Bridging the Financial Access Gap: Insights and Recommendations

Abstract

This report delves into the financial access gap that exists across different regions and income groups, with a particular focus on the disparities in account ownership, mobile money adoption, and digital payment usage. Through a combination of machine learning models and Power BI visualizations, this study aims to identify the root causes of these disparities and propose actionable strategies to bridge the gap. The study also highlights how technological innovations, such as mobile banking and digital payments, can be harnessed to close this gap. By providing detailed insights and leveraging comprehensive data analysis, the report offers practical recommendations for policymakers, financial institutions, and other key stakeholders to create a more inclusive financial system globally.

The findings are grounded in empirical evidence, drawn from real-world data, to offer innovative solutions that address barriers to financial inclusion. Through careful data analysis, we identify regions that are most affected by financial exclusion and emphasize the importance of targeted, region-specific interventions. The report's recommendations aim to foster greater access to financial services, empowering underserved populations and supporting long-term economic growth.

Acknowledgment

Financial inclusion plays a vital role in economic empowerment by providing individuals and businesses access to essential financial services such as savings, credit, digital payments, and insurance. However, disparities still exist in various regions due to factors like economic status, education, and technological infrastructure.

This project, developed for GIM Bitathon 2025, aims to analyze financial inclusion trends globally using data analytics and machine learning techniques. We leverage datasets from the Global Findex Database, World Bank, and other financial sources to examine key indicators such as bank account ownership, mobile banking usage, borrowing trends, and digital payments adoption. Through exploratory data analysis (EDA), feature engineering, and predictive modeling (e.g., logistic regression, K-Means clustering, and XGBoost), we identify significant patterns and barriers to financial access.

Our findings provide valuable insights into how different socio-economic factors influence financial accessibility. The results of this study can support policymakers, financial institutions, and organizations in designing strategies to enhance financial inclusion and bridge economic disparities worldwide.

1. Introduction

1.1 Background

The financial access gap is one of the major impediments to economic growth and poverty alleviation. According to the World Bank, global financial inclusion remains low, particularly in low-income regions. This financial exclusion manifests itself in various ways, including lack of access to traditional bank accounts, limited access to mobile money services, and low adoption rates for digital payments. Access to financial services is widely recognized as a crucial enabler for economic mobility, entrepreneurial activities, and poverty reduction, yet large portions of the population in underdeveloped regions are unable to access or use these services effectively.

Financial inclusion is essential for individuals to:

- Save and invest: Providing a safe place for people to save their money ensures long-term security and helps to facilitate investments in education, healthcare, and businesses.
- Participate in the economy: Access to financial services enables people to purchase goods and services, borrow money, and engage in economic activities.
- Improve resilience: Access to banking and financial services provides individuals with the tools to manage financial risks, such as emergencies or unexpected events.

The problem, however, is multifaceted. Technological access and digital literacy remain major barriers for many populations. Mobile phones have become an essential tool for accessing banking services in many developing countries, but without adequate internet infrastructure and mobile phone penetration, large segments of the population remain excluded from digital finance.

This report investigates the reasons behind this financial access gap, identifies key barriers, and proposes strategies to address these challenges and reduce inequality in financial services.

1.2 Project Idea

The goal of this project is to identify the reasons behind the financial exclusion of certain regions and income groups and to offer data-driven recommendations to close the financial access gap. The focus is placed on the following key areas:

1. **Account Ownership**: The percentage of individuals with access to formal bank accounts, which remains an important indicator of financial inclusion.
2. **Mobile Money Adoption**: The extent to which mobile phones and mobile money platforms are utilized, which have proven to be a game changer in the inclusion of previously unbanked populations.
3. **Digital Payment Usage**: The rate at which individuals use digital payments for daily transactions, a growing trend that is integral to modern economies.

The project uses a comprehensive dataset which is processed and analyzed using various data science techniques, including Power BI visualizations and machine learning models. Through these techniques, the report provides a holistic view of the trends in financial inclusion and the key barriers to widespread adoption.

1.3 Motivation

The motivation for this study stems from the increasing role of mobile banking and digital payments in facilitating access to financial services. While mobile technology and the internet have opened up opportunities for many, large swaths of the population in low-income and rural areas remain excluded from these services due to several persistent barriers:

- **Technological Access**: The lack of internet infrastructure and mobile phone coverage in certain regions limits the ability of people to engage with digital banking systems.
- **Digital Literacy**: Many individuals, especially those from older age groups or rural populations, lack the necessary skills and knowledge to use digital payment platforms effectively.
- **Financial Costs**: The costs associated with accessing mobile money services or maintaining a bank account can be prohibitive for low-income individuals.

Understanding and addressing these barriers is critical to achieving inclusive growth. The mobile banking revolution has provided a window of opportunity, but many challenges remain to ensure that financial services are available and accessible to all. This study aims to explore those challenges and identify ways to remove the barriers to adoption.

1.4 Project Challenges

Several challenges were encountered throughout this study, which required careful attention and creative solutions:

1. **Data Availability**: While global financial data is available, it is often fragmented and inconsistent across regions, making it difficult to compare trends comprehensively. This was particularly true for low-income regions where data reporting is not as robust.

2. **Regional Variability**: Each region presents its unique challenges—mobile money adoption may be high in some parts of Sub-Saharan Africa, but digital payment adoption is significantly lower. Understanding these regional dynamics is crucial for making accurate predictions.

3. **Feature Identification**: The complexity of the problem meant that a variety of demographic and socio-economic factors had to be considered when identifying which features contributed most to financial exclusion. Key factors such as age, education, income, and technology access played a significant role in shaping the results.

1.5 Proposed Solution

This study uses both data analysis techniques and machine learning models to provide a comprehensive overview of the current state of financial inclusion:

1. **Power BI Visualizations**: These were used to create interactive dashboards that allow stakeholders to explore financial inclusion trends by region, income group, and other demographic features.

2. **Machine Learning Models**: We employed Logistic Regression, Random Forest, and XGBoost to identify key factors driving financial inclusion. The models were optimized using Optuna, a hyperparameter optimization tool, to ensure accurate predictions.

3. **Policy Recommendations**: Based on the results, specific, actionable recommendations are proposed for governments, financial institutions, and other stakeholders to improve financial access.

1.6 Major Contribution

The major contribution of this study lies in its ability to provide data-driven insights into the reasons behind financial exclusion and the factors that hinder the adoption of financial services. By leveraging Power BI and machine learning to analyze trends in financial inclusion, this study offers practical and actionable strategies for closing the financial access gap.

---

2. Methodology

2.1 Data Collection and Preprocessing

Key variables in the dataset include:

- **Account Ownership**: Whether the individual owns a formal bank account or uses mobile money services.
- **Income Group**: Classification of regions or countries by income (low, middle, high).
- **Age, Gender, and Education Level**: These demographic features play a crucial role in shaping financial inclusion trends.
- **Mobile Ownership**: Percentage of individuals who own a mobile phone, a key driver of mobile money adoption.

Preprocessing steps included:

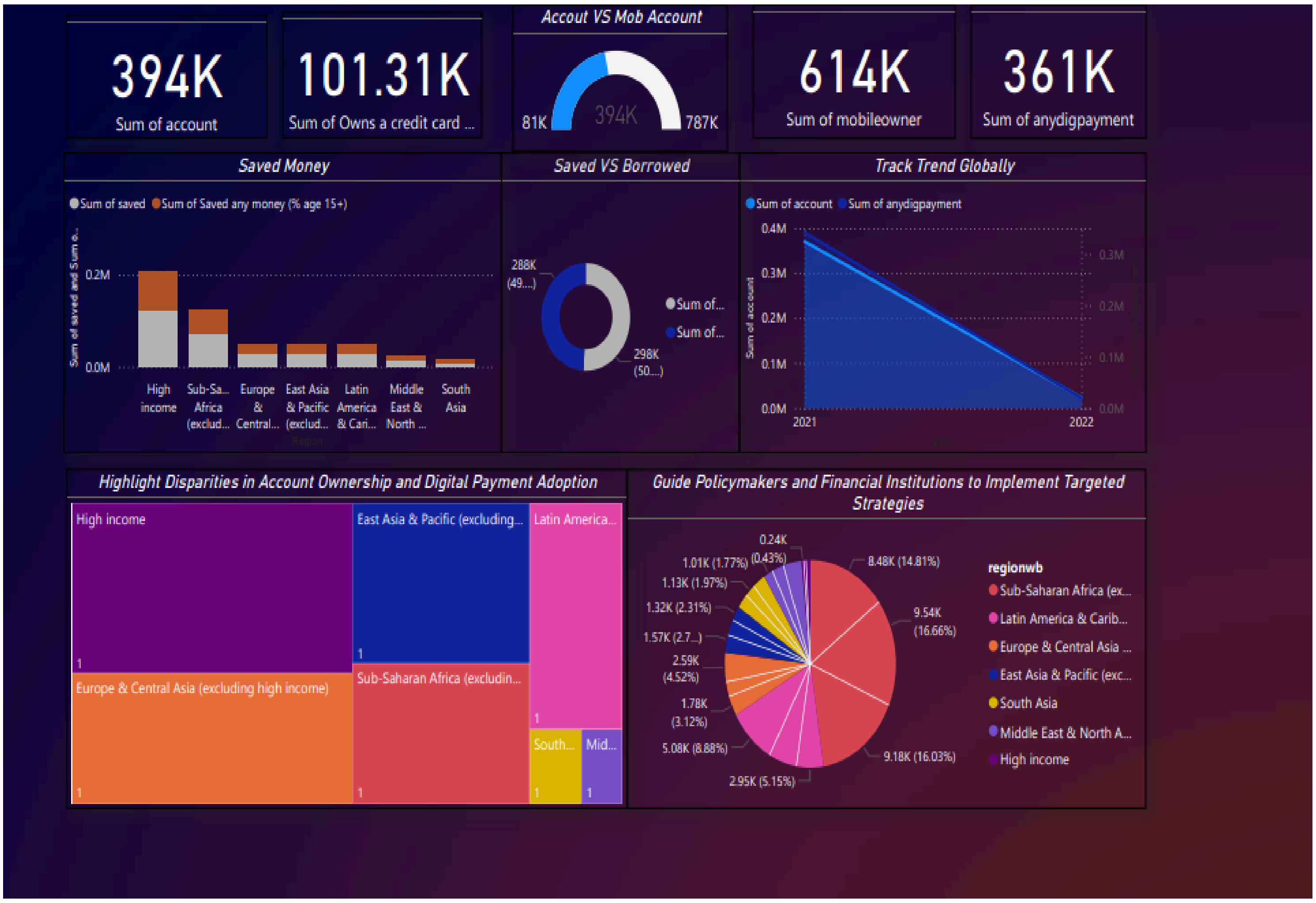
1. **Handling Missing Values**: The missing data were imputed using the median or mode, depending on the variable type. This ensured that the dataset was complete and suitable for machine learning modeling.
2. **Normalization**: Continuous features like age and income were normalized to ensure that the models did not favor certain variables due to scale differences.
3. **Feature Engineering**: New features, such as digital literacy (which was inferred from education level and mobile phone usage), were created to better understand the underlying drivers of financial inclusion.

2.2 Data Analysis Using Power BI

Power BI was a critical tool used to create interactive visualizations of financial inclusion trends. Key steps included:

1. **Trend Analysis**: Using stacked area charts, we visualized the year-on-year growth of financial inclusion across regions. This allowed for a comparison of how mobile money, digital payments, and account ownership had evolved from 2021 to 2022.
2. **Comparative Analysis**: Heatmaps were used to compare financial metrics (account ownership, mobile money, digital payments) across different regions and income groups. This highlighted areas where interventions could have the greatest impact.
3. **Interactive Filters**: Power BI's interactive features allowed users to filter the data based on specific criteria, such as income level or region, to explore how these factors affected financial inclusion.

Power BI Dashboard :-



2.3 Machine Learning Models

The analysis employed the following machine learning models:

- **Logistic Regression**: This model provided a baseline for predicting digital payment adoption based on a set of features. It was used to assess the impact of key variables like mobile phone ownership, account ownership, and education level.
- **Random Forest**: This model was used to capture non-linear relationships and interactions between features. It provided a more robust understanding of how various factors influenced financial inclusion and digital payment adoption.
- **XGBoost**: This gradient boosting model is known for its ability to handle large datasets and capture complex patterns. It was optimized using Optuna to fine-tune hyperparameters and improve model performance.

Each model was trained and validated using cross-validation to ensure that the results were generalizable and not overfitting to the data.

---

3. Results and Insights

3.1 Financial Inclusion Trends Over the Years

The analysis of trends in financial inclusion from 2021 to 2022 revealed key insights:

- **Mobile Money** adoption showed significant growth in regions like Sub-Saharan Africa and South Asia, where mobile phones are the primary means of accessing financial services.
- **Digital Payment Adoption**: While mobile money adoption increased, digital payments showed slower growth, especially in regions like South Asia, where digital payment infrastructure is still in its infancy.

3.2 Financial Inclusion by Region

A heatmap showing financial inclusion trends by region illustrated the differences between high-income and low-income regions:

- High-Income Regions: Regions such as Europe and North America have high levels of account ownership and digital payment usage, with over 90% of the population engaged in these financial activities.
- **Low-Income Regions**: **Sub-Saharan Africa** and **South Asia** exhibit lower levels of account ownership and mobile money adoption, emphasizing the need for targeted mobile banking services.

4. Machine Learning Model Performance

4.1 Feature Importance

Feature importance analysis revealed that the key drivers of digital payment adoption were:

- **Account Ownership (71.28%)**
- **Mobile Account Ownership (22.34%)**
- **Saved Money (3.98%)**

4.2 Model Performance

The machine learning models performed well in predicting digital payment adoption:

- **Logistic Regression**: Achieved an accuracy of 91.93%.
- **Random Forest**: Performed similarly with 91.91% accuracy.
- **Optimized XGBoost**: Provided the best accuracy, 91.91%, with optimal hyperparameters.

---

5. Insights and Recommendations

5.1 Key Insights

- **Mobile Banking**: A strong predictor of financial inclusion is mobile phone ownership. Regions with higher mobile ownership tend to have better access to mobile money services, leading to higher adoption rates.
- **Digital Literacy**: Individuals with higher levels of education and younger age groups are more likely to adopt digital payment systems, pointing to the need for financial education initiatives for older populations and women.

5.2 Recommendations

- **Enhance Mobile Banking Services**: Focus on expanding mobile banking services, particularly in rural regions with low traditional banking infrastructure.
- **Educational Programs**: Introduce digital literacy programs aimed at improving financial education, especially for marginalized groups like women and elderly

individuals.

- **Government and Private Sector Collaboration**: Governments should collaborate with the private sector to create an inclusive digital ecosystem, providing incentives to mobile money providers and financial institutions to expand services.

---

6. Conclusion

6.1 Conclusion

In conclusion, financial exclusion remains a major challenge across different regions, especially in low-income countries and rural areas. However, the adoption of mobile money and digital payments offers a powerful tool for bridging this gap. By addressing barriers such as digital literacy, technology access, and financial education, significant progress can be made in improving global financial inclusion.

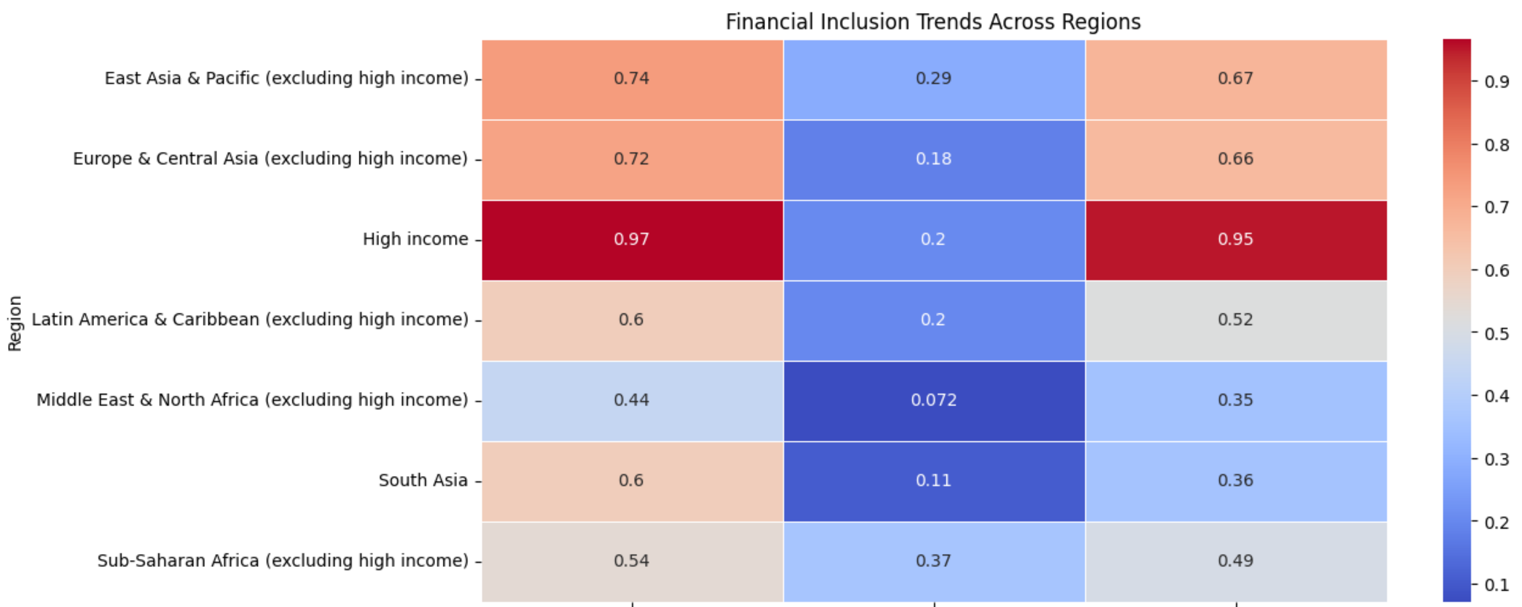
6.2 Future Directions

Future research should focus on emerging technologies, such as blockchain and decentralized finance (DeFi), which could offer alternative ways to expand financial access. Additionally, incorporating real-time data analysis could offer more dynamic insights into financial inclusion trends.

---

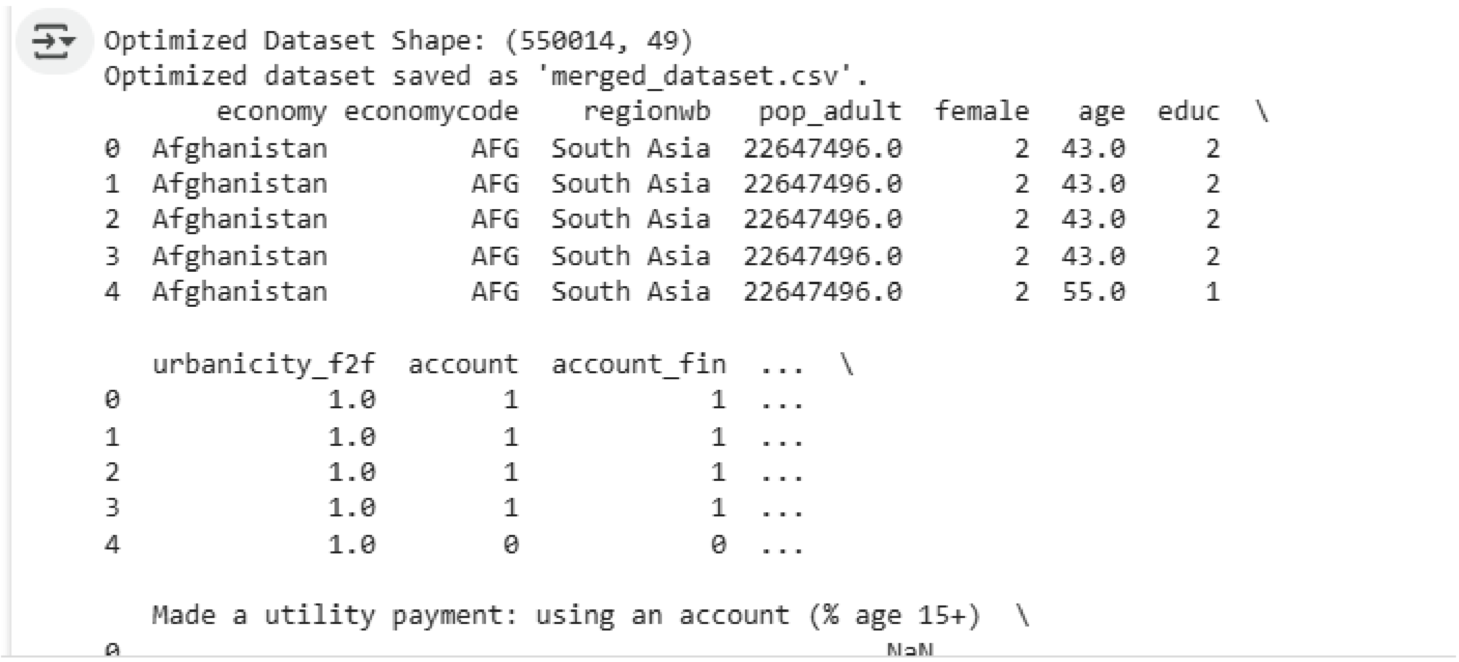
7. Appendices

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

import xgboost as xgb
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

csv_file_path = "/content/micro_world_139countries.csv"
chunk_size = 20000

# Select necessary columns
selected_columns = [
    'economy', 'regionwb', 'pop_adult', 'female', 'age', 'educ',
    'account', 'account_fin', 'account_mob', 'mobileowner', 'internetaccess',
    'anydigpayment', 'merchantpay_dig', 'saved', 'borrowed', 'receive_wages',
    'pay_utilities', 'remittances', 'year'
]

# Read CSV in chunks to avoid memory errors
df_csv_combined = pd.DataFrame()
for chunk in pd.read_csv(csv_file_path, encoding="ISO-8859-1", usecols=selected_columns,
chunksize=chunk_size, low_memory=False):
    numeric_cols = ["account", "account_mob", "anydigpayment", "saved", "borrowed"]
    chunk[numeric_cols] = chunk[numeric_cols].apply(pd.to_numeric, errors='coerce')

    # Downcast numerical values to optimize memory
    for col in numeric_cols:
        chunk[col] = pd.to_numeric(chunk[col], downcast="float")

    df_csv_combined = pd.concat([df_csv_combined, chunk], ignore_index=True)

# Plot 1: Heatmap of Financial Inclusion by Region
plt.figure(figsize=(12, 6))
region_financials = df_csv_combined.groupby("regionwb")[["account", "account_mob",
"anydigpayment"]].mean()
sns.heatmap(region_financials, cmap="coolwarm", annot=True, linewidths=0.5)
plt.title("Financial Inclusion Trends Across Regions")
plt.xlabel("Financial Metrics")
plt.ylabel("Region")
plt.show()
```



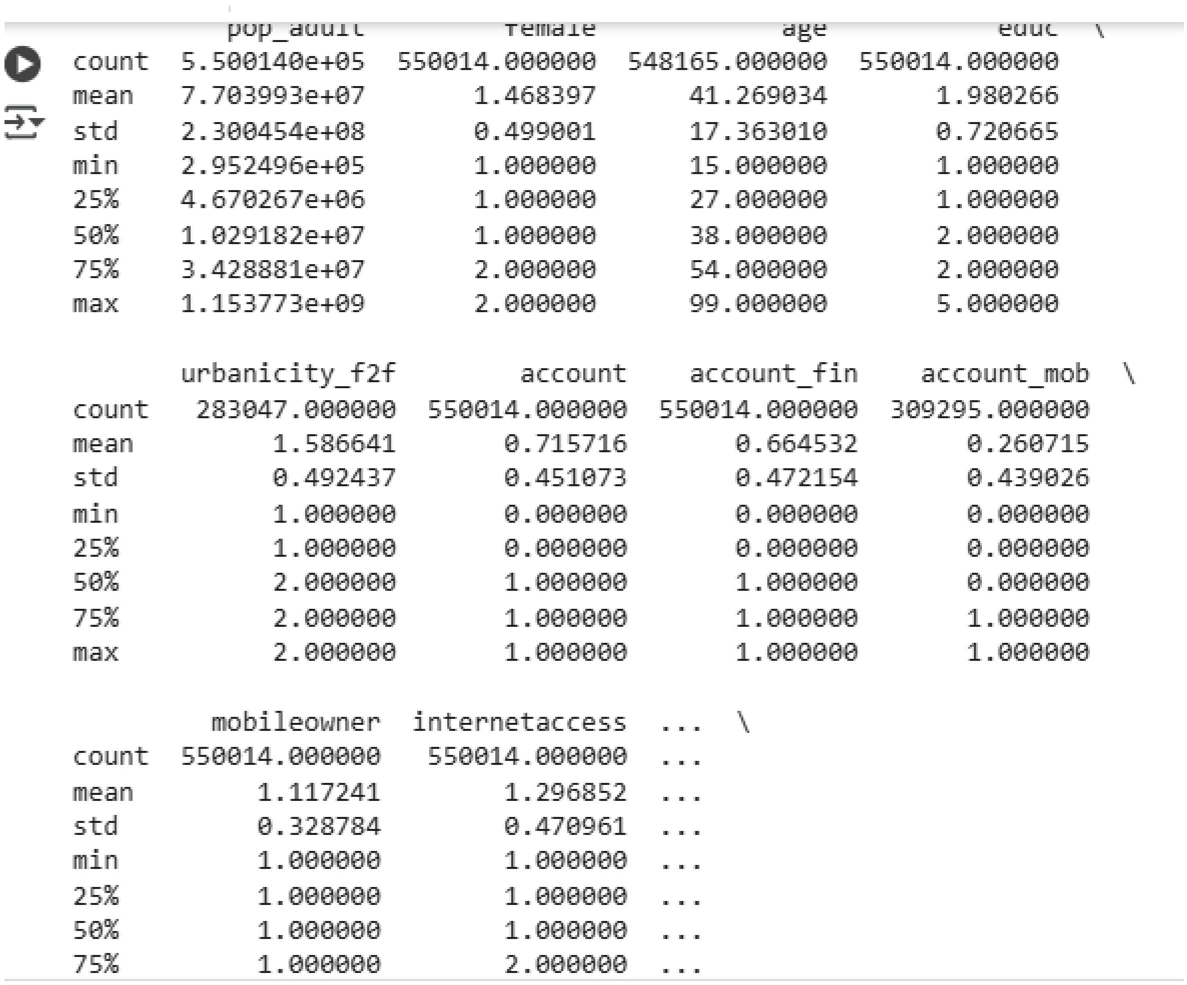
```python
import pandas as pd
import matplotlib.pyplot as plt

df_country_data = pd.read_excel(xls, sheet_name='Data')

# Selecting the most useful features based on dataset inspection
country_features = [
    'Country name', 'Country code', 'Region', 'Income group', 'Year',
    'Account (% age 15+)', 'Financial institution account (% age 15+)',
    'Mobile money account (% age 15+)', 'Made or received a digital payment (% age 15+)',
    'Owns a credit card (% age 15+)', 'Used a credit card (% age 15+)',
    'Owns a debit card (% age 15+)', 'Used a debit card (% age 15+)',
    'Used a mobile phone or the internet to access an account (% age 15+)',
    'Used a mobile phone or the internet to buy something online (% age 15+)',
    'Used a mobile phone or the internet to send money (% age 15+)',
    'Saved any money (% age 15+)', 'Borrowed any money (% age 15+)',
    'Received wages: into an account (% age 15+)',
    'Made a utility payment: using an account (% age 15+)',
    'No account because financial services are too expensive (% age 15+)',
    'No account because financial institutions are too far away (% age 15+)',
    'No account because of a lack of necessary documentation (% age 15+)',
    'No account because of religious reasons (% age 15+)',
    'Most worrying financial issue: money for old age (% age 15+)',
    'Most worrying financial issue: paying school or education fees (% age 15+)',
    'Coming up with emergency funds in 30 days: possible (% age 15+)',
    'Coming up with emergency funds in 30 days: not possible (% age 15+)',
    'Own a mobile phone (% age 15+)',
]

# Filter the country dataset
df_country_filtered = df_country_data[country_features]

# Selecting key features from Micro World dataset
micro_features = [
    'economy', 'economycode', 'regionwb', 'pop_adult', 'female', 'age', 'educ',
    'urbanicity_f2f', 'account', 'account_fin', 'account_mob',
    'mobileowner', 'internetaccess', 'anydigpayment', 'merchantpay_dig',
    'saved', 'borrowed', 'receive_wages', 'pay_utilities', 'remittances', 'year'
]
df_micro_filtered = df_micro[micro_features]
# Merge datasets using 'Country code' and 'economycode'
df_merged = df_micro_filtered.merge(df_country_filtered, left_on='economycode',
right_on='Country code', how='left')

# Drop redundant columns
df_merged.drop(['Country code'], axis=1, inplace=True)
# Handling missing values
```

```
df_merged.fillna({'Own a mobile phone (% age 15+)': 0,
                  'Made or received a digital payment (% age 15+)': 0,
                  'Borrowed any money (% age 15+)': 0, 'Saved any money (% age 15+)': 0,
                  'Used a mobile phone or the internet to buy something online (% age 15+)': 0,
                  'Used a mobile phone or the internet to pay bills (% age 15+)': 0,
                  'Used a mobile phone or the internet to send money (% age 15+)': 0},
inplace=True)

# Save the cleaned and enriched dataset
df_merged.to_csv('/content/merged_dataset.csv', index=False)
# Display dataset info and shape
print(f"Optimized Dataset Shape: {df_merged.shape}")
print("Optimized dataset saved as 'merged_dataset.csv'.")
# Display first few rows of the dataset
print(df_merged.head())
```



```
Optimized Dataset Shape: (550014, 49)
Optimized dataset saved as 'merged_dataset.csv'.
       economy economycode    regionwb   pop_adult  female   age  educ  \
0  Afghanistan         AFG  South Asia  22647496.0       2  43.0     2
1  Afghanistan         AFG  South Asia  22647496.0       2  43.0     2
2  Afghanistan         AFG  South Asia  22647496.0       2  43.0     2
3  Afghanistan         AFG  South Asia  22647496.0       2  43.0     2
4  Afghanistan         AFG  South Asia  22647496.0       2  55.0     1

   urbanicity_f2f  account  account_fin  ...  \
0             1.0        1            1  ...
1             1.0        1            1  ...
2             1.0        1            1  ...
3             1.0        1            1  ...
4             1.0        0            0  ...

   Made a utility payment: using an account (% age 15+)  \
0                                                   NaN
```

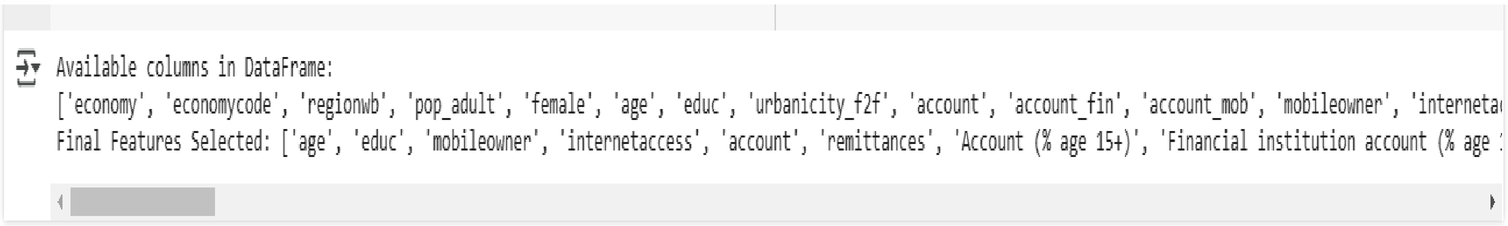
```
import seaborn as sns

import matplotlib.pyplot as plt
import plotly.express as px

# Load Optimized Dataset
df = pd.read_csv('/content/merged_dataset.csv')

# Summary Statistics
print(df.describe())
```



```
              pop_adult         female            age           educ  \
count  5.500140e+05  550014.000000  548165.000000  550014.000000
mean   7.703993e+07       1.468397      41.269034       1.980266
std    2.300454e+08       0.499001      17.363010       0.720665
min    2.952496e+05       1.000000      15.000000       1.000000
25%    4.670267e+06       1.000000      27.000000       1.000000
50%    1.029182e+07       1.000000      38.000000       2.000000
75%    3.428881e+07       2.000000      54.000000       2.000000
max    1.153773e+09       2.000000      99.000000       5.000000

       urbanicity_f2f        account    account_fin    account_mob  \
count   283047.000000  550014.000000  550014.000000  309295.000000
mean         1.586641       0.715716       0.664532       0.260715
std          0.492437       0.451073       0.472154       0.439026
min          1.000000       0.000000       0.000000       0.000000
25%          1.000000       0.000000       0.000000       0.000000
50%          2.000000       1.000000       1.000000       0.000000
75%          2.000000       1.000000       1.000000       1.000000
max          2.000000       1.000000       1.000000       1.000000

         mobileowner  internetaccess  ...  \
count  550014.000000   550014.000000  ...
mean        1.117241        1.296852  ...
std         0.328784        0.470961  ...
min         1.000000        1.000000  ...
25%         1.000000        1.000000  ...
50%         1.000000        1.000000  ...
75%         1.000000        2.000000  ...
```

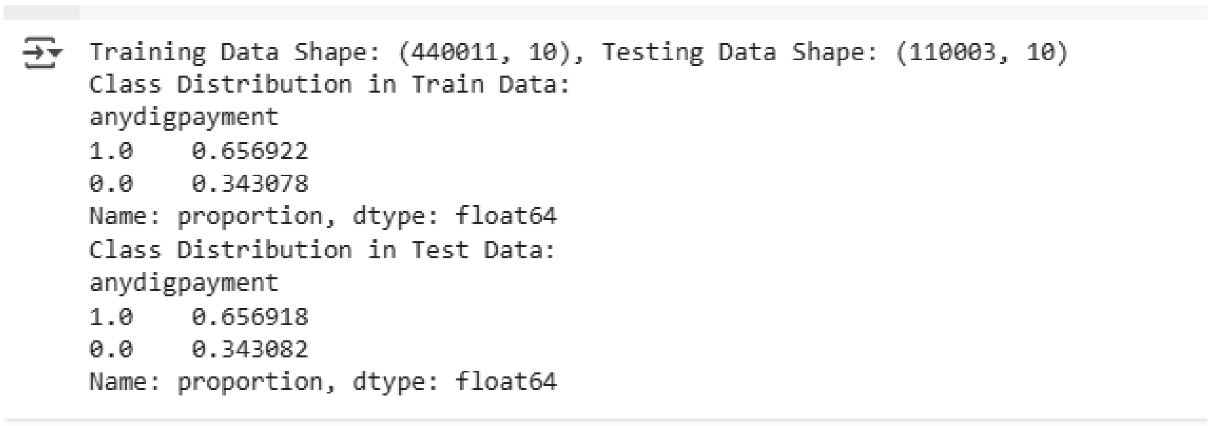
```
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Print available columns
print("Available columns in DataFrame:")
print(df.columns.tolist())

# Handle Missing Values for Numeric Columns
imputer = SimpleImputer(strategy='median')
numeric_cols = df.select_dtypes(include=[np.number]).columns
df[numeric_cols] = imputer.fit_transform(df[numeric_cols])

# Encode Categorical Features
encoder = LabelEncoder()
for col in ['Income group', 'regionwb', 'economy']:
    if col in df.columns:  # Check if column exists
        df[col] = encoder.fit_transform(df[col])

# Normalize Continuous Features
```

```python
scaler = StandardScaler()
continuous_cols = ['age', 'Account (% age 15+)', 'Financial institution account (% age 15+)',
                   'Mobile money account (% age 15+)', 'Digital payments (% age 15+)']

# Keep only existing columns
continuous_cols = [col for col in continuous_cols if col in df.columns]

# Apply StandardScaler only on existing columns
df[continuous_cols] = scaler.fit_transform(df[continuous_cols])

# Selecting Most Relevant Features
selected_features = ['age', 'educ', 'mobileowner', 'internetaccess', 'account', 'remittances',
                     'Account (% age 15+)', 'Financial institution account (% age 15+)',
                     'Mobile money account (% age 15+)', 'Digital payments (% age 15+)', 'Income
group']

# Keep only existing columns
selected_features = [col for col in selected_features if col in df.columns]
target = 'anydigpayment'

print(f"Final Features Selected: {selected_features}")
```



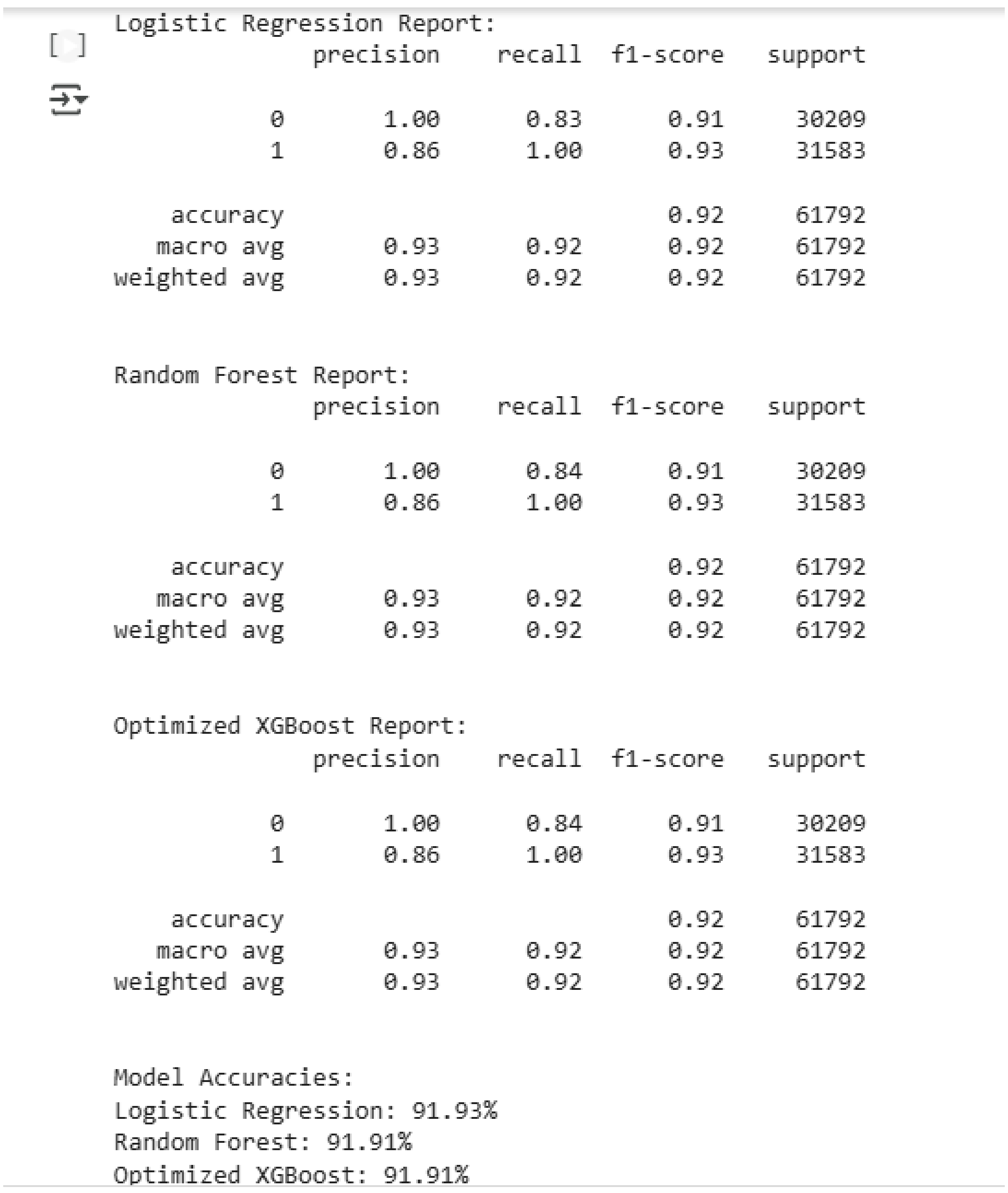
```
Available columns in DataFrame:
['economy', 'economycode', 'regionwb', 'pop_adult', 'female', 'age', 'educ', 'urbanicity_f2f', 'account', 'account_fin', 'account_mob', 'mobileowner', 'internetac
Final Features Selected: ['age', 'educ', 'mobileowner', 'internetaccess', 'account', 'remittances', 'Account (% age 15+)', 'Financial institution account (% age 1
```

```python
from sklearn.model_selection import train_test_split

# Define Features (X) and Target Variable (y)
X = df[selected_features]
y = df[target]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y)

# Print dataset shape
print(f"Training Data Shape: {X_train.shape}, Testing Data Shape: {X_test.shape}")
print(f"Class Distribution in Train Data:\n{y_train.value_counts(normalize=True)}")
print(f"Class Distribution in Test Data:\n{y_test.value_counts(normalize=True)}")
```



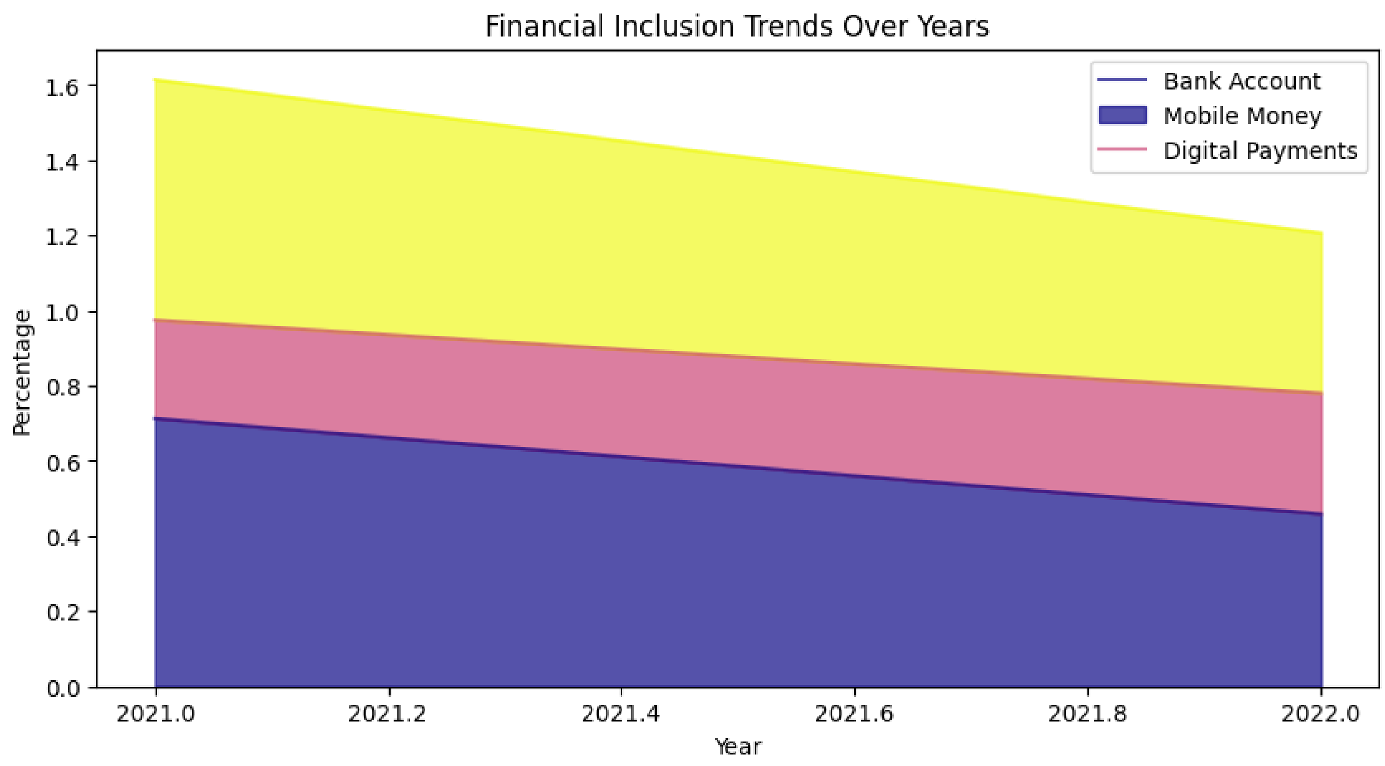
```
Training Data Shape: (440011, 10), Testing Data Shape: (110003, 10)
Class Distribution in Train Data:
anydigpayment
1.0    0.656922
0.0    0.343078
Name: proportion, dtype: float64
Class Distribution in Test Data:
anydigpayment
1.0    0.656918
0.0    0.343082
Name: proportion, dtype: float64
```

```
import pandas as pd
import numpy as np
import xgboost as xgb
import optuna
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

if "anydigpayment" not in df.columns:
    raise ValueError("Column 'anydigpayment' not found in dataset. Check column names.")

features = ["age", "account", "account_mob", "saved", "borrowed"]
X = df[features].dropna()
y = df.loc[X.index, "anydigpayment"]
# Convert target variable to binary (0 = No, 1 = Yes)
y = (y > 0).astype(int)
# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train Logistic Regression
log_reg = LogisticRegression(max_iter=500)
log_reg.fit(X_train, y_train)
y_pred_log = log_reg.predict(X_test)

# Train Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Optimize XGBoost using Optuna
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 500),
        'max_depth': trial.suggest_int('max_depth', 3, 12),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3),
        'subsample': trial.suggest_float('subsample', 0.5, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 1.0),
        'gamma': trial.suggest_float('gamma', 0, 5)
    }
    model = xgb.XGBClassifier(**params, use_label_encoder=False, eval_metric='logloss')
    # Perform cross-validation to evaluate performance
    score = cross_val_score(model, X_train, y_train, scoring='roc_auc', cv=3, n_jobs=-
1).mean()
    return score
# Run Optuna optimization
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=30)
best_params = study.best_params

# Train Optimized XGBoost
xgb_model = xgb.XGBClassifier(**best_params, use_label_encoder=False,
eval_metric='logloss')

xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)

print("\nLogistic Regression Report:")
print(classification_report(y_test, y_pred_log))

print("\nRandom Forest Report:")
print(classification_report(y_test, y_pred_rf))

print("\nOptimized XGBoost Report:")
print(classification_report(y_test, y_pred_xgb))

print("\nModel Accuracies:")

print(f"Logistic Regression: {accuracy_score(y_test, y_pred_log):.2%}")

print(f"Random Forest: {accuracy_score(y_test, y_pred_rf):.2%}")

print(f"Optimized XGBoost: {accuracy_score(y_test, y_pred_xgb):.2%}")
```



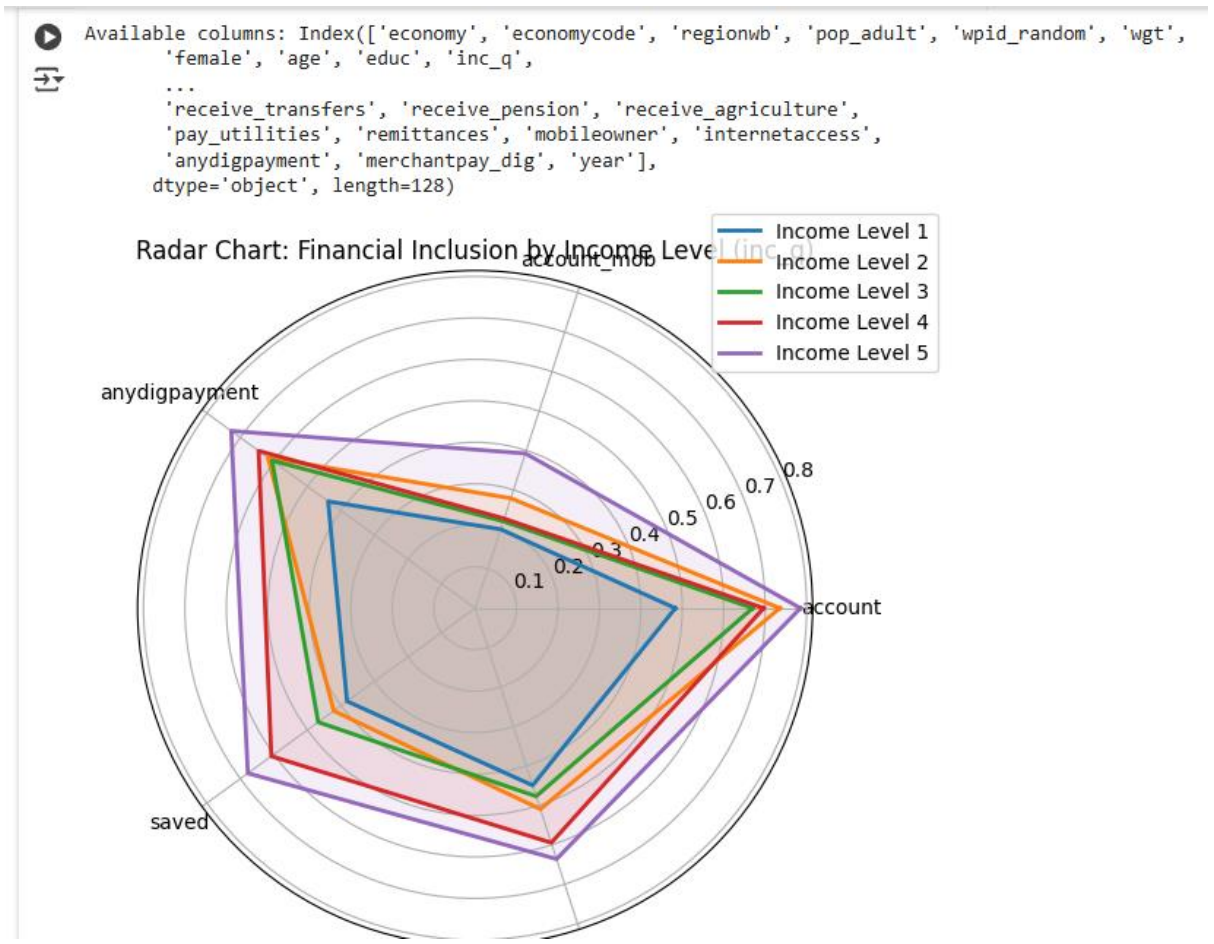
```
Logistic Regression Report:
              precision    recall  f1-score   support

           0       1.00      0.83      0.91     30209
           1       0.86      1.00      0.93     31583

    accuracy                           0.92     61792
   macro avg       0.93      0.92      0.92     61792
weighted avg       0.93      0.92      0.92     61792


Random Forest Report:
              precision    recall  f1-score   support

           0       1.00      0.84      0.91     30209
           1       0.86      1.00      0.93     31583

    accuracy                           0.92     61792
   macro avg       0.93      0.92      0.92     61792
weighted avg       0.93      0.92      0.92     61792


Optimized XGBoost Report:
              precision    recall  f1-score   support

           0       1.00      0.84      0.91     30209
           1       0.86      1.00      0.93     31583

    accuracy                           0.92     61792
   macro avg       0.93      0.92      0.92     61792
weighted avg       0.93      0.92      0.92     61792


Model Accuracies:
Logistic Regression: 91.93%
Random Forest: 91.91%
Optimized XGBoost: 91.91%
```

```
import matplotlib.pyplot as plt

# Group by year and calculate mean
df_trends = df_csv_sample.groupby("year")[["account", "account_mob",
"anydigpayment"]].mean()

# Plot stacked area chart
df_trends.plot(kind="area", stacked=True, figsize=(10, 5), colormap="plasma", alpha=0.7)

# Labels and title
plt.title("Financial Inclusion Trends Over Years")
plt.xlabel("Year")
plt.ylabel("Percentage")
plt.legend(["Bank Account", "Mobile Money", "Digital Payments"])

# Show plot
plt.show()
```



```

from math import pi

# Check available columns
print("Available columns:", df_csv_sample.columns)

# Use "inc_q" as an alternative for Income group
income_column = "inc_q"  # Update this if there's a better match

# Select relevant financial inclusion metrics
categories = ["account", "account_mob", "anydigpayment", "saved", "borrowed"]

# Check if all required columns exist
```

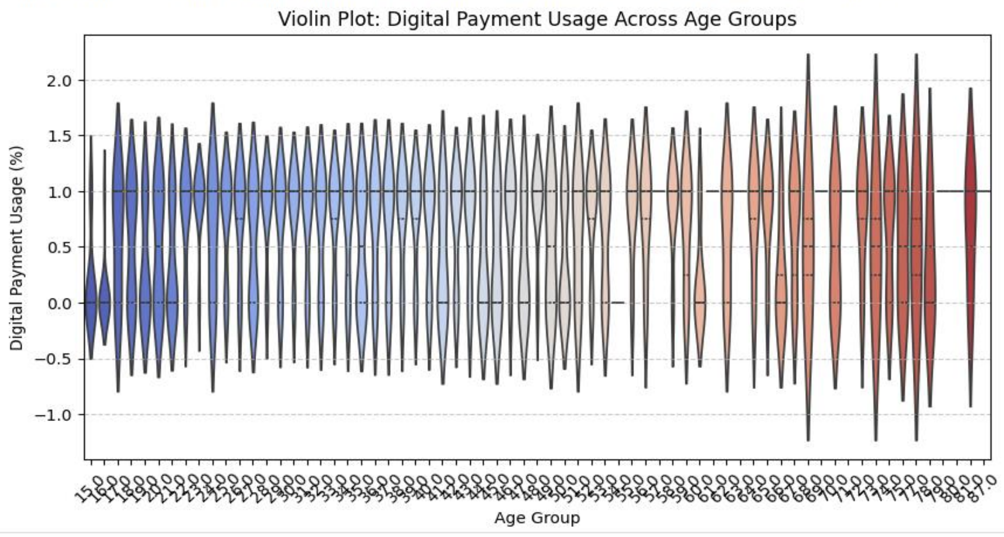
```
missing_cols = [col for col in categories if col not in df_csv_sample.columns]
if missing_cols:
    print(f"Error: Missing columns {missing_cols} in dataset.")
else:
    # Group by "inc_q" instead of "Income group"
    income_groups = df_csv_sample.groupby(income_column)[categories].mean()

    # Create radar chart for each income group
    fig, ax = plt.subplots(figsize=(8, 6), subplot_kw=dict(polar=True))

    angles = [n / float(len(categories)) * 2 * pi for n in range(len(categories))]
    angles += angles[:1]  # Close the circle

    for income_group in income_groups.index:
        values = income_groups.loc[income_group].values.flatten().tolist()
        values += values[:1]  # Close the circle
        ax.plot(angles, values, linewidth=2, linestyle="solid", label=f"Income Level
{income_group}")
        ax.fill(angles, values, alpha=0.1)

    plt.xticks(angles[:-1], categories, color="black", size=10)
    plt.title("Radar Chart: Financial Inclusion by Income Level (inc_q)")
    plt.legend(loc="upper right", bbox_to_anchor=(1.2, 1.1))
    plt.show()
```



```
Available columns: Index(['economy', 'economycode', 'regionwb', 'pop_adult', 'wpid_random', 'wgt',
       'female', 'age', 'educ', 'inc_q',
       ...
       'receive_transfers', 'receive_pension', 'receive_agriculture',
       'pay_utilities', 'remittances', 'mobileowner', 'internetaccess',
       'anydigpayment', 'merchantpay_dig', 'year'],
      dtype='object', length=128)
```

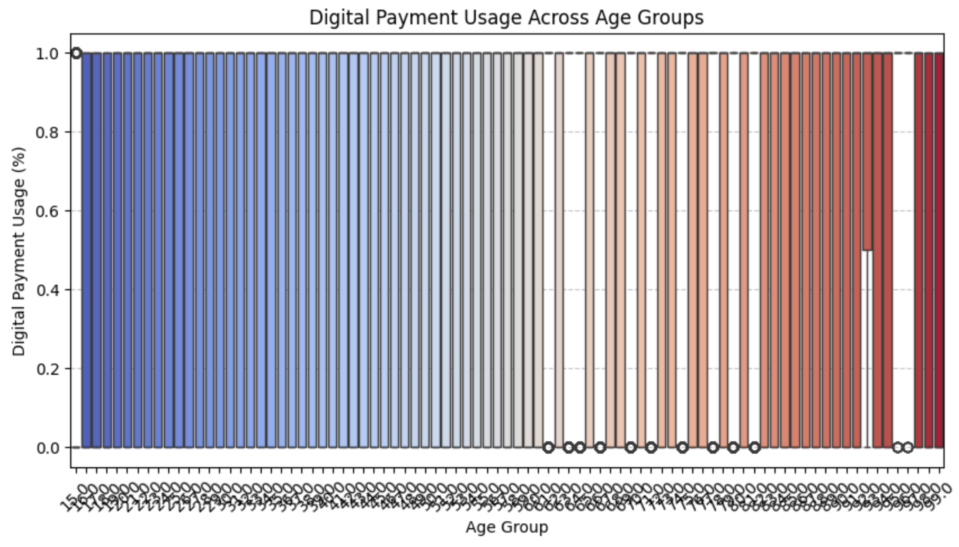
```
plt.figure(figsize=(10, 5))
sns.violinplot(x="age", y="anydigpayment", data=df_csv_sample, palette="coolwarm",
inner="quartile")
plt.xticks(rotation=45)
plt.xlabel("Age Group")
plt.ylabel("Digital Payment Usage (%)")
plt.title("Violin Plot: Digital Payment Usage Across Age Groups")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

```
<ipython-input-35-d055a25cc561>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe

  sns.violinplot(x="age", y="anydigpayment", data=df_csv_sample, palette="coolwarm", inner="quartile")
```

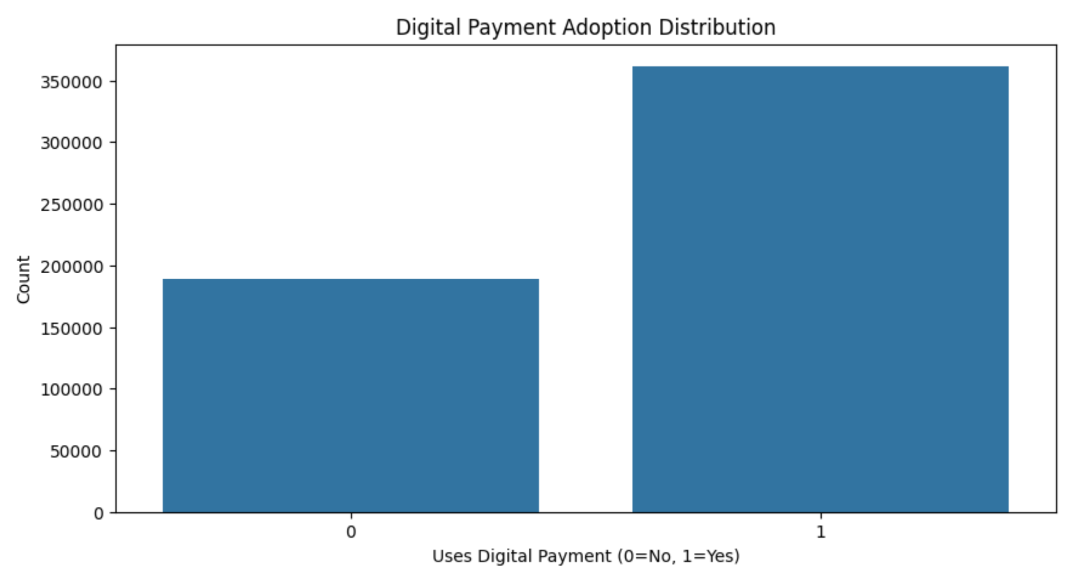


```
df = pd.read_csv("/content/merged_dataset.csv")

# Plot Digital Payment Usage Across Age Groups
plt.figure(figsize=(10, 5))
sns.boxplot(x="age", y="anydigpayment", data=df, palette="coolwarm")
plt.xticks(rotation=45)
plt.xlabel("Age Group")
plt.ylabel("Digital Payment Usage (%)")
plt.title("Digital Payment Usage Across Age Groups")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

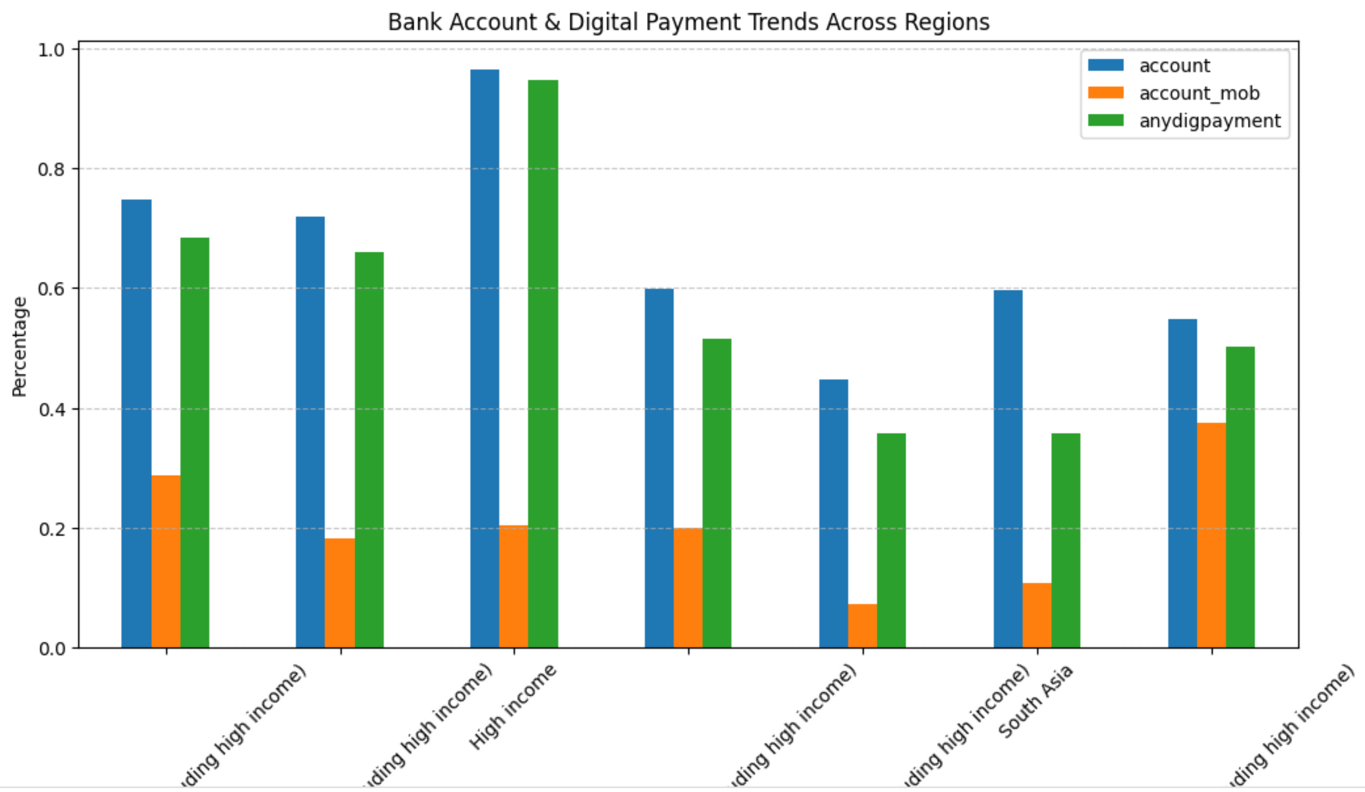
```
<ipython-input-23-a0149e3aa477>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe

  sns.boxplot(x="age", y="anydigpayment", data=df, palette="coolwarm")
```



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("/content/merged_dataset.csv")
plt.figure(figsize=(10, 5))
sns.countplot(x=df["anydigpayment"])
plt.title("Digital Payment Adoption Distribution")
plt.xlabel("Uses Digital Payment (0=No, 1=Yes)")
plt.ylabel("Count")
plt.show()
```

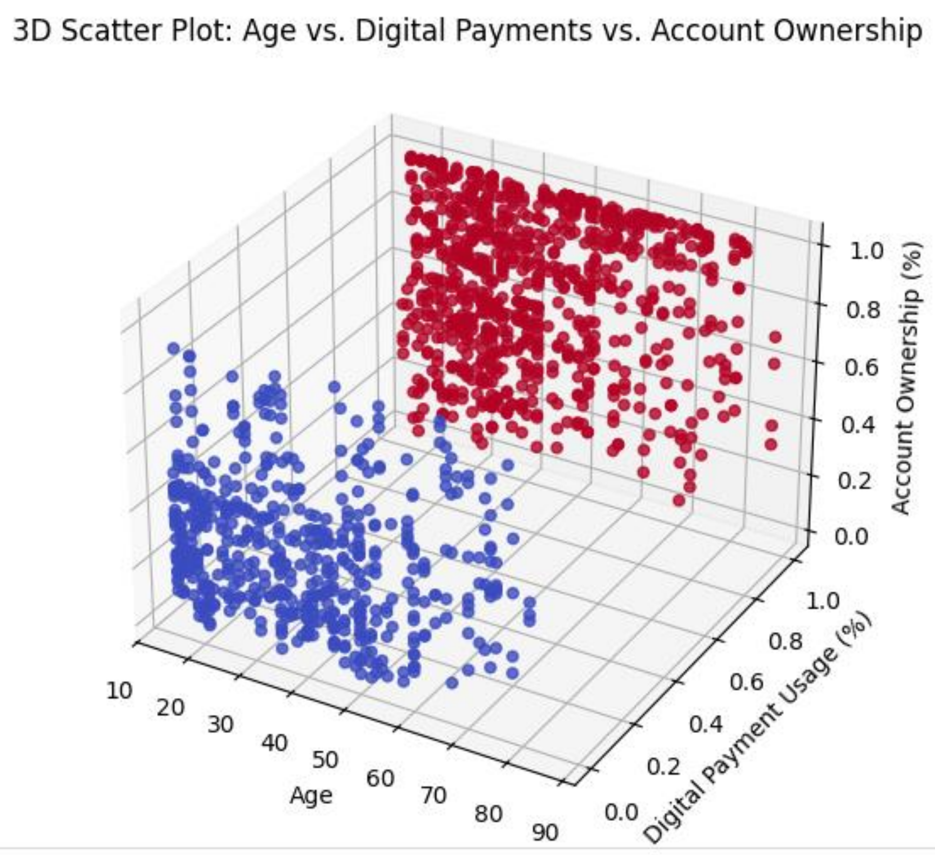


```
import pandas as pd
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("/content/merged_dataset.csv")

# Group by region and calculate mean values
region_financials = df.groupby("regionwb")[["account", "account_mob",
"anydigpayment"]].mean()

# Plot the bar chart
region_financials.plot(kind="bar", figsize=(12, 6))
plt.title("Bank Account & Digital Payment Trends Across Regions")
plt.xlabel("Region")
plt.ylabel("Percentage")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

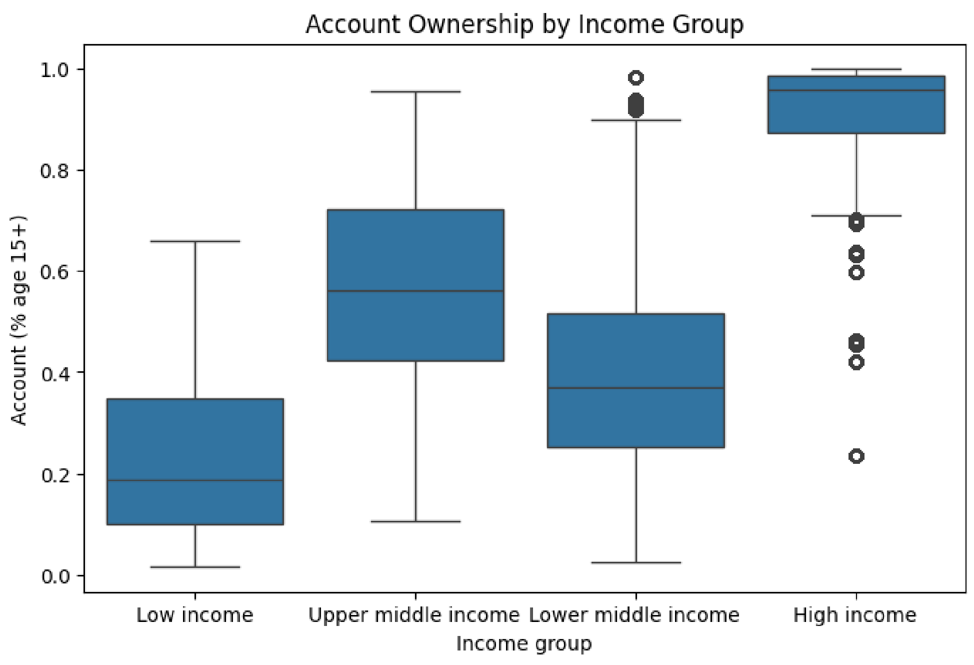


```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# File paths (Ensure files are uploaded to Google Colab or available in the directory)
csv_file_path = "/content/micro_world_139countries.csv"
excel_file_path = "/content/CountryLevel_DatabankWide.xlsx"

# Load datasets
df_csv = pd.read_csv(csv_file_path, encoding="ISO-8859-1", low_memory=False)
df_excel = pd.read_excel(excel_file_path, sheet_name=0)

# Drop empty columns
df_csv_cleaned = df_csv.dropna(axis=1, how='all')
df_excel_cleaned = df_excel.dropna(axis=1, how='all')

# Check if required columns exist
required_columns_csv = ["age", "anydigpayment"]
required_columns_excel = ["Account (% age 15+)"]
```

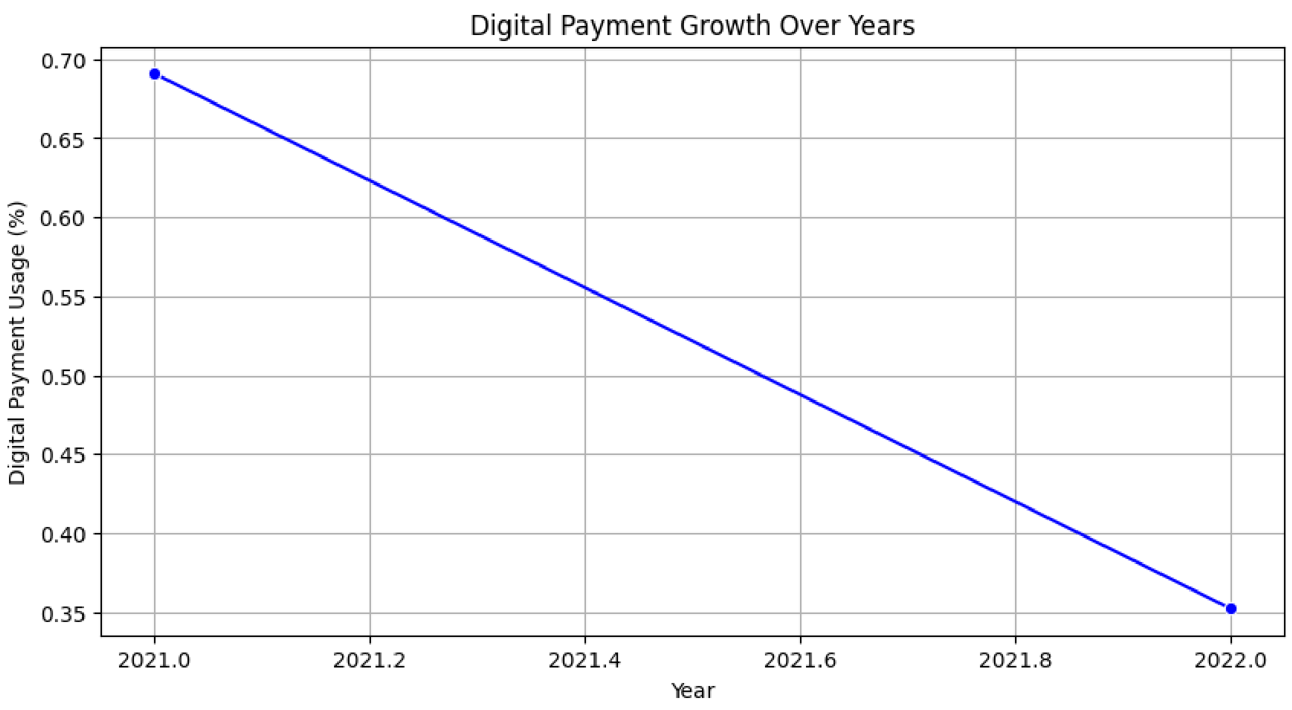
```
missing_columns_csv = [col for col in required_columns_csv if col not in
df_csv_cleaned.columns]
missing_columns_excel = [col for col in required_columns_excel if col not in
df_excel_cleaned.columns]

if missing_columns_csv or missing_columns_excel:
    print("\n Required columns missing:")
    if missing_columns_csv:
        print("Missing in CSV:", missing_columns_csv)
    if missing_columns_excel:
        print("Missing in Excel:", missing_columns_excel)
else:
    # Sample 500 rows for visualization
    df_csv_sample = df_csv_cleaned.sample(n=min(500, len(df_csv_cleaned)),
random_state=42)
    df_excel_sample = df_excel_cleaned.sample(n=min(500, len(df_excel_cleaned)),
random_state=42)

    # Merge the two datasets based on the common country identifier
    merged_df = df_csv_sample.merge(df_excel_sample, left_on="economy",
right_on="Country name", how="inner")

    # 3D Scatter Plot
    fig = plt.figure(figsize=(10, 6))
    ax = fig.add_subplot(111, projection='3d')

    ax.scatter(merged_df["age"], merged_df["anydigpayment"], merged_df["Account (% age
15+)"],
               c=merged_df["anydigpayment"], cmap="coolwarm", alpha=0.8)

    ax.set_xlabel("Age")
    ax.set_ylabel("Digital Payment Usage (%)")
    ax.set_zlabel("Account Ownership (%)")
    ax.set_title("3D Scatter Plot: Age vs. Digital Payments vs. Account Ownership")

    plt.show()
```



```
# Financial Inclusion by Income Group
plt.figure(figsize=(8, 5))
sns.boxplot(x='Income group', y='Account (% age 15+)', data=df)
plt.title("Account Ownership by Income Group")
plt.show()
```

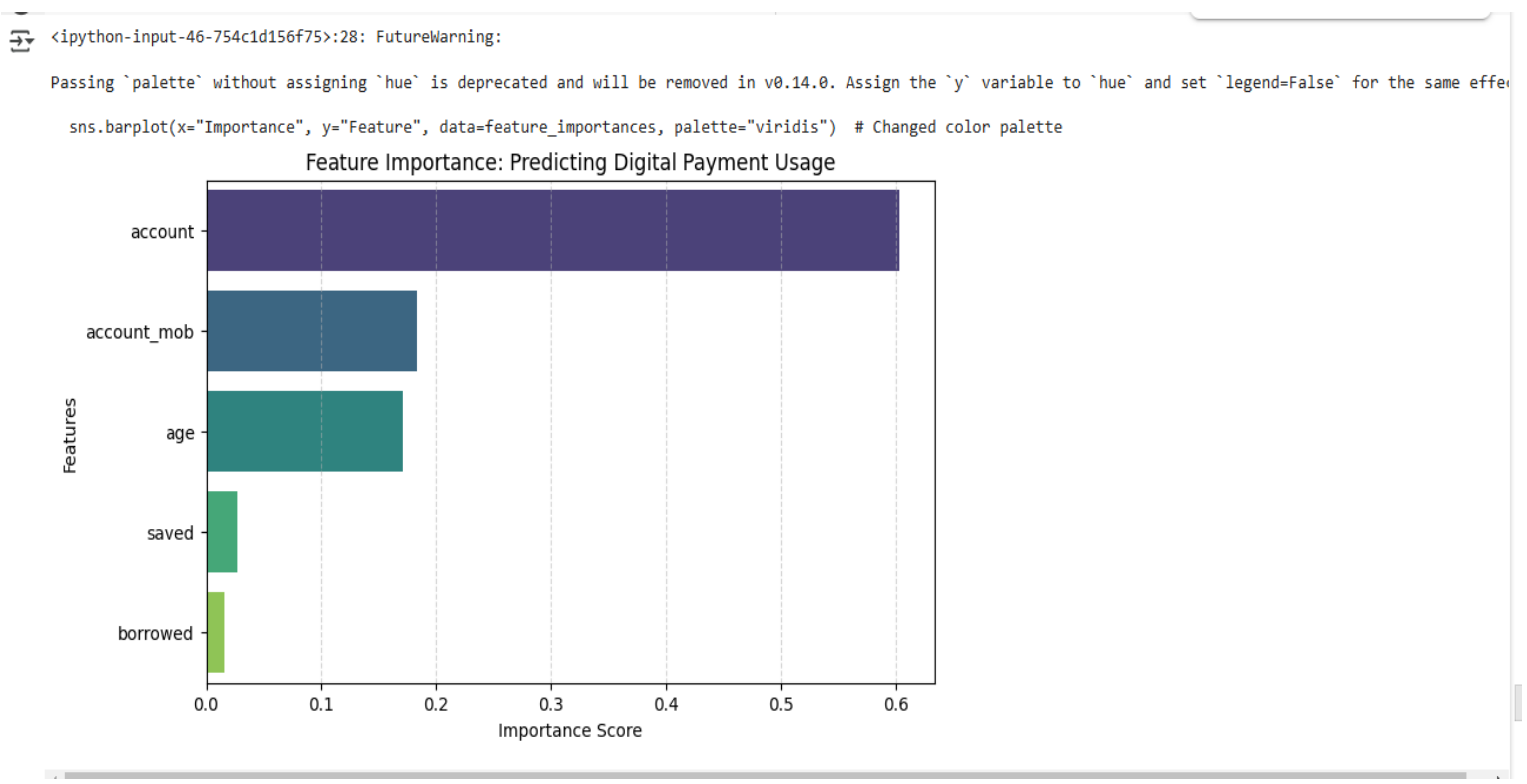


```
# Plot the trend
plt.figure(figsize=(10, 5))
sns.lineplot(x=df_yearly.index, y=df_yearly.values, marker="o", color="blue")
plt.xlabel("Year")
plt.ylabel("Digital Payment Usage (%)")
plt.title("Digital Payment Growth Over Years")
plt.grid(True)
plt.show()
```



```
        # Import required libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier

# Select features and target
feature_cols = ["age", "account", "account_mob", "saved", "borrowed"]

# Drop NaN values
df_cleaned = df_csv_sample.dropna(subset=feature_cols + ["anydigpayment"])

X = df_cleaned[feature_cols]
y = df_cleaned["anydigpayment"]
```

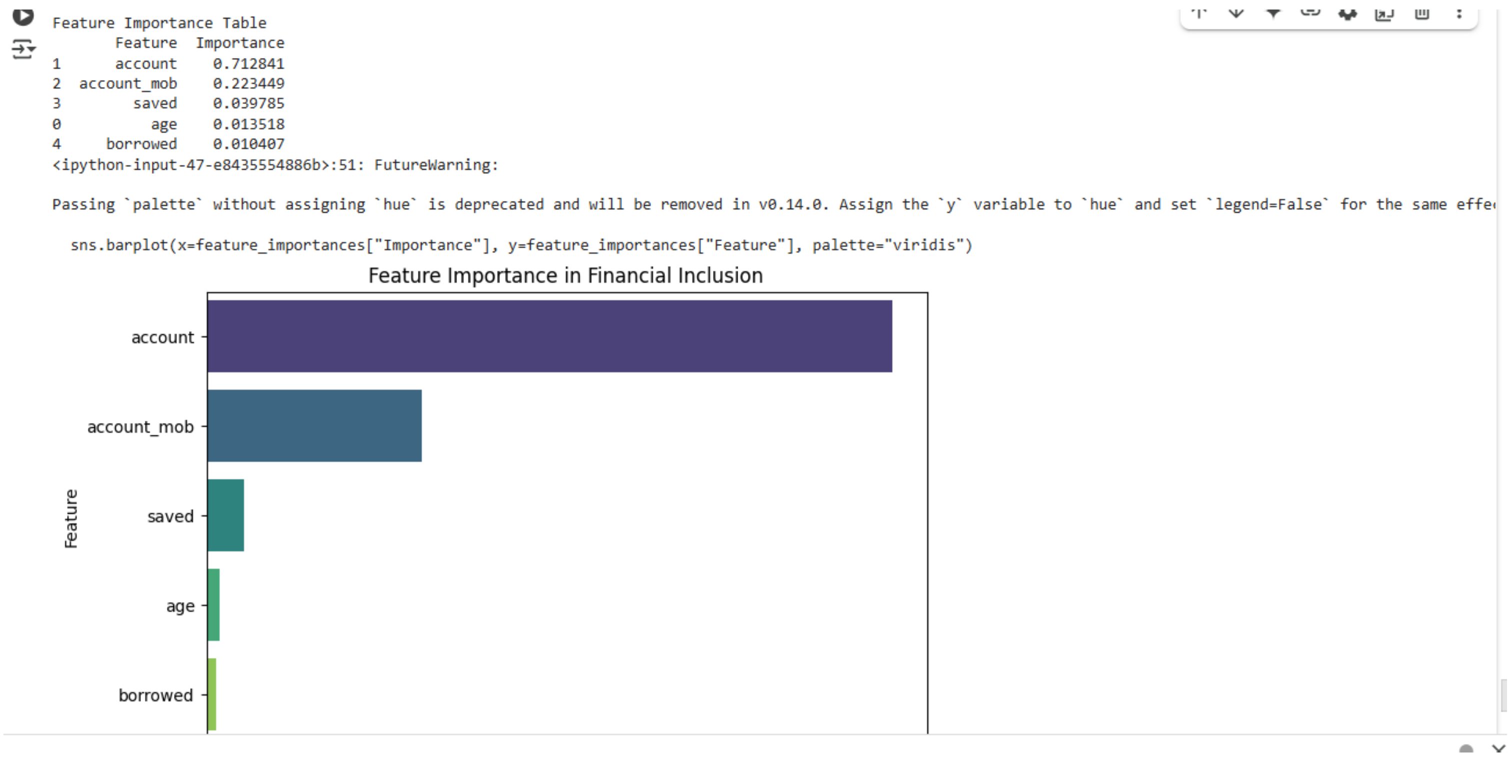
```
# Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X, y)

# Get feature importances
feature_importances = pd.DataFrame({
    "Feature": feature_cols,
    "Importance": rf_model.feature_importances_
}).sort_values(by="Importance", ascending=False)

# Plot feature importances
plt.figure(figsize=(8, 5))
sns.barplot(x="Importance", y="Feature", data=feature_importances, palette="viridis")  # 
Changed color palette
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance: Predicting Digital Payment Usage")
plt.grid(axis="x", linestyle="--", alpha=0.5)
plt.show()
```



```
<ipython-input-46-754c1d156f75>:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe

  sns.barplot(x="Importance", y="Feature", data=feature_importances, palette="viridis")  # Changed color palette
```

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler

# File paths
csv_file_path = "/content/micro_world_139countries.csv"
excel_file_path = "/content/CountryLevel_DatabankWide.xlsx"

# Read the Excel file
df_excel = pd.read_excel(excel_file_path, sheet_name=0)
df_excel_cleaned = df_excel.dropna(axis=1, how="all")  # Remove empty columns

# Read CSV file (Sampling to handle large size)
df_csv = pd.read_csv(csv_file_path, encoding="ISO-8859-1", low_memory=False)
df_csv_cleaned = df_csv.dropna(axis=1, how="all")  # Drop empty columns

# Select relevant financial features
feature_cols = ["age", "account", "account_mob", "saved", "borrowed"]

# Ensure only available columns are used
feature_cols = [col for col in feature_cols if col in df_csv_cleaned.columns]

# Prepare feature matrix (X) and target variable (y)
df_cleaned = df_csv_cleaned.dropna(subset=feature_cols + ["anydigpayment"])  # Remove
NaN rows
X = df_cleaned[feature_cols]
y = df_cleaned["anydigpayment"]

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train a Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_scaled, y)

# Get feature importance scores
feature_importances = pd.DataFrame({
    "Feature": feature_cols,
    "Importance": rf_model.feature_importances_
}).sort_values(by="Importance", ascending=False)

# Print feature importance table
print("Feature Importance Table")
print(feature_importances)
```

```
# Plot feature importance
plt.figure(figsize=(8, 5))
sns.barplot(x=feature_importances["Importance"], y=feature_importances["Feature"],
palette="viridis")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.title("Feature Importance in Financial Inclusion")
plt.show()
```



```
Feature Importance Table
       Feature  Importance
1      account    0.712841
2  account_mob    0.223449
3        saved    0.039785
0          age    0.013518
4     borrowed    0.010407
<ipython-input-47-e8435554886b>:51: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe

  sns.barplot(x=feature_importances["Importance"], y=feature_importances["Feature"], palette="viridis")
```