AD campaign recommender

CAPSTONE – 01

Model Building

Praveen Selvaraj
Praveenselvaraj.ps@gmail.com
praveenselvaraj@arizona.edu
MS in Data Science
University of Arizona (Upgrad)

Index

Slide titles
Overall flow of Analysis
Reading the data
Data Cleaning
Feature Engineering
Exploratory Data Analysis
Data Preparation
Model Building
Scenario 1 – Gender Prediction
Scenario 1 – Age Group Prediction
Scenario 2 – Gender Prediction
Scenario 2 – Age Group Prediction
Scenario 1 – Model Performance
Scenario 2 – Model Performance

Overall Flow of the Analysis and Model

- Install required libraries through requirements.txt file
  - 1 cell hidden
- Import of Packages
  - [ ] 5 cells hidden
- Read Data
  - [ ] 44 cells hidden

- EDA and Visualization
  - Usage Through the week
    - [ ] 1 cell hidden
  - Usage Every Hour
    - [ ] 1 cell hidden
  - Usage Every Hour for each Gender
    - [ ] 1 cell hidden
  - Age Group vs Event Count Comparison. Requested Age Groups - 0-24, 25-32, 33-45, 46+
    - [ ] 4 cells hidden
  - Mapping Hour to Time of the Day - Morning, Afternoon, Evening, Night
    - [ ] 2 cells hidden
  - Getting Frequent Travellers
    - [ ] 2 cells hidden
  - Top 10 Mobile Applications
    - [ ] 3 cells hidden
  - Mapping Categories to High Level Categories
    - [ ] 19 cells hidden
  - Mobile Brand Data
    - [ ] 8 cells hidden
  - Cluster Based on Location Data using DBSCAN
    - [ ] 6 cells hidden
  - Geospatial visualization

- Data Preparation
  - Merge All necessary DataFrames
    - [ ] 2 cells hidden
  - Label Encoding
    - [ ] 1 cell hidden
  - Train Test Split
    - [ ] 2 cells hidden
  - Standard Scaler
    - [ ] 1 cell hidden
  - Target Encoding for "Gender Prediction"
    - [ ] 3 cells hidden

Flow of the Analysis and Model Building

Scenario 1

Model Building

Writing Model Functions

[ ] 9 cells hidden

Scenario 1 - Devices with Events

Gender Prediction

Gender Prediction - Logistic Regression

[ ] 3 cells hidden

Gender Prediction - Random Forest Classifier

[ ] 1 cell hidden

Gender Prediction - Random Forest Hyperparameter Tuning

Gender Prediction - Random Forest - Model 1

[ ] 1 cell hidden

Gender Prediction - Random Forest - Model 2

[ ] 1 cell hidden

Gender Prediction - Random Forest - Model 3

[ ] 1 cell hidden

Gender Prediction - Random Forest - Model 4

[ ] 1 cell hidden

Gender Prediction - Random Forest - Model 5

[ ] 1 cell hidden

Gender Prediction - Random Forest - Model Selection

[ ] 1 cell hidden

Gender Prediction - Random Forest - Best Parameters

[ ] 1 cell hidden

Gender Prediction - XG Boost

16 cells hidden

Gender Prediction - Model Stacking

[ ] 8 cells hidden

Gender Prediction - Final Model Selection

[ ] 1 cell hidden

Gender Prediction - Exporting Model to a Pickle File Scenario 1

[ ] 1 cell hidden

Age Group Prediction

Age Group Prediction - Data Preparation

[ ] 2 cells hidden

Age Group Prediction - Logistic Regression

[ ] 2 cells hidden

Age Group Prediction - Logistic Model Selection

[ ] 1 cell hidden

Age Group Prediction - Random Forest Classifier

[ ] 1 cell hidden

Age Group Prediction - Random Forest Hyperparameter Tuning

Age Group Prediction - Random Forest Model 1

[ ] 1 cell hidden

Age Group Prediction - Random Forest Model 2

[ ] 1 cell hidden

Age Group Prediction - Random Forest Model 3

[ ] 1 cell hidden

Age Group Prediction - Random Forest Model 4

[ ] 1 cell hidden

Age Group Prediction - Random Forest Model 5

[ ] 1 cell hidden

Age Group Prediction - Random Forest Model Selection

[ ] 1 cell hidden

Age Group Prediction - Random Forest - Best Parameters

[ ] 1 cell hidden

Age Group Prediction - XG Boost

[ ] 1 cell hidden

Age Group Prediction - XG Boost Hyperparameter Tuning

[ ] 14 cells hidden

Age Group Prediction - Model Stacking

[ ] 8 cells hidden

Age Group Prediction - Final Model Selection

[ ] 1 cell hidden

Age Group Prediction - Exporting Model to a Pickle File Scenario 1

[ ] 1 cell hidden

Flow of the Analysis and Model Building

Scenario 2

- Scenario 2 - Devices without Events
  - Data Preparation
    - 6 cells hidden
  - Label Encoding
    - [ ] 1 cell hidden
  - Train Test Split
    - [ ] 2 cells hidden
  - Target Encoder for Gender Prediction
    - [ ] 2 cells hidden
- Gender Prediction
  - Gender Prediction - Logistic Regression
    - [ ] 5 cells hidden
  - Gender Prediction - Random Forest Classifier
    - [ ] 1 cell hidden
  - Gender Prediction - Random Forest Hyperparameter Tuning
    - Gender Prediction - Random Forest - Model 1
      - [ ] 1 cell hidden
    - Gender Prediction - Random Forest - Model 2
      - [ ] 1 cell hidden
    - Gender Prediction - Random Forest - Model 3
      - [ ] 1 cell hidden
    - Gender Prediction - Random Forest - Model Selection
      - [ ] 1 cell hidden
    - Gender Prediction - Random Forest - Best Parameters
      - 1 cell hidden
  - Gender Prediction - XG Boost
    - [ ] 12 cells hidden
  - Gender Prediction - Model Stacking
    - [ ] 6 cells hidden
  - Gender Prediction - Final Model Selection
    - [ ] 1 cell hidden
  - Gender Prediction - Exporting Model to a Pickle File Scenario 2
    - [ ] 1 cell hidden
- Age Group Prediction
  - Age Group Prediction - train test Split
    - [ ] 1 cell hidden
  - Age Group Prediction - Logistic Regression
    - [ ] 2 cells hidden
  - Age Group Prediction - Logistic Model Selection
    - [ ] 1 cell hidden
  - Age Group Prediction - Random Forest Classifier
    - [ ] 1 cell hidden
  - Age Group Prediction - Random Forest Hyperparameter Tuning
    - [ ] 10 cells hidden
  - Age Group Prediction - XG Boost
    - [ ] 1 cell hidden
  - Age Group Prediction - XG Boost Hyperparameter Tuning
    - [ ] 10 cells hidden
  - Age Group Prediction - Model Stacking
    - [ ] 8 cells hidden
  - Age Group Prediction - Final Model Selection
    - [ ] 1 cell hidden

Reading the data

Read Data

```
dfAppEvents = pd.read_csv("https://uoacapstone.s3.amazonaws.com/app_events.csv")

dfTrainEvents = pd.read_csv("https://uoacapstone.s3.amazonaws.com/train_event_data.csv")

dfMetaAppEvents = pd.read_csv("https://uoacapstone.s3.amazonaws.com/app_events_meta_data.csv",quoting=csv.QUOTE_NONE)

dfMobileBrandTrain = pd.read_csv("https://uoacapstone.s3.amazonaws.com/train_mobile_brand.csv")
```

Renaming columns in each DataFrame ¶

```
dfMetaAppEvents.columns = ["AppID", "LabelID", "Category"]

dfAppEvents.columns = ["EventID", "AppID", "IsInstalled", "IsActive"]

dfTrainEvents.columns = ["DeviceID", "Gender", "Age", "GroupTrain", "EventID", "DateTimestamp", "Latitude", "Longitude"]

dfMobileBrandTrain.columns = ["DeviceID", "Gender", "Age", "GroupTrain", "MobilePhoneBrand", "DeviceModel"]
```

```
dfMetaAppEvents.columns, dfAppEvents.columns, dfTrainEvents.columns, dfMobileBrandTrain.columns
```

```
(Index(['AppID', 'LabelID', 'Category'], dtype='object'),
 Index(['EventID', 'AppID', 'IsInstalled', 'IsActive'], dtype='object'),
 Index(['DeviceID', 'Gender', 'Age', 'GroupTrain', 'EventID', 'DateTimestamp',
        'Latitude', 'Longitude'],
       dtype='object'),
 Index(['DeviceID', 'Gender', 'Age', 'GroupTrain', 'MobilePhoneBrand',
        'DeviceModel'],
       dtype='object'))
```

Basic Stats on the Datasets

```
dfMetaAppEvents.head()
```

	app_id	label_id	category
0	app_id	label_id	category
1	7324884708820027918	251	Finance
2	-4494216993218550286	251	Finance
3	6058196446775239644	406	unknown
4	6058196446775239644	407	DS_P2P net loan

```
dfAppEvents.head()
```

	event_id	app_id	is_installed	is_active
0	2	5927333115845830913	1	1
1	2	-5720078949152207372	1	0
2	2	-1633887856876571208	1	0
3	2	-653184325010919369	1	1
4	2	8693964245073640147	1	1

```
dfTrainEvents.head()
```

	device_id	gender	age	group_train	event_id	datetimestamp	latitude	longitude
0	-7548291590301750000	M	33	M32+	2369465.0	2016-05-03 15:55:35	33.98	116.79
1	-7548291590301750000	M	33	M32+	1080869.0	2016-05-03 06:07:16	33.98	116.79
2	-7548291590301750000	M	33	M32+	1079338.0	2016-05-04 03:28:02	33.98	116.79
3	-7548291590301750000	M	33	M32+	1078881.0	2016-05-04 02:53:08	33.98	116.79
4	-7548291590301750000	M	33	M32+	1068711.0	2016-05-03 15:59:35	33.98	116.79

```
dfMobileBrandTrain.head()
```

	device_id	gender	age	group_train	phone_brand	device_model
0	-7548291590301750000	M	33	M32+	Huawei	è□£è€€3C
1	6943568600617760000	M	37	M32+	Xiaomi	xnote
2	5441349705980020000	M	40	M32+	OPPO	R7s
3	-5393876656119450000	M	33	M32+	Xiaomi	MI 4
4	4543988487649880000	M	53	M32+	samsung	Galaxy S4

Data Cleaning

MetaAppEvents ¶

```
dfMetaAppEvents = dfMetaAppEvents.iloc[1:,:].reset_index(drop=True)
dfMetaAppEvents.head(3)
```

	AppID	LabelID	Category
0	7324884708820027918	251	Finance
1	-4494216993218550286	251	Finance
2	6058196446775239644	406	unknown

```
dfMetaAppEvents.AppID = dfMetaAppEvents.AppID.astype("int64")

dfMetaAppEvents.LabelID = dfMetaAppEvents.LabelID.astype("int64")
```

```
dfMetaAppEvents.Category = dfMetaAppEvents.Category.str.upper()
```

MobileBrandTrain

```
dfMobileBrandTrain.head()
```

	DeviceID	Gender	Age	GroupTrain	MobilePhoneBrand	DeviceModel
0	-7548291590301750000	M	33	M32+	Huawei	è□£è€€3C
1	6943568600617760000	M	37	M32+	Xiaomi	xnote
2	5441349705980020000	M	40	M32+	OPPO	R7s
3	-5393876656119450000	M	33	M32+	Xiaomi	MI 4
4	4543988487649880000	M	53	M32+	samsung	Galaxy S4

Dealing with Junk Data such as in the 1st row in column DeviceModel - è□£è€€3C

```
#Function to Clean Gibberish Data
def cleanJunk(str):
    if str.isascii():
        return str

    else:
        a="".join(list(map(lambda x:x if x.isascii() else "",str)))
        return a

dfMobileBrandTrain["DeviceModel"] = dfMobileBrandTrain["DeviceModel"].apply(cleanJunk)
dfMobileBrandTrain.head()
```

	DeviceID	Gender	Age	GroupTrain	MobilePhoneBrand	DeviceModel
0	-7548291590301750000	M	33	M32+	Huawei	3C
1	6943568600617760000	M	37	M32+	Xiaomi	xnote
2	5441349705980020000	M	40	M32+	OPPO	R7s
3	-5393876656119450000	M	33	M32+	Xiaomi	MI 4
4	4543988487649880000	M	53	M32+	samsung	Galaxy S4

AppEvents Data

All Rows seem to have 1 value for column - IsInstalled. Will drop this column

```
dfAppEvents = dfAppEvents[["EventID", "AppID", "IsActive"]]
dfAppEvents.nunique()
```

```
EventID     1488096
AppID         19237
IsActive          2
dtype: int64
```

TrainEvents Data

Format the type for DateTimestamp

```
dfTrainEvents["DateTimestamp"]=pd.to_datetime(dfTrainEvents["DateTimestamp"])
```

Feature Engineering

Format the type for DateTimestamp

```
dfTrainEvents["DateTimestamp"]=pd.to_datetime(dfTrainEvents["DateTimestamp"])
```

Create new features from DateTimestamp Column, such as Hour, WeekDay, DayName and WeekNum

```
dfTrainEvents["Hour"] = dfTrainEvents.DateTimestamp.dt.hour

dfTrainEvents["WeekDay"] = dfTrainEvents.DateTimestamp.dt.day_of_week

dfTrainEvents["DayName"] = dfTrainEvents.DateTimestamp.dt.day_name()

dfTrainEvents["WeekNum"] = dfTrainEvents.DateTimestamp.dt.weekofyear
```

```
print("Total Number of Records without Events ", "\t- ", dfTrainEvents.EventID.isnull().sum())

print("Total Number of Records with Events ", "\t\t- ", dfTrainEvents.EventID[~(dfTrainEvents.EventID.isnull())].shape[0])

print("Total Number of Records ", "\t\t\t- ", dfTrainEvents.shape[0])

print("Percent of Devices without Events ", "\t\t- ",
        round(100*dfTrainEvents.EventID.isnull().sum()
              /dfTrainEvents.EventID[~(dfTrainEvents.EventID.isnull())].shape[0], 2), "%")
```

```
Total Number of Records without Events       -  51335
Total Number of Records with Events          -  1215598
Total Number of Records                      -  1266933
Percent of Devices without Events            -  4.22 %
```

Create a DataFrame where the EventIDs are not null

```
dfTrainEventsWithEventData = dfTrainEvents[~(dfTrainEvents.EventID.isnull())]

dfTrainEventsWithEventData.shape[0]
```

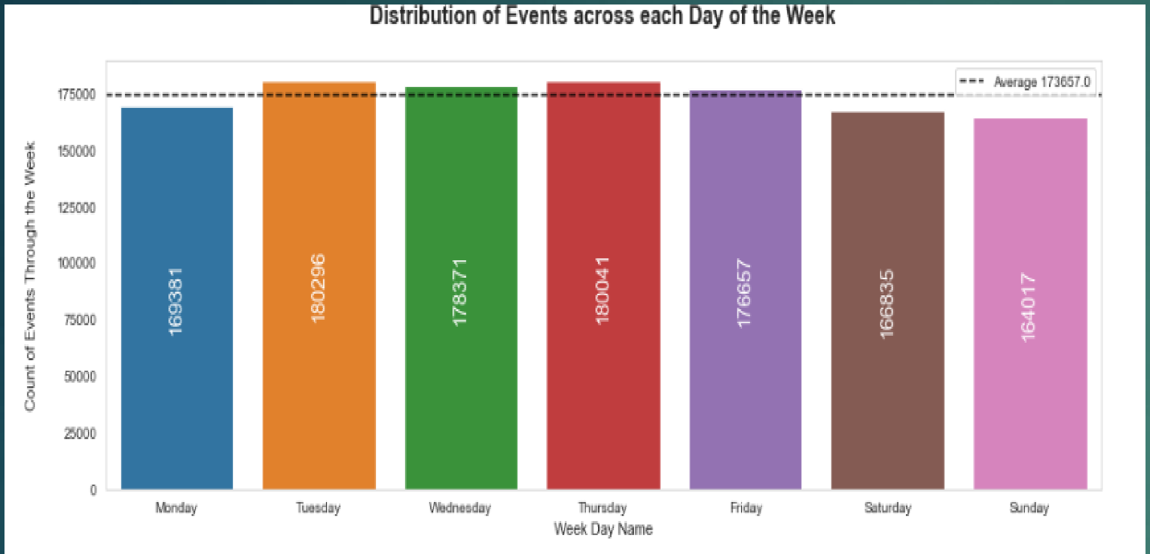
```
1215598
```

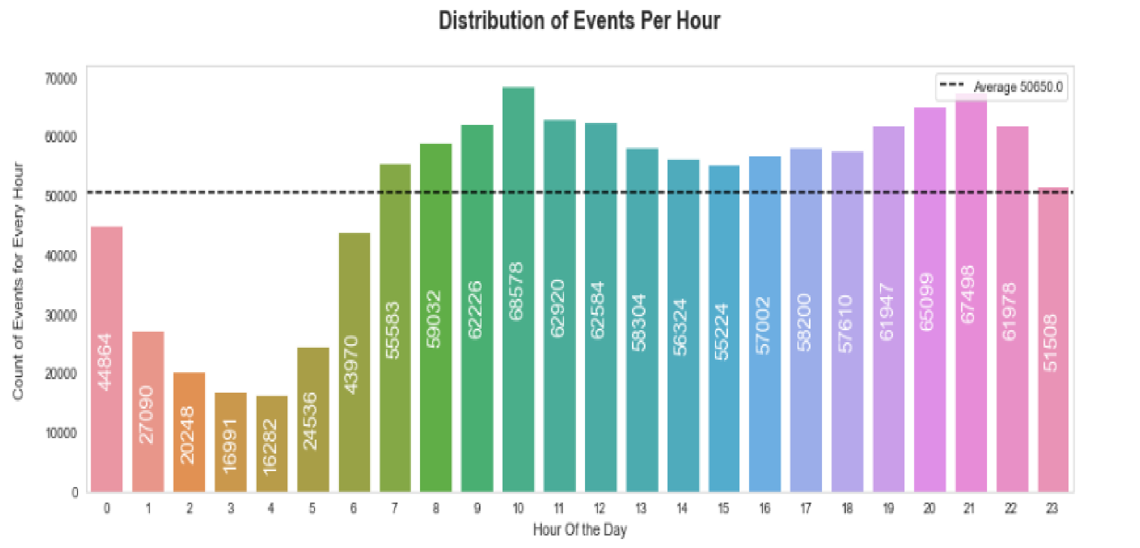
```
dfTrainEventsWithEventData["Coordinates"] = dfTrainEventsWithEventData[["Latitude", "Longitude"]]
    .apply(lambda x: (x["Latitude"], x["Longitude"]), axis = 1)

dfTrainEventsWithEventData.head()
```

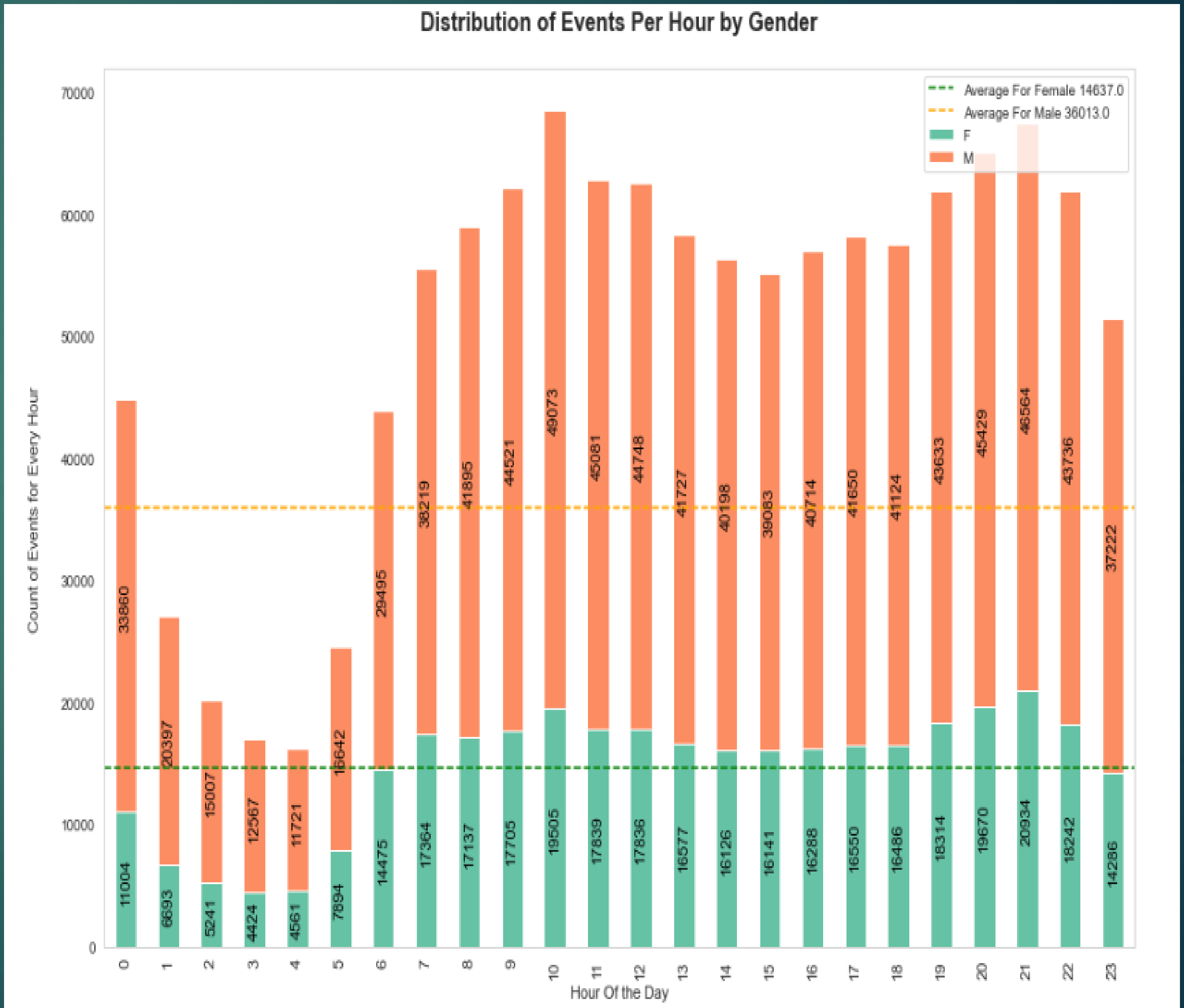
	DeviceID	Gender	Age	GroupTrain	EventID	DateTimestamp	Latitude	Longitude	Hour	WeekDay	DayName	WeekNum	Coordinates
0	-7548291590301750000	M	33	M32+	2369465.0	2016-05-03 15:55:35	33.98	116.79	15.0	1.0	Tuesday	18.0	(33.98, 116.79)
1	-7548291590301750000	M	33	M32+	1080869.0	2016-05-03 06:07:16	33.98	116.79	6.0	1.0	Tuesday	18.0	(33.98, 116.79)
2	-7548291590301750000	M	33	M32+	1079338.0	2016-05-04 03:28:02	33.98	116.79	3.0	2.0	Wednesday	18.0	(33.98, 116.79)
3	-7548291590301750000	M	33	M32+	1078881.0	2016-05-04 02:53:08	33.98	116.79	2.0	2.0	Wednesday	18.0	(33.98, 116.79)
4	-7548291590301750000	M	33	M32+	1068711.0	2016-05-03 15:59:35	33.98	116.79	15.0	1.0	Tuesday	18.0	(33.98, 116.79)

```
dfTrainEventsWithEventData.EventID = dfTrainEventsWithEventData.EventID.astype("int64")

dfTrainEventsWithEventData.Hour = dfTrainEventsWithEventData.Hour.astype("int64")

dfTrainEventsWithEventData.WeekDay = dfTrainEventsWithEventData.WeekDay.astype("int64")

dfTrainEventsWithEventData.WeekNum = dfTrainEventsWithEventData.WeekNum.astype("int64")

dfTrainEventsWithEventData.info()
```

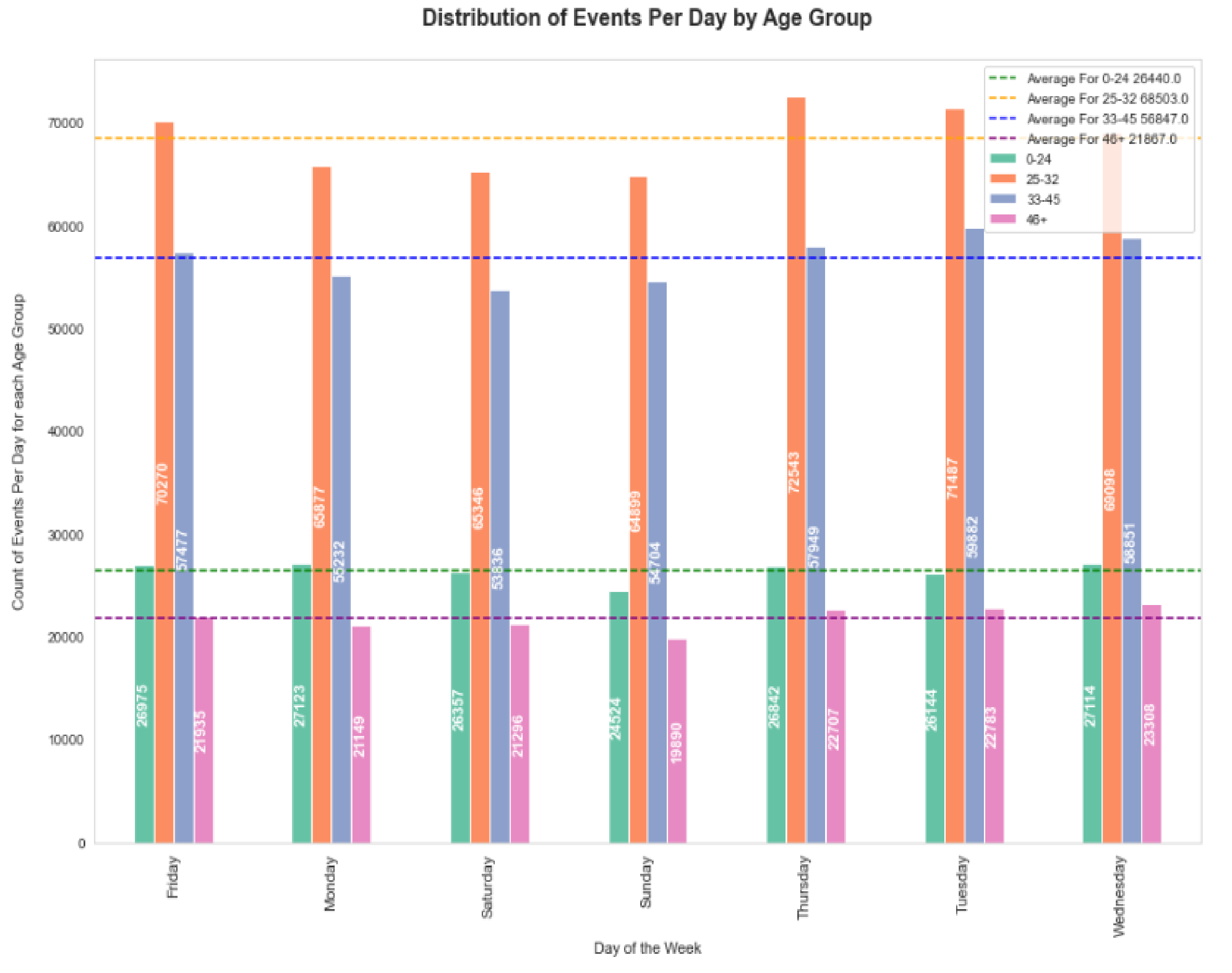
Exploratory Data Analysis – Slide 1

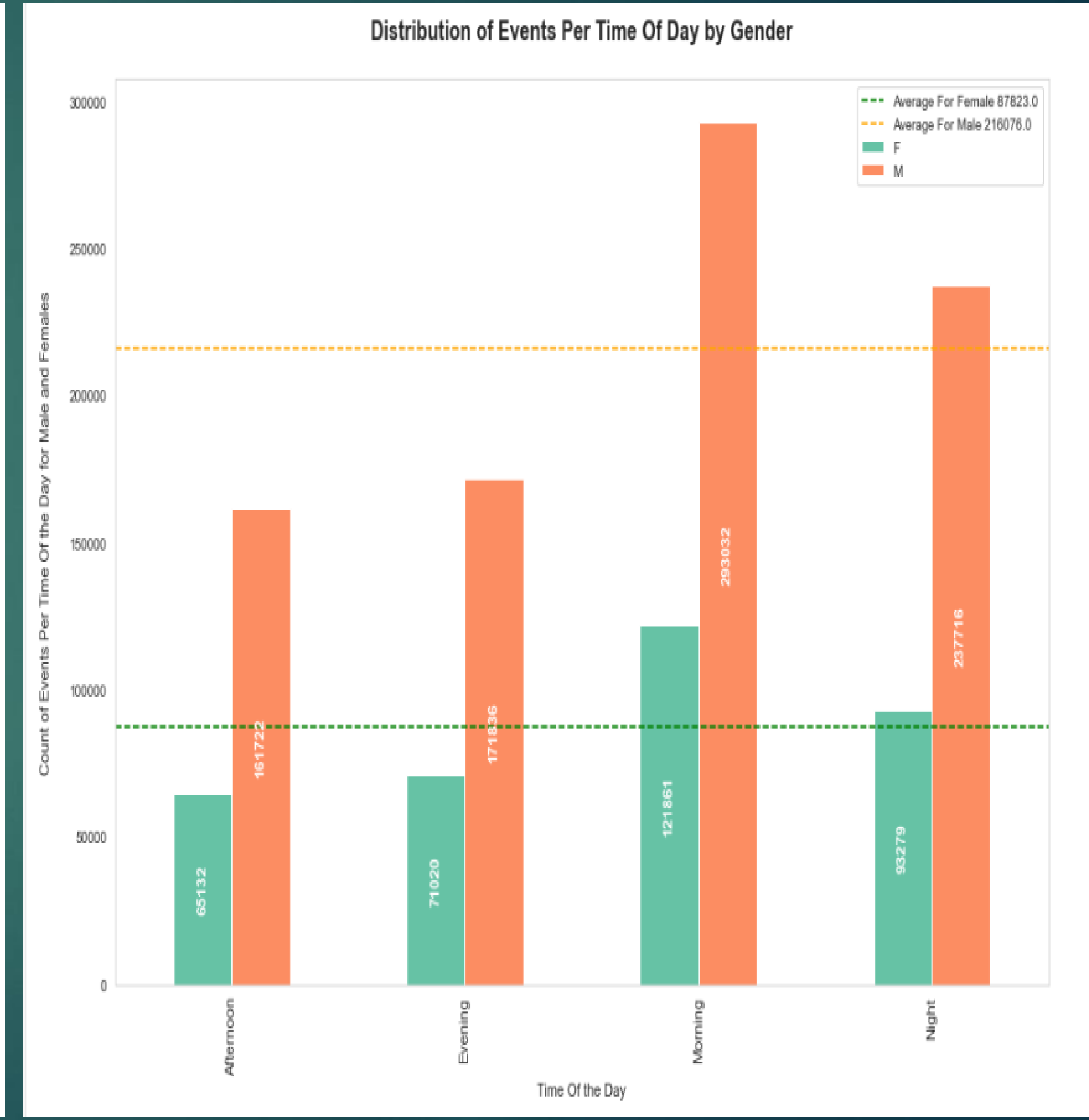




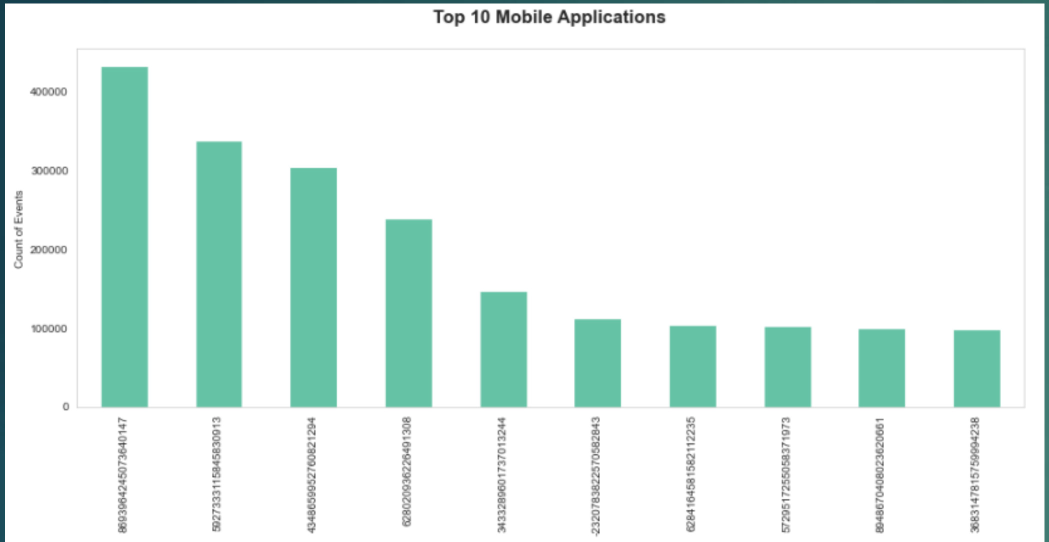


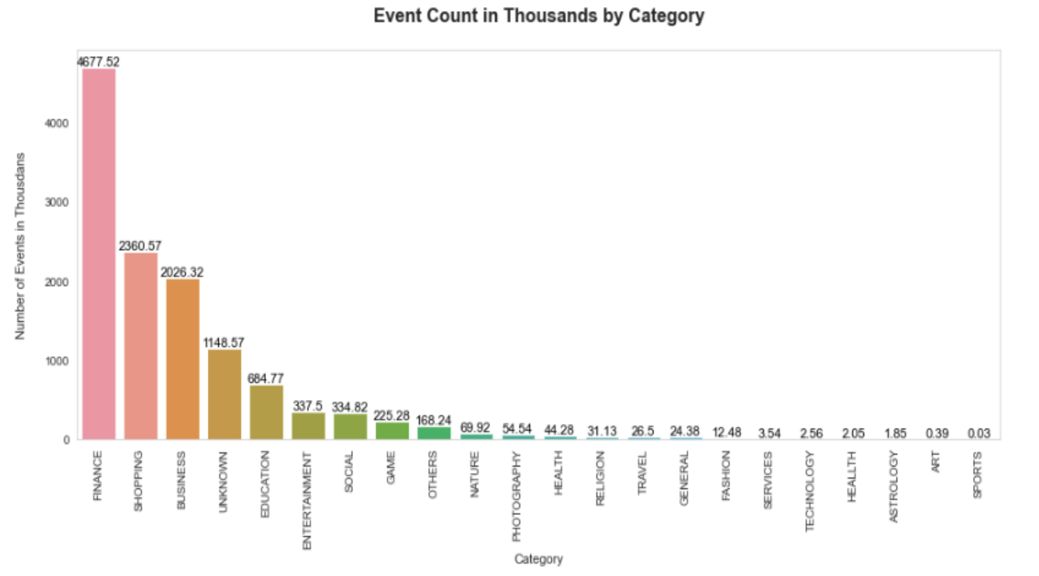
Exploratory Data Analysis – Slide 2

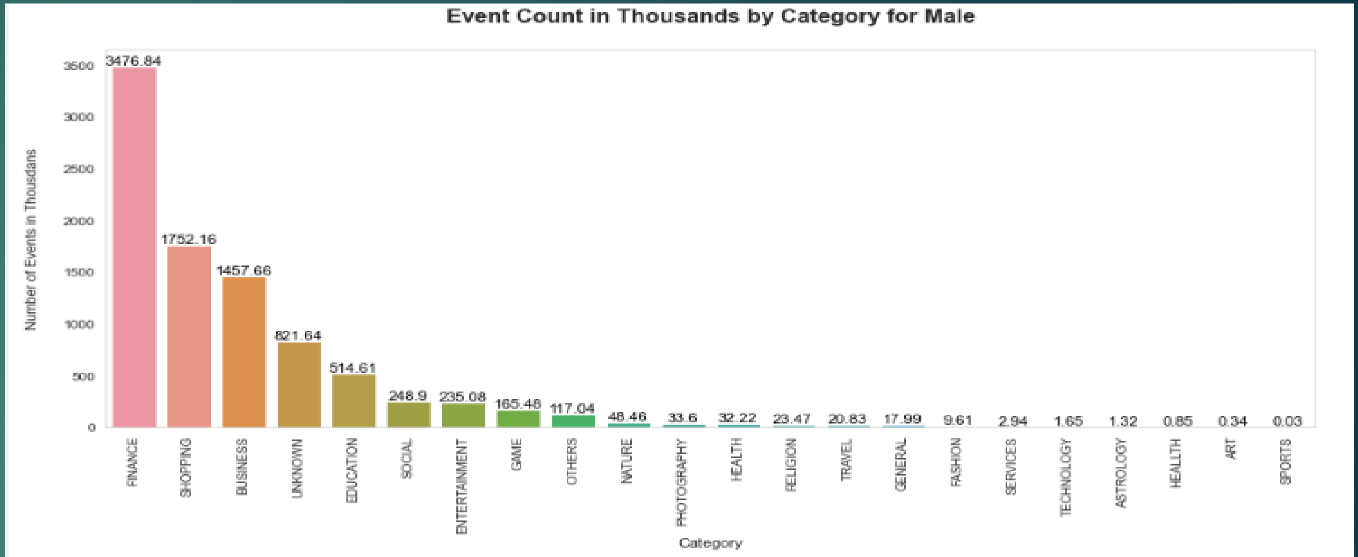


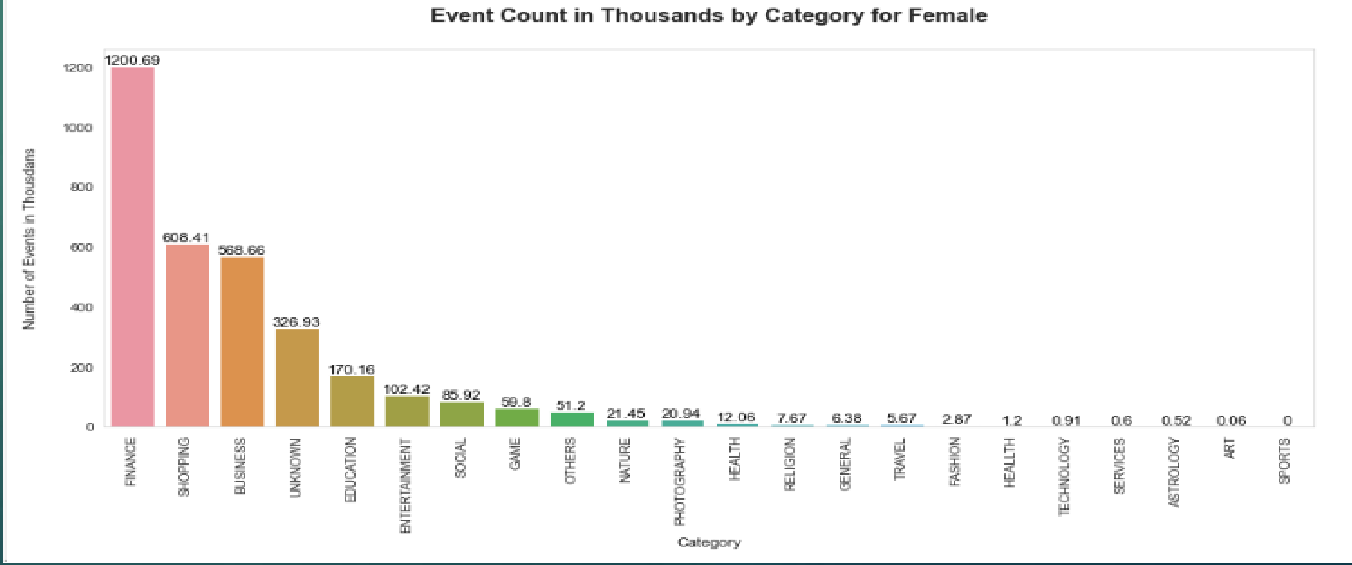


Exploratory Data Analysis – Slide 3

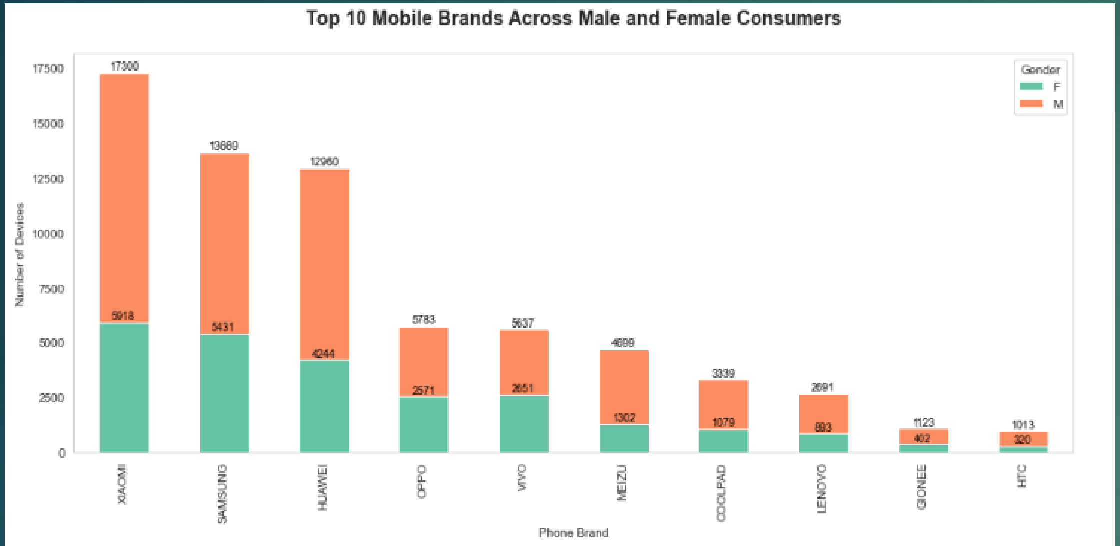


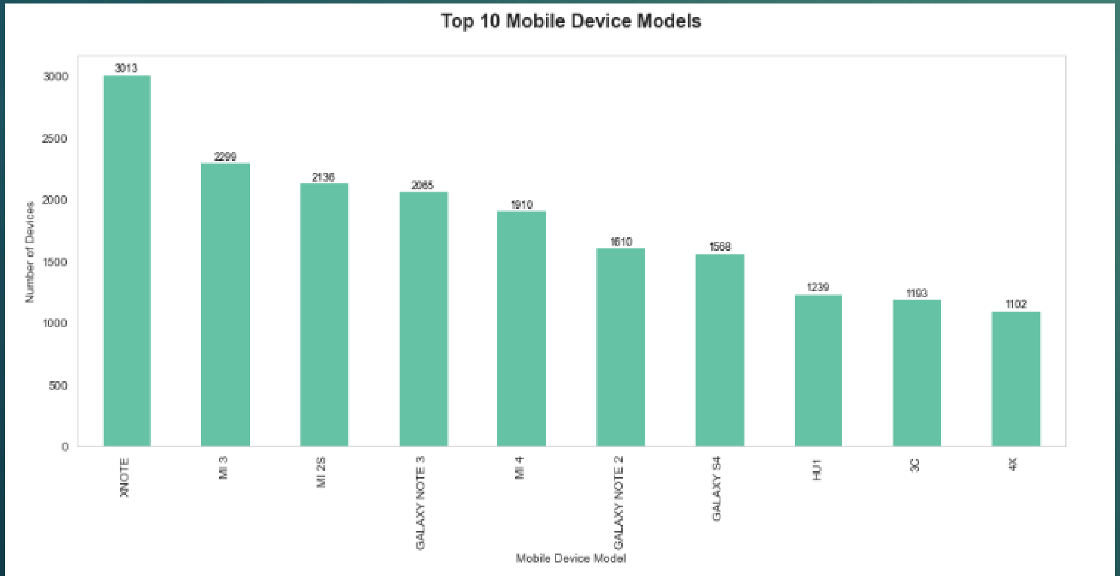


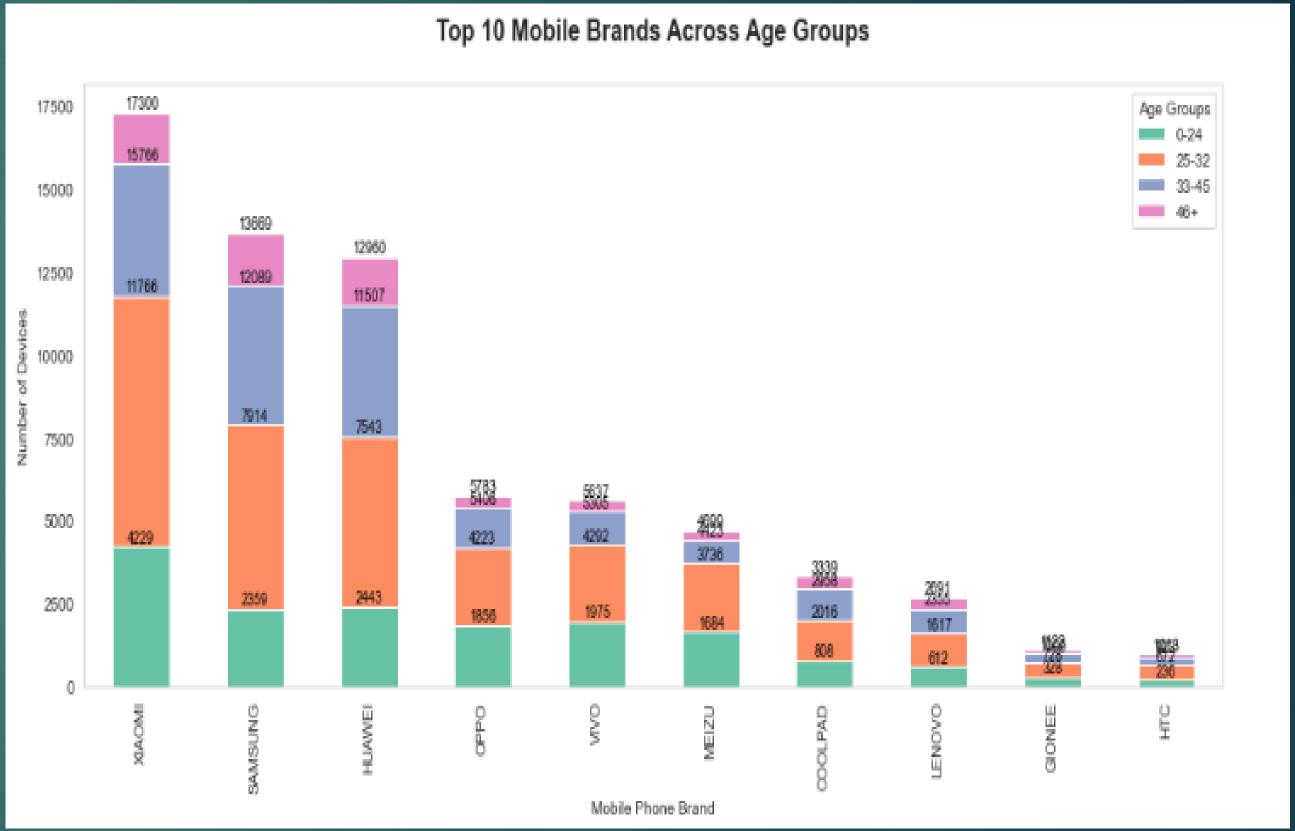




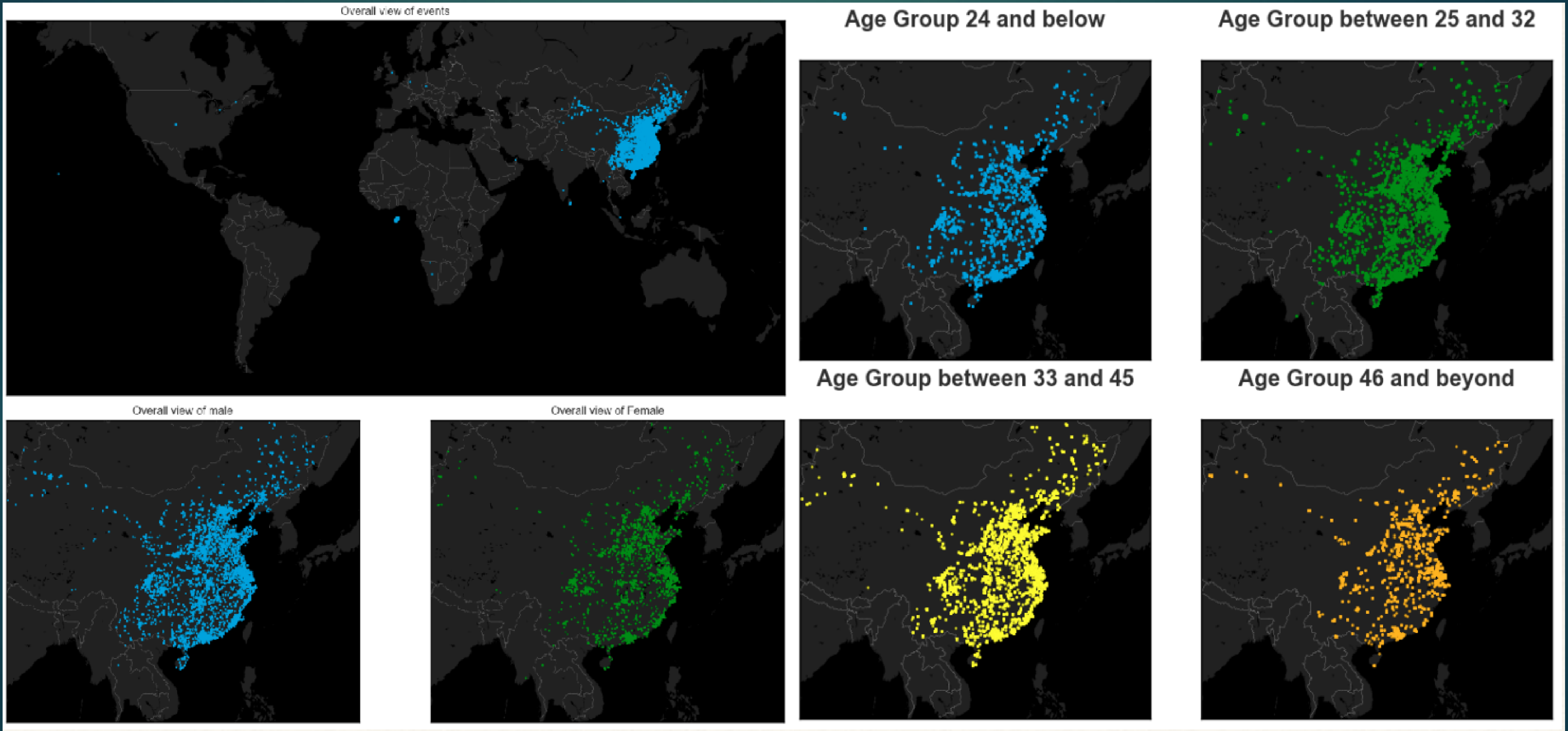
Exploratory Data Analysis – Slide 4



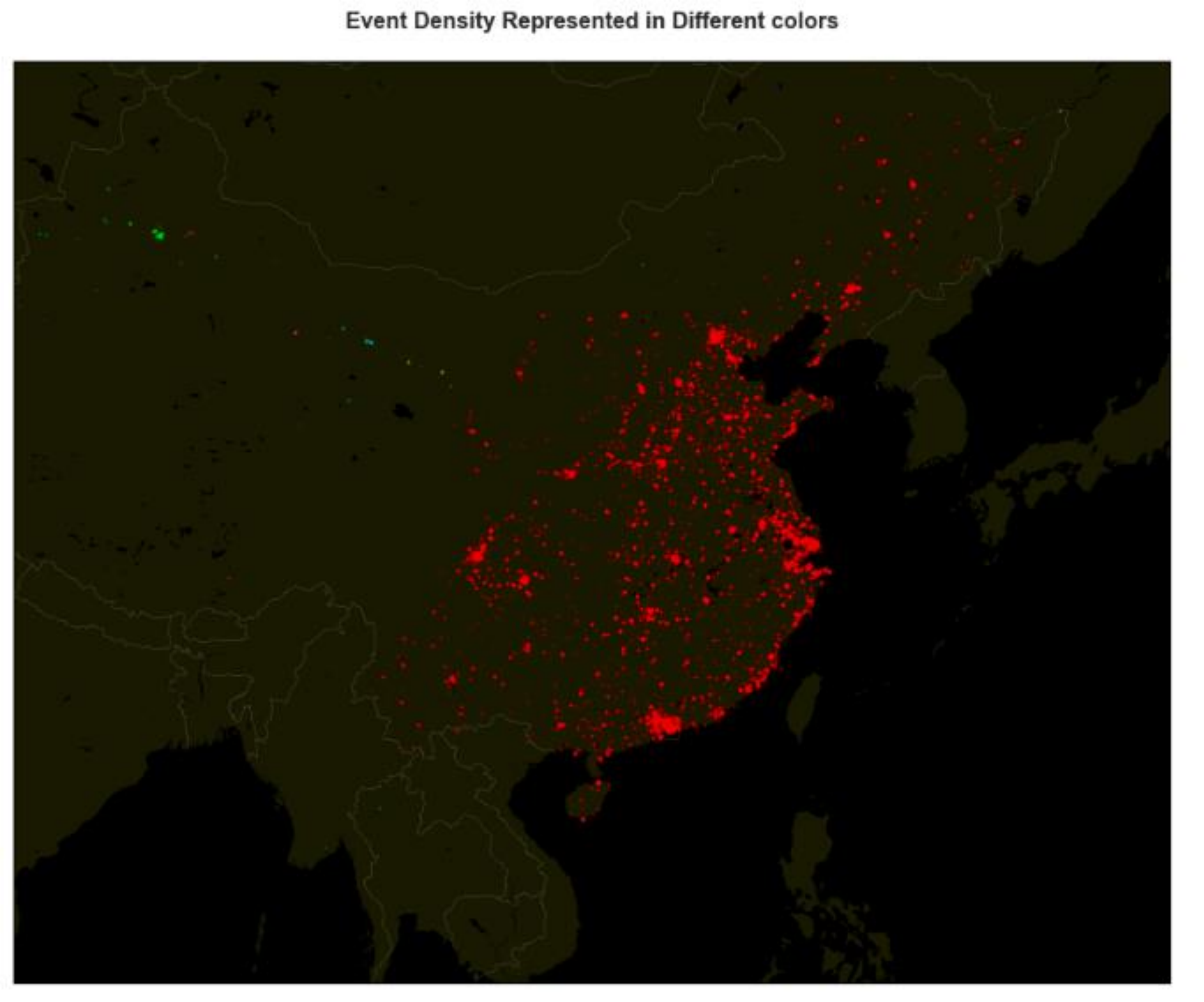




Exploratory Data Analysis – Slide 5



Exploratory Data Analysis – Slide 6



Data Preparation

Merge All necessary DataFrames ¶

```
dfDeviceData = dfTrainEventsWithEventData[["DeviceID", "Gender", "AgeGroup"]].drop_duplicates()
dfDeviceEventsMerged = pd.merge(dfDeviceData, dfDeviceLocationTimeOfDay, how="left", on="DeviceID")
dfDeviceEventsMerged = pd.merge(dfDeviceEventsMerged, dfDeviceTopCategoryApp, how="left", on="DeviceID")
dfDeviceEventsMerged = pd.merge(dfDeviceEventsMerged, dfDeviceCluster, how="left", on="DeviceID")
dfDeviceEventsMerged = pd.merge(dfDeviceEventsMerged, dfDeviceEventCount, how="left", on="DeviceID")
dfDeviceEventsMerged = pd.merge(dfDeviceEventsMerged, dfMobileBrand, how="left", on="DeviceID")

print(dfDeviceEventsMerged.shape)

dfDeviceEventsMerged.drop(["Morning", "Evening"], axis=1, inplace=True)

dfDeviceEventsMerged["TravellerType"].fillna("Unknown", inplace=True)

dfDeviceEventsMerged["Cluster"].fillna(-99,inplace=True)

dfDeviceEventsMerged["Cluster"] = dfDeviceEventsMerged["Cluster"].astype("category")

dfDeviceEventsMerged.head()
```

(23310, 11)

	DeviceID	Gender	AgeGroup	TravellerType	HighLevelCategory	Cluster	EventCount	MobilePhoneBrand	DeviceModel
0	-7548291590301750000	M	33-45	Infrequent	BUSINESS	0.0	292	HUAWEI	3C
1	6943568600617760000	M	33-45	Unknown	FINANCE	-99.0	1	XIAOMI	XNOTE
2	5441349705980020000	M	33-45	Unknown	FINANCE	-99.0	1	OPPO	R7S
3	-5393876656119450000	M	33-45	Unknown	FINANCE	-99.0	4	XIAOMI	MI 4
4	4543988487649880000	M	46+	Frequent	FINANCE	0.0	115	SAMSUNG	GALAXY S4

Train Test Split

```
dfTrainTest = pd.read_csv("train_test_split.csv")

dfTrainTest.columns = ["DeviceID", "Gender", "Age", "Group", "TrainTestFlag"]
dfTrainTest.head()
```

	DeviceID	Gender	Age	Group	TrainTestFlag
0	-7548291590301750000	M	33	M32+	train
1	6943568600617760000	M	37	M32+	train
2	5441349705980020000	M	40	M32+	train
3	-5393876656119450000	M	33	M32+	train
4	4543988487649880000	M	53	M32+	train

Label Encoding

```
#Defining Target Columns
target_columns = ['Gender', 'AgeGroup']

# Create an instance of the LabelEncoder
encoder = LabelEncoder()

# Apply label encoding on the selected columns
dfDeviceEventsMergedEncoded = dfDeviceEventsMerged.copy()
label_mappings = {}

for column in target_columns:
    encoded_labels = encoder.fit_transform(dfDeviceEventsMerged[column])
    dfDeviceEventsMergedEncoded[column] = encoded_labels
    label_mappings[column] = dict(zip(encoder.classes_, encoder.transform(encoder.classes_)))

# Print the mappings for each column
for column, mappings in label_mappings.items():
    print(f"Label Mappings for column '{column}':")
    for label, encoded_label in mappings.items():
        print(f"{label} : {encoded_label}")
    print()
```

```
Label Mappings for column 'Gender':
F : 0
M : 1

Label Mappings for column 'AgeGroup':
0-24 : 0
25-32 : 1
33-45 : 2
46+ : 3
```

Standard Scaler

```
num_columns=["EventCount"]

# Apply StandardScaler
scaler = StandardScaler()
dfTrain[num_columns] = scaler.fit_transform(dfTrain[num_columns].values)
dfTest[num_columns] = scaler.transform(dfTest[num_columns].values)
# Replace numerical columns with scaled values in the DataFrame
```

Target Encoding for "Gender Prediction"

```
target_columns = ['Gender']

cat_columns = ["TravellerType", "HighLevelCategory", "Cluster", "MobilePhoneBrand", "DeviceModel"]
encoder = ce.TargetEncoder(cols=cat_columns)

encoder.fit(dfTrain, dfTrain["Gender"])
# train_df.shape


dfTrain = encoder.transform(dfTrain)
dfTest = encoder.transform(dfTest)
```

Model Building - 1

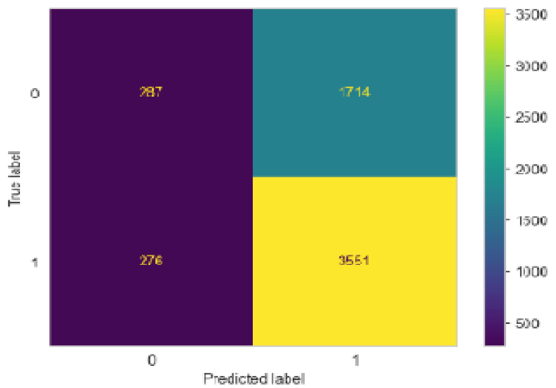
- Scenario 1 – With Events
  - Prediction - **Gender**
  - Models :–
    - Logistic
    - Random Forest
    - XG Boost
    - Model Stacking

- Scenario 1 - With Events
  - Prediction -A**geGroup**
  - Models :–
    - Logistic
    - Random Forest
    - XG Boost
    - Model Stacking

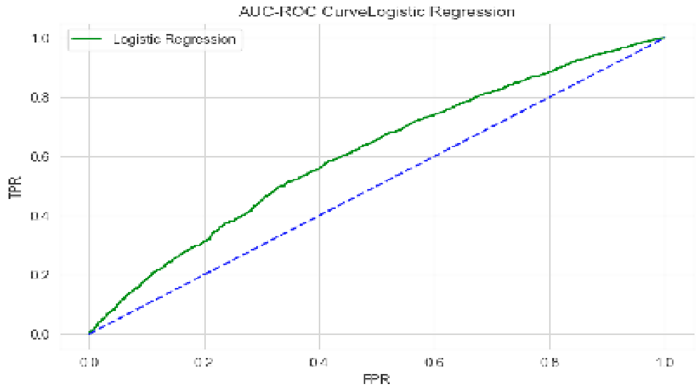
Scenario 1: Gender Prediction, Logistic Reg

```
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_gender_1 = logistic_regression(X_train,
        y_gender_train,X_test,y_gender_test)

evaluation_metrics(lr_gender_1,X_test,y_gender_test,"Logistic Regression",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6535449553877831
Gender Model training Accuracy: 0.6712046676581627
Gender Model Precision: 0.6744539411206077
Gender Model Recall: 0.9278308456161484
Gender Model F1 Score: 0.7811262648482182
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```

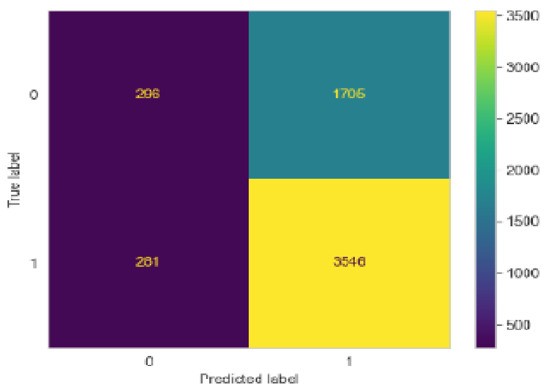



```
---------------Decile Analysis---------------
Top Probability: 0.5555661857200138
Bottom Probability: 0.6351541775130005
KS Statistic
Decile 0: 0.0006009929448654305
Decile 1: 0.0008884243532279349
Decile 2: 0.005252155735563091
Decile 3: 5.226025607524276e-05
Decile 4: 0.002743663443950834
Decile 5: 0.002926574340214260
Decile 6: 0.002064280114972505
Decile 7: 0.004860203314998742
Decile 8: 0.002691403187875640
Decile 9: 0.001045205121505077
---------------ROC Curve---------------
ROC AUC SCORE ->  0.6087205286826145
```

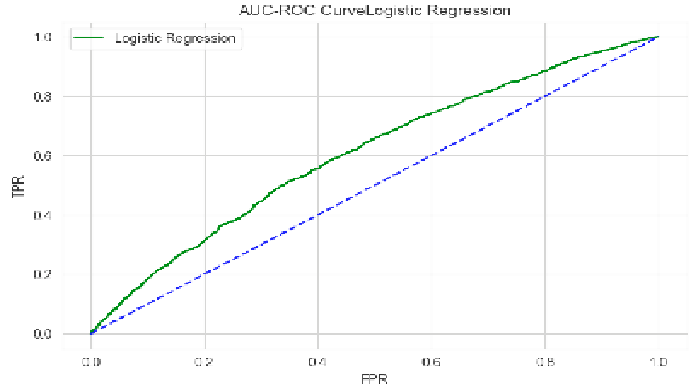



```
#Applying L1 Regularization
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_gender_2 = logistic_regression(X_train,
        y_gender_train,X_test,y_gender_test,penalty='l1', solver='liblinear')
evaluation_metrics(lr_gender_2,X_test,y_gender_test,"Logistic Regression",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6592312971859985
Gender Model training Accuracy: 0.6721193947483846
Gender Model Precision: 0.6752999428680231
Gender Model Recall: 0.9265743402142671
Gender Model F1 Score: 0.7812293456708525
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Decile Analysis---------------
Top Probability: 0.5540670318181501
Bottom Probability: 0.6336251419362263
KS Statistic
Decile 0: 0.0006009929448654305
Decile 1: 0.0008884243532279349
Decile 2: 0.005252155735563091
Decile 3: 5.226025607524276e-05
Decile 4: 0.002743663443950834
Decile 5: 0.002926574340214260
Decile 6: 0.002064280114972505
Decile 7: 0.004860203814998742
Decile 8: 0.002691403187875640
Decile 9: 0.001045205121505077
---------------ROC Curve---------------
ROC AUC SCORE ->  0.607403575452932
```



Scenario 1 : Gender Prediction, Random Forest

Gender Prediction - Random Forest - Model 5

```python
param_grid={
            "n_estimators":[80],
           "min_samples_split":[2,5],
           "min_samples_leaf":[5],
           "max_leaf_nodes":[80,100],
           "max_depth":[50],
           "oob_score":[True],
            "bootstrap":[True]

           }

y_gender_pred,accuracy_gender,training_accuracy_gender,cv_gender,best_score_gender,best_params_gender_5 =
        cross_validation(X_train,y_gender_train,X_test,y_gender_test,rf_gender_1,param_grid)
print("Gender Model Accuracy:",accuracy_gender)
print("Gender Model training Accuracy:",training_accuracy_gender)
print("Best CV Score:",best_score_gender)
print("Best Paramater:",best_params_gender_5)
```

```
Gender Model Accuracy: 0.6573438572409059
Gender Model training Accuracy: 0.6912824619608741
Best CV Score: 0.6724631452986097
Best Paramater: {'bootstrap': True, 'max_depth': 50, 'max_leaf_nodes': 100, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 80, 'oob_score': True}
```

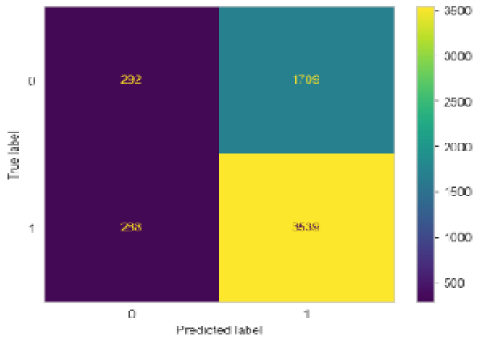
Gender Prediction - Random Forest - Model Selection

```python
# Since Model 5 has best acccuracy score taking model 5 as final model
best_params_gender=best_params_gender_5
```

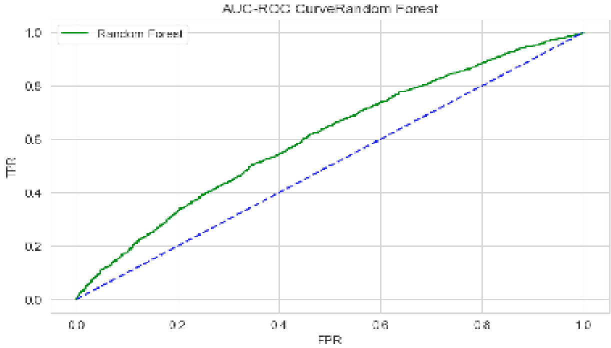
Gender Prediction - Random Forest - Best Parameters

```python
#Best Random Forest Model

y_pred,accuracy,training_accuracy,recall,precision,F1,rf_gender = random_forest(X_train,y_gender_train,X_test,
        y_gender_test,**best_params_gender)
evaluation_metrics(rf_gender,X_test,y_gender_test,"Random Forest",accuracy,training_accuracy,recall,precision,
        F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6573438572409059
Gender Model training Accuracy: 0.6912824619608741
Gender Model Precision: 0.6743521341463414
Gender Model Recall: 0.9247452312516331
Gender Model F1 Score: 0.7799449035812672
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Decile Analysis---------------
Top Probability: 0.5565927512322382
Bottom Probability: 0.6370836299390606
KS Statistic
Decile 0: 0.0006009929448654305
Decile 1: 0.000888424353279349
Decile 2: 0.005252155735563091
Decile 3: 5.226025607524276e-05
Decile 4: 0.0027436634439508834
Decile 5: 0.0029255743402142608
Decile 6: 0.0020642801149725054
Decile 7: 0.0048607038149987475
Decile 8: 0.0026914031878756406
Decile 9: 0.0018452051215050773
---------------ROC Curve---------------
ROC AUC SCORE -> 0.6056710343354785
```



Scenario 1 : Gender Prediction, X G Boost

Gender Prediction - XG Boost Tuning Model 5

```
param_grid={"n_estimators":[50],
            "max_depth":[2,3],
            "learning_rate":[0.3,0.4],
             'min_child_weight': [3,4],
             'gamma': [0.2, 0.3]


            }

y_gender_pred,accuracy_gender,training_accuracy_gender,cv_gender,best_score_gender,best_params_gender_5 =
        cross_validation(X_train,y_gender_train,X_test,y_gender_test,xgb_gender_1,param_grid)
print("Gender Model Accuracy:",accuracy_gender)
print("Gender Model training Accuracy:",training_accuracy_gender)
print("Best CV Score:",best_score_gender)
print("Best Paramater:",best_params_gender_5)
```

```
Gender Model Accuracy: 0.6557995881949211
Gender Model training Accuracy: 0.6847614689394806
Best CV Score: 0.6726918670047767
Best Paramater: {'gamma': 0.3, 'learning_rate': 0.3, 'max_depth': 3, 'min_child_weight': 4, 'n_estimators': 50}
```

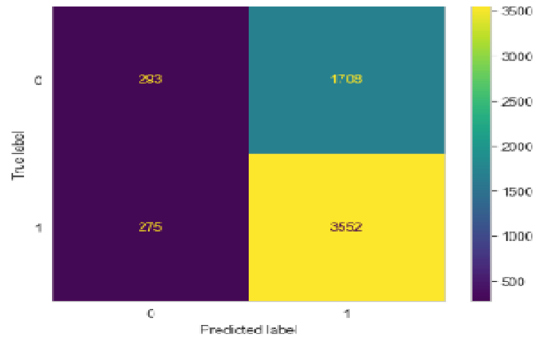
Gender Prediction - XG Boost Tuning Model Selection

```
# Since Model 1 has best acccuracy score taking model 1 as final model
best_params_gender=best_params_gender_1
```

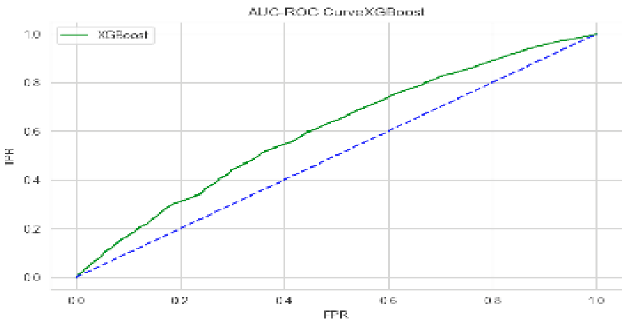
Gender Prediction - XG Boost Model with Best Parameters

```
#Best XG Boost Model

y_pred,accuracy,training_accuracy,recall,precision,F1,xgb_gender = xg_boost(X_train,y_gender_train,X_test,
        y_gender_test,**best_params_gender)
evaluation_metrics(xgb_gender,X_test,y_gender_test,"XGBoost",accuracy,training_accuracy,recall,precision,F1,
        "Gender")
```

```
----------------Model Statistics----------------
Gender Model Accuracy: 0.6597460533346603
Gender Model training Accuracy: 0.6774958539068756
Gender Model Precision: 0.6752851711026516
Gender Model Recall: 0.9281421478955247
Gender Model F1 Score: 0.7817761637504128
----------------Confusion Matrix----------------

<Figure size 800x500 with 0 Axes>
```




```
----------------Decile Analysis----------------
Top Probability: 0.54701
Bottom Probability: 0.6412411
KS Statistic
Decile 0: 0.0036009325448654305
Decile 1: 0.0008884243532793349
Decile 2: 0.005252155735563091
Decile 3: 5.225025637524276e-05
Decile 4: 0.0027436534439508834
Decile 5: 0.0029265743402142608
Decile 6: 0.0020642301149725054
Decile 7: 0.004860233614996 7425
Decile 8: 0.0026914031878756406
Decile 9: 0.0010452851215050773
----------------ROC Curve----------------
ROC AUC SCORE ->  0.6627117092094141
```

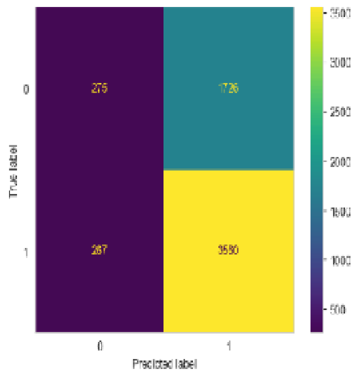


Scenario 1 : Gender Prediction, Stacking

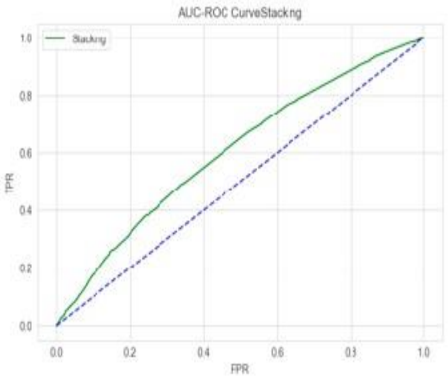
Gender Prediction - Model Stacking - 1

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[lr_gender,rf_gender]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_gender_1=model_stacking(X_train,y_gender_train,
        X_test,y_gender_test,classifiers,xgb_gender)
evaluation_metrics(stacking_gender_1,X_test,y_gender_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6588381998391215
Gender Model training Accuracy: 0.6785836360771078
Gender Model Precision: 0.6734771093454408
Gender Model Recall: 0.9302325581395349
Gender Model F1 Score: 0.7813014375868584
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```



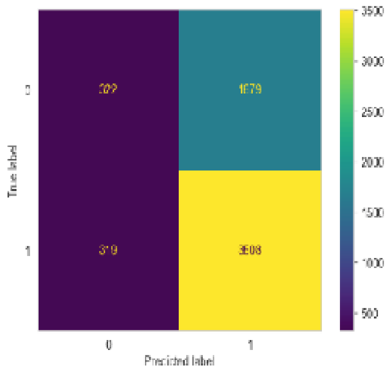

```
---------------Decile Analysis---------------
Top Probability: 0.54731107
Bottom Probability: 0.6428404
KS Statistic
Decile 0: 0.0006089329448654305
Decile 1: 0.0008884243532279349
Decile 2: 0.005252155735563091
Decile 3: 5.2258256875242 76e-05
Decile 4: 0.002743653443950 8834
Decile 5: 0.002925574340214260 8
Decile 6: 0.002864230114972 5054
Decile 7: 0.004868228338149 987425
Decile 8: 0.002691443187875 6406
Decile 9: 0.0018452851215050773
---------------ROC Curve---------------
ROC AUC SCORE ->  0.6347950825509647
```



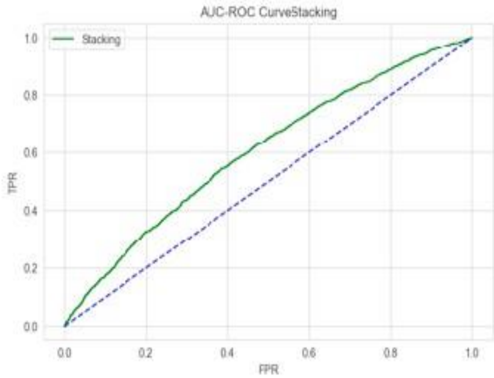
Gender Prediction - Model Stacking - 2

```
#### Stacking XGBoost and random forest as classifier and XGBoost as Logistic regression Meta classifier
classifiers=[xgb_gender,rf_gender]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_gender_2=model_stacking(X_train,y_gender_train,
        X_test,y_gender_test,classifiers,lr_gender)
evaluation_metrics(stacking_gender_2,X_test,y_gender_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6571722717913521
Gender Model training Accuracy: 0.6844182587804599
Gender Model Precision: 0.6763061459903685
Gender Model Recall: 0.9166448915559585
Gender Model F1 Score: 0.7783447966824717
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```



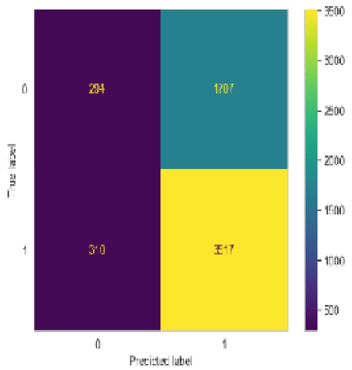

```
---------------Decile Analysis---------------
Top Probability: 0.5444834438962366
Bottom Probability: 0.6446314072494459
KS Statistic
Decile 0: 0.0006089329448654305
Decile 1: 0.0008884243532279349
Decile 2: 0.005252155735563091
Decile 3: 5.226825687524276e-05
Decile 4: 0.002743653443950 8834
Decile 5: 0.002925574340214260 8
Decile 6: 0.002864230114972 5054
Decile 7: 0.004868228338149 987425
Decile 8: 0.002691443187875 6406
Decile 9: 0.0018452851215050773
---------------ROC Curve---------------
ROC AUC SCORE ->  0.6054092760256924
```



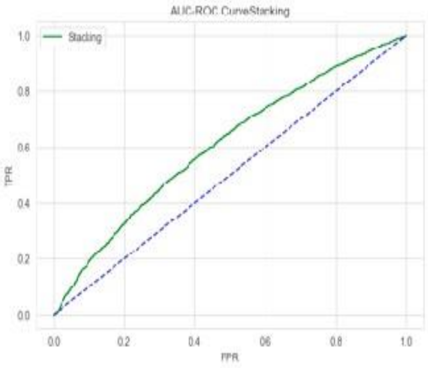
Gender Prediction - Model Stacking - 3

```
#### Stacking XGBoost and Logistic regression as classifier and XGBoost as Random Forest Meta classifier
classifiers=[lr_gender,xgb_gender]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_gender_3=model_stacking(X_train,y_gender_train,
        X_test,y_gender_test,classifiers,rf_gender)
evaluation_metrics(stacking_gender_3,X_test,y_gender_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6539121482498285
Gender Model training Accuracy: 0.6760668115776227
Gender Model Precision: 0.6732388973966389
Gender Model Recall: 0.9189960830833551
Gender Model F1 Score: 0.7771516959451394
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Decile Analysis---------------
Top Probability: 0.5502532518772229
Bottom Probability: 0.6365223613797085
KS Statistic
Decile 0: 0.0006089329448654305
Decile 1: 0.0008884243532279349
Decile 2: 0.005252155735563091
Decile 3: 5.226825687524276e-05
Decile 4: 0.002743653443950 8834
Decile 5: 0.002925574340214260 8
Decile 6: 0.002864230114972 5054
Decile 7: 0.004868228338149 987425
Decile 8: 0.002691443187875 6406
Decile 9: 0.0018452851215050773
---------------ROC Curve---------------
ROC AUC SCORE ->  0.6065696533017264
```



Scenario 1 : Gender Prediction, Final Model

Gender Prediction - Final Model Selection

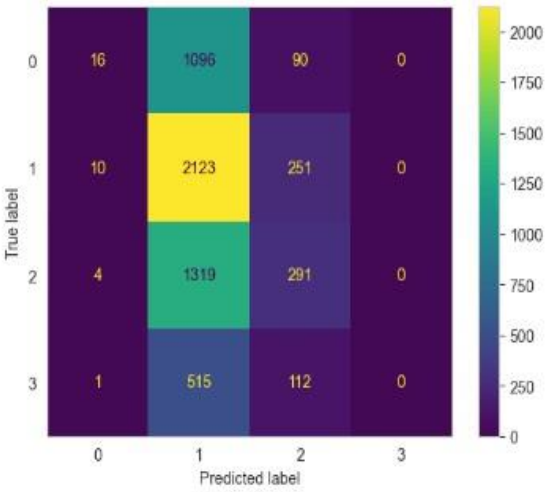
```
# Logistic Regression Gives best accuracy so considering Logistic Regression model as final model
final_model_gender = lr_gender
```

Scenario 1 : AgeGroup Prediction, Logistic Reg.

Age Group Prediction - Logistic Regression

```
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_age_group_1 = logistic_regression(X_train,
        y_age_group_train,X_test,y_age_group_test,average="weighted")
evaluation_metrics(lr_age_group_1,X_test,y_age_group_test,"Logistic Regression",accuracy,training_accuracy,
        recall,precision,F1,"Age Group",multiclass=True)
```

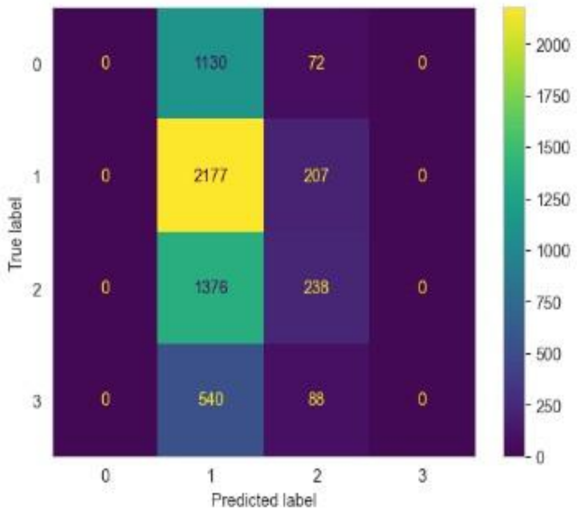
```
---------------Model Statistics---------------
Age Group Model Accuracy: 0.4169526424159231
Age Group Model training Accuracy: 0.4159707127330969
Age Group Model Precision: 0.38663323370759994
Age Group Model Recall: 0.4169526424159231
Age Group Model F1 Score: 0.30725067530201844
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2607
```

```
#Applying L1 Regularization
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_age_group_2 = logistic_regression(X_train,
        y_age_group_train,X_test,y_age_group_test,average="weighted",penalty='l1', solver='liblinear')
evaluation_metrics(lr_age_group_2,X_test,y_age_group_test,"Logistic Regression",accuracy,training_accuracy,
        recall,precision,F1,"Age Group",multiclass=True)
```

```
---------------Model Statistics---------------
Age Group Model Accuracy: 0.414378860672615
Age Group Model training Accuracy: 0.416027914426267
Age Group Model Precision: 0.279444850883026
Age Group Model Recall: 0.414378860672615
Age Group Model F1 Score: 0.29353896262283374
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2606
```

Scenario 1 : AgeGroup Prediction, Random Forest

Age Group Prediction - Random Forest Model 1

```
param_grid={
            "n_estimators":[120,130],
           "min_samples_split":[2,3],
           "min_samples_leaf":[3,5],
           "max_leaf_nodes":[100,120],
           "max_depth":[50],
           "oob_score":[True],
            "bootstrap":[True]

           }

y_age_group_pred,accuracy_age_group,training_accuracy_age_group,cv_age_group,best_score_age_group,
        best_params_age_group_1 = cross_validation(X_train,y_age_group_train,X_test,y_age_group_test,
        rf_age_group_1,param_grid)
print("age_group Model Accuracy:",accuracy_age_group)
print("age_group Model training Accuracy:",training_accuracy_age_group)
print("Best CV Score:",best_score_age_group)
print("Best Paramater:",best_params_age_group_1)
```

```
age_group Model Accuracy: 0.41815374056280025
age_group Model training Accuracy: 0.44274110513671205
Best CV Score: 0.41534148046686004
Best Paramater: {'bootstrap': True, 'max_depth': 50, 'max_leaf_nodes': 100, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 120, 'oob_score': True}
```

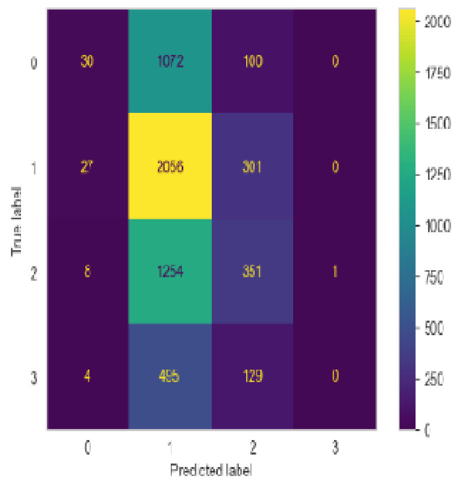
Age Group Prediction - Random Forest Model Selection

```
# All Models have similar Accuracy Taking model 1 as final model
best_params_age_group=best_params_age_group_1
```

Age Group Prediction - Random Forest - Best Parameters

```
#Best Random Forest Model
y_pred,accuracy,training_accuracy,recall,precision,F1,rf_age_group = randon_forest(X_train,y_age_group_train,
        X_test,y_age_group_test,average="weighted",**best_params_age_group)
evaluation_metrics(rf_age_group,X_test,y_age_group_test,"Random Forest",accuracy,training_accuracy,recall,
        precision,F1,"age_group",multiclass=True)
```

```
---------------Model Statistics---------------
age_group Model Accuracy: 0.41815374056280025
age_group Model training Accuracy: 0.44274110513671205
age_group Model Precision: 0.37245509320271114
age_group Model Recall: 0.41815374056280025
age_group Model F1 Score: 0.31931241974438135
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2572
```

Scenario 1: AgeGroup Prediction, XGBoost

Age Group Prediction - XG Boost Model 1

```
param_grid={"n_estimators":[10,20],
            "max_depth":[3,6],
            "learning_rate":[0.1,0.3],
             'min_child_weight': [2, 3],
             'gamma': [0.1, 0.2]

            }

y_age_group_pred,accuracy_age_group,training_accuracy_age_group,cv_age_group,best_score_age_group,
        best_params_age_group_1 = cross_validation(X_train,y_age_group_train,X_test,y_age_group_test,
        xgb_age_group_1,param_grid)
print("age_group Model Accuracy:",accuracy_age_group)
print("age_group Model training Accuracy:",training_accuracy_age_group)
print("Best CV Score:",best_score_age_group)
print("Best Paramater:",best_params_age_group_1)
```

```
age_group Model Accuracy: 0.4226149622512011
age_group Model training Accuracy: 0.42060404987987643
Best CV Score: 0.4154557137179274
Best Paramater: {'gamma': 0.1, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 2, 'n_estimators': 10}
```

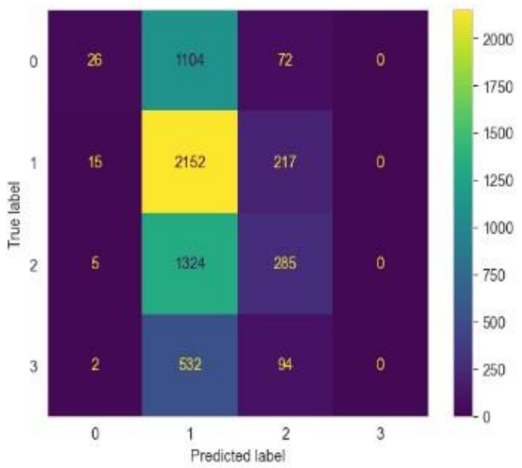
Age Group Prediction - XG Boost Tuning Model Selection

```
# Since Model 1 has best acccuracy score taking model 1 as final model
best_params_age_group=best_params_age_group_1
```

Age Group Prediction - XG Boost Model with Best Parameters

```
#Base XG Boost Model

y_pred,accuracy,training_accuracy,recall,precision,F1,xgb_age_group = xg_boost(X_train,y_age_group_train,
        X_test,y_age_group_test,average="weighted",**best_params_age_group)
evaluation_metrics(xgb_age_group,X_test,y_age_group_test,"XG Boost",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
---------------Model Statistics---------------
Age Group Model Accuracy: 0.4226149622512011
Age Group Model training Accuracy: 0.42060404987987643
Age Group Model Precision: 0.40207347838117236
Age Group Model Recall: 0.4226149622512011
Age Group Model F1 Score: 0.31262486922894567
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```



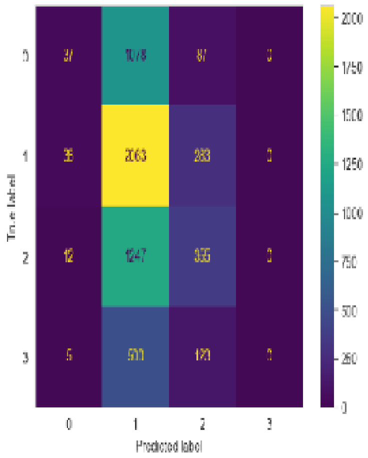

```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2926
```

Scenario 1 : AgeGroup Prediction, Stacking

Age Group Prediction - Model 1

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[lr_age_group,rf_age_group]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_age_group_1=model_stacking(X_train,
        y_age_group_train,X_test,y_age_group_test,classifiers,xgb_age_group,average='weighted')
evaluation_metrics(stacking_age_group_1,X_test,y_age_group_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
==============Model Statistics=============
Age Group Model Accuracy: 0.4212422785547701
Age Group Model training Accuracy: 0.4235213362315524
Age Group Model Precision: 0.3715274480377117
Age Group Model Recall: 0.4212422786547701
Age Group Model F1 Score: 0.3237521818592823
--------------Confusion Matrix-------------

&lt;Figure size 800x500 with 0 Axes&gt;
```

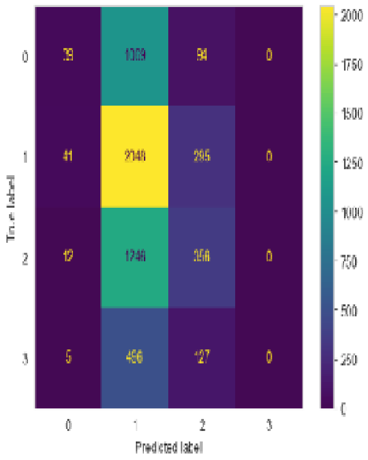



```
--------------Multiclass Log Loss--------------
Multiclass Log Loss: 1.2917
```

Age Group Prediction - Model 2

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[xgb_age_group,rf_age_group]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_age_group_2=model_stacking(X_train,
        y_age_group_train,X_test,y_age_group_test,classifiers,lr_age_group,average='weighted')
evaluation_metrics(stacking_age_group_2,X_test,y_age_group_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
==============Model Statistics==============
Age Group Model Accuracy: 0.4191832532601235
Age Group Model training Accuracy: 0.4318727834343897
Age Group Model Precision: 0.3683985251815096
Age Group Model Recall: 0.4191832532601235
Age Group Model F1 Score: 0.3230286266185266
--------------Confusion Matrix--------------

&lt;Figure size 800x500 with 0 Axes&gt;
```

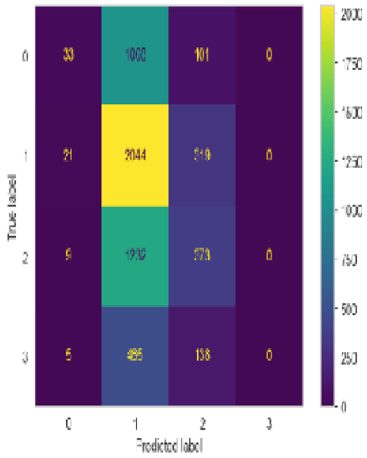



```
==============Multiclass Log Loss==============
Multiclass Log Loss: 1.2573
```

Age Group Prediction - Model 3

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[lr_age_group,xgb_age_group]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_age_group_3=model_stacking(X_train,
        y_age_group_train,X_test,y_age_group_test,classifiers,rf_age_group,average='weighted')
evaluation_metrics(stacking_age_group_3,X_test,y_age_group_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
--------------Model Statistics--------------
Age Group Model Accuracy: 0.4283043514878087
Age Group Model training Accuracy: 0.4207384532652157
Age Group Model Precision: 0.3841899497652787
Age Group Model Recall: 0.4283043514878087
Age Group Model F1 Score: 0.3237320811587844
==============Confusion Matrix==============

&lt;Figure size 800x500 with 0 Axes&gt;
```




```
--------------Multiclass Log Loss--------------
Multiclass Log Loss: 1.2599
```

Scenario 1 : AgeGroup Prediction, Final Model

Age Group Prediction - Final Model Selection

```
# Random Forest Gives best accuracy so considering XG Boost model as final model
final_model_age_group = xgb_age_group
```

Model Building - 2

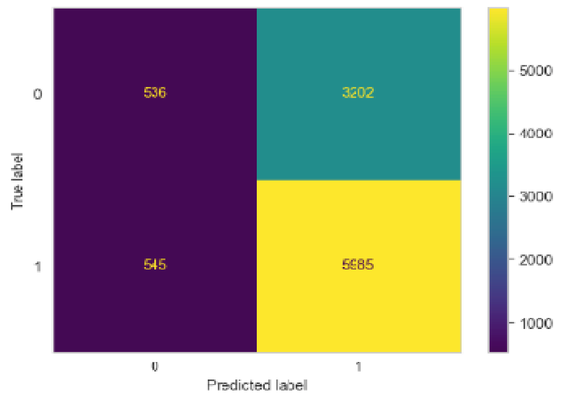
- Scenario 1 – Without Events
  - Prediction - **Gender**
  - Models :–
    - Logistic
    - Random Forest
    - XG Boost
    - Model Stacking

- Scenario 1 - Without Events
  - Prediction -**AgeGroup**
  - Models :–
    - Logistic
    - Random Forest
    - XG Boost
    - Model Stacking

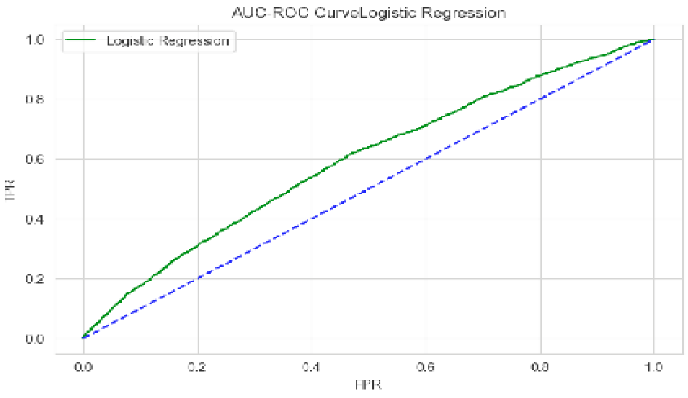
Scenario 2: Gender Prediction, Logistic Reg

```
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_gender_1 = logistic_regression(X_train,
        y_gender_train,X_test,y_gender_test)
evaluation_metrics(lr_gender_1,X_test,y_gender_test,"Logistic Regression",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.6350798597584729
Gender Model training Accuracy: 0.6456765773005089
Gender Model Precision: 0.651464025253075
Gender Model Recall: 0.9165390505359877
Gender Model F1 Score: 0.7615957243748807
---------------Confusion Matrix---------------
<Figure size 800x500 with 0 Axes>
```

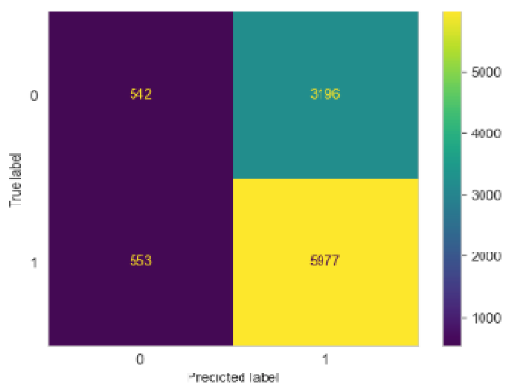



```
---------------Decile Analysis---------------
Top Probability: 0.20917048862414023
Bottom Probability: 0.8698126143751280
KS Statistic
Decile 0: 0.0032159264931087284
Decile 1: 0.003062787136294015
Decile 2: 0.003215926493108756
Decile 3: 0.002450229709903519
Decile 4: 0.0033690658499234694
Decile 5: 0.0006125574272588530
Decile 6: 0.0004594180704440287
Decile 7: 0.0013782542113323082
Decile 8: 0.001990811638591161
Decile 9: 0.0007656967840734552
---------------ROC Curve---------------
ROC AUC SCORE ->  0.5943410746957902
```

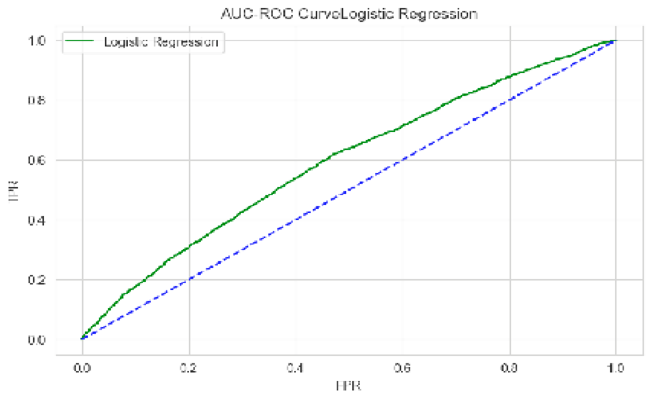



```
#Applying L1 Regularization
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_gender_2 = logistic_regression(X_train,
        y_gender_train,X_test,y_gender_test,penalty='l1', solver='liblinear')
evaluation_metrics(lr_gender_2,X_test,y_gender_test,"Logistic Regression",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.634885079859758
Gender Model training Accuracy: 0.6457983295590133
Gender Model Precision: 0.6515861768232857
Gender Model Recall: 0.9153139356814701
Gender Model F1 Score: 0.7612558109915303
---------------Confusion Matrix---------------
<Figure size 800x500 with 0 Axes>
```




```
---------------Decile Analysis---------------
Top Probability: 0.20483807417177302
Bottom Probability: 0.873673962145846
KS Statistic
Decile 0: 0.0032159264931087284
Decile 1: 0.003062787136294015
Decile 2: 0.003215926493108756
Decile 3: 0.002450229709903519
Decile 4: 0.0033690658499234694
Decile 5: 0.0006125574272588530
Decile 6: 0.0004594180704440287
Decile 7: 0.0013782542113323082
Decile 8: 0.001990811638591161
Decile 9: 0.0007656967840734552
---------------ROC Curve---------------
ROC AUC SCORE ->  0.594020620964142
```



Scenario 2: Gender Prediction, Random Forest

Gender Prediction - Random Forest - Model 2

```
param_grid={
                "n_estimators":[100,120],
               "min_samples_split":[2,3],
               "min_samples_leaf":[3,5],
               "max_leaf_nodes":[120,130],
               "max_depth":[50],
               "oob_score":[True],
                "bootstrap":[True]

               }

y_gender_pred,accuracy_gender,training_accuracy_gender,cv_gender,best_score_gender,best_params_gender_2 =
        cross_validation(X_train,y_gender_train,X_test,y_gender_test,rf_gender_1,param_grid)
print("Gender Model Accuracy:",accuracy_gender)
print("Gender Model training Accuracy:",training_accuracy_gender)
print("Best CV Score:",best_score_gender)
print("Best Paramater:",best_params_gender_2)
```

```
Gender Model Accuracy: 0.639851967276977
Gender Model training Accuracy: 0.6662770594394526
Best CV Score: 0.6523242506148489
Best Paramater: {'bootstrap': True, 'max_depth': 50, 'max_leaf_nodes': 120, 'min_samples_leaf': 5, 'min_samples_split': 2, 'n_estimators': 120, 'oob_score': True}
```

Gender Prediction - Random Forest - Model Selection

```
# Since Model 2 has best acccuracy score taking model 2 as final model
best_params_gender=best_params_gender_2
```

Scenario 2: Gender Prediction, X G Boost

Gender Prediction - XG Boost Tuning Model 3

```
param_grid={"n_estimators":[20],
            "max_depth":[6,8],
            "learning_rate":[0.3,0.4],
             'min_child_weight': [2, 3],
             'gamma': [0.2, 0.3]

            }

y_gender_pred,accuracy_gender,training_accuracy_gender,cv_gender,best_score_gender,best_params_gender_3 =
         cross_validation(X_train,y_gender_train,X_test,y_gender_test,xgb_gender_1,param_grid)
print("Gender Model Accuracy:",accuracy_gender)
print("Gender Model training Accuracy:",training_accuracy_gender)
print("Best CV Score:",best_score_gender)
print("Best Paramater:",best_params_gender_3)
```

```
Gender Model Accuracy: 0.638975457732762
Gender Model training Accuracy: 0.66620400808435
Best CV Score: 0.6527138578420629
Best Paramater: {'gamma': 0.3, 'learning_rate': 0.4, 'max_depth': 6, 'min_child_weight': 2, 'n_estimators': 20}
```

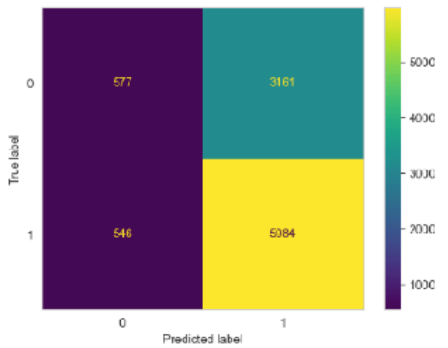
Gender Prediction - XG Boost Tuning Model Selection

```
# Since Model 3 has best acccuracy score taking model 3 as final model
best_params_gender=best_params_gender_3
```

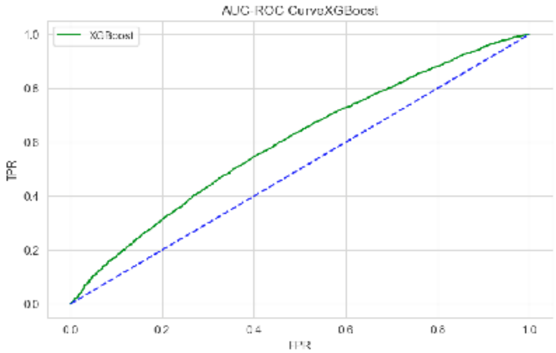
Gender Prediction - XG Boost Model with Best Parameters

```
#Best XG Boost Model

y_pred,accuracy,training_accuracy,recall,precision,F1,xgb_gender = xg_boost(X_train,y_gender_train,X_test,
        y_gender_test,**best_params_gender)
evaluation_metrics(xgb_gender,X_test,y_gender_test,"XGBoost",accuracy,training_accuracy,recall,precision,F1,
        "Gender")
```

```
---------------Model Statistics---------------
Gender Model Accuracy: 0.638975457732762
Gender Model training Accuracy: 0.66620400808435
Gender Model Precision: 0.6543466375068343
Gender Model Recall: 0.9153085911179173
Gender Model F1 Score: 0.7635087719298246
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Decile Analysis---------------
Top Probability: 0.03222484
Bottom Probability: 0.9703369
KS Statistic
Decile 0: 0.0032159264931087284
Decile 1: 0.0030627871362940 15
Decile 2: 0.003215926493108736
Decile 3: 0.002450229709035 19
Decile 4: 0.0033690658499234594
Decile 5: 0.0006125574272588 53
Decile 6: 0.0004594180704440287
Decile 7: 0.0013782542113323082
Decile 8: 0.001990811638591151
Decile 9: 0.0007656967840734552
---------------ROC Curve---------------
ROC AUC SCORE ->  0.5997812499743949
```

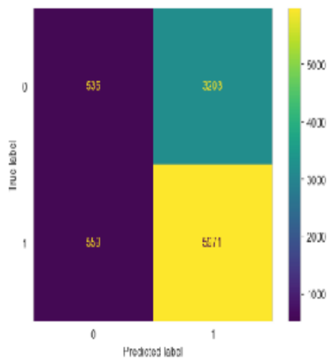


Scenario 2: Gender Prediction, Stacking

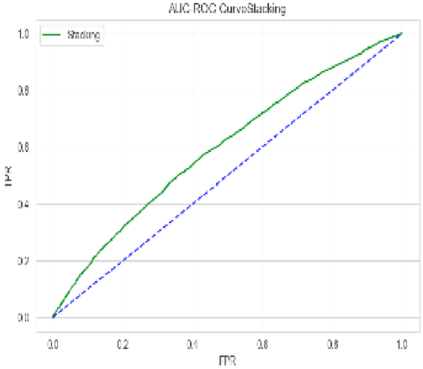
Gender Prediction - Model Stacking - 1

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[lr_gender,rf_gender]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_gender_1=model_stacking(X_train,y_gender_train,
        X_test,y_gender_test,classifiers,xgb_gender)
evaluation_metrics(stacking_gender_1,X_test,y_gender_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
==============Model Statistics==============
Gender Model Accuracy: 0.6335198185181145
Gender Model training Accuracy: 0.6570725886985203
Gender Model Precision: 0.5508611292783955
Gender Model Recall: 0.9143950993540582
Gender Model F1 Score: 0.7504431931849211
--------------Confusion Matrix--------------

<Figure size 802x503 with 0 Axes>
```



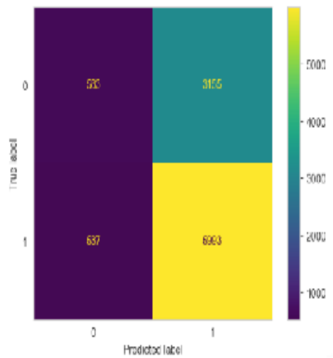

```
==============Decile Analysis==============
Top Probability: 0.113807255
Bottom Probability: 0.9591871
KS Statistic
Decile 0: 0.0032159264931087284
Decile 1: 0.0038627871362940015
Decile 2: 0.0032159264931089756
Decile 3: 0.0024582297098351519
Decile 4: 0.0033690655849923464
Decile 5: 0.0006125574272503053
Decile 6: 0.0004594180704440287
Decile 7: 0.0013782542113323082
Decile 8: 0.0019908116385911161
Decile 9: 0.0007656936784873455
==============ROC Curve==============
ROC AUC SCORE ->  0.5961215577089082
```



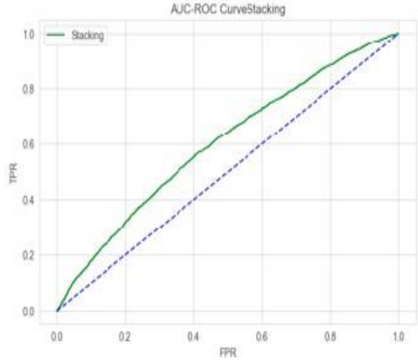
Gender Prediction - Model Stacking - 2

```
#### Stacking XGBoost and random forest as classifier and XGBoost as Logistic regression Meta classifier
classifiers=[xgb_gender,rf_gender]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_gender_2=model_stacking(X_train,y_gender_train,
        X_test,y_gender_test,classifiers,lr_gender)
evaluation_metrics(stacking_gender_2,X_test,y_gender_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
--------------Model Statistics--------------
Gender Model Accuracy: 0.6404363869731204
Gender Model training Accuracy: 0.6672510775074878
Gender Model Precision: 0.655115872321819
Gender Model Recall: 0.9177641653905053
Gender Model F1 Score: 0.7645187794361525
--------------Confusion Matrix--------------

<Figure size 800x500 with 0 Axes>
```



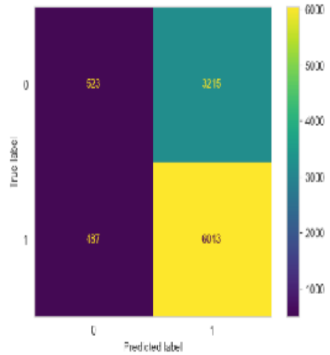

```
--------------Decile Analysis--------------
Top Probability: 0.14329574433172027
Bottom Probability: 0.8569322234179531
KS Statistic
Decile 0: 0.0032159264931087284
Decile 1: 0.0030627871362234015
Decile 2: 0.0032159264931088756
Decile 3: 0.0024502297090303515
Decile 4: 0.0033690065584992734694
Decile 5: 0.0006125574272258853
Decile 6: 0.0004594180784440287
Decile 7: 0.0013782542113323082
Decile 8: 0.0019908116385911161
Decile 9: 0.0007656967840734552
--------------ROC Curve--------------
ROC AUC SCORE ->  0.6023082746872638
```



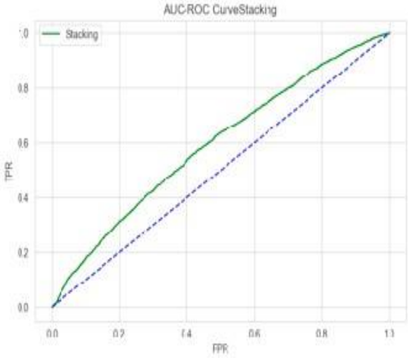
Gender Prediction - Model Stacking - 3

```
#### Stacking XGBoost and Logistic regression as classifier and XGBoost as Random Forest Meta classifier
classifiers=[lr_gender,xgb_gender]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_gender_3=model_stacking(X_train,y_gender_train,
        X_test,y_gender_test,classifiers,rf_gender)
evaluation_metrics(stacking_gender_3,X_test,y_gender_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Gender")
```

```
--------------Model Statistics--------------
Gender Model Accuracy: 0.6394824074795481
Gender Model training Accuracy: 0.6597754888353179
Gender Model Precision: 0.6527327718569453
Gender Model Recall: 0.9254211332312404
Gender Model F1 Score: 0.765518115024863
--------------Confusion Matrix--------------

<Figure size 800x500 with 0 Axes>
```




```
--------------Decile Analysis--------------
Top Probability: 0.030957167308282023
Bottom Probability: 0.9349110389973698
KS Statistic
Decile 0: 0.0032159264931087284
Decile 1: 0.0030627871362940015
Decile 2: 0.0032159264931088756
Decile 3: 0.0024502297090903519
Decile 4: 0.0033690265849923464
Decile 5: 0.0006125574272258853
Decile 6: 0.0004594180704440287
Decile 7: 0.0013782542113323082
Decile 8: 0.0019908116385891161
Decile 9: 0.0007656957840734552
--------------ROC Curve--------------
ROC AUC SCORE ->  0.5945971672906133
```



Scenario 2: Gender Prediction, Final Model

Gender Prediction - Final Model Selection

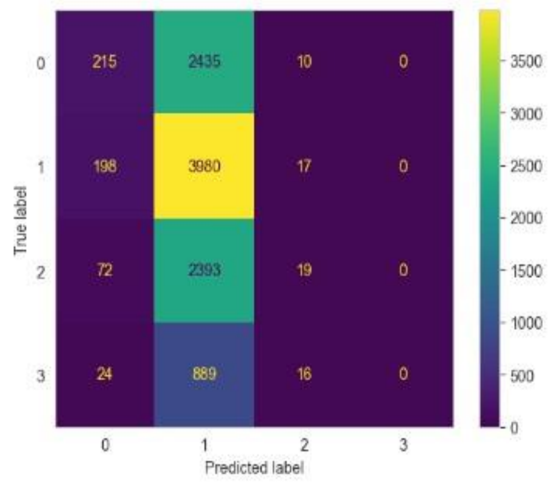
```
# Since Model 3 gives best accuracy, considering it is best Stacking Model
stacking_gender=stacking_gender_3
```

Scenario 2: AgeGroup Prediction, Logistic Reg.

Age Group Prediction - Logistic Regression

```
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_age_group_1 = logistic_regression(X_train,
        y_age_group_train,X_test,y_age_group_test,average="weighted")
evaluation_metrics(lr_age_group_1,X_test,y_age_group_test,"Logistic Regression",accuracy,training_accuracy,
        recall,precision,F1,"Age Group",multiclass=True)
```

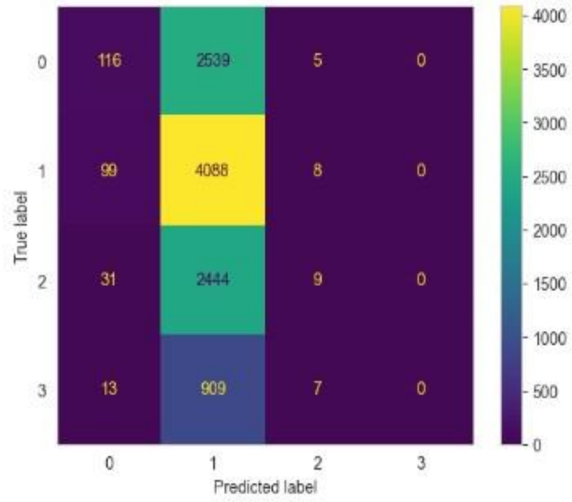
```
---------------Model Statistics---------------
Age Group Model Accuracy: 0.4104012465913518
Age Group Model training Accuracy: 0.4112547787761463
Age Group Model Precision: 0.3512447776486367
Age Group Model Recall: 0.4104012465913518
Age Group Model F1 Score: 0.27285826321268924
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2583
```

```
#Applying L1 Regularization
y_pred,accuracy,training_accuracy,recall,precision,F1,lr_age_group_2 = logistic_regression(X_train,
        y_age_group_train,X_test,y_age_group_test,average="weighted",penalty='l1', solver='liblinear')
evaluation_metrics(lr_age_group_2,X_test,y_age_group_test,"Logistic Regression",accuracy,training_accuracy,
        recall,precision,F1,"Age Group",multiclass=True)
```

```
---------------Model Statistics---------------
Age Group Model Accuracy: 0.41030385664199454
Age Group Model training Accuracy: 0.411717437358463
Age Group Model Precision: 0.35845350686465827
Age Group Model Recall: 0.41030385664199454
Age Group Model F1 Score: 0.2579705605911575
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2583
```

Scenario 2: AgeGroup Prediction, Random Forest

Age Group Prediction - Random Forest Model 1

```
param_grid={
            "n_estimators":[120,130],
           "min_samples_split":[2,3],
           "min_samples_leaf":[3,5],
           "max_leaf_nodes":[100,120],
           "max_depth":[50],
           "oob_score":[True],
            "bootstrap":[True]

           }

y_age_group_pred,accuracy_age_group,training_accuracy_age_group,cv_age_group,best_score_age_group,
        best_params_age_group_1 = cross_validation(X_train,y_age_group_train,X_test,y_age_group_test,
        rf_age_group_1,param_grid)
print("age_group Model Accuracy:",accuracy_age_group)
print("age_group Model training Accuracy:",training_accuracy_age_group)
print("Best CV Score:",best_score_age_group)
print("Best Paramater:",best_params_age_group_1)
```

```
age_group Model Accuracy: 0.4120568757304246
age_group Model training Accuracy: 0.42048359997077944
Best CV Score: 0.4081866218618355
Best Paramater: {'bootstrap': True, 'max_depth': 50, 'max_leaf_nodes': 100, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 130, 'oob_score': True}
```

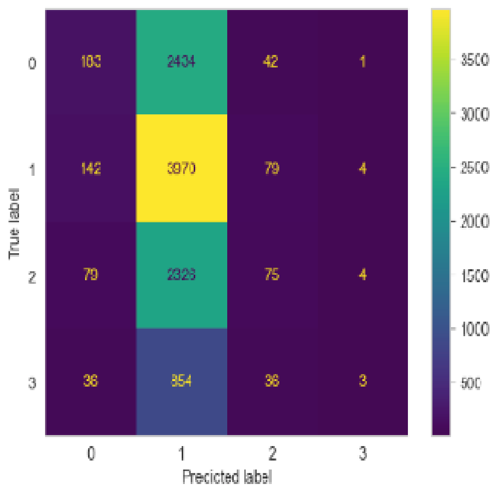
Age Group Prediction - Random Forest Model Selection

```
# Since Model 1 has best acccuracy score taking model 1 as final model
best_params_age_group=best_params_age_group_1
```

Age Group Prediction - Random Forest - Best Parameters

```
#Best Random Forest Model
y_pred,accuracy,training_accuracy,recall,precision,F1,rf_age_group = random_forest(X_train,y_age_group_train,
        X_test,y_age_group_test,average="weighted",**best_params_age_group)
evaluation_metrics(rf_age_group,X_test,y_age_group_test,"Random Forest",accuracy,training_accuracy,recall,
        precision,F1,"age_group",multiclass=True)
```

```
---------------Model Statistics---------------
age_group Model Accuracy: 0.4120568757304246
age_group Model training Accuracy: 0.42048359997077944
age_group Model Precision: 0.3778037482766203
age_group Model Recall: 0.4120568757304246
age_group Model F1 Score: 0.27994600716422957
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```




```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2545
```

Scenario 2: AgeGroup Prediction, XGBoost

Age Group Prediction - XG Boost Model 2

```
param_grid={"n_estimators":[10,20],
            "max_depth":[6,9],
            "learning_rate":[0.3,0.5],
             'min_child_weight': [2, 3],
             'gamma': [0.1, 0.2]

            }

y_age_group_pred,accuracy_age_group,training_accuracy_age_group,cv_age_group,best_score_age_group,
        best_params_age_group_1 = cross_validation(X_train,y_age_group_train,X_test,y_age_group_test,
        xgb_age_group_1,param_grid)
print("age_group Model Accuracy:",accuracy_age_group)
print("age_group Model training Accuracy:",training_accuracy_age_group)
print("Best CV Score:",best_score_age_group)
print("Best Paramater:",best_params_age_group_1)
```

```
age_group Model Accuracy: 0.41264121542656795
age_group Model training Accuracy: 0.4216767721041225
Best CV Score: 0.4077483137312197
Best Paramater: {'gamma': 0.1, 'learning_rate': 0.5, 'max_depth': 6, 'min_child_weight': 3, 'n_estimators': 10}
```

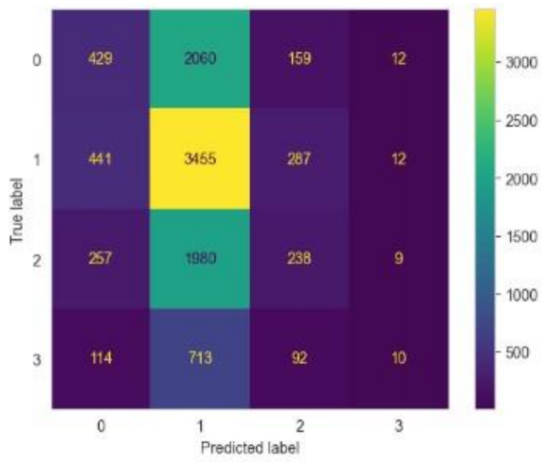
Age Group Prediction - XG Boost Tuning Model Selection

```
# Since Model 2 has best acccuracy score taking model 2 as final model
best_params_age_group=best_params_age_group_2
```

Age Group Prediction - XG Boost Model with Best Parameters

```
#Base XG Boost Model

y_pred,accuracy,training_accuracy,recall,precision,F1,xgb_age_group = xg_boost(X_train,y_age_group_train,
        X_test,y_age_group_test,average="weighted",**best_params_age_group)
evaluation_metrics(xgb_age_group,X_test,y_age_group_test,"XG Boost",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
[14:36:19] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

---------------Model Statistics---------------
Age Group Model Accuracy: 0.4024152707440592
Age Group Model training Accuracy: 0.4351669223464095
Age Group Model Precision: 0.3567617072607607
Age Group Model Recall: 0.4024152707440592
Age Group Model F1 Score: 0.32177559773047437
---------------Confusion Matrix---------------

<Figure size 800x500 with 0 Axes>
```



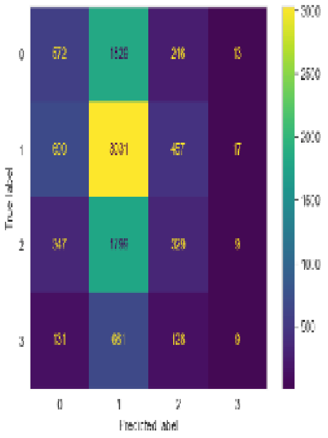

```
---------------Multiclass Log Loss---------------
Multiclass Log Loss: 1.2906
```

Scenario 2: AgeGroup Prediction, Stacking

Age Group Prediction - Model 1

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[lr_age_group,rf_age_group]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_age_group_1=model_stacking(X_train,
        y_age_group_train,X_test,y_age_group_test,classifiers,xgb_age_group,average='weighted')
evaluation_metrics(stacking_age_group_1,X_test,y_age_group_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
[14:38:23] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

--------------Model Statistics--------------
Age Group Model Accuracy: 0.38381379841829
Age Group Model training Accuracy: 0.39681679004521879
Age Group Model Precision: 0.33908709777967593
Age Group Model Recall: 0.38381379841829
Age Group Model F1 Score: 0.32778390549364486
==============Confusion Matrix==============

<Figure size 800x500 with 0 Axes>
```

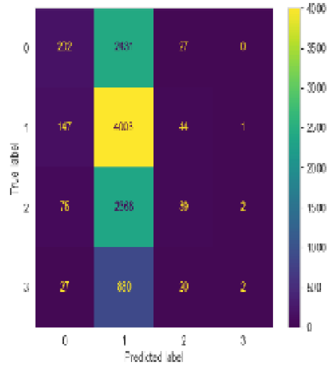



```
--------------Multiclass Log Loss--------------
Multiclass Log Loss: 1.3336
```

Age Group Prediction - Model 2

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[xgb_age_group,rf_age_group]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_age_group_2=model_stacking(X_train,
        y_age_group_train,X_test,y_age_group_test,classifiers,lr_age_group,average='weighted')
evaluation_metrics(stacking_age_group_2,X_test,y_age_group_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
[14:40:57] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

[14:41:25] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

[14:41:51] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

[14:42:21] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

--------------Model Statistics--------------
Age Group Model Accuracy: 0.41351772497078304
Age Group Model training Accuracy: 0.41924172693340346
Age Group Model Precision: 0.39370955626165593
Age Group Model Recall: 0.41351772497078304
Age Group Model F1 Score: 0.27695128827580468
==============Confusion Matrix==============

<Figure size 800x500 with 0 Axes>
```

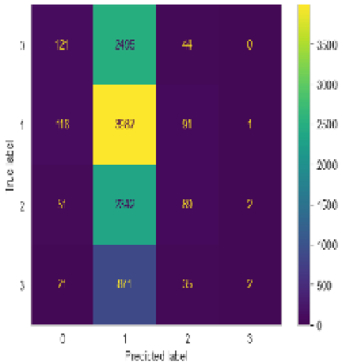



```
--------------Multiclass Log Loss--------------
Multiclass Log Loss: 1.2545
```

Age Group Prediction - Model 3

```
#### Stacking Logistic regression and random forest as classifier and XGBoost as Meta classifier
classifiers=[lr_age_group,xgb_age_group]
y_pred,accuracy,training_accuracy,recall,precision,F1,stacking_age_group_3=model_stacking(X_train,
        y_age_group_train,X_test,y_age_group_test,classifiers,rf_age_group,average='weighted')
evaluation_metrics(stacking_age_group_3,X_test,y_age_group_test,"Stacking",accuracy,training_accuracy,recall,
        precision,F1,"Age Group",multiclass=True)
```

```
[22:28:58] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

[22:29:13] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

[22:29:28] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

[22:29:45] WARNING: /Users/runner/work/xgboost/xgboost/python-package/build/temp.macosx-10.9-x86_64-cpython-38/xgboost/src/learner.cc:767:
Parameters: { "bootstrap", "max_leaf_nodes", "min_samples_leaf", "min_samples_split", "oob_score" } are not used.

==============Model Statistics==============
Age Group Model Accuracy: 0.40894839735099336
Age Group Model training Accuracy: 0.41778764847687925
Age Group Model Precision: 0.38877659044560386
Age Group Model Recall: 0.40894839735099336
Age Group Model F1 Score: 0.27174322957920393
==============Confusion Matrix==============

<Figure size 800x500 with 0 Axes>
```




```
==============Multiclass Log Loss==============
Multiclass Log Loss: 1.2566
```

Scenario 2: AgeGroup Prediction, Final Model

Age Group Prediction - Final Model Selection

```
# Stacking Gives best accuracy so considering Stacking model as final model
final_model_age_group = stacking_age_group
```

Scenario 1 - Model Performance

Scenario	Target	Modeling Type	Model Name	Accuracy	Training Accuracy	Precision	Recall	F1 Score	ROC Score
Scenario 1	Gender Prediction	Logistic Regression	lr_gender_1	65.854	67.120	67.445	92.788	78.112	60.872
		Logistic Regression - L1 Regularization	lr_gender_2	65.923	67.211	67.529	92.657	78.122	60.740
		Random Forest 1	best_params_gender_1	65.511	69.408				
		Random Forest 2	best_params_gender_2	65.511	69.408				
		Random Forest 3	best_params_gender_3	65.562	69.299				
		Random Forest 4	best_params_gender_4	65.614	69.225				
		Random Forest 5	best_params_gender_5	65.734	69.128				
		Random Forest Best Parameters	rf_gender	65.734	69.128	67.435	92.474	77.994	60.567
		XG Boost 1	best_params_gender_1	65.974	67.749				
		XG Boost 2	best_params_gender_2	65.957	67.881				
		XG Boost 3	best_params_gender_3	65.837	67.904				
		XG Boost 4	best_params_gender_4	65.923	68.150				
		XG Boost 5	best_params_gender_5	65.579	68.476				
		XG Boost Best Parameters	xgb_gender	65.974	67.749	67.528	92.814	78.177	60.271
		Stacking Model 1	stacking_gender_1	65.803	67.858	67.347	93.023	78.130	60.479
		Stacking Model 2	stacking_gender_2	65.717	68.441	67.630	91.664	77.834	60.540
		Stacking Model 3	stacking_gender_3	65.391	67.606	67.323	91.899	77.715	60.656
	Age Group Prediction	Logistic Regression	lr_age_group_1	41.695	41.597	38.663	41.695	30.725	
		Logistic Regression - L1 Regularization	lr_age_group_2	41.437	41.602	27.944	41.437	29.353	
		Random Forest 1	best_params_age_group_1	41.815	44.274				
		Random Forest 2	best_params_age_group_2	41.815	44.274				
		Random Forest 3	best_params_age_group_3	41.815	44.274				
		Random Forest 4	best_params_age_group_4	41.815	44.274				
		Random Forest 5	best_params_age_group_5	41.815	44.274				
		Random Forest Best Parameters	rf_age_group	41.815	44.274	37.245	41.815	31.931	
		XG Boost 1	best_params_age_group_1	42.261	42.060				
		XG Boost 2	best_params_age_group_2	41.969	41.718				
		XG Boost 3	best_params_age_group_3	42.227	42.157				
		XG Boost 4	best_params_age_group_4	41.935	42.821				
		XG Boost 5	best_params_age_group_5	41.798	43.358				
		XG Boost Best Parameters	xgb_age_group	42.261	42.060	40.207	42.261	31.262	
		Stacking Model 1	stacking_age_group_1	42.124	42.352	37.152	42.124	32.375	
		Stacking Model 2	stacking_age_group_2	41.918	43.187	36.839	41.918	32.302	
		Stacking Model 3	stacking_age_group_3	42.038	42.071	38.418	42.038	32.373	

Scenario 2 – Model performance

Scenario	Target	Modeling Type	Model Name	Accuracy	Training Accuracy	Precision	Recall	F1 Score	ROC Score
Scenario 2	Gender Prediction	Logistic Regression	lr_gender_1	63.507	64.567	65.146	91.653	76.159	59.434
		Logistic Regression - L1 Regularization	lr_gender_2	63.488	64.579	65.158	91.531	76.125	59.402
		Random Forest 1	best_params_gender_1	63.936	66.605				
		Random Forest 2	best_params_gender_2	63.985	66.627				
		Random Forest 3	best_params_gender_3	63.975	66.720				
		Random Forest Best Parameters	rf_gender	63.985	66.627	65.278	92.649	76.591	60.276
		XG Boost 1	best_params_gender_1	63.751	66.598				
		XG Boost 2	best_params_gender_2	63.761	66.457				
		XG Boost 3	best_params_gender_3	63.897	66.620				
		XG Boost Best Parameters	xgb_gender	63.897	66.620	65.434	91.638	76.350	59.978
		Stacking Model 1	stacking_gender_1	63.361	65.707	65.086	91.439	76.044	59.612
		Stacking Model 2	stacking_gender_2	60.043	66.725	65.511	91.776	76.451	60.230
		Stacking Model 3	stacking_gender_3	63.946	65.977	65.273	92.542	76.551	59.459
	Age Group Prediction	Logistic Regression	lr_age_group_1	41.040	41.125	35.124	41.040	27.285	
		Logistic Regression - L1 Regularization	lr_age_group_2	41.030	41.171	35.845	41.030	25.797	
		Random Forest 1	best_params_age_group_1	41.205	42.048				
		Random Forest 2	best_params_age_group_2	41.166	41.843				
		Random Forest 3	best_params_age_group_3	41.205	42.048				
		Random Forest Best Parameters	rf_age_group	41.205	42.048	37.780	41.205	27.994	
		XG Boost 1	best_params_age_group_1	41.059	41.371				
		XG Boost 2	best_params_age_group_2	41.264	42.167				
		XG Boost 3	best_params_age_group_3	41.020	41.427				
		XG Boost Best Parameters	xgb_age_group	40.241	43.516	35.676	40.241	32.177	
		Stacking Model 1	stacking_age_group_1	38.381	39.301	33.990	38.381	32.778	
		Stacking Model 2	stacking_age_group_2	41.351	41.924	39.370	41.351	27.695	
		Stacking Model 3	stacking_age_group_3	40.894	41.770	38.877	40.894	27.174	

Thank You ☺!