MGSCFinalProject

November 30, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from plotnine import *
from sklearn.linear_model import LogisticRegression, Lasso, ElasticNet #␣
 ↪Logistic Regression Model
from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.model_selection import train_test_split # simple TT split cv
import pandas as pd
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪ConfusionMatrixDisplay,\
 f1_score, recall_score, precision_score, roc_auc_score
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
 ↪mean_absolute_error #model evaluation
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
 ↪mean_absolute_error #model evaluation
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.linear_model import LogisticRegression, Lasso, ElasticNet #␣
 ↪Logistic Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
 ↪mean_absolute_error #model evaluation
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
 ↪mean_absolute_error #model evaluation
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪ConfusionMatrixDisplay,\
 f1_score, recall_score, precision_score, roc_auc_score
from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.cluster import DBSCAN
```

[2]:
```
cleanedData = pd.read_csv('cleanDataV2.csv')
appActivity = pd.read_csv('appActivity.csv')
appActivityMonths = pd.read_csv('appActivityMonths.csv')
```

```
appFilled = pd.read_csv('appFilled.csv')
```

#Data Exploration

[3]:
```
cleanedData.shape
```

[3]:
```
(38611, 26)
```

[4]:
```
cleanedData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38611 entries, 0 to 38610
Data columns (total 26 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ID                       38611 non-null  int64
 1   Language                 38611 non-null  object
 2   Subscription Type        38611 non-null  object
 3   Subscription Event Type  38611 non-null  object
 4   Purchase Store           38611 non-null  object
 5   Purchase Amount          25432 non-null  float64
 6   Currency                 25433 non-null  object
 7   Subscription Start Date  38611 non-null  object
 8   Subscription Expiration  38611 non-null  object
 9   Months Subscribed        38611 non-null  int64
 10  Demo User                38611 non-null  object
 11  Free Trial User          38611 non-null  object
 12  Free Trial Start Date    5408 non-null   object
 13  Free Trial Expiration    5408 non-null   object
 14  Auto Renew               38610 non-null  object
 15  Country                  38611 non-null  object
 16  User Type                38611 non-null  object
 17  Lead Platform            38611 non-null  object
 18  Email Subscriber         38611 non-null  object
 19  Push Notifications       38611 non-null  object
 20  Send Count               27525 non-null  float64
 21  Open Count               27525 non-null  float64
 22  Click Count              27525 non-null  float64
 23  Unique Open Count        27525 non-null  float64
 24  Unique Click Count       27525 non-null  float64
 25  Open Percentage          38611 non-null  object
dtypes: float64(6), int64(2), object(18)
memory usage: 7.7+ MB
```

[5]:
```
cleanedData.describe()
```

[5]:

	ID	Purchase Amount	Months Subscribed	Send Count	\
count	38611.000000	25432.000000	38611.000000	27525.000000	
mean	20180.489627	87.869182	167.338271	33.177475	
std	11550.045465	68.608895	352.128589	59.164482	
min	1.000000	0.000000	0.000000	1.000000	
25%	10208.500000	38.850000	3.000000	4.000000	
50%	20266.000000	61.650000	9.000000	10.000000	
75%	30214.500000	119.000000	15.000000	35.000000	
max	40000.000000	996.030000	963.000000	4370.000000	

	Open Count	Click Count	Unique Open Count	Unique Click Count
count	27525.000000	27525.000000	27525.000000	27525.000000
mean	8.467284	2.214242	4.004469	0.372316
std	37.975546	29.849956	13.336503	1.148135
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	1.000000	0.000000
75%	6.000000	0.000000	2.000000	0.000000
max	4365.000000	4348.000000	196.000000	44.000000

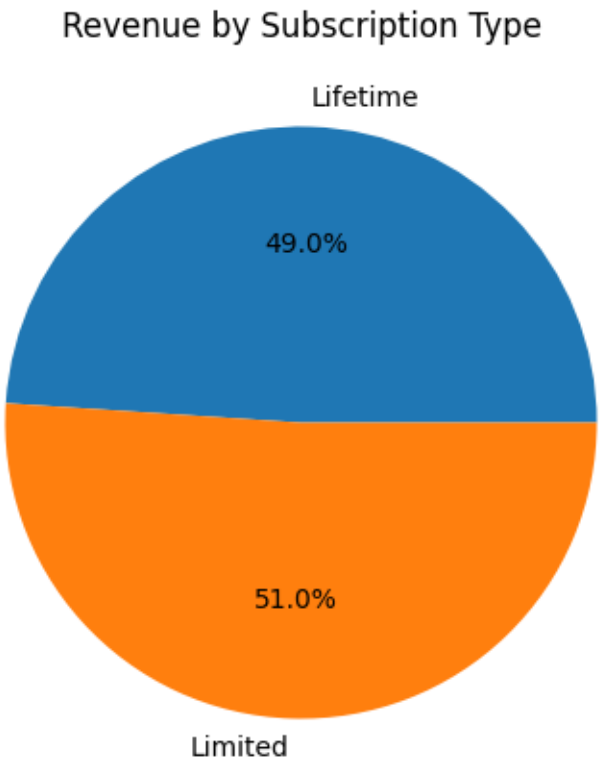
1 Customer Value Insights

It is important to note for these insights, we replaced all App purchase amounts in the subscriber data with the averages observed from the Web Purchase amount data based on language and subscription type. This allows us to perserve the most data, and get as close as possible to real purchase amounts based on the language and subscription type. Using this method allowed us to perserve almost half the number of observations in this dataset. This module includes various visuals and insight generation code the group used to identify trends and insights we wanted to pursue further.

[6]:
```
appfilled = cleanedData.loc[appFilled['Purchase Store'] == 'Web']
subTypeRev = appfilled.groupby('Subscription Type').sum()
plot = subTypeRev['Purchase Amount'].plot.pie(autopct='%.1f%%',figsize=(6, 5))
plot.set_title('Revenue by Subscription Type')
plot.set_ylabel(None)
```

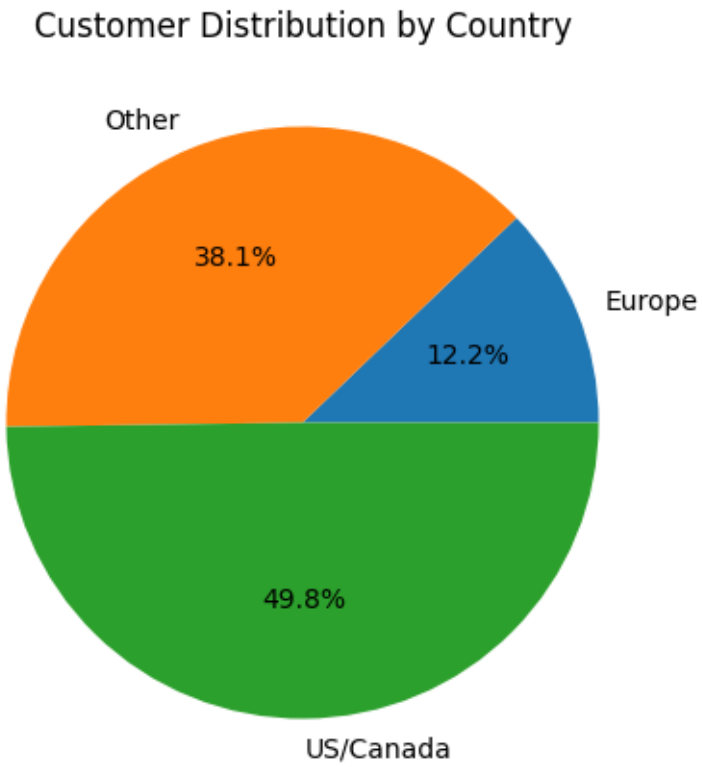
```
<ipython-input-6-701a4996a0e0>:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.sum is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[6]:
```
Text(0, 0.5, '')
```

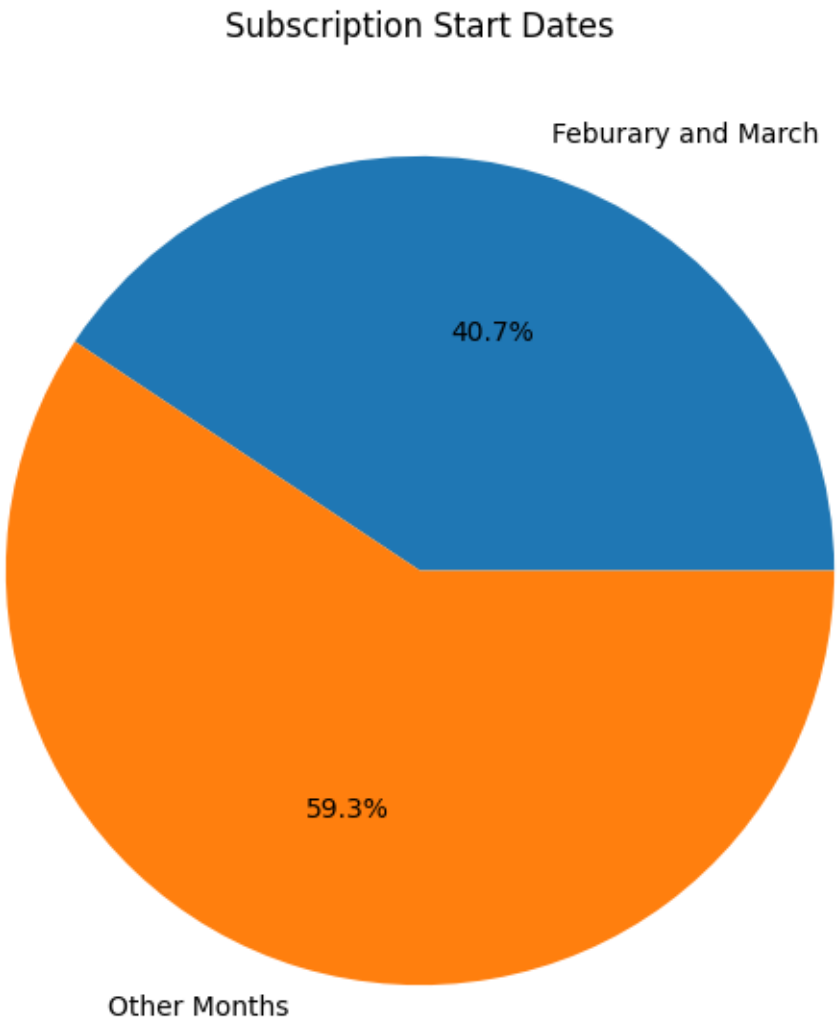


[7]:
```
countryDistro = cleanedData.groupby('Country').count()
plot = countryDistro['ID'].plot.pie(autopct='%.1f%%',figsize=(6, 5))
plot.set_title('Customer Distribution by Country')
plot.set_ylabel(None)
```

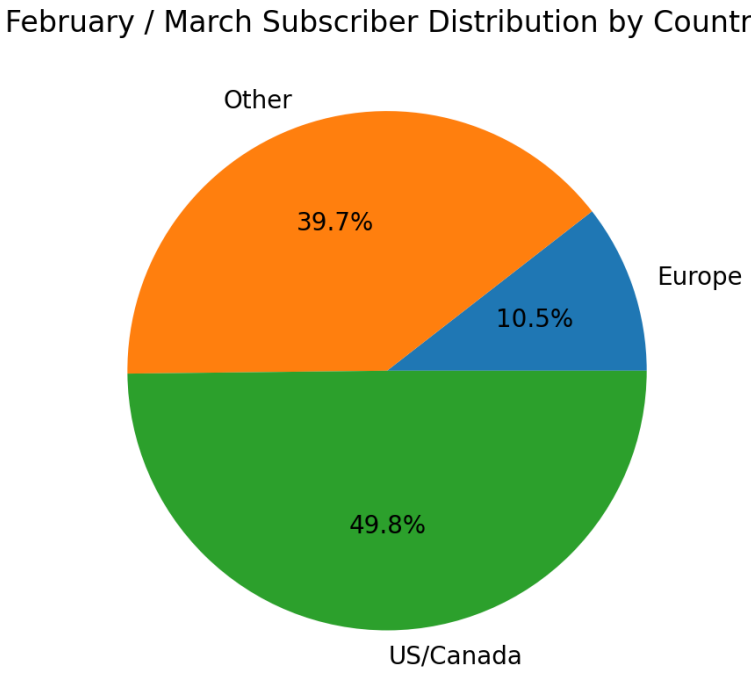
[7]: Text(0, 0.5, '')

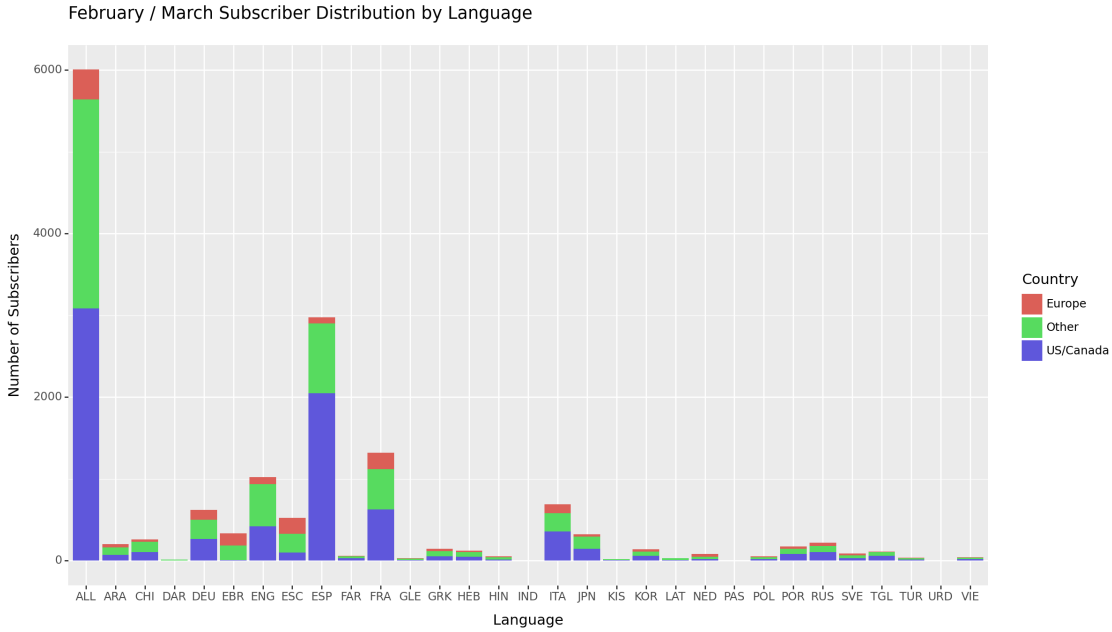


[8]:
```
dataMonths = pd.read_csv('cleanDataMonths.csv')
janFeb = dataMonths.loc[(dataMonths['Months'] == 3) | (dataMonths['Months'] ==␣
↪2)]
janFebCount = janFeb['ID'].count()
allData = dataMonths['ID'].count()
otherMonths = allData-janFebCount
data = [janFebCount, otherMonths]
labels = ['Feburary and March', 'Other Months']
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = labels,autopct='%.1f%%')
plt.title('Subscription Start Dates')
# show plot
plt.show()
```



[9]:
```
dataMonths = pd.read_csv('cleanDataMonths.csv')
janFeb = dataMonths.loc[(dataMonths['Months'] == 3) | (dataMonths['Months'] ==␣
↪2)]
counts = janFeb.groupby('Country').count()
plot = counts['ID'].plot.pie(autopct='%.1f%%')
plot.set_ylabel(None)
plot.set_title('February / March Subscriber Distribution by Country')
(ggplot(janFeb, aes(x='Language', fill='Country')) + geom_bar() +␣
↪theme(figure_size = (12, 7)) + labs(title = 'February / March Subscriber␣
↪Distribution by Language', x="Language",y='Number of Subscribers'))
```


February / March Subscriber Distribution by Country
Other
39.7%
Europe
10.5%
49.8%
US/Canada


February / March Subscriber Distribution by Language
Number of Subscribers
6000
4000
2000
0
ALL ARA CHI DAR DEU EBR ENG ESC ESP FAR FRA GLE GRK HEB HIN IND ITA JPN KIS KOR LAT NED PAS POL POR RUS SVE TGL TUR URD VIE
Language
Country
Europe
Other
US/Canada

[9]: <Figure Size: (1200 x 700)>

[10]:
```
webPurchases = cleanedData.copy()
webPurchases = webPurchases.loc[(cleanedData['Purchase Store']=='Web')].count()
subgroup = cleanedData.loc[(cleanedData['Subscription␣
 ↪Type']=='Lifetime')&(cleanedData['Language']=='ALL')&(cleanedData['Purchase␣
 ↪Store']=='Web')].count()
print(subgroup['Purchase Amount'] / (webPurchases['Purchase Amount']))
```

```
0.17917292638570412
```

[11]:
```
webPurchases = cleanedData.copy()
webPurchases = webPurchases.loc[(cleanedData['Purchase Store']=='Web')].sum()
subgroup = cleanedData.loc[(cleanedData['Subscription␣
 ↪Type']=='Lifetime')&(cleanedData['Language']=='ALL')&(cleanedData['Purchase␣
 ↪Store']=='Web')].sum()
print(subgroup['Purchase Amount'] / (webPurchases['Purchase Amount']))
```

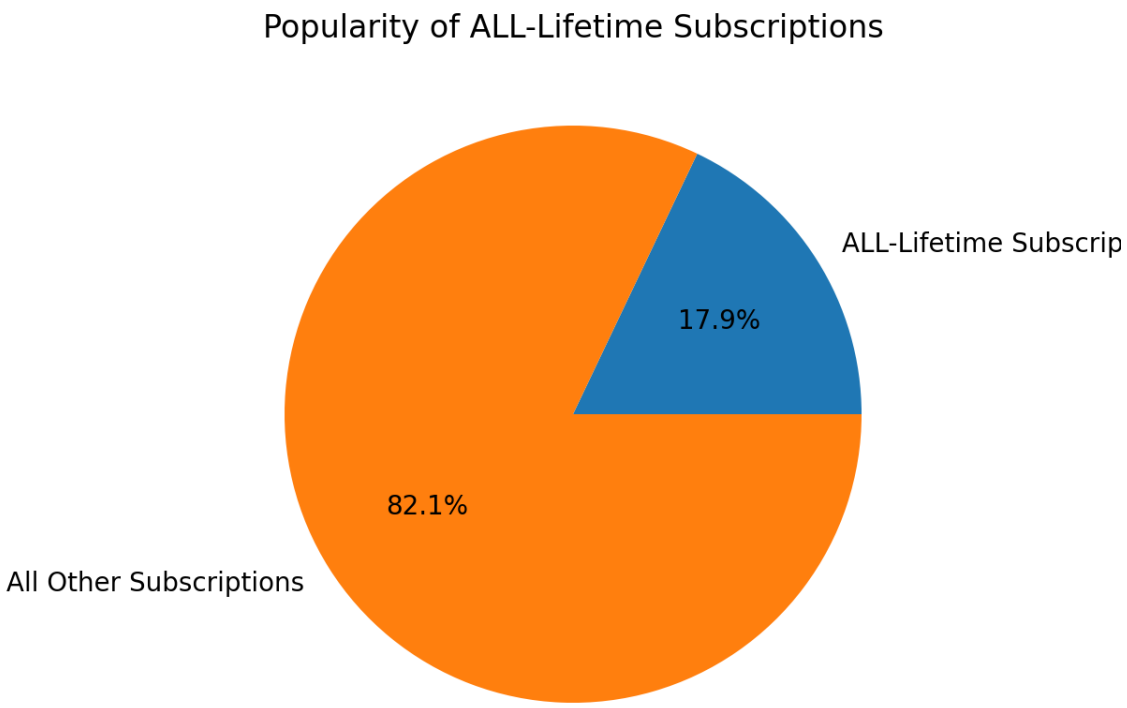
```
0.4177812375834651

<ipython-input-11-8508a8131256>:2: FutureWarning: The default value of
numeric_only in DataFrame.sum is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
<ipython-input-11-8508a8131256>:3: FutureWarning: The default value of
numeric_only in DataFrame.sum is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
```

[12]:
```
languageRevenue = appFilled.copy()
eachLangLifeTime = languageRevenue.loc[languageRevenue['Purchase Store'] ==␣
 ↪'Web'].groupby('Language')
langProfit = cleanedData.loc[languageRevenue['Purchase Store'] == 'Web']
langProfit['Lifetime-All'] = 'All Other Subscriptions'
langProfit.loc[(langProfit["Language"] == "ALL") & (langProfit["Subscription␣
 ↪Type"] == "Lifetime"), "Lifetime-All"] = 'ALL-Lifetime Subscriptions'
sorted=langProfit.groupby('Lifetime-All').count()
plot = sorted['ID'].plot.pie(autopct='%.1f%%', figsize=(6, 5))
plot.set_title('Popularity of ALL-Lifetime Subscriptions')
plot.set_ylabel(None)
```

```
<ipython-input-12-2678e650ba83>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[12]:
```
Text(0, 0.5, '')
```



Average Purchase Amount by Country This shows us the average purchase price per country. This information is based only on WEB purchases so we have the closest to accurate purchase amounts.

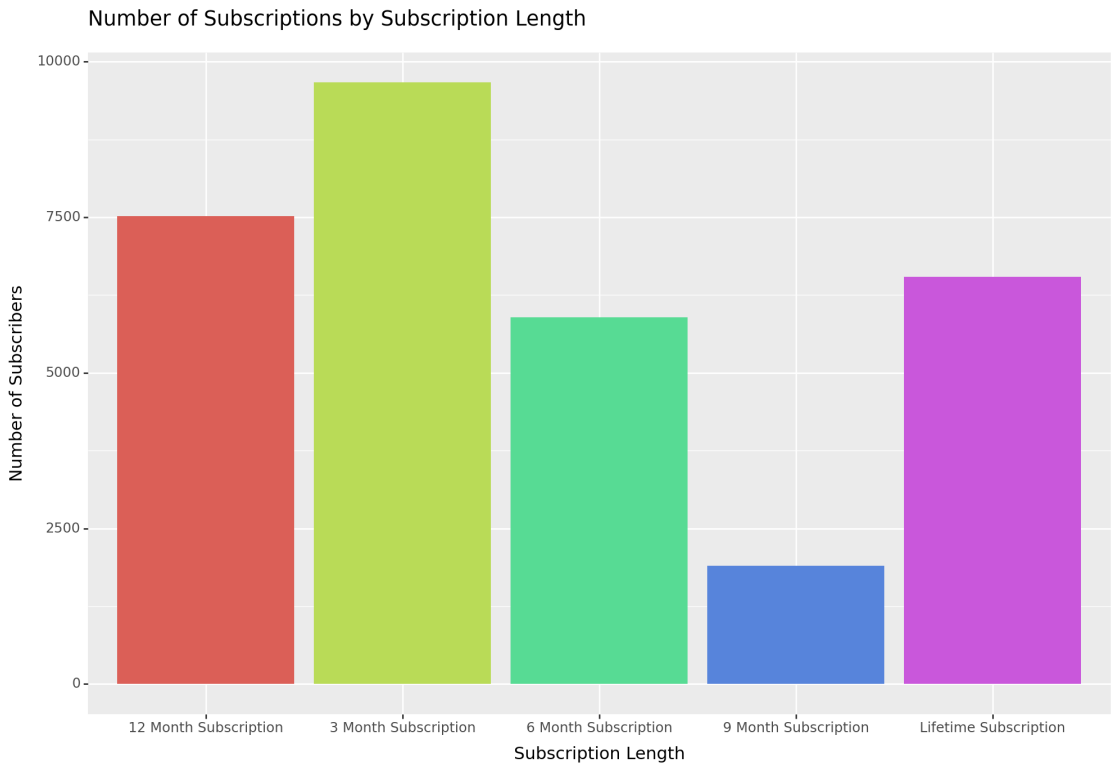
[13]:
```
country = cleanedData.loc[(cleanedData['Purchase Store'] == 'Web')]
countryPurchaseMean = pd.DataFrame(country.groupby(['Country']).mean().
 ↪apply(lambda x: round(x, 2)))
countryPurchaseMean['Purchase Amount']
# country['Months Subscribed'].mean()
# trialGroup['Subscription Type'].count()
```

```
<ipython-input-13-06aca8e098e1>:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[13]:
```
Country
Europe       64.90
Other        87.97
US/Canada    94.88
Name: Purchase Amount, dtype: float64
```

Number of Subscriptions by Subscription Length: From our grouping below, we are able to see our most popular subscription lengths. We have displayed the top ten lengths, but it is important to note that the top 6 are on certain time blocks. It is also important to note that 1000 represents a lifetime subscription. From this data, we can see that the 3 month subscription period is our most popular but number of subscription types, followed by 12 months, lifetime and 6 months.

[14]:
```
subLengthCount = pd.DataFrame(appFilled.groupby(['Months Subscribed']).count().
 ↪apply(lambda x: round(x, 2)))
sortedCount = subLengthCount['Purchase Amount'].sort_values(ascending=False).
 ↪head(5)
sortedCount = sortedCount.reset_index()
for index,row in sortedCount.iterrows():
  length = row['Months Subscribed']
  newVal = f'{length} Month Subscription'
  if length != 1000:
    sortedCount.loc[sortedCount['Months Subscribed'] == length, 'Months␣
 ↪Subscribed'] = newVal
  else:
    sortedCount.loc[sortedCount['Months Subscribed'] == length, 'Months␣
 ↪Subscribed'] = 'Lifetime Subscription'
(ggplot(sortedCount, aes(x='Months Subscribed', y='Purchase Amount',␣
 ↪fill='Months Subscribed')) + geom_bar(stat='identity', show_legend=False) +␣
 ↪theme(figure_size = (10, 7)) + labs(title = 'Number of Subscriptions by␣
 ↪Subscription Length', x="Subscription Length",y='Number of Subscribers'))
```



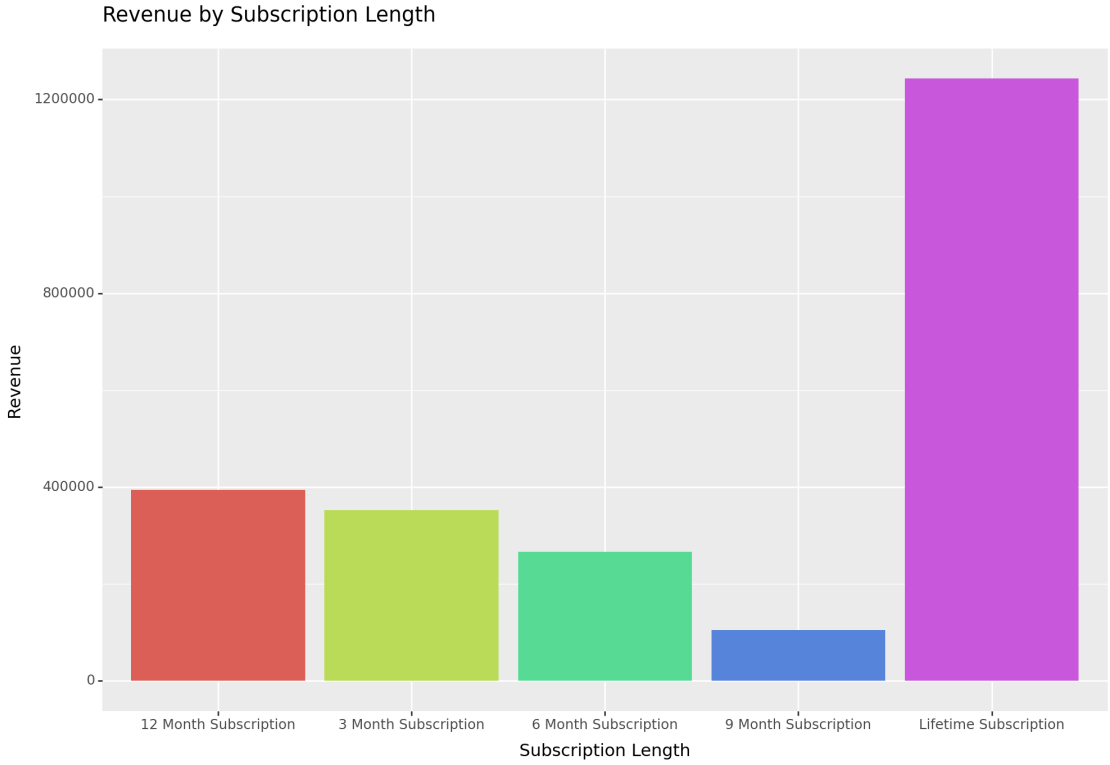
[14]: <Figure Size: (1000 x 700)>

Number of Subscriptions by Revenue: It is first important to again note that, 1000 represents a lifetime subscription. From a different look from the data above, while we observed lifetime being our 3rd ranked subscription by popularity, we actually see it is by far our Number 1 in driving revenue. With almost a 4x, the next highest revenue, we clearly understand and visualize that our lifetime subscribers drive the most revenue overall based on subscription types.

[15]:
```
subLengthRevenue = pd.DataFrame(appFilled.groupby(['Months Subscribed']).sum().
 ↪apply(lambda x: round(x, 2)))
subLengthRevenue = subLengthRevenue['Purchase Amount'].
 ↪sort_values(ascending=False).head(5)
subLengthRevenue = subLengthRevenue.reset_index()

for index,row in subLengthRevenue.iterrows():
  length = int(row['Months Subscribed'])
  newVal = f'{length} Month Subscription'
  if length != 1000:
    subLengthRevenue.loc[subLengthRevenue['Months Subscribed'] == length,␣
 ↪'Months Subscribed'] = newVal
  else:
```

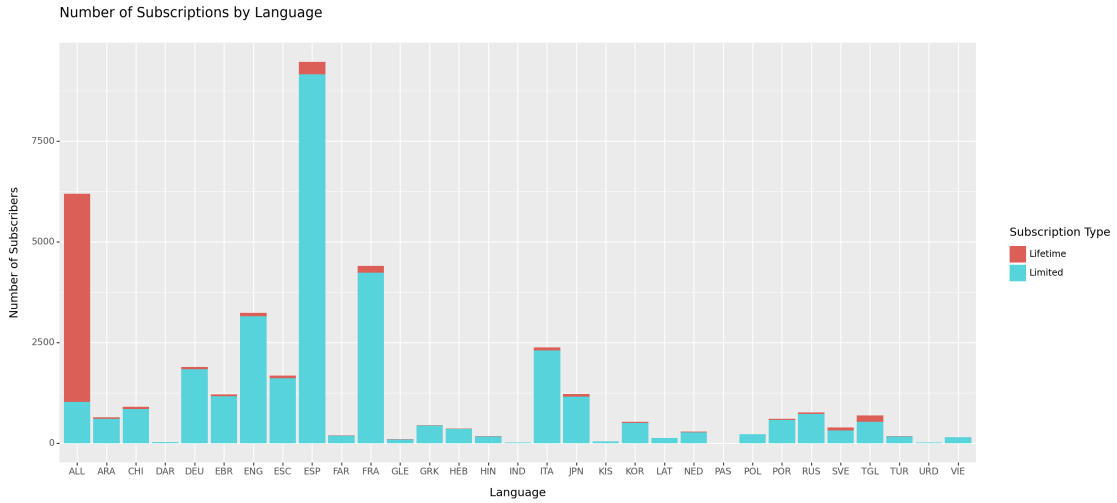
```
    subLengthRevenue.loc[subLengthRevenue['Months Subscribed'] == length,␣
 ↪'Months Subscribed'] = 'Lifetime Subscription'
(ggplot(subLengthRevenue, aes(x='Months Subscribed', y='Purchase Amount', fill␣
 ↪= 'Months Subscribed')) + geom_bar(stat='identity', show_legend=False) +␣
 ↪theme(figure_size = (10, 7)) + labs(title = 'Revenue by Subscription␣
 ↪Length',x="Subscription Length", y='Revenue'))
```

```
<ipython-input-15-7785b511f7d6>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.sum is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```



[15]:
```
<Figure Size: (1000 x 700)>
```

[16]:
```
(ggplot(cleanedData, aes(x='Language', fill = 'Subscription Type')) +␣
 ↪geom_bar() + theme(figure_size = (15, 7)) + labs(title = 'Number of␣
 ↪Subscriptions by Language', y='Number of Subscribers'))
# (ggplot(languages, aes(x = 'Language', fill = 'Subscription Type')))
```

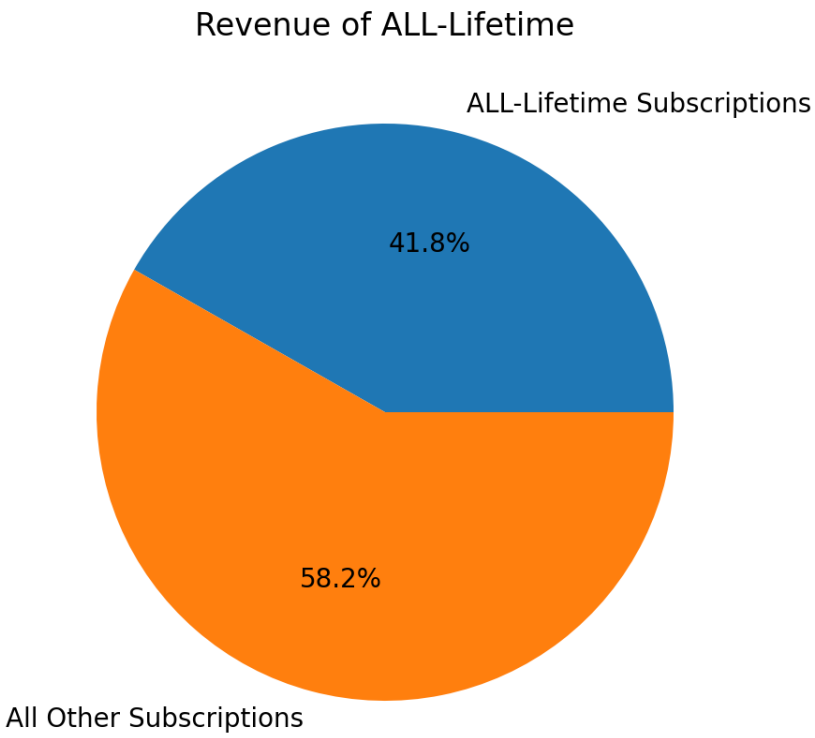


[16]: <Figure Size: (1500 x 700)>

[17]:
```
sorted=langProfit.groupby('Lifetime-All').sum()
plot = sorted['Purchase Amount'].plot.pie(autopct='%.1f%%',figsize=(6, 5))
plot.set_title('Revenue of ALL-Lifetime')
plot.set_ylabel(None)
```

```
<ipython-input-17-c3cd3c0e8ea0>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.sum is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[17]: Text(0, 0.5, '')



#Merging App Activity Data with the Subscriber data. In this code, we are taking the counts of all User IDs that have recorder interactions in the App Activity dataset and we are including this number of interactions in 5 different columns relating back into those specific App interactions.

[18]:
```
#merge in completed and other stuff.
interaction = pd.DataFrame(appActivity.groupby('ID').count())
interaction['ID'] = interaction.index
interaction.reset_index(drop = True, inplace = True)
interaction.rename(columns={'Unnamed: 0': 'Total Interaction'})
merged = appFilled.merge(interaction,on='ID',how='left')
merged.rename(columns={'Unnamed: 0': 'Total Interaction'})
merged.columns = ['ID', 'Language', 'Subscription Type', 'Subscription Event␣
 ↪Type',
       'Purchase Store', 'Purchase Amount', 'Currency',
       'Subscription Start Date', 'Subscription Expiration',
       'Months Subscribed', 'Demo User', 'Free Trial User',
       'Free Trial Start Date', 'Free Trial Expiration', 'Auto Renew',
       'Country', 'User Type', 'Lead Platform', 'Email Subscriber',
```

```
       'Push Notifications', 'Send Count', 'Open Count', 'Click Count',
       'Unique Open Count', 'Unique Click Count', 'Total App Interactions',
       'App Launch Count', 'App Start Count']
merged['App Launch Count'] = 0
merged['App Start Count'] = 0
merged['App Completed Count'] = 0
merged['App Other Count'] = 0
merged['App Onboarding Count'] = 0
merged
```

[18]:

```
           ID Language Subscription Type Subscription Event Type  \
0           1      POR           Limited        INITIAL_PURCHASE
1           2      EBR           Limited        INITIAL_PURCHASE
2           3      ESP           Limited        INITIAL_PURCHASE
3           4      KOR           Limited        INITIAL_PURCHASE
4           5      ENG           Limited        INITIAL_PURCHASE
…       …      …                 …                    …
38606   39996      DEU           Limited        INITIAL_PURCHASE
38607   39997      FRA           Limited                RENEWAL
38608   39998      TUR           Limited                RENEWAL
38609   39999      FRA           Limited        INITIAL_PURCHASE
38610   40000      ESP           Limited                RENEWAL

      Purchase Store  Purchase Amount Currency Subscription Start Date  \
0                App            55.83      USD                12/28/18
1                Web            39.00      USD                11/28/19
2                Web             0.00      USD                12/31/18
3                App            52.97      USD                 11/7/19
4                App            54.49      USD                 8/13/19
…              …                …      …                    …
38606            Web            48.74      USD                 6/20/19
38607            App            52.08      USD                 2/24/19
38608            Web            12.50      USD                 5/15/19
38609            App            52.08      USD                12/30/19
38610            App            61.65      USD                 12/2/18

      Subscription Expiration  Months Subscribed  … Open Count Click Count  \
0                     6/28/19                  6  …        7.0         0.0
1                     2/28/20                  3  …        3.0         0.0
2                    12/31/19                 12  …        0.0         0.0
3                      2/7/20                  3  …        0.0         0.0
4                    11/13/19                  3  …        5.0         1.0
…                       …                …  …  …        …
38606                 4/14/20                  9  …        0.0         0.0
38607                 8/23/19                  5  …        NaN         NaN
38608                 8/16/19                  3  …        NaN         NaN
38609                 3/30/20                  3  …        0.0         0.0
```

```
38610                12/5/19                12  …        NaN         NaN

      Unique Open Count Unique Click Count Total App Interactions  \
0                   6.0                0.0                   14.0
1                   1.0                0.0                   77.0
2                   0.0                0.0                   76.0
3                   0.0                0.0                   38.0
4                   5.0                1.0                  151.0
…                   …                  …                     …
38606               0.0                0.0                    NaN
38607               NaN                NaN                    NaN
38608               NaN                NaN                    NaN
38609               0.0                0.0                    NaN
38610               NaN                NaN                    NaN

      App Launch Count App Start Count App Completed Count App Other Count  \
0                    0               0                   0                0
1                    0               0                   0                0
2                    0               0                   0                0
3                    0               0                   0                0
4                    0               0                   0                0
…                    …               …                   …                …
38606                0               0                   0                0
38607                0               0                   0                0
38608                0               0                   0                0
38609                0               0                   0                0
38610                0               0                   0                0

      App Onboarding Count
0                        0
1                        0
2                        0
3                        0
4                        0
…                        …
38606                    0
38607                    0
38608                    0
38609                    0
38610                    0

[38611 rows x 31 columns]
```

[19]:
```
activityType = pd.DataFrame(appActivity.groupby(['ID', 'App Activity Type']).
  ↪count())
activityType = activityType.reset_index()
activityType
```

```
for index,row in activityType.iterrows():
  id = row['ID']
  actType = row['App Activity Type']
  if actType == 'App Launch':
    merged.loc[merged['ID'] == id, 'App Launch Count'] = row['App Session␣
 ↪Platform']
  elif actType == 'Completed':
    merged.loc[merged['ID'] == id, 'App Completed Count'] = row['App Session␣
 ↪Platform']
  elif actType == 'Start':
    merged.loc[merged['ID'] == id, 'App Start Count'] = row['App Session␣
 ↪Platform']
  elif actType == 'Onboarding':
    merged.loc[merged['ID'] == id, 'App Onboarding Count'] = row['App Session␣
 ↪Platform']
  elif actType == 'Other':
    merged.loc[merged['ID'] == id, 'App Other Count'] = row['App Session␣
 ↪Platform']
merged
```

[19]:

	ID	Language	Subscription Type	Subscription Event Type	\
0	1	POR	Limited	INITIAL_PURCHASE	
1	2	EBR	Limited	INITIAL_PURCHASE	
2	3	ESP	Limited	INITIAL_PURCHASE	
3	4	KOR	Limited	INITIAL_PURCHASE	
4	5	ENG	Limited	INITIAL_PURCHASE	
…	…	…	…	…	
38606	39996	DEU	Limited	INITIAL_PURCHASE	
38607	39997	FRA	Limited	RENEWAL	
38608	39998	TUR	Limited	RENEWAL	
38609	39999	FRA	Limited	INITIAL_PURCHASE	
38610	40000	ESP	Limited	RENEWAL	

	Purchase Store	Purchase Amount	Currency	Subscription Start Date	\
0	App	55.83	USD	12/28/18	
1	Web	39.00	USD	11/28/19	
2	Web	0.00	USD	12/31/18	
3	App	52.97	USD	11/7/19	
4	App	54.49	USD	8/13/19	
…	…	…	…	…	
38606	Web	48.74	USD	6/20/19	
38607	App	52.08	USD	2/24/19	
38608	Web	12.50	USD	5/15/19	
38609	App	52.08	USD	12/30/19	
38610	App	61.65	USD	12/2/18	

	Subscription Expiration	Months Subscribed	…	Open Count	Click Count	\

```
0                    6/28/19                 6  …        7.0         0.0
1                    2/28/20                 3  …        3.0         0.0
2                   12/31/19                12  …        0.0         0.0
3                     2/7/20                 3  …        0.0         0.0
4                   11/13/19                 3  …        5.0         1.0
…                        …                …  …        …          …
38606                4/14/20                 9  …        0.0         0.0
38607                8/23/19                 5  …        NaN         NaN
38608                8/16/19                 3  …        NaN         NaN
38609                3/30/20                 3  …        0.0         0.0
38610                12/5/19                12  …        NaN         NaN

      Unique Open Count Unique Click Count Total App Interactions  \
0                   6.0                0.0                   14.0
1                   1.0                0.0                   77.0
2                   0.0                0.0                   76.0
3                   0.0                0.0                   38.0
4                   5.0                1.0                  151.0
…                   …                  …                     …
38606               0.0                0.0                    NaN
38607               NaN                NaN                    NaN
38608               NaN                NaN                    NaN
38609               0.0                0.0                    NaN
38610               NaN                NaN                    NaN

      App Launch Count App Start Count App Completed Count App Other Count  \
0                   12               0                   2               0
1                   27              12                  16              22
2                   39               0                  37               0
3                   15               8                   6               9
4                   63              38                  21              29
…                   …                …                   …               …
38606                0               0                   0               0
38607                0               0                   0               0
38608                0               0                   0               0
38609                0               0                   0               0
38610                0               0                   0               0

      App Onboarding Count
0                        0
1                        0
2                        0
3                        0
4                        0
…                        …
38606                    0
38607                    0
```

```
38608                    0
38609                    0
38610                    0

[38611 rows x 31 columns]
```

2 Various Models with WEB Purchase Stores only

These models include a linear regression to predict price, a random forest model to predict price, and coefficent plots, to graph importance the model weighed variables with for further analysis. This model performed decently well in predicting power, but the group only used the importance plots to identify variables to look further into and analyze.

[20]:
```
purchaseForrest = cleanedData.loc[cleanedData['Purchase Store']=='Web']
purchaseForrest = purchaseForrest.dropna(axis=0, subset=['Send Count', 'Open␣
 ↪Count', 'Click Count', 'Unique Open Count', 'Unique Click Count'])
#There is a single NULL for autorenew so dropped.
purchaseForrest = purchaseForrest.dropna(subset=['Auto Renew'])
purchaseForrest['Auto Renew'] = purchaseForrest['Auto Renew'].replace({'Off':␣
 ↪0, 'On': 1})
purchaseForrest['Email Subscriber'] = purchaseForrest['Email Subscriber'].
 ↪replace({'No': 0, 'Yes': 1})
purchaseForrest['Push Notifications'] = purchaseForrest['Push Notifications'].
 ↪replace({'No': 0, 'Yes': 1})

purchaseForrest = pd.get_dummies(purchaseForrest, columns= ['Language',␣
 ↪'Subscription Type', 'Subscription Event Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Country', 'User Type', 'Lead Platform'])

predictors = ['Language_ALL',
       'Language_ARA', 'Language_CHI', 'Language_DAR', 'Language_DEU',
       'Language_EBR', 'Language_ENG', 'Language_ESC', 'Language_ESP',
       'Language_FAR', 'Language_FRA', 'Language_GLE', 'Language_GRK',
       'Language_HEB', 'Language_HIN', 'Language_IND', 'Language_ITA',
       'Language_JPN', 'Language_KIS', 'Language_KOR', 'Language_LAT',
       'Language_NED', 'Language_PAS', 'Language_POL', 'Language_POR',
       'Language_RUS', 'Language_SVE', 'Language_TGL', 'Language_TUR',
       'Language_URD', 'Language_VIE', 'Subscription Type_Lifetime',
       'Subscription Type_Limited', 'Subscription Event Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
        'Country_Europe', 'Country_Other', 'Auto Renew',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Unknown', 'Lead Platform_Web']
```

```
X = purchaseForrest[predictors]
y = purchaseForrest['Purchase Amount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=27)

rf = RandomForestRegressor(n_estimators=100, random_state=0)

rf.fit(X_train, y_train)

# Predict on the test set results

y_pred_100 = rf.predict(X_test)



# Check accuracy score

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. format(rf.
 ↪score(X_test, y_test)))

# important_features_dict = {}
# for idx, val in enumerate(rf.feature_importances_):
#     important_features_dict[idx] = val

# important_features_list = sorted(important_features_dict,
#                                  key=important_features_dict.get,
#                                  reverse=True)

# print(f'5 most important features: {important_features_list[:5]}')
# print(f'{predictors[32]}, {predictors[0]}, {predictors[40]},␣
 ↪{predictors[20]}, {predictors[33]}')
```

```
Model accuracy score with 100 decision-trees : 0.8040
```

[21]:
```
web = cleanedData.loc[cleanedData['Purchase Store']=='Web']
#There is a single NULL for autorenew so dropped.
web = web.dropna(subset=['Auto Renew'])
predictors = ['Months Subscribed', 'Language_ALL',
       'Language_ARA', 'Language_CHI', 'Language_DAR', 'Language_DEU',
       'Language_EBR', 'Language_ENG', 'Language_ESC', 'Language_ESP',
       'Language_FAR', 'Language_FRA', 'Language_GLE', 'Language_GRK',
       'Language_HEB', 'Language_HIN', 'Language_IND', 'Language_ITA',
       'Language_JPN', 'Language_KIS', 'Language_KOR', 'Language_LAT',
       'Language_NED', 'Language_PAS', 'Language_POL', 'Language_POR',
       'Language_RUS', 'Language_SVE', 'Language_TGL', 'Language_TUR',
```

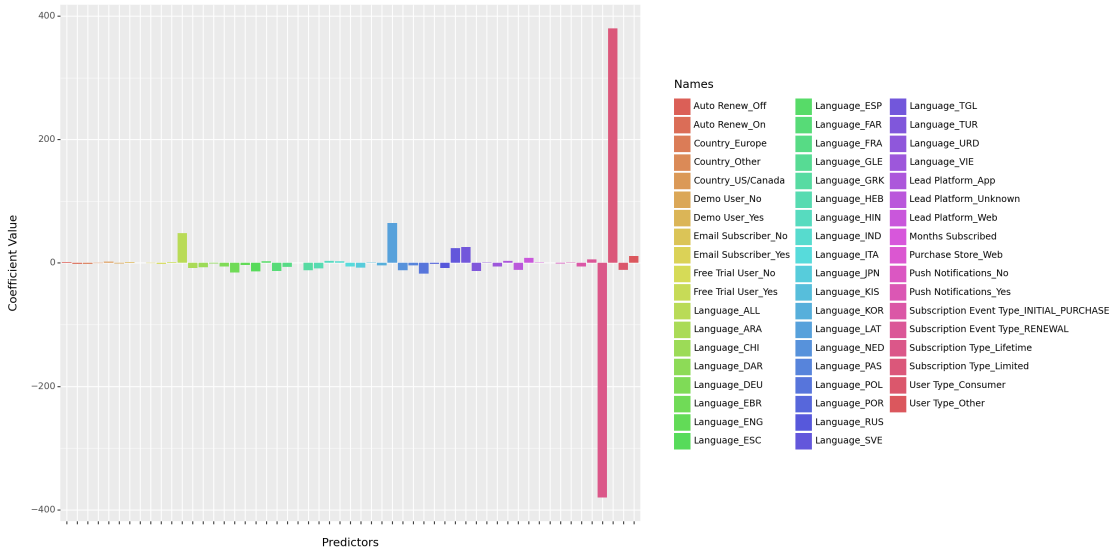
```
       'Language_URD', 'Language_VIE', 'Subscription Type_Lifetime',
       'Subscription Type_Limited', 'Subscription Event Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
       'Auto Renew_Off', 'Auto Renew_On', 'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Unknown', 'Lead Platform_Web',
       'Email Subscriber_No', 'Email Subscriber_Yes', 'Push Notifications_No',
       'Push Notifications_Yes']

web = pd.get_dummies(web, columns= ['Language', 'Subscription Type',␣
 ↪'Subscription Event Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Auto Renew', 'Country', 'User Type', 'Lead Platform',
       'Email Subscriber', 'Push Notifications'])


X = web[predictors]
y = web["Purchase Amount"]

#80/20 TTS
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=27)

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
#data frames for comparing results of the model in sample and out of sample.
true_vs_pred_test = pd.DataFrame({"Predicted": y_pred_test, "True": y_test})
true_vs_pred_train = pd.DataFrame({"Predicted": y_pred_train, "True": y_train})
trainingR2 = lr.score(X_train, y_train)
testingR2 = lr.score(X_test, y_test)
trainingMAE = mean_absolute_error(true_vs_pred_train["Predicted"],␣
 ↪true_vs_pred_train["True"])
testingMAE = mean_absolute_error(true_vs_pred_test["Predicted"],␣
 ↪true_vs_pred_test["True"])

print(f"The TRAINING R2 is {trainingR2} / The TESTING R2 is {testingR2}.")
print(f"The TRAINING MAE {trainingMAE} / The TESTING MAE is: {testingMAE}.")
```

```
The TRAINING R2 is 0.7945252986657305 / The TESTING R2 is 0.7891464127503411.
The TRAINING MAE 19.116683763070174 / The TESTING MAE is: 19.092985453059217.
```

[22]:
```
coefficients = pd.DataFrame({"Coef": lr.coef_,
                             "Names": predictors})
```

```
print(lr.intercept_)
print(lr.coef_)
pd.options.display.float_format = '{:.0f}'.format
coefficients.sort_values(by=['Coef'], ascending=False)
(ggplot(coefficients, aes(x = "Names", y = "Coef", fill = "Names" )) +␣
 ↪geom_bar(stat = "identity") + theme(axis_text_x=element_blank(),␣
 ↪figure_size=(15,8)) + ylab("Coefficient Value") + xlab("Predictors"))
```

```
-322.6710538177396
[ 9.21028277e-01  4.79636819e+01 -8.10814522e+00 -6.83322088e+00
 -9.15559994e-01 -5.66855683e+00 -1.55545792e+01 -3.58254278e+00
 -1.37312747e+01  2.86795199e+00 -1.29011026e+01 -6.32535880e+00
  1.25736336e-02 -1.17782049e+01 -8.76429792e+00  3.14289503e+00
  2.94833542e+00 -6.05042788e+00 -7.44505953e+00  6.17981495e-01
 -3.72699895e+00  6.48482842e+01 -1.16771780e+01 -4.04020284e+00
 -1.72885635e+01 -1.60355696e+00 -8.40722275e+00  2.40491728e+01
  2.57081225e+01 -1.31615282e+01  9.55792746e-01 -5.55120921e+00
 -3.80149110e+02  3.80149110e+02 -5.76933147e+00  5.76933147e+00
  0.00000000e+00 -1.23498220e+00  1.23498220e+00 -1.49404438e+00
  1.49404438e+00  1.32831668e+00 -1.32831668e+00 -1.55514772e+00
 -2.54127224e-01  1.80927495e+00 -1.13612929e+01  1.13612929e+01
  3.24013089e+00 -1.14566845e+01  8.21655359e+00  3.87985073e-01
 -3.87985073e-01 -8.69645946e-01  8.69645946e-01]
```



[22]: <Figure Size: (1500 x 800)>

[23]:
```
webMonths = cleanedData.loc[(cleanedData['Purchase Store']=='Web') &␣
 ↪(cleanedData['Subscription Type']=='Limited')]
#There is a single NULL for autorenew so dropped.
webMonths = webMonths.dropna(subset=['Auto Renew'])
predictors = ["Purchase Amount", 'Language_ALL',
       'Language_ARA', 'Language_CHI', 'Language_DAR', 'Language_DEU',
       'Language_EBR', 'Language_ENG', 'Language_ESC', 'Language_ESP',
       'Language_FAR', 'Language_FRA', 'Language_GLE', 'Language_GRK',
       'Language_HEB', 'Language_HIN', 'Language_IND', 'Language_ITA',
       'Language_JPN', 'Language_KIS', 'Language_KOR', 'Language_LAT',
       'Language_NED', 'Language_PAS', 'Language_POL', 'Language_POR',
       'Language_RUS', 'Language_SVE', 'Language_TGL', 'Language_TUR',
       'Language_URD', 'Language_VIE', 'Subscription Event␣
 ↪Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
       'Auto Renew_Off', 'Auto Renew_On', 'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Unknown', 'Lead Platform_Web',
       'Email Subscriber_No', 'Email Subscriber_Yes', 'Push Notifications_No',
       'Push Notifications_Yes']

webMonths = pd.get_dummies(webMonths, columns= ['Language', 'Subscription Event␣
 ↪Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Auto Renew', 'Country', 'User Type', 'Lead Platform',
       'Email Subscriber', 'Push Notifications'])


X = webMonths[predictors]
y = webMonths['Months Subscribed']

#80/20 TTS
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=27)

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred_train = lr.predict(X_train)
y_pred_test = lr.predict(X_test)
#data frames for comparing results of the model in sample and out of sample.
true_vs_pred_test = pd.DataFrame({"Predicted": y_pred_test, "True": y_test})
true_vs_pred_train = pd.DataFrame({"Predicted": y_pred_train, "True": y_train})
trainingR2 = lr.score(X_train, y_train)
testingR2 = lr.score(X_test, y_test)
```

```
trainingMAE = mean_absolute_error(true_vs_pred_train["Predicted"],␣
 ↪true_vs_pred_train["True"])
testingMAE = mean_absolute_error(true_vs_pred_test["Predicted"],␣
 ↪true_vs_pred_test["True"])

print(f"The TRAINING R2 is {trainingR2} / The TESTING R2 is {testingR2}.")
print(f"The TRAINING MAE {trainingMAE} / The TESTING MAE is: {testingMAE}.")
```

```
The TRAINING R2 is 0.22598875871743862 / The TESTING R2 is 0.2270653825815373.
The TRAINING MAE 4.290113345566247 / The TESTING MAE is: 4.398161228382149.
```

[24]:
```
coefficients = pd.DataFrame({"Coef": lr.coef_,
                             "Names": predictors})
coefficients.sort_values(by=['Coef'], ascending=False)
```

[24]:

	Coef	Names
23	6	Language_PAS
28	4	Language_TGL
1	4	Language_ALL
27	3	Language_SVE
21	3	Language_LAT
45	3	User Type_Other
37	2	Free Trial User_No
52	2	Push Notifications_Yes
46	2	Lead Platform_App
40	1	Auto Renew_On
41	1	Country_Europe
35	1	Demo User_No
33	1	Subscription Event Type_RENEWAL
19	1	Language_KIS
50	0	Email Subscriber_Yes
6	0	Language_EBR
48	0	Lead Platform_Web
8	0	Language_ESC
13	0	Language_GRK
0	0	Purchase Amount
34	-0	Purchase Store_Web
30	-0	Language_URD
3	-0	Language_CHI
4	-0	Language_DAR
49	-0	Email Subscriber_No
14	-0	Language_HEB
29	-1	Language_TUR
43	-1	Country_US/Canada
42	-1	Country_Other
26	-1	Language_RUS
2	-1	Language_ARA

```
11    -1                                Language_FRA
32    -1  Subscription Event Type_INITIAL_PURCHASE
5     -1                                Language_DEU
31    -1                                Language_VIE
17    -1                                Language_ITA
25    -1                                Language_POR
10    -1                                Language_FAR
36    -1                               Demo User_Yes
24    -1                                Language_POL
22    -1                                Language_NED
16    -1                                Language_IND
12    -1                                Language_GLE
18    -1                                Language_JPN
39    -1                             Auto Renew_Off
9     -2                                Language_ESP
7     -2                                Language_ENG
51    -2                      Push Notifications_No
20    -2                                Language_KOR
47    -2                      Lead Platform_Unknown
38    -2                        Free Trial User_Yes
15    -2                                Language_HIN
44    -3                         User Type_Consumer
```

3 [Auto Renew] Logistical Regression Filled Purchase Data

[25]:
```
webLogit = appFilled.copy()
#There is a single NULL for autorenew so dropped.
webLogit = webLogit.dropna(subset=['Auto Renew'])
predictors = ["Purchase Amount", 'Language_ALL',
       'Language_ARA', 'Language_CHI', 'Language_DAR', 'Language_DEU',
       'Language_EBR', 'Language_ENG', 'Language_ESC', 'Language_ESP',
       'Language_FAR', 'Language_FRA', 'Language_GLE', 'Language_GRK',
       'Language_HEB', 'Language_HIN', 'Language_IND', 'Language_ITA',
       'Language_JPN', 'Language_KIS', 'Language_KOR', 'Language_LAT',
       'Language_NED',  'Language_POL', 'Language_POR',
       'Language_RUS', 'Language_SVE', 'Language_TGL', 'Language_TUR',
       'Language_URD', 'Language_VIE', 'Subscription Event␣
 ↪Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
        'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Unknown', 'Lead Platform_Web',
       'Email Subscriber_No', 'Email Subscriber_Yes', 'Push Notifications_No',
       'Push Notifications_Yes']
```

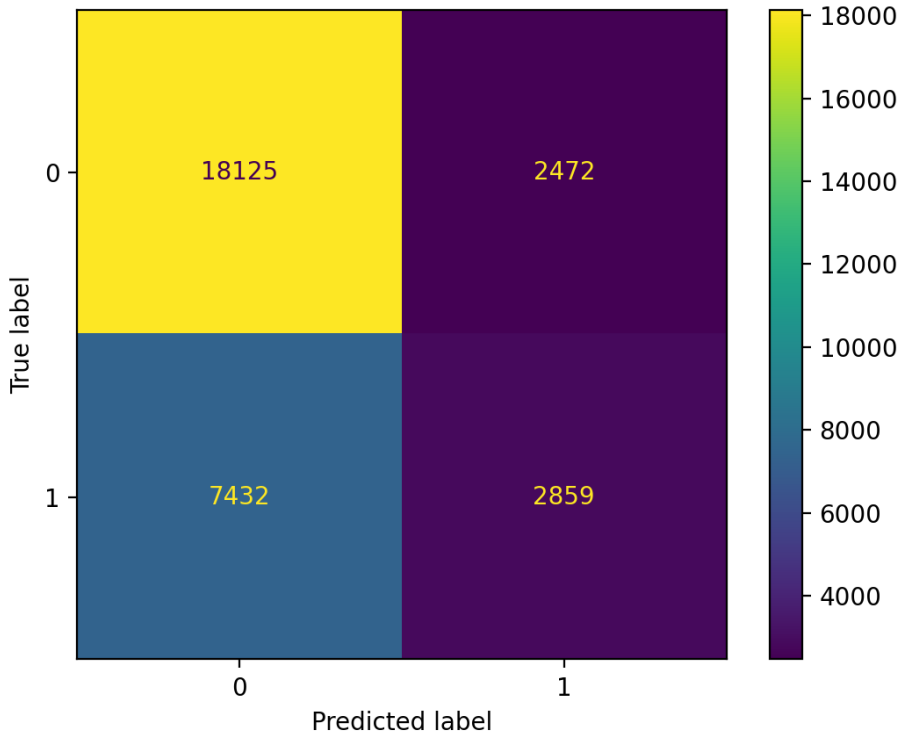
```
webLogit = pd.get_dummies(webLogit, columns= ['Language', 'Subscription Event␣
 ↪Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Auto Renew', 'Country', 'User Type', 'Lead Platform',
       'Email Subscriber', 'Push Notifications'])


X = webLogit[predictors]
y = webLogit['Auto Renew_On']

#80/20 TTS
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=5)


logit = LogisticRegression(penalty = "none")

logit.fit(X_train, y_train)


print("Train Accuracy:", end=" ")
print(accuracy_score(y_train, logit.predict(X_train)))
ConfusionMatrixDisplay.from_predictions(y_train, logit.predict(X_train))

print("Test Accuracy:", end=" ")
print(accuracy_score(y_test, logit.predict(X_test)))
ConfusionMatrixDisplay.from_predictions(y_test, logit.predict(X_test))
```

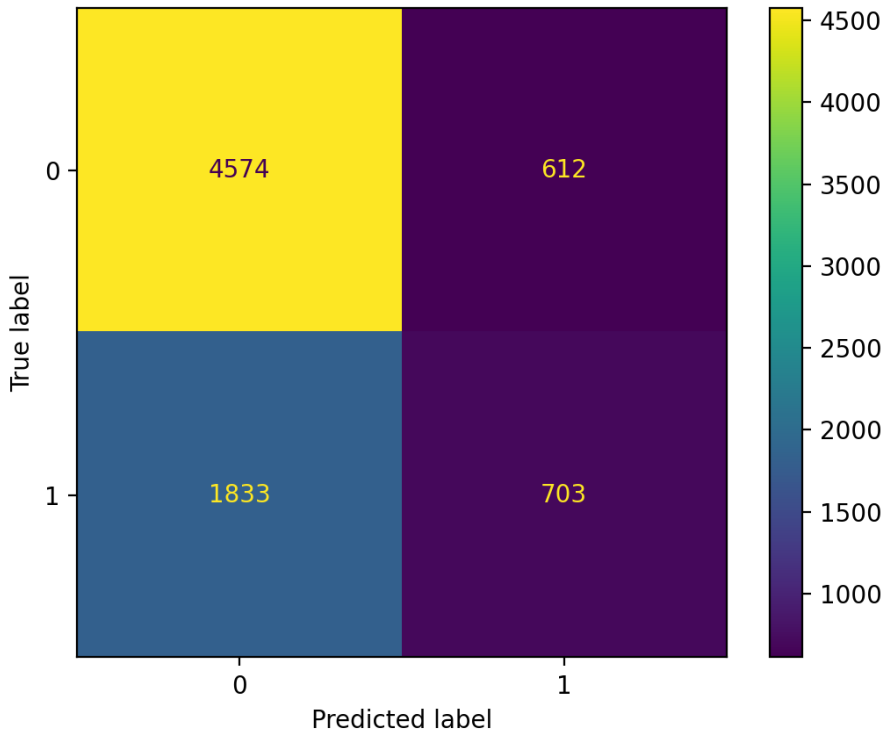
```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1173:
FutureWarning: `penalty='none'`has been deprecated in 1.2 and will be removed in
1.4. To keep the past behaviour, set `penalty=None`.
```

```
Train Accuracy: 0.6793576793576793
Test Accuracy: 0.6833721833721834
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

[25]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x792732763250>


18125
2472
7432
2859
0
1
True label
0
1
Predicted label
18000
16000
14000
12000
10000
8000
6000
4000



4 [Auto Renew] Logistical Regression with Merged Data

We see that there is no real gain from merging our data. These results are the same as if we just used the subscriber information with no app activity. This is an interesting outcome since we saw a ~2% increase of accuracy in our random forrest model with heavy importances on the new data introduced.

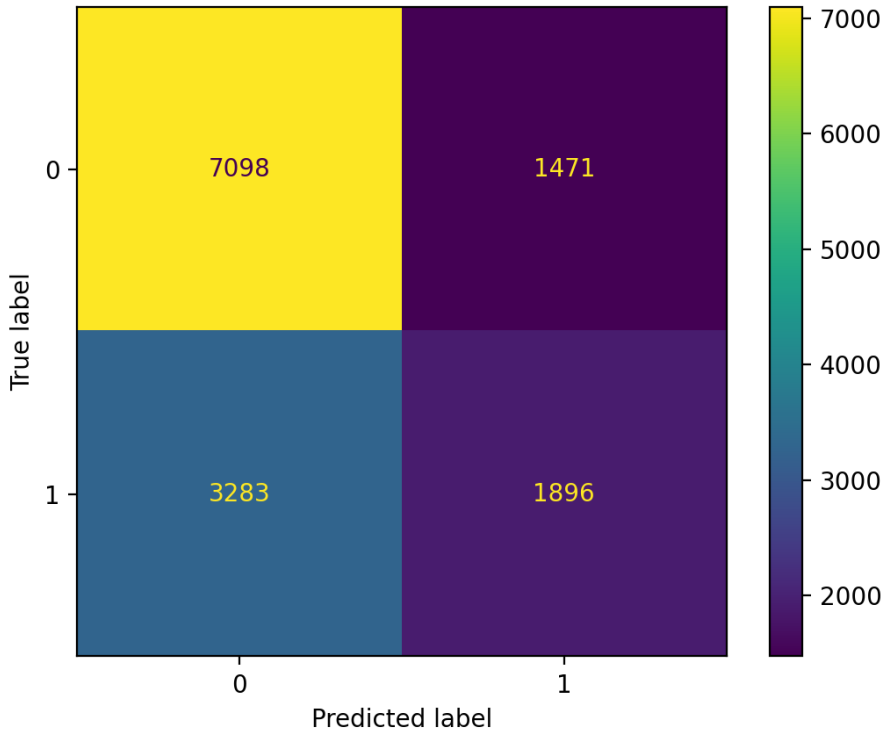
[26]:
```
mergedRF = merged.copy()
mergedRF = mergedRF.dropna(subset=['Send Count', 'Open Count', 'Click Count',␣
 ↪'Unique Open Count', 'Unique Click Count', 'Total App Interactions'])
#There is a single NULL for autorenew so dropped.
mergedRF = mergedRF.dropna(subset=['Auto Renew'])
mergedRF['Auto Renew'] = mergedRF['Auto Renew'].replace({'Off': 0, 'On': 1})
mergedRF['Email Subscriber'] = mergedRF['Email Subscriber'].replace({'No': 0,␣
 ↪'Yes': 1})
mergedRF['Push Notifications'] = mergedRF['Push Notifications'].replace({'No':␣
 ↪0, 'Yes': 1})

predictors = ["Purchase Amount", 'Subscription Event Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
```

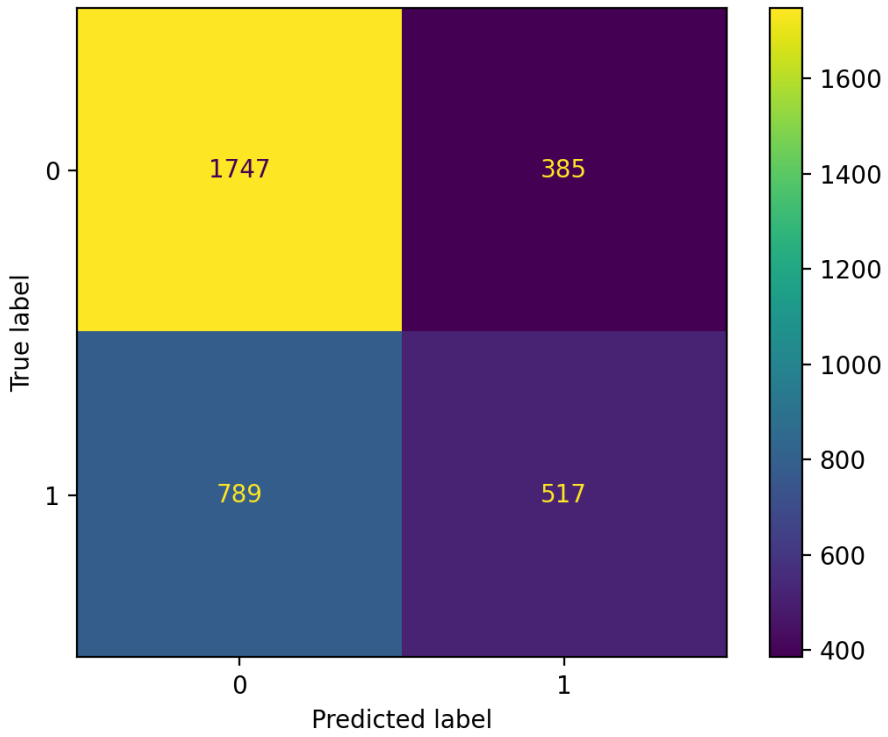
```
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
        'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Web', 'Total App Interactions', 'App␣
 ↪Launch Count', 'App Start Count', 'App Completed Count', 'App Other Count',␣
 ↪'App Onboarding Count']

mergedRF = pd.get_dummies(mergedRF, columns= ['Subscription Event Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Country', 'User Type', 'Lead Platform'])

X = mergedRF[predictors]
y = mergedRF['Auto Renew']

#80/20 TTS
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=5)


logit = LogisticRegression(penalty = "none")

logit.fit(X_train, y_train)


print("Train Accuracy:", end=" ")
print(accuracy_score(y_train, logit.predict(X_train)))
ConfusionMatrixDisplay.from_predictions(y_train, logit.predict(X_train))

print("Test Accuracy:", end=" ")
print(accuracy_score(y_test, logit.predict(X_test)))
ConfusionMatrixDisplay.from_predictions(y_test, logit.predict(X_test))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1173:
FutureWarning: `penalty='none'`has been deprecated in 1.2 and will be removed in
1.4. To keep the past behaviour, set `penalty=None`.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

```
Train Accuracy: 0.6542042478906023
Test Accuracy: 0.6585223967422921
```

[26]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7927302fcfa0>





5 [Auto Renew] Random Forrest Classifier Model

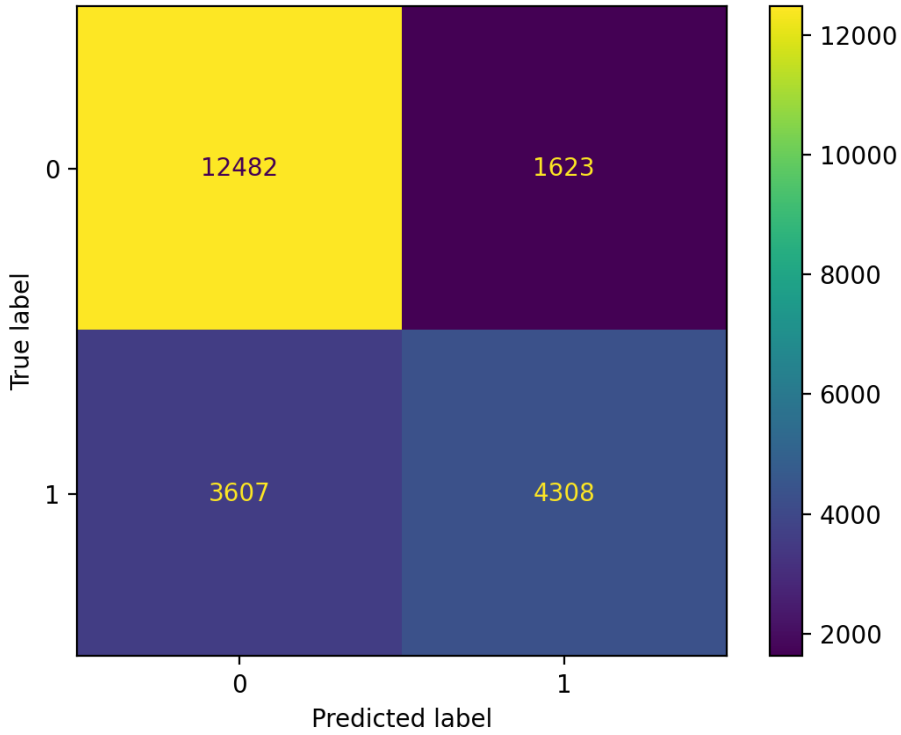
This model was used to classify users who will or will not have auto-renew on. In this forrest, we elected to only use data that was available for any app interaction metrics including Send Count, Open Count, Click Count, Unique Open Count, and Unique Click Count. With these metrics having null values, we decided for this model only to drop any rows with NULL values. This made our dataset go from 38,611 rows to 27,525. ## Results: We notice that this model predicts with about a 68.6% accuracy rate, and the forest weighs heavily on purchase price to classify appropriately. We notice a slight overfitting on our training data, but this difference does not seem like something we would need to worry about.

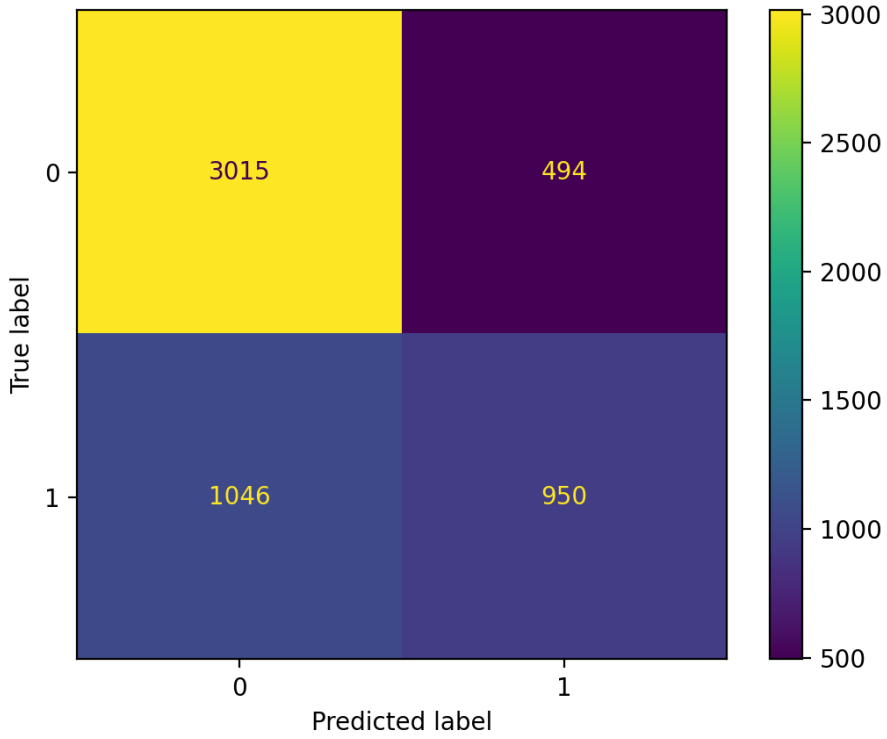
[27]:
```
autoRenew = appFilled.dropna(axis=0, subset=['Send Count', 'Open Count', 'Click␣
 ↪Count', 'Unique Open Count', 'Unique Click Count'])
#There is a single NULL for autorenew so dropped.
autoRenew = autoRenew.dropna(subset=['Auto Renew'])
autoRenew['Auto Renew'] = autoRenew['Auto Renew'].replace({'Off': 0, 'On': 1})
autoRenew['Email Subscriber'] = autoRenew['Email Subscriber'].replace({'No': 0,␣
 ↪'Yes': 1})
autoRenew['Push Notifications'] = autoRenew['Push Notifications'].replace({'No':
 ↪ 0, 'Yes': 1})
```

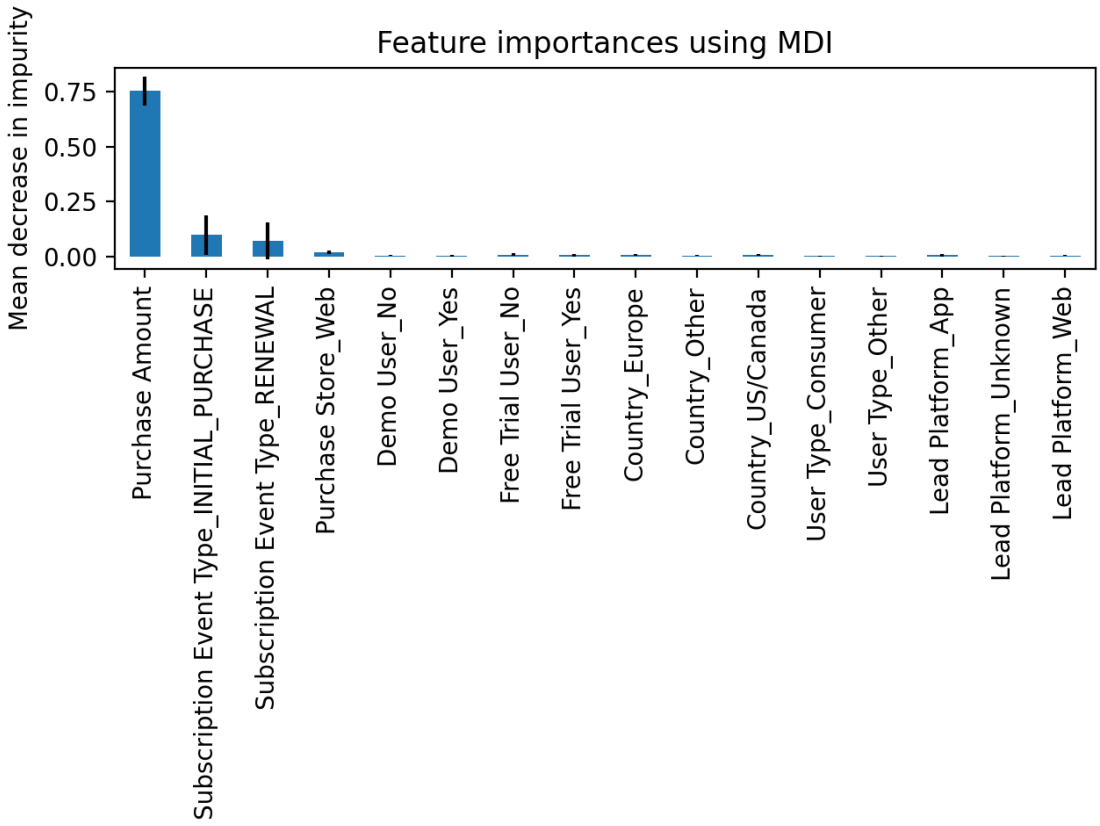
```
predictors = ["Purchase Amount", 'Subscription Event Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
        'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Unknown', 'Lead Platform_Web']

autoRenew = pd.get_dummies(autoRenew, columns= ['Subscription Event Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Country', 'User Type', 'Lead Platform'])

X = autoRenew[predictors]
y = autoRenew['Auto Renew']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=27)

rfc_100 = RandomForestClassifier(n_estimators=100, random_state=0)

rfc_100.fit(X_train, y_train)

# Predict on the test set results

y_pred_100 = rfc_100.predict(X_test)



# Check accuracy score

print("Train Accuracy:", end=" ")
print(accuracy_score(y_train, rfc_100.predict(X_train)))
ConfusionMatrixDisplay.from_predictions(y_train, rfc_100.predict(X_train))

print("Test Accuracy:", end=" ")
print(accuracy_score(y_test, rfc_100.predict(X_test)))
ConfusionMatrixDisplay.from_predictions(y_test, rfc_100.predict(X_test))


# feature_scores = pd.Series(rfc_100.feature_importances_, index=X_train.
 ↪columns).sort_values(ascending=False)

# feature_scores

std = np.std([tree.feature_importances_ for tree in rfc_100.estimators_],␣
 ↪axis=0)
```

```
forest_importances = pd.Series(rfc_100.feature_importances_, index=X_train.
 ↪columns)

fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```

```
Train Accuracy: 0.762488646684832
Test Accuracy: 0.7202543142597638
```




3015
494
1046
950
0
1
True label
0
1
Predicted label
3000
2500
2000
1500
1000
500



#[Auto Renew] Random Forrest Classifier With Merged Data We decided that Application Interaction data provided was an important metric to merge into our subscriber data. After doing so, we were left with data that showed us how much each user interacted with the app through various app activities. Once we merged this dataset, we fit another random forrest to this data, and checked its predicting power for classifying whether Auto Renew is on or not. We notice that with the merging of this data, we have improved our test accuracy by about 2% from the random forrest above. But we have a significant overfitting issue.

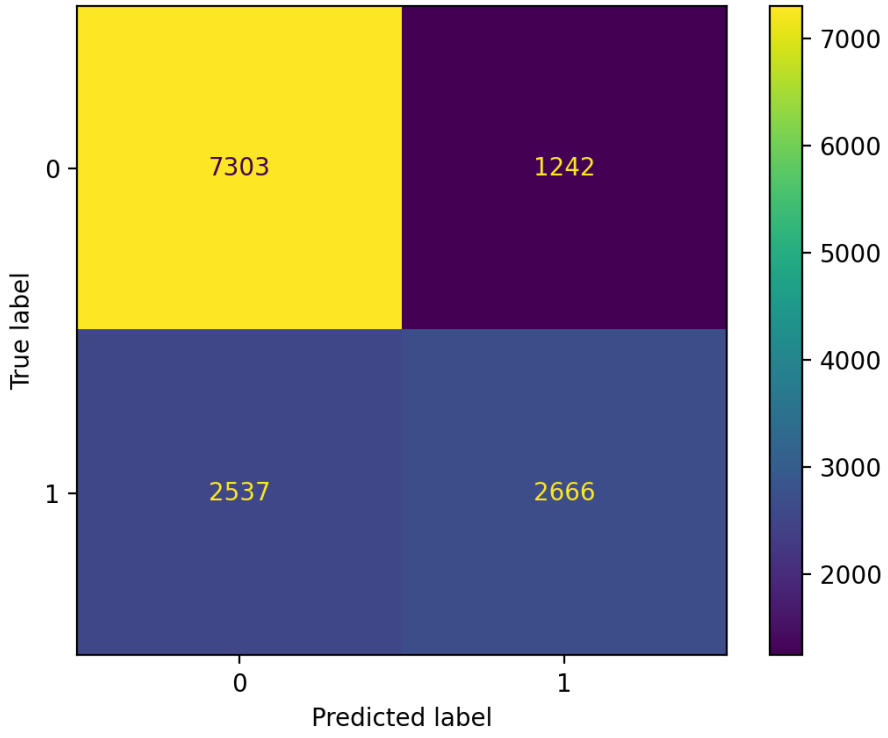
##Results We believe there is an insight to pull from this model. The fact that the forest weighs the new data introduced heavily, I think it is hard not to believe that there is a correlate of this data with Auto Renew. This is something the team will use other avenues to review closer.

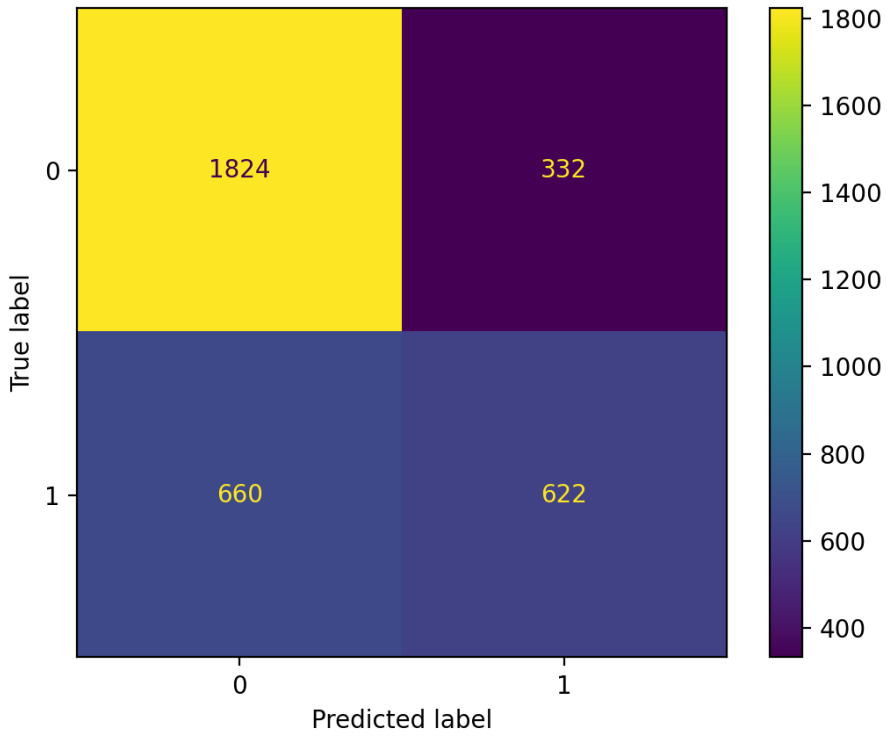
[28]:
```
from sklearn import tree
#Took out total app interactions.
mergedRF = merged.copy()
mergedRF = mergedRF.dropna(subset=['Send Count', 'Open Count', 'Click Count',␣
 ↪'Unique Open Count', 'Unique Click Count', 'Total App Interactions'])
#There is a single NULL for autorenew so dropped.
mergedRF = mergedRF.dropna(subset=['Auto Renew'])
mergedRF['Auto Renew'] = mergedRF['Auto Renew'].replace({'Off': 0, 'On': 1})
```

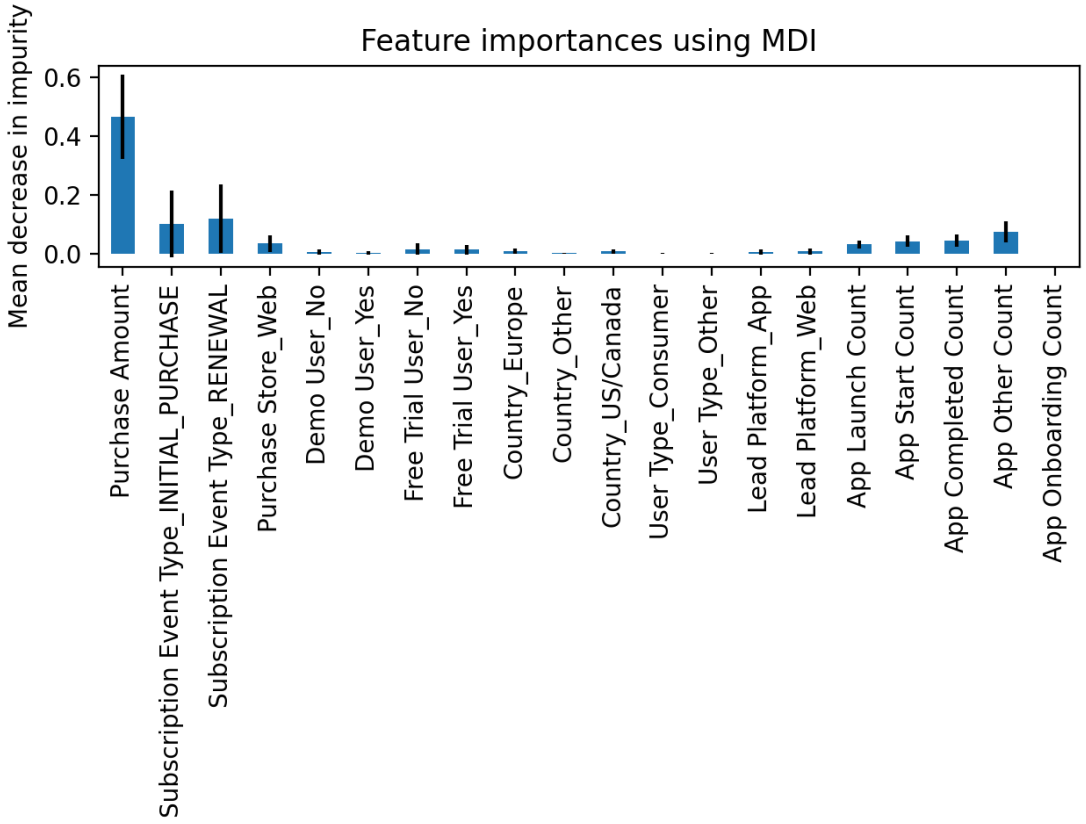
```
mergedRF['Email Subscriber'] = mergedRF['Email Subscriber'].replace({'No': 0,␣
 ↪'Yes': 1})
mergedRF['Push Notifications'] = mergedRF['Push Notifications'].replace({'No':␣
 ↪0, 'Yes': 1})

predictors = ["Purchase Amount", 'Subscription Event Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
        'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
       'Lead Platform_App', 'Lead Platform_Web', 'App Launch Count', 'App Start␣
 ↪Count', 'App Completed Count', 'App Other Count', 'App Onboarding Count']

mergedRF = pd.get_dummies(mergedRF, columns= ['Subscription Event Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Country', 'User Type', 'Lead Platform'])

X = mergedRF[predictors]
y = mergedRF['Auto Renew']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=27)

rfc_100 = RandomForestClassifier(n_estimators=100,
                                 max_depth = 7,
                                 random_state=0)

rfc_100.fit(X_train, y_train)

# Predict on the test set results

y_pred_100 = rfc_100.predict(X_test)



# Check accuracy score

print("Train Accuracy:", end=" ")
print(accuracy_score(y_train, rfc_100.predict(X_train)))
ConfusionMatrixDisplay.from_predictions(y_train, rfc_100.predict(X_train))

print("Test Accuracy:", end=" ")
print(accuracy_score(y_test, rfc_100.predict(X_test)))
ConfusionMatrixDisplay.from_predictions(y_test, rfc_100.predict(X_test))
```

```
# feature_scores = pd.Series(rfc_100.feature_importances_, index=X_train.
 ↪columns).sort_values(ascending=False)

# feature_scores

std = np.std([tree.feature_importances_ for tree in rfc_100.estimators_],␣
 ↪axis=0)
forest_importances = pd.Series(rfc_100.feature_importances_, index=X_train.
 ↪columns)

fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```

```
Train Accuracy: 0.7251236543497236
Test Accuracy: 0.7114601512507271
```




1824
332
660
622
0
1
True label
0
1
Predicted label
1800
1600
1400
1200
1000
800
600
400



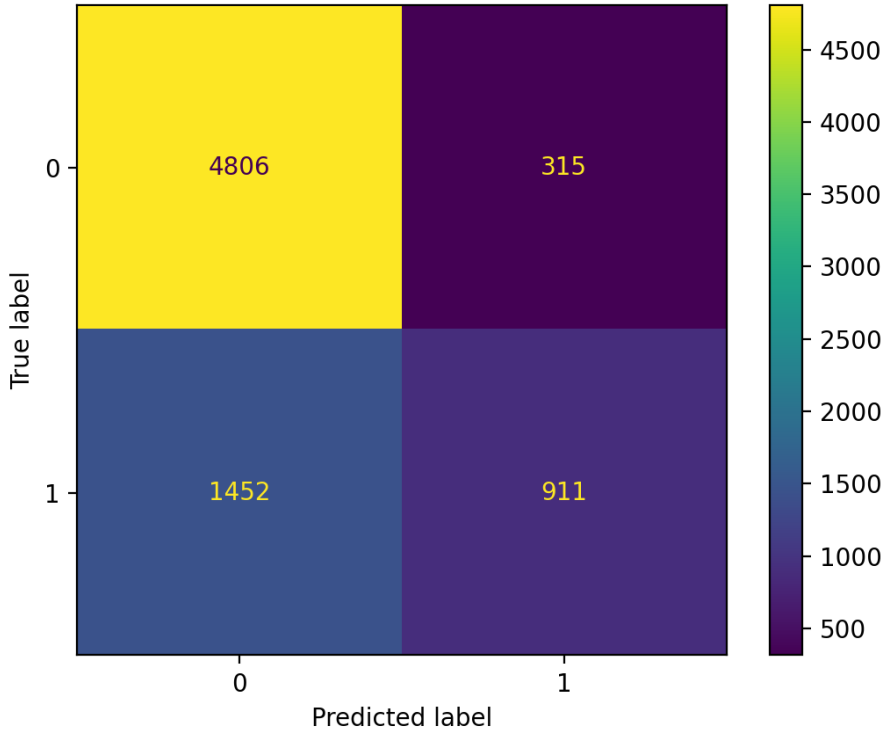
Ran a second time to ensure the App Averages do not affect the final outcome.

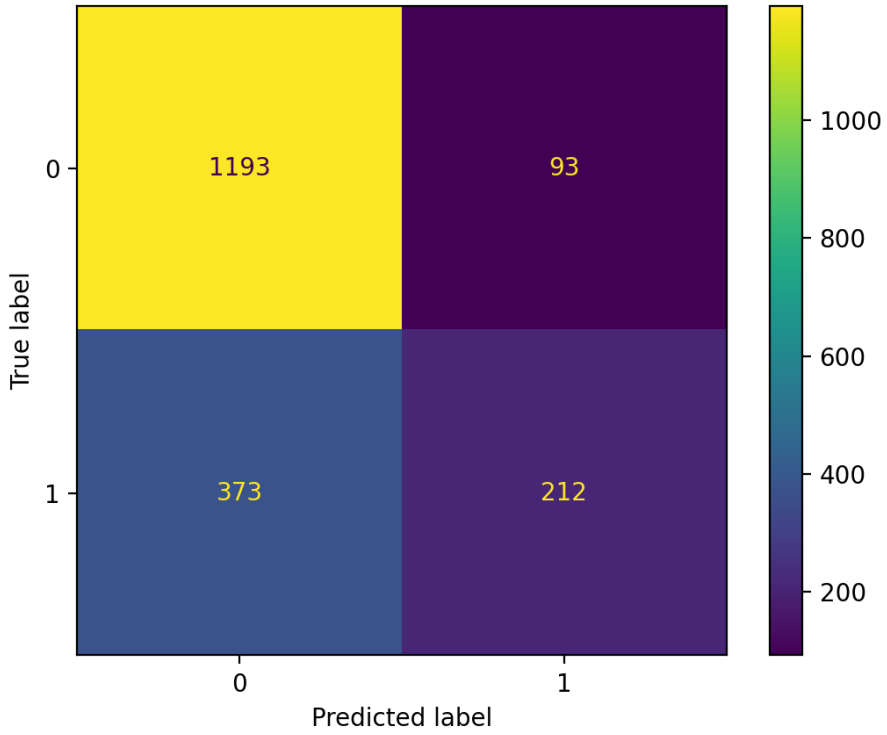
[29]:
```
from sklearn import tree
#Took out total app interactions.
mergedRF = merged.copy()
mergedRF = mergedRF.loc[mergedRF['Purchase Store'] == 'Web']
mergedRF = mergedRF.dropna(subset=['Send Count', 'Open Count', 'Click Count',␣
 ↪'Unique Open Count', 'Unique Click Count', 'Total App Interactions'])
#There is a single NULL for autorenew so dropped.
mergedRF = mergedRF.dropna(subset=['Auto Renew'])
mergedRF['Auto Renew'] = mergedRF['Auto Renew'].replace({'Off': 0, 'On': 1})
mergedRF['Email Subscriber'] = mergedRF['Email Subscriber'].replace({'No': 0,␣
 ↪'Yes': 1})
mergedRF['Push Notifications'] = mergedRF['Push Notifications'].replace({'No':␣
 ↪0, 'Yes': 1})

predictors = ["Purchase Amount", 'Subscription Event Type_INITIAL_PURCHASE',
       'Subscription Event Type_RENEWAL', 'Purchase Store_Web', 'Demo User_No',
       'Demo User_Yes', 'Free Trial User_No', 'Free Trial User_Yes',
        'Country_Europe', 'Country_Other',
       'Country_US/Canada', 'User Type_Consumer', 'User Type_Other',
```

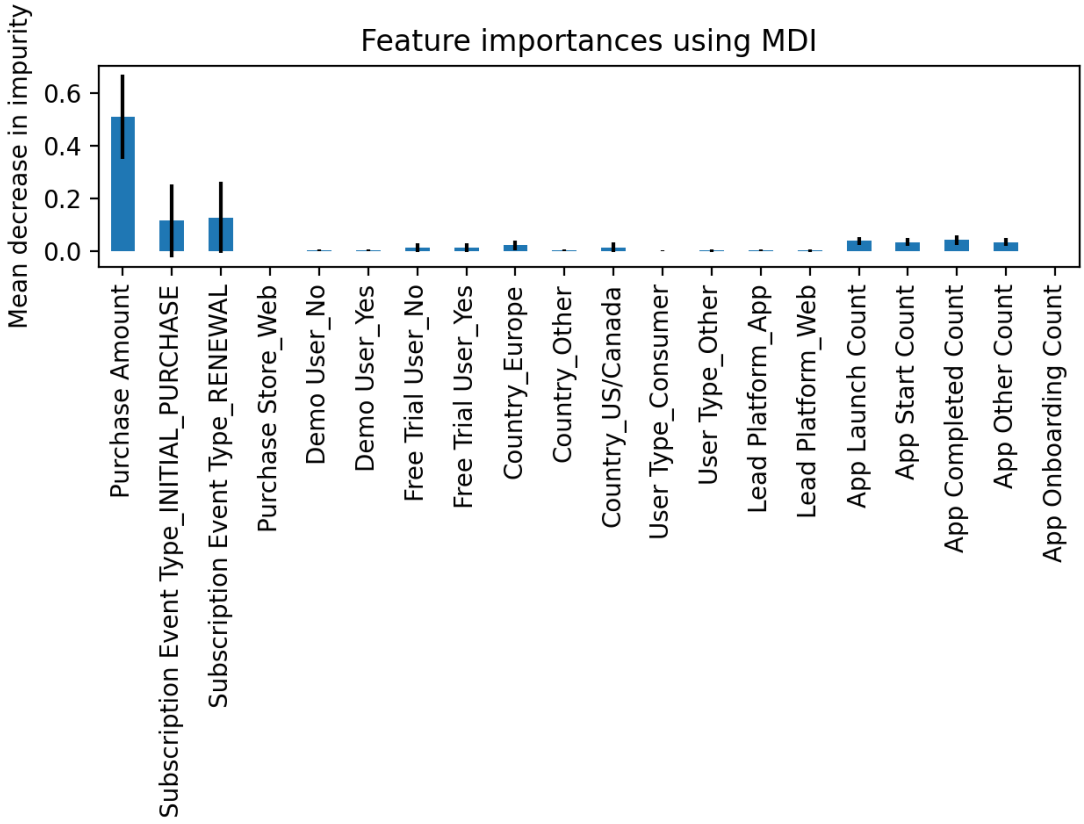
```
       'Lead Platform_App', 'Lead Platform_Web', 'App Launch Count', 'App Start␣
 ↪Count', 'App Completed Count', 'App Other Count', 'App Onboarding Count']

mergedRF = pd.get_dummies(mergedRF, columns= ['Subscription Event Type',
       'Purchase Store', 'Demo User',
       'Free Trial User', 'Country', 'User Type', 'Lead Platform'])

X = mergedRF[predictors]
y = mergedRF['Auto Renew']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=27)

rfc_100 = RandomForestClassifier(n_estimators=100,
                                 max_depth = 7,
                                 random_state=0)

rfc_100.fit(X_train, y_train)

# Predict on the test set results

y_pred_100 = rfc_100.predict(X_test)



# Check accuracy score

print("Train Accuracy:", end=" ")
print(accuracy_score(y_train, rfc_100.predict(X_train)))
ConfusionMatrixDisplay.from_predictions(y_train, rfc_100.predict(X_train))

print("Test Accuracy:", end=" ")
print(accuracy_score(y_test, rfc_100.predict(X_test)))
ConfusionMatrixDisplay.from_predictions(y_test, rfc_100.predict(X_test))


# feature_scores = pd.Series(rfc_100.feature_importances_, index=X_train.
 ↪columns).sort_values(ascending=False)

# feature_scores

std = np.std([tree.feature_importances_ for tree in rfc_100.estimators_],␣
 ↪axis=0)
forest_importances = pd.Series(rfc_100.feature_importances_, index=X_train.
 ↪columns)

fig, ax = plt.subplots()
```

```
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```

```
Train Accuracy: 0.7638963121325495
Test Accuracy: 0.7509353287012293
```




1193
93
373
212
0
1
True label
0
1
Predicted label
1000
800
600
400
200



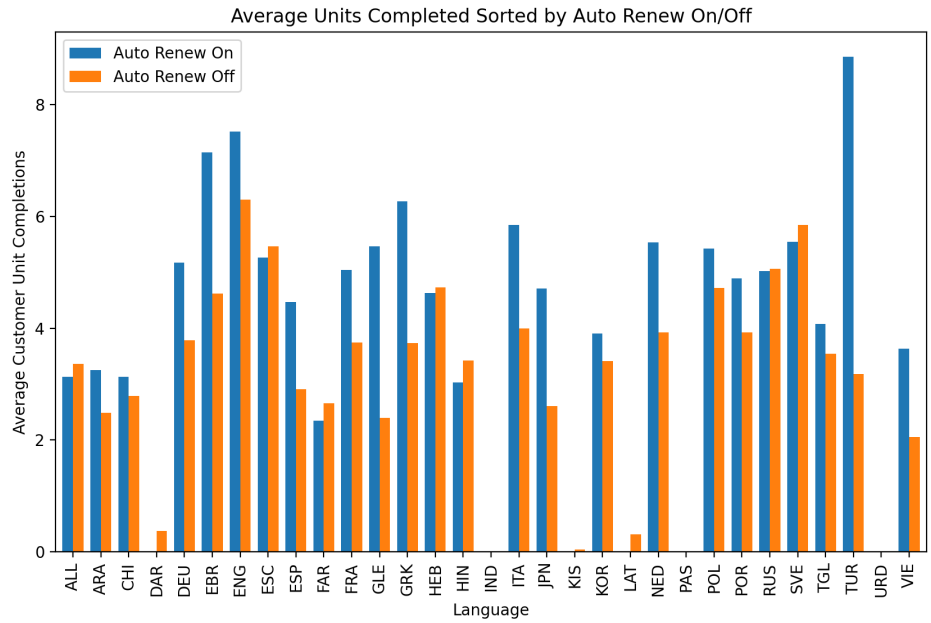
#Identifying trends in our non-auto renew customers We ran various visuals based on the classifiers above to plot variables the model weighed as important. From those metrics, we were able to pull the graphs below showing us significant behavioral differences between auto renew on and off customers.

[30]:
```
languageInteractionsAutoRenew = merged.copy()
interactionHypoth = languageInteractionsAutoRenew.
 ↪loc[(languageInteractionsAutoRenew['Subscription Type'] == 'Limited')]
interactionHypoth = interactionHypoth.groupby(['Language', 'Auto Renew']).mean()
interactionHypoth = interactionHypoth['App Completed Count']
interactionHypoth = interactionHypoth.reset_index()
interactionHypoth['Auto Renew On'] = 0
interactionHypoth['Auto Renew Off'] = 0
interactionHypoth = interactionHypoth.dropna()
for index,row in interactionHypoth.iterrows():
  auto = row['Auto Renew']
  lang = row['Language']
  interactions = row['App Completed Count']
  if auto == 'Off':
    interactionHypoth.loc[interactionHypoth['Language'] == lang, 'Auto Renew␣
 ↪Off'] = interactions
```

```
  elif auto == 'On':
    interactionHypoth.loc[(interactionHypoth['Language'] == lang) &␣
 ↪(interactionHypoth['Auto Renew'] == 'Off'), 'Auto Renew On'] = interactions

interactionHypoth.drop(interactionHypoth[interactionHypoth['Auto Renew'] ==␣
 ↪'On'].index, inplace=True)
interactionHypoth = interactionHypoth.set_index('Language')
interactionHypoth[['Auto Renew On', 'Auto Renew Off']].plot.bar(figsize=(10,6),␣
 ↪width=.75)
plt.ylabel('Average Customer Unit Completions')
plt.title('Average Units Completed Sorted by Auto Renew On/Off')
```

```
<ipython-input-30-56fd38ca2f77>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[30]: `Text(0.5, 1.0, 'Average Units Completed Sorted by Auto Renew On/Off')`

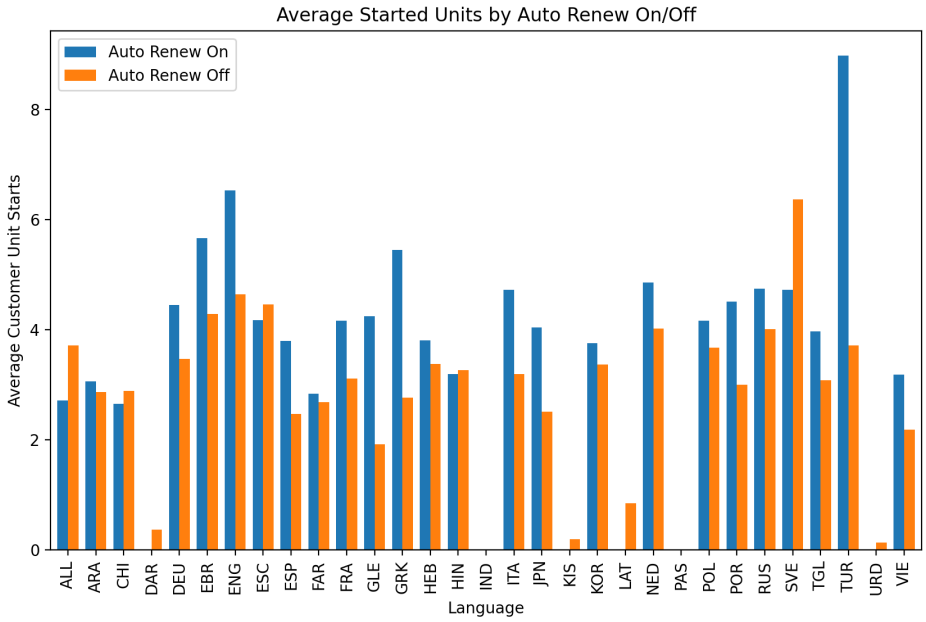


[31]:
```
languageInteractionsAutoRenew = merged.copy()
interactionHypoth = languageInteractionsAutoRenew.
 ↪loc[(languageInteractionsAutoRenew['Subscription Type'] == 'Limited')]
interactionHypoth = interactionHypoth.groupby(['Language', 'Auto Renew']).mean()
interactionHypoth = interactionHypoth['App Start Count']
interactionHypoth = interactionHypoth.reset_index()
```

```
interactionHypoth['Auto Renew On'] = 0
interactionHypoth['Auto Renew Off'] = 0
interactionHypoth = interactionHypoth.dropna()
for index,row in interactionHypoth.iterrows():
  auto = row['Auto Renew']
  lang = row['Language']
  interactions = row['App Start Count']
  if auto == 'Off':
    interactionHypoth.loc[interactionHypoth['Language'] == lang, 'Auto Renew␣
 ↪Off'] = interactions
  elif auto == 'On':
    interactionHypoth.loc[(interactionHypoth['Language'] == lang) &␣
 ↪(interactionHypoth['Auto Renew'] == 'Off'), 'Auto Renew On'] = interactions

interactionHypoth.drop(interactionHypoth[interactionHypoth['Auto Renew'] ==␣
 ↪'On'].index, inplace=True)
interactionHypoth = interactionHypoth.set_index('Language')
interactionHypoth[['Auto Renew On', 'Auto Renew Off']].plot.bar(figsize=(10,6),␣
 ↪width=.75)
plt.ylabel('Average Customer Unit Starts')
plt.title('Average Started Units by Auto Renew On/Off')
```

```
<ipython-input-31-db371014eae7>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

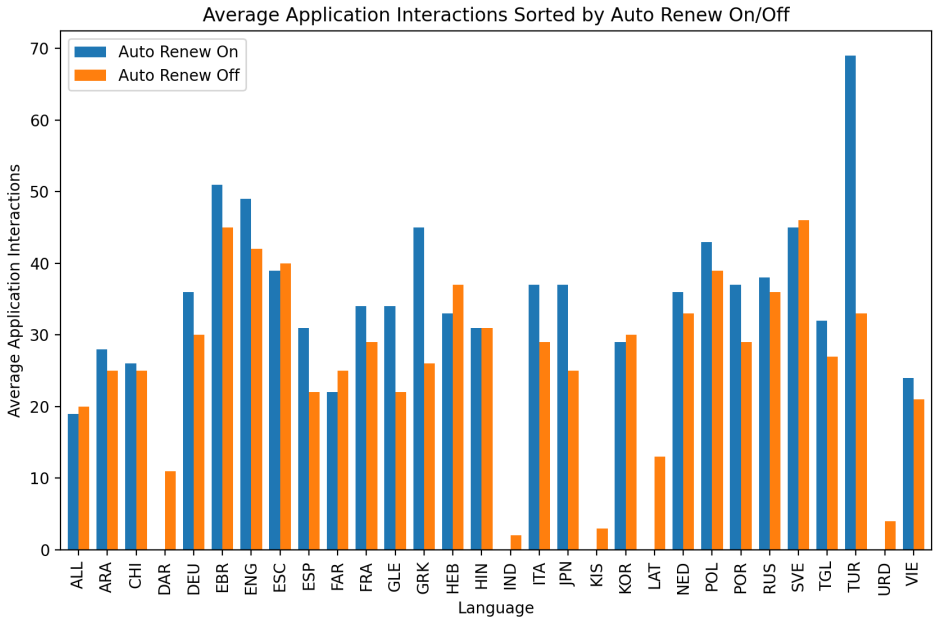
[31]: Text(0.5, 1.0, 'Average Started Units by Auto Renew On/Off')



[32]:
```
languageInteractionsAutoRenew = merged.copy()
interactionHypoth = languageInteractionsAutoRenew.
 ↪loc[(languageInteractionsAutoRenew['Subscription Type'] == 'Limited')]
interactionHypoth = interactionHypoth.groupby(['Language', 'Auto Renew']).mean()
interactionHypoth = interactionHypoth['Total App Interactions']
interactionHypoth = interactionHypoth.reset_index()
interactionHypoth['Auto Renew On'] = 0
interactionHypoth['Auto Renew Off'] = 0
interactionHypoth = interactionHypoth.dropna()
for index,row in interactionHypoth.iterrows():
  auto = row['Auto Renew']
  lang = row['Language']
  interactions = row['Total App Interactions']
  if auto == 'Off':
    interactionHypoth.loc[interactionHypoth['Language'] == lang, 'Auto Renew␣
 ↪Off'] = int(interactions)
  elif auto == 'On':
    interactionHypoth.loc[(interactionHypoth['Language'] == lang) &␣
 ↪(interactionHypoth['Auto Renew'] == 'Off'), 'Auto Renew On'] =␣
 ↪int(interactions)

interactionHypoth.drop(interactionHypoth[interactionHypoth['Auto Renew'] ==␣
 ↪'On'].index, inplace=True)
interactionHypoth = interactionHypoth.set_index('Language')
interactionHypoth[['Auto Renew On', 'Auto Renew Off']].plot.bar(figsize=(10,6),␣
 ↪width=.75)
plt.ylabel('Average Application Interactions')
plt.title('Average Application Interactions Sorted by Auto Renew On/Off')
```

```
<ipython-input-32-e56ffa13f7ca>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[32]:
```
Text(0.5, 1.0, 'Average Application Interactions Sorted by Auto Renew On/Off')
```



##Non-Renewers are likely not an email subscriber. Building a profile of those not continuing: Of the users with auto renew on We have almost a 1:1 relationship of email subscribers vs not But when they do not have auto renew on, there are about 56% of people who are not subscribed to emails. ## Our Marketing Of those people with auto renew off, they are recieving more than double the emails on average than someone with auto renew on. We clearly are not engaging enough with our emails to drive these users to turn on auto renew, or are they just looking for better deals than they have? ## Are renewers more motivated to learn? Those with auto renew off on average are paying $15 or $6 less than someone with auto renew on. ## Longer Subscribers tend to have auto renew on. Users that have auto renew off, tend to have an average subscription length of 7 months. Users have have auto renew on, tend to have a subscription length of 10 months.

[33]:
```
languageInteractionsAutoRenew = merged.copy()
interactionHypoth = languageInteractionsAutoRenew.
  ↪loc[(languageInteractionsAutoRenew['Subscription Type'] == 'Limited')]
interactionHypoth = interactionHypoth.groupby(['Auto Renew']).mean()
# interactionHypoth = interactionHypoth.groupby(['Auto Renew','Email␣
  ↪Subscriber']).count()
interactionHypoth['Months Subscribed']
```

```
<ipython-input-33-dea6c6a768e3>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[33]:
```
Auto Renew
Off    7
On    10
Name: Months Subscribed, dtype: float64
```

[34]:
```
languageInteractionsAutoRenew = merged.copy()
interactionHypoth = languageInteractionsAutoRenew.
 ↪loc[(languageInteractionsAutoRenew['Subscription Type'] == 'Limited') &␣
 ↪(languageInteractionsAutoRenew['Months Subscribed'] == 24)]
interactionHypoth = interactionHypoth.groupby(['Auto Renew']).mean()
# interactionHypoth = interactionHypoth.groupby(['Auto Renew','Email␣
 ↪Subscriber']).count()
interactionHypoth['Purchase Amount']
```

```
<ipython-input-34-1ee2ee2d51ee>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[34]:
```
Auto Renew
Off    57
On     60
Name: Purchase Amount, dtype: float64
```

Purchase amount average by country, without app purchases

[35]:
```
country = cleanedData.loc[(cleanedData['Subscription Type'] == 'Limited') &␣
 ↪(cleanedData['Purchase Store'] == 'Web')]
countryPurchaseMean = pd.DataFrame(country.groupby(['Country']).mean().
 ↪apply(lambda x: round(x, 2)))
countryPurchaseMean['Purchase Amount']
# country['Months Subscribed'].mean()
# trialGroup['Subscription Type'].count()
```

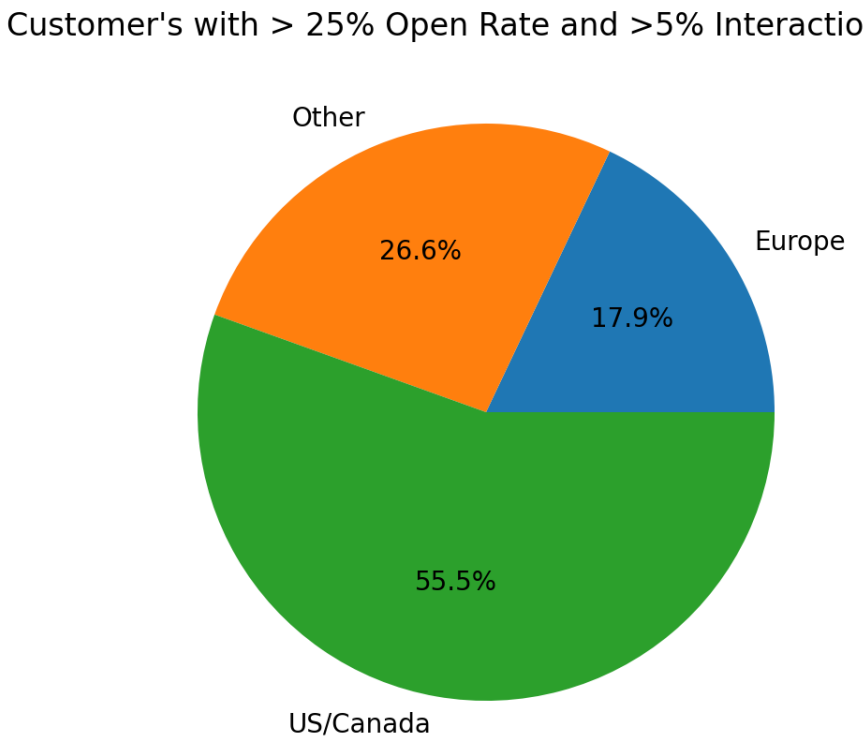
```
<ipython-input-35-ea2387f06b30>:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
```

[35]:
```
Country
Europe      47
Other       56
US/Canada   62
Name: Purchase Amount, dtype: float64
```

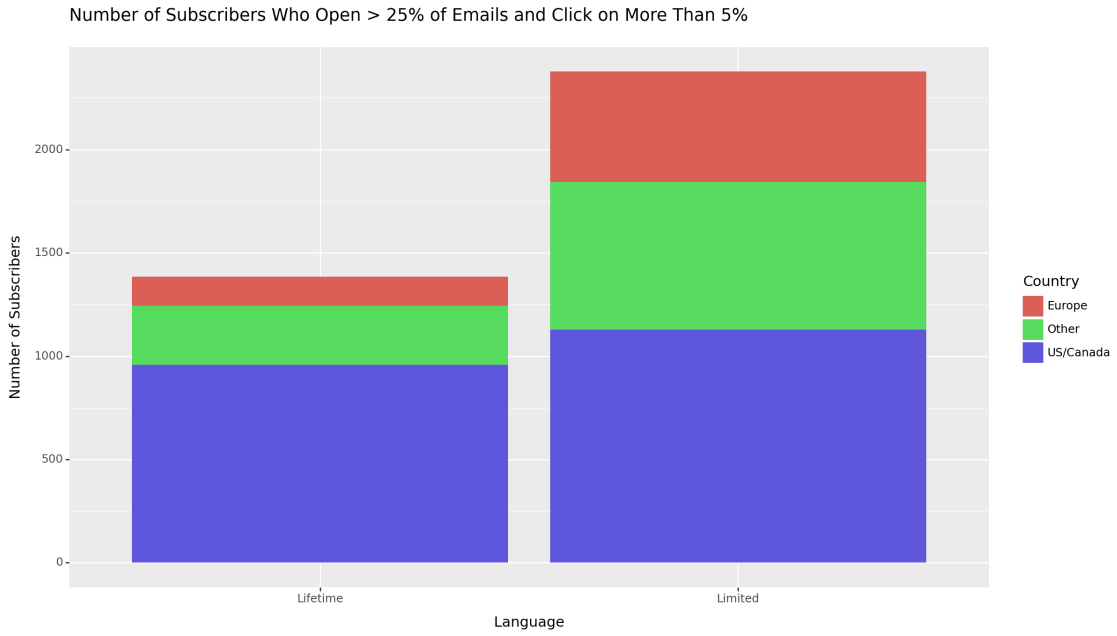
#Customers more likely to buy other things Aside from the customers who currently have All-Lifetime products, we have compiled the top languages who have > 25% open rate on their email notifications as well as a 5% interaction (click rate) within those opened emails. We believe this best

captures the most likely customers to buy additional products based on their interaction habits. If we were to add promotions, deals or packages in these emails, these customers have the highest likelyhood to open the emails, which in turn can convert to those customers buying additional products they are notified about.

[36]:
```
clicks = pd.read_csv('cleanDataV2Click.csv')
clickPerc = clicks.copy()
clickPerc = clickPerc.drop(clickPerc[clickPerc['Open Percentage'] == '#VALUE!'].
 ↪index)
clickPerc = clickPerc.drop(clickPerc[clickPerc['Click Percent'] == '#VALUE!'].
 ↪index)
clickPerc['Open Percentage'] = clickPerc['Open Percentage'].astype(float)
clickPerc['Click Percent'] = clickPerc['Click Percent'].astype(float)
clickPerc['Open Percentage'].mean()
over50 = clickPerc.loc[(clickPerc['Open Percentage'] >= .25) &␣
 ↪(clickPerc['Click Percent'] > .05)]
country = over50.groupby('Country').count()
country
plot = country['ID'].plot.pie(autopct='%.1f%%', figsize=(6, 5))
plot.set_title('Customer\'s with > 25% Open Rate and >5% Interaction Rate')
plot.set_ylabel(None)
(ggplot(over50, aes(x='Language', fill='Country')) + geom_bar() +␣
 ↪theme(figure_size = (12, 7)) + labs(title = 'Number of Subscribers Who Open␣
 ↪> 25% of Emails and Click on More Than 50%', x="Language",y='Number of␣
 ↪Subscribers'))
over50AllLifetime = over50.loc[(over50['Language'] == 'ALL') &␣
 ↪(over50['Subscription Type'] == 'Lifetime')]
```

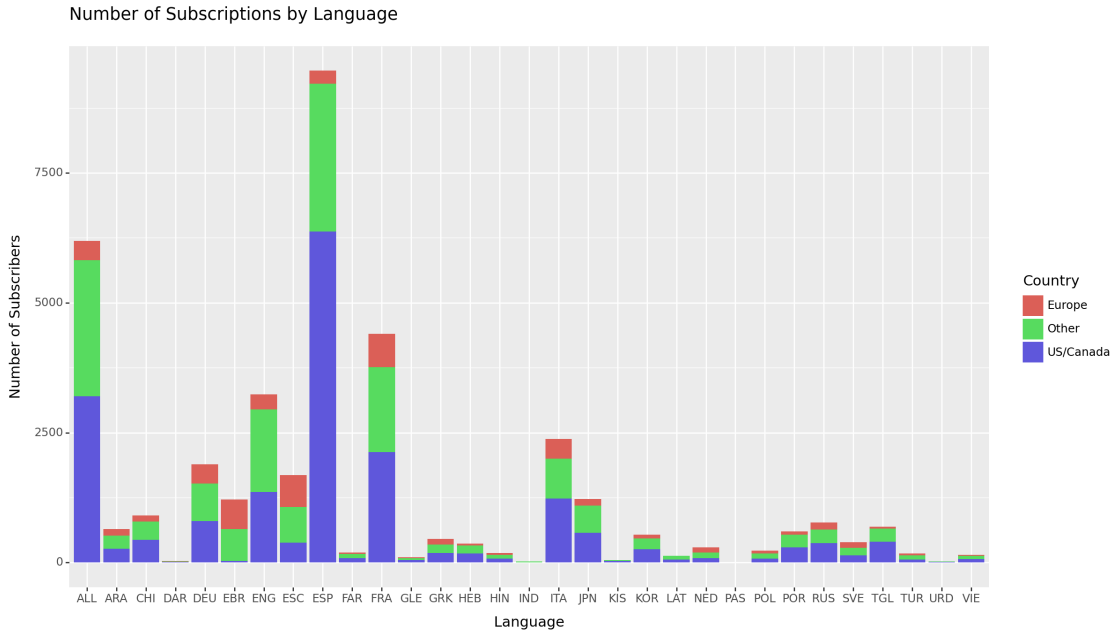


[37]:
```
(ggplot(over50, aes(x='Subscription Type', fill='Country')) + geom_bar() +␣
↪theme(figure_size = (12, 7)) + labs(title = 'Number of Subscribers Who Open␣
↪> 25% of Emails and Click on More Than 5%', x="Language",y='Number of␣
↪Subscribers'))
```



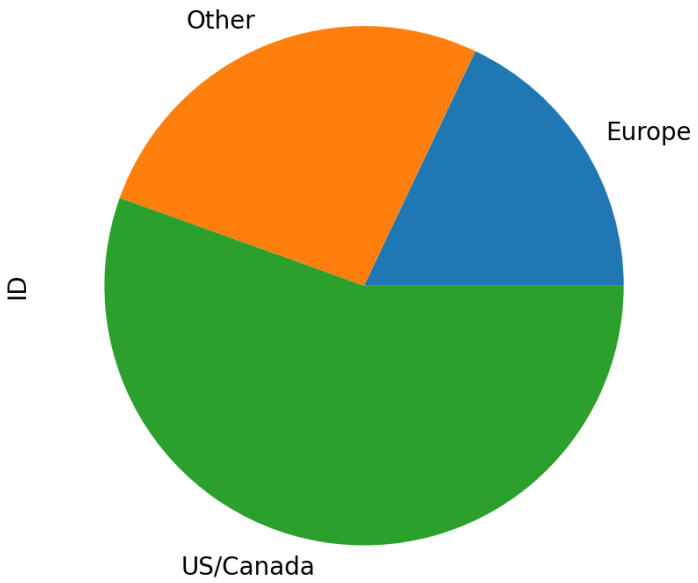
[37]: <Figure Size: (1200 x 700)>

[38]:
```
(ggplot(cleanedData, aes(x='Language', fill='Country')) + geom_bar() +␣
↪theme(figure_size = (12, 7)) + labs(title = 'Number of Subscriptions by␣
↪Language', x="Language",y='Number of Subscribers'))
```



[38]: <Figure Size: (1200 x 700)>

[39]:
```
over50Countries = over50.groupby(['Country']).count()
plot = over50Countries['ID'].plot.pie()
```



[40]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

#Clustering Models

[41]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
import pandas as pd
import scipy.cluster

from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering
```

```
import numpy as np
```

[42]:
```
data = pd.read_csv('appFilled.csv')
data.shape
```

[42]:
```
(38611, 25)
```

[43]:
```
counts_to_fill = ['Send Count', 'Open Count', 'Click Count', 'Unique Open␣
↪Count', 'Unique Click Count']
```

[44]:
```
data[counts_to_fill] = data[counts_to_fill].fillna(0)
```

[45]:
```
data.head()
```

[45]:

	ID	Language	Subscription Type	Subscription Event Type	Purchase Store	\
0	1	POR	Limited	INITIAL_PURCHASE	App	
1	2	EBR	Limited	INITIAL_PURCHASE	Web	
2	3	ESP	Limited	INITIAL_PURCHASE	Web	
3	4	KOR	Limited	INITIAL_PURCHASE	App	
4	5	ENG	Limited	INITIAL_PURCHASE	App	

	Purchase Amount	Currency	Subscription Start Date	Subscription Expiration	\
0	56	USD	12/28/18	6/28/19	
1	39	USD	11/28/19	2/28/20	
2	0	USD	12/31/18	12/31/19	
3	53	USD	11/7/19	2/7/20	
4	54	USD	8/13/19	11/13/19	

	Months Subscribed	…	Country	User Type	Lead Platform	Email Subscriber	\
0	6	…	US/Canada	Consumer	App	Yes	
1	3	…	Other	Consumer	Web	No	
2	12	…	US/Canada	Consumer	Web	Yes	
3	3	…	US/Canada	Consumer	App	Yes	
4	3	…	US/Canada	Consumer	Web	Yes	

	Push Notifications	Send Count	Open Count	Click Count	Unique Open Count	\
0	Yes	63	7	0	6	
1	Yes	4	3	0	1	
2	Yes	1	0	0	0	
3	Yes	14	0	0	0	
4	Yes	80	5	1	5	

	Unique Click Count
0	0
1	0
2	0
3	0

```
4                  1

[5 rows x 25 columns]
```

[46]:
```
data.dropna()
```

[46]:
```
           ID Language Subscription Type Subscription Event Type  \
8           9      DEU           Limited        INITIAL_PURCHASE
9          10      ESP           Limited                RENEWAL
11         12      ESP           Limited        INITIAL_PURCHASE
15         17      ALL          Lifetime        INITIAL_PURCHASE
29         32      ESP          Lifetime        INITIAL_PURCHASE
…         …       …                …                       …
38373   39760      FRA           Limited                RENEWAL
38490   39878      FRA           Limited                RENEWAL
38504   39892      ESC           Limited                RENEWAL
38563   39953      ESP           Limited        INITIAL_PURCHASE
38610   40000      ESP           Limited                RENEWAL

      Purchase Store  Purchase Amount Currency Subscription Start Date  \
8                Web                43      USD                   3/3/20
9                App                62      USD                  3/21/20
11               Web                36      USD                  10/17/19
15               Web               199      USD                   3/25/20
29               App               175      USD                   3/29/20
…              …                 …      …                       …
38373            App                52      USD                  12/30/18
38490            App                52      USD                   2/12/19
38504            App                44      USD                   12/2/18
38563            Web                41      USD                   6/22/19
38610            App                62      USD                   12/2/18

      Subscription Expiration  Months Subscribed  …    Country User Type  \
8                      6/8/20                  3  …     Europe  Consumer
9                      3/6/21                 11  …  US/Canada  Consumer
11                    1/21/20                  3  …  US/Canada  Consumer
15                     1/1/99               1000  …  US/Canada  Consumer
29                     1/1/99               1000  …  US/Canada  Consumer
…                        …                …  …        …       …
38373                  1/2/20                 12  …  US/Canada  Consumer
38490                 2/15/20                 12  …  US/Canada  Consumer
38504                 12/5/19                 12  …  US/Canada  Consumer
38563                 9/25/19                  3  …  US/Canada  Consumer
38610                 12/5/19                 12  …  US/Canada  Consumer

      Lead Platform Email Subscriber Push Notifications Send Count Open Count  \
8              Web                No                 Yes         52         11
```

```
9               App               No               Yes          1          0
11              Web              Yes               Yes         94          0
15              Web              Yes               Yes          8          6
29              Web              Yes               Yes          8          2
…               …                …                 …            …          …
38373           App              Yes               Yes         16          0
38490           App              Yes               Yes         89         63
38504           App              Yes               Yes         31          0
38563           Web              Yes               Yes         17          0
38610           App              Yes               Yes          0          0

      Click Count Unique Open Count Unique Click Count
8                0                 5                 0
9                0                 0                 0
11               0                 0                 0
15               3                 2                 1
29               0                 1                 0
…            …                 …                 …
38373            0                 0                 0
38490            0                56                 0
38504            0                 0                 0
38563            0                 0                 0
38610            0                 0                 0

[5407 rows x 25 columns]
```

[47]:
```
data.shape
data.dtypes
```

[47]:
```
ID                          int64
Language                   object
Subscription Type          object
Subscription Event Type    object
Purchase Store             object
Purchase Amount           float64
Currency                   object
Subscription Start Date    object
Subscription Expiration    object
Months Subscribed           int64
Demo User                  object
Free Trial User            object
Free Trial Start Date      object
Free Trial Expiration      object
Auto Renew                 object
Country                    object
User Type                  object
Lead Platform              object
```

```
Email Subscriber            object
Push Notifications          object
Send Count                 float64
Open Count                 float64
Click Count                float64
Unique Open Count          float64
Unique Click Count         float64
dtype: object
```

[48]:
```
data.drop(columns=['Subscription Start Date', 'Subscription Expiration', 'Free␣
↪Trial Start Date', 'Free Trial Expiration'], inplace=True)
data.head()
```

[48]:

	ID	Language	Subscription Type	Subscription Event Type	Purchase Store	\
0	1	POR	Limited	INITIAL_PURCHASE	App	
1	2	EBR	Limited	INITIAL_PURCHASE	Web	
2	3	ESP	Limited	INITIAL_PURCHASE	Web	
3	4	KOR	Limited	INITIAL_PURCHASE	App	
4	5	ENG	Limited	INITIAL_PURCHASE	App	

	Purchase Amount	Currency	Months Subscribed	Demo User	Free Trial User	…	\
0	56	USD	6	Yes	No	…	
1	39	USD	3	No	No	…	
2	0	USD	12	No	No	…	
3	53	USD	3	Yes	No	…	
4	54	USD	3	No	No	…	

	Country	User Type	Lead Platform	Email Subscriber	Push Notifications	\
0	US/Canada	Consumer	App	Yes	Yes	
1	Other	Consumer	Web	No	Yes	
2	US/Canada	Consumer	Web	Yes	Yes	
3	US/Canada	Consumer	App	Yes	Yes	
4	US/Canada	Consumer	Web	Yes	Yes	

	Send Count	Open Count	Click Count	Unique Open Count	Unique Click Count
0	63	7	0	6	0
1	4	3	0	1	0
2	1	0	0	0	0
3	14	0	0	0	0
4	80	5	1	5	1

[5 rows x 21 columns]

[49]:
```
data.to_csv('clean_no_clusters.csv', index=False)
```

[50]:
```
categorical_cols = data.select_dtypes(include=['object', 'category']).columns
numerical_cols = data.select_dtypes(include=['int64', 'float64']).columns
```

[51]:
```
categorical_cols
```

[51]:
```
Index(['Language', 'Subscription Type', 'Subscription Event Type',
       'Purchase Store', 'Currency', 'Demo User', 'Free Trial User',
       'Auto Renew', 'Country', 'User Type', 'Lead Platform',
       'Email Subscriber', 'Push Notifications'],
      dtype='object')
```

[52]:
```
numerical_cols = numerical_cols.drop('ID')
```

[53]:
```
numerical_cols
```

[53]:
```
Index(['Purchase Amount', 'Months Subscribed', 'Send Count', 'Open Count',
       'Click Count', 'Unique Open Count', 'Unique Click Count'],
      dtype='object')
```

[54]:
```
df_encoded = pd.get_dummies(data, columns=categorical_cols)
df_encoded = df_encoded.drop('ID', axis=1)
df_encoded
```

[54]:

	Purchase Amount	Months Subscribed	Send Count	Open Count	\
0	56	6	63	7	
1	39	3	4	3	
2	0	12	1	0	
3	53	3	14	0	
4	54	3	80	5	
…	…	…	…	…	
38606	49	9	1	0	
38607	52	5	0	0	
38608	12	3	0	0	
38609	52	3	2	0	
38610	62	12	0	0	

	Click Count	Unique Open Count	Unique Click Count	Language_ALL	\
0	0	6	0	0	
1	0	1	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	1	5	1	0	
…	…	…	…	…	
38606	0	0	0	0	
38607	0	0	0	0	
38608	0	0	0	0	
38609	0	0	0	0	
38610	0	0	0	0	

	Language_ARA	Language_CHI	…	Country_US/Canada	User Type_Consumer	\

```
0                0             0  …                  1                   1
1                0             0  …                  0                   1
2                0             0  …                  1                   1
3                0             0  …                  1                   1
4                0             0  …                  1                   1
…              …             …  …                  …                   …
38606            0             0  …                  0                   0
38607            0             0  …                  0                   0
38608            0             0  …                  0                   1
38609            0             0  …                  1                   1
38610            0             0  …                  1                   1

       User Type_Other  Lead Platform_App  Lead Platform_Unknown  \
0                    0                  1                      0
1                    0                  0                      0
2                    0                  0                      0
3                    0                  1                      0
4                    0                  0                      0
…                  …                  …                      …
38606                1                  0                      1
38607                1                  0                      1
38608                0                  0                      0
38609                0                  0                      0
38610                0                  1                      0

       Lead Platform_Web  Email Subscriber_No  Email Subscriber_Yes  \
0                      0                    0                     1
1                      1                    1                     0
2                      1                    0                     1
3                      0                    0                     1
4                      1                    0                     1
…                    …                    …                     …
38606                  0                    1                     0
38607                  0                    1                     0
38608                  1                    1                     0
38609                  1                    0                     1
38610                  0                    0                     1

       Push Notifications_No  Push Notifications_Yes
0                          0                       1
1                          0                       1
2                          0                       1
3                          0                       1
4                          0                       1
…                        …                       …
38606                      1                       0
38607                      1                       0
```

```
38608                     0                     1
38609                     0                     1
38610                     0                     1

[38611 rows x 63 columns]
```

[55]:
```
df_numeric = data[numerical_cols]
```

[56]:
```
df_combined = pd.concat([df_numeric, df_encoded], axis=1)
df_combined
```

[56]:
```
       Purchase Amount  Months Subscribed  Send Count  Open Count  \
0                   56                  6          63           7
1                   39                  3           4           3
2                    0                 12           1           0
3                   53                  3          14           0
4                   54                  3          80           5
…                  …                   …           …           …
38606               49                  9           1           0
38607               52                  5           0           0
38608               12                  3           0           0
38609               52                  3           2           0
38610               62                 12           0           0

       Click Count  Unique Open Count  Unique Click Count  Purchase Amount  \
0                0                  6                   0               56
1                0                  1                   0               39
2                0                  0                   0                0
3                0                  0                   0               53
4                1                  5                   1               54
…              …                  …                   …              …
38606            0                  0                   0               49
38607            0                  0                   0               52
38608            0                  0                   0               12
38609            0                  0                   0               52
38610            0                  0                   0               62

       Months Subscribed  Send Count  …  Country_US/Canada  \
0                      6          63  …                  1
1                      3           4  …                  0
2                     12           1  …                  1
3                      3          14  …                  1
4                      3          80  …                  1
…                    …          …  …                 …
38606                  9           1  …                  0
38607                  5           0  …                  0
38608                  3           0  …                  0
```

```
38609                   3           2  …                   1
38610                  12           0  …                   1

       User Type_Consumer  User Type_Other  Lead Platform_App  \
0                       1                0                  1
1                       1                0                  0
2                       1                0                  0
3                       1                0                  1
4                       1                0                  0
…                     …                …                  …
38606                   0                1                  0
38607                   0                1                  0
38608                   1                0                  0
38609                   1                0                  0
38610                   1                0                  1

       Lead Platform_Unknown  Lead Platform_Web  Email Subscriber_No  \
0                          0                  0                    0
1                          0                  1                    1
2                          0                  1                    0
3                          0                  0                    0
4                          0                  1                    0
…                        …                  …                  …
38606                      1                  0                    1
38607                      1                  0                    1
38608                      0                  1                    1
38609                      0                  1                    0
38610                      0                  0                    0

       Email Subscriber_Yes  Push Notifications_No  Push Notifications_Yes
0                         1                      0                       1
1                         0                      0                       1
2                         1                      0                       1
3                         1                      0                       1
4                         1                      0                       1
…                       …                      …                     …
38606                     0                      1                       0
38607                     0                      1                       0
38608                     0                      0                       1
38609                     1                      0                       1
38610                     1                      0                       1

[38611 rows x 70 columns]
```

[57]:
```
# List of columns to add
columns_to_add = ['Send Count', 'Open Count', 'Click Count', 'Unique Open␣
 ↪Count', 'Unique Click Count']
```

```
# Adding a new column with the sum of the specified columns
df_combined['Total Count'] = df_combined[columns_to_add].sum(axis=1)

# Fill missing values in 'Total Count' column with 0
df_combined['Total Count'] = df_combined['Total Count'].fillna(0)

# Interpolate missing values in 'Total Count' column
df_combined['Total Count'] = df_combined['Total Count'].interpolate()

df_combined
```

[57]:

	Purchase Amount	Months Subscribed	Send Count	Open Count	\
0	56	6	63	7	
1	39	3	4	3	
2	0	12	1	0	
3	53	3	14	0	
4	54	3	80	5	
…	…	…	…	…	
38606	49	9	1	0	
38607	52	5	0	0	
38608	12	3	0	0	
38609	52	3	2	0	
38610	62	12	0	0	

	Click Count	Unique Open Count	Unique Click Count	Purchase Amount	\
0	0	6	0	56	
1	0	1	0	39	
2	0	0	0	0	
3	0	0	0	53	
4	1	5	1	54	
…	…	…	…	…	
38606	0	0	0	49	
38607	0	0	0	52	
38608	0	0	0	12	
38609	0	0	0	52	
38610	0	0	0	62	

	Months Subscribed	Send Count	…	User Type_Consumer	\
0	6	63	…	1	
1	3	4	…	1	
2	12	1	…	1	
3	3	14	…	1	
4	3	80	…	1	
…	…	…	…	…	
38606	9	1	…	0	
38607	5	0	…	0	

```
38608                  3           0  …                   1
38609                  3           2  …                   1
38610                 12           0  …                   1

       User Type_Other  Lead Platform_App  Lead Platform_Unknown  \
0                    0                  1                      0
1                    0                  0                      0
2                    0                  0                      0
3                    0                  1                      0
4                    0                  0                      0
…                  …                  …                      …
38606                1                  0                      1
38607                1                  0                      1
38608                0                  0                      0
38609                0                  0                      0
38610                0                  1                      0

       Lead Platform_Web  Email Subscriber_No  Email Subscriber_Yes  \
0                      0                    0                     1
1                      1                    1                     0
2                      1                    0                     1
3                      0                    0                     1
4                      1                    0                     1
…                    …                    …                     …
38606                  0                    1                     0
38607                  0                    1                     0
38608                  1                    1                     0
38609                  1                    0                     1
38610                  0                    0                     1

       Push Notifications_No  Push Notifications_Yes  Total Count
0                          0                       1          152
1                          0                       1           16
2                          0                       1            2
3                          0                       1           28
4                          0                       1          184
…                        …                       …          …
38606                      1                       0            2
38607                      1                       0            0
38608                      0                       1            0
38609                      0                       1            4
38610                      0                       1            0

[38611 rows x 71 columns]
```

[58]:
```
# # df_combined = df_combined.drop_duplicates().reset_index(drop=True)
```

```
# df_combined = df_combined[df_combined['Click Count'] <= 1500].
 ↪reset_index(drop=True)
```

[ ]:
```
linkage_matrix = linkage(df_combined, method='ward')  # Adjust the method as␣
 ↪needed
```

[ ]:
```
plt.figure(figsize=(12, 6))
dendrogram(linkage_matrix, labels=df_combined.index, orientation='top',␣
 ↪distance_sort='descending')
plt.title('Dendrogram')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()
```

[ ]:
```
# Create an instance of the AgglomerativeClustering model with the desired␣
 ↪number of clusters (n_clusters)
n_clusters = 2  # Adjust this according to your requirements
hac = AgglomerativeClustering(n_clusters=n_clusters)

# Fit the model to your data
cluster_labels = hac.fit_predict(df_combined)
```

[ ]:
```
data['Cluster'] = cluster_labels
data.to_csv('clean_w_clusters.csv', index=False)
```

[ ]:
```
# Step 1: Check the lengths
print(len(df_combined['Total Count']))
print(len(df_combined['Purchase Amount']))

# Step 2: Check for missing values
print(df_combined['Total Count'].isnull().sum())
print(df_combined['Purchase Amount'].isnull().sum())

# Step 3: Handle missing values (example: drop rows with missing values)
df_combined = df_combined.dropna(subset=['Total Count', 'Purchase Amount'])
```

[ ]:
```
unique_clusters, cluster_counts = np.unique(cluster_labels, return_counts=True)


for cluster, count in zip(unique_clusters, cluster_counts):
    cluster_indices = np.where(cluster_labels == cluster)[0]
    cluster_data = df_combined.iloc[cluster_indices].dropna(subset=['Total␣
 ↪Count', 'Purchase Amount'])

    # Check if the cluster_data is not empty
    if not cluster_data.empty:
```

```
        plt.scatter(cluster_data['Click Count'], cluster_data['Purchase␣
 ↪Amount'], label=f'Cluster {cluster}')

plt.xlabel('Click  Count')
plt.ylabel('Purchase Amount')
plt.title('Clustered Data')
plt.legend()
plt.show()
```

[ ]:
```
df_combined.to_csv('data_with_cluster_labels.csv', index=False)
```

[ ]:
```
from sklearn.decomposition import PCA

# Perform PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(df_combined)

# Create a DataFrame for the PCA result
df_pca = pd.DataFrame(data = pca_result, columns = ['PC1', 'PC2'])

# Add the cluster labels to df_pca
df_pca['Cluster'] = cluster_labels

# Plot the clusters
plt.figure(figsize=(10,10))
for cluster in df_pca['Cluster'].unique():
    plt.scatter(df_pca[df_pca['Cluster'] == cluster]['PC1'],␣
 ↪df_pca[df_pca['Cluster'] == cluster]['PC2'], label=f'Cluster {cluster}')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Clusters')
plt.legend()
plt.show()
```

[ ]:
```
# doesn't show this cells output when downloading PDF
!pip install gwpy &> /dev/null

# installing necessary files
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!sudo apt-get update
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic

# installing pypandoc
!pip install pypandoc

# connecting your google drive
```

```
from google.colab import drive
drive.mount('/content/drive')

# copying your file over. Change "Class6-Completed.ipynb" to whatever your file␣
  ↪is called (see top of notebook)
!cp "drive/My Drive/Colab Notebooks/MGSCFinalProject.ipynb" ./

# Again, replace "Class6-Completed.ipynb" to whatever your file is called (see␣
  ↪top of notebook)
!jupyter nbconvert --to PDF "MGSCFinalProject.ipynb"
```