.65 | 104.08 | |
| 2 | -2582852392726770000 | F | 1.0 | samsung | train | 251.0 | Finance | 25-32 | 303 | 30.65 | 104.08 | |
| 3 | -2582852392726770000 | F | 1.0 | samsung | train | 1008.0 | Pay | 0-24 | 303 | 30.65 | 104.08 | |
| 4 | -2582852392726770000 | F | 1.0 | samsung | train | 786.0 | Low risk | 0-24 | 303 | 30.65 | 104.08 | |
| 5 | -2582852392726770000 | F | 1.0 | samsung | train | 780.0 | Moderate profitability | 32+ | 303 | 30.65 | 104.08 | |
| 6 | -2582852392726770000 | F | 1.0 | samsung | train | 773.0 | High Flow | 32+ | 303 | 30.65 | 104.08 | |

## Model Building

## Model Building

### Scenario 1( all information ) will be implemented using final_df dataset

In [153]:
```
final_df = pd.read_csv("final_df.csv")
final_df.head()
```

Out[153]:

| | device_id | gender | cluster_id | phone_brand | train_test_flag | label_id | age_group | total_events | longitude_median | latitude_median | week_day | hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -2582852392726770000 | F | 1.0 | samsung | train | 548.0 | 32+ | 303 | 30.65 | 104.08 | 1 | 2 |
| 1 | -2582852392726770000 | F | 1.0 | samsung | train | 262.0 | 32+ | 303 | 30.65 | 104.08 | 1 | 2 |
| 2 | -2582852392726770000 | F | 1.0 | samsung | train | 251.0 | 25-32 | 303 | 30.65 | 104.08 | 1 | 2 |
| 3 | -2582852392726770000 | F | 1.0 | samsung | train | 1008.0 | 0-24 | 303 | 30.65 | 104.08 | 1 | 2 |
| 4 | -2582852392726770000 | F | 1.0 | samsung | train | 786.0 | 0-24 | 303 | 30.65 | 104.08 | 1 | 2 |

In [158]:
```
final_df.isnull()
```

Out[158]:

| | device_id | gender | cluster_id | phone_brand | train_test_flag | label_id | age_group | total_events | longitude_median | latitude_median | week_day | hour | fin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False | False | False | False | False | False | False | |
| 1 | False | False | False | False | False | False | False | False | False | False | False | False | |
| 2 | False | False | False | False | False | False | False | False | False | False | False | False | |
| 3 | False | False | False | False | False | False | False | False | False | False | False | False | |
| 4 | False | False | False | False | False | False | False | False | False | False | False | False | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 2628570 | False | False | False | False | False | False | True | False | False | False | False | False | |
| 2628571 | False | False | False | False | False | False | True | False | False | False | False | False | |
| 2628572 | False | False | False | False | False | False | True | False | False | False | False | False | |
| 2628573 | False | False | False | False | False | False | True | False | False | False | False | False | |
| 2628574 | False | False | False | False | False | False | True | False | False | False | False | False | |

2628575 rows × 13 columns

## Predicting Gender for Scenario 1

## Predicting Gender for Scenario 1

In [183]:
```
#Converting categorial columns to dummy variables
categorical_columns = ['week_day','hour','age_group','category_code','phone_brands_code']
pd.get_dummies(data=final_df, columns=categorical_columns, drop_first=True).columns
```

Out[183]:
```
Index(['device_id', 'gender', 'cluster_id', 'phone_brand', 'train_test_flag',
       'label_id', 'total_events', 'longitude_median', 'latitude_median',
       'final_category', 'week_day_1', 'week_day_2', 'week_day_3',
       'week_day_4', 'week_day_5', 'week_day_6', 'hour_1', 'hour_2', 'hour_3',
       'hour_4', 'hour_5', 'hour_6', 'hour_7', 'hour_8', 'hour_9', 'hour_10',
       'hour_11', 'hour_12', 'hour_13', 'hour_14', 'hour_15', 'hour_16',
       'hour_17', 'hour_18', 'hour_19', 'hour_20', 'hour_21', 'hour_22',
       'hour_23', 'age_group_25-32', 'age_group_32+', 'category_code_1',
       'category_code_2', 'category_code_3', 'category_code_4',
       'category_code_5', 'category_code_6', 'category_code_7',
       'category_code_8', 'category_code_9', 'category_code_10',
       'phone_brands_code_1', 'phone_brands_code_2', 'phone_brands_code_3',
       'phone_brands_code_4', 'phone_brands_code_5', 'phone_brands_code_6',
       'phone_brands_code_7', 'phone_brands_code_8', 'phone_brands_code_9',
       'phone_brands_code_10', 'phone_brands_code_11', 'phone_brands_code_12',
       'phone_brands_code_13', 'phone_brands_code_14', 'phone_brands_code_15',
       'phone_brands_code_16', 'phone_brands_code_17', 'phone_brands_code_18',
       'phone_brands_code_19', 'phone_brands_code_20', 'phone_brands_code_21',
       'phone_brands_code_22', 'phone_brands_code_23', 'phone_brands_code_24',
       'phone_brands_code_25', 'phone_brands_code_26', 'phone_brands_code_27',
       'phone_brands_code_28', 'phone_brands_code_29', 'phone_brands_code_30',
       'phone_brands_code_31', 'phone_brands_code_32', 'phone_brands_code_33',
       'phone_brands_code_34', 'phone_brands_code_35', 'phone_brands_code_36',
       'phone_brands_code_37', 'phone_brands_code_38'],
      dtype='object')
```

In [184]:
```
final_df=pd.get_dummies(data=final_df, columns=categorical_columns, drop_first=True)
```

In [185]:
```
#Test data prep
final_df["gender"].value_counts()
final_df["gender"]=final_df["gender"].apply(lambda x : 0 if x=='F' else 1)
final_df["gender"].value_counts()
```

Out[185]:
```
1    648530
0    213834
Name: gender, dtype: int64
```

In [186]:
```
df_train=final_df[final_df["train_test_flag"]=="train"]
df_train["gender"].value_counts()
```

Out[186]:
```
1    482523
0    160710
Name: gender, dtype: int64
```

In [187]:
```
df_test=final_df[final_df["train_test_flag"]=="test"]
df_test["gender"].value_counts()
```

Out[187]:
```
1    166007
0     53124
Name: gender, dtype: int64
```

In [188]:
```
y_train=df_train["gender"]
y_test=df_test["gender"]
```

In [189]:
```
y_train.value_counts()
```

Out[189]:
```
1    482523
0    160710
Name: gender, dtype: int64
```

In [190]:
```
x_train=df_train.drop(columns=["gender","final_category","train_test_flag","device_id","phone_brand"])
x_test=df_test.drop(columns=["gender","final_category","train_test_flag","device_id","phone_brand"])
```

In [191]:
```
x_train.columns
```

Out[191]:
```
Index(['cluster_id', 'label_id', 'total_events', 'longitude_median',
       'latitude_median', 'week_day_1', 'week_day_2', 'week_day_3',
       'week_day_4', 'week_day_5', 'week_day_6', 'hour_1', 'hour_2', 'hour_3',
       'hour_4', 'hour_5', 'hour_6', 'hour_7', 'hour_8', 'hour_9', 'hour_10',
       'hour_11', 'hour_12', 'hour_13', 'hour_14', 'hour_15', 'hour_16',
       'hour_17', 'hour_18', 'hour_19', 'hour_20', 'hour_21', 'hour_22',
       'hour_23', 'age_group_25-32', 'age_group_32+', 'category_code_1',
       'category_code_2', 'category_code_3', 'category_code_4',
       'category_code_5', 'category_code_6', 'category_code_7',
       'category_code_8', 'category_code_9', 'category_code_10',
       'phone_brands_code_1', 'phone_brands_code_2', 'phone_brands_code_3',
       'phone_brands_code_4', 'phone_brands_code_5', 'phone_brands_code_6',
       'phone_brands_code_7', 'phone_brands_code_8', 'phone_brands_code_9',
       'phone_brands_code_10', 'phone_brands_code_11', 'phone_brands_code_12',
       'phone_brands_code_13', 'phone_brands_code_14', 'phone_brands_code_15',
       'phone_brands_code_16', 'phone_brands_code_17', 'phone_brands_code_18',
       'phone_brands_code_19', 'phone_brands_code_20', 'phone_brands_code_21',
       'phone_brands_code_22', 'phone_brands_code_23', 'phone_brands_code_24',
       'phone_brands_code_25', 'phone_brands_code_26', 'phone_brands_code_27',
       'phone_brands_code_28', 'phone_brands_code_29', 'phone_brands_code_30',
       'phone_brands_code_31', 'phone_brands_code_32', 'phone_brands_code_33',
       'phone_brands_code_34', 'phone_brands_code_35', 'phone_brands_code_36',
       'phone_brands_code_37', 'phone_brands_code_38'],
      dtype='object')
```

In [192]:
```
x_train.head()
```

Out[192]:

| | cluster_id | label_id | total_events | longitude_median | latitude_median | week_day_1 | week_day_2 | week_day_3 | week_day_4 | week_day_5 | ... | phone_brands_coc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 548.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 1 | 1.0 | 262.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 2 | 1.0 | 251.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 3 | 1.0 | 1008.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 4 | 1.0 | 786.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |

5 rows × 84 columns

In [193]:
```
from sklearn.preprocessing import StandardScaler
x_train["total_events"]=StandardScaler().fit_transform(x_train[["total_events"]])
```

In [194]:
```
x_train["longitude_median"]=StandardScaler().fit_transform(x_train[["longitude_median"]])
x_train["latitude_median"]=StandardScaler().fit_transform(x_train[["latitude_median"]])
```

In [195]:
```
x_test["total_events"]=StandardScaler().fit_transform(x_test[["total_events"]])
x_test["longitude_median"]=StandardScaler().fit_transform(x_test[["longitude_median"]])
x_test["latitude_median"]=StandardScaler().fit_transform(x_test[["latitude_median"]])
```

In [196]:
```
x_train
```

Out[196]:

| | cluster_id | label_id | total_events | longitude_median | latitude_median | week_day_1 | week_day_2 | week_day_3 | week_day_4 | week_day_5 | ... | phone_brar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 548.0 | -0.041834 | 0.043053 | -0.266312 | 1 | 0 | 0 | 0 | 0 | ... | |
| 1 | 1.0 | 262.0 | -0.041834 | 0.043053 | -0.266312 | 1 | 0 | 0 | 0 | 0 | ... | |
| 2 | 1.0 | 251.0 | -0.041834 | 0.043053 | -0.266312 | 1 | 0 | 0 | 0 | 0 | ... | |
| 3 | 1.0 | 1008.0 | -0.041834 | 0.043053 | -0.266312 | 1 | 0 | 0 | 0 | 0 | ... | |
| 4 | 1.0 | 786.0 | -0.041834 | 0.043053 | -0.266312 | 1 | 0 | 0 | 0 | 0 | ... | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2429467 | 1.0 | 235.0 | -0.765311 | 1.155613 | 0.257222 | 0 | 0 | 0 | 0 | 0 | ... | |
| 2429468 | 1.0 | 129.0 | -0.765311 | 1.155613 | 0.257222 | 0 | 0 | 0 | 0 | 0 | ... | |
| 2429469 | 1.0 | 548.0 | -0.765311 | 1.155613 | 0.257222 | 0 | 0 | 0 | 0 | 0 | ... | |
| 2429470 | 1.0 | 168.0 | -0.765311 | 1.155613 | 0.257222 | 0 | 0 | 0 | 0 | 0 | ... | |
| 2429471 | 1.0 | 549.0 | -0.765311 | 1.155613 | 0.257222 | 0 | 0 | 0 | 0 | 0 | ... | |

643233 rows × 84 columns

In [197]:
```
len(x_train.columns)
```

Out[197]:
```
84
```

In [198]:
```
len(x_test.columns)
```

Out[198]:
```
84
```

In [199]:
```
y_train.value_counts()
```

Out[199]:
```
1    482523
0    160710
Name: gender, dtype: int64
```

## Hyperparameter tuning

Hyperparameter tuning

In [227]:
```
logic1 = LogisticRegression()
logic2 = RandomForestClassifier(random_state=1)
xgb = XGBClassifier()

stacking_Classifier = StackingCVClassifier(classifiers=[logic1, logic2], meta_classifier=xgb, use_probas=True, cv=3)
```

In [228]:
```
# A parameter grid for XGBoost
params = {
          'randomforestclassifier__max_depth': [2,5, 10],
          'randomforestclassifier__n_estimators': [5,10,15]
         }
```

In [229]:
```
grid = GridSearchCV(estimator=stacking_Classifier,
                    param_grid=params,
                    cv=5,
                    refit=True)
```

In [230]:
```
for clf, label in zip([logic1, logic2, stacking_Classifier],
                      ['lr',
                       'Random Forest',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, x_train.values, y_train.values, cv=3, scoring='roc_auc')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

```
Accuracy: 0.53 (+/- 0.03) [lr]
Accuracy: 0.51 (+/- 0.02) [Random Forest]
[15:48:32] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'b
inary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[15:52:22] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'b
inary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[15:55:43] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'b
inary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy: nan (+/- nan) [StackingClassifier]
```

In [231]:
```
grid.fit(x_train.values,y_train.values)
```

```
e 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
[16:30:36] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objectiv
e 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
[16:31:40] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objectiv
e 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
[16:32:44] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objectiv
e 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
[16:33:47] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objectiv
e 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old beha
vior.
```

# Best model estimator

In [235]:
```
best_model_gender=grid.best_estimator_
```

In [236]:
```
# Saving the model as pickle file
import pickle
filename = 'gender_pred.sav'
pickle.dump(best_model_gender, open(filename, 'wb'))
```

In [239]:
```
# Uploading the model in s3 bucket
!aws s3 cp gender_pred.sav s3://s3accessec2/final_submission/gender_pred.sav .
```

```
Unknown options: .
```

In [240]:
```
y_pred=best_model_gender.predict(x_test.values)
```

In [241]:
```
y_pred
```

Out[241]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [242]:
```
y_probabilites=best_model_gender.predict_proba(x_test)
```

# Model Evaluation

### Model evaluation

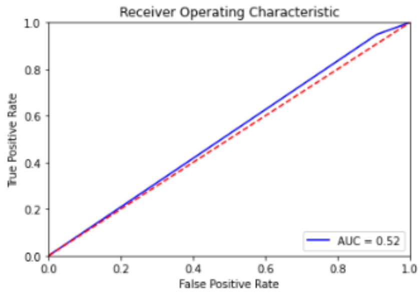
```
:n [243]: Statistic evaluation
          s(data=None,target=None, prob=None):
          ata['target0'] = 1 - data[target]
          ata['bucket'] = pd.qcut(data[prob], 10)
          rouped = data.groupby('bucket', as_index = False)
          stable = pd.DataFrame()
          stable['min_prob'] = grouped.min()[prob]
          stable['max_prob'] = grouped.max()[prob]
          stable['events']   = grouped.sum()[target]
          stable['nonevents'] = grouped.sum()['target0']
          stable = kstable.sort_values(by="min_prob", ascending=False).reset_index(drop = True)
          stable['event_rate'] = (kstable.events / data[target].sum()).apply('{0:.2%}'.format)
          stable['nonevent_rate'] = (kstable.nonevents / data['target0'].sum()).apply('{0:.2%}'.format)
          stable['cum_eventrate']=(kstable.events / data[target].sum()).cumsum()
          stable['cum_noneventrate']=(kstable.nonevents / data['target0'].sum()).cumsum()
          stable['KS'] = np.round(kstable['cum_eventrate']-kstable['cum_noneventrate'], 3) * 100

          Formatting
          stable['cum_eventrate']= kstable['cum_eventrate'].apply('{0:.2%}'.format)
          stable['cum_noneventrate']= kstable['cum_noneventrate'].apply('{0:.2%}'.format)
          stable.index = range(1,11)
          stable.index.rename('Decile', inplace=True)
          d.set_option('display.max_columns', 9)
          rint(kstable)

          Display KS
          rom colorama import Fore
          rint(Fore.RED + "KS is " + str(max(kstable['KS']))+"%"+ " at decile " + str((kstable.index[kstable['KS']==max(kstable['KS'])][0]))
          eturn(kstable)
```

```
:n [244]: y_probabilites[:, 1]

Out[244]: array([0.82052237, 0.81744003, 0.82052237, ..., 0.6761523 , 0.721873  ,
                 0.721873  ], dtype=float32)
```

```
:n [245]: ks_df = pd.DataFrame(y_test)
          ks_df["prob"]=y_probabilites[:, 1]
          ks_df.rename(columns={"gender":"target"},inplace=True)
```

```
:n [246]: ks_df.count

Out[246]: <bound method DataFrame.count of          target      prob
          9221          1  0.820522
          9222          1  0.817440
          9223          1  0.820522
          9224          1  0.818288
          9225          1  0.768948
          ...         ...       ...
          2428818       1  0.721873
          2428819       1  0.710298
          2428820       1  0.676152
          2428821       1  0.721873
```

```
			Model Evaluation for scenario 1 (All data Present)

confusion matrix
 [[  4843  48281]
 [  8498 157509]]


Accuracy : 0.7408901524658765


Classification_Report:

              precision    recall  f1-score   support

           0       0.36      0.09      0.15     53124
           1       0.77      0.95      0.85    166007

    accuracy                           0.74    219131
   macro avg       0.56      0.52      0.50    219131
weighted avg       0.67      0.74      0.68    219131
```

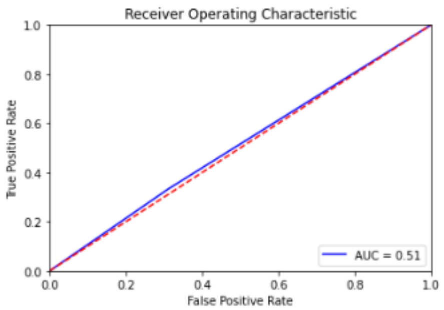



In [254]:
```
# Applying KS statistic
y_pred=[1 if x >0.838421 else 0 for x in y_probabilites[:, 1]]
```

In [255]:
```
conf_mat=confusion_matrix(y_test, y_pred)
accuracy_=accuracy_score(y_test, y_pred)
c_report_=classification_report(y_test,y_pred)
fpr, tpr, threshold=roc_curve(y_test,y_pred)
roc_auc = auc(fpr, tpr)
auc_gender=auc(sorted(y_test),sorted(y_pred))
```

In [256]:
```
# Printing all the evaluation metrics
print("\t\t\tModel Evaluation for scenario 1 (All data Present) KS STATISTIC\n")
print("confusion matrix\n",conf_mat)
print("\n\nAccuracy :",accuracy_)
print("\n\nClassification_Report: \n\n" ,c_report_)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
```

```
                         Model Evaluation for scenario 1 (All data Present) KS STATISTIC

confusion matrix
 [[ 36593  16531]
 [110711  55296]]


Accuracy : 0.41933364060767303


Classification_Report:

              precision    recall  f1-score   support

           0       0.25      0.69      0.37     53124
           1       0.77      0.33      0.46    166007

    accuracy                           0.42    219131
   macro avg       0.51      0.51      0.42    219131
weighted avg       0.64      0.42      0.44    219131
```




# Predicting Age Scenario 1

**Multiclass classification for Age because the linear regression is predicting in decimals but we can't have age in points for the output campaings.**

### Predicting Age Scenario 1

**We will be using multiclass classification for Age because the linear regression is predicting in decimals but we can't have age in points for the output campaings.**

In [2]:
```
final_df = pd.read_csv("final_dff.csv")
final_df.columns
```

Out[2]:
```
Index(['device_id', 'gender', 'cluster_id', 'phone_brand', 'train_test_flag',
       'label_id', 'age_group', 'total_events', 'longitude_median',
       'latitude_median', 'week_day', 'hour', 'final_category',
       'category_code', 'phone_brands_code'],
      dtype='object')
```

In [3]:
```
#Converting categorial columns to dummy variables
categorical_columns = ['week_day','hour','category_code','phone_brands_code']
pd.get_dummies(data=final_df, columns=categorical_columns, drop_first=True).columns
```

Out[3]:
```
Index(['device_id', 'gender', 'cluster_id', 'phone_brand', 'train_test_flag',
       'label_id', 'age_group', 'total_events', 'longitude_median',
       'latitude_median', 'final_category', 'week_day_1', 'week_day_2',
       'week_day_3', 'week_day_4', 'week_day_5', 'week_day_6', 'hour_1',
       'hour_2', 'hour_3', 'hour_4', 'hour_5', 'hour_6', 'hour_7', 'hour_8',
       'hour_9', 'hour_10', 'hour_11', 'hour_12', 'hour_13', 'hour_14',
       'hour_15', 'hour_16', 'hour_17', 'hour_18', 'hour_19', 'hour_20',
       'hour_21', 'hour_22', 'hour_23', 'category_code_1', 'category_code_2',
       'category_code_3', 'category_code_4', 'category_code_5',
       'category_code_6', 'category_code_7', 'category_code_8',
       'category_code_9', 'category_code_10', 'phone_brands_code_1',
       'phone_brands_code_2', 'phone_brands_code_3', 'phone_brands_code_4',
       'phone_brands_code_5', 'phone_brands_code_6', 'phone_brands_code_7',
       'phone_brands_code_8', 'phone_brands_code_9', 'phone_brands_code_10',
       'phone_brands_code_11', 'phone_brands_code_12', 'phone_brands_code_13',
       'phone_brands_code_14', 'phone_brands_code_15', 'phone_brands_code_16',
       'phone_brands_code_17', 'phone_brands_code_18', 'phone_brands_code_19',
       'phone_brands_code_20', 'phone_brands_code_21', 'phone_brands_code_22',
       'phone_brands_code_23', 'phone_brands_code_24', 'phone_brands_code_25',
       'phone_brands_code_26', 'phone_brands_code_27', 'phone_brands_code_28',
       'phone_brands_code_29', 'phone_brands_code_30', 'phone_brands_code_31',
       'phone_brands_code_32', 'phone_brands_code_33', 'phone_brands_code_34',
       'phone_brands_code_35', 'phone_brands_code_36', 'phone_brands_code_37',
       'phone_brands_code_38'],
      dtype='object')
```

In [4]:
```
final_df=pd.get_dummies(data=final_df, columns=categorical_columns, drop_first=True)
```

In [5]:
```
df_train=final_df[final_df["train_test_flag"]=="train"]
df_test=final_df[final_df["train_test_flag"]=="test"]
```

In [6]:
```
y_train=final_df["age_group"]
y_test=final_df["age_group"]
```

In [7]:
```
y_train.value_counts()
```

Out[7]:
```
32+      393320
25-32    343276
0-24     125768
Name: age_group, dtype: int64
```

In [8]:
```
y_test.value_counts()
```

Out[8]:
```
32+      393320
25-32    343276
0-24     125768
Name: age_group, dtype: int64
```

In [9]:
```
df_train.columns
```

Out[9]:
```
Index(['device_id', 'gender', 'cluster_id', 'phone_brand', 'train_test_flag',
       'label_id', 'age_group', 'total_events', 'longitude_median',
       'latitude_median', 'final_category', 'week_day_1', 'week_day_2',
       'week_day_3', 'week_day_4', 'week_day_5', 'week_day_6', 'hour_1',
       'hour_2', 'hour_3', 'hour_4', 'hour_5', 'hour_6', 'hour_7', 'hour_8',
       'hour_9', 'hour_10', 'hour_11', 'hour_12', 'hour_13', 'hour_14',
       'hour_15', 'hour_16', 'hour_17', 'hour_18', 'hour_19', 'hour_20',
       'hour_21', 'hour_22', 'hour_23', 'category_code_1', 'category_code_2',
       'category_code_3', 'category_code_4', 'category_code_5',
       'category_code_6', 'category_code_7', 'category_code_8',
       'category_code_9', 'category_code_10', 'phone_brands_code_1',
       'phone_brands_code_2', 'phone_brands_code_3', 'phone_brands_code_4',
       'phone_brands_code_5', 'phone_brands_code_6', 'phone_brands_code_7',
       'phone_brands_code_8', 'phone_brands_code_9', 'phone_brands_code_10',
       'phone_brands_code_11', 'phone_brands_code_12', 'phone_brands_code_13',
       'phone_brands_code_14', 'phone_brands_code_15', 'phone_brands_code_16',
       'phone_brands_code_17', 'phone_brands_code_18', 'phone_brands_code_19',
       'phone_brands_code_20', 'phone_brands_code_21', 'phone_brands_code_22',
       'phone_brands_code_23', 'phone_brands_code_24', 'phone_brands_code_25',
       'phone_brands_code_26', 'phone_brands_code_27', 'phone_brands_code_28',
       'phone_brands_code_29', 'phone_brands_code_30', 'phone_brands_code_31',
       'phone_brands_code_32', 'phone_brands_code_33', 'phone_brands_code_34',
       'phone_brands_code_35', 'phone_brands_code_36', 'phone_brands_code_37',
       'phone_brands_code_38'],
      dtype='object')
```

In [10]:
```
x_train=final_df.drop(columns=["gender","age_group","final_category","train_test_flag","device_id","phone_brand"])
x_test=final_df.drop(columns=["gender","age_group","final_category","train_test_flag","device_id","phone_brand"])
```

In [11]:
```
x_train.head()
```

Out[11]:

| | cluster_id | label_id | total_events | longitude_median | latitude_median | week_day_1 | week_day_2 | week_day_3 | week_day_4 | week_day_5 | ... | phone_brands_co |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 548.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 1 | 1.0 | 262.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 2 | 1.0 | 251.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 3 | 1.0 | 1008.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |
| 4 | 1.0 | 786.0 | 303 | 30.65 | 104.08 | 1 | 0 | 0 | 0 | 0 | ... | |

5 rows × 82 columns

In [12]:
```
from sklearn.preprocessing import StandardScaler
x_train["total_events"]=StandardScaler().fit_transform(x_train[["total_events"]])
```

In [13]:
```
x_train["longitude_median"]=StandardScaler().fit_transform(x_train[["longitude_median"]])
x_train["latitude_median"]=StandardScaler().fit_transform(x_train[["latitude_median"]])
```

In [14]:
```
x_test["total_events"]=StandardScaler().fit_transform(x_test[["total_events"]])
x_test["longitude_median"]=StandardScaler().fit_transform(x_test[["longitude_median"]])
x_test["latitude_median"]=StandardScaler().fit_transform(x_test[["latitude_median"]])
```

In [15]:
```
x_train
```

Out[15]:

| | cluster_id | label_id | total_events | longitude_median | latitude_median | week_day_1 | week_day_2 | week_day_3 | week_day_4 | week_day_5 | ... | phone_brand |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 548.0 | -0.009729 | 0.034592 | -0.272229 | 1 | 0 | 0 | 0 | 0 | ... | |
| 1 | 1.0 | 262.0 | -0.009729 | 0.034592 | -0.272229 | 1 | 0 | 0 | 0 | 0 | ... | |
| 2 | 1.0 | 251.0 | -0.009729 | 0.034592 | -0.272229 | 1 | 0 | 0 | 0 | 0 | ... | |
| 3 | 1.0 | 1008.0 | -0.009729 | 0.034592 | -0.272229 | 1 | 0 | 0 | 0 | 0 | ... | |
| 4 | 1.0 | 786.0 | -0.009729 | 0.034592 | -0.272229 | 1 | 0 | 0 | 0 | 0 | ... | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 862359 | 1.0 | 235.0 | -0.755984 | 1.145120 | 0.256304 | 0 | 0 | 0 | 0 | 0 | ... | |
| 862360 | 1.0 | 129.0 | -0.755984 | 1.145120 | 0.256304 | 0 | 0 | 0 | 0 | 0 | ... | |
| 862361 | 1.0 | 548.0 | -0.755984 | 1.145120 | 0.256304 | 0 | 0 | 0 | 0 | 0 | ... | |
| 862362 | 1.0 | 168.0 | -0.755984 | 1.145120 | 0.256304 | 0 | 0 | 0 | 0 | 0 | ... | |
| 862363 | 1.0 | 549.0 | -0.755984 | 1.145120 | 0.256304 | 0 | 0 | 0 | 0 | 0 | ... | |

862364 rows × 82 columns

## Hyperparameter Tuning

### Hyperparameter tuning

In [21]:
```
logic1 = LogisticRegression(multi_class="multinomial")
logic2 = RandomForestClassifier(random_state=4)
xgb = XGBClassifier()

stacking_Classifier = StackingCVClassifier(classifiers=[logic1, logic2], meta_classifier=xgb, use_probas=True, cv=3)
```

In [22]:
```
for clf, label in zip([logic1, logic2, stacking_Classifier],
                      ['lr',
                       'Random Forest',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, x_train.values, y_train.values, cv=4, scoring='roc_auc_ovr')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

```
Accuracy: 0.50 (+/- 0.00) [lr]
Accuracy: 0.50 (+/- 0.00) [Random Forest]
[04:47:45] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[04:59:27] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[05:11:52] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[05:24:17] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy: 0.50 (+/- 0.00) [StackingClassifier]
```

In [29]:
```
# Parameters for XGboost
params = {
          'randomforestclassifier__max_depth': [2,5,10],
          'randomforestclassifier__n_estimators': [5,10,20]
         }
```

In [30]:
```
grid = GridSearchCV(estimator=stacking_Classifier,
                    param_grid=params,
                    cv=5,
                    refit=True)
```

In [31]:
```
grid.fit(x_train.values,y_train.values)
```

```
[07:40:49] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the object
ive 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old
behavior.
[07:44:32] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the object
ive 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old
behavior.
[07:48:05] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the object
ive 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old
behavior.
[07:51:23] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the object
ive 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old
behavior.
```

Out[31]:
```
GridSearchCV(cv=5,
             estimator=StackingCVClassifier(classifiers=[LogisticRegression(multi_class='multinomial'),
                                                         RandomForestClassifier(random_state=4)],
                                            cv=3,
                                            meta_classifier=XGBClassifier(base_score=None,
                                                                          booster=None,
                                                                          colsample_bylevel=None,
```

In [32]:
```
cv_keys = ('mean_test_score', 'std_test_score', 'params')
```

In [33]:
```
for r, _ in enumerate(grid.cv_results_['mean_test_score']):
    print("%0.3f +/- %0.2f %r"
          % (grid.cv_results_[cv_keys[0]][r],
             grid.cv_results_[cv_keys[1]][r] / 2.0,
             grid.cv_results_[cv_keys[2]][r]))
```

```
0.454 +/- 0.00 {'randomforestclassifier__max_depth': 2, 'randomforestclassifier__n_estimators': 5}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 2, 'randomforestclassifier__n_estimators': 10}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 2, 'randomforestclassifier__n_estimators': 20}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 5, 'randomforestclassifier__n_estimators': 5}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 5, 'randomforestclassifier__n_estimators': 10}
0.454 +/- 0.00 {'randomforestclassifier__max_depth': 5, 'randomforestclassifier__n_estimators': 20}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 10, 'randomforestclassifier__n_estimators': 5}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 10, 'randomforestclassifier__n_estimators': 10}
0.454 +/- 0.00 {'randomforestclassifier__max_depth': 10, 'randomforestclassifier__n_estimators': 20}
```

### Hyperparameter tuning

In [21]:
```
logic1 = LogisticRegression(multi_class="multinomial")
logic2 = RandomForestClassifier(random_state=4)
xgb = XGBClassifier()

stacking_Classifier = StackingCVClassifier(classifiers=[logic1, logic2], meta_classifier=xgb, use_probas=True, cv=3)
```

In [22]:
```
for clf, label in zip([logic1, logic2, stacking_Classifier],
                      ['lr',
                       'Random Forest',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, x_train.values, y_train.values, cv=4, scoring='roc_auc_ovr')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

```
Accuracy: 0.50 (+/- 0.00) [lr]
Accuracy: 0.50 (+/- 0.00) [Random Forest]
[04:47:45] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[04:59:27] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[05:11:52] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[05:24:17] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy: 0.50 (+/- 0.00) [StackingClassifier]
```

In [29]:
```
# Parameters for XGboost
params = {
          'randomforestclassifier__max_depth': [2,5,10],
          'randomforestclassifier__n_estimators': [5,10,20]
         }
```

In [30]:
```
grid = GridSearchCV(estimator=stacking_Classifier,
                    param_grid=params,
                    cv=5,
                    refit=True)
```

In [31]:
```
grid.fit(x_train.values,y_train.values)
```

```
booster=None,
colsample_bylevel=None,
colsample_bynode=None,
colsample_bytree=None,
enable_categorical=False,
gamma=None,
gpu_id=None,
importance_type=None,
in...
monotone_constraints=None,
n_estimators=100,
n_jobs=None,
num_parallel_tree=None,
predictor=None,
random_state=None,
reg_alpha=None,
reg_lambda=None,
scale_pos_weight=None,
subsample=None,
```

In [32]:
```
cv_keys = ('mean_test_score', 'std_test_score', 'params')
```

In [33]:
```
for r, _ in enumerate(grid.cv_results_['mean_test_score']):
    print("%0.3f +/- %0.2f %r"
          % (grid.cv_results_[cv_keys[0]][r],
             grid.cv_results_[cv_keys[1]][r] / 2.0,
             grid.cv_results_[cv_keys[2]][r]))
```

```
0.454 +/- 0.00 {'randomforestclassifier__max_depth': 2, 'randomforestclassifier__n_estimators': 5}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 2, 'randomforestclassifier__n_estimators': 10}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 2, 'randomforestclassifier__n_estimators': 20}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 5, 'randomforestclassifier__n_estimators': 5}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 5, 'randomforestclassifier__n_estimators': 10}
0.454 +/- 0.00 {'randomforestclassifier__max_depth': 5, 'randomforestclassifier__n_estimators': 20}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 10, 'randomforestclassifier__n_estimators': 5}
0.453 +/- 0.00 {'randomforestclassifier__max_depth': 10, 'randomforestclassifier__n_estimators': 10}
0.454 +/- 0.00 {'randomforestclassifier__max_depth': 10, 'randomforestclassifier__n_estimators': 20}
```

In [34]:
```
stacking_age_class=stacking_Classifier.fit(x_train.values,y_train.values)
```

# Best model estimator

### Best model estimator

In [35]:
```
best_model_ageclass=grid.best_estimator_
```

In [36]:
```
# Saving the model as pickle file
import pickle
filename = 'age_pred.sav'
pickle.dump(best_model_ageclass, open(filename, 'wb'))
```

In [37]:
```
# Uploading the model in s3 bucket and jupyter notebook
!aws s3 cp age_pred.sav s3://s3accessec2/final_submission/age_pred.sav .
```

```
Unknown options: .
```

In [40]:
```
y_pred=best_model_ageclass.predict(x_test.values)
```

In [41]:
```
y_pred
```

Out[41]:
```
array(['32+', '32+', '32+', ..., '32+', '32+', '32+'], dtype=object)
```

In [42]:
```
y_probabilites=best_model_ageclass.predict_proba(x_test)
```

# Model evaluation

### Model evaluation

```
In [44]: y_probabilites[:, 1]

Out[44]: array([0.38966367, 0.38792554, 0.3977278 , ..., 0.39272413, 0.39129588,
                0.39272413], dtype=float32)
```

```
In [55]: # Since it is a multiclass classification problem, we will not be using the roc_auc curve
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,roc_curve,auc
conf_mat=confusion_matrix(y_test, y_pred)
accuracy_=accuracy_score(y_test, y_pred)
c_report_=classification_report(y_test,y_pred)
#fpr, tpr, threshold=roc_curve(y_test,y_pred)
#roc_auc = auc(fpr, tpr)
```

```
In [57]: # Confusion matrix evaluation
print("\t\t\tModel Evaluation for scenario 1 (All data Present)\n")
print("confusion matrix\n",conf_mat)
print("\n\nAccuracy :",accuracy_)
print("\n\nClassification_Report: \n\n" ,c_report_)
'''plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()'''
```

```
                        Model Evaluation for scenario 1 (All data Present)

confusion matrix
 [[    24   4328 121416]
 [    62  11888 331326]
 [    88  13496 379736]]


Accuracy : 0.45415624956514883


Classification_Report:

              precision    recall  f1-score   support

        0-24       0.14      0.00      0.00    125768
       25-32       0.40      0.03      0.06    343276
         32+       0.46      0.97      0.62    393320

    accuracy                           0.45    862364
   macro avg       0.33      0.33      0.23    862364
weighted avg       0.39      0.45      0.31    862364
```

```
Out[57]: "plt.title('Receiver Operating Characteristic')\nplt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)\nplt.legend(loc = 'lo
wer right')\nplt.plot([0, 1], [0, 1],'r--')\nplt.xlim([0, 1])\nplt.ylim([0, 1])\nplt.ylabel('True Positive Rate')\nplt.xlabel
('False Positive Rate')\nplt.show()"
```