AZZURRO - BREAKAGE MODEL - EDA & ML EXERCISE

DR. DAVE SHAW

CONTENT

OVERVIEW

Use the dataset provided to conduct an Exploratory Data Analysis, and create a Machine Learning model which aims to predict Breakage.

A fully contained end-to-end solution has been created to address the problem. In addition, the solution is extensible to other ML problems and datasets.

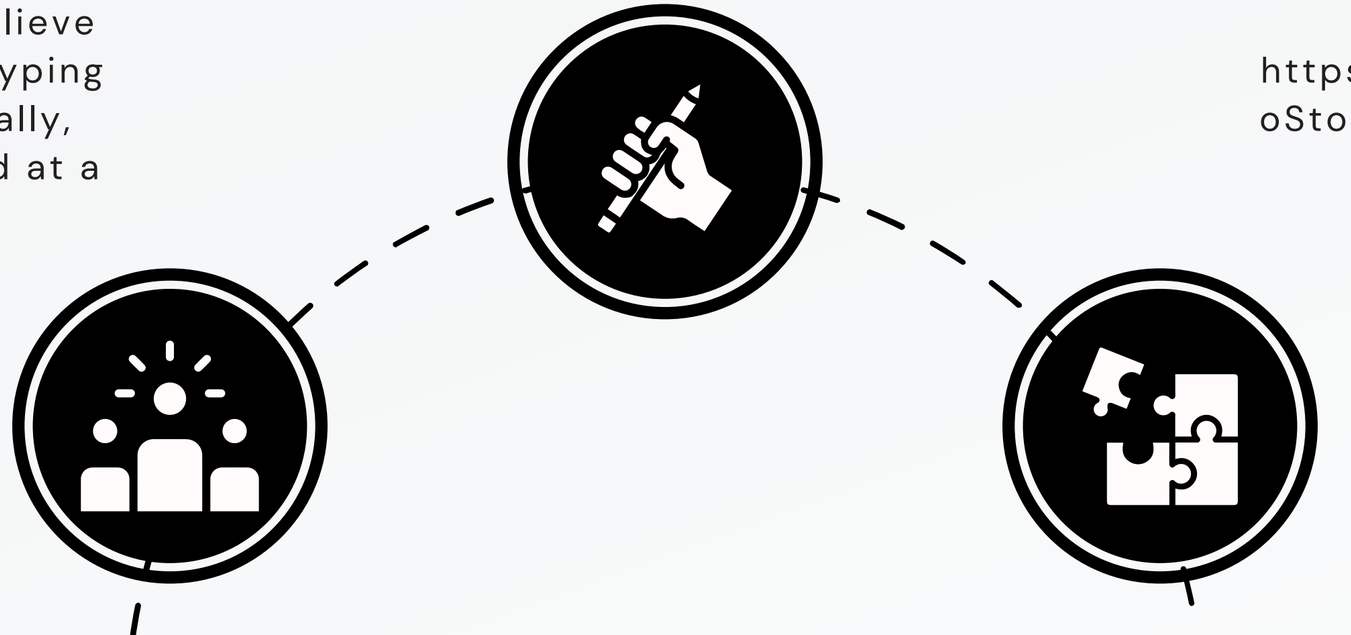
CREATE ENVIRONMENT & REPOSITORY

Use VSCode

For this task, I have chosen to use VSCode, running locally on my machine. I appreciate that cloud options are available, but I believe that model prototyping is done best locally, and then exported at a later time.

Create Env

I have created a new Anaconda environment, using the minimum required packages for the solution. This env can be easily recreated using the .yaml file stored.

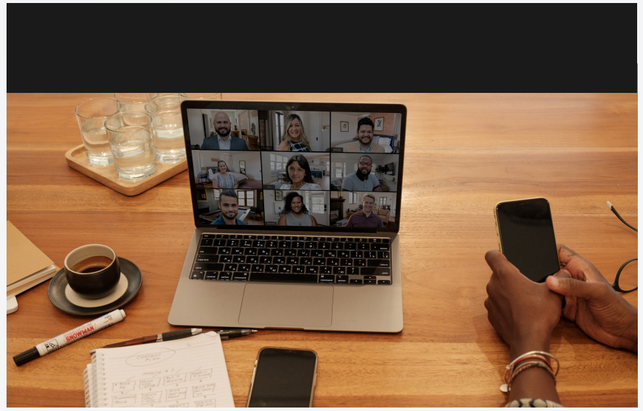


Store Using Git

The entire solution (excluding data) has been uploaded to a public GitHub repository.

https://github.com/RoboStock1985/Azzurro_Assessment

EXPLORATORY DATA ANALYSIS

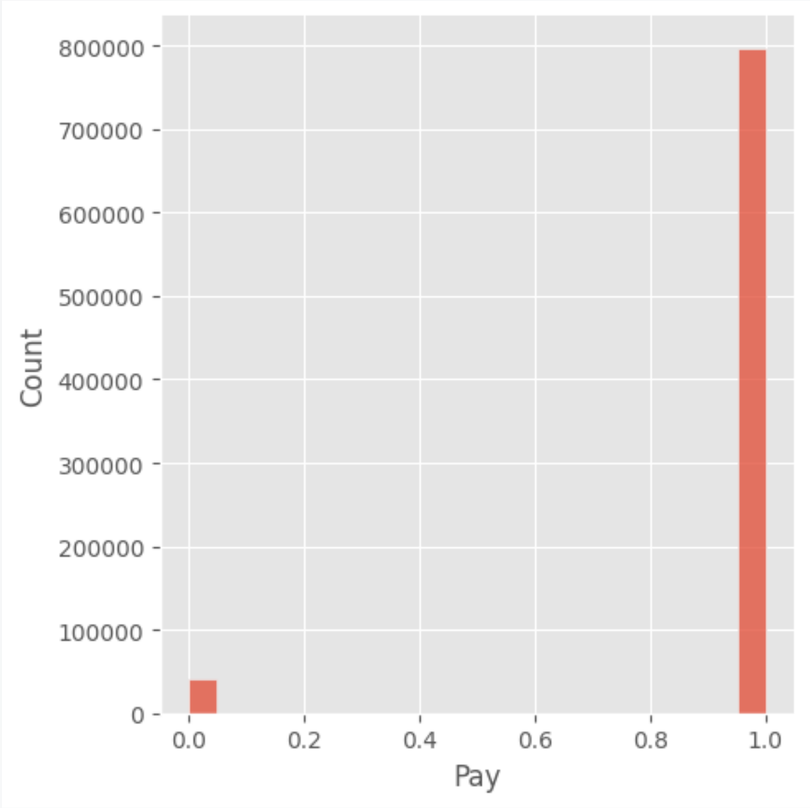


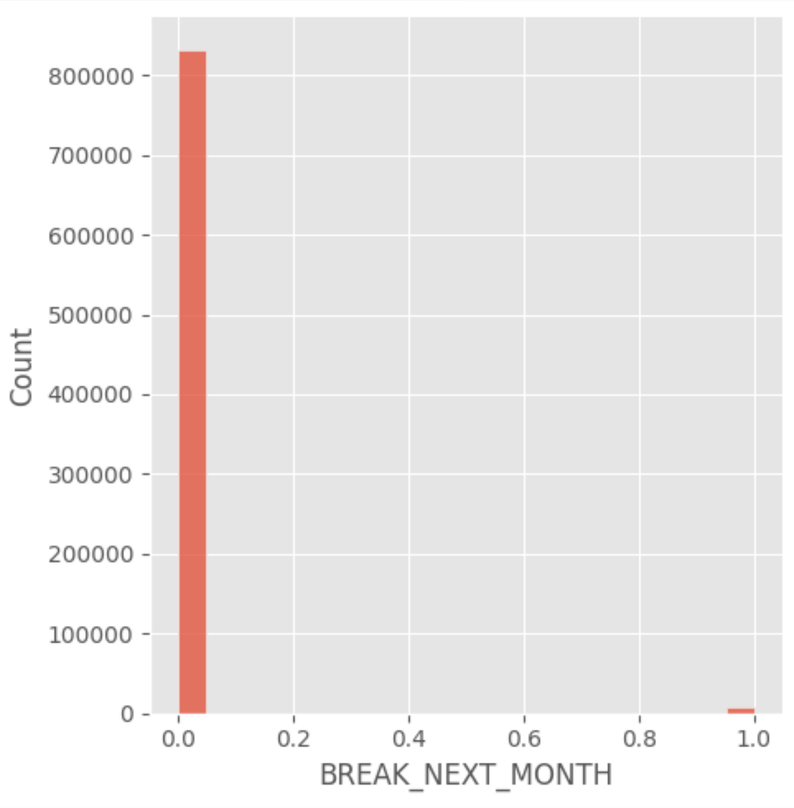
- The full code is available for this solution, either directly or from the public GitHub repository.
- The following steps were included as part of the Exploratory Data Analysis and Data Cleaning phases.

- Load the data. The data was provided in the form of a 363Mb .csv file. This was separated using the tab delimiter.
- Examine the head of the data.
- Remove duplicate rows.
- Further examine the shape of the data and data types.
- Reformat Month Columns and create dummies.
- Remove columns which only contain one value.
- Check for NULL values - Lots were found, but none in dependent variable, which is good.

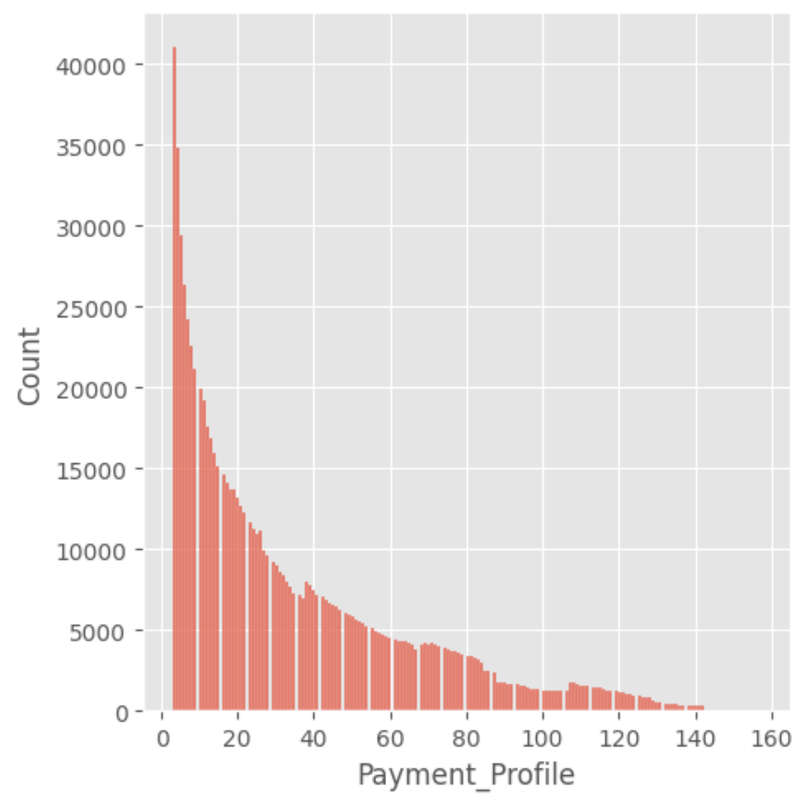
EXPLORATORY DATA ANALYSIS

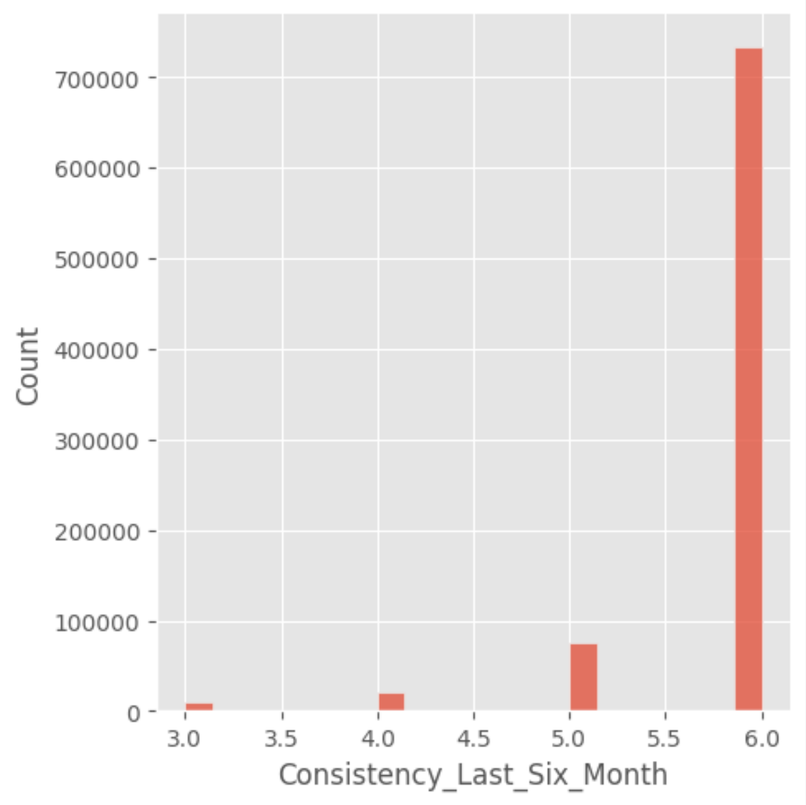
- Before removing or replacing nulls, should examine each variable in more detail.
- Plotted each variables 1D distribution using Seaborn Displot & CatPlot.



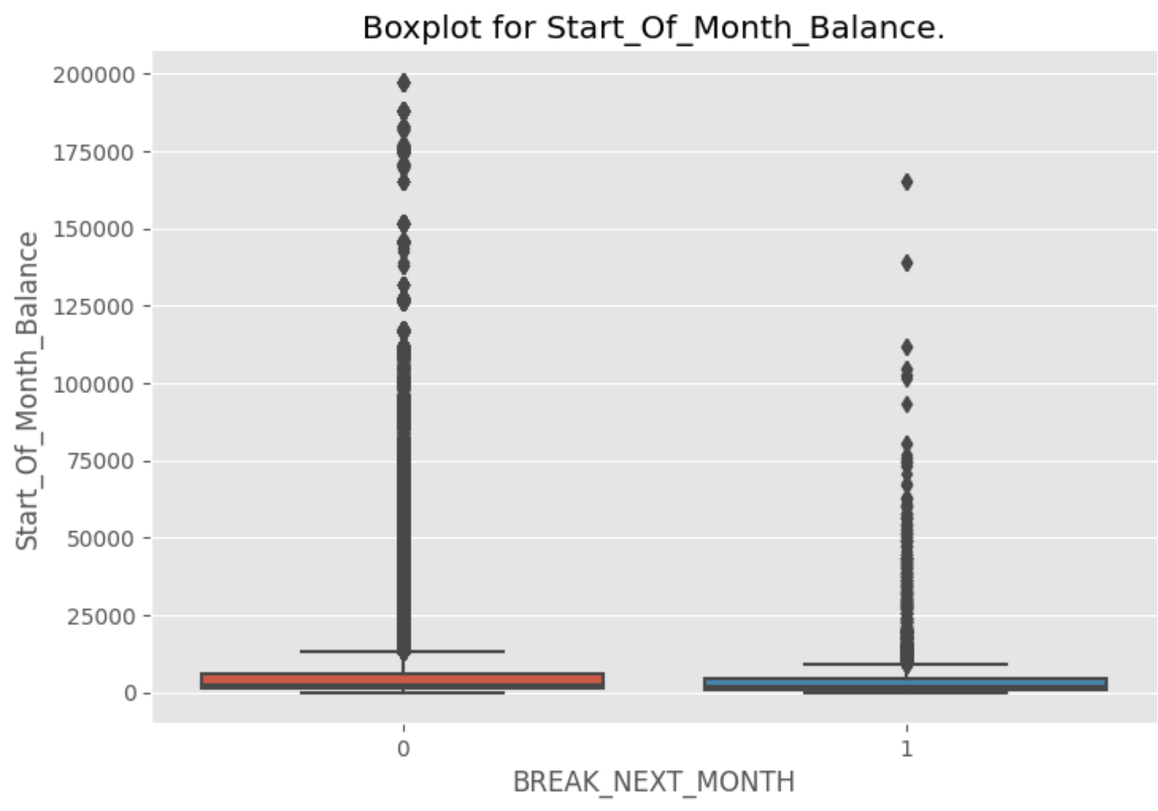


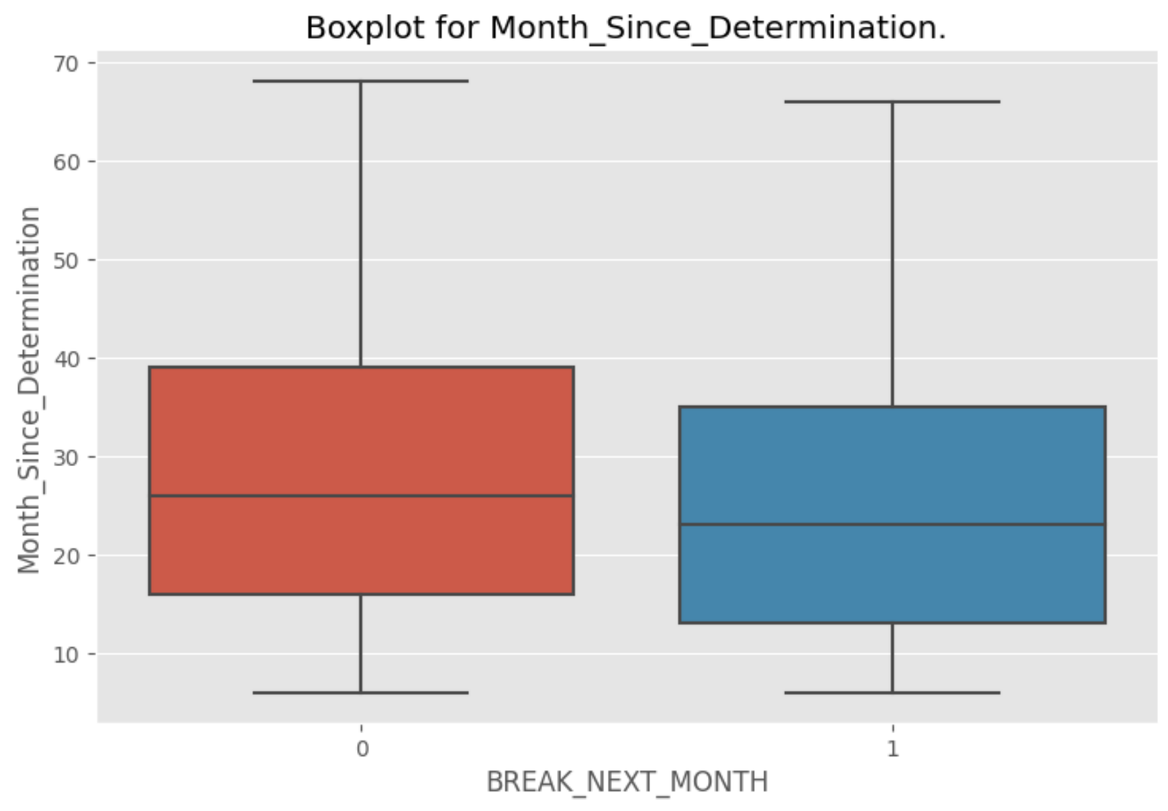
EXPLORATORY DATA ANALYSIS



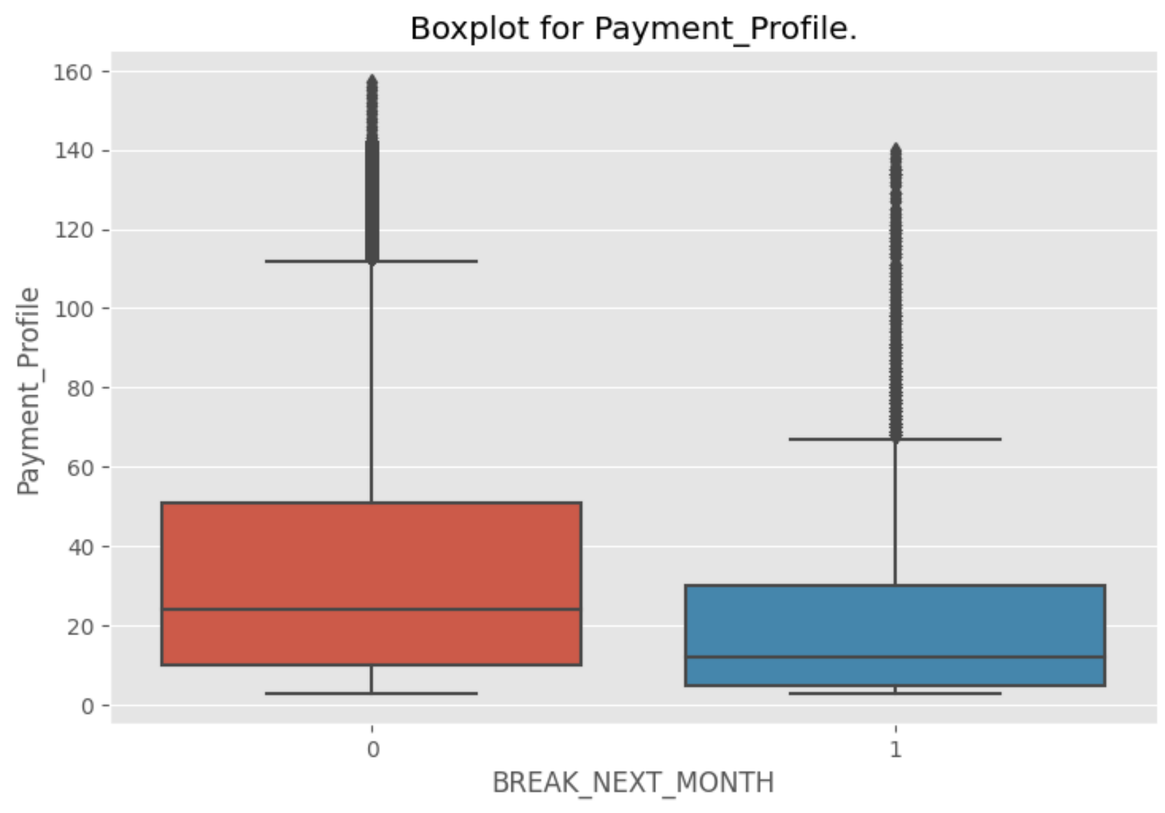


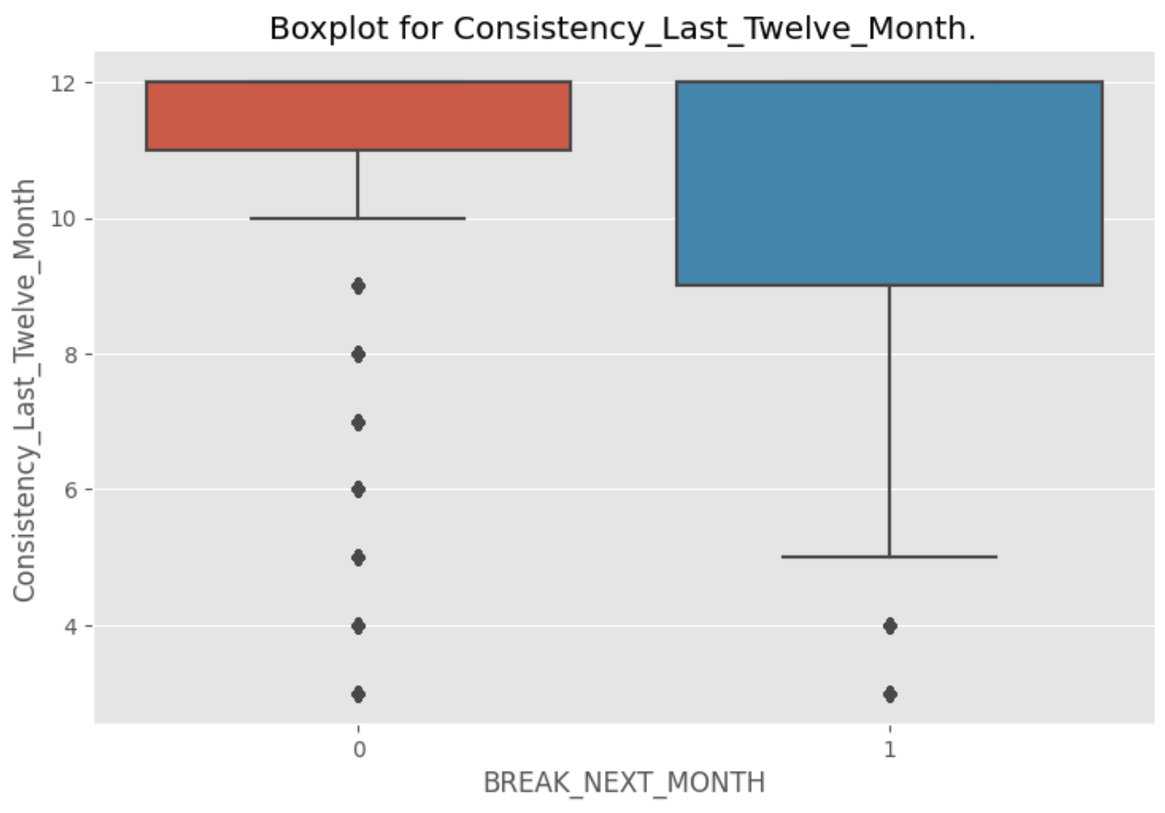
EXPLORATORY DATA ANALYSIS



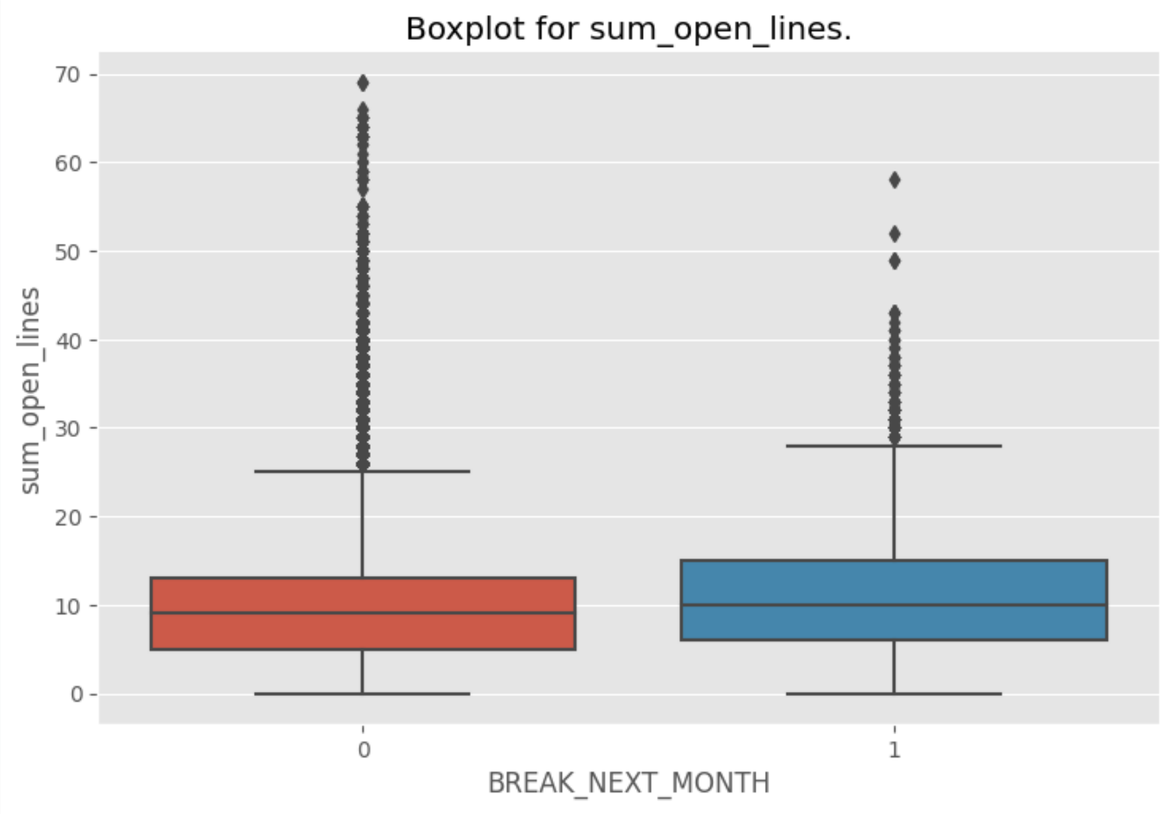


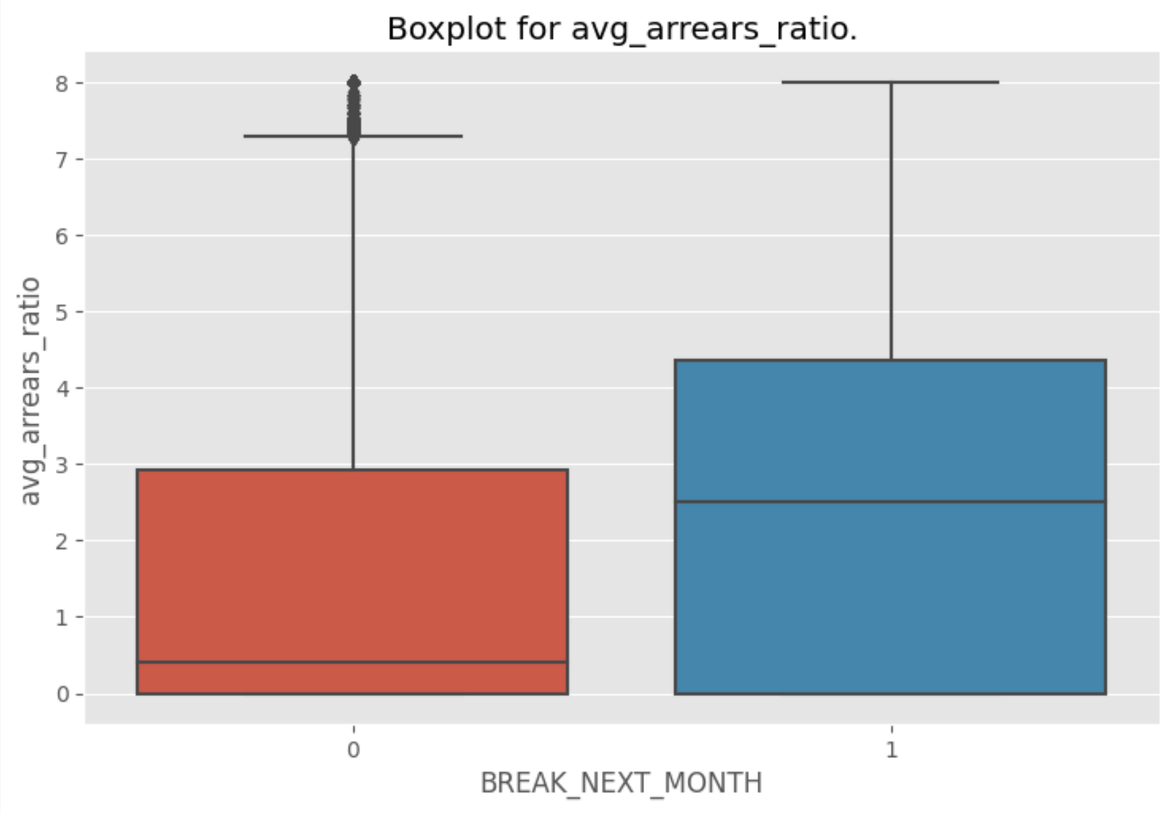
EXPLORATORY DATA ANALYSIS



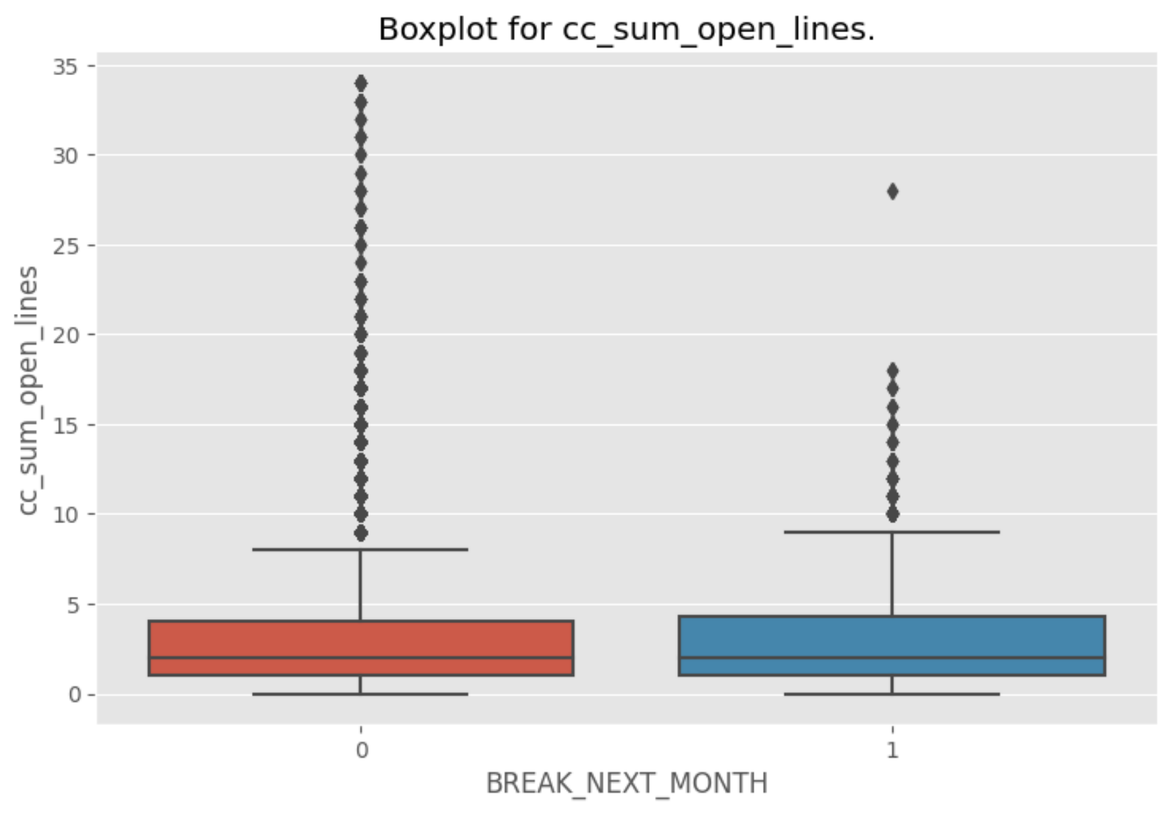


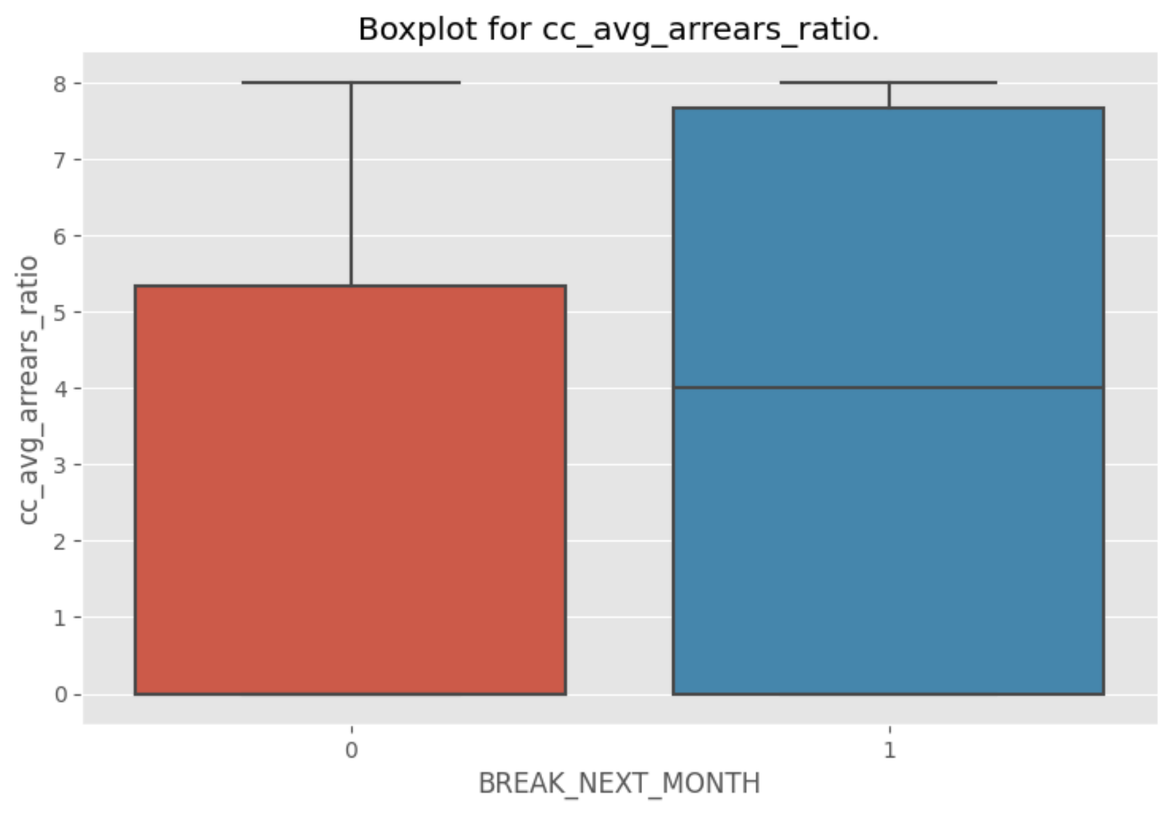
EXPLORATORY DATA ANALYSIS





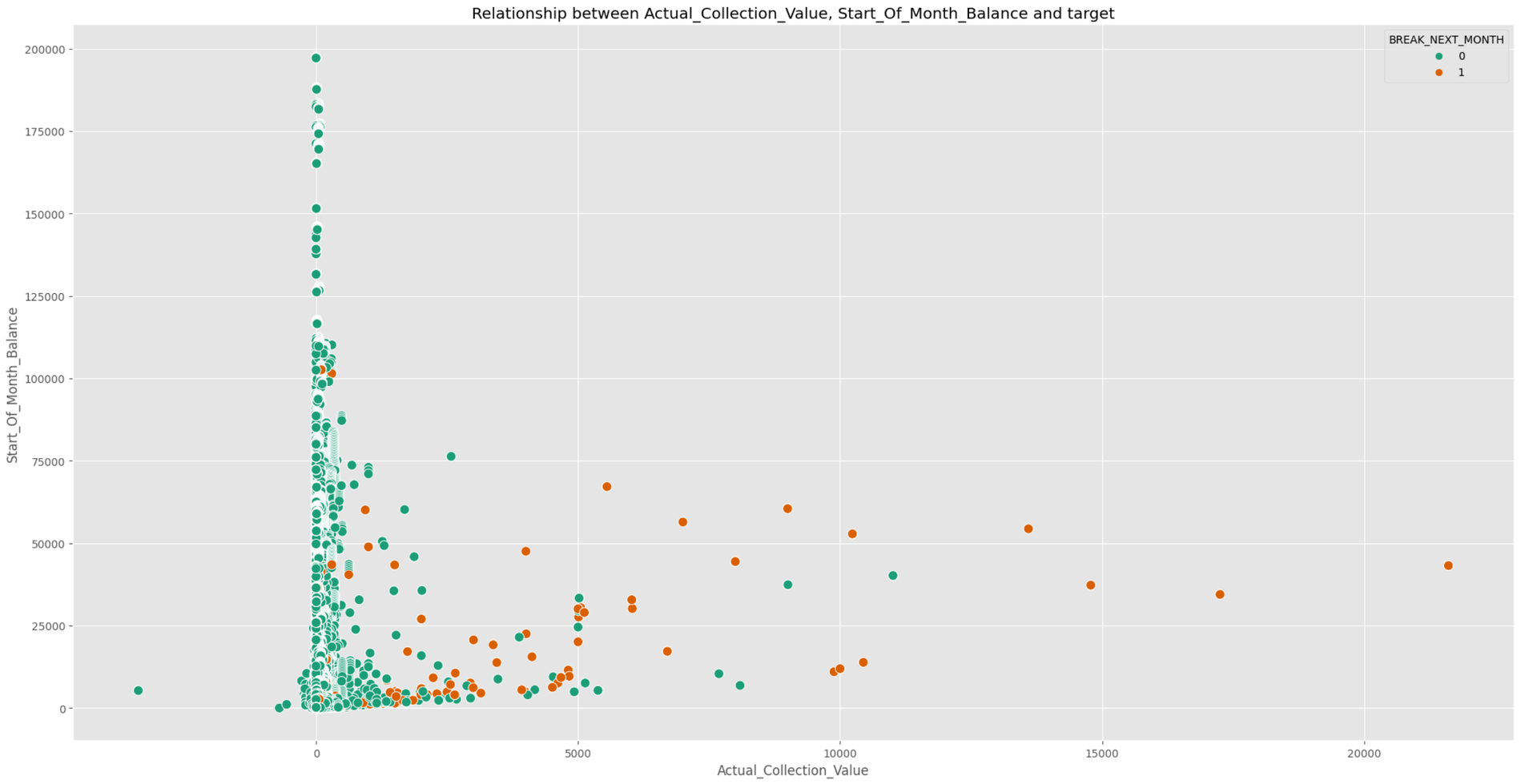
EXPLORATORY DATA ANALYSIS



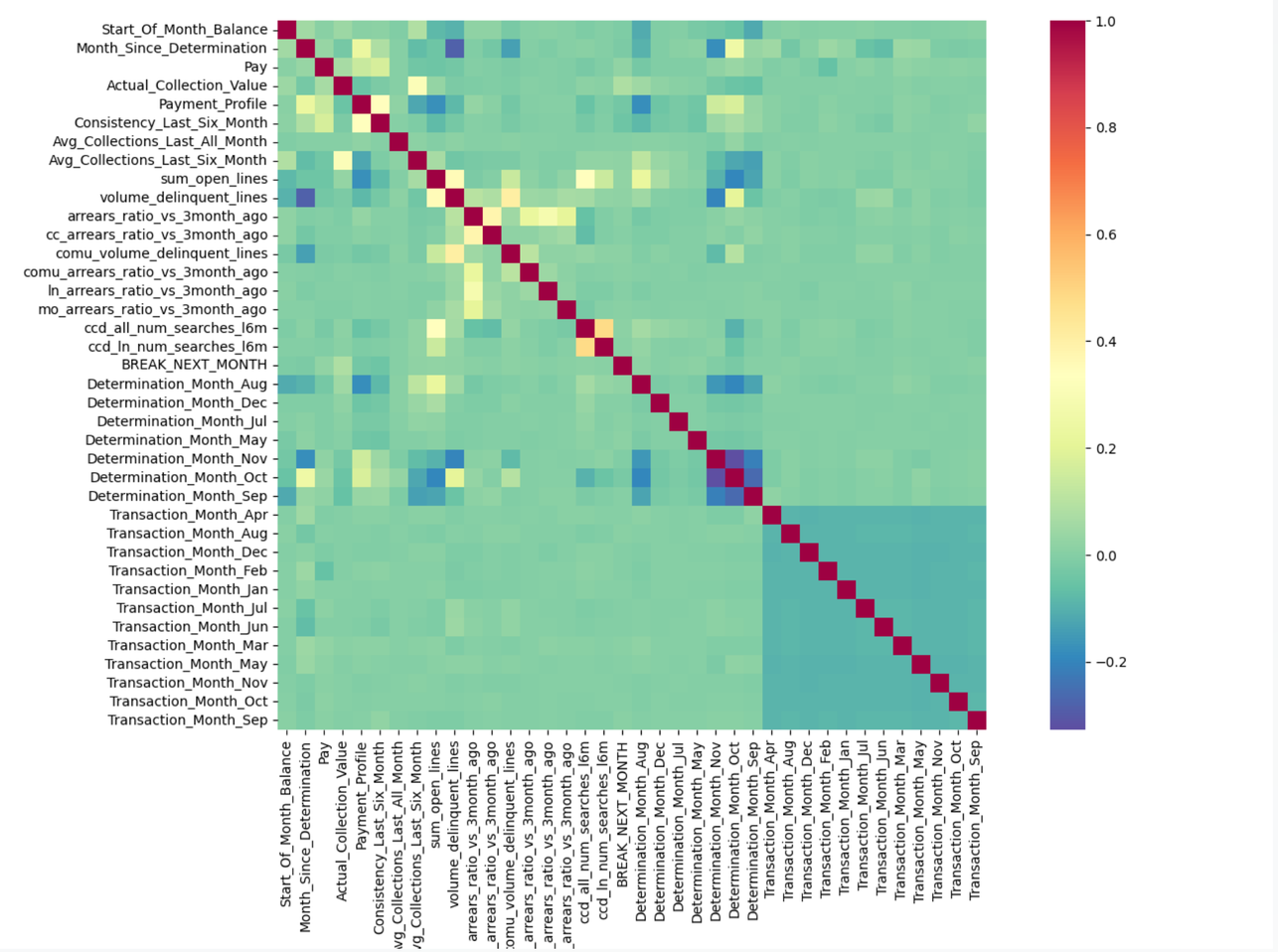


EXPLORATORY DATA ANALYSIS

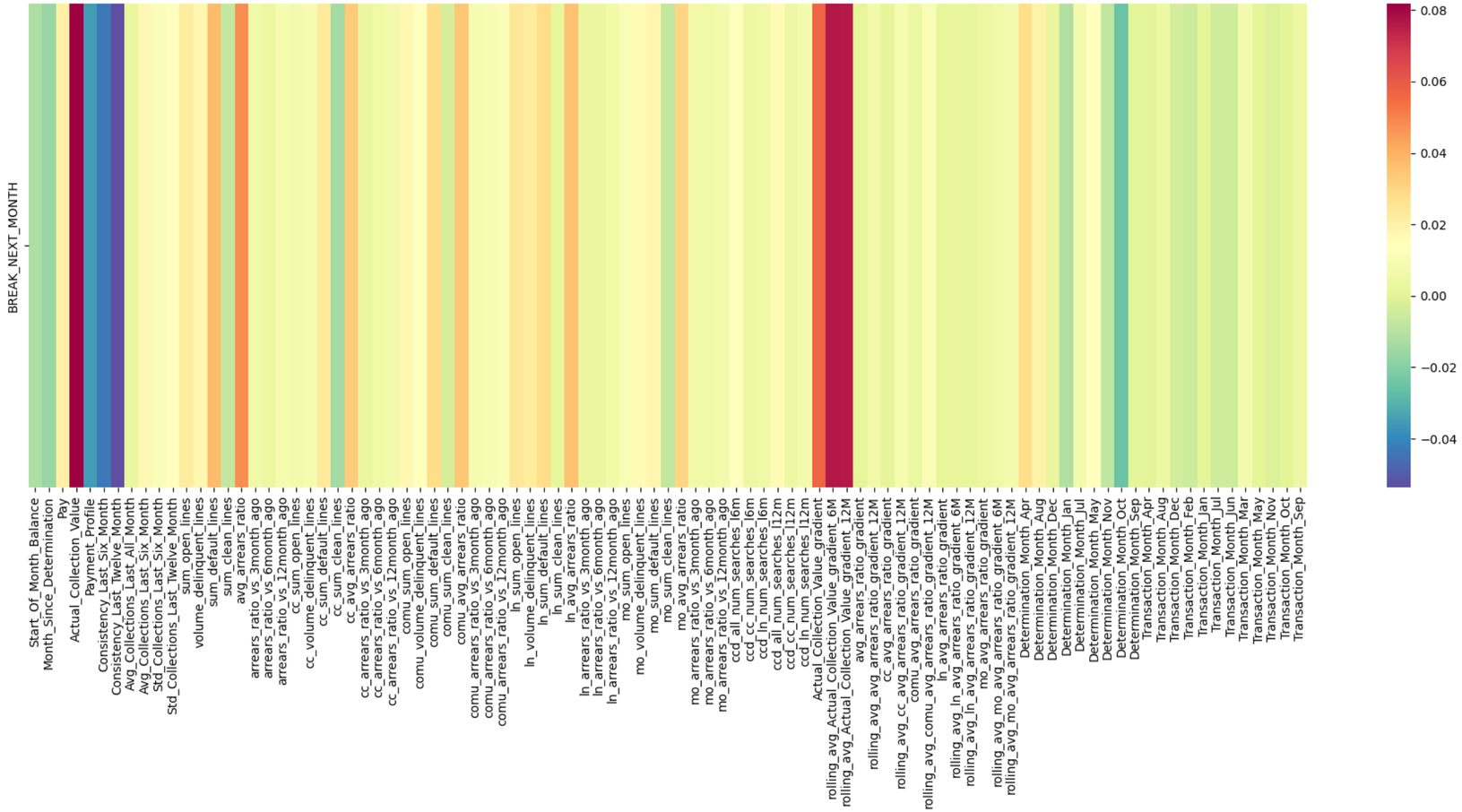
EXPLORATORY DATA ANALYSIS



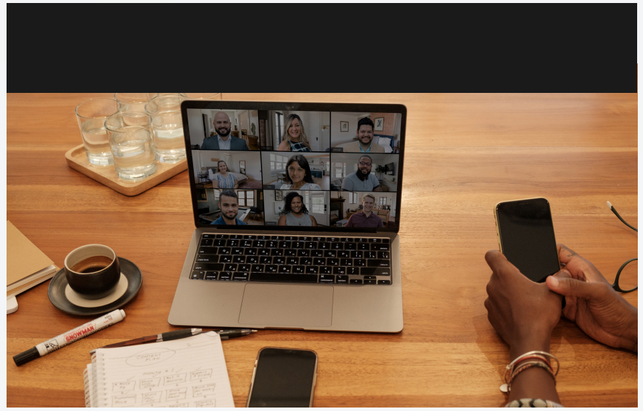
EXPLORATORY DATA ANALYSIS



EXPLORATORY DATA ANALYSIS



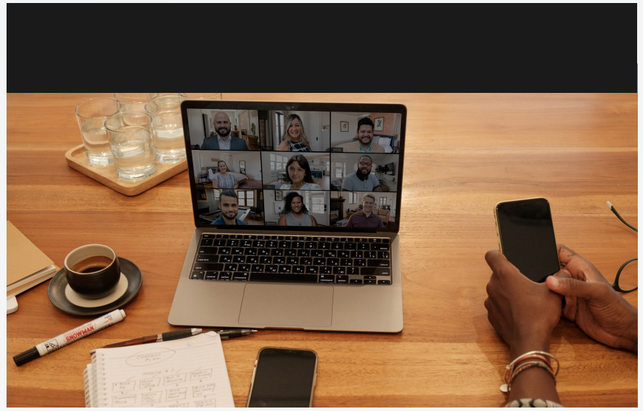
EXPLORATORY DATA ANALYSIS



- This EDA shows that there are numerous variables with a non-normal distribution, and significant numbers of outliers. As a consequence, we should consider removing outliers and scaling the data. this should help to improve ML model prediction accuracy.
- Next Steps...

- Drop INDIVIDUAL_ID column, as this will not be used in determination. There are thousands of unique values for this, which would cause overfitting, as well as providing too many features to train a model.
- Investigate number of Breakage vs non-Breakage.
- Target Variable Counts: 831,257 vs 6,720 BREAK_NEXT_MONTH occurence rate is : 0.81%.
- This illustrates a massive class imbalance, which mayb need to be addressed using Oversampling or Undersampling.
- Replace NULLs with Zeros. too many NULLS to drop. Replacing with Zero seems sensible as it is a common value, and makes most sense for Bureau variables.

EXPLORATORY DATA ANALYSIS



- Added as optional flags before training:

- Remove Outliers
- Balance Data
- Remove Cross-Correlated Columns

- By using an approach based on upper and lower 1% of data, outliers could be removed. However, due to the extreme distributions and the high number of variables, this caused a large proportion of rows to be discarded. As this may impact ML performance, it has been implemented optionally.
- Data will be optionally balanced using the SMOTE library and Oversampling.
- Features which have high cross-correlations with each other, can cause deterioration of model performance, due to over-reliance on one fundamental data pattern. These are removed by discarding columns which are 85% or more linearly correlated.

MACHINE LEARNING ARCHITECTURE

The majority of the time spent on this task was focused upon developing a robust training architecture, which can be re-used for other problems.

MACHINE LEARNING ARCHITECTURE

.git	12/11/2023 12:19	File folder
Analysis	10/11/2023 12:29	File folder
Data	12/11/2023 18:42	File folder
Documentation	12/11/2023 13:56	File folder
Environment	12/11/2023 12:16	File folder
.gitignore	07/11/2023 10:31	

- A modular approach was taken to ensure readability, adherence to proper Python design principles, and ease of re-usability.

__pycache__	12/11/2023 17:08	File folder	
balance_data	10/11/2023 12:19	Python Source File	1 KB
classic_ml	12/11/2023 17:07	Python Source File	6 KB
data_cleaning	12/11/2023 16:36	Python Source File	2 KB
date_utilities	10/11/2023 12:32	Python Source File	2 KB
eda	12/11/2023 17:09	Python Source File	9 KB
helper_functions	10/11/2023 12:39	Python Source File	4 KB
lazy_predict	10/11/2023 12:40	Python Source File	3 KB
ml_parameters	12/11/2023 16:58	Python Source File	3 KB
neural_networks	12/11/2023 17:31	Python Source File	3 KB
plotting	12/11/2023 16:15	Python Source File	2 KB

MACHINE LEARNING ARCHITECTURE

- After Data has been cleaned and processed during the EDA phase, it can be passed to an ML Model for training.
- Firstly, a third-party library called Lazy predict is used to run a large number of ML Models. This will provide a baseline of which model might be best to use.
- Then two models from the best performing will be trained without parameter hypertuning, for efficiency and speed.
- The best model from this will then be trained using a parameter grid, which can help to optimise performance.
- In addition, a non-classical ML approach of a Basic neural Network will be attempted.

MACHINE LEARNING RESULTS

- Lazy Predict ran for numerous Machine Learning Models, in concert with combinations of Balance Data, Remove Outliers, and Remove Cross-Correlated Features.

- The results were sadly quite underwhelming, as follows:

Best Model – Decision Tree
F1 Score : 0.061
Recall : 0.066 (6.6%)
Precision : 0.056 (5.6%)
Accuracy : 0.984 (98.4%)
Remove Outliers : False
Balance Data : False
Remove Cross Correlations : False
NFeatures : 85

MACHINE LEARNING RESULTS

- As a consequence, I opted to train a Decision Tree Classifier, and a SGD Model. The trained Decision Tree slightly exceeded the results from Lazy Predict. The SGD Classifier was sub-optimal.

Best Model - Decision Tree
F1 Score : 0.092
Recall : 0.098 (9.8%)
Precision : 0.087 (8.7%)
Accuracy : 0.984 (98.4%)
Remove Outliers : False
Balance Data : False
Remove Cross Correlations : False
NFeatures : 10

- The next step was to tune the Decision Tree. I used a small parameter grid, as time was running short. The performance of this model was in fact lower than the un-tuned version.

- The Neural Network model did not surpass the Decision Tree.

FUTURE IMPROVEMENTS

- Of course, ML model performance turned out to be quite poor, Improvements to the process could be made by looking into the three following areas:

More comprehensive Exploratory Data Analysis. I feel like I may have missed some potentially transformative variable. Having domain knowledge of the features in-depth would be greatly beneficial for this.

1ST

More time to run combinations of ML Models and parameters. This would be beneficial, but would not likely cause dramatic improvements unless the underlying data issue is addressed.

2ND

Access to the wider dataset, and creation of new variables and other feature engineering techniques may help. Determination of other useful variables to add to the dataset. E.g. Address History

3RD

THANKS FOR LISTENING.