# AI Project: Evaluating Bank Customer Churn

## Project Goals and Business Value

The aim of this project is to predict customer churn, a common business key performance indicator (KPI). Churn is when customers stop using a product, and reducing it is crucial for business success. Using AI to understand factors that influence churn can help define effective business strategies, and it is also one of the goals of the project.

We will implement a machine learning algorithm, Logistic Regression to perform Binary Classification to determine the 'churn status' of a customer given a set of features. Additionally, we will compute the Shapley values for our model in an attempt to define what features have the most impact in influencing a customer's churn status.

## Dataset

For our project we will be using the 'Bank Customer Churn Dataset' which is freely available in Kaggle. The dataset is mock customer data of account holders from the fictional **ABC Multinational Bank** containing around 10,000 samples and 11 features. These features include variables such as: Credit Score, Country of Residence, Gender, Tenure (Years having an account in the bank), etc.

We decided to split our data into a training, validation and test set in the following percentages: 70:10:20, respectively.

## Project Management Framework

For our project management methodology we have opted to go to a standard Agile methodology focusing on Scrums. We will prioritise features in our development, and work on the most important features first, and adapt if necessary.

## Team Organization and Roles

Our team is consisted of 2 people only, Martin and Sameera. Considering we are both AIS students with multi-disciplinary skills we have not defined strict roles to each other. Additionally, due to the size of the team and ease of communication we discuss and take our project decisions together.

# Technical Specifications

The entirety of this project can be found on BankCustomerChurn Github repository. For our project organization we have implemented the template cookie-cutter found here. We have used a conda environment for our libraries, and have created the appropriate *'requirements.txt'* file on the root of our repository to ease this installation.

Due to the context of the problem we knew we had to implement Logistic Regression algorithm, and could use a variation of this model. You can find our exploratory notebook here 1.0-initial-data-exploration.ipynb. We explored many models but our best performance was achieved by passing our features through a polynomial transformation of power 3. We chose a best performing model, and we industrialized our code by the usage of mlflow.

**Managing ML Development Cycle using MLFlow**

For managing the entire Machine Learning development cycle, we used MLFlow. It is an open-source platform streamline the process of experimentation, tracking, deploying and collaboration.

In our ML model development with logistic regression, we experimented and tracked by logging the parameters for "regularization_param". We reproduced the experiments and compared our model with different values of "regularization_param" like [100, 10, 1, 0.1, 0.01, 0.001, 0.0001]. The performance metrics we analysed are accuracy, precision, and f1-score.

The different experiments we did can be found in "runs.csv" located at "BankCustomerChurn/"

**C in the x-axis – regularisation value**

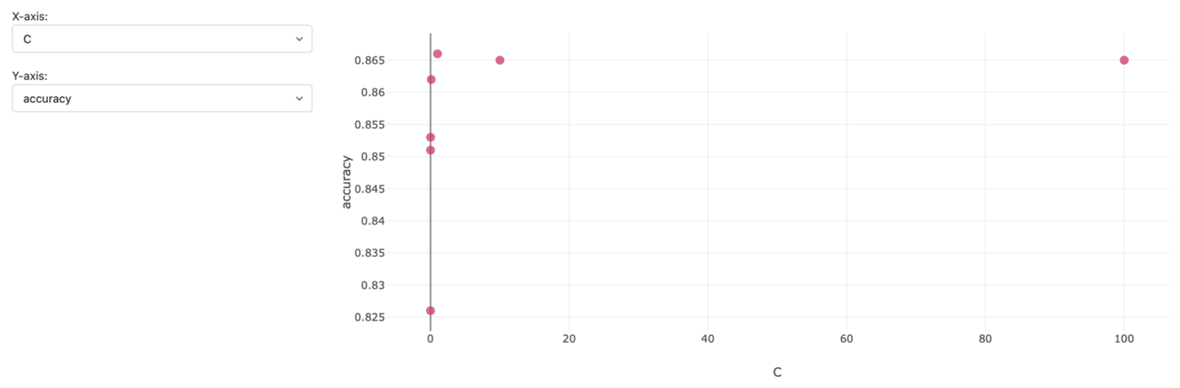


*Figure 1: reg vs accuracy*

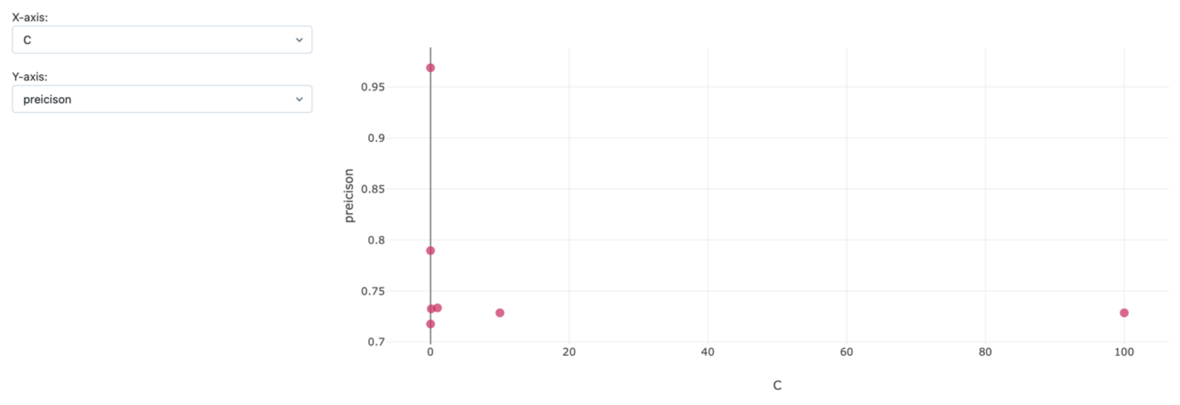


*Figure 2: reg vs precision*

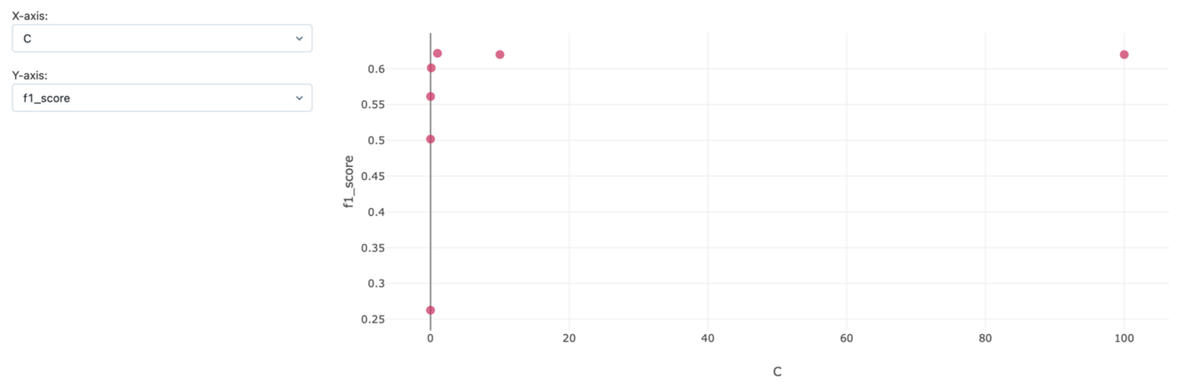


*Figure 3: reg vs f1_score*

From all the above metrics, we analysed and concluded that the model works best when the "regularisation_param" is **1.0**

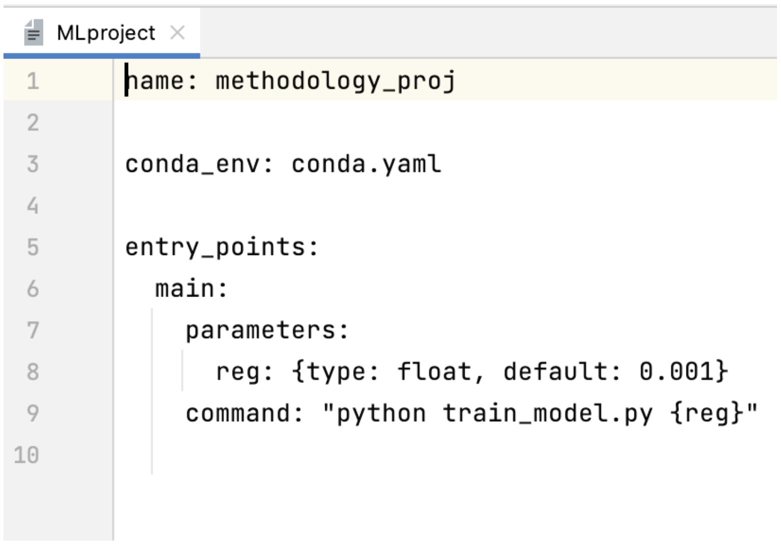
## <u>Creating MLFlow projects</u>

The ML model is packaged along with its dependencies, so that it can be reused, easily shared, and deployed. Also, the packaged model can be installed and run independently of the original training environment and perform consistently over time.

There are 2 files for we need to add for doing this process

- The "MLproject" file: It tells us where to look for the dependencies.
- The "conda.yaml" file: It specifies all the dependencies we need to run our project.

The files are located at the "src/models/<file>" path

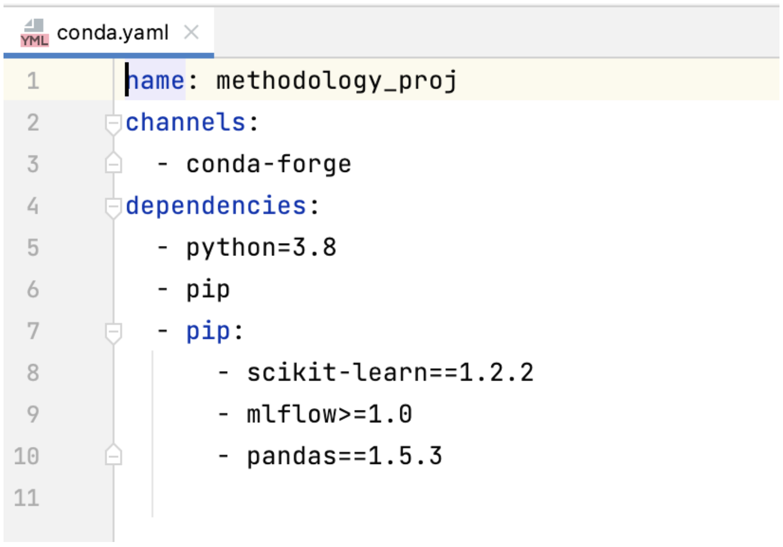

MLproject

```yaml
name: methodology_proj

conda_env: conda.yaml

entry_points:
  main:
    parameters:
      reg: {type: float, default: 0.001}
    command: "python train_model.py {reg}"
```

*Figure 4: MLproject file*

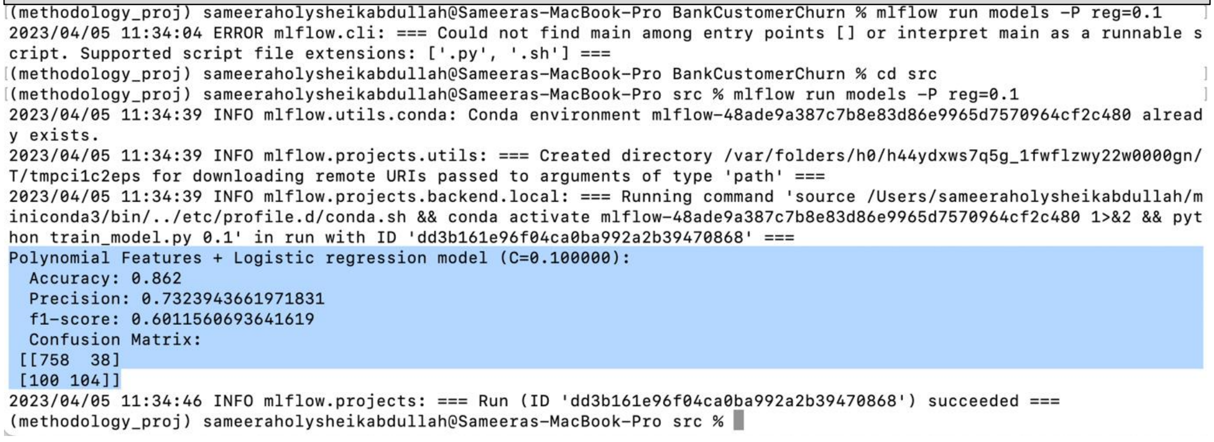

conda.yaml

```yaml
name: methodology_proj
channels:
  - conda-forge
dependencies:
  - python=3.8
  - pip
  - pip:
      - scikit-learn==1.2.2
      - mlflow>=1.0
      - pandas==1.5.3
```

*Figure 5: conda.yaml*

To run this project, go to "BankCustomerChurn/src" and type the following command with different values of reg.

```
mlflow run models -P reg=0.1
```



```
(methodology_proj) sameeraholysheikabdullah@Sameeras-MacBook-Pro BankCustomerChurn % mlflow run models -P reg=0.1
2023/04/05 11:34:04 ERROR mlflow.cli: === Could not find main among entry points [] or interpret main as a runnable s
cript. Supported script file extensions: ['.py', '.sh'] ===
(methodology_proj) sameeraholysheikabdullah@Sameeras-MacBook-Pro BankCustomerChurn % cd src
(methodology_proj) sameeraholysheikabdullah@Sameeras-MacBook-Pro src % mlflow run models -P reg=0.1
2023/04/05 11:34:39 INFO mlflow.utils.conda: Conda environment mlflow-48ade9a387c7b8e83d86e9965d7570964cf2c480 alread
y exists.
2023/04/05 11:34:39 INFO mlflow.projects.utils: === Created directory /var/folders/h0/h44ydxws7q5g_1fwflzwy22w0000gn/
T/tmpci1c2eps for downloading remote URIs passed to arguments of type 'path' ===
2023/04/05 11:34:39 INFO mlflow.projects.backend.local: === Running command 'source /Users/sameeraholysheikabdullah/m
iniconda3/bin/../etc/profile.d/conda.sh && conda activate mlflow-48ade9a387c7b8e83d86e9965d7570964cf2c480 1>&2 && pyt
hon train_model.py 0.1' in run with ID 'dd3b161e96f04ca0ba992a2b39470868' ===
Polynomial Features + Logistic regression model (C=0.100000):
  Accuracy: 0.862
  Precision: 0.7323943661971831
  f1-score: 0.6011560693641619
  Confusion Matrix:
 [[758  38]
 [100 104]]
2023/04/05 11:34:46 INFO mlflow.projects: === Run (ID 'dd3b161e96f04ca0ba992a2b39470868') succeeded ===
(methodology_proj) sameeraholysheikabdullah@Sameeras-MacBook-Pro src %
```

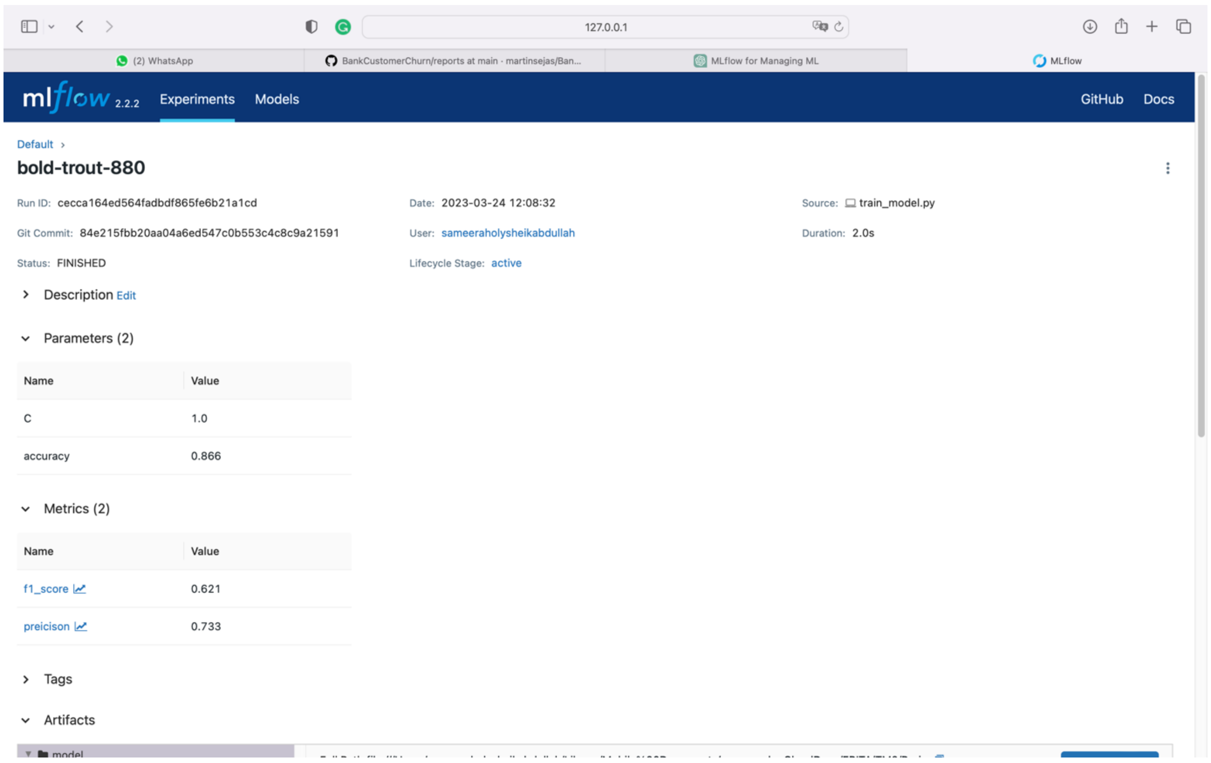
*Figure 6: mlflow run*

To run this project directly from GitHub, type the following command

```
mlflow run git@github.com:martinsejas/BankCustomerChurn.git -P reg=0.1
```

## MLFlow models

We used the log_model( ) to save the model. They can be found in the "mlruns/0/" folder, where each subdirectory represents different runs.



**bold-trout-880**

Run ID: cecca164ed564fadbdf865fe6b21a1cd
Date: 2023-03-24 12:08:32
Source: train_model.py
Git Commit: 84e215fbb20aa04a6ed547c0b553c4c8c9a21591
User: sameeraholysheikabdullah
Duration: 2.0s
Status: FINISHED
Lifecycle Stage: active

› Description Edit

⌄ Parameters (2)

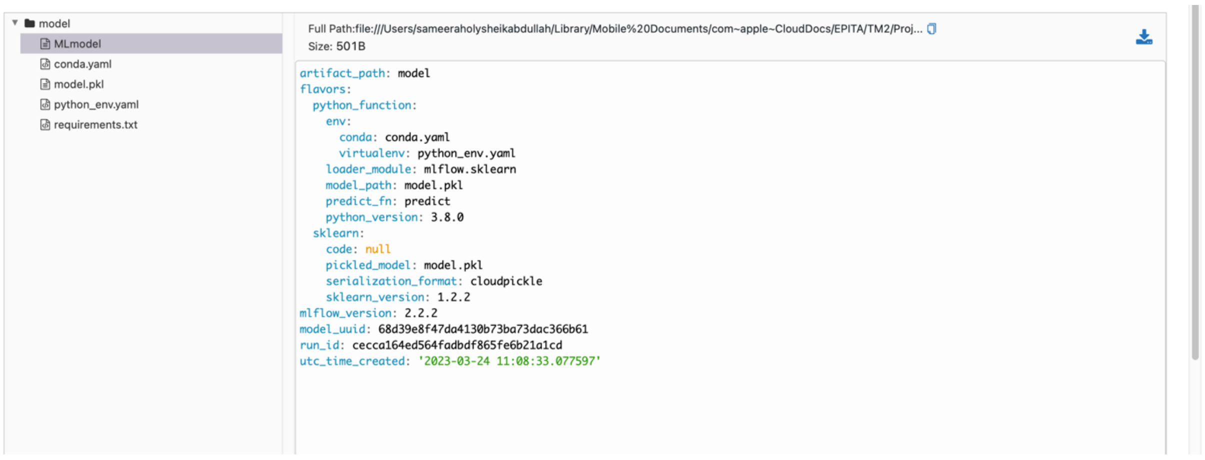
| Name | Value |
|---|---|
| C | 1.0 |
| accuracy | 0.866 |

⌄ Metrics (2)

| Name | Value |
|---|---|
| f1_score | 0.621 |
| preicison | 0.733 |

› Tags

⌄ Artifacts



Full Path:file:///Users/sameeraholysheikabdullah/Library/Mobile%20Documents/com~apple~CloudDocs/EPITA/TM2/Proj...
Size: 501B

```
artifact_path: model
flavors:
  python_function:
    env:
      conda: conda.yaml
      virtualenv: python_env.yaml
    loader_module: mlflow.sklearn
    model_path: model.pkl
    predict_fn: predict
    python_version: 3.8.0
  sklearn:
    code: null
    pickled_model: model.pkl
    serialization_format: cloudpickle
    sklearn_version: 1.2.2
mlflow_version: 2.2.2
model_uuid: 68d39e8f47da4130b73ba73dac366b61
run_id: cecca164ed564fadbdf865fe6b21a1cd
utc_time_created: '2023-03-24 11:08:33.077597'
```

## MODEL PREDICTION WITH THE TEST DATA

### Test splitting the dataset

Using the train_test_split from sklearn.model_selection, we splitted the test dataset with 10% of the whole data.

The test dataset can be found at “project-outputs/test_dataset.csv”

### Saving the Encoder, Scaler and polynomial model using Joblib

With the help of joblib, we saved the fitted encoder, standard Scaler and polynomial model, which can be later used for prediction.

```
joblib.dump(model_name, “save_model”)
```

### Saving the best MLFlow model

The best MLFlow model we have is with regularisation parameter 1.0. By using the following code in the train_model.py, we can save the model and use it for prediction.

```
mlflow.sklearn.save_model(poly_reg_model, “model_logistic”)
```

It creates a folder with dependencies files both for pyenv and conda, MLmodel, and model.pkl file.

The saved model can be found at “src/models/model_logistic”

### Model prediction with MLFlow

The test data is imported, and all the saved models are loaded using Joblib to preprocess the data before we do the prediction.

```
joblib.load(‘model_name’)
```

The saved mlflow model is loaded using mlflow.sklearn

```
mlflow.sklearn.load_model(‘model_logistic’)
```

The loaded model is used to predict the preprocessed data and it is stored in “predict_output.csv” which is in “project_outputs” directory.

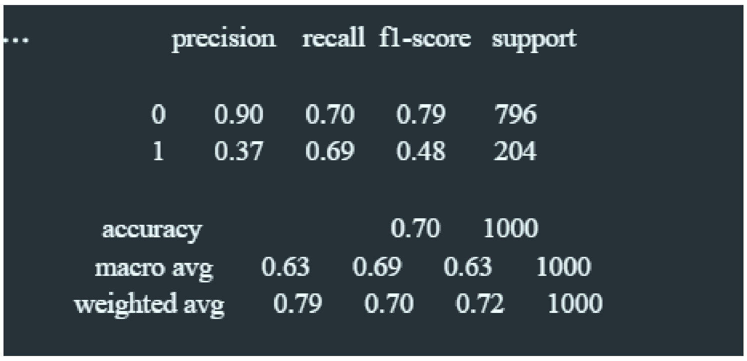
## Extracting Explanations from our Machine Learning Model using Shapley values.

*This report is extracted from the notebook* 1.1-exploring-shapley-values.ipynb *under /notebooks/ in the repository.*

After exploring 5 different machine learning logistic regression models in  notebook *1.0-initial-data-exploration.ipynb* our we have decided to go for the a LinearRegression model, with varying alphas (C) which is the model that we will be using to investigate the shapley values.

The model chosen is not the best performing model, the best performing model is a complex linear regression with features to the power of 3. This made it very difficult to analyze the most impactful features in our model. We know it is a regression problem, so we decided to scale it back to a standard linear regression to aid our analysis.

The performing metrics for this model is the following:



```
...              precision    recall  f1-score   support

           0       0.90      0.70      0.79       796
           1       0.37      0.69      0.48       204

    accuracy                           0.70      1000
   macro avg       0.63      0.69      0.63      1000
weighted avg       0.79      0.70      0.72      1000
```

*Figure 7- Performance metrics of simple logistic regression binary classifier*

With 0 and 1 representing churn.

Shapley values are useful to see the impact of each feature, and to break down how a prediction is made. We will be using the `shap` library to implement this.

It is important to note this datasets have been concatenated after being transformed. The shap plots will output the features with names such as: Feature 0, Feature 1, Feature 2... but they preserve the order from the original input. Hence we can simply map them here:

**Feature Description**

*Feature 0: "Credit Score"*

*Feature 1: "Age"*

*Feature 2: "Tenure"*

*Feature 3: "Balance"*

*Feature 4: "products_number"*

*Feature 5: "credit_card"*

*Feature 6: "active_member"*

*Feature 7: "estimated_salary"*

*Feature 8: "Credit Card"*

*Feature 9: "Gender (M)"*

*Feature 10: "Gender (F)"*

*Feature 11: "Country: France"*

*Feature 12: "Country: Germany"*

*Feature 13: "Country: Spain"*

If we visualize the Shapley values for one prediction of the dataset we can see:

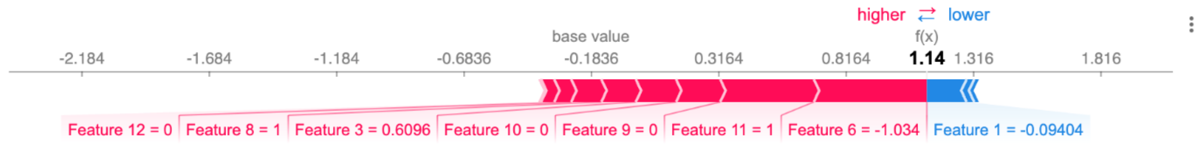


*Figure 8- Shapley values for one prediction*

Here we can see that for this particular instance, *Feature 6*: *"active member"* was the leading factor in deciding customer churn, in addition to the country they belonged to *(Feature 11 [France]),* indicating the bank can perform better in some countries, or has less competition in certain countries increasing their churn rate. It seems like the bank balance, or the amount of money in the bank *(Feature 3)* also increased the churn rate. Also seems like in this particular case, age negatively impacted customer retention *(Feature 1).*

If we continue and visualize for all points against dataset, we can also detect what appears to be the most meaningful feature, and that is '*Feature 1'* which is 'Age', and we can clearly see that the older a customer is, the more likely he/she is on staying a loyal customer of the bank. While the reverse is true, the younger the customer, the more likely they are of switching banks.

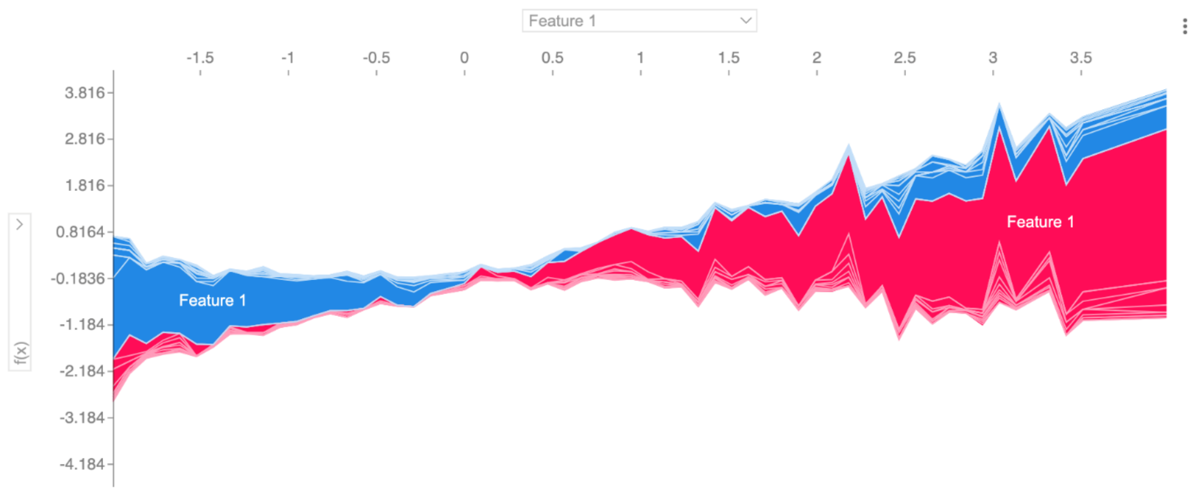


*Figure 9 - Impact of most dominant feature (age) in customer churn. Blue represents negative impact (no customer retention), red is positive impact (customer retention)*

If we plot a summarized plot for each class on the whole dataset, we can see which features are the most relevant, and how their values (negative to positive) impact the final prediction.

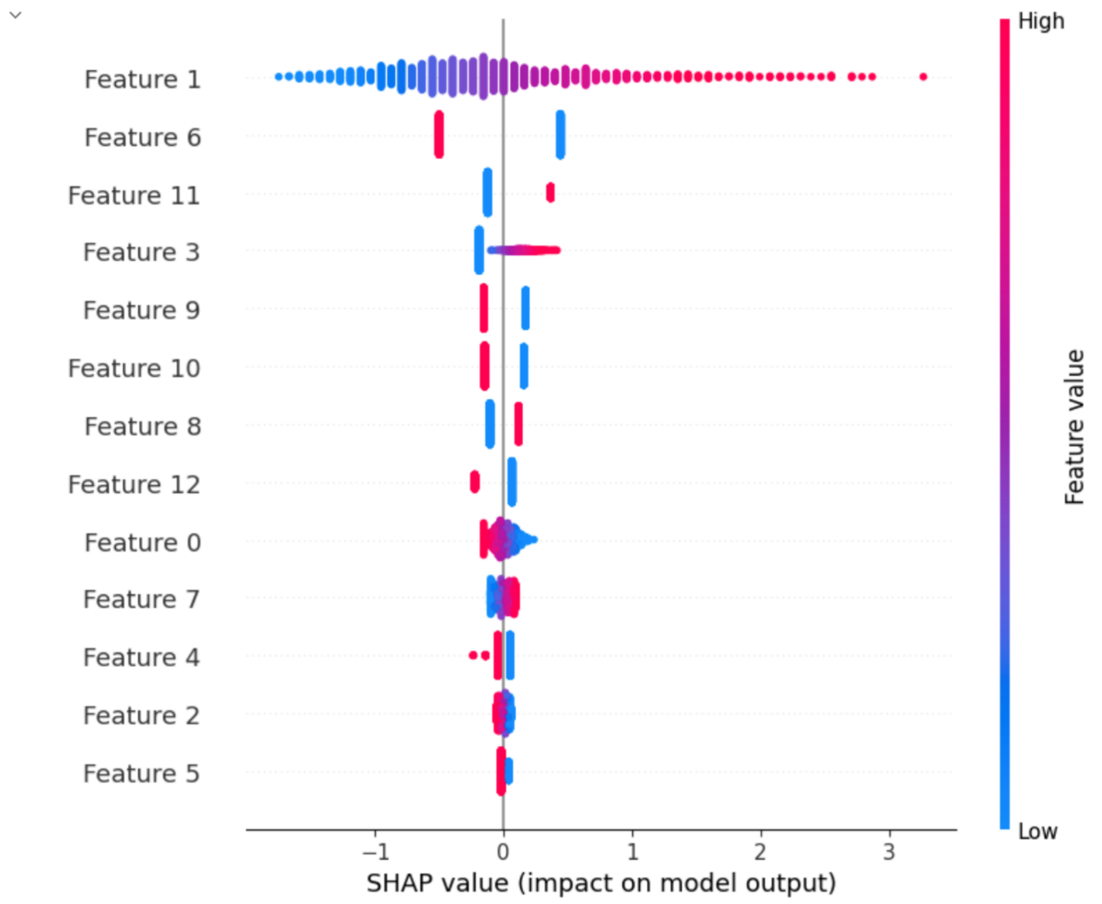


*Figure 10- Summarized Plot for all features. We can see age (Feature 1), customer activity (Feature 6) , and if they are based in France (Feature 11) are key contributors for churn. Additionally we can see expected human*

*behaviour on churn prediction, for example on Feature 0 (Credit Score) we can see that customers with lower credit scores are more likely to stay as customers with the bank, likely because it is harder for them to open new accounts.*

Shapley values were invaluable to find the factors that led to customers trends, not only that it also helped detect in which countries were performing better, in this case France. This analysis could allow the company to dictate region level policy, and target specific customer groups to increase its services. It would also be very valuable to conduct a Shapely value analysis per country, and discover what features are the most impactful at a regional level, as customer culture will affect the churn, which can help scope the bank's strategy for the country.

In conclusion, we can see here that the implementation of machine learning, and extracting understanding of what is influencing customer churn could be a great asset to define corporate strategy, and would provide great ROI to the company. Hence, our project goals were met, and the project has been a great success.