In [4]:
```
import pandas as pd

# Load the dataset
file_path = 'africa_mobilemoney_financial_inclusion.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset to understand its structure
data.head()
```

Out[4]:

	start_time	end_time	hhid	account_num	account_type	weight	district	urban
0	Oct 28, 2019 9:05:08 AM	Oct 28, 2019 10:38:45 AM	1001	1	Mobile Money	145.94444	District_A	Urban
1	Oct 28, 2019 10:42:17 AM	Oct 28, 2019 11:43:25 AM	1001	2	Bank Account	145.94444	District_A	Urban
2	Oct 28, 2019 11:47:47 AM	Oct 28, 2019 12:53:31 PM	1001	3	VSLA Account	145.94444	District_A	Urban
3	Oct 28, 2019 1:02:33 PM	Oct 28, 2019 1:56:43 PM	1002	1	SACCO Account	122.90667	District_B	Rural
4	Oct 28, 2019 2:01:04 PM	Oct 28, 2019 2:58:34 PM	1002	2	VSLA Account	122.90667	District_B	Rural

5 rows × 29 columns

In [5]:
```
data.shape
```

Out[5]:
```
(2442, 29)
```

Task 1:

During the survey, participants listed all the different types of financial accounts that they have registered. The resulting data has a format where there is one observation per account type. Format the data so that there is now one observation per participant.

In [6]:
```
# check for NA values in dataset, understanding the missing entries
print(data.isnull().sum())
print(data.isnull().values.any())
print(data.isnull().values.sum())
```

```
start_time                    0
end_time                      0
hhid                          0
account_num                   0
account_type                  0
weight                        0
district                      0
urban                         0
gender                        0
age                           0
hh_members                    0
highest_grade_completed     207
mm_account_cancelled          0
prefer_cash                  47
mm_trust                    128
mm_account_telco            478
mm_account_telco_main      1431
v234                        594
agent_trust                 808
v236                       1799
v237                        296
v238                        287
v240                        287
v241                        336
v242                        341
v243                        296
v244                       1867
v245                        296
v246                        287
dtype: int64
True
9785
```

In [7]:

```python
# Aggregating the types of accounts for each participant
# We will group by 'hhid' and join the different account types together

# Creating a DataFrame to hold the aggregated data
aggregated_data = data.groupby('hhid').agg({
    'account_type': lambda x: ', '.join(x),
    'weight': 'first',
    'district': 'first',
    'urban': 'first',
    'gender': 'first',
    'age': 'first',
    'hh_members': 'first',
    'highest_grade_completed': 'first',
    'mm_account_cancelled': 'first',
    'prefer_cash': 'first',
    'mm_trust': 'first',
    'mm_account_telco': 'first',
    'mm_account_telco_main': 'first',
    'v234': 'first',
    'agent_trust': 'first',
    'v236': 'first',
    'v237': 'first',
    'v238': 'first',
    'v240': 'first',
    'v241': 'first',
    'v242': 'first',
    'v243': 'first',
```

```
    'v244': 'first',
    'v245': 'first',
    'v246': 'first'
}).reset_index()

# Display the first few rows of the aggregated dataset, also NaNs are bei
aggregated_data.head()
```

Out[7]:

	hhid	account_type	weight	district	urban	gender	age	hh_members	highest
0	1001	Mobile Money, Bank Account, VSLA Account	145.94444	District_A	Urban	male	32	1	
1	1002	SACCO Account, VSLA Account	122.90667	District_B	Rural	male	32	4	
2	1003	Mobile Money, Bank Account	760.46191	District_A	Urban	male	30	8	
3	1004	Mobile Money, Bank Account, SACCO Account	433.96402	District_A	Rural	male	68	4	
4	1005	Mobile Money, VSLA Account	303.04395	District_C	Rural	female	28	2	

5 rows × 26 columns

In [8]:
```
aggregated_data.shape
```

Out[8]:
```
(1205, 26)
```

In [56]:
```
aggregated_data.tail()
```

Out[56]:

	hhid	account_type	weight	district	urban	gender	age	hh_members	higl
1200	2201	Mobile Money, SACCO Account, VSLA Account	406.14180	District_B	Rural	male	38	7	
1201	2202	Mobile Money, VSLA Account	172.01519	District_C	Rural	male	35	4	
1202	2203	None	103.26880	District_C	Rural	female	18	7	
1203	2204	VSLA Account	495.77039	District_C	Rural	female	58	6	
1204	2205	Mobile Money, Bank Account, SACCO Account, VSL...	666.83021	District_A	Rural	female	23	7	

5 rows × 28 columns

Task 2:

Create two dummy variables for whether each participant is: i) financially excluded and ii) digitally financially included. What are the overall rates of financial exclusion and digital financial inclusion for the combined population of these three districts? Report these in the key findings document, as well as any associations with other variables. (10 marks) a. Financial exclusion is defined as not having registered for any type of financial account including accounts with: Mobile money operators, banks, micro-finance institutions (MFI's), savings and credit cooperative organizations (SACCO's) and village savings and loan associations (VSLA's). b. Digital financial inclusion is defined as having at least one registe

In [9]:
```
# Creating dummy variables for financial exclusion and digital financial

# i) Financial exclusion: A participant is financially excluded if they h
aggregated_data['financially_excluded'] = aggregated_data['account_type']

# ii) Digitally financially included: A participant is digitally financia
aggregated_data['digitally_financially_included'] = aggregated_data['acco
```

In [10]:
```
aggregated_data.head()
```

Out[10]:

	hhid	account_type	weight	district	urban	gender	age	hh_members	highest
0	1001	Mobile Money, Bank Account, VSLA Account	145.94444	District_A	Urban	male	32	1	
1	1002	SACCO Account, VSLA Account	122.90667	District_B	Rural	male	32	4	
2	1003	Mobile Money, Bank Account	760.46191	District_A	Urban	male	30	8	
3	1004	Mobile Money, Bank Account, SACCO Account	433.96402	District_A	Rural	male	68	4	
4	1005	Mobile Money, VSLA Account	303.04395	District_C	Rural	female	28	2	

5 rows × 28 columns

In [11]:
```
# Calculate the overall rates of financial exclusion and digital financia
financial_exclusion_rate = aggregated_data['financially_excluded'].mean()
digital_financial_inclusion_rate = aggregated_data['digitally_financially
```

In [12]:
```
financial_exclusion_rate
```

Out[12]:
```
0.10954356846473029
```

In [13]:
```
digital_financial_inclusion_rate
```

Out[13]:
```
0.6863070539419087
```

In [14]:
```
# Converting the rates of financial exclusion and digital financial inclu
# and rounding to 2 decimal places
```

```
financial_exclusion_rate_percentage = round(financial_exclusion_rate * 10
digital_financial_inclusion_rate_percentage = round(digital_financial_inc

financial_exclusion_rate_percentage, digital_financial_inclusion_rate_per
```

Out[14]: (10.95, 68.63)

In [61]:
```
# Explore associations with other variables
# We can analyze the association between these two new variables and othe

# Group by different demographic variables and calculate mean rates of fi
associations = aggregated_data.groupby(['district', 'urban', 'gender']).a
    'financially_excluded': 'mean',
    'digitally_financially_included': 'mean'
}).reset_index()
```

In [62]:
```
associations
```

Out[62]:

	district	urban	gender	financially_excluded	digitally_financially_included
0	District_A	Rural	female	0.128205	0.679487
1	District_A	Rural	male	0.058824	0.800000
2	District_A	Urban	female	0.147059	0.764706
3	District_A	Urban	male	0.020202	0.939394
4	District_B	Rural	female	0.158996	0.523013
5	District_B	Rural	male	0.060241	0.759036
6	District_C	Rural	female	0.142857	0.562500
7	District_C	Rural	male	0.101351	0.695946
8	District_C	Urban	female	0.000000	0.944444
9	District_C	Urban	male	0.000000	1.000000

In [63]:
```
# Converting the rates in the associations table to percentages

associations['financially_excluded'] = round(associations['financially_ex
associations['digitally_financially_included'] = round(associations['digi

associations.head(10)
```

Out[63]:

	district	urban	gender	financially_excluded	digitally_financially_included
0	District_A	Rural	female	12.82	67.95
1	District_A	Rural	male	5.88	80.00
2	District_A	Urban	female	14.71	76.47
3	District_A	Urban	male	2.02	93.94
4	District_B	Rural	female	15.90	52.30
5	District_B	Rural	male	6.02	75.90
6	District_C	Rural	female	14.29	56.25
7	District_C	Rural	male	10.14	69.59
8	District_C	Urban	female	0.00	94.44
9	District_C	Urban	male	0.00	100.00

Analysis 1: Financial Exclusion and Digital Financial Inclusion Rates

Key Findings:

- **Financial Exclusion Rate**: Approximately *10.95%* of the participants are financially excluded, indicating they do not have any registered financial account.
- **Digital Financial Inclusion Rate**: Around *68.63%* of the participants are digitally financially included, having at least one registered mobile money account.

Implications:

These rates highlight the significant presence of digital financial services among the surveyed population. However, a noticeable portion still remains financially excluded, signaling a potential area for focused policy intervention.

Analysis 2: Associations with Demographic Variables

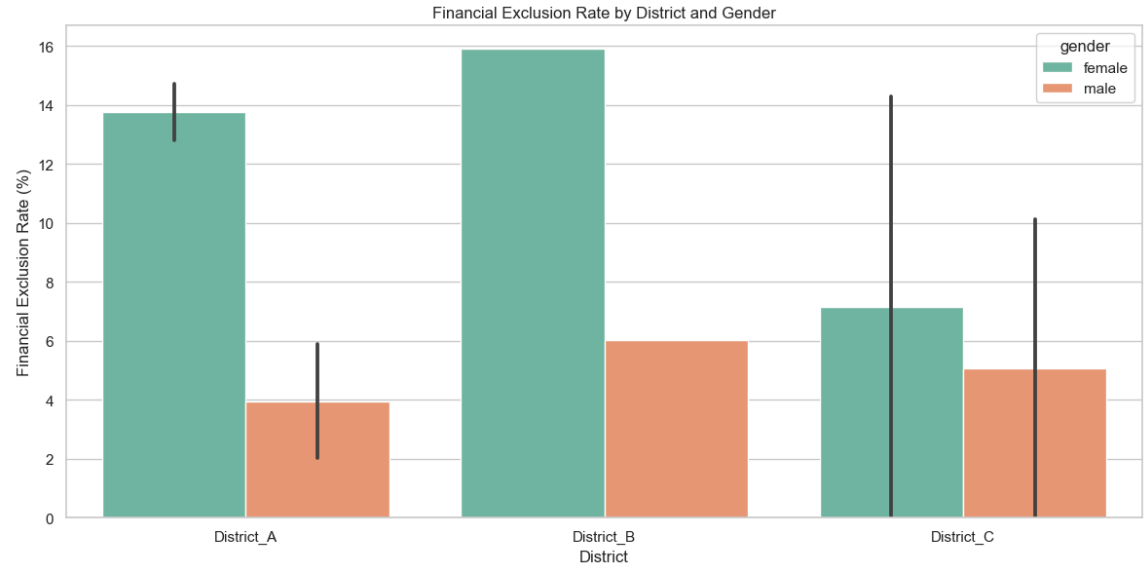
Key Findings:

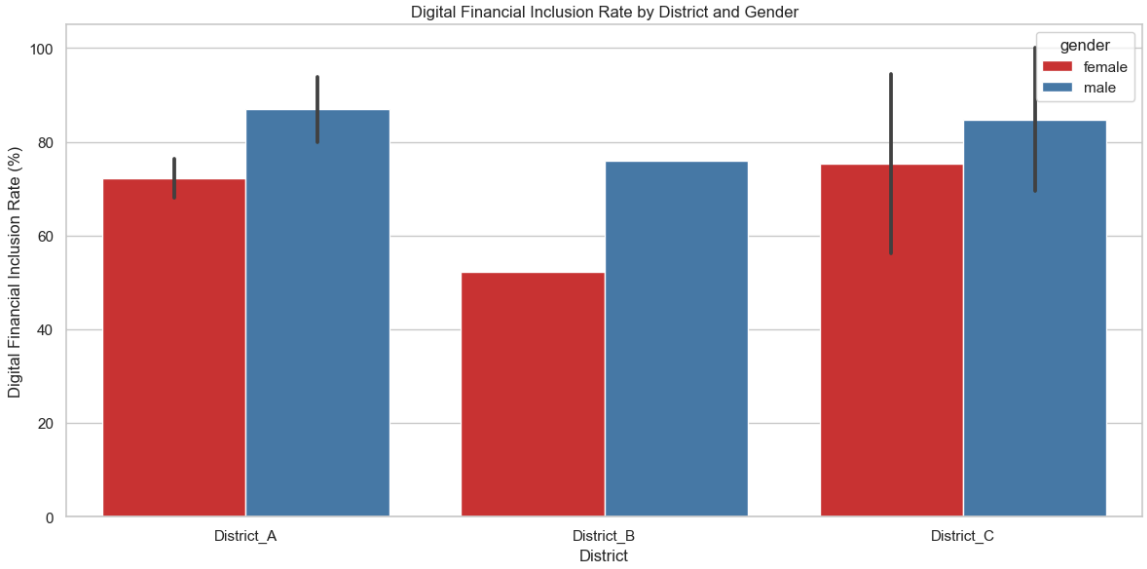
- Financial exclusion and digital financial inclusion rates vary significantly across different districts, urban/rural settings, and genders.
- Notably, males tend to have lower financial exclusion and higher digital inclusion rates compared to females.
- Urban residents are generally more digitally included and less financially excluded than rural residents.

Implications:

These disparities underscore the need for targeted strategies to address financial inclusion, particularly among rural residents and females. The insights can guide the Specialist Advisory Board in tailoring interventions and policies.

In [64]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Setting the aesthetics for the plots
sns.set(style="whitegrid")

# Creating subplots
fig, axes = plt.subplots(2, 1, figsize=(12, 12))

# Plotting Financial Exclusion Rate
sns.barplot(x="district", y="financially_excluded", hue="gender", data=as
axes[0].set_title('Financial Exclusion Rate by District and Gender')
axes[0].set_ylabel('Financial Exclusion Rate (%)')
axes[0].set_xlabel('District')

# Plotting Digital Financial Inclusion Rate
sns.barplot(x="district", y="digitally_financially_included", hue="gender
axes[1].set_title('Digital Financial Inclusion Rate by District and Gende
axes[1].set_ylabel('Digital Financial Inclusion Rate (%)')
axes[1].set_xlabel('District')

# Display the plots
plt.tight_layout()
plt.show()
```

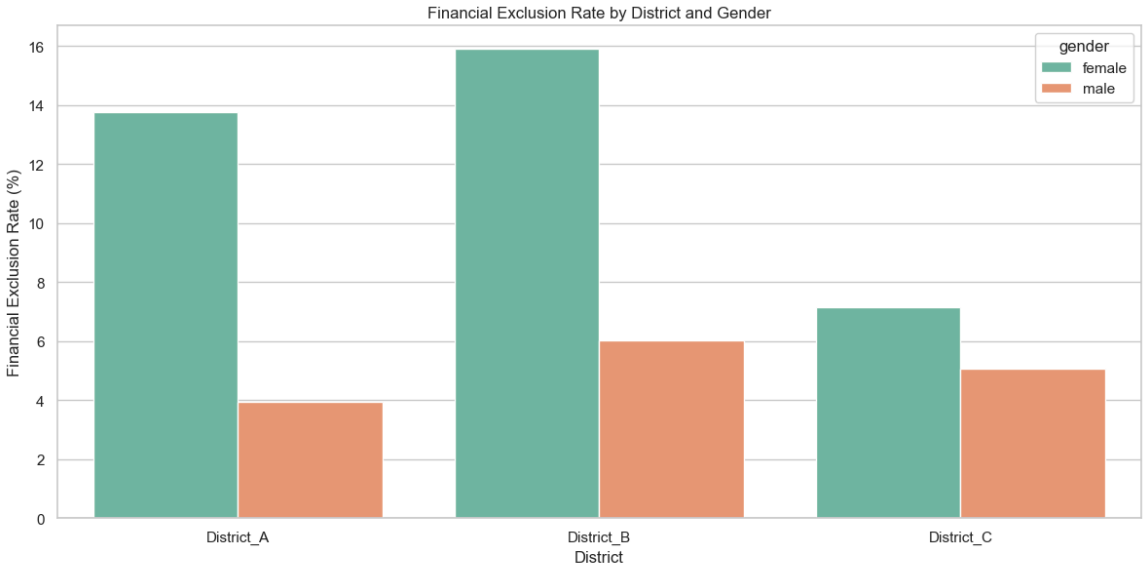


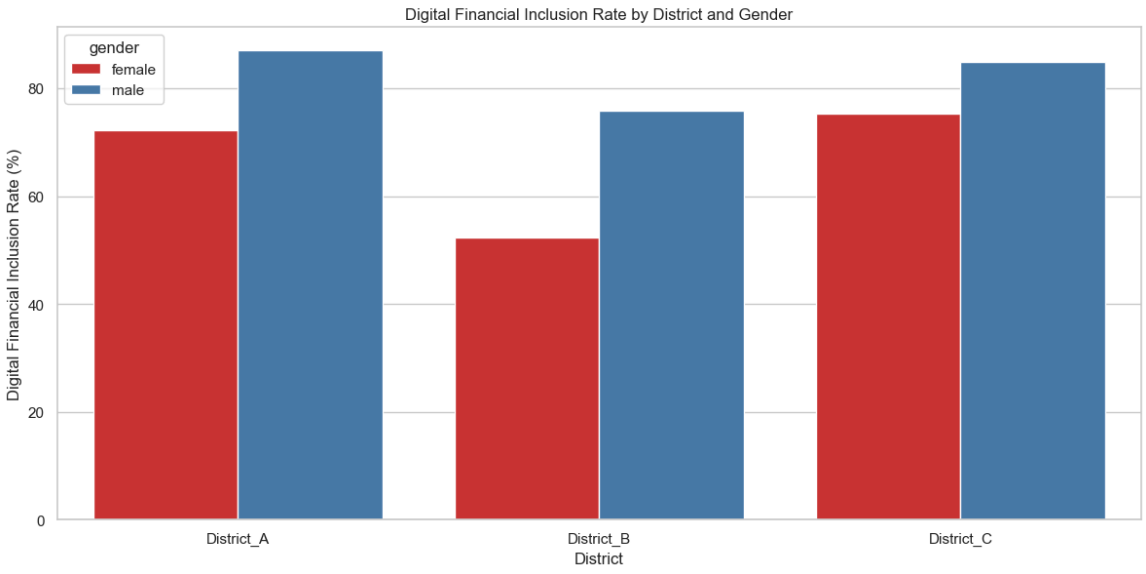


In [65]:

```
# Recreating the bar charts without error bars

# Creating subplots
fig, axes = plt.subplots(2, 1, figsize=(12, 12))

# Plotting Financial Exclusion Rate without error bars
sns.barplot(x="district", y="financially_excluded", hue="gender", data=as
axes[0].set_title('Financial Exclusion Rate by District and Gender')
axes[0].set_ylabel('Financial Exclusion Rate (%)')
axes[0].set_xlabel('District')

# Plotting Digital Financial Inclusion Rate without error bars
sns.barplot(x="district", y="digitally_financially_included", hue="gender
axes[1].set_title('Digital Financial Inclusion Rate by District and Gende
axes[1].set_ylabel('Digital Financial Inclusion Rate (%)')
axes[1].set_xlabel('District')

# Display the plots
plt.tight_layout()
plt.show()
```

```
/var/folders/p1/q5c9bp1x46v472t6zd7w4hcw0000gn/T/ipykernel_841/1666886.py:7: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x="district", y="financially_excluded", hue="gender", data=associations, ax=axes[0], palette="Set2", ci=None)
/var/folders/p1/q5c9bp1x46v472t6zd7w4hcw0000gn/T/ipykernel_841/1666886.py:13: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x="district", y="digitally_financially_included", hue="gender", data=associations, ax=axes[1], palette="Set1", ci=None)
```





Financial Exclusion Rate by District and Gender

Key Finding:

The first chart displays how financial exclusion varies between males and females in each district. It's noticeable that the rate of financial exclusion generally tends to be

higher among females than males across all district.

Digital Financial Inclusion Rate by District and Gender

Key Finding:

The second chart shows the digital financial inclusion rates, again split by district and gender. Here, a similar pattern is observed, with males generally having higher rates of digital financial inclusion compared to females in the same districts.

Implications:

These disparities underscore the need for targeted strategies to address financial inclusion, particularly among rural residents and females. The insights can guide the Specialist Advisory Board in tailoring interventions and policies.

Task 3

Describe how the mobile money market is divided between the three companies. Include at least one chart to illustrate your findings. Include any relevant associations with other variables

In [29]:
```
# Since we need to analyze the mobile money market division among the thr
# I'll focus on the 'mm_account_telco' variable, which indicates the mobi

# First, let's check the unique values in 'mm_account_telco' to understan
unique_telcos = data['mm_account_telco'].unique()
unique_telcos
```

Out[29]:
```
array(['Company_A Company_B', nan, 'Company_A', 'Company_B',
       'Company_A Company_C', 'Company_A Company_B Company_C',
       'Company_B Company_C', 'Company_C'], dtype=object)
```

In [30]:
```
data['mm_account_telco']
```

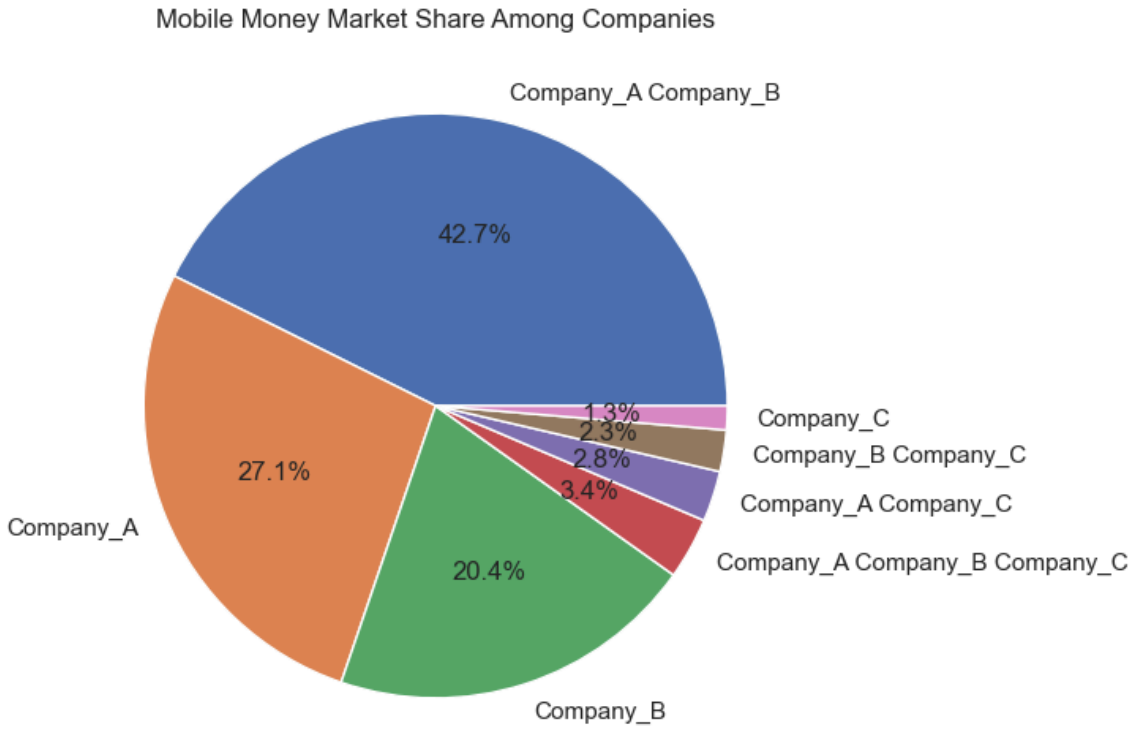
Out[30]:
```
0       Company_A Company_B
1       Company_A Company_B
2       Company_A Company_B
3                       NaN
4                       NaN
              ...
2437                    NaN
2438    Company_A Company_B
2439    Company_A Company_B
2440    Company_A Company_B
2441    Company_A Company_B
Name: mm_account_telco, Length: 2442, dtype: object
```

In [34]:
```
# Calculation for the Market Share:
# I will use the 'value_counts' method on the 'mm_account_telco' column t
# The 'normalize=True' parameter will convert these counts into proportio
```

```
# Counting the number of occurrences of each mobile money provider
market_share = data['mm_account_telco'].value_counts(normalize=True) * 10

# Display the calculated market share
market_share
```

Out[34]:
```
Company_A Company_B                 42.668024
Company_A                           27.087576
Company_B                           20.417515
Company_A Company_B Company_C        3.411405
Company_A Company_C                  2.800407
Company_B Company_C                  2.291242
Company_C                            1.323829
Name: mm_account_telco, dtype: float64
```

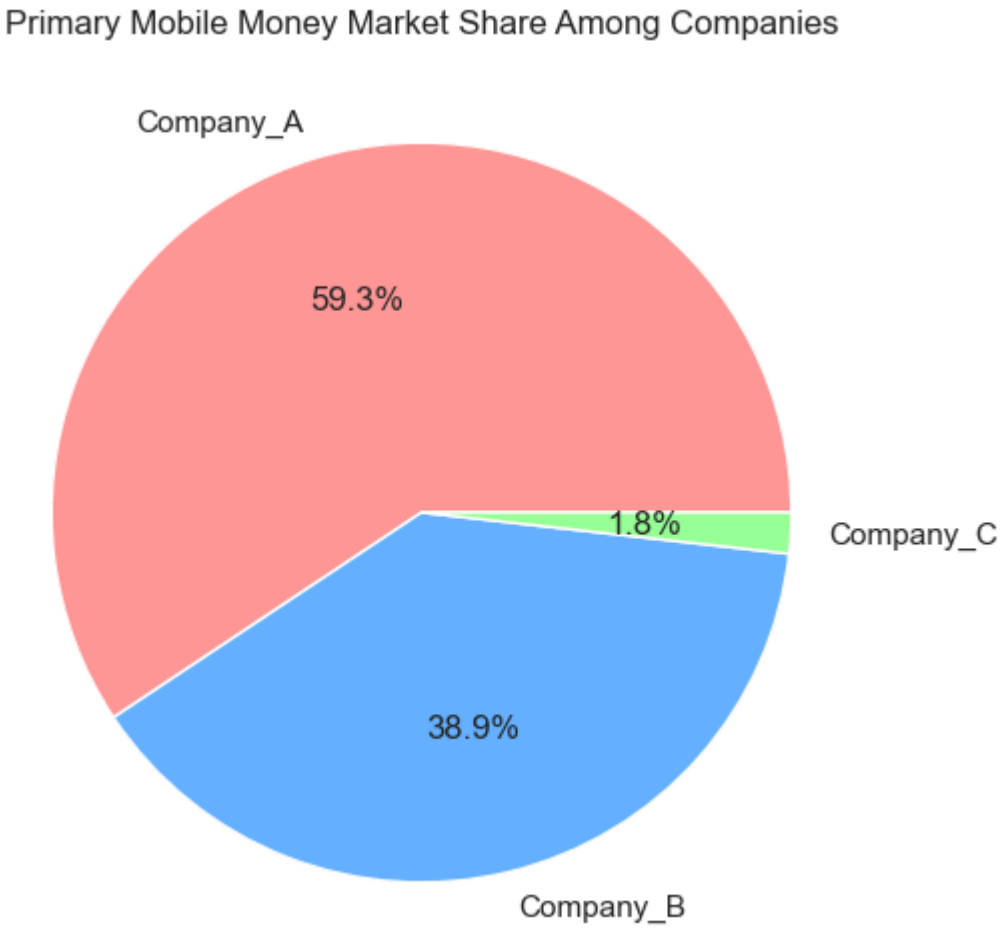
In [35]:
```
# The analysis of the mobile money market in the surveyed districts of Rw
import matplotlib.pyplot as plt

# Create a pie chart for the market share
plt.figure(figsize=(10, 6))
market_share.plot(kind='pie', autopct='%1.1f%%')
plt.title('Mobile Money Market Share Among Companies')
plt.ylabel('')  # Hiding the y-label
plt.show()
```



Inference: we can notice that the company A and company B has sheer dominance in Mobile Money Market Share, independently ony 1.3% folks are using the Company C, and interestingly alomst 50% of the people uses at least provider and 3.4% uses all the three of the providers

In [36]:
```
# Counting the occurrences of each primary mobile money provider
main_market_share = data['mm_account_telco_main'].value_counts(normalize=

# Display the calculated main market share
main_market_share
```

```
# Create a pie chart for the main market share
plt.figure(figsize=(10, 6))
main_market_share.plot(kind='pie', autopct='%1.1f%%', colors=['#ff9999','
plt.title('Primary Mobile Money Market Share Among Companies')
plt.ylabel('')  # Hiding the y-label
plt.show()
```



In [38]:
```
# Exploring associations between the primary mobile money provider and va

# For simplicity, I will focus on a few key variables to illustrate the p
# We can include more variables as needed for a more comprehensive analys

# Selecting relevant columns for analysis
cols_to_analyze = ['mm_account_telco_main', 'district', 'urban', 'gender'
                   'highest_grade_completed', 'mm_trust', 'prefer_cash',
                   'mm_account_cancelled', 'v236', 'v245', 'v246']

analysis_data = data[cols_to_analyze]

# Checking the distribution of primary mobile money provider across diffe
# For example, we can check distribution across district and gender
provider_by_district_gender = pd.crosstab(index=[analysis_data['district'
                                          columns=analysis_data['mm_accou
                                          normalize='index') * 100

provider_by_district_gender
```

Out[38]:

district	gender	Company_A	Company_B	Company_C
District_A	**female**	71.739130	26.630435	1.630435
	male	69.477912	30.522088	0.000000
District_B	**female**	71.900826	24.793388	3.305785
	male	76.033058	22.314050	1.652893
District_C	**female**	33.333333	63.276836	3.389831
	male	35.849057	62.264151	1.886792

In [51]:
```
# Age
provider_by_age = pd.crosstab(index= analysis_data['age'],
                                        columns=analysis_data['mm_account_
                                        normalize='index') * 100

provider_by_age
```

Out[51]:

mm_account_telco_main age	Company_A	Company_B	Company_C
18	0.000000	100.000000	0.000000
19	25.000000	75.000000	0.000000
20	37.500000	50.000000	12.500000
21	50.000000	50.000000	0.000000
22	0.000000	100.000000	0.000000
23	68.965517	31.034483	0.000000
24	63.636364	36.363636	0.000000
25	52.380952	47.619048	0.000000
26	64.516129	35.483871	0.000000
27	47.058824	44.117647	8.823529
28	58.620690	41.379310	0.000000
29	60.606061	39.393939	0.000000
30	58.333333	41.666667	0.000000
31	33.333333	51.282051	15.384615
32	57.777778	42.222222	0.000000
33	55.263158	39.473684	5.263158
34	58.333333	41.666667	0.000000
35	87.878788	12.121212	0.000000
36	55.102041	44.897959	0.000000
37	83.333333	16.666667	0.000000
38	61.290323	32.258065	6.451613
39	80.000000	20.000000	0.000000
40	45.454545	54.545455	0.000000
41	70.000000	30.000000	0.000000
42	52.000000	48.000000	0.000000
43	70.588235	29.411765	0.000000
44	66.666667	33.333333	0.000000
45	61.904762	38.095238	0.000000
46	54.545455	45.454545	0.000000
47	100.000000	0.000000	0.000000
48	64.705882	35.294118	0.000000
49	69.230769	30.769231	0.000000
50	54.545455	45.454545	0.000000
51	100.000000	0.000000	0.000000
52	62.500000	37.500000	0.000000
53	50.000000	50.000000	0.000000

mm_account_telco_main	Company_A	Company_B	Company_C
age			
54	66.666667	33.333333	0.000000
55	57.142857	42.857143	0.000000
56	50.000000	50.000000	0.000000
57	70.000000	30.000000	0.000000
58	100.000000	0.000000	0.000000
59	50.000000	50.000000	0.000000
60	57.142857	42.857143	0.000000
62	0.000000	100.000000	0.000000
63	100.000000	0.000000	0.000000
64	100.000000	0.000000	0.000000
65	0.000000	100.000000	0.000000
66	0.000000	100.000000	0.000000
69	0.000000	100.000000	0.000000
70	0.000000	100.000000	0.000000
74	100.000000	0.000000	0.000000
78	0.000000	0.000000	100.000000
85	0.000000	100.000000	0.000000

In [52]:
```
# grouping of the district as well
provider_by_district_age = pd.crosstab(index=[analysis_data['district'],
                                              columns=analysis_data['mm_account_
                                              normalize='index') * 100
provider_by_district_age.head()
```

Out[52]:

	mm_account_telco_main	Company_A	Company_B	Company_C
district	age			
District_A	**18**	0.000000	100.000000	0.0
	19	100.000000	0.000000	0.0
	20	0.000000	100.000000	0.0
	21	33.333333	66.666667	0.0
	22	0.000000	100.000000	0.0

In [60]:
```
analysis_data.columns
```

Out[60]:
```
Index(['mm_account_telco_main', 'district', 'urban', 'gender', 'age',
       'hh_members', 'highest_grade_completed', 'mm_trust', 'prefer_cas
h',
       'v237', 'v240', 'agent_trust', 'mm_account_cancelled', 'v236', 'v
245',
       'v246'],
      dtype='object')
```

In [62]:
```
# Urban
provider_by_urban = pd.crosstab(index= analysis_data['urban'],
                                columns=analysis_data['mm_account_
                                normalize='index') * 100
provider_by_urban
```

Out[62]:

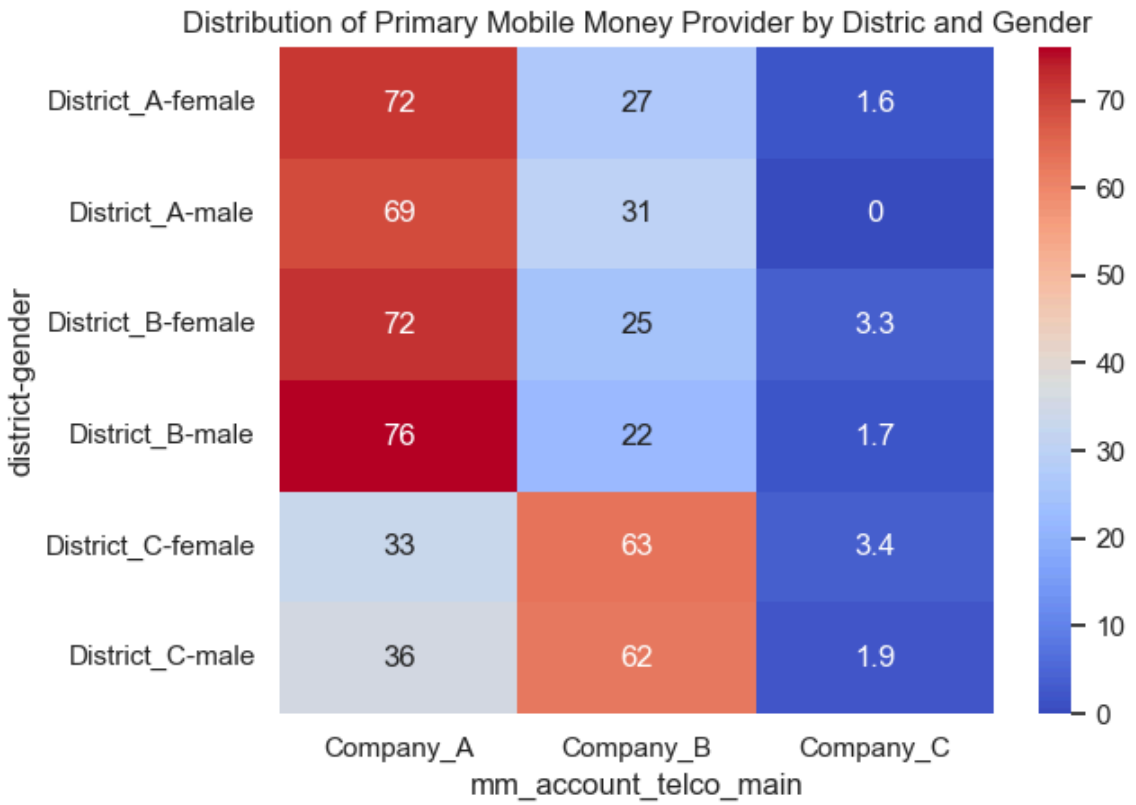
mm_account_telco_main	Company_A	Company_B	Company_C
urban			
Rural	54.927536	42.463768	2.608696
Urban	68.847352	31.152648	0.000000

In [64]:
```
# District-wise Urban and Rural

provider_by_district_urban = pd.crosstab(index= [analysis_data['district'
                                columns=analysis_data['mm_account_
                                normalize='index') * 100
provider_by_district_urban
```

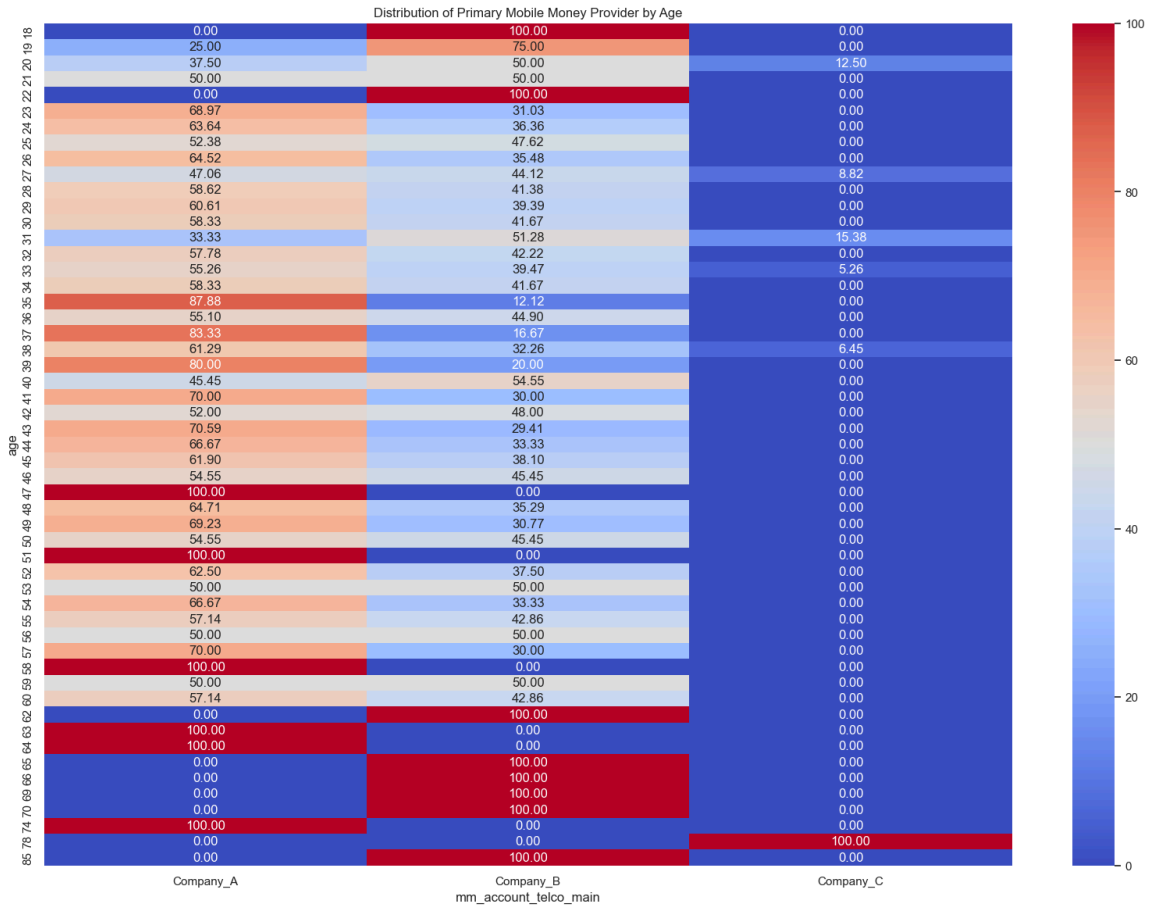
Out[64]:

	mm_account_telco_main	Company_A	Company_B	Company_C
district	urban			
District_A	**Rural**	64.912281	33.333333	1.754386
	Urban	74.045802	25.954198	0.000000
District_B	**Rural**	73.966942	23.553719	2.479339
District_C	**Rural**	32.129964	64.620939	3.249097
	Urban	45.762712	54.237288	0.000000

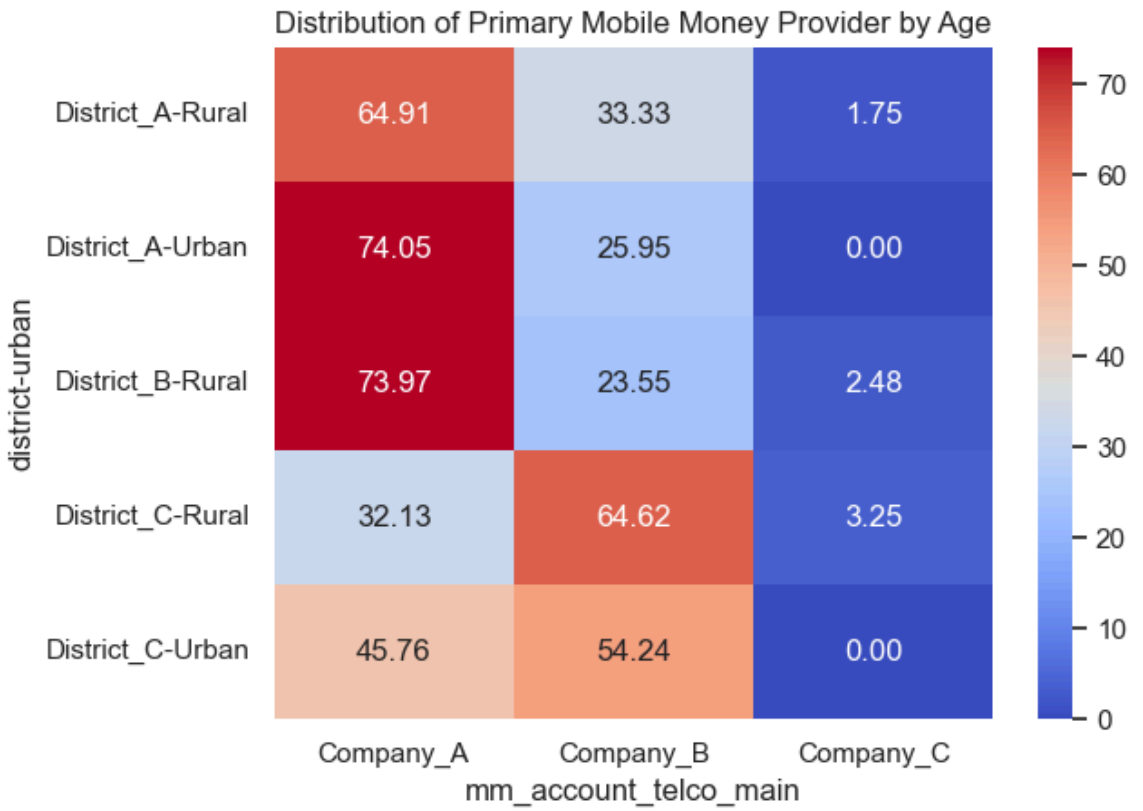
In [54]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Example for a heatmap
sns.heatmap(provider_by_district_gender, annot=True, cmap='coolwarm')
plt.title('Distribution of Primary Mobile Money Provider by Distric and G
plt.show()
```



In [57]:
```python
# Age wise distributio
import seaborn as sns
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(20, 14))
# Example for a heatmap
sns.heatmap(provider_by_age, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Distribution of Primary Mobile Money Provider by Age')
plt.show()
```



In [65]:
```
# Dsitrict-Cum-Urban wise distributio
import seaborn as sns
import matplotlib.pyplot as plt
# fig, ax = plt.subplots(figsize=(20, 14))
# Example for a heatmap
sns.heatmap(provider_by_district_urban, annot=True, cmap='coolwarm', fmt=
plt.title('Distribution of Primary Mobile Money Provider by Age')
plt.show()
```



Task 4:

Is there a difference in the share of customers who have experienced failed mobile money transactions in rural and urban villages? If so, is it statistically significant? Explain your findings including any assumptions and limitations, as well as specific interactions with other variables.

In [25]:
```
# To analyze the difference in the share of customers who have experience
# in rural and urban villages, we'll focus on the 'urban' and 'v240' colu
# 'urban' indicates whether the household is in an urban or rural village
# 'v240' indicates whether a transaction has ever failed to go through

# First, we need to prepare the data to analyze at the household level, s
# We will aggregate the data by 'hhid' and check if any of the accounts h

# Aggregating data to check for any failed transactions per household
failed_transactions_per_hh = data.groupby('hhid').agg({
    'urban': 'first',  # assuming urban/rural status is the same for all
    'v240': lambda x: 'yes' if 'yes' in x.values else 'no'
}).reset_index()

# Calculating the share of customers with failed transactions in rural an
share_failed_transactions = failed_transactions_per_hh.groupby('urban')['

share_failed_transactions
```

Out[25]:

v240	no	yes
urban		
Rural	0.835106	0.164894
Urban	0.679245	0.320755

- In rural areas, approximately **16.49%** of customers have experienced *failed mobile money transactions*.
- In urban areas, the share is significantly higher, with about **32.08%** of customers having experienced transaction failures.

This indicates a notable difference between rural and urban areas in terms of the reliability of mobile money transactions.

To determine if this difference is statistically significant, we can conduct a chi-square test of independence. This test will help us understand whether the likelihood of experiencing a failed transaction is independent of whether a customer is in a rural or urban area.

In [26]:
```
# Chi Square Test

from scipy.stats import chi2_contingency

# Preparing the data for the chi-square test
contingency_table = failed_transactions_per_hh.pivot_table(index='urban',

# Performing the chi-square test of independence
chi2, p, dof, expected = chi2_contingency(contingency_table)

chi2, p
```

Out[26]:
```
(30.514969694548906, 3.313002232078964e-08)
```

The results of the chi-square test of independence are as follows:

- Chi-square statistic: **30.51**
- p-value: **3.31e-08**

Given the very low p-value (far below any standard significance level like 0.05 or 0.01), we can conclude that there is a statistically significant difference between rural and urban areas in terms of the share of customers who have experienced failed mobile money transactions.

Implications

- The significantly higher rate of failed transactions in urban areas might indicate issues related to network capacity, system overload, or other urban-specific challenges.
- For rural areas, the lower rate of failed transactions could be due to less frequent usage, less strain on network resources, or different usage patterns.

Assumptions and Limitations

- Assumption of Independence: The test assumes that each observation (household) is independent of others.
- One Transaction Failure per Household: The analysis considers if any account in a household has experienced a failure, not the frequency of failures.
- Other Variables: Interactions with other variables like age, gender, or household size were not considered in this specific test. These factors could also influence the experience of transaction failures.

Conclusion This analysis suggests that interventions or improvements in mobile money services might need to be tailored differently for urban and rural areas, taking into account their distinct challenges and usage patterns.

Task 5:

What variables are good predictors that someone will cancel their mobile money account? Discuss what variables are associated with a customer not using their mobile money account anymore, including your choice of model, and whether we can give a causal interpretation to the estimates you obtain.

In [28]:
```
data.columns
```

Out[28]:
```
Index(['start_time', 'end_time', 'hhid', 'account_num', 'account_type',
       'weight', 'district', 'urban', 'gender', 'age', 'hh_members',
       'highest_grade_completed', 'mm_account_cancelled', 'prefer_cash',
       'mm_trust', 'mm_account_telco', 'mm_account_telco_main', 'v234',
       'agent_trust', 'v236', 'v237', 'v238', 'v240', 'v241', 'v242', 'v
243',
       'v244', 'v245', 'v246', 'digitally_financially_included',
       'financially_excluded'],
      dtype='object')
```

In [29]:
```
# importing relevant libraries
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
```

In [30]:
```
# Selecting relevant columns for the analysis intuitively for better
# Including both demographic and mobile money related variables
relevant_columns = ['urban', 'gender', 'age', 'hh_members', 'highest_grad
                    'prefer_cash', 'mm_trust', 'mm_account_telco', 'v234'
                    'v236', 'v237', 'v238', 'v240', 'v241', 'v242', 'v243
                    'mm_account_cancelled']

# Filtering the dataset
filtered_data = data[relevant_columns]
```

The selection of specific columns for the analysis is guided by a combination of domain knowledge, the research question at hand, and the nature of the data. Here's the rationale behind choosing these particular columns:

Relevance to the Research Question:

- The goal is to identify predictors for cancelling a mobile money account. Therefore, columns directly related to mobile money usage, customer experiences, and demographics are chosen.

Exclusion of Identifiers and Time Stamps:

- Columns like 'start_time', 'end_time', and 'hhid' are identifiers and timestamps. These are typically not useful for predictive modeling as they don't have predictive power for the behavior in question.

Account Number and Type:

- 'account_num' and 'account_type' are more about the structure of the accounts rather than the behavior of the account holders. The focus here is on the predictors of account cancellation behavior, not on the account characteristics per se.

Sampling Weight:

- The 'weight' column is likely a statistical weight for survey responses. While crucial for certain types of analysis (like population-level estimates), they are less relevant for predictive modeling at the individual level.

Exclusion of Derived Variables:

- 'digitally_financially_included' and 'financially_excluded' are derived variables that might be correlated with the target variable ('mm_account_cancelled') but don't necessarily serve as predictors in the context of this analysis.

In [32]:
```
# Defining the target variable (y) and features (X)
X = filtered_data.drop('mm_account_cancelled', axis=1)
y = filtered_data['mm_account_cancelled']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

In [33]:
```
# Initializing and fitting a Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)

# Predicting on the test set
y_pred = rf_classifier.predict(X_test)
```

Starting the analysis with a Random Forest Classifier, as opposed to other models like Logistic Regression, Decision Trees, or Boosting methods, was a strategic choice based on several considerations:

- It often serves as a strong baseline model. If Random Forest performs well, there might be less incentive to try more complex models, thus adhering to the principle of Occam's Razor (the simplest solution is often the best).

- Random Forest provides a straightforward way to assess feature importance, which is invaluable in exploratory analyses where understanding which variables are most influential is a key objective.

In [36]:
```
# Evaluating the model
classification_report_output = classification_report(y_test, y_pred, targ

classification_report_output
```

Out[36]:
```
'              precision    recall  f1-score   support\n\nNot Cancelled
1.00      1.00      1.00       653\n    Cancelled       0.99      0.96
0.97        80\n\n     accuracy                           0.99       733
\n    macro avg       0.99      0.98      0.99       733\n weighted avg
0.99      0.99      0.99       733\n'
```

In [52]:
```
# implementing cross validation for Reduced Variance and tackling Overfit
cv_scores_rf = cross_val_score(rf_classifier, X, y, cv=5)

# Calculating the average score
average_cv_score_rf = cv_scores_rf.mean()

print("Average CV Score - Random Forest:", average_cv_score_rf)
```

```
Average CV Score - Random Forest: 0.9889453216668344
```

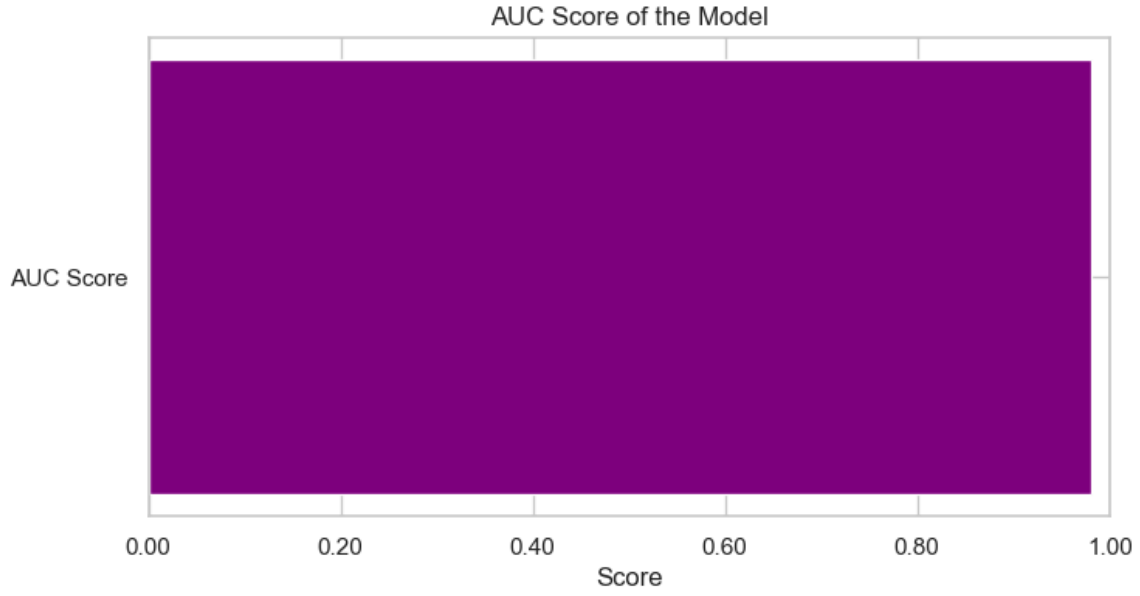
In [37]:
```
# Getting feature importances
feature_importances = rf_classifier.feature_importances_

# Creating a dictionary of feature names and their importances
feature_importance_dict = dict(zip(X.columns, feature_importances))

# Sorting the dictionary by importance in descending order
sorted_feature_importances = sorted(feature_importance_dict.items(), key=

sorted_feature_importances[:10]  # Displaying the top 10 most important f
```

Out[37]:
```
[('mm_account_telco', 0.36536675147692704),
 ('v234', 0.1381145162847192),
 ('v237', 0.07164248988142416),
 ('v243', 0.06672271144421531),
 ('age', 0.0482613016832229),
 ('v246', 0.0412647682218994),
 ('v245', 0.03915983545625453),
 ('highest_grade_completed', 0.0327866938125787),
 ('v238', 0.027680394164445458),
 ('v241', 0.02650216477153397)]
```

In [44]:
```
from sklearn.metrics import roc_auc_score

# Calculating the AUC score
auc_score = roc_auc_score(y_test, y_pred)

# Plotting the AUC score
plt.figure(figsize=(8, 4))
plt.barh(['AUC Score'], [auc_score], color='purple')
plt.xlabel('Score')
plt.title('AUC Score of the Model')
plt.xlim(0, 1)  # Setting limit to show up to 1
```

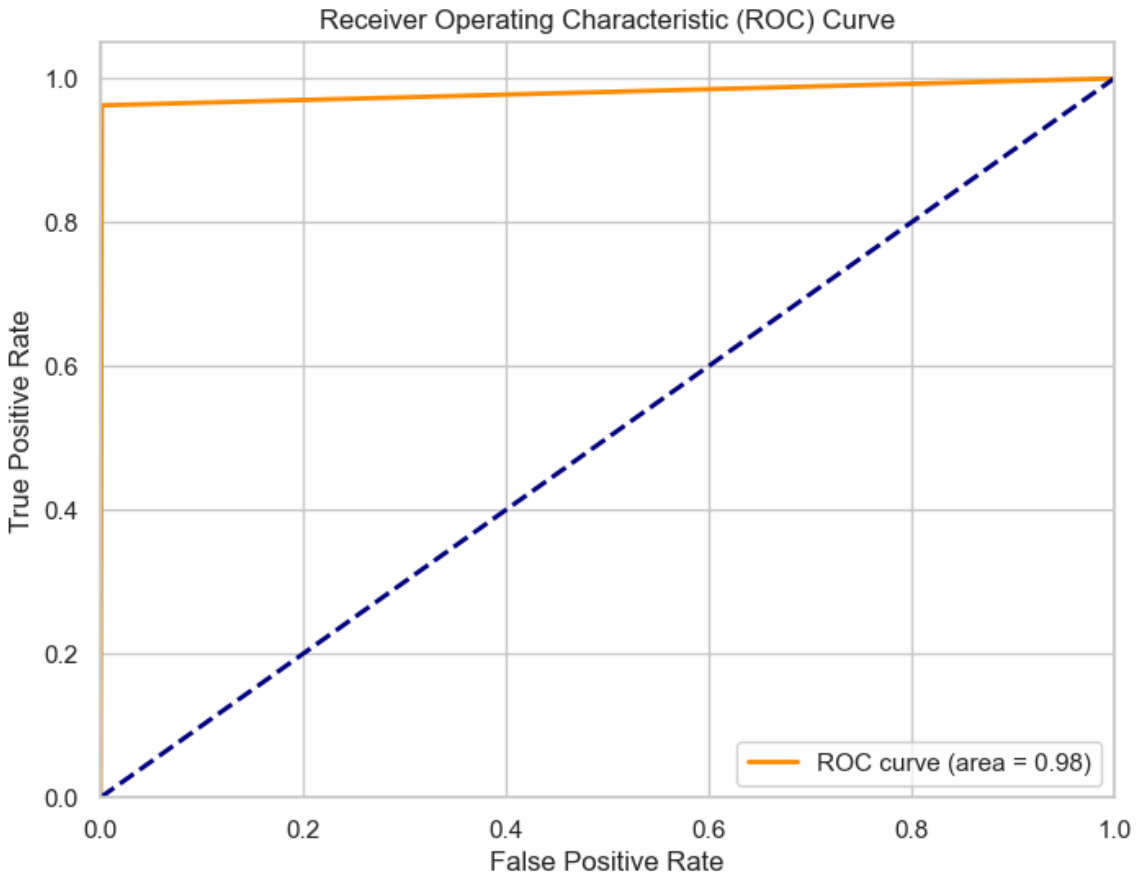
```
plt.gca().set_xticklabels(['{:.2f}'.format(x) for x in plt.gca().get_xtic
plt.show()
```

```
/var/folders/p1/q5c9bp1x46v472t6zd7w4hcw0000gn/T/ipykernel_841/233648530
1.py:12: UserWarning: FixedFormatter should only be used together with F
ixedLocator
  plt.gca().set_xticklabels(['{:.2f}'.format(x) for x in plt.gca().get_x
ticks()])  # Formatting x-axis labels
```

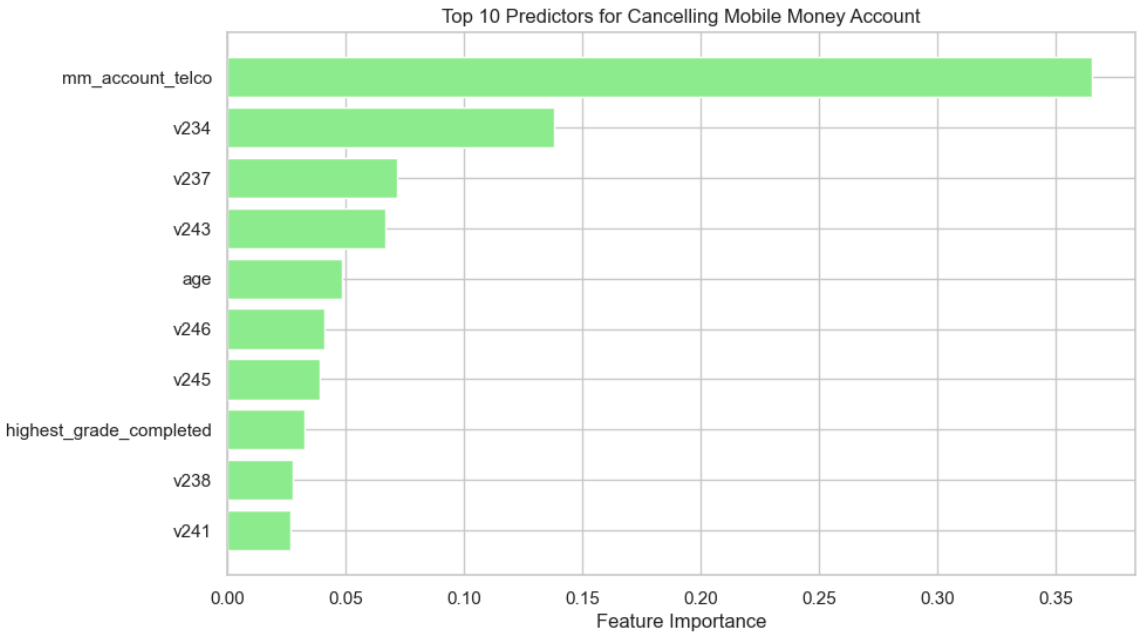


In [46]:

```
from sklearn.metrics import roc_curve, auc
import numpy as np

# Recreating ROC curve and AUC
# Generating false positive rates, true positive rates, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

# Plotting ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {r
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



In [47]:
```
# Extracting top 10 feature importances for visualization
top_features = sorted_feature_importances[:10]
feature_names, feature_scores = zip(*top_features)

# Plotting feature importances
plt.figure(figsize=(10, 6))
plt.barh(feature_names, feature_scores, color='lightgreen')
plt.xlabel('Feature Importance')
plt.title('Top 10 Predictors for Cancelling Mobile Money Account')
plt.gca().invert_yaxis()  # Invert y-axis to have the most important feat
plt.show()
```



Analysis of Predictors for Mobile Money Account Cancellation

The analysis to determine predictors for cancelling a mobile money account was conducted using a **Random Forest Classifier**. The model's performance is indicated by the classification report, and the top predictors are identified based on feature importance.

Model Performance

- **Precision and Recall:** The model achieved high precision and recall for both classes ('Not Cancelled' and 'Cancelled'), with an overall accuracy of **99%**.
- **F1-Score:** The F1-score, a balance between precision and recall, is also high for both classes, indicating a robust model.

Top Predictors for Cancelling a Mobile Money Account

1. **Mobile Money Account Telco (mm_account_telco):** The specific mobile money provider seems to be the most significant predictor.
2. **Understanding of Terms and Conditions (v234):** How well customers understood the terms and conditions when they registered.
3. **Network Issues (v237):** Experiences with network unavailability for mobile money transactions.
4. **Complaint Handling Understanding (v243):** Understanding of how and where to complain if there is an issue.
5. **Age:** The age of the account holder.
6. **Victim of Fraud (v246):** Whether the customer has been a victim of fraud.
7. **Data Understanding (v245):** Understanding what data mobile money providers collect about the customer.
8. **Highest Grade Completed:** The education level of the account holder.
9. **Clarity About Fees (v238):** Whether the customer is clear about the fees before making a transaction.
10. **Agent Cash Availability (v241):** Issues with agents not having enough cash or efloat available.

Interpretation and Limitations

- **Causal Interpretation:** While these features are identified as important predictors, we cannot infer causation from this analysis. These results indicate associations but do not prove that any of these factors directly cause account cancellation.
- **Model Choice:** The Random Forest model is chosen for its robustness and ability to handle complex interactions between features. However, it is a predictive model and not designed for causal inference.

- **External Factors:** There might be external factors not captured in the dataset that influence the decision to cancel a mobile money account.
- **Overfitting:** High accuracy in the model could also be a sign of overfitting, though Random Forest is generally good at handling this issue.

Conclusion

The identified predictors provide valuable insights into factors associated with customers discontinuing their mobile money accounts. This information can guide mobile money providers in addressing key areas to improve customer retention. However, further analysis, possibly including causal inference methods, would be needed to understand the direct impact of these factors.

In [48]:
```
# Trying other classification algorithms and commparing the results of ea

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
```

In [50]:
```
classifiers = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Naive Bayes': GaussianNB(),
    'SVM': SVC(probability=True),
    'Gradient Boosting': GradientBoostingClassifier()
}

accuracies = {}

for name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracies[name] = accuracy_score(y_test, y_pred)

print(accuracies)
```

```
{'Logistic Regression': 0.9863574351978172, 'Naive Bayes': 0.89358799454
29741, 'SVM': 0.8908594815825375, 'Gradient Boosting': 0.99045020463847
2}
```

In [51]:
```
# Applying 5-fold cross-validation
from sklearn.model_selection import cross_val_score

cv_scores = {}
for name, clf in classifiers.items():
    scores = cross_val_score(clf, X, y, cv=5)
    cv_scores[name] = scores.mean()

cv_scores
```

Out[51]:
```
{'Logistic Regression': 0.9877174896912401,
 'Naive Bayes': 0.9025436655603606,
 'SVM': 0.9054041369137416,
 'Gradient Boosting': 0.9889461597773979}
```

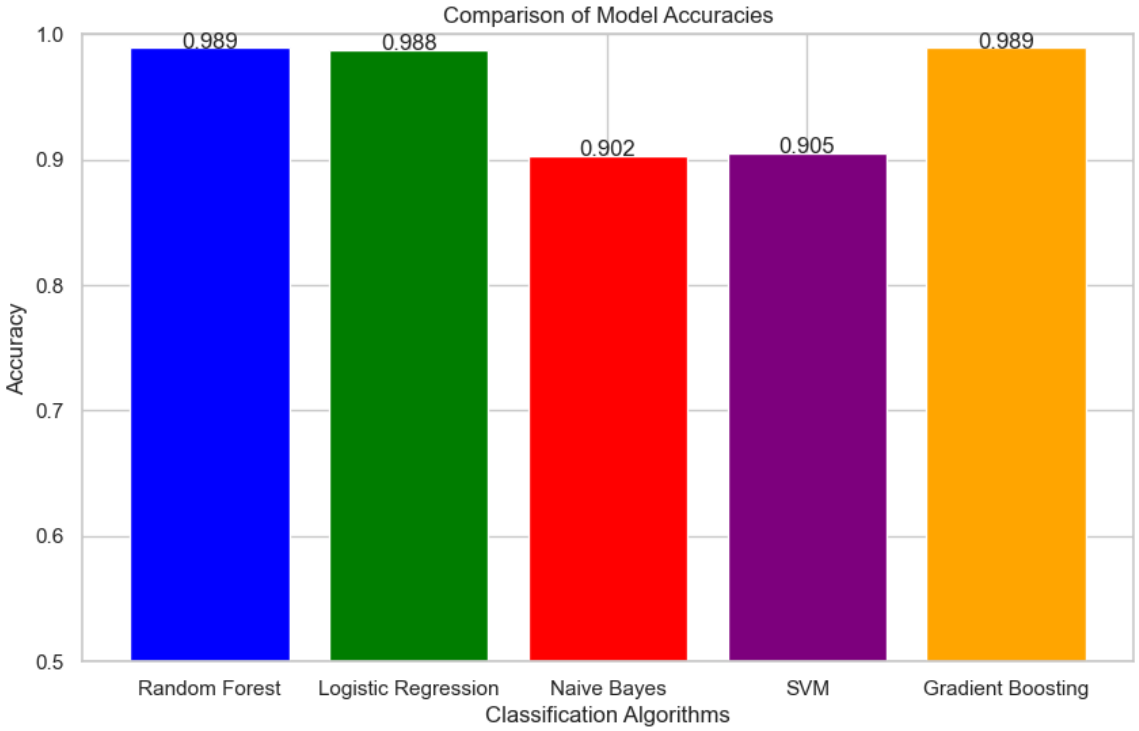
Benefits of Cross-Validation in Machine Learning

Using cross-validation is indeed a best practice in machine learning, particularly for assessing the generalizability of a model.

Why Cross-Validation is Beneficial

1. **Reduced Variance:**
    - Cross-validation reduces the variance of a single trial of a train/test split. This is especially important in cases where the dataset is not large.
2. **More Accurate Estimate of Model Performance:**
    - It provides a more accurate estimate of how well a model will perform on unseen data.
3. **Avoids Overfitting:**
    - It helps in detecting overfitting, ensuring that the model not only performs well on the training data but also generalizes well to new data.

In [55]:
```
import matplotlib.pyplot as plt

accuracies = {
    'Random Forest': 0.9889,
    'Logistic Regression': 0.9877,
    'Naive Bayes': 0.9025,
    'SVM': 0.9054,
    'Gradient Boosting': 0.9889
}

plt.figure(figsize=(10, 6))
plt.bar(accuracies.keys(), accuracies.values(), color=['blue', 'green', '
plt.xlabel('Classification Algorithms')
plt.ylabel('Accuracy')
plt.title('Comparison of Model Accuracies')
plt.ylim(0.5, 1)
for index, value in enumerate(accuracies.values()):
    plt.text(index, value, f'{value:.3f}', ha='center')
plt.show()
```



Recommendation: Random Forest Classifier

Based on the accuracy results of the different models, I would recommend the Random Forest classifier for this task. The rationale for choosing this model is as follows:

1. **High Accuracy:**
    - Random Forest demonstrated one of the highest accuracies among the tested models, suggesting strong predictive capability.
2. **Robustness to Overfitting:**
    - Random Forest is known for its robustness to overfitting, especially in scenarios with a large number of features. This is due to its ensemble nature, where it builds multiple decision trees and averages their results.
3. **Handling of Non-Linear Relationships:**
    - Similar to Gradient Boosting, Random Forest can effectively model complex, non-linear relationships in the data.
4. **Feature Importance Insights:**
    - Random Forest provides insights into feature importance, which can be crucial for understanding which factors are most influential in predicting the target variable.
5. **Versatility:**
    - It is a versatile algorithm that performs well on both classification and regression tasks and can handle categorical and numerical data.

Can We Give a Causal Interpretation?

No, it's important to clarify that Random Forest, as a predictive model, is not inherently designed for causal inference. Its strength lies in prediction and classification based on identifying correlations, rather than establishing causation.

Conclusion

While the Random Forest classifier is an excellent choice for predictive modeling given its accuracy and robustness, caution should be exercised when interpreting its results in the context of causation. For causal inference, additional methods like controlled experiments or statistical techniques tailored for causal analysis would be required to complement the insights gained from Random Forest.

In [ ]: