Churn for Bank Accounts

Team: **JPM**organ**S**

Jinwoo Oh, **M**injae Kwon,
Pranav Reddy, **S**rihith Duggi

TABLE OF CONTENTS

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

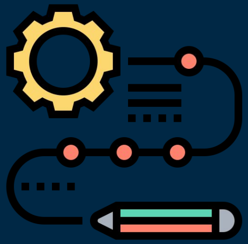
Goal – Predicting if the customers are likely to turn over

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
- **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 "Excited" column because it's a indicator that shows if the user actually left the bank or not.

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

```
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

```
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

```
gender_transformed_df = pd.DataFrame(gender_transformed)

churn_features_df.reset_index(drop=True, inplace=True)
gender_transformed_df.reset_index(drop=True, inplace=True)

churn_features_transformed_df = pd.concat([churn_features_df,gender_transformed_df], axis=1)

churn_features_transformed_df = churn_features_transformed_df.drop(columns=["Gender"], axis=1)

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

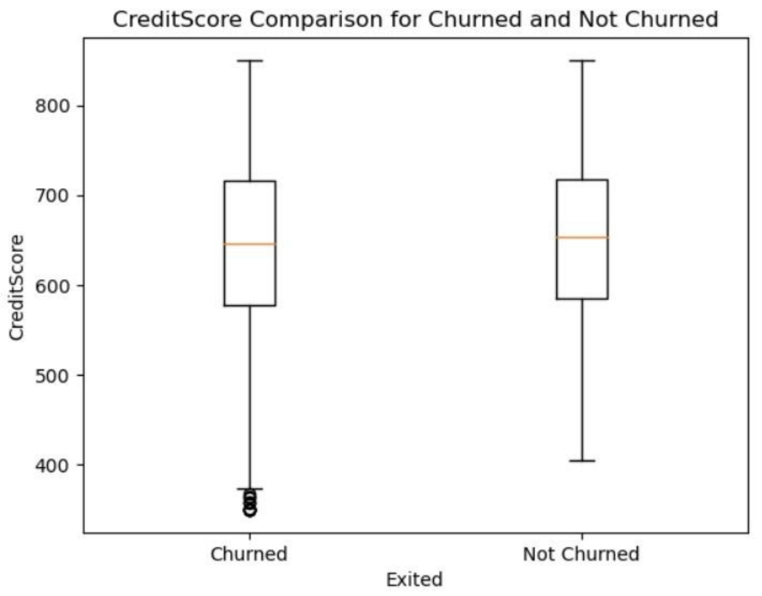
Method – Scaling (tried)

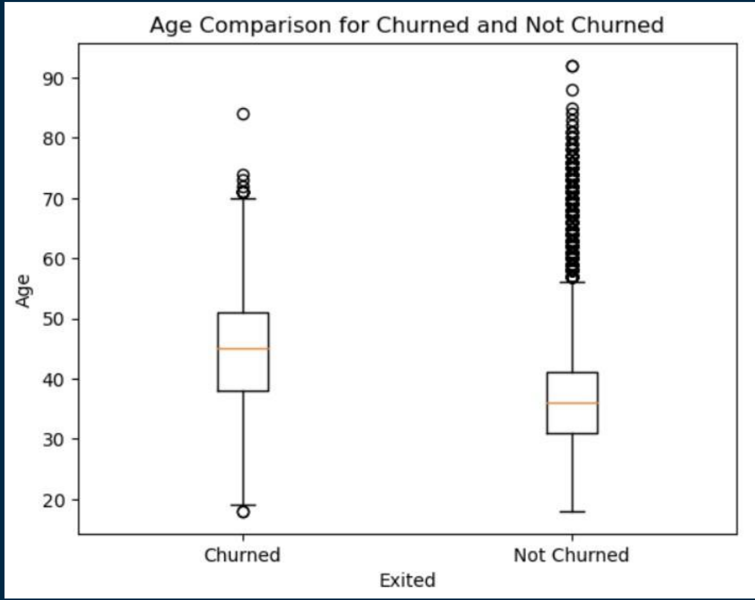
```
# from sklearn.preprocessing import StandardScaler
# standard = StandardScaler()

# churn_features_scaled_df = churn_features_transformed_df.copy()

# churn_features_scaled_df[
#     ['CreditScore',
#     'Age',
#     'Tenure',
#     'Balance',
#     'EstimatedSalary']] = standard.fit_transform(churn_features_transformed_df[['CreditScore', 'Age','Tenure','Bal

```

```
# from sklearn.preprocessing import MinMaxScaler

# # Create an instance of MinMaxScaler
# minmax = MinMaxScaler()

# # Specify the columns to be scaled
# columns_to_scale = ['CreditScore', 'Age', 'Tenure', 'Balance', 'EstimatedSalary']

# # Apply MinMaxScaler to the selected columns
# churn_features_scaled_df[columns_to_scale] = minmax.fit_transform(churn_features_scaled_df[columns_to_scale])

```

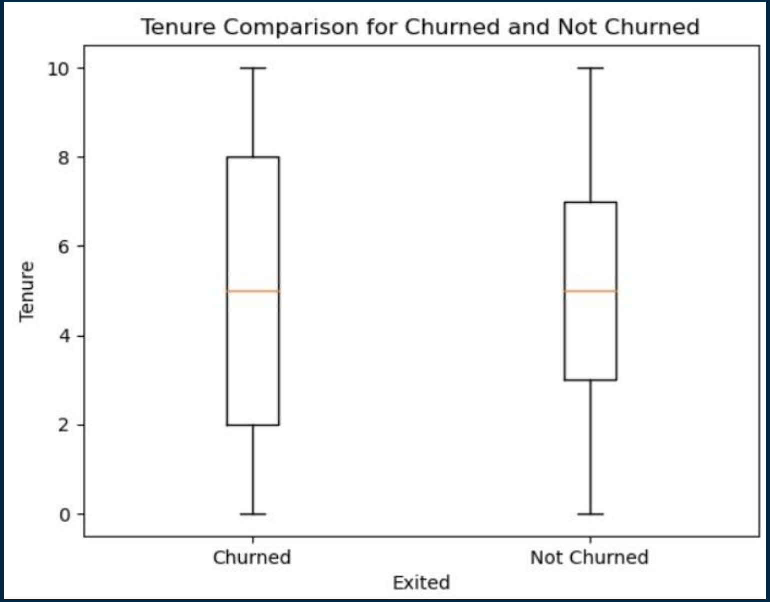
```
# churn_features_scaled_df.head()
```

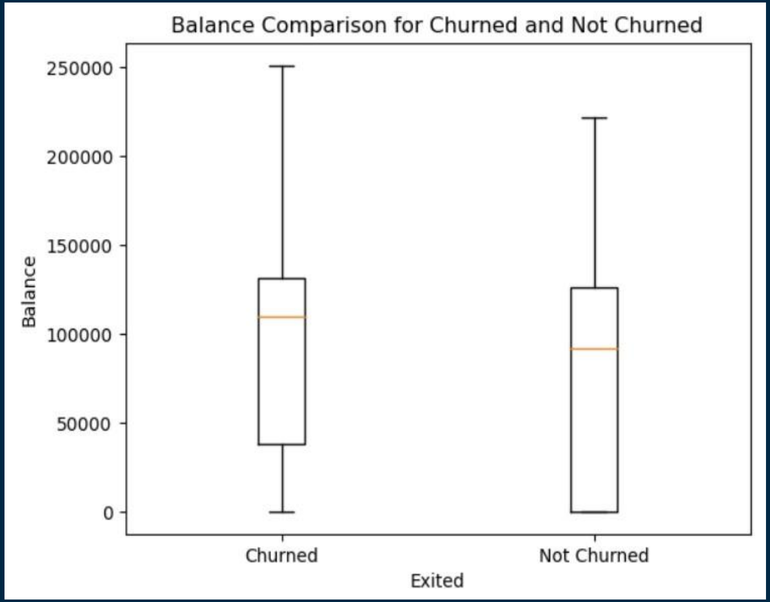
Method – Data Visualization



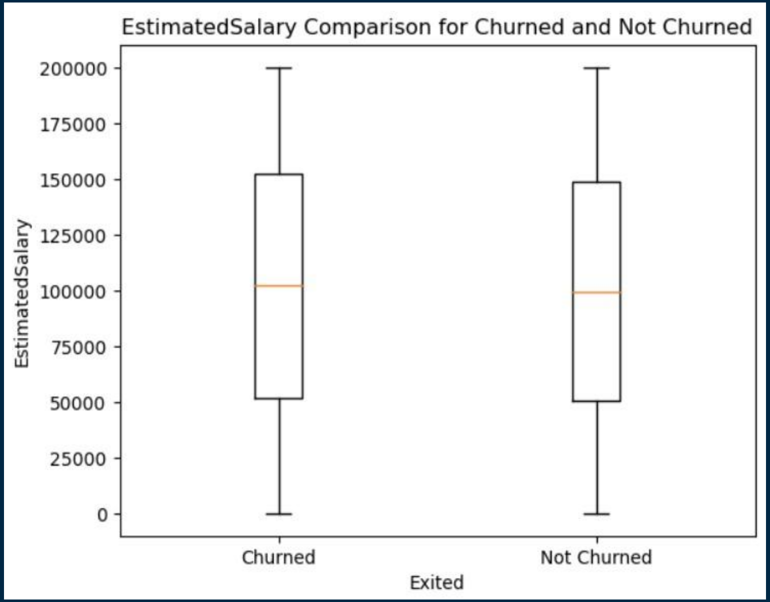


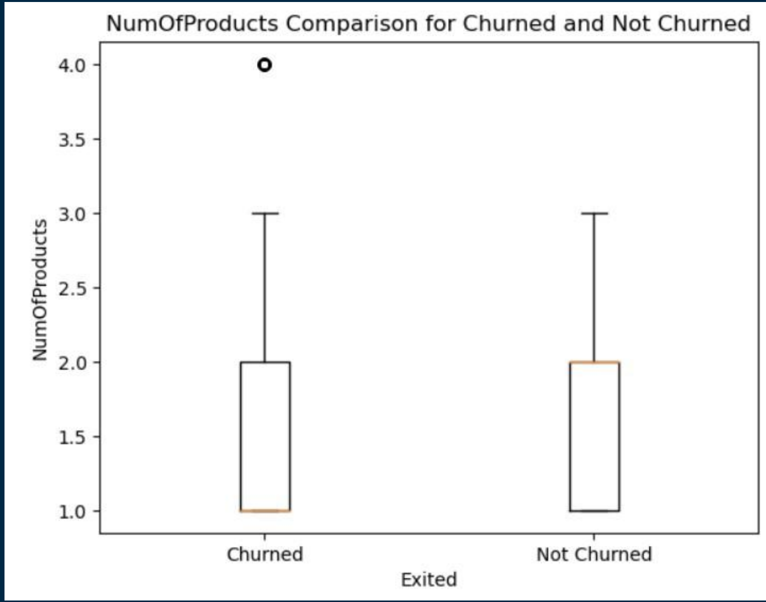
Method – Data Visualization



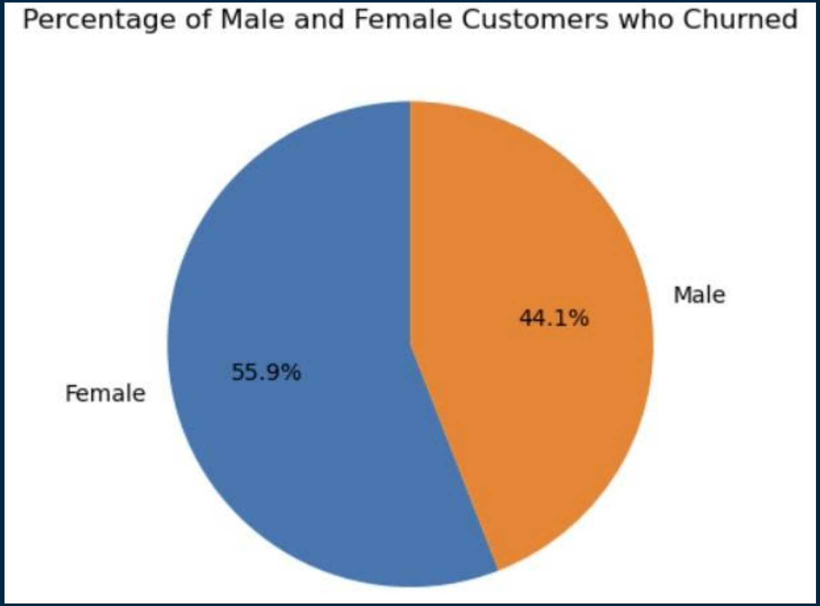


Method – Data Visualization





Method – Data Visualization



Method – Model Training

Training various classifiers

```python
# Now let's define our models
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.metrics import accuracy_score


lr_classifier = LogisticRegression(solver='lbfgs',max_iter=10000)
mlp_classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                               hidden_layer_sizes=(8, 2), random_state=11,max_iter=50000)
dt_classifier = DecisionTreeClassifier()
grad_boost = GradientBoostingClassifier()
random_forest = RandomForestClassifier()


# train our models
lr_classifier.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
mlp_classifier.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
dt_classifier.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
grad_boost.fit(churn_features_transformed_df.to_numpy(), churn_label_df.to_numpy().ravel())
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
```

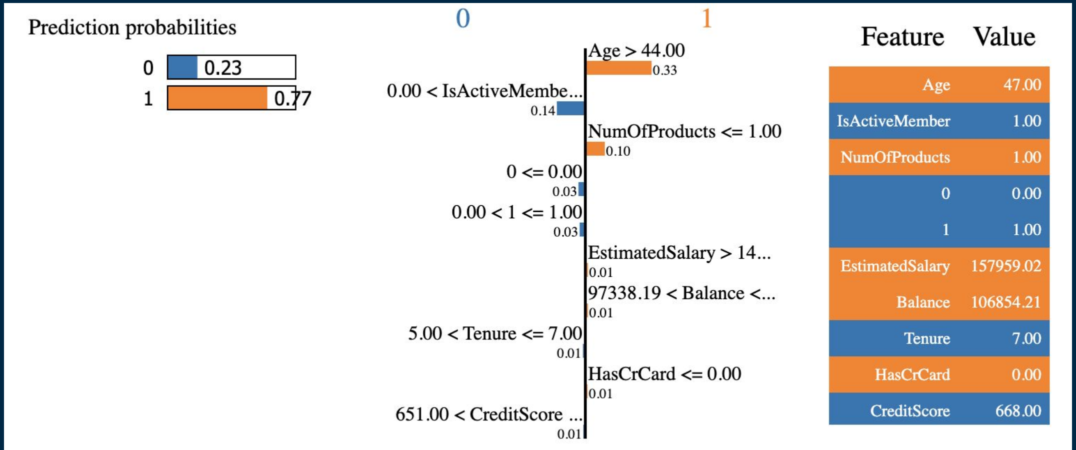
Method – Model Selection

Save the best model which is Random Forest Classifier.

```
import pickle

file_to_write = open("churn_best_model.saved","wb")
pickle.dump(random_forest,file_to_write)
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
```



- Credit Score > 651 -> Less likely churn
- Gender "Male" -> Less likely churn
- Age > 44 -> More likely churn
- 5 < Tenure < 7 -> less likely churn
- 97338.19 < Balance -> more likely churn
- NumofProducts <= 1 -> more likely churn
- HasCrcard = 0 -> more likely churn
- IsActiveMemebrr = 1 -> Less likely churn
- EstimatedSalary > 140000 -> more likely churn

Method – Model Bias and Fairness

```python
def class_wise_acc(y_actual, y_predicted):
    total_p = 0
    total_n = 0
    TP = 0
    TN = 0
    for i in range(len(y_predicted)):
        if y_actual[i] == 1:
            total_p += 1
            if y_actual[i] == y_predicted[i]:
                TP += 1
        if y_actual[i] == 0:
            total_n += 1
            if y_actual[i] == y_predicted[i]:
                TN += 1
    return (TP / total_p, TN / total_n)

class_1_acc_male, class_0_acc_male = class_wise_acc(y_actual_male, y_predicted_male)
class_1_acc_female, class_0_acc_female = class_wise_acc(y_actual_female, y_predicted_female)

print(f"Class 1 (i.e., Churn) accuracy for Male = {class_1_acc_male}")
print(f"Class 0 (i.e., Not Churn) accuracy for Male = {class_0_acc_male}")
print(f"Class 1 (i.e., Churn) accuracy for Female = {class_1_acc_female}")
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
```

- Better accuracy for Class 1 is needed
- There can be bias between Male and Female as we see the 10% difference

Result – Demo


```python
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

```python
# Load the trained model
model_file = open("churn_best_model.saved", "rb")
random_forest = pickle.load(model_file)
model_file.close()

# Prepare a sample input
CreditScore = 668
Gender = "Male"
Age = 47
Tenure = 7
Balance = 106854.21
NumOfProducts = 1
HasCrCard = 0
IsActiveMember = 1
EstimatedSalary = 157959.02

# Perform one-hot encoding for the "Gender" feature
gender_transformed = {"Female": [1, 0], "Male": [0, 1]}  # Update with the mapping used during training
Gender = gender_transformed[Gender]

# Create a numpy array of the input data, including the one-hot encoded "Gender" feature
input_data = np.array([[CreditScore, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalar

# Predict using the trained decision tree classifier
y_predicted = random_forest.predict(input_data)

# Interpret the prediction result
if y_predicted[0] == 1:
    print("The person is likely to churn")
elif y_predicted[0] == 0:
    print("The person is not likely to churn")
else:
    print("Invalid prediction")
```

```
The person is likely to churn
```

- More data needed to be collected because of the imbalance between churn and not churn data
- 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
- Need additional model with another dataset to identify what is the reason they left in order to prevent the issue

Thanks for Listening!

Any Question?

Github Link: https://github.com/starJin2003/Churn_for_Bank_Customers_Analysis.git