MLC Capstone Project - Model Building

By Krishnabansuri K

Model Building

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

	event_id	app_id	is_installed	is_active	label_id	category
0	2	-5720078949152207372	1	0	704	Property Industry 2.0
1	2	-1633887856876571208	1	0	1007	P2P net loan
2	2	-1633887856876571208	1	0	783	High risk
3	2	-1633887856876571208	1	0	779	Higher income
4	2	-1633887856876571208	1	0	775	Liquid medium

	device_id	phone_brand	device_model	gender	age	group_name
0	-9223067244542180000	vivo	Y19T	M	24	M0-24
1	-9223042152723780000	Xiaomi	MI 3	\N	\N	\N
2	-9222956879900150000	samsung	Galaxy Note 2	M	36	M32+
3	-9222896629442490000	OPPO	A31	\N	\N	\N
4	-9222894989445030000	Gionee	ELIFE E7 Mini	\N	\N	\N

	event_id	device_id	event_timestamp	longitude	latitude	gender	age	group_name
0	1315995	-9222956879900150000	2016-05-06 15:42:15	113.24	23.19	M	36	M32+
1	2068832	-9222956879900150000	2016-05-07 12:20:13	113.24	23.19	M	36	M32+
2	1481001	-9222956879900150000	2016-05-06 15:34:54	113.24	23.19	M	36	M32+
3	2111353	-9222956879900150000	2016-05-07 12:09:01	113.24	23.19	M	36	M32+
4	1650018	-9222956879900150000	2016-05-06 15:32:26	113.24	23.19	M	36	M32+

Model Building

- List of data cleaning techniques applied such as missing value treatment, etc.
  - Across all the 3 data sets, app data, events and non-events data, determined the % of missing values and retained columns where missing values is <40%
  - Events data that have latitude and longitude between -1 and 1 and equals 0 are eliminated as they have no importance
  - Converted gender data (M, F) to 1 and 2 respectively
  - Replaced any special characters in Gender and Age which are the target variables with 0 or median value

Model Building

- Feature engineering techniques that were used along with proper reasoning to support why the technique was used
  - Created features for such as Median Latitude and Median Longitude for different event ids

```
#Grouping by event_id and taking the median of latitude
lat_events = train_event_data.groupby("event_id")["latitude"].apply(lambda x: np.median([float(s) for s in x]))
#Grouping by event_id and taking the median of longitude
long_events = train_event_data.groupby("event_id")["longitude"].apply(lambda x: np.median([float(s) for s in x]))
#setting to the original data
train_event_data['event_med_lat']=train_event_data.index.map(lat_events)
train_event_data['event_med_long']=train_event_data.index.map(long_events)
train_event_data.head(10)
```

]:

	event_id	device_id	event_timestamp	longitude	latitude	gender	age	group_name	event_timestamp_ts	dayofweek	hour	event_med_lat
0	1315995	-9222956879900150000	2016-05-06 15:42:15	113.24	23.19	1	36	M32+	2016-05-06 15:42:15	4	15	NaN
1	2068832	-9222956879900150000	2016-05-07 12:20:13	113.24	23.19	1	36	M32+	2016-05-07 12:20:13	5	12	31.24
2	1481001	-9222956879900150000	2016-05-06 15:34:54	113.24	23.19	1	36	M32+	2016-05-06 15:34:54	4	15	NaN
3	2111353	-9222956879900150000	2016-05-07 12:09:01	113.24	23.19	1	36	M32+	2016-05-07 12:09:01	5	12	29.70
4	1650018	-9222956879900150000	2016-05-06 15:32:26	113.24	23.19	1	36	M32+	2016-05-06 15:32:26	4	15	23.28
5	1687118	-9222956879900150000	2016-05-06 15:39:42	113.24	23.19	1	36	M32+	2016-05-06 15:39:42	4	15	28.66

Model Building

- Feature engineering techniques that were used along with proper reasoning to support why the technique was used
  - Used information related to the location of the users (latitude and longitude data) to create features representing changes in the latitude and longitude details at different times of the day

```
train_event_data['hourbin'] = [1 if ((x>=1)&(x<=6)) else 2 if ((x>=7)&(x<=12)) else 3 if ((x>=13)&(x<=18)) else 4 for x in tr
#Grouping by event_id and taking the median of latitude
lat_events_hour = train_event_data.groupby("latitude")["hourbin"].apply(lambda x: " ".join('0'+str(s) for s in x))
#Grouping by event_id and taking the median of longitude
long_events_hour = train_event_data.groupby("longitude")["hourbin"].apply(lambda x: " ".join('0'+str(s) for s in x))
#setting to the original data
train_event_data['hourbin_lat']=train_event_data.index.map(lat_events_hour)
train_event_data['hourbin_long']=train_event_data.index.map(long_events_hour)
train_event_data.head(10)
```

longitude	latitude	gender	age	group_name	event_timestamp_ts	dayofweek	hour	event_med_lat	event_med_long	hourbin	hourbin_lat	hourbin_long
113.24	23.19	1	36	M32+	2016-05-06 15:42:15	4	15	NaN	NaN	3	04 03 03 03 02 03 03 04 04 03 03 02 03 02 02 0...	04 03 03 03 02 03 03 04 04 03 03 02 03 02 02 0...
113.24	23.19	1	36	M32+	2016-05-07 12:20:13	5	12	31.24	121.38	2	01 02 01 02 02 01 01 02 01 04 01 02 01 04 03 0...	01 02 01 02 02 01 01 02 01 04 01 02 01 04 03 0...

Model Building

- Feature engineering techniques that were used along with proper reasoning to support why the technique was used
    - Created a feature called Average Events, which can give you an estimate of how long the users' mobile phones are active.

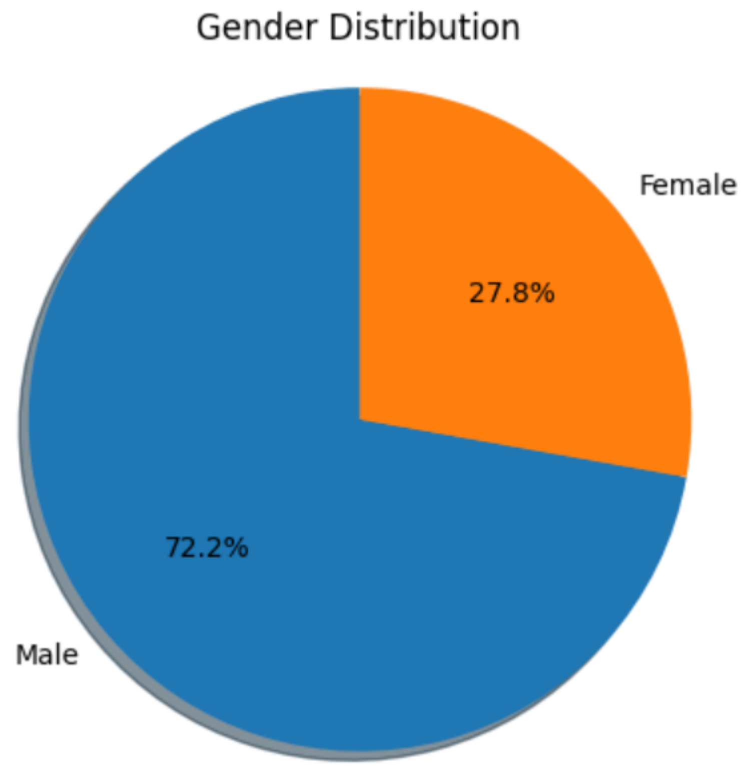
```
#Grouping by event_id and taking the median of latitude
apps_active = train_app_data.groupby(['event_id'])['is_active'].apply(lambda x: np.mean([float(s) for s in x]))
apps_active
#setting to the original data
train_app_data['average_events']=train_app_data.index.map(apps_active)
train_app_data.head(10)
```

	event_id	app_id	is_installed	is_active	label_id	category	average_events
0	2	-5720078949152207372	1	0	704	Property Industry 2.0	NaN
1	2	-1633887856876571208	1	0	1007	P2P net loan	NaN
2	2	-1633887856876571208	1	0	783	High risk	0.361111
3	2	-1633887856876571208	1	0	779	Higher income	NaN
4	2	-1633887856876571208	1	0	775	Liquid medium	NaN
5	2	5927333115845830913	1	1	172	IM	NaN
6	2	-1633887856876571208	1	0	757	P2P	0.373874
7	2	-1633887856876571208	1	0	756	Internet banking	0.304348

Model Building

- 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]

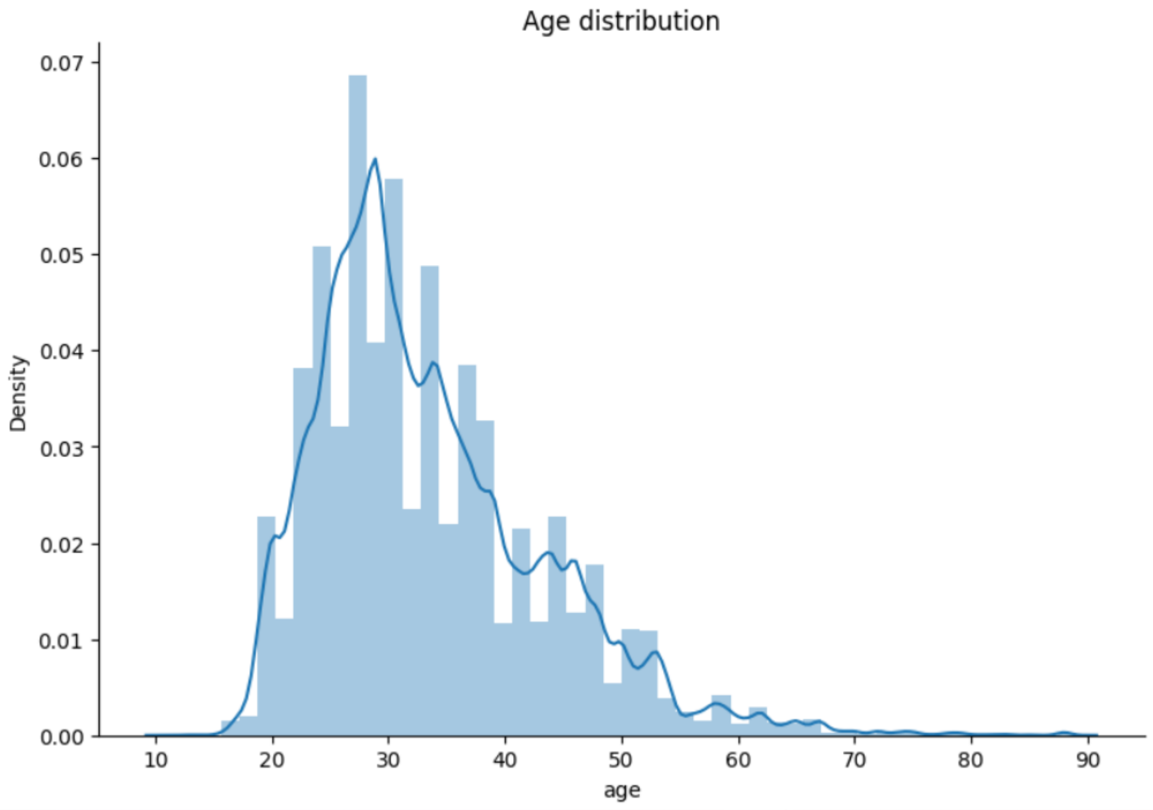
```
#https://matplotlib.org/3.1.1/gallery/pie_and_polar_charts/pie_features.html
gender_dict=train_event_data['gender'].value_counts()
gender_labels=['Male','Female']
male_percentage=(gender_dict['M']*100)//train_event_data.shape[0]
female_percentage=(gender_dict['F']*100)//train_event_data.shape[0]
sizes=[male_percentage,female_percentage]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=gender_labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.title('Gender Distribution')
plt.show()
```



Model Building

- 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]

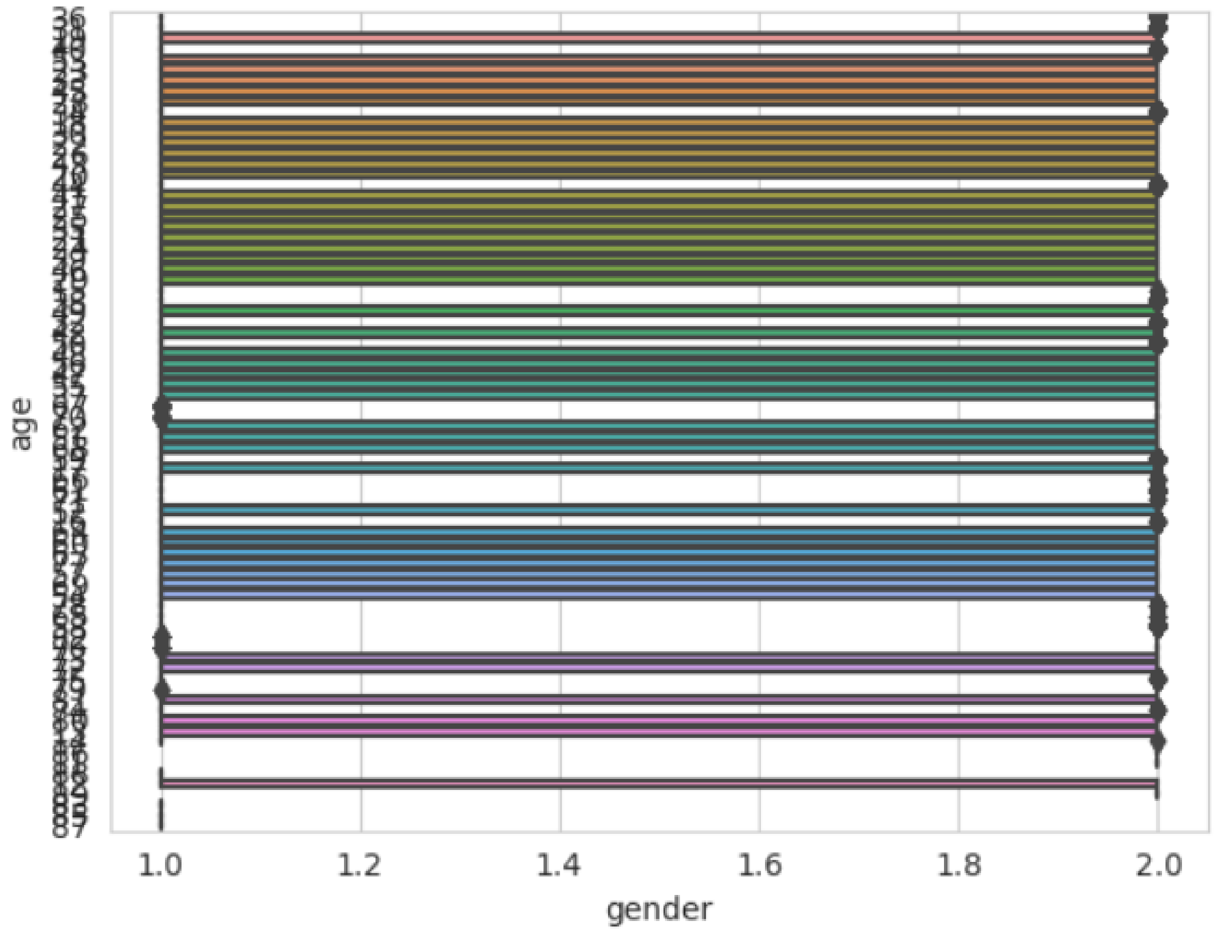
```
#age distribution
fig = plt.figure(figsize=(9, 6))
sns.distplot(train_event_data.age, ax=fig.gca())
plt.title('Age distribution')
sns.despine()
```



Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
  - Boxplot analysis for gender and age [bivariate].

```python
import seaborn as sns
sns.set_style('whitegrid')
ax= sns.boxplot(x='gender',y='age',data=train_event_data)
```

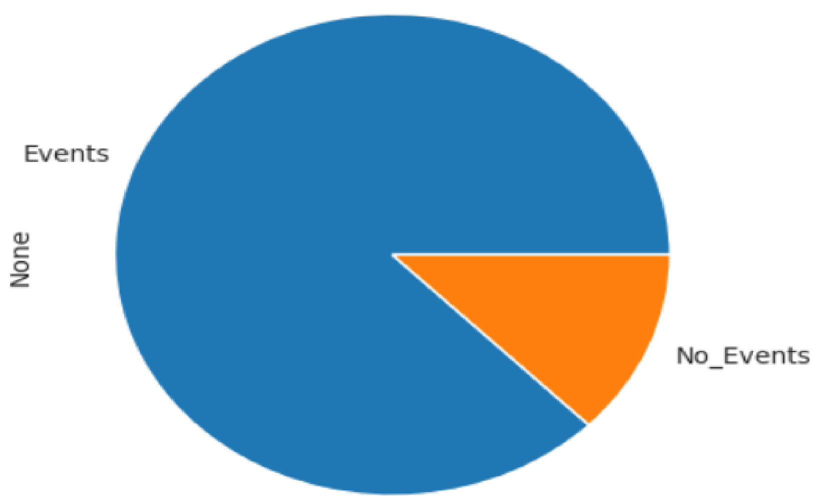


Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
  - Plot the percentage of the device_ids with and without event data.

```
plt.figure()
devices_events = np.in1d(train_non_event_data['device_id'].values,train_event_data['device_id'].values)
pd.Series(devices_events).map({True:'No_Events',False:'Events'}).value_counts().plot.pie()
plt.title("Devices with events and no events (Train Data)")
plt.show()
print("Devices with Events Percentage in Train Data: ",(list(devices_events).count(True)/len(devices_events))*100," %")
print("Devices with No Events Percentage in Train Data: ",(list(devices_events).count(False)/len(devices_events))*100, " %")
```

Devices with events and no events (Train Data)



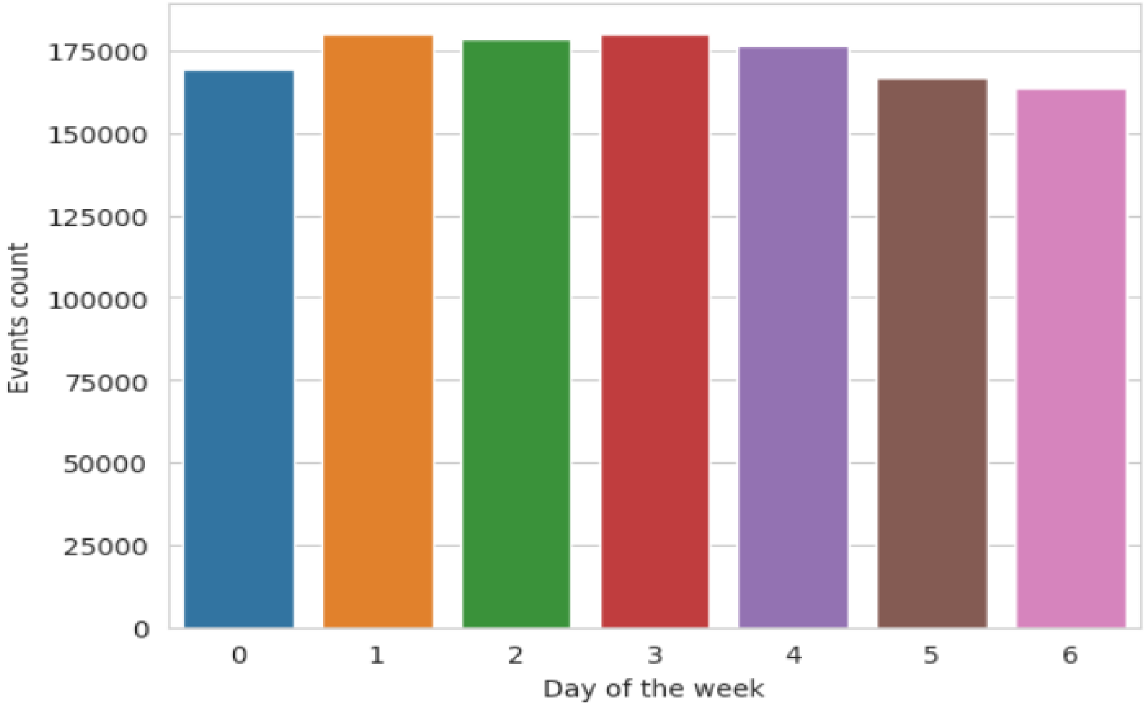

```
Devices with Events Percentage in Train Data:  12.498130781226635  %
Devices with No Events Percentage in Train Data:  87.50186921877336  %
```

Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
    - Plot a graph representing the distribution of events over different days of a week.

```
events_data = train_event_data[['event_id','dayofweek']]
event_counts = events_data["dayofweek"].value_counts().reset_index()
ax = sns.barplot(x = event_counts["index"], y = event_counts["dayofweek"])
ax.set_xlabel('Day of the week')
ax.set_ylabel('Events count')
```

```
]: Text(0, 0.5, 'Events count')
```

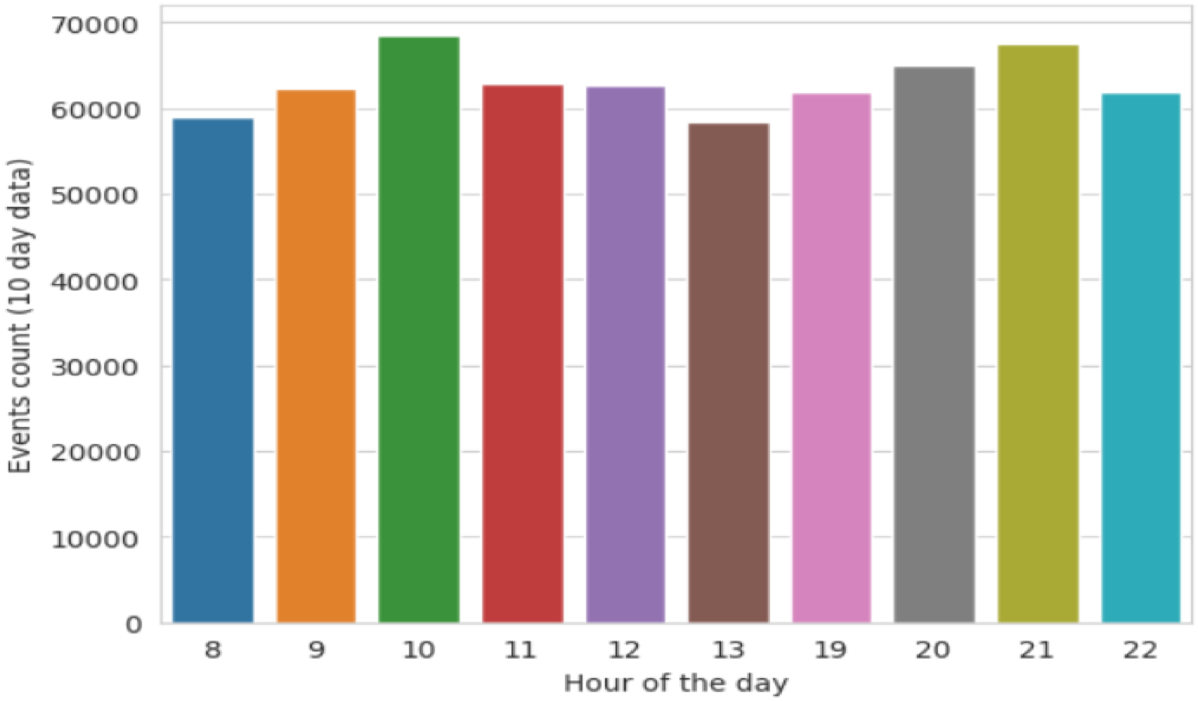


Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
    - Plot a graph representing the distribution of events per hour [for one-week data].

```
events_hour = train_event_data[['event_id','hour']]
event_hour_counts = events_hour["hour"].value_counts().reset_index().head(10)
ax = sns.barplot(x = event_hour_counts["index"], y = event_hour_counts["hour"])
ax.set_xlabel('Hour of the day')
ax.set_ylabel('Events count (10 day data)')
```

```
]: Text(0, 0.5, 'Events count (10 day data)')
```



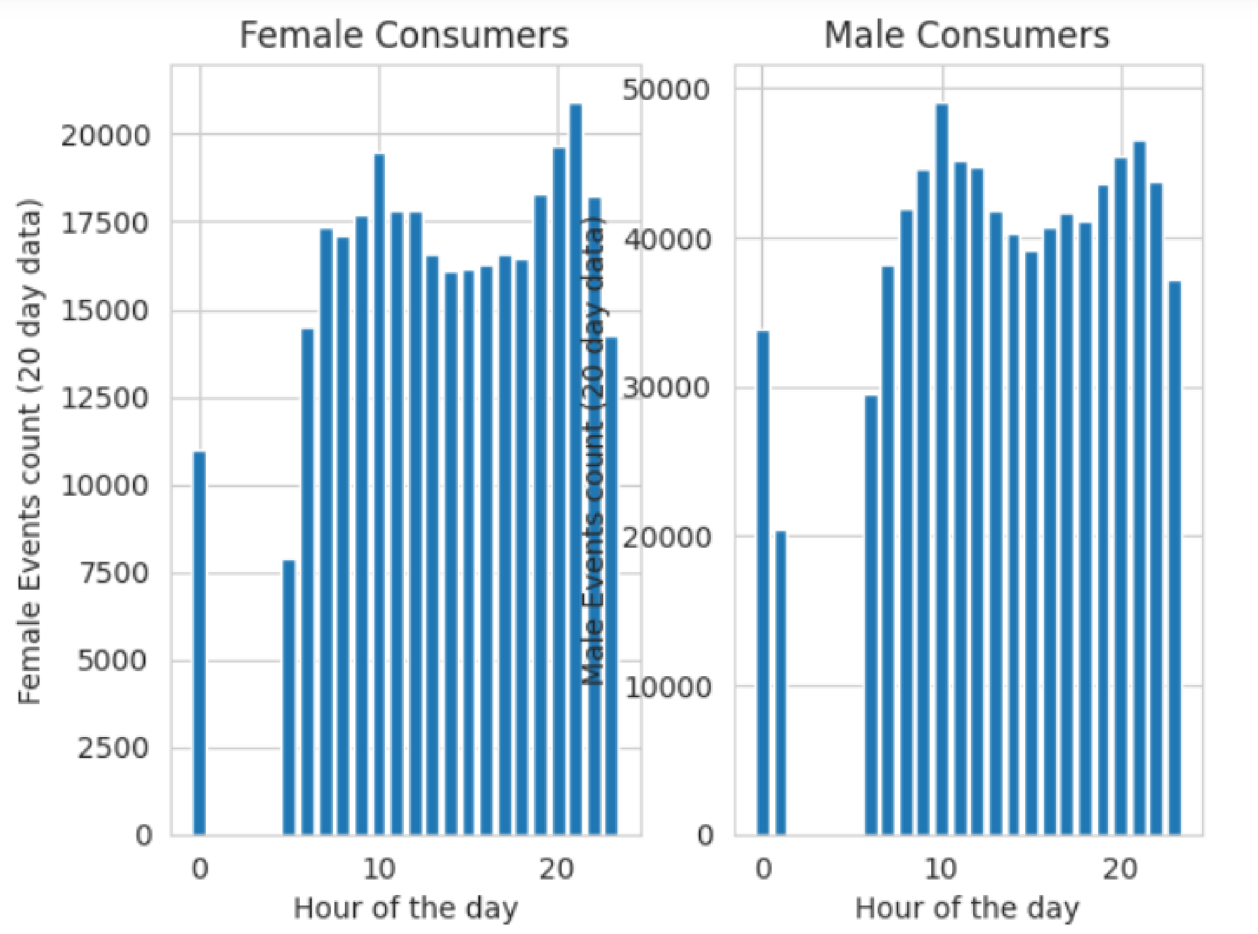
Model Building

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

```
plt.subplot(1, 2, 1) # row 1, col 2 index 1
plt.bar(f_event_hour_counts["index"], f_event_hour_counts["hour"])
plt.title("Female Consumers")
plt.xlabel('Hour of the day')
plt.ylabel('Female Events count (20 day data)')

plt.subplot(1, 2, 2) # index 2
plt.bar(m_event_hour_counts["index"], m_event_hour_counts["hour"])
plt.title("Male Consumers")
plt.xlabel('Hour of the day')
plt.ylabel('Male Events count (20 day data)')

plt.show()
```

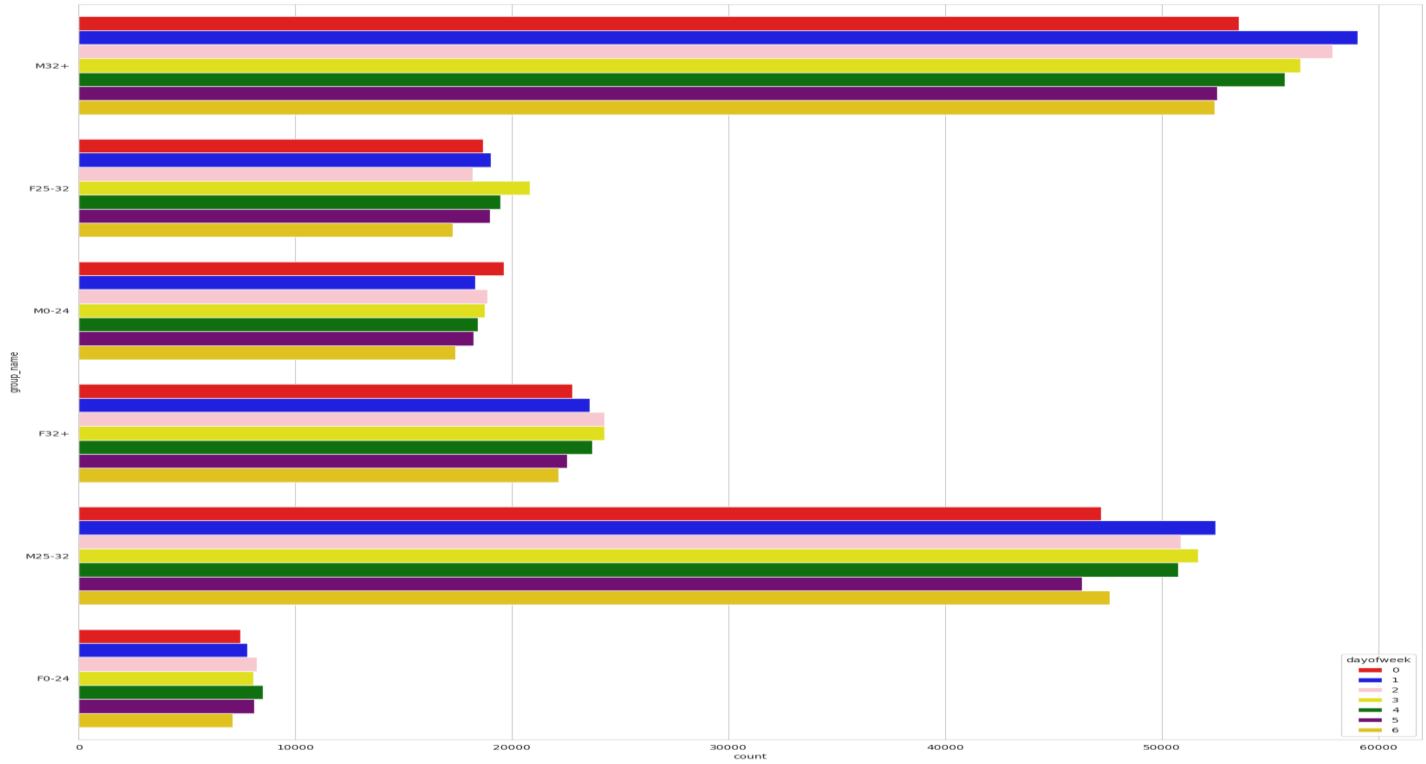


Model Building

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

```
age_events_hour = train_event_data[['dayofweek','group_name']]
age_events_counts = age_events_hour['dayofweek'].value_counts().reset_index().head(10)['index']

plt.figure(figsize=(20,20))
sns.countplot(y="group_name", hue="dayofweek", data=age_events_hour[age_events_hour['dayofweek'].isin(age_events_counts)], pa
```



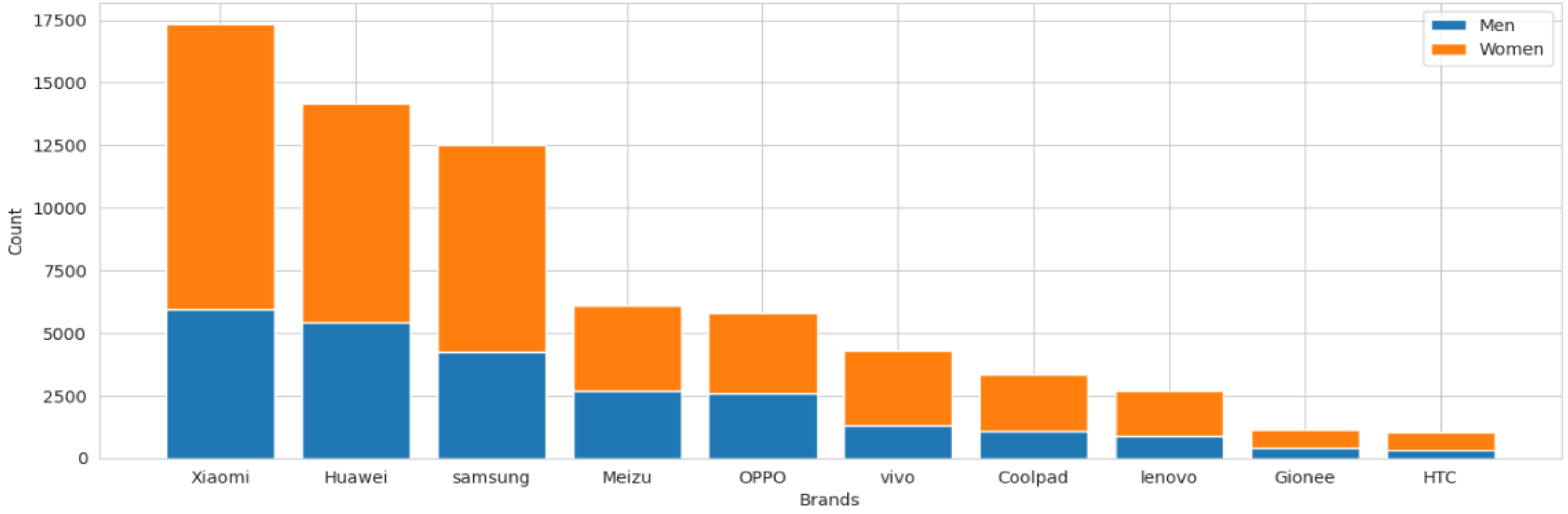
Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
    - Stacked bar chart for the top 10 mobile brands across male and female consumers.

```
# Plotting to see if any brand is popular among any gender
plt.figure(figsize=(15,5))

ind = np.arange(len(list(brand_counts_female['index'])))
plot_1 = plt.bar(ind, brand_counts_male['phone_brand'])
plot_2 = plt.bar(ind, brand_counts_female['phone_brand'], bottom=brand_counts_male['phone_brand'])

plt.xticks(ind, list(brand_counts_female['index']))
plt.xlabel('Brands')
plt.ylabel('Count')
plt.legend((plot_1,plot_2),('Men','Women'))
plt.show()
```



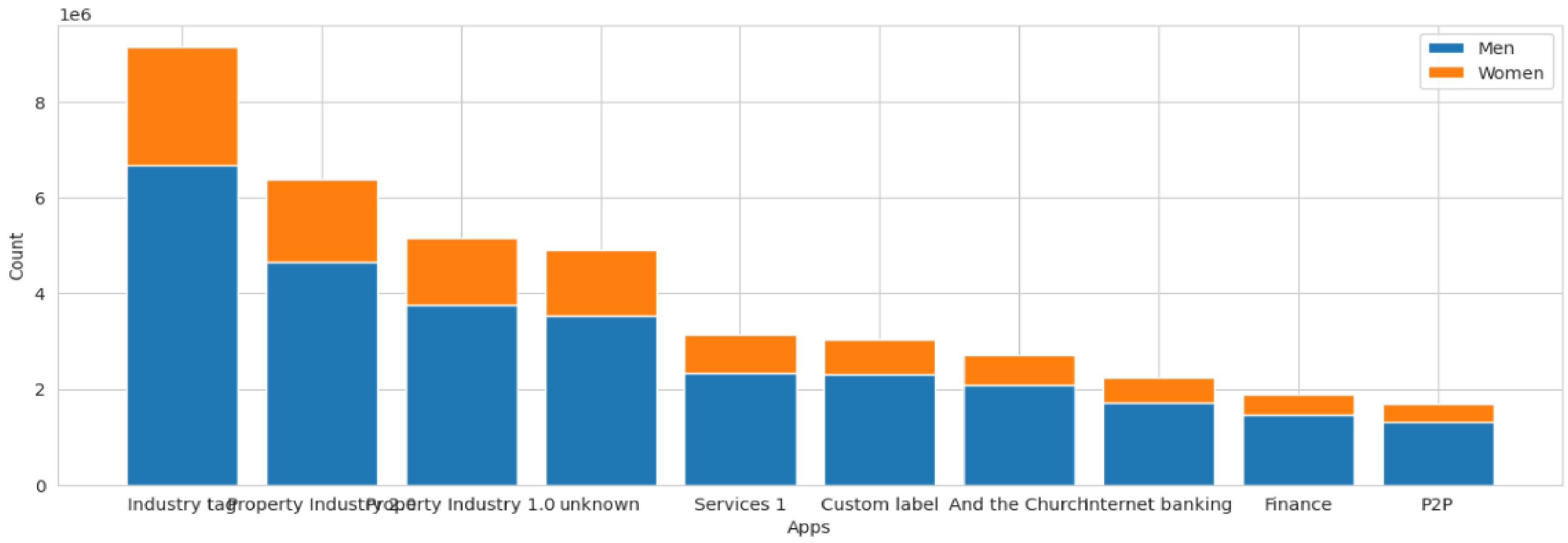
Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
  - Prepare a chart representing the ten frequently used applications and their respective male and female percentage.

```
plt.figure(figsize=(15,5))

ind = np.arange(len(list(app_counts_female['index'])))
plot_1 = plt.bar(ind, app_counts_male['category'])
plot_2 = plt.bar(ind, app_counts_female['category'], bottom=app_counts_male['category'])

plt.xticks(ind, list(app_counts_female['index']))
plt.xlabel('Apps')
plt.ylabel('Count')
plt.legend((plot_1,plot_2),('Men','Women'))
plt.show()
```

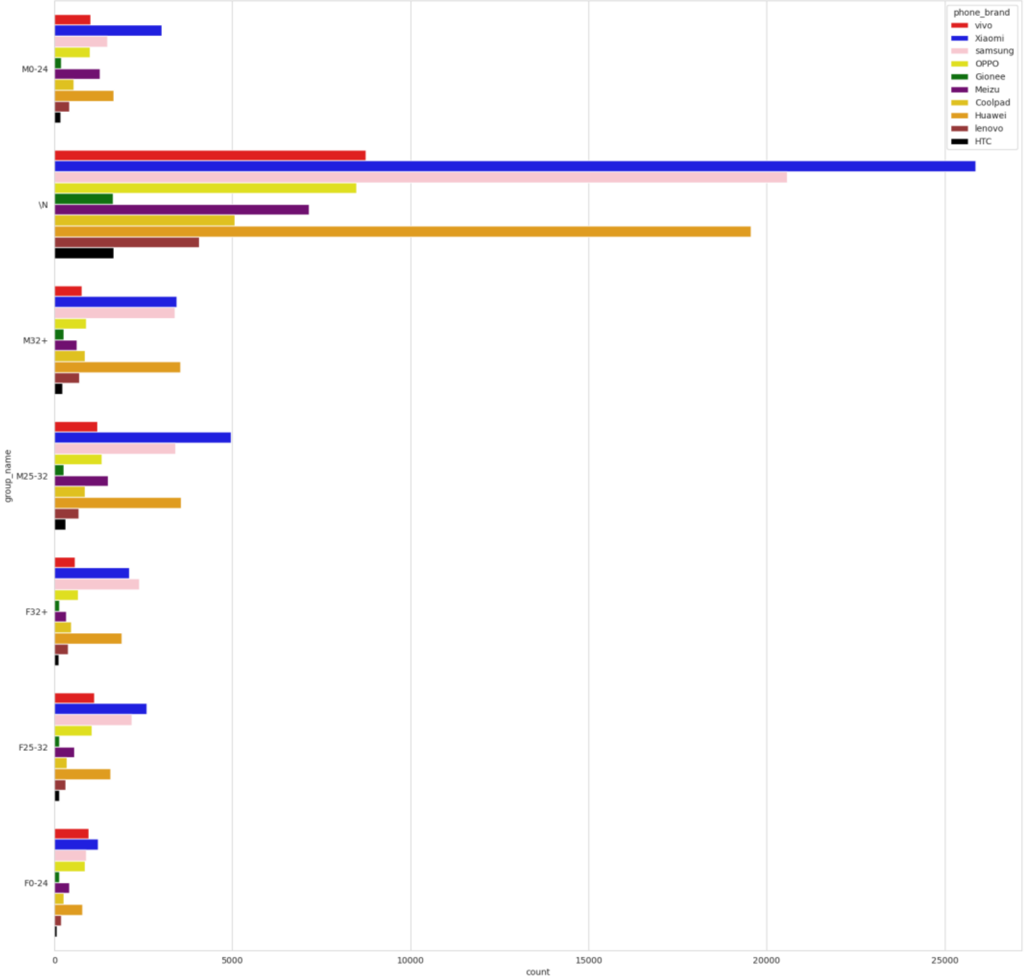


Model Building

- Outputs to the various EDA and Visualisation codes along with the corresponding results and the insights gathered from each EDA and visualisation
    - 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_mobile_brand_events = customer_group['phone_brand'].value_counts().reset_index().head(10)['index']

plt.figure(figsize=(20,20))
sns.countplot(y="group_name", hue="phone_brand", data=customer_group[customer_group['phone_brand'].isin(top_mobile_brand_even
```



Model Building

- Geospatial visualisations along with the insights gathered from this visualisation
  - Plot the visualisation plot for a sample of 1 lakh data points.

```
# Set up plot
df_events_sample = train_event_data.sample(n=100000)
plt.figure(1, figsize=(12,6))

# Mercator of World
m1 = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')

m1.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m1.drawmapboundary(fill_color='#000000')                # black background
m1.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders

# Plot the data
mxy = m1(df_events_sample["longitude"].tolist(), df_events_sample["latitude"].tolist())
m1.scatter(mxy[0], mxy[1], s=3, c="#1292db", lw=0, alpha=1, zorder=5)

plt.title("Global view of events")
plt.show()
```

Model Building

- Geospatial visualisations along with the insights gathered from this visualisation
  - Plot the visualisation plot for a sample of 1 lakh data points.

Global view of events

Model Building

- Geospatial visualisations along with the insights gathered from this visualisation
  - Compare the event visualisation plots based on the users' gender information. [This can be done on the sample of 1 lakh data points.]

```
# Male/female plot
plt.figure(4, figsize=(12,6))

plt.subplot(121)
m4a = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')
m4a.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m4a.drawmapboundary(fill_color='#000000')                # black background
m4a.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders
mxy = m4a(df_m["longitude"].tolist(), df_m["latitude"].tolist())
m4a.scatter(mxy[0], mxy[1], s=5, c="#1292db", lw=0, alpha=0.1, zorder=5)
plt.title("Global Male events")

plt.subplot(122)
m4b = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')
m4b.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m4b.drawmapboundary(fill_color='#000000')                # black background
m4b.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders
mxy = m4b(df_f["longitude"].tolist(), df_f["latitude"].tolist())
m4b.scatter(mxy[0], mxy[1], s=5, c="#fd3096", lw=0, alpha=0.1, zorder=5)
plt.title("Global Female events")

plt.show()
```

Model Building

- Geospatial visualisations along with the insights gathered from this visualisation
  - Compare the event visualisation plots based on the users' gender information. [This can be done on the sample of 1 lakh data points.]

Model Building

- Geospatial visualisations along with the insights gathered from this visualisation
  - Compare the event visualisation plots based on the following age groups:
  - a) 0–24
  - b) 25–32
  - c) 32+

```python
df_M024 = df_events_sample[df_events_sample["group_name"]=='M0-24']
df_F024 = df_events_sample[df_events_sample["group_name"]=='F0-24']
df_M024.append(df_F024)

df_M2532 = df_events_sample[df_events_sample["group_name"]=='M25-32']
df_F2532 = df_events_sample[df_events_sample["group_name"]=='F25-32']
df_M2532.append(df_F2532)

df_M32 = df_events_sample[df_events_sample["group_name"]=='M32+']
df_F32 = df_events_sample[df_events_sample["group_name"]=='F32+']
df_M32.append(df_F32)

# 0-24 plot
plt.figure(5, figsize=(8,6))

plt.subplot(121)
m4a = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')
m4a.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m4a.drawmapboundary(fill_color='#000000')                # black background
m4a.drawcountries(linewidth=0.1, color="w")              # thin white line for country
mxy = m4a(df_M024["longitude"].tolist(), df_M024["latitude"].tolist())
m4a.scatter(mxy[0], mxy[1], s=5, c="#1292db", lw=0, alpha=0.1, zorder=5)
plt.title("Global events for age grp 0-24")
```

```python
# 25-32 plot
m4b = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')
m4b.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m4b.drawmapboundary(fill_color='#000000')                # black background
m4b.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders
mxy = m4b(df_M2532["longitude"].tolist(), df_M2532["latitude"].tolist())
m4b.scatter(mxy[0], mxy[1], s=5, c="#fd3096", lw=0, alpha=0.1, zorder=5)
plt.title("Global events for age grp 25-32")

plt.show()

#32+ plot
plt.figure(5, figsize=(8,6))

plt.subplot(121)
m4a = Basemap(projection='merc',
             llcrnrlat=-60,
             urcrnrlat=65,
             llcrnrlon=-180,
             urcrnrlon=180,
             lat_ts=0,
             resolution='c')
m4a.fillcontinents(color='#191919',lake_color='#000000') # dark grey land, black lakes
m4a.drawmapboundary(fill_color='#000000')                # black background
m4a.drawcountries(linewidth=0.1, color="w")              # thin white line for country borders
mxy = m4a(df_M32["longitude"].tolist(), df_M32["latitude"].tolist())
m4a.scatter(mxy[0], mxy[1], s=5, c="#1292db", lw=0, alpha=0.1, zorder=5)
plt.title("Global events for age grp 32+")

plt.show()
```

Model Building

- Geospatial visualisations along with the insights gathered from this visualisation
  - Compare the event visualisation plots based on the following age groups:
  - a) 0–24
  - b) 25–32
  - c) 32+

Model Building

- Results interpreting the clusters formed as part of DBSCAN Clustering and how the cluster information is being used
  - Used DBSCAN clustering to reduce the events data based on lat and long. Code for the same

```
from sklearn.cluster import DBSCAN
from geopy.distance import great_circle
from shapely.geometry import MultiPoint
#coords = train_event_data[['latitude','longitude']].to_numpy()
```

```
kms_per_radian = 6371.0088
epsilon = 1.5/kms_per_radian
db = DBSCAN(eps= 0.5, min_samples= 10, algorithm = 'ball_tree', metric = 'haversine').fit(np.radians(coords))
```

```
cluster_labels = db.labels_
num_clusters = len(set(cluster_labels))
clusters = pd.Series([coords[cluster_labels == n] for n in range(num_clusters)])
print('Number of clusters: {}'.format(num_clusters))

def get_centermost_point(cluster):
    centroid = (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
    centermost_point = min(cluster, key=lambda point: great_circle(point, centroid).m)
    return tuple(centermost_point)
centermost_points = clusters.map(get_centermost_point)

lats, lons = zip(*centermost_points)
rep_points = pd.DataFrame({'longitude':lons, 'latitude':lats})

rs = rep_points.apply(lambda row: train_event_data[(train_event_data['latitude']==row['latitude']) &amp;&amp; (df['longitude'
```

Model Building

- A brief summary of any additional subtask that was performed and may have improved the data cleaning and feature generation step
  - I have applied following methods for data cleaning and feature generation
  - Convert categorical data to numerical data
  - one hot/label encoding from pandas
  - csr matrix from scipy - sparse matrix

Model Building

- All the data preparation steps that were used before applying the ML algorithm
  - Got rid of duplicate device ids in non events data
  - Encoding the brands using LabelEncoder
  - Concatenating Phone Brand and Model and encoding the same
  - Encoding the device models using LabelEncoder same way as done for Brands
  - Dropping columns timestamp, timestamp_ts, phone_brand, device_model, group_name
  - Read the test and train data and divide the data for events and without events
  - Checking and extracting the Device Ids which have Event Details for Train Data
  - Setting device_id as index for Train, Test Data
  - Created columns trainrow, testrow in Train and Test Data to indicate which row a particular device belongs to and this will be useful in our One-hot encoded Sparse Matrix Creation, in which we will specify number of rows in the sparse matrix
  - Getting the csr matrix from the encoded columns for events data and non-events data
  - Stacking all the features together for GENDER/AGE analysis ---- WITH EVENTS

Model Building

- All the data preparation steps that were used before applying the ML algorithm
  - Stacking all the features together - GENDER/AGE analysis WITHOUT EVENTS
  - Stacking all the features together for GENDER/AGE analysis WITH EVENTS
  - Saving Data without Events for GENDER and AGE prediction
  - Saving Devices with Events for GENDER and AGE prediction

Model Building

- 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.)
  - Segregated the data that is data that has event data and that doesn't have event data
  - Scenario 1: latitude-longitude data, application id data, event data and devices data
  - Scenario 2: only have the mobile phone, brand and device data available.
  - Gender and Age Prediction
  - **Gender - scenario1**
  - used logistic regression, XGboost classifier with GridSearchCV, Randomforest classifier and finally used StackingCVClassifier
  - **Age - scenario1**
  - used linear regression, XGBoost regressor with GridsearchCV, Randomforest regressor and finally used stackingCVregressor
  - The same set of models were applied for scenario 2 also

Model Building

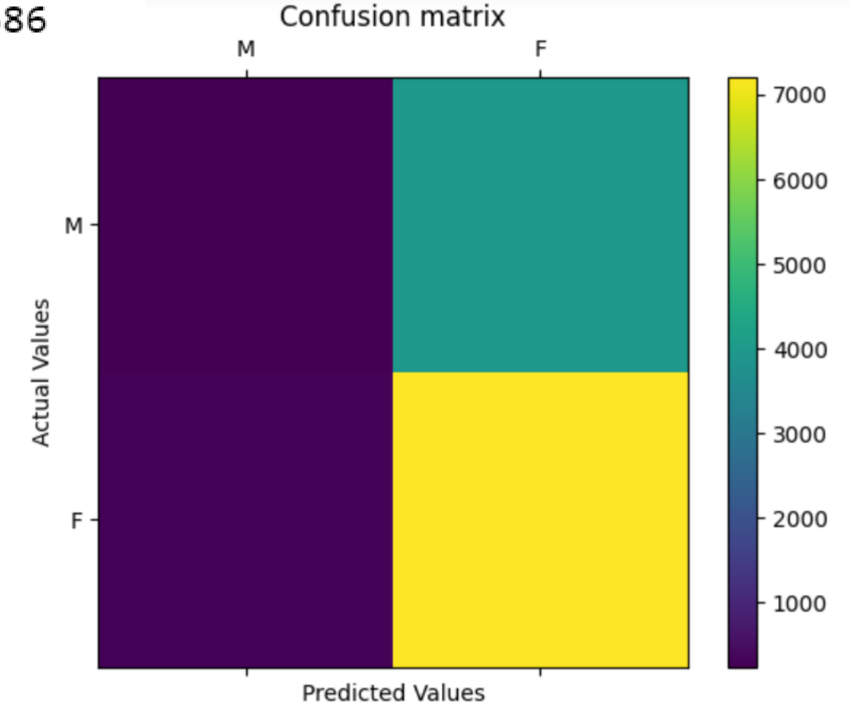
- The reason for using regression or classification for age prediction
- When Age was classified there were quite a lot of bins. Classification could have been adopted if the bins were limited. Since the bins were quite a lot, regression was adopted.

Model Building

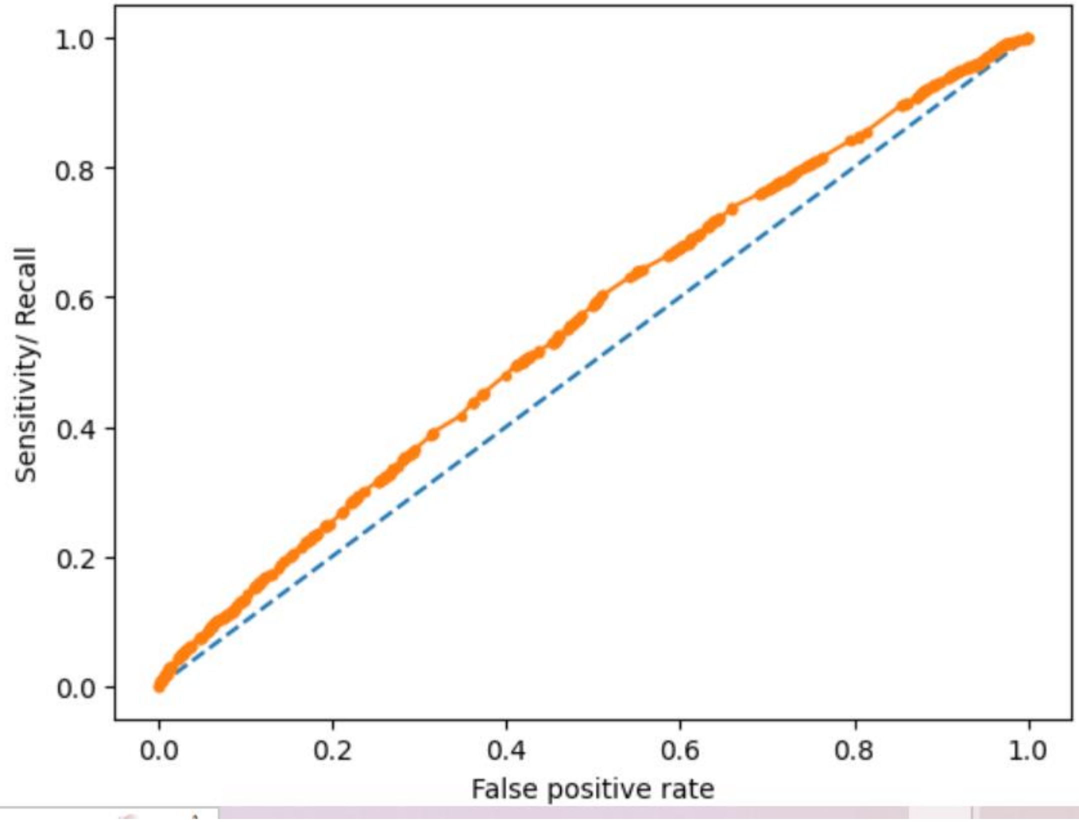
- The outcomes of the evaluation metrics (results for both Scenario 1 and Scenario 2 must be shown separately).
    - Model Evaluation for Gender prediction without events

```
Logloss: 12.58

Precision: 0.645352
Recall: 0.959521
F1 score: 0.771686
```




```
AUC - Test Set: 55.65%
```

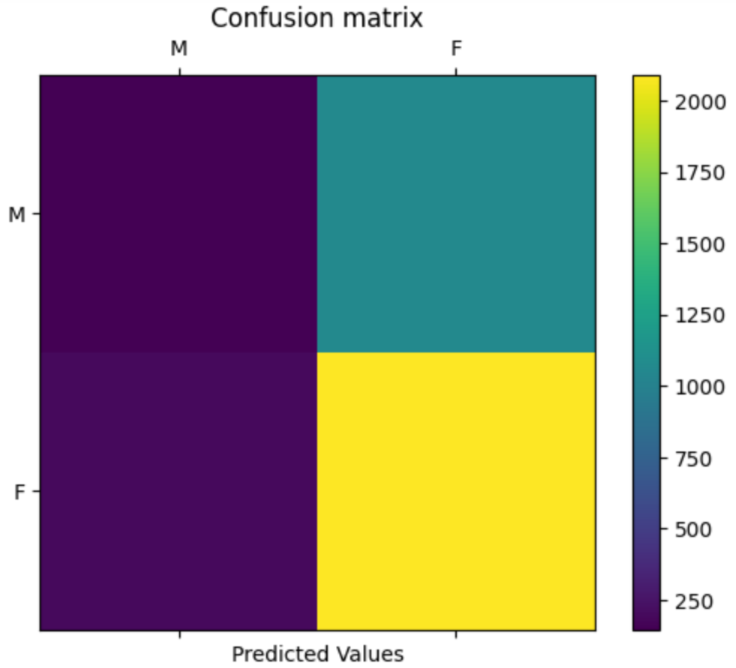


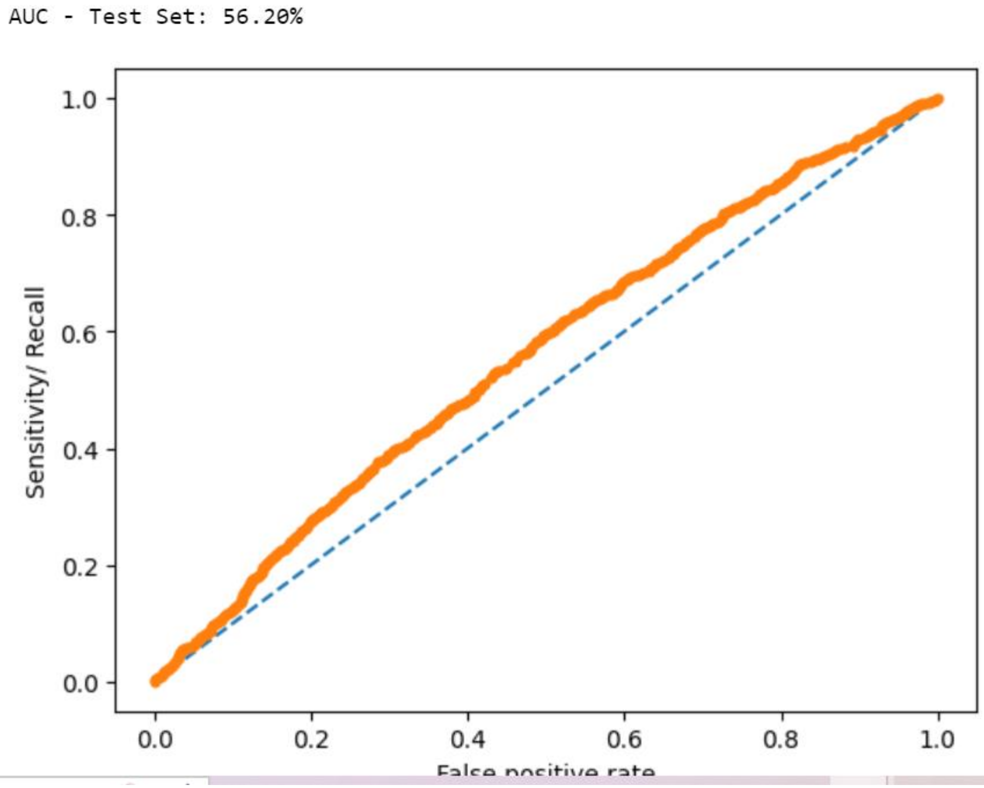
Model Building

- The outcomes of the evaluation metrics (results for both Scenario 1 and Scenario 2 must be shown separately).
    - Model Evaluation for Gender prediction with events

```
Precision: 0.662968
Recall: 0.914298
F1 score: 0.768609
```

```
Logloss: 12.43
```

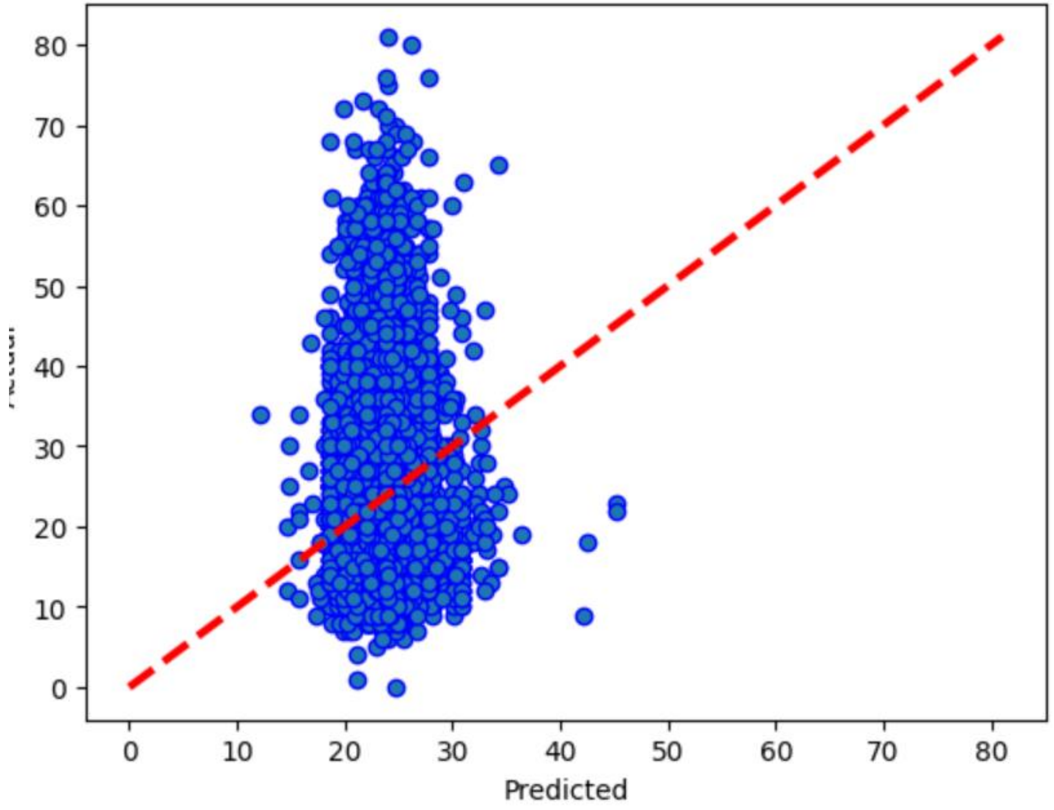




Model Building

- The outcomes of the evaluation metrics (results for both Scenario 1 and Scenario 2 must be shown separately).
  - Model Evaluation for Age prediction without events

```
The model performance for testing set
--------------------------------------
MAE is 7.435100520644608
MSE is 97.86113400826038
R2 score is -0.006406576024700961
```



Model Building

- The outcomes of the evaluation metrics (results for both Scenario 1 and Scenario 2 must be shown separately).
    - Model Evaluation for Age prediction with events

```
The model performance for testing set
--------------------------------------
MAE is 20.79857690426957
MSE is 533.7037308104443
R2 score is -4.276435738638995
```