My Report for Bank Account Prediction System

1. Imported Libraries:

Fundamental libraries for data processing and visualization are imported:

* pandas
* numpy
* seaborn
* matplotlib

2. Load Data:

Training and testing datasets are loaded using

“pd.read\_csv()”

:

```python

training\_set = pd.read\_csv("Train.csv")

testing\_set = pd.read\_csv("Test.csv")

```

3. Data Exploration

- Basic insights from the data are generated using `info()`, `head()`, and `describe()` methods.

- Missing values are checked using `isnull()` and `isnull().sum()`.

4. Outlier Detection and Handling

- Outliers in the `age\_of\_respondent` column are detected using the mean and standard deviation.

- Outliers are visualized using a boxplot.

- Outliers are removed using the Trimming method:

```python

upper\_limit = training\_set['age\_of\_respondent'].mean() + 3\*training\_set['age\_of\_respondent'].std()

lower\_limit = training\_set['age\_of\_respondent'].mean() - 3\*training\_set['age\_of\_respondent'].std()

new\_data = training\_set.loc[(training\_set['age\_of\_respondent']<upper\_limit)&(training\_set['age\_of\_respondent']>lower\_limit)]

```

5. \*\*Data Visualization\*\*

- Relationships between the target variable (`bank\_account`) and other features are visualized using `countplot`.

6. Data Preprocessing

- Categorical values are transformed into numerical values using `LabelEncoder` and

pd.get\_dummies()

.

- A preprocessing function is created to handle these transformations and scaling:

```python

def preprocessing\_data(data):

float\_array = data[["household\_size", "age\_of\_respondent", "year"]].values.astype(float)

categ = ["relationship\_with\_head", "marital\_status", "education\_level", "job\_type", "country"]

data = pd.get\_dummies(data, prefix\_sep="\_", columns=categ)

le = LabelEncoder()

data["location\_type"] = le.fit\_transform(data["location\_type"])

data["cellphone\_access"] = le.fit\_transform(data["cellphone\_access"])

data["gender\_of\_respondent"] = le.fit\_transform(data["gender\_of\_respondent"])

data = data.drop(["uniqueid"], axis=1)

scaler = MinMaxScaler(feature\_range=(0, 1))

data = scaler.fit\_transform(data)

return data

```

7. Model Selection and Training

- The data is split into training and validation sets using `train\_test\_split`.

- An `XGBClassifier` model is created and trained.

- The model's performance is evaluated using accuracy and confusion matrix.

### 8. \*\*Hyperparameter Optimization\*\*

- Grid Search and Bayesian Optimization are used to find the best hyperparameters for the model.

- The optimized model is evaluated again.

### 9. \*\*Prediction and Submission\*\*

- The best model is used to predict the target variable for the testing set.

- The results are saved to a CSV file for submission:

```python

submission = pd.DataFrame({"uniqueid": testing\_set["uniqueid"] + " x " + testing\_set["country"], "bank\_account": testing\_set.bank\_account})

submission.to\_csv('first\_submission.csv', index=False)

```

### Key Code Snippets

- \*\*Import Libraries\*\*:

```python

import pandas as pd

import numpy as np

import seaborn as sb

from matplotlib import pyplot as plt

```

- \*\*Load Data\*\*:

```python

training\_set = pd.read\_csv("Train.csv")

testing\_set = pd.read\_csv("Test.csv")

```

- \*\*Data Preprocessing\*\*:

```python

def preprocessing\_data(data):

float\_array = data[["household\_size", "age\_of\_respondent", "year"]].values.astype(float)

categ = ["relationship\_with\_head", "marital\_status", "education\_level", "job\_type", "country"]

data = pd.get\_dummies(data, prefix\_sep="\_", columns=categ)

le = LabelEncoder()

data["location\_type"] = le.fit\_transform(data["location\_type"])

data["cellphone\_access"] = le.fit\_transform(data["cellphone\_access"])

data["gender\_of\_respondent"] = le.fit\_transform(data["gender\_of\_respondent"])

data = data.drop(["uniqueid"], axis=1)

scaler = MinMaxScaler(feature\_range=(0, 1))

data = scaler.fit\_transform(data)

return data

```

- \*\*Model Selection and Training\*\*:

```python

from xgboost import XGBClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, accuracy\_score