



Churn for Bank Accounts



Team: **JPMorganS**
Jinwoo Oh, Minjae Kwon,
Pranav Reddy, Srihith Duggi



TABLE OF CONTENTS

01

Introduction

02

Dataset

03

Method

04

Result

05

Conclusion

What is “Churn?”

In business, the rate at which customers stop doing business with a company over a given period of time.

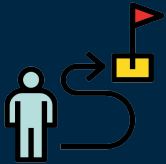
Introduction



Background – SVB bankruptcy made us think of preventing bank churn



Goal – Predicting if the customers are likely to turn over



Purpose – Essential for banks to recognize if they are in danger or not

Introduction



Dataset

- Dataset from Kaggle
- 14 columns / 10,000 rows
- 3 Types of features: Numerical, Categorical, and Binary



Method

- Data Curation, Cleaning, ETL
- Data Visualization
- Model Training, evaluating, and Selection
- Explaining a trained model's predictions using LIME
- Check the model bias and fairness

Dataset

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	0

Numerical	CreditScore, Age, Tenure, Balance, NumofProducts, EstimatedSalary
Categorical	RowNumber, Customerid, Surname, Geography, Gender
Binary	HasCrCard, IsActiveMeber, Exited

Dataset

- **RowNumber**: corresponds to the record (row) number
- **CustomerId**: contains random values
- **Surname**: the surname of a customer
- **CreditScore**: customer's credit score
- **Geography**: a customer's location
- **Gender**: customer's gender identity
- **Age**: customer's age
- **Tenure**: the number of years that the customer has been a client of the bank
- **Balance**: how much left in the customer's bank account
- **NumOfProducts**: the number of products that a customer has purchased through the bank
- **HasCrCard**: denotes whether or not a customer has a credit card
- **IsActiveMember**: if the customers active or not
- **EstimatedSalary**: customers' salary
- **Exited**: whether or not the customer left the bank

Method – Data Curation and ETL

Divide into features and label

We are examining if the user will churn or not, so the label should be "Exited" column because it's a indicator that shows if the user actually left the bank or not.

```
1 churn_features_df = churn_df[["CreditScore", "Gender", "Age", "Tenure", "Balance", "NumOfProducts", "HasCrCard", "I  
2 churn_label_df = churn_df[["Exited"]]
```

```
1 churn_features_df.head()
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	Female	42	2	0.00	1	1	1	101348.88
1	608	Female	41	1	83807.86	1	0	1	112542.58
2	502	Female	42	8	159660.80	3	1	0	113931.57
3	699	Female	39	1	0.00	2	0	0	93826.63
4	850	Female	43	2	125510.82	1	1	1	79084.10

```
1 churn_label_df.head()
```

	Exited
0	1
1	0
2	1
3	0
4	0

Method – Data Curation and ETL

Add the one hot encoded data to the original data

```
1 gender_transformed_df = pd.DataFrame(gender_transformed)
2
3 churn_features_df.reset_index(drop=True, inplace=True)
4 gender_transformed_df.reset_index(drop=True, inplace=True)
5
6 churn_features_transformed_df = pd.concat([churn_features_df, gender_transformed_df], axis=1)
7
8 churn_features_transformed_df = churn_features_transformed_df.drop(columns=["Gender"], axis=1)
9
10 churn_features_transformed_df.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	0	1
0	619	42	2	0.00	1	1	1	101348.88	1.0	0.0
1	608	41	1	83807.86	1	0	1	112542.58	1.0	0.0
2	502	42	8	159660.80	3	1	0	113931.57	1.0	0.0
3	699	39	1	0.00	2	0	0	93826.63	1.0	0.0
4	850	43	2	125510.82	1	1	1	79084.10	1.0	0.0

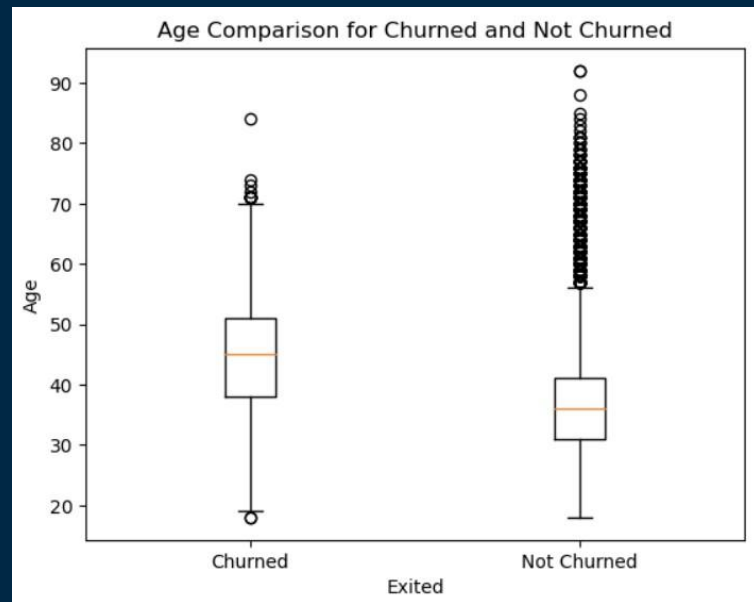
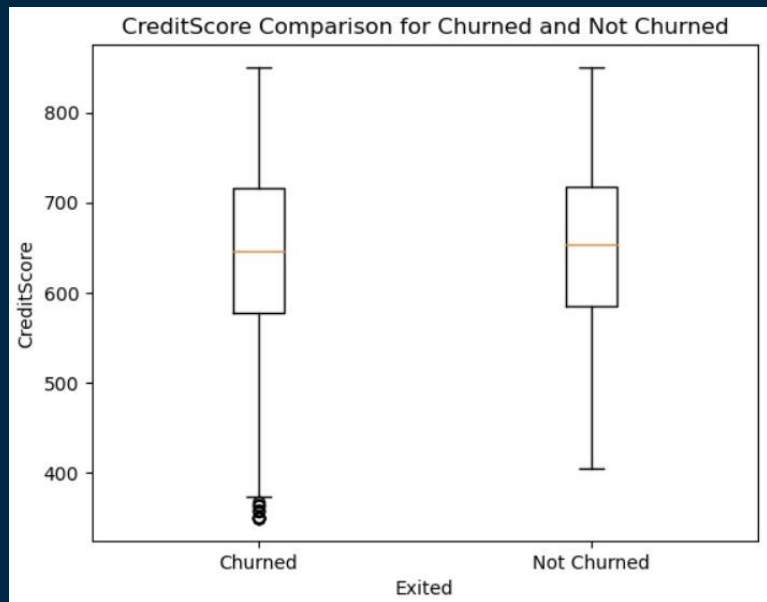
Method – Scaling (tried)

```
: 1 # from sklearn.preprocessing import StandardScaler
2 # standard = StandardScaler()
3
4 # churn_features_scaled_df = churn_features_transformed_df.copy()
5
6 # churn_features_scaled_df[
7 #     ['CreditScore',
8 #      'Age',
9 #      'Tenure',
10 #      'Balance',
11 #      'EstimatedSalary']] = standard.fit_transform(churn_features_transformed_df[['CreditScore', 'Age', 'Tenure', 'Balance', 'EstimatedSalary']])
12
```

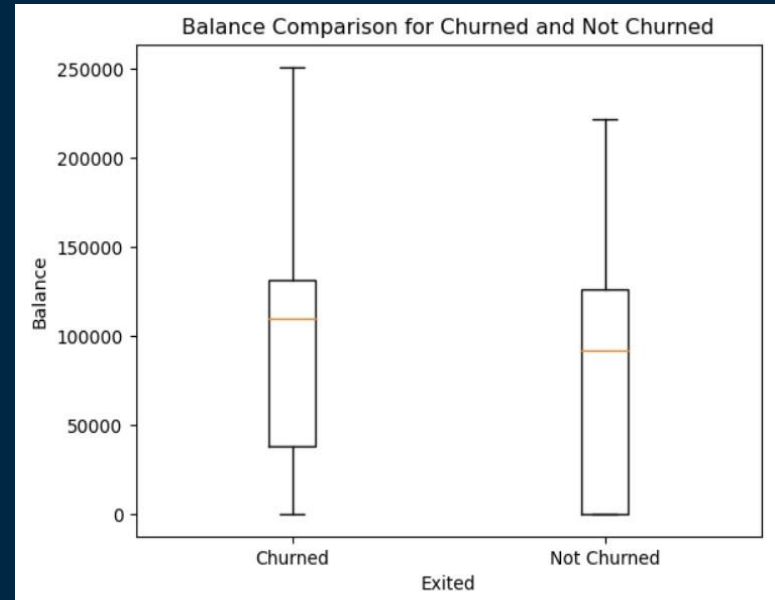
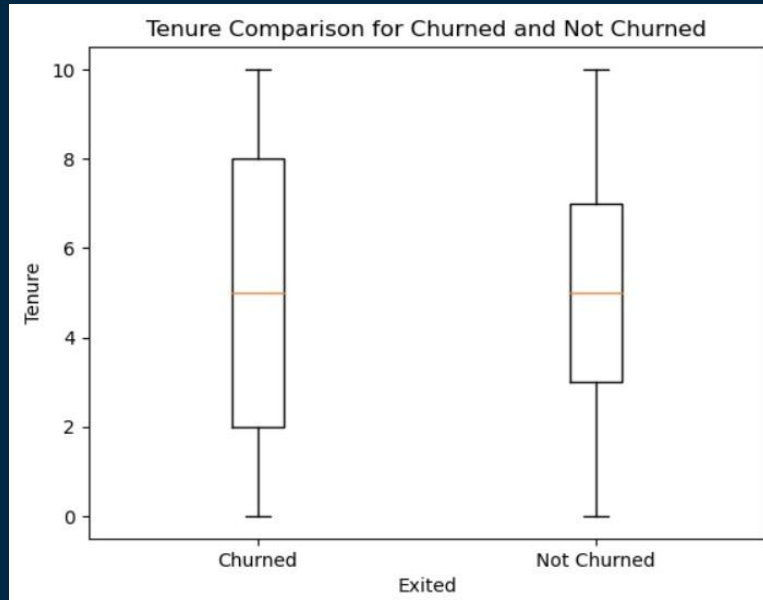
```
: 1 # from sklearn.preprocessing import MinMaxScaler
2
3 # # Create an instance of MinMaxScaler
4 # minmax = MinMaxScaler()
5
6 # # Specify the columns to be scaled
7 # columns_to_scale = ['CreditScore', 'Age', 'Tenure', 'Balance', 'EstimatedSalary']
8
9 # # Apply MinMaxScaler to the selected columns
10 # churn_features_scaled_df[columns_to_scale] = minmax.fit_transform(churn_features_scaled_df[columns_to_scale])
11
```

```
: 1 # churn_features_scaled_df.head()
```

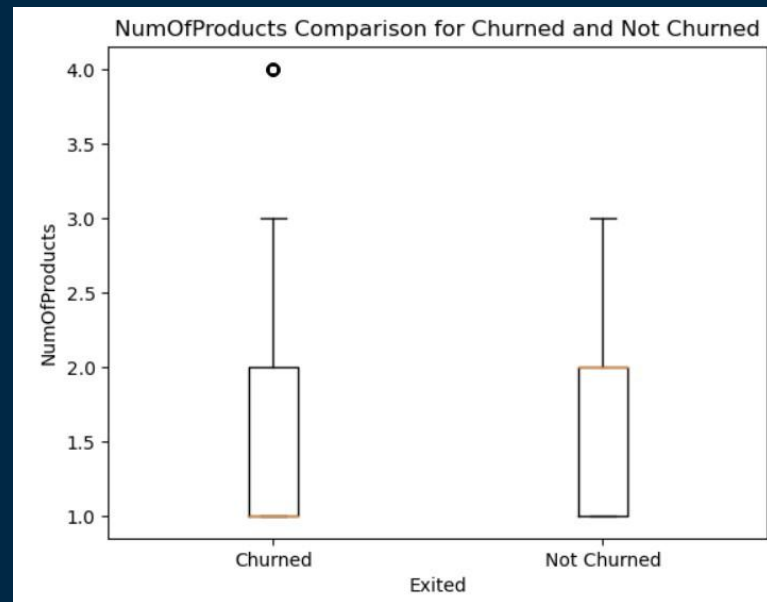
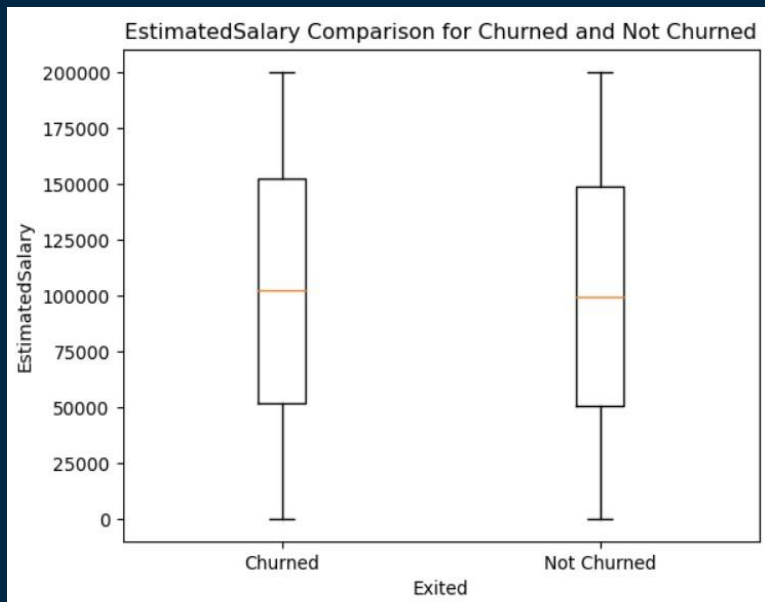
Method - Data Visualization



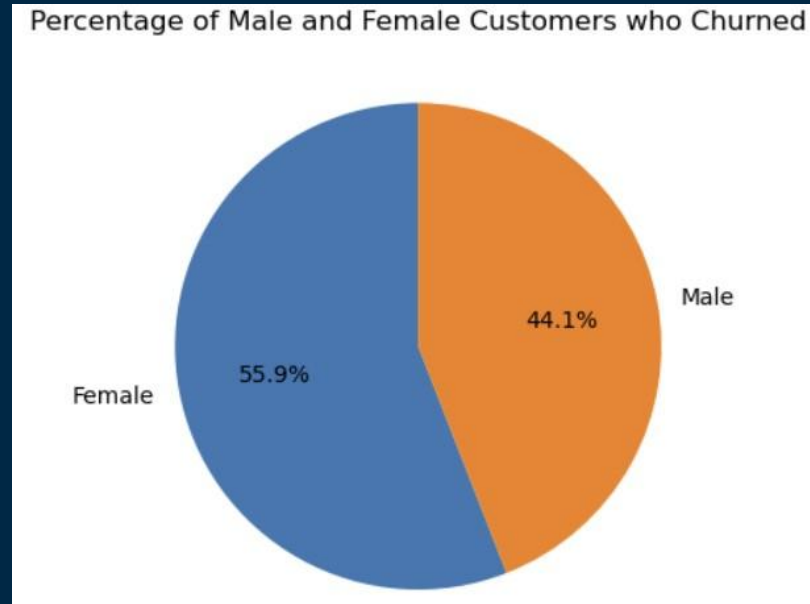
Method - Data Visualization



Method - Data Visualization



Method - Data Visualization



Method – Model Training

Training various classifiers

```
1  # Now let's define our models
2  from sklearn.linear_model import LogisticRegression
3  from sklearn.neural_network import MLPClassifier
4  from sklearn.tree import DecisionTreeClassifier
5  from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
6  from sklearn.metrics import accuracy_score
7
8
9  lr_classifier = LogisticRegression(solver='lbfgs',max_iter=10000)
10 mlp_classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
11                               hidden_layer_sizes=(8, 2), random_state=11,max_iter=50000)
12 dt_classifier = DecisionTreeClassifier()
13 grad_boost = GradientBoostingClassifier()
14 random_forest = RandomForestClassifier()
15
16
17 # train our models
18 lr_classifier.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
19 mlp_classifier.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
20 dt_classifier.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
21 grad_boost.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
22 random_forest.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
```

Method – Model Evaluating

Test.csv

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9499	15701932	Millar	586	France	Female	52	6	140900.97	1	1	1	67288.89	0
9500	15700813	Igwebuike	522	Germany	Female	25	5	94049.92	2	1	0	103269	0
9501	15645600	Obidimkpa	739	Spain	Female	27	8	98926.4	1	1	1	106969.98	0
9502	15634146	Hou	835	Germany	Male	18	2	142872.36	1	1	1	117632.63	0
9503	15686743	Moody	790	Spain	Male	29	3	46057.96	2	1	1	189777.66	0

Accuracy of the Logistic Classifier = 0.8021978021978022

Accuracy of the MLP Classifier = 0.8211788211788211

Accuracy of the Decision Tree Classifier = 0.7842157842157842

Accuracy of the Gradient Boosting Classifier = 0.8631368631368631

Accuracy of the Random Forest Classifier = 0.8641358641358642

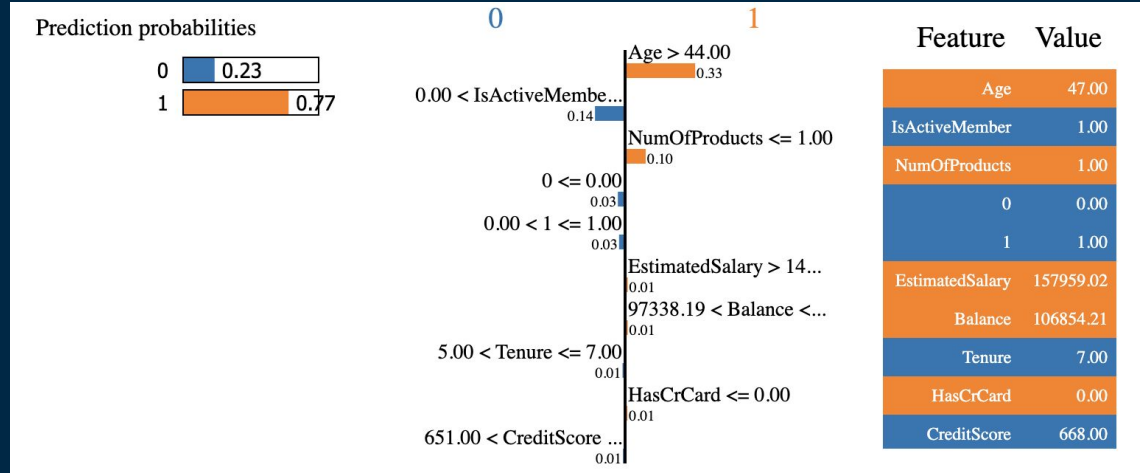
Method – Model Selection

Save the best model which is Random Forest Classifier.

```
1 import pickle
2
3 file_to_write = open("churn_best_model.saved", "wb")
4 pickle.dump(random_forest, file_to_write)
5 file_to_write.close()
```

Method – Model Evaluating

```
CreditScore = 668
Gender = "Male"
Age = 47
Tenure = 7
Balance = 106854.21
NumOfProducts = 1
HasCrCard = 0
IsActiveMember = 1
EstimatedSalary = 157959.02
```



Method – Model Bias and Fairness

```
43 def class_wise_acc(y_actual, y_predicted):
44     total_p = 0
45     total_n = 0
46     TP = 0
47     TN = 0
48     for i in range(len(y_predicted)):
49         if y_actual[i] == 1:
50             total_p += 1
51             if y_actual[i] == y_predicted[i]:
52                 TP += 1
53         if y_actual[i] == 0:
54             total_n += 1
55             if y_actual[i] == y_predicted[i]:
56                 TN += 1
57     return (TP / total_p, TN / total_n)
58
59 class_1_acc_male, class_0_acc_male = class_wise_acc(y_actual_male, y_predicted_male)
60 class_1_acc_female, class_0_acc_female = class_wise_acc(y_actual_female, y_predicted_female)
61
62 print(f"Class 1 (i.e., Churn) accuracy for Male = {class_1_acc_male}")
63 print(f"Class 0 (i.e., Not Churn) accuracy for Male = {class_0_acc_male}")
64 print(f"Class 1 (i.e., Churn) accuracy for Female = {class_1_acc_female}")
65 print(f"Class 0 (i.e., Not Churn) accuracy for Female = {class_0_acc_female}")
```

```
Class 1 (i.e., Churn) accuracy for Male = 0.36470588235294116
Class 0 (i.e., Not Churn) accuracy for Male = 0.9705263157894737
Class 1 (i.e., Churn) accuracy for Female = 0.4574468085106383
Class 0 (i.e., Not Churn) accuracy for Female = 0.9510086455331412
```

Result – Demo

Credit Score: 754

Age: 22

Tenure: 1

Balance: 81156.00

NumOfPro...: 1

☒ Has Credit Card

☐ Is Active Member

Est. Salary: 70000.00

Gender: Male

Predict Churn

```
import ipywidgets as widgets
from IPython.display import display


# Define input widgets
credit_score = widgets.IntSlider(min=300, max=850, description="Credit Score")
age = widgets.IntSlider(min=18, max=100, description="Age")
tenure = widgets.IntSlider(min=0, max=10, description="Tenure")
balance = widgets.FloatSlider(min=0, max=250000, description="Balance")
num_of_products = widgets.IntSlider(min=1, max=4, description="NumOfProducts")
has_cr_card = widgets.Checkbox(value=True, description="Has Credit Card")
is_active_member = widgets.Checkbox(value=True, description="Is Active Member")
estimated_salary = widgets.FloatSlider(min=0, max=200000, description="Est. Salary")
gender = widgets.Dropdown(options=["Male", "Female"], description="Gender")
```

Conclusion

```
4 # Load the trained model
5 model_file = open("churn_best_model.saved", "rb")
6 random_forest = pickle.load(model_file)
7 model_file.close()
8
9 # Prepare a sample input
10 CreditScore = 668
11 Gender = "Male"
12 Age = 47
13 Tenure = 7
14 Balance = 106854.21
15 NumOfProducts = 1
16 HasCrCard = 0
17 IsActiveMember = 1
18 EstimatedSalary = 157959.02
19
20 # Perform one-hot encoding for the "Gender" feature
21 gender_transformed = {"Female": [1, 0], "Male": [0, 1]} # Update with the mapping used during training
22 Gender = gender_transformed[Gender]
23
24 # Create a numpy array of the input data, including the one-hot encoded "Gender" feature
25 input_data = np.array([[CreditScore, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary,
26
27 # Predict using the trained decision tree classifier
28 y_predicted = random_forest.predict(input_data)
29
30 # Interpret the prediction result
31 if y_predicted[0] == 1:
32     print("The person is likely to churn")
33 elif y_predicted[0] == 0:
34     print("The person is not likely to churn")
35 else:
36     print("Invalid prediction")
37
```

The person is likely to churn

- Random Forest Classifier worked reasonably
- More accuracy needed
- Bank would be able to predict if the user will churn or not using this model
- There might be a bias in the model, so further investigation is needed
- This is a model to predict if a user will turn over or not, but we need a model to identify what is the reason they left in order to prevent the issue

The background is a dark blue gradient. It features several small, hollow squares in white, light blue, and light orange. Some of these squares are connected to thin vertical lines of the same color. The text is centered in a large, white, sans-serif font.

Thanks for Listening!
Any Question?