

# **Credit Card Customer Churning**

## **(Banking Sector)**

Milestone: Final Project Report

Group 12  
Hrithik Puri  
Saketh Kottissa

617-935-8046(Hrithik Puri)  
617-238-8114 (Saketh Kottissa)

puri.hr@northeastern.edu  
kottissa.s@northeastern.edu

Percentage of Effort Contributed by Hrithik Puri: 50%  
Percentage of Effort Contributed by Saketh Kottissa: 50%

Signature of Student 1: Hrithik Puri  
Signature of Student 2: Saketh Kottissa

Submission Date: 04/23/23

## Problem Definition:

- The banking sector is facing a significant challenge in retaining customers. Many customers are choosing to switch to other banks, resulting in a loss of revenue for the banks. The banking sector needs to identify the factors that are causing customers to leave and develop strategies to retain them.
- To address this problem, we will build a predictive model that can predict customer churn in the banking sector. The model will use historical customer data, including demographic information, account information, and transaction history, to identify patterns that are associated with customer churn.
- The model will be trained on a large dataset of past customer information and will be able to predict which customers are at risk of leaving in the future. This will allow banks to proactively reach out to these customers and implement retention strategies before they leave.
- Overall, the predictive model and customer feedback analysis will provide the banking sector with valuable insights into customer churn and will allow them to proactively retain customers, resulting in increased revenue and customer satisfaction.

## Data Sources:

*Kaggle Dataset:*

<https://www.kaggle.com/datasets/whenamancodes/credit-card-customers-prediction>

## Data Description:

1. The dataset we are going to use is BankChurners.csv. It contains 10127 records and 23 columns at present
2. The Info of the dataset is as follows:

#	Column	Non-Null Count
0	CLIENTNUM	10127 non-null
1	Attrition_Flag	10127 non-null
2	Customer_Age	10127 non-null
3	Gender	10127 non-null
4	Dependent_count	10127 non-null
5	Education_Level	10127 non-null
6	Marital_Status	10127 non-null
7	Income_Category	10127 non-null
8	Card_Category	10127 non-null
9	Months_on_book	10127 non-null
10	Total_Relationship_Count	10127 non-null
11	Months_Inactive_12_mon	10127 non-null
12	Contacts_Count_12_mon	10127 non-null
13	Credit_Limit	10127 non-null
14	Total_Revolving_Bal	10127 non-null
15	Avg_Open_To_Buy	10127 non-null
16	Total_Amt_Chng_Q4_Q1	10127 non-null
17	Total_Trans_Amt	10127 non-null
18	Total_Trans_Ct	10127 non-null
19	Total_Ct_Chng_Q4_Q1	10127 non-null
20	Avg_Utilization_Ratio	10127 non-null
21	Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1	10127 non-null
22	Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2	10127 non-null

# Exploratory Data Analysis

- Value count of target class

```
df['Attrition_Flag'].value_counts()
```

```
Existing Customer    8500
Attrited Customer    1627
Name: Attrition_Flag, dtype: int64
```

- Checked for duplicate values

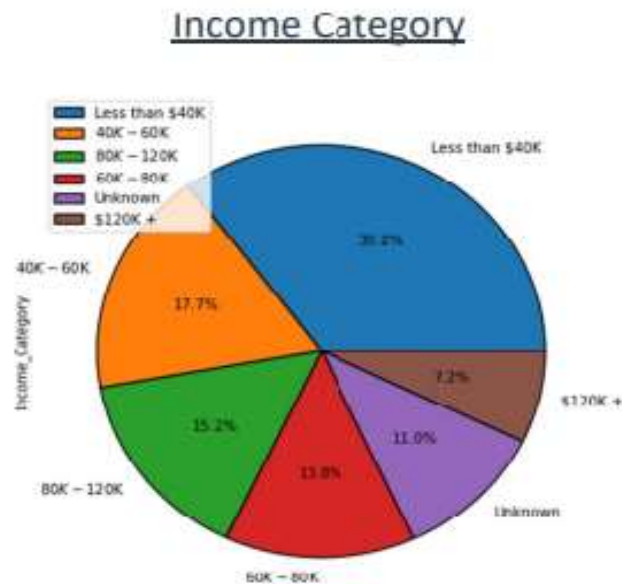
```
df.duplicated().sum()
```

```
0
```

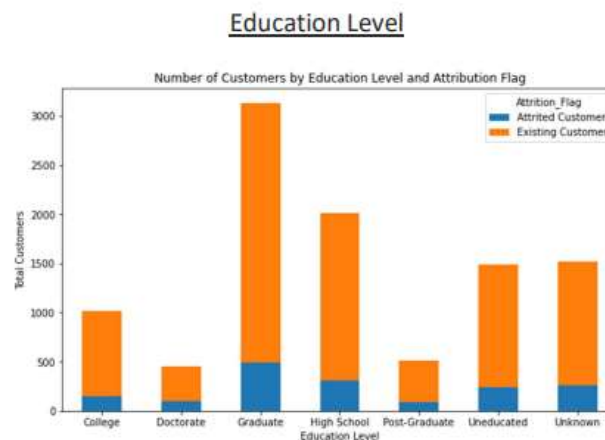
- Used describe() to check all the statistical properties of the given dataset i.e mean, median,25th, 50th, 75th, min, and max values.

	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal
count	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000
mean	46.325960	2.346203	35.928409	3.812580	2.341167	2.455317	8631.953698	1162.814061
std	8.016814	1.298908	7.986416	1.554408	1.010622	1.106225	9088.776650	814.987335
min	26.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1438.300000	0.000000
25%	41.000000	1.000000	31.000000	3.000000	2.000000	2.000000	2555.000000	359.000000
50%	46.000000	2.000000	36.000000	4.000000	2.000000	2.000000	4549.000000	1276.000000
75%	52.000000	3.000000	40.000000	5.000000	3.000000	3.000000	11067.500000	1784.000000
max	73.000000	5.000000	56.000000	6.000000	6.000000	6.000000	34516.000000	2517.000000

## Data Visualization

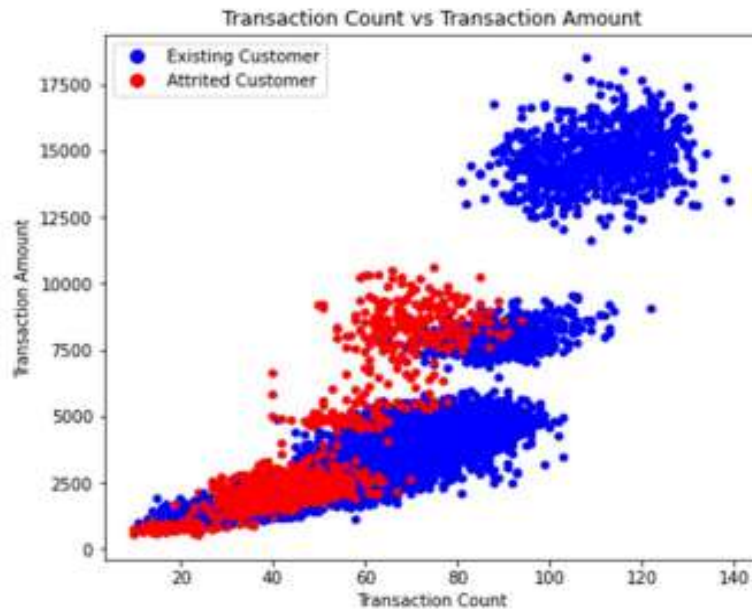


- The majority of credit card users were those whose annual income was under \$40,000 USD. This group is our primary target, therefore rather than concentrating on marketing credit cards to those with high incomes, we stand a higher chance of succeeding by promoting to people with incomes under 40K USD.



- Our largest market is university graduates (Graduates). Similar to the income category's insight, we should give this segment's (Graduate) persons (promotion priority), as doing so will raise our acquisition rate.

## Transactions Count vs Transactions Amount



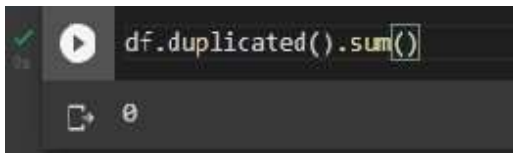
- Based on their transaction counts vs. transaction amounts, we can see that there are 3 clusters. Only from this plot, we can conclude that individuals with high transaction counts and transaction amounts had a lower likelihood of attrition—not a single person left the top right cluster. In order to lower their attrition rate, the middle and bottom clusters may be further examined and surveyed

## Data Pre-processing:

The dataset has 23 features consisting of customer profile and credit card usage. It has 10,127 rows. The tools that were used are Google Colab and Python. We checked whether our dataset has a duplicate record or not. We also checked whether we have missing values or not in the dataset that will affect the machine learning algorithm.

### Duplicate Values:

We can see in the picture below; we don't have any duplicates in the dataset

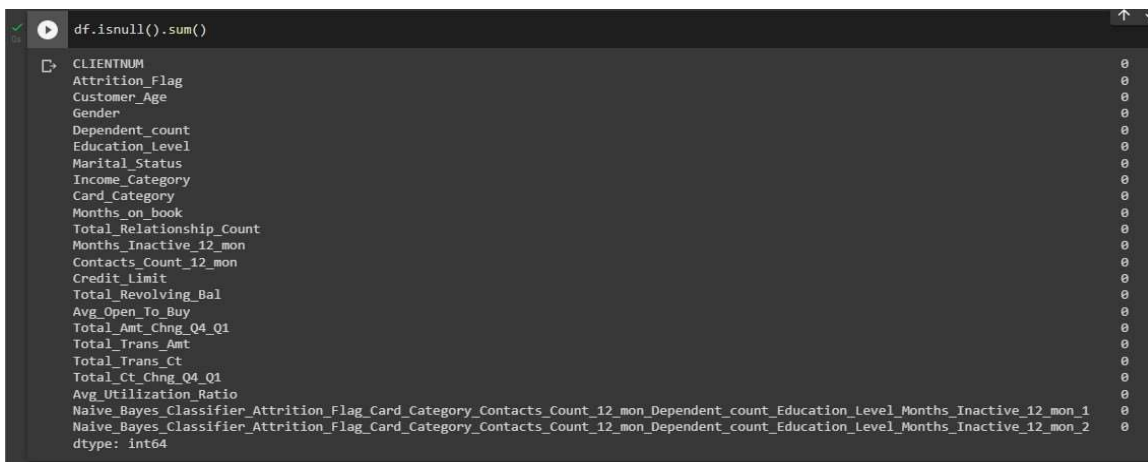


```
df.duplicated().sum()
```

0

### Missing Values:

We can see in the picture below; we don't have any missing values either in the dataset



```
df.isnull().sum()
```

CLIENTNUM	0
Attrition_Flag	0
Customer_Age	0
Gender	0
Dependent_count	0
Education_Level	0
Marital_Status	0
Income_Category	0
Card_Category	0
Months_on_book	0
Total_Relationship_Count	0
Months_Inactive_12_mon	0
Contacts_Count_12_mon	0
Credit_Limit	0
Total_Revolving_Bal	0
Avg_Open_To_Buy	0
Total_Amt_Chng_Q4_Q1	0
Total_Trans_Amt	0
Total_Trans_Ct	0
Total_Ct_Chng_Q4_Q1	0
Avg_Utilization_Ratio	0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1	0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2	0

dtype: int64

### Feature Encoding:

Feature encoding is done so that our machine can read our categorical data

```
✓ [18] le=LabelEncoder()
      df.Attrition_Flag=le.fit_transform(df.Attrition_Flag)
      df.Education_Level=le.fit_transform(df.Education_Level)
      df.Income_Category=le.fit_transform(df.Income_Category)
```

```
✓ [19] mapCard = {"Blue":0, "Silver":1, "Gold":2, "Platinum":3}
      df["Card_Category"] = df["Card_Category"].map(mapCard)
```

```
✓ [20] gender=pd.get_dummies(df.Gender,prefix='Gender')
      marital_status=pd.get_dummies(df.Marital_Status,prefix='Marital_status')
```

## Train and Validation Split:

- To make sure our data is not overfitting we will use a simple train test split with a 75:25 ratio.

```
✓ [24] X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.75,random_state=10)
```

## Data Scaling:

- Because some of our features have different units and value ranges—for instance, Credit Limit has values in the thousands of dollars while Customer Age values are below 100—some algorithms may assume that Credit Limit has a greater impact on the target than Customer Age. As a result, we need to standardize all our features to ensure that our model is not skewed.

```
✓ [25] sc=MinMaxScaler(feature_range=(0,1))
      X_train_sc = sc.fit_transform(X_train)
      X_test_sc = sc.transform(X_test)
```

## Data Imbalance Correction:

- The dataset target is imbalanced so SMOTE function was used to correct it.

```
[26] smt = SMOTE(sampling_strategy=0.8)
      X_train_sc_sm, y_train_sm = smt.fit_resample(X_train_sc, y_train)
```

(Before)

```
✓ [28] df1.value_counts()
      1    2118
      0     413
      dtype: int64
```

(After)

```
✓ [29] df2.value_counts()
      1    2118
      0   1694
      dtype: int64
```

## Model Building and Performance Evaluation

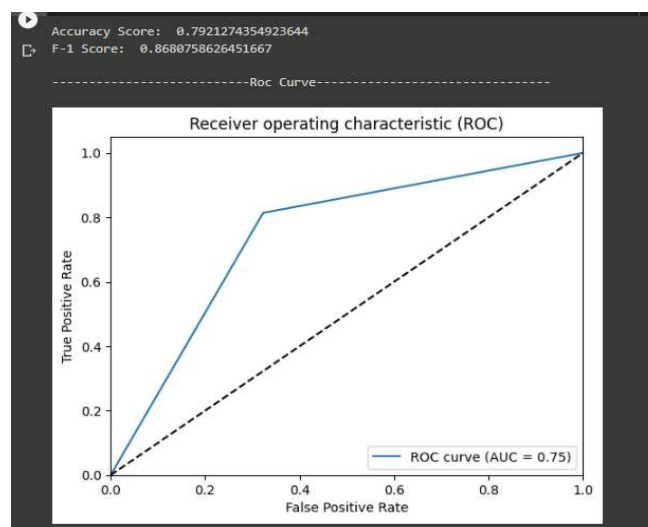
### KNeighbor Classifier

- A grid search was done to obtain the best hyperparameter to train the model using: `{'n_neighbors': 3, 'weights': 'distance'}`
- Training the model

```
[ ] model1=KNeighborsClassifier(n_neighbors=3,weights='distance')  
    model1.fit(X_train_sc_sm,y_train_sm) #trained model
```

```
▼ KNeighborsClassifier  
KNeighborsClassifier(n_neighbors=3, weights='distance')
```

- 79.21% of accuracy was achieved on the validation data
- An F-1 score of 86.80% indicates high False Positives and high False Negatives, hence not correctly identifying non-churning instances.
- The ROC curve was away from the top-left corner, with an AUC value of 0.75, indicating poor discrimination between the non-churning and churning classes





## Logistics Regression

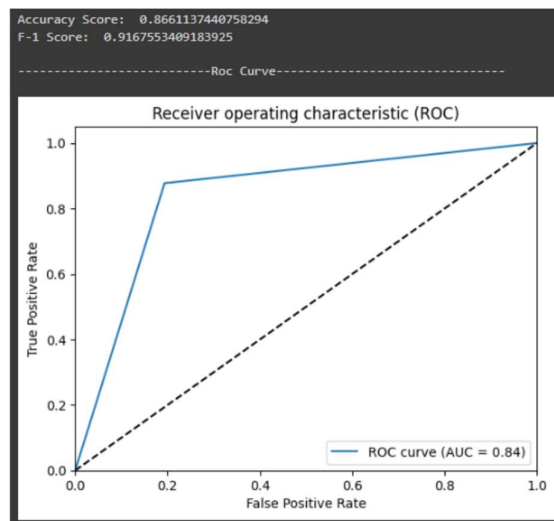
- A grid search was done to obtain the best hyperparameter to train the model using:  $C=5$ , penalty = 'l2' and solver = 'liblinear'
- Training the model

```
[ ] model2=LogisticRegression(C=5,penalty='l2',solver='liblinear')
model2.fit(X_train_sc_sm,y_train_sm)
```

LogisticRegression

LogisticRegression(C=5, solver='liblinear')

- 86.61% of accuracy was achieved on the validation data
- An F-1 score of 91.67% indicates low False Positives and low False Negatives, hence correctly identifying non-churning instances.
- The ROC curve was away from the top-left corner, with an AUC value of 0.84, indicating fair discrimination between the non-churning and churning classes.



## Decision Tree Classifier

- A grid search was done to obtain the best hyperparameter to train the model using: `{'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 5}`
- Training the model

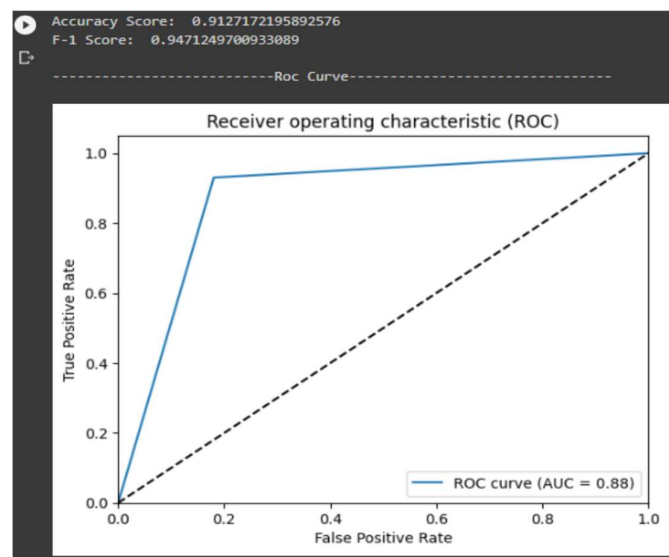
```
model3=DecisionTreeClassifier(criterion='entropy',max_depth=9,min_samples_split=5)
model3.fit(X_train_sc_sm,y_train_sm)
```

DecisionTreeClassifier

```
DecisionTreeClassifier(criterion='entropy', max_depth=9, min_samples_split=5)
```

- 91.27% of accuracy was achieved on the validation data
- An F-1 score of 94.71% indicates low False Positives and low False Negatives, hence correctly identifying non-churning instances.

The ROC curve was closer to the top-left corner, with an AUC value of 0.88, indicating good discrimination between the non-churning and churning classes



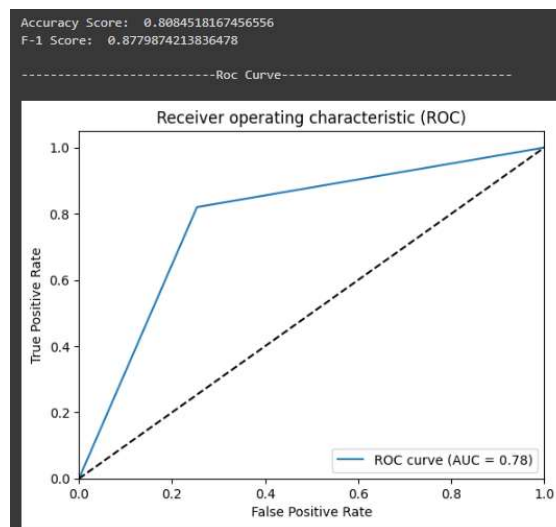
## Naive Bayes

- A grid search was done to obtain the best hyperparameter to train the model using: `{'var_smoothing': 1e-09}`
- Training the model

```
[ ] model4=GaussianNB(var_smoothing=1e-9)
model4.fit(X_train_sc_sm,y_train_sm)
```

▼ GaussianNB  
GaussianNB()

- 80.84% of accuracy was achieved on the validation data
- An F-1 score of 87.79% indicates high False Positives and high False Negatives, hence not correctly identifying non-churning instances.
- The ROC curve was away from the top-left corner, with an AUC value of 0.78, indicating poor discrimination between the non-churning and churning classes



## Neural Network

- A grid search was used to find the best model using 64 neurons in input layers, 32 neurons in hidden layer with RELU as activation function and output layer is single neuron with sigmoid as activation function

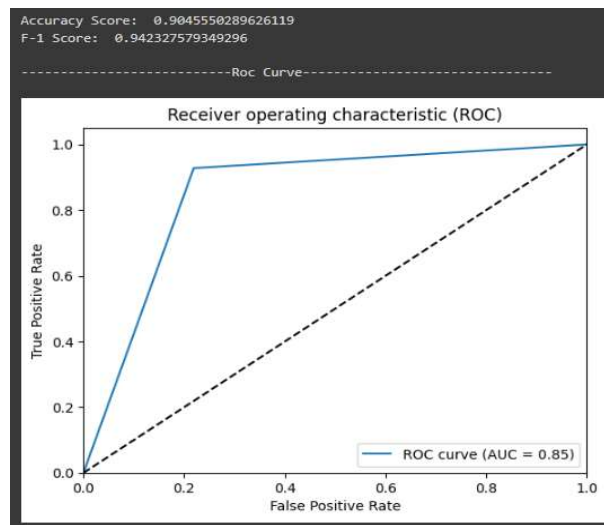
```
[ ] model = Sequential()

# Add layers to the model
model.add(Dense(units=64, activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))

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

[ ] model.fit(X_train_sc_sm, y_train_sm, epochs=25, batch_size=32, validation_data=(X_test_sc, y_test))
```

- 90.45% of accuracy was achieved on the validation data
- An F-1 score of 94.23% indicates low False Positives and low False Negatives, hence correctly identifying non-churning instances.
- The ROC curve was a little closer to the top-left corner, with an AUC value of 0.85, indicating fair discrimination between the non-churning and churning classes.



## **Final Results**

Model	Accuracy Score	F-1 Score	AUC
KNeighbor Classifier	0.7921	0.8680	0.75
Logistic Regression	0.8661	0.9167	0.84
Decision Tree Classifier	0.9127	0.9471	0.88
Naive Bayes	0.8084	0.8779	0.78
Neural Network	0.9045	0.9423	0.85

## **Conclusion**

Based on the evaluation metrics used, the Decision Tree Classifier and Neural Network were the top-performing models for the classification task. The Decision Tree Classifier had the highest accuracy score, F1 score, and AUC, indicating that it was the most effective model in distinguishing between positive and negative classes. The Neural Network also performed well, with a high accuracy score, F1 score, and AUC.

Therefore, it is recommended to prioritize the Decision Tree Classifier model for this task, as it had the highest overall performance. However, the Neural Network also showed promising results and may be a good alternative if computational complexity or interpretability is a concern.

## **Project Outcomes:**

The banking sector is highly competitive, and retaining customers is crucial for its long-term success. With the help of customer feedback analysis, banks can identify key areas where they need to improve their services and customer experience. By identifying the reasons why customers churn, banks can take proactive measures to address these issues and improve their customer retention rates. This can lead to increased customer loyalty and revenue, which is essential for the growth and sustainability of any business.

The development of a predictive model to identify customer churn is a valuable tool for banks. The decision tree classifier and neural network models have shown promising results and can

be used to predict which customers are at risk of leaving. This can allow banks to take appropriate measures to retain these customers before they churn. These measures can include targeted marketing campaigns, personalized offers, and proactive customer service, among others. By taking these actions, banks can show their customers that they are valued and that their concerns are being addressed, which can lead to increased loyalty and customer satisfaction.

In addition to improving customer retention rates, the use of predictive models can also help banks to optimize their resources. By identifying which customers are most likely to churn, banks can focus their efforts on retaining those customers and allocate their resources accordingly. This can lead to more efficient use of resources and improved profitability.

---

---