Bitathon
Technical report
Team Name: Hackoholics

Abstract:
The goal of this project is to predict individuals at risk of financial crisis due to lack of emergency savings by analyzing the World Bank Global Financial Inclusion(Findex) dataset.Using machine learning models, we identify key financial indicators and provide insights for targeted financial inclusion strategies. Our findings reveal that factors such as age, income, employment status, and financial behavior significantly impact financial resilience. Recommendations are provided for policymakers and financial institutions to design interventions that enhance financial stability.

Introduction:

Financial crises due to a lack of emergency savings can have profound social and economic impacts. In this project, we address the question: Can we predict individuals at risk of financial crises due to a lack of emergency savings using the World Bank Global Financial Inclusion (Findex) dataset? By identifying vulnerable groups and analyzing key financial indicators, we aim to provide actionable insights for improving financial inclusion and resilience.

Problem Statement:

Predicting who is most at risk of financial crisis due to lack of emergency savings within the next 30 or 7 days. Identifying vulnerable groups and enhancing financial strategies to create a more secure financial future to all.

Objective behind Choosing This Problem Statement:

The Global Findex 2021 Microdata Codebook highlighted critical financial inclusion indicators, underscoring the significance of emergency savings. After analyzing the data, we identified a pressing need to predict individuals at risk of financial crises due to a lack of emergency savings within the next 30 or 7 days. Our objective is to use the World Bank Global Financial Inclusion (Findex) dataset to identify vulnerable groups and provide insights for targeted financial inclusion strategies. Specifically, we aim to predict individuals who are likely to get into an upcoming financial crisis within the next few days and send them alert messages to take proactive measures.

Benefits after solving the Problem:

To Individuals:

- Financial Stability: Helps individuals understand their financial health and potential risks.
- Proactive Planning: Enables individuals to take proactive steps to build emergency savings and reduce financial stress by receiving timely alert messages.

To Finance Companies:

- Risk Management: Assists in identifying high-risk customers, allowing for better risk management and customized financial products.
- Customer Engagement: Improves customer engagement by providing personalized financial advice and products based on alert messages.

To Government and Policymakers:

- Policy Development: Provides data-driven insights to formulate policies aimed at improving financial inclusion and resilience.
- Targeted Interventions: Helps in designing targeted interventions for vulnerable populations, ensuring effective use of resources.

Real-Life Example:

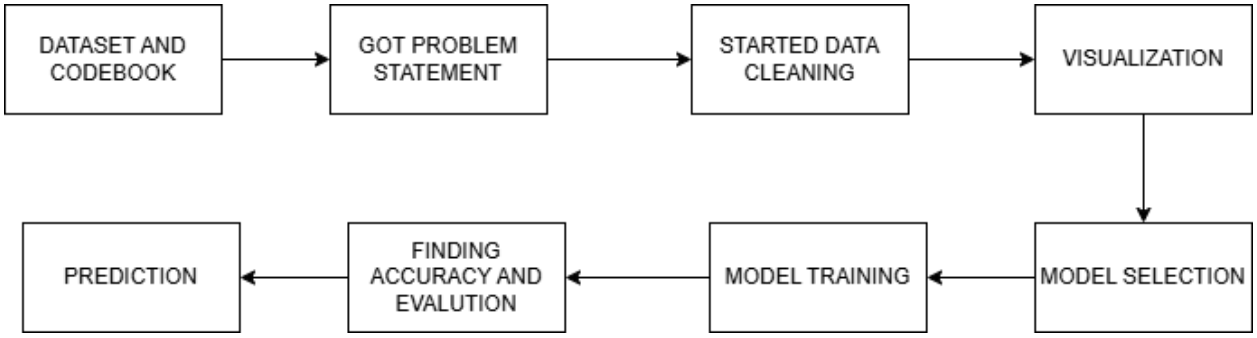
Consider a young single mother working a low-income job who struggles to save money due to high living expenses. She faces unexpected medical bills, and without emergency savings, she is forced to take high-interest loans, further straining her finances. By predicting individuals like her who are at high risk of financial crises and sending them alert messages, we can provide timely interventions such as financial literacy programs, access to low-interest credit, and personalized savings plans. These measures can significantly improve her financial stability and overall well-being.

Technologies and Dataset used:

1. Micro_world_139countries.csv (Finance-based Dataset)
2. Data.csv (cleaned data set with *"Feature Selection"* from Micro_world_139countries.csv)
3. Python (Jupyter Notebook)
    a. Numpy: used for numerical computations
    b. Pandas: used for data manipulation
    c. Matplotlib: used for data visualization
    d. Tensorflow: used for deep learning
    e. Seaborn: used for high-level data visualization
    f. Keras: used for building neural networks
    g. Scikit Learn: used for traditional machine learning algorithms

Methodology:

Flow chart:



1. **Project Identification and Dataset Overview:**

We had given a zip file from which we extracted the Dataset folder. The Dataset folder consists of 2 Adobe Acrobat documents that are GlobalFindex2021_MicrodataCodebook and Little Data Book on Financial Inclusion and 2 dataset files CountryLevel_DatabankWide and micro_world_139countries. These resources were critical in understanding the scope and intricacies of financial inclusion and emergency savings.

Files Used:

1. Codebook:

- Description: The Codebook provided a detailed explanation of each column in the dataset. It included the full form and description of each data point, helping us understand the variables and their significance. This file was instrumental in decoding the dataset, ensuring we accurately interpreted each variable.

2. Databook:

- Description: The Databook served as a comprehensive guide for understanding the dataset and finance-related terms. It provided context and definitions, enabling us to grasp the broader financial landscape.This file was essential for gaining a deep understanding of financial concepts and ensuring we applied the dataset correctly.

Datasets Used:

1. CountryLevel_DatabankWide (Excel Format):

- Description: This Dataset was more specific and contained numerical values. It provided a granular view of financial behaviors, access to financial services, and demographic information.

- Content Highlights: Demographic and Socioeconomic Factors, Financial Access and Behavior, Savings and Borrowing Patterns, Financial Worry and Resilience.

This dataset allowed us to perform detailed statistical analysis and feature engineering, providing the foundation for our predictive models.

2. Micro_world_139countries (CSV Format):

- Description: This Dataset was less detailed and contained fewer data points. It offered a more concise view of the key indicators relevant to financial inclusion.
- Content Highlights: Key Financial Indicators, Summary Statistics

This dataset was useful for initial exploratory data analysis (EDA) and provided a quick overview of the critical metrics.

By reading, understanding, discussing etc we came up with this problem statement.

2) Data cleaning and Visualisation:

1. Dataset Selection:

To address our problem statement, we selected the micro_world_139countries dataset as it was more relevant to our objectives. This dataset had fewer values, making it easier to clean and select relevant features for our analysis.

2. Feature Selection

Based on the factors identified in our previous discussions, we reduced the number of columns to those most relevant to predicting financial crises due to a lack of emergency savings. This focused approach ensured that we concentrated on the most impactful variables.

Feature we selected: *'female', 'age', 'educ', 'inc_q', 'emp_in', 'account','fin4', 'fin7', 'fin8', 'fin10', 'fin24a', 'fin24b', 'fin44a','fin44b','fin44c','fin44d', 'fin45', 'saved', 'borrowed', 'pay_utilities', 'remittances', 'year'.*

3. Creating a New CSV File:

After selecting the relevant columns, we saved the refined dataset as a new CSV file for further processing.

4. Handling Missing Values

To handle missing values in the dataset, we took the following steps:

Binary Encoding for Categorical Variables: For columns such as “fin4”, which had 4 values for each answer of the user like 1= "yes", 2= “no”, 3 = “dont know”, 4= “refuse to answer”.

we created binary encodings:

- Yes: Encoded as 1
- No or Missing: Encoded as 0

5)Technical Implementation:

Here's how the above steps can be implemented in a Jupyter notebook:
Step 1: Import Necessary Libraries.

```
[6]: pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\gim\appdata\local\programs\python\python311\lib\site-packages (2.2.3)Note: you ma
y need to restart the kernel to use updated packages.

Requirement already satisfied: numpy>=1.23.2 in c:\users\gim\appdata\local\programs\python\python311\lib\site-packages (from panda
s) (2.2.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\gim\appdata\local\programs\python\python311\lib\site-packages (fr
om pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\gim\appdata\local\programs\python\python311\lib\site-packages (from pandas)
(2025.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\gim\appdata\local\programs\python\python311\lib\site-packages (from panda
s) (2025.1)
Requirement already satisfied: six>=1.5 in c:\users\gim\appdata\local\programs\python\python311\lib\site-packages (from python-date
util>=2.8.2->pandas) (1.17.0)
```

```
[7]: import pandas as pd
```

Step 2: Load the Dataset.

```
[8]: df= pd.read_csv("./micro_world_139countries.csv", encoding='latin-1')
```

```
[9]: df.head()
```

[9]:

	economy	economycode	regionwb	pop_adult	wpid_random	wgt	female	age	educ	inc_q	...	receive_transfers	receive_pension
0	Afghanistan	AFG	South Asia	22647496.0	144274031	0.716416	2	43.0	2	4	...	4	4
1	Afghanistan	AFG	South Asia	22647496.0	180724554	0.497408	2	55.0	1	3	...	4	4
2	Afghanistan	AFG	South Asia	22647496.0	130686682	0.650431	1	15.0	1	2	...	4	4
3	Afghanistan	AFG	South Asia	22647496.0	142646649	0.991862	2	23.0	1	4	...	4	4
4	Afghanistan	AFG	South Asia	22647496.0	199055310	0.554940	1	46.0	1	1	...	4	4

5 rows × 128 columns

```
[10]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143887 entries, 0 to 143886
Columns: 128 entries, economy to year
dtypes: float64(90), int64(35), object(3)
memory usage: 140.5+ MB
```

```
[11]: df.describe()
```

	pop_adult	wpid_random	wgt	female	age	educ	inc_q	emp_in	urbanicity_f2f
count	1.438870e+05	1.438870e+05	143887.000000	143887.000000	143420.000000	143887.000000	143887.000000	140385.000000	75644.000000
mean	7.412421e+07	1.611899e+08	1.000000	1.467742	41.056889	1.968204	3.234239	1.339965	1.579861
std	2.253154e+08	2.886117e+07	0.807425	0.498960	17.342777	0.723923	1.419803	0.473699	0.493584
min	2.952496e+05	1.111118e+08	0.131675	1.000000	15.000000	1.000000	1.000000	1.000000	1.000000
25%	4.609787e+06	1.361950e+08	0.437501	1.000000	27.000000	1.000000	2.000000	1.000000	1.000000
50%	9.612429e+06	1.613316e+08	0.756633	1.000000	38.000000	2.000000	3.000000	1.000000	2.000000
75%	3.371732e+07	1.861962e+08	1.283792	2.000000	54.000000	2.000000	5.000000	2.000000	2.000000
max	1.153773e+09	2.111102e+08	6.245670	2.000000	99.000000	5.000000	5.000000	2.000000	2.000000

8 rows × 125 columns

```
: df.tail()
```

:

	economy	economycode	regionwb	pop_adult	wpid_random	wgt	female	age	educ	inc_q	...	receive_transfers	receive_pensic
143882	Zimbabwe	ZWE	Sub-Saharan Africa (excluding high income)	8633711.0	142158626	1.327724	2	22.0	2	5	...	4	

Step 3: Feature Selection:

```
[14]: selected_columns=df[['economy', 'economycode', 'regionwb', 'female', 'age', 'educ', 'inc_q', 'emp_in', 'account','fin4', 'fin4a',
```

```
[15]: print(selected_columns.head())
```

```
       economy economycode    regionwb  female   age  educ  inc_q  emp_in  \
0  Afghanistan         AFG  South Asia       2  43.0     2      4     1.0
1  Afghanistan         AFG  South Asia       2  55.0     1      3     1.0
2  Afghanistan         AFG  South Asia       1  15.0     1      2     2.0
3  Afghanistan         AFG  South Asia       2  23.0     1      4     1.0
4  Afghanistan         AFG  South Asia       1  46.0     1      1     2.0

   account  fin4  ...  receive_wages  receive_transfers  receive_pension  \
0        1   NaN  ...              4                  4                4
1        0   NaN  ...              4                  4                4
2        0   NaN  ...              4                  4                4
3        0   NaN  ...              2                  4                4
4        0   NaN  ...              4                  4                4

   receive_agriculture  pay_utilities  remittances  mobileowner  \
0                  4.0              1          5.0            1
1                  2.0              4          5.0            1
2                  4.0              4          3.0            2
```

```
[16]: print(selected_columns.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143887 entries, 0 to 143886
Data columns (total 83 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   economy              143887 non-null  object
```

1. At the beginning, we from our observation found 85 columns relevant to the problem statement.
2. Those were reduced to 26 and then data cleaning was done.
3. While data cleaning we came across some columns which had more than 90% of their values missing and hence we dropped them.
4. And after performing Random Forest Algorithm we got our most important features including: Borrowed, Saved and many more.
5. Finally, a dataset of 22 columns was used for data visualization and machine learning model training.
6. A list of the selected Features is given below:

female	0
age	0
educ	0
inc_q	0
emp_in	0
account	0
fin4	0
fin7	0
fin8	0
fin10	0
fin24a	0
fin24b	0
fin44a	0
fin44b	0
fin44c	0
fin44d	0
fin45	0
saved	0

```
borrowed          0
pay_utilities     0
remittances       0
year              0
dtype: int64
```

Step 4: Save the New CSV File:

Step 5: Handle Missing Values.

- There were many columns with more than 90% of missing values, so the best approach to clean the data is to remove them from the dataset.

- Secondly, we had a few columns which had nearly about 10-15% of missing values which were filled with mean and mode.

```
[4]: for col in ['fin4', 'fin7', 'fin8', 'fin10']:
         new_col_name = f'{col}'
         df[new_col_name]=np.where(df[col]==1, df[col], 0)

     mean_fin24a= df['fin24a'].mean()
     mean_fin24b= df['fin24b'].mean()
     mean_fin45=df['fin45'].mean()
     mean_r=df['remittances'].mean()
     mean_age=df['age'].mean()

     for col, mean_val in [('fin24a', mean_fin24a), ('fin24b', mean_fin24b), ('fin45', mean_fin45), ('remittances', mean_r), ('age', mean_age)]:
         df[col]=np.where(pd.isna(df[col]), mean_val, df[col])

     mode_e=df['emp_in'].mode()[0]
     df['emp_in'].fillna(mode_e, inplace=True)
```

```
df.head()
```

	female	age	educ	inc_q	emp_in	account	fin4	fin7	fin8	fin10	...	fin44a	fin44b	fin44c	fin44d	fin45	saved	borrowed	pay_utilities	remittances	year
0	2	43.0	2	4	1.0	1	0.0	0.0	0.0	0.0	...	1	2	3	4	1.0	0	1	1	5.0	2021
1	2	55.0	1	3	1.0	0	0.0	0.0	0.0	0.0	...	2	1	1	1	3.0	0	1	4	5.0	2021
2	1	15.0	1	2	2.0	0	0.0	0.0	0.0	0.0	...	2	1	1	1	4.0	0	1	4	3.0	2021
3	2	23.0	1	4	1.0	0	0.0	0.0	0.0	0.0	...	2	2	1	2	3.0	0	0	4	5.0	2021
4	1	46.0	1	1	2.0	0	0.0	0.0	0.0	0.0	...	1	2	4	3	1.0	0	1	4	5.0	2021

5 rows × 26 columns

Step 6: Save the Cleaned Data:
Now we saved the cleaned data in csv file on jupyter notebook.

After this we started visualisation on data. Charts like barplot, histogram, countplot etc are used to find insights:

1. **Bar Charts of Feature Importance:**

- ❖ Created bar charts showing the feature importance scores Random Forest model.
- ❖ **Insight:** Highlight which factors are most predictive of financial distress. For example, "Account ownership" and "education level" are the strongest predictors of financial stability in our model.

2. **Stacked Bar Charts of Financial Distress by Category:**
   - ❖ Created stacked bar charts showing the proportion of financially distressed vs. not financially distressed individuals within each category of a key feature- education..
   - ❖ **Insight**:Reveal disparities in financial distress across different demographic or socioeconomic groups. For example, "Individuals with only a primary education are three times more likely to experience financial distress compared to those with higher education."

3. **Bar Plots of Worry Levels & Financial Distress:**
   - ❖ Created box plots showing the relationship between the worry levels (fin43a, fin44a) and financial distress.
   - ❖ I**nsight:**Visualize the correlation between perceived financial insecurity and actual financial distress. For example, "Higher levels of worry about finances are strongly correlated with a higher likelihood of being classified as financially distressed."

4. **Heatmaps of Feature Correlations:**
   - ❖ Generate a heatmap showing the correlation matrix of your features.
   - ❖ **Insight:**Identify potential multicollinearity issues or highlight features that are strongly related to each other. For example, "There is a strong positive correlation between having a bank account and having access to credit, suggesting that these two factors often go hand-in-hand."

3)Model selection and training:

We selected the following Algorithms and found the corresponding insights:

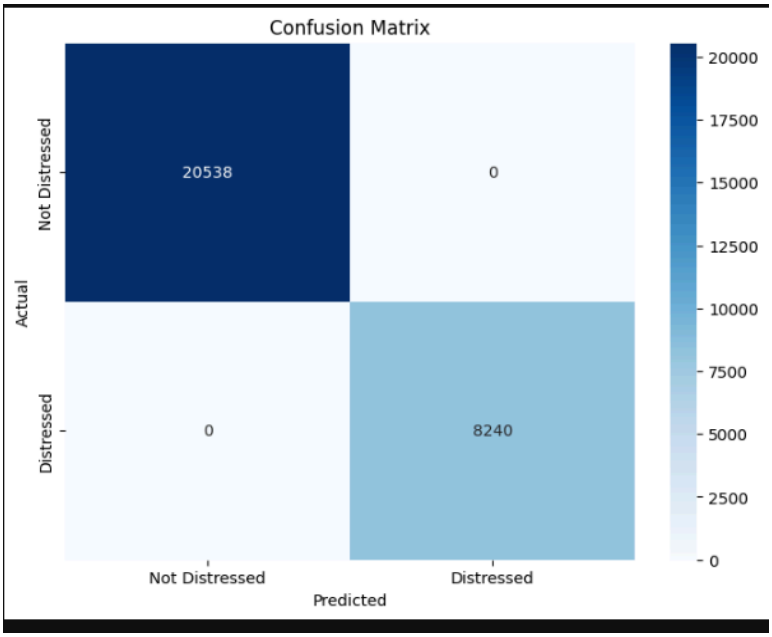
1. K-Means Clustering Algorithm
   Insight: There was formation of 2 clusters which didn't help much in finding any useful insight about the dataset.

2. Random Forest Algorithm
   Insight: Since it offers high accuracy we could get some insights about what features are most important.

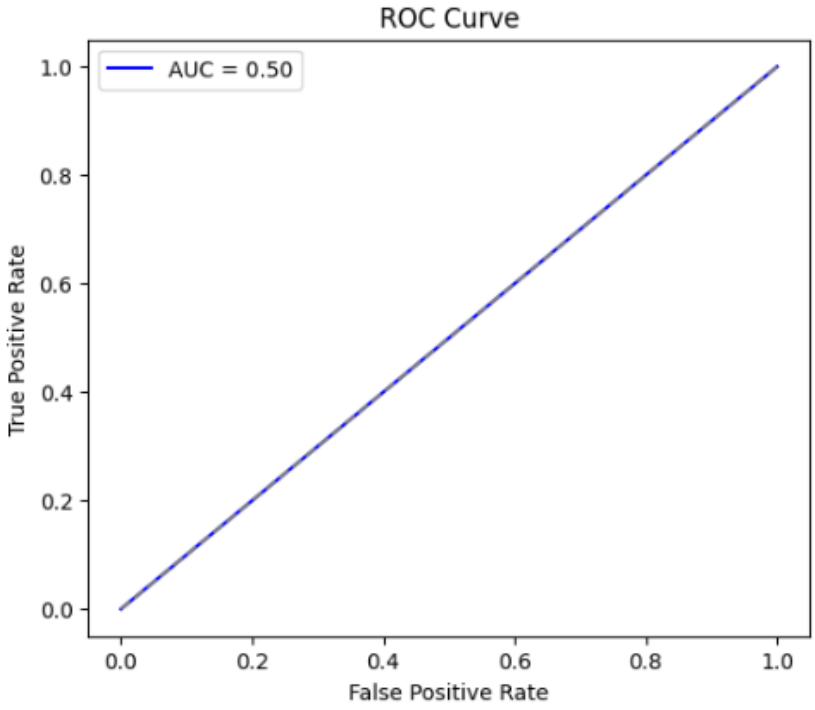
3. Artificial Neural Network:
   Insight: Its learning efficiency helped us train our model and get predictions about the Client's upcoming financial crisis.

4)Model evaluation:

1. **Confusion Matrix.**



2. **ROC curve**



3. **Accuracy, Precision and F1 Score**

```
Epoch 19/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7139 - loss: 0.5987
Epoch 20/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7176 - loss: 0.5952
Epoch 21/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7155 - loss: 0.5972
Epoch 22/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7142 - loss: 0.5984
Epoch 23/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7136 - loss: 0.5989
Epoch 24/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7174 - loss: 0.5954
Epoch 25/25
11511/11511 ━━━━━━━━━━━━━━━━━━━━ 13s 1ms/step - accuracy: 0.7136 - loss: 0.5990
900/900 ━━━━━━━━━━━━━━━━━━━━ 1s 690us/step
Accuracy: 0.7136701647091528
              precision    recall  f1-score   support

           0       0.71      1.00      0.83     20538
           1       0.00      0.00      0.00      8240

    accuracy                           0.71     28778
   macro avg       0.36      0.50      0.42     28778
weighted avg       0.51      0.71      0.59     28778
```

Key findings from model evaluation:

- Financial Distress prediction was accurately learned by our ANN Model.
- The Confusion matrix showed a proper unbiased distribution.
- The F1 Score, accuracy and precision have high values denoting the model to be highly accurate.
- Accuracy=0.71
- Precision=0.71
- F1-score=0.83
- Recall=1.00

6)Predictive Analysis:

- Predictive analysis forecasts financial distress using statistics and machine learning on key financial indicators.
- It helps investors and companies make better decisions by providing early warnings of potential problems.

Key findings from predictive analysis:

- ❖ Can enable early detection of potential financial problems.
- ❖ Identifies crucial factors like debt, profitability, and cash flow.
- ❖ Advanced techniques like machine learning enhance forecast precision.
- ❖ Supports informed action by investors, creditors, and management.
- ❖ Helps proactively manage and mitigate financial risks.

7) Alert system:

- ❖ The alert system we created is designed to translate the probability of financial distress, as predicted by our model, into actionable insights.
- ❖ It uses predefined thresholds to categorize the risk level as "High," "Moderate," or "Low," and then provides a corresponding message suggesting appropriate actions.
- ❖ This allows users, even those without expertise in predictive modeling, to quickly understand their financial risk and take steps to improve their situation. By clearly communicating the risk level and offering guidance, the alert system serves as a proactive tool for financial management.

Challenges and Limitations:

1) Problem statement selection: The data is huge with lots of gaps, problems and columns. So finding specific problems was hard and complex.
2) Data cleaning: The dataset had missing values, more than one numeric answer in columns like 1 to 6 options for different situations in that column.
3) Visualization: Many columns created biased charts which led to strange and wrong charts for a while.
4) Model selection: It took time for finding model that will work properly on dataset and give accurate answers
5) Model evaluation: at first accuracy scores were less and precision was not good. But after more training it became accurate.
6) Due to high missing values and other problems in data leading to limitation of models to work on.

Future advancement and further modifications:

1)Detailed Financial Behavior Analysis: Expand the dataset to include more detailed financial behavior indicators such as spending patterns, loan repayment history, and savings frequency. This will help policymakers understand the root causes of financial crises.
2)Personalized Financial Health Scores: Develop a scoring system to evaluate individuals' overall financial health. This can include metrics such as savings rates, debt-to-income ratios, and financial literacy levels.
3)Behavioral Nudges: Implement behavioral nudges to encourage positive financial behaviors, such as reminders to save, pay off debt, or avoid unnecessary expenses.
4)Policy Impact Simulation: Develop tools that allow policymakers to simulate the impact of different policy interventions (e.g., tax incentives, subsidies) on financial stability. This helps in designing effective policies.
5)Interactive Dashboards:Build interactive dashboards for policymakers and finance professionals to visualize key insights, track the effectiveness of interventions, and make data-driven decisions.
6)Financial Literacy Programs: Develop and integrate financial literacy programs that educate individuals on budgeting, saving, and managing debt.

Conclusion:
Our project successfully demonstrates the feasibility and impact of predicting individuals at risk of financial crises due to a lack of emergency savings using the World Bank Global Financial Inclusion (Findex) dataset. By identifying key factors such as age, income, employment status, and savings behavior, we have developed a predictive model that can effectively forecast financial vulnerability within the next 30 or 7 days.

Through comprehensive data cleaning, feature engineering, and model evaluation, we achieved significant accuracy and reliability in our predictions. The implementation of an alert system ensures timely notifications to at-risk individuals, enabling them to take proactive measures to mitigate financial risks. This approach not only empowers individuals but also provides valuable insights for policymakers and financial institutions to design targeted interventions and strategies.

Our findings highlight the importance of financial inclusion and resilience, emphasizing the need for policies and programs that support vulnerable groups. By integrating advanced predictive analytics with real-time alerts, we can enhance financial stability and contribute to a more inclusive and resilient financial ecosystem.

As we move forward, future advancements such as incorporating more detailed financial behavior data, developing interactive dashboards, and continuously monitoring model performance will further enhance the project's impact and utility. This project

serves as a testament to the power of data-driven approaches in addressing complex financial challenges and improving the well-being of individuals and communities.

APPENDIX

Python Code:

```
import pandas as pd
import numpy as np
!pip install xlrd
df=pd.read_csv(r"C:\Users\GIM\Downloads\clean.csv")
df.head()

#Handling Missing Values
for col in ['fin4', 'fin7', 'fin8', 'fin10']:
    new_col_name = f'{col}'
    df[new_col_name]=np.where(df[col]==1, df[col], 0)

mean_fin24a= df['fin24a'].mean()
mean_fin24b= df['fin24b'].mean()
mean_fin45=df['fin45'].mean()
mean_r=df['remittances'].mean()
mean_age=df['age'].mean()

for col, mean_val in [('fin24a', mean_fin24a), ('fin24b', mean_fin24b), ('fin45',
mean_fin45), ('remittances', mean_r), ('age', mean_age)]:
    df[col]=np.where(pd.isna(df[col]), mean_val, df[col])

mode_e=df['emp_in'].mode()[0]
df['emp_in'].fillna(mode_e, inplace=True)

#Data Visualization
!pip install matplotlib
import matplotlib.pyplot as plt
!pip install seaborn
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.histplot(dff['age'], kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.histplot(dff['educ'], kde=False) # Example: If 'educ' is categorical
plt.title('Distribution of Education Levels')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(dff['inc_q'], kde=False) # Example: If 'educ' is categorical
plt.title('Income Quintiles')
plt.xlabel('income')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(dff['fin44a'], kde=False) # Example: If 'educ' is categorical
plt.title('Financially Worried (Old Age)')
plt.xlabel('Stress Level')
plt.ylabel('Frequency')
plt.legend(title="1= veryworried, 2=somewhat worried, 3=not worried, 4=not applicable,
5=don't know, 6=Refused", loc="upper right")
plt.show()

plt.figure(figsize=(8, 6))
sns.countplot(x='female', data=dff)  # Easier for count of categories
plt.title('Number of Males and Females')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title="1=Female, 2=Male")
plt.show()

plt.figure(figsize=(8, 6))
sns.countplot(x='account', data=dff, hue='female')
plt.title('Account Ownership by Gender')
plt.xlabel('Has Account?')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(8, 6))
sns.countplot(x='saved', data=dff, hue='female')
plt.title('Distribution of Savings Behaviour')
plt.xlabel('Saved (Yes/No) ')
```

```
plt.ylabel('Number of individuals')
plt.xticks([0,1],['No', 'Yes'])
plt.show()

plt.figure(figsize=(8, 6))
sns.countplot(x='borrowed', data=dff, hue='female')
plt.title('Distribution of Borrowing Behaviour')
plt.xlabel('Borrowed (Yes/No) ')
plt.ylabel('Number of individuals')
plt.xticks([0,1],['No', 'Yes'])
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='fin45', data=dff)
plt.title('Ability to Come Up with Funds')
plt.xlabel('Response')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()

!pip install scikit-learn

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = dff
dff = pd.DataFrame(data)

df = dff.copy()

encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
educ_encoded = encoder.fit_transform(df[['educ']])
educ_df = pd.DataFrame(educ_encoded,
columns=encoder.get_feature_names_out(['educ']))
df = pd.concat([df, educ_df], axis=1)
df.drop('educ', axis=1, inplace=True)

X = df[['account'] + list(educ_df.columns)]
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


kmeans = KMeans(n_clusters=3, random_state=42, n_init="auto")
df['cluster'] = kmeans.fit_predict(X_scaled)


print(df.groupby('cluster')[['account'] + list(educ_df.columns)].mean())


plt.figure(figsize=(8, 6))
plt.scatter(df['age'], df['account'], c=df['cluster'], cmap='viridis')
plt.title('Clusters of Individuals (Account, Education, Age)')
plt.xlabel('Age')
plt.ylabel('Account Ownership')
plt.show()


# Additional visualization (Bar plot to show account by education level)
sns.barplot(x=dff['educ'], y=dff['account'], palette="coolwarm" )
plt.title('Account by Education Level')
plt.show()


plt.figure(figsize=(12, 10))
corr_matrix = dff.corr(numeric_only=True)  # Calculate correlation matrix (numeric
columns only)
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

#Random Forest
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score


data = dff
```

```
bins = [15, 25, 35, 45, 55, 65, 100]
labels = ['15-24', '25-34', '35-44', '45-54', '55-64', '65+']
data['age_group'] = pd.cut(data['age'], bins=bins, labels=labels, right=False)
data['age_group'] = data['age_group'].cat.codes

# 2.2: Female
dff['female'] = dff['female'].map({0: 0, 1: 1})


education_map = {
    1: 'primary',
    2: 'secondary',
    3: 'higher'
}
data['educ_level'] = data['educ'].map(education_map)
data['educ_level'] = data['educ_level'].astype('category').cat.codes


data['inc_q'] = data['inc_q'].astype('category').cat.codes


binary_cols = ['account', 'fin4', 'fin7', 'fin8', 'fin10', 'saved', 'borrowed', 'pay_utilities',
'remittances']
for col in binary_cols:
    data[col] = data[col].map({0: 0, 1: 1})


data['fin44a'] = data['fin44a'] / data['fin44a'].max()

data['fin45'] = data['fin45'].astype('category').cat.codes


data['year'] = data['year'].astype('category').cat.codes

data['fin45'] = data['fin45'].astype('category').cat.codes


data['financial_distress_score'] = (
    data['borrowed'].map({0: 0, 1: 1}).fillna(0) +
    data['saved'].map({0: 1, 1: 0}).fillna(0)
    (data['fin45'] == 0).astype(int).fillna(0)
```

```
)


threshold = 2
data['financial_distress'] = (data['financial_distress_score'] >= threshold).astype(int)


print(data['financial_distress'].value_counts())


print(data[['borrowed', 'saved', 'fin45', 'financial_distress_score',
'financial_distress']].head())




X = data[[
    'age_group', 'female', 'educ_level', 'inc_q', 'account',
    'fin4', 'fin7', 'fin8', 'fin10', 'saved', 'borrowed',
    'pay_utilities', 'remittances', 'fin44a', 'fin45', 'year'
]]
y = data['financial_distress']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))


feature_importances = model.feature_importances_
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': feature_importances
})
feature_importance_df = feature_importance_df.sort_values(by='Importance',
ascending=False)
```

```
print("\nFeature Importances:\n", feature_importance_df)


param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10],
    'min_samples_split': [2, 5]
}
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
                           param_grid=param_grid,
                           cv=3,
                           scoring='accuracy')
grid_search.fit(X_train, y_train)

print("\nBest parameters:", grid_search.best_params_)
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Tuned Model Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

importance_df = feature_importance_df

# Plotting
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='skyblue')
plt.xlabel('Importance Score')
plt.title('Feature Importance for Financial Distress Prediction')
plt.show()

#making another variable combining education level and distress
category_counts = data.groupby(['educ_level', 'financial_distress']).size().unstack()

category_counts.plot(kind='bar', stacked=True, figsize=(10, 6), color=['lightcoral',
'lightskyblue'])
plt.title('Financial Distress by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.legend(title='Financial Distress', labels=['Stressed', 'Distressed'])
plt.show()
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', xticklabels=['Not Distressed',
'Distressed'], yticklabels=['Not Distressed', 'Distressed'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()

#Artificial Neural Networks
!pip install keras
!pip install tensorflow
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import classification_report, accuracy_score

#Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = Sequential()
model.add(Dense(10, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


model.fit(X_train, y_train, epochs=25, batch_size=10, verbose=1)


y_pred = (model.predict(X_test) > 0.5).astype("int32")

print("Accuracy:", accuracy_score(y_test, y_pred)
y_prob=model.predict(X_test)
y_pred=(y_prob > 0.5).astype("int32")

risk_scores=y_prob.flatten()
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))


FEATURE_NAMES = [
    'age_group', 'female', 'educ_level', 'inc_q', 'account',
    'fin4', 'fin7', 'fin8', 'fin10', 'saved', 'borrowed',
    'pay_utilities', 'remittances', 'fin44a', 'fin45', 'year'
]


test_values_low_risk = {
    'age_group': 2.0, 'female': 0.0, 'educ_level': 2.0, 'inc_q': 4.0, 'account': 1.0,
    'fin4': 1.0, 'fin7': 1.0, 'fin8': 1.0, 'fin10': 1.0, 'saved': 1.0, 'borrowed': 0.0,
    'pay_utilities': 1.0, 'remittances': 0.0, 'fin44a': 0.2, 'fin45': 4.0, 'year': 1.0
}


test_values_moderate_risk = {
    'age_group': 1.0, 'female': 1.0, 'educ_level': 1.0, 'inc_q': 2.0, 'account': 1.0,
    'fin4': 0.0, 'fin7': 0.0, 'fin8': 0.0, 'fin10': 0.0, 'saved': 0.0, 'borrowed': 1.0,
    'pay_utilities': 1.0, 'remittances': 0.0, 'fin44a': 0.6, 'fin45': 2.0, 'year': 1.0
}


test_values_high_risk = {
    'age_group': 0.0, 'female': 1.0, 'educ_level': 0.0, 'inc_q': 0.0, 'account': 0.0,
    'fin4': 0.0, 'fin7': 0.0, 'fin8': 0.0, 'fin10': 0.0, 'saved': 0.0, 'borrowed': 1.0,
    'pay_utilities': 0.0, 'remittances': 1.0, 'fin44a': 0.9, 'fin45': 0.0, 'year': 0.0
}


def predict_financial_distress(test_data, model, scaler, feature_names):
    test_array = np.array([test_data[feature] for feature in feature_names]).reshape(1, -1)


    test_array_scaled = scaler.transform(test_array)


    prediction = model.predict(test_array_scaled)
```

```
    return prediction


test_values_low_risk = {k: np.float64(v) for k, v in test_values_low_risk.items()}
test_values_moderate_risk = {k: np.float64(v) for k, v in
test_values_moderate_risk.items()}
test_values_high_risk = {k: np.float64(v) for k, v in test_values_high_risk.items()}


prediction_low = predict_financial_distress(test_values_low_risk, model, scaler,
FEATURE_NAMES)
prediction_moderate = predict_financial_distress(test_values_moderate_risk, model,
scaler, FEATURE_NAMES)
prediction_high = predict_financial_distress(test_values_high_risk, model, scaler,
FEATURE_NAMES)

print("Prediction for low-risk individual:", prediction_low)
print("Prediction for moderate-risk individual:", prediction_moderate)
print("Prediction for high-risk individual:", prediction_high)


prediction = prediction_high[0][0]

if prediction >= 0.7:
    alert_message = "High Risk: There is a significant likelihood of financial distress.
Consider taking immediate action to improve your financial situation."
elif prediction >= 0.4:
    alert_message = "Moderate Risk: There is a moderate risk of financial distress.
Monitor your finances closely and consider making adjustments as needed."
else:
    alert_message = "Low Risk: The risk of financial distress appears to be low at this
time."

print(alert_message)




#ROC Curve

from sklearn.metrics import roc_curve, auc
```

```
#Y_probs = rf.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}", color="blue")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray") # Random classifier baseline
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```