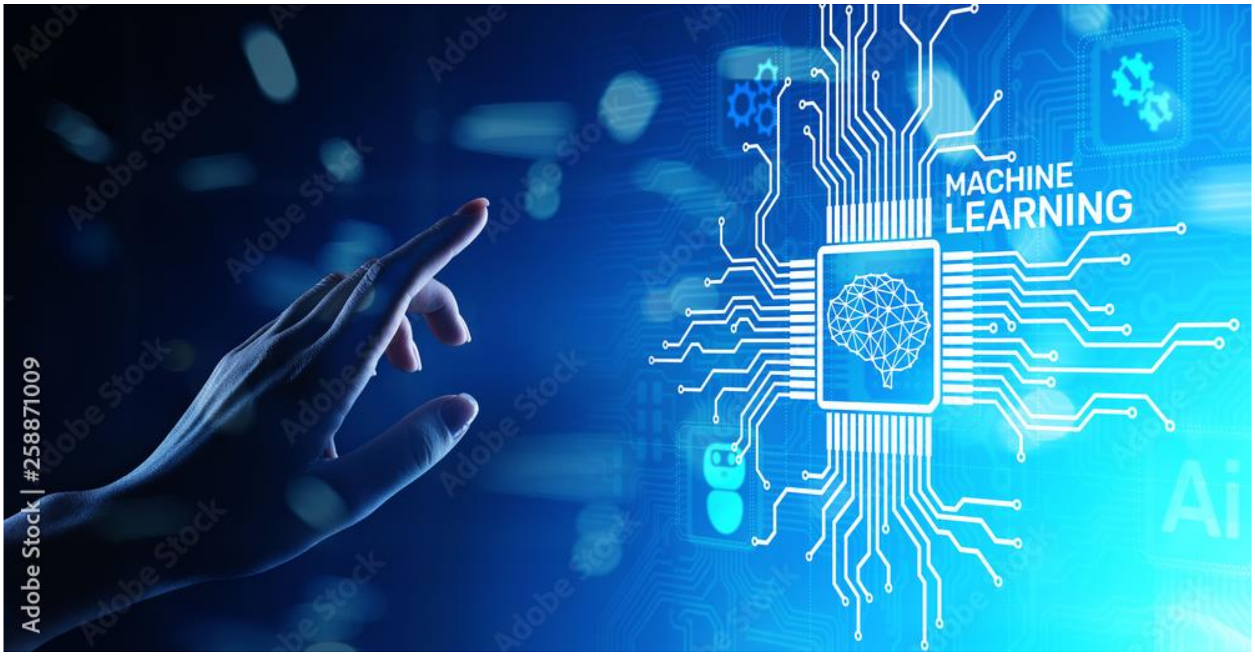
Machine Learning Model Workflow Report



Report Prepared by:

Aya Hamdy

Agenda

1. **Introduction**
    - Overview of the Machine Learning model workflow
    - Objective of the report
2. **Data Preparation**
    - Data pre-processing
    - Handling outliers
    - Data augmentation techniques
    - Standardizing Data
3. **Model Selection with PyCaret**
    - Overview of PyCaret
    - Rationale for using PyCaret
    - Algorithm comparison
4. **Modelling**
    - Gradient Boosting Classifier (GBC)
    - Logistic Regression
    - Support Vector Classifier (SVC)
5. **TensorBoard Workflow**
    - Visualization of model metrics
    - Workflow tracking
6. **Conclusion**
    - Summary of findings

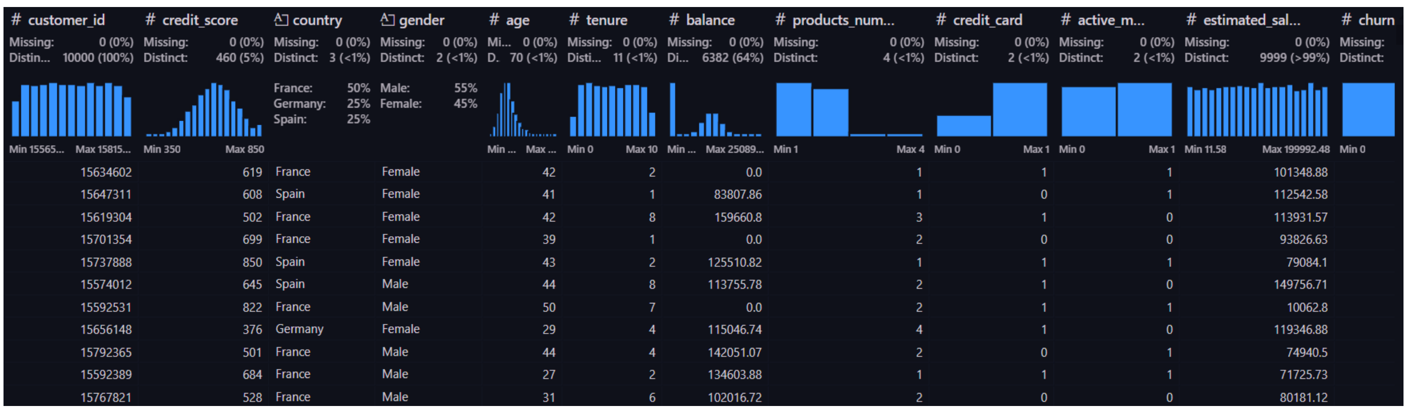
1. Introduction

This report outlines the workflow followed to build a Machine Learning model, covering data preparation, model selection, and evaluation. The primary objective was to develop a robust model that accurately classifies data using techniques like Logistic Regression, Support Vector Classifier (SVC), and Gradient Boosting Classifier. PyCaret was employed for algorithm comparison, and TensorBoard was used to track the workflow visually.

2. Data Preparation

2.1. Data Pre-Processing

The dataset consists of 10,000 entries and 11 columns, including features like credit_score, age, balance, and the target variable churn. The data was cleaned to remove missing values and inconsistencies.



# customer_id	# credit_score	country	gender	# age	# tenure	# balance	# products_num...	# credit_card	# active_m...	# estimated_sal...	# churn
Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Mi... 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing:
Distin... 10000 (100%)	Distinct: 460 (5%)	Distinct: 3 (<1%)	Distinct: 2 (<1%)	D. 70 (<1%)	Disti... 11 (<1%)	Di... 6382 (64%)	Distinct: 4 (<1%)	Distinct: 2 (<1%)	Distinct: 2 (<1%)	Distinct: 9999 (>99%)	Distinct:
Min 15565... Max 15815...	Min 350 Max 850	France: 50% Germany: 25% Spain: 25%	Male: 55% Female: 45%	Min ... Max ...	Min 0 Max 10	Min ... Max 25089...	Min 1 Max 4	Min 0 Max 1	Min 0 Max 1	Min 11.58 Max 199992.48	Min 0
15634602	619	France	Female	42	2	0.0	1	1	1	101348.88	
15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	
15619304	502	France	Female	42	8	159660.8	3	1	0	113931.57	
15701354	699	France	Female	39	1	0.0	2	0	0	93826.63	
15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	
15574012	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	
15592531	822	France	Male	50	7	0.0	2	1	1	10062.8	
15656148	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	
15792365	501	France	Male	44	4	142051.07	2	0	1	74940.5	
15592389	684	France	Male	27	2	134603.88	1	1	1	71725.73	
15767821	528	France	Male	31	6	102016.72	2	0	0	80181.12	

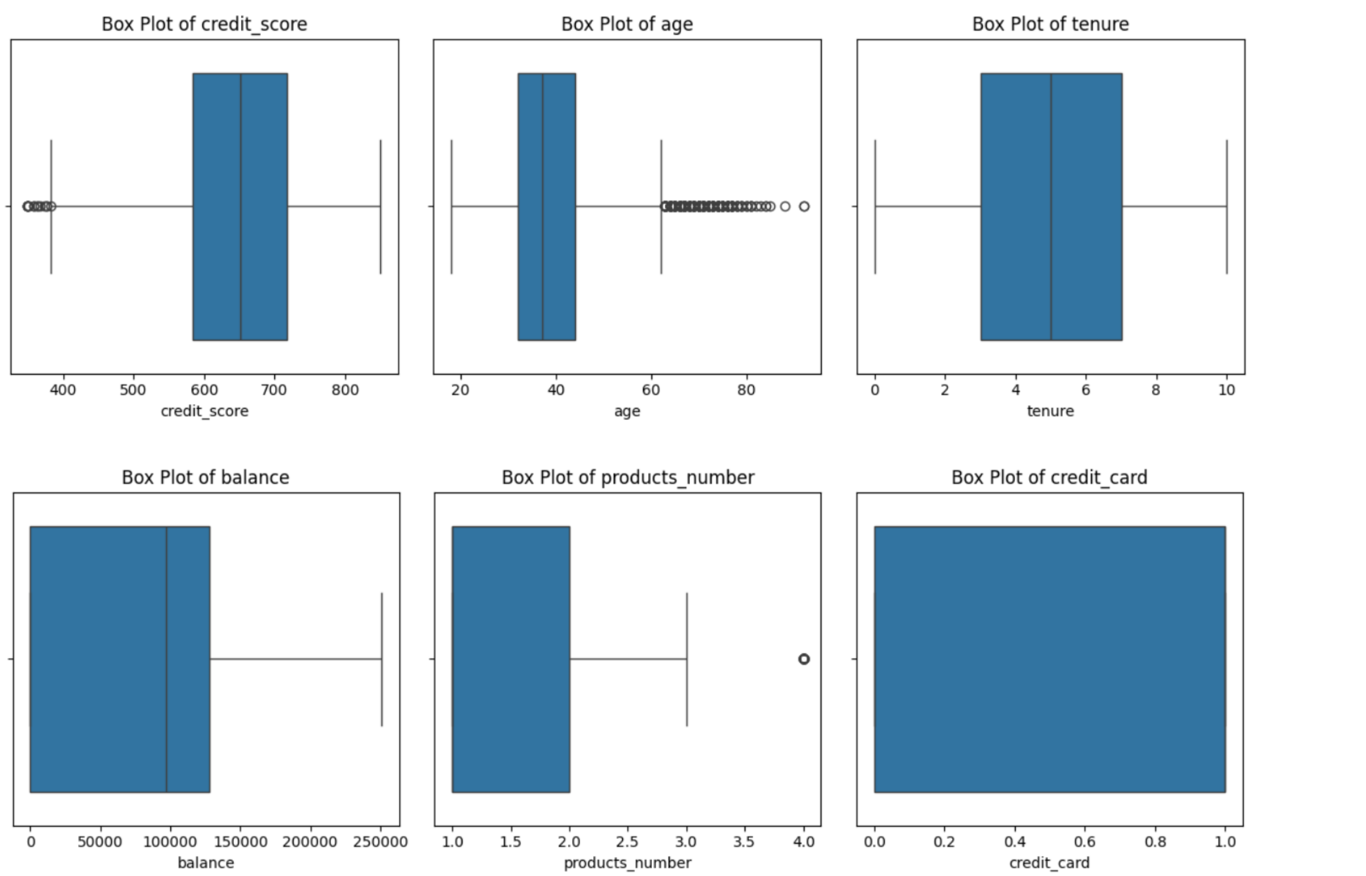
Data Information:

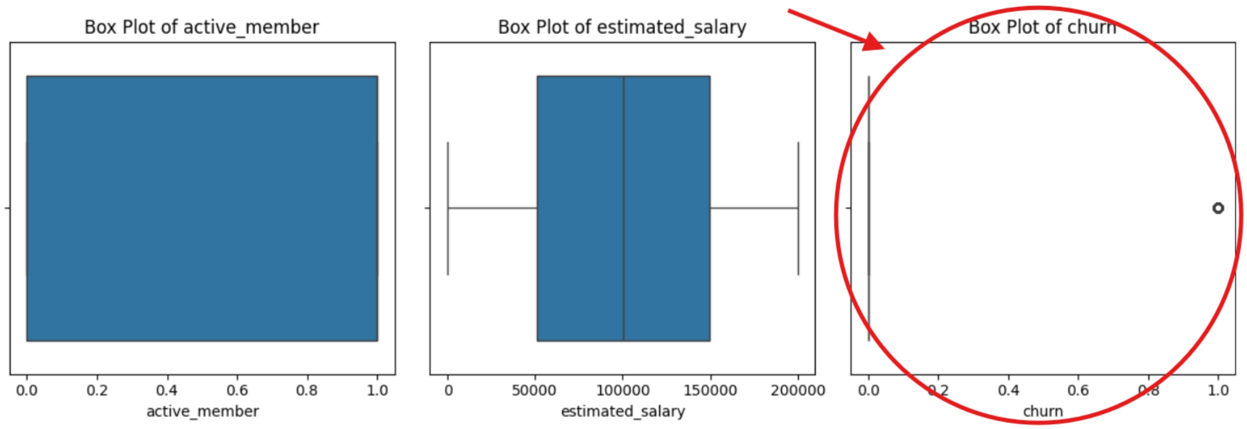
```
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   credit_score      10000 non-null  int64
 1   country           10000 non-null  object
 2   gender            10000 non-null  object
 3   age               10000 non-null  int64
 4   tenure            10000 non-null  int64
 5   balance           10000 non-null  float64
 6   products_number   10000 non-null  int64
 7   credit_card       10000 non-null  int64
 8   active_member     10000 non-null  int64
 9   estimated_salary  10000 non-null  float64
 10  churn             10000 non-null  int64
dtypes: float64(2), int64(7), object(2)
```

2.2. Handling Outliers

Outliers were identified in the dataset as follows:

```
Data Shape: (10000, 11)
Total Outliers in credit_score: 15 -- 0.15%
Total Outliers in age: 359 -- 3.59%
Total Outliers in tenure: 0 -- 0.0%
Total Outliers in balance: 0 -- 0.0%
Total Outliers in products_number: 60 -- 0.6%
Total Outliers in credit_card: 0 -- 0.0%
Total Outliers in active_member: 0 -- 0.0%
Total Outliers in estimated_salary: 0 -- 0.0%
Total Outliers in churn: 2037 -- 20.37%
```





Some Observations from above plots:

- We can see that data is highly imbalanced. Almost **80%** of our data is from class 0 (not exited) and **20%** data is from class 1 (exited).

```
# Check the class distribution before augmentation
print("class distribution before augmentation: ")
print(data['churn'].value_counts())
```

✓ 0.0s

```
class distribution before augmentation:
churn
0    7963
1    2037
```

- In a real life also we only care about the people who are quitting or leaving (Exited) the bank, and we only want to analyze the patterns of those people.

2.3. Data Augmentation

I used data augmentation techniques to balance the dataset between the "Exited" and "Not Exited" classes. Since the original dataset had an imbalance between the number of customers who left the bank (Exited) and those who stayed (Not Exited), augmentation was applied to ensure the model receives equal representation from both classes. This step helps improve the model's ability to accurately predict customer churn without being biased toward the majority class.

```
# Check the class distribution after augmentation
print("class distribution after augmentation: ")
print(df_resampled['churn'].value_counts())
```

```
class distribution after augmentation:
churn
1    7963
0    7963
```


```
Data Shape: (15926, 11)
Total Outliers in credit_score: 34 -- 0.21%
Total Outliers in age: 253 -- 1.59%
Total Outliers in tenure: 0 -- 0.0%
Total Outliers in balance: 0 -- 0.0%
Total Outliers in products_number: 64 -- 0.41%
Total Outliers in credit_card: 0 -- 0.0%
Total Outliers in active_member: 0 -- 0.0%
Total Outliers in estimated_salary: 0 -- 0.0%
Total Outliers in churn: 0 -- 0.0%
Cleaned Data Shape: (15575, 11)
```

The cleaned data shape was (15575, 11), indicating that outliers were addressed effectively to improve model performance.

2.4. Standardizing Data

Before training the Gradient Boosting model, the data was standardized to ensure that all features contributed equally to the model.

Why Standardization?
Standardizing the data ensures that all features are on the same scale, preventing models from assigning disproportionate importance to features with larger ranges.

3. PyCaret for Algorithm Selection

3.1. Overview of PyCaret

PyCaret is an open-source, low-code machine learning library that streamline the process of finding the most suitable algorithm, PyCaret was used. PyCaret simplifies the comparison of machine learning models and automatically tunes hyperparameters.

3.2. Rationale for Using PyCaret

Why PyCaret?

The key reason for using PyCaret in this workflow was to speed up the model comparison process. Rather than manually coding each model, PyCaret allowed us to automatically test a wide range of classifiers. This significantly reduces the time required to identify the best performing model for our churn prediction task.

```
save_model(tuned_model, 'best_churn_model')
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.8678	0.8684	0.4789	0.7797	0.5928	0.5193	0.5416	
rf	Random Forest Classifier	0.8609	0.8550	0.4455	0.7663	0.5629	0.4871	0.5127	
lightgbm	Light Gradient Boosting Machine	0.8608	0.8586	0.4878	0.7307	0.5842	0.5047	0.5200	
et	Extra Trees Classifier	0.8578	0.8481	0.4492	0.7428	0.5592	0.4806	0.5026	
ada	Ada Boost Classifier	0.8575	0.8468	0.4818	0.7177	0.5761	0.4945	0.5089	
xgboost	Extreme Gradient Boosting	0.8543	0.8509	0.4877	0.6972	0.5735	0.4891	0.5006	
lr	Logistic Regression	0.8314	0.7965	0.3284	0.6636	0.4388	0.3529	0.3834	
ridge	Ridge Classifier	0.8287	0.7954	0.2276	0.7399	0.3470	0.2797	0.3448	
lda	Linear Discriminant Analysis	0.8287	0.7954	0.3410	0.6409	0.4447	0.3545	0.3794	
svm	SVM - Linear Kernel	0.8205	0.7699	0.3143	0.6247	0.4056	0.3157	0.3468	
knn	K Neighbors Classifier	0.8194	0.7545	0.3314	0.5929	0.4246	0.3276	0.3474	
nb	Naive Bayes	0.8069	0.7870	0.4114	0.5268	0.4613	0.3460	0.3503	
dummy	Dummy Classifier	0.7987	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	
dt	Decision Tree Classifier	0.7963	0.6906	0.5137	0.4940	0.5032	0.3753	0.3756	
qda	Quadratic Discriminant Analysis	0.5223	0.5290	0.5074	0.1982	0.2536	0.0199	0.0270	

3.3. Algorithm Comparison

The table from PyCaret shows the performance of various models based on several metrics, including **Accuracy**, **AUC**, **Recall**, **Precision**, **F1-Score**, **Kappa**, and **MCC**. Here are the highlights:

- **Gradient Boosting Classifier (GBC)** performed the best with:
  - **Accuracy**: 0.8678
  - **AUC**: 0.8834
  - **Recall**: 0.4789
  - **Precision**: 0.7797
  - **F1-Score**: 0.5928
  - **Kappa**: 0.5198
  - **MCC**: 0.5416

- **Random Forest Classifier (RF)** and **LightGBM** also performed well but with slightly lower metrics compared to GBC.
- Other models like **Logistic Regression (LR)** and **Support Vector Machine (SVM)** showed moderate performance but were outperformed by tree-based models in terms of accuracy and AUC.
- **Decision Tree Classifier (DT)** had the lowest performance among the models tested with an accuracy of **0.7963**.

These results indicate that **Gradient Boosting Classifier (GBC)** was the best choice for this churn prediction task. The decision to proceed with GBC was driven by its higher accuracy and AUC scores, which are critical for predicting customer churn.

4. Modelling

4.1. Gradient Boosting Classifier

Gradient Boosting Classifier was used to build a stronger model by combining several weak learners (decision trees). This ensemble method reduces variance and improves model accuracy.

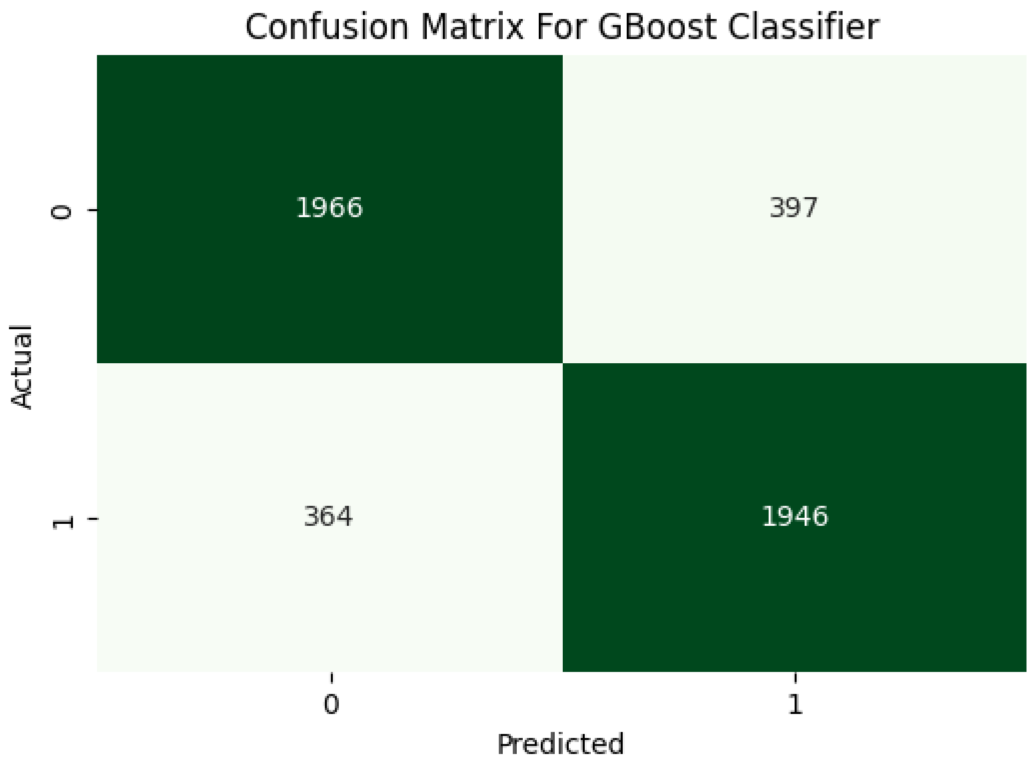
- **Optimized Accuracy:** 84%
- **AUC-ROC:** 0.9197
- **Best Hyperparameters:**
    - Learning Rate: 0.15
    - Max Depth: 4
    - Min Samples Leaf: 2
    - Min Samples Split: 5
    - N Estimators: 250

Classification Report:

```
Optimized Accuracy: 0.8371495827091804
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.83      0.84      2363
           1       0.83      0.84      0.84      2310

    accuracy                           0.84      4673
   macro avg       0.84      0.84      0.84      4673
weighted avg       0.84      0.84      0.84      4673

AUC-ROC: 0.9196970612967228
```

Confusion Matrix for GBC

- **True Positives (TP):** 1946
- **False Positives (FP):** 397
- **True Negatives (TN):** 1966
- **False Negatives (FN):** 364



This confusion matrix illustrates the model's performance in classifying the churn variable.

4.2. Logistic Regression

Logistic Regression was chosen as the baseline model due to its simplicity and interpretability. It provides a quick way to get insights into the dataset, especially for binary classification tasks.

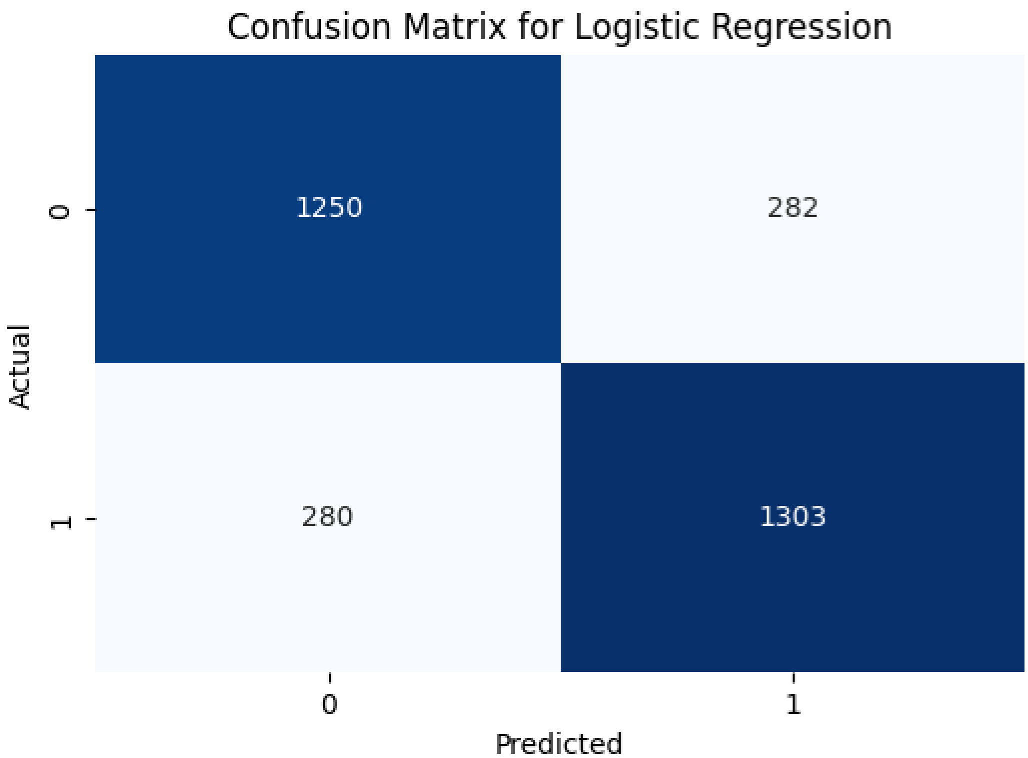
- **Accuracy:** 82%
- **AUC-ROC:** 0.9026

Classification Report:

```
Accuracy: 0.8195826645264848
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.82      0.82      1532
           1       0.82      0.82      0.82      1583

    accuracy                           0.82      3115
   macro avg       0.82      0.82      0.82      3115
weighted avg       0.82      0.82      0.82      3115

AUC-ROC: 0.9026483244789201
```

Confusion Matrix for Logistic Regression

- **True Positives (TP):** 1303
- **False Positives (FP):** 282
- **True Negatives (TN):** 1250
- **False Negatives (FN):** 280



This confusion matrix illustrates the model's performance in classifying the churn variable.

4.3. Support Vector Classifier (SVC)

The SVC was chosen for its ability to handle high-dimensional data and separate classes that are not linearly separable.

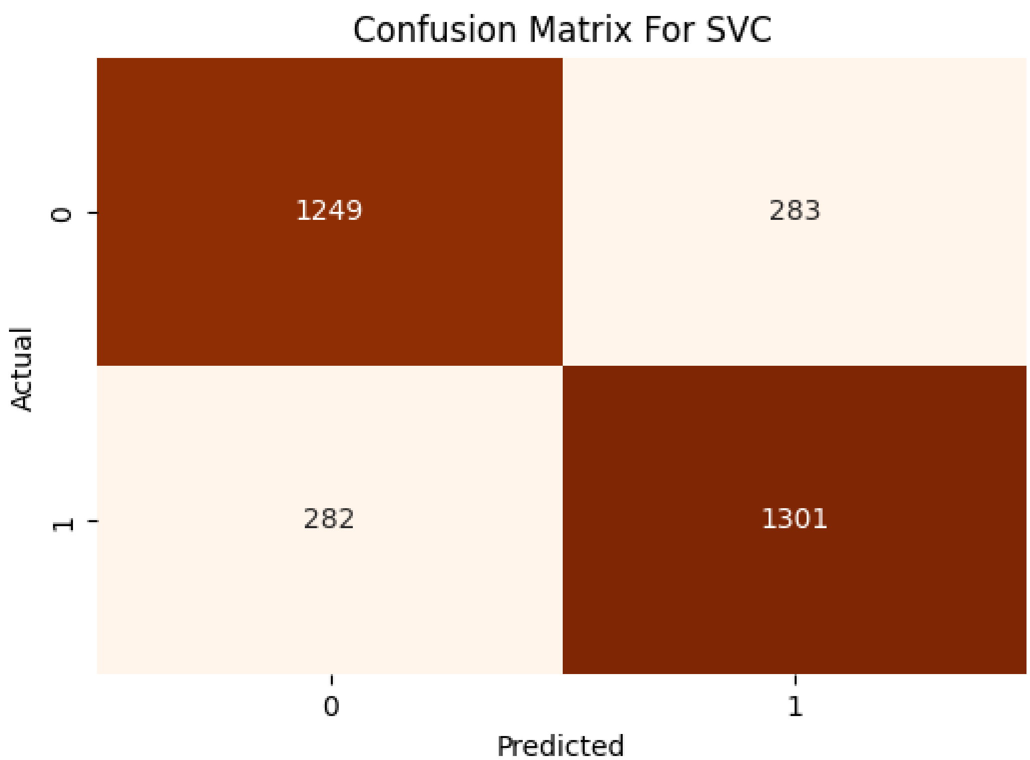
- **Accuracy:** 81.86%
- **AUC-ROC:** 0.9027

Classification Report:

```
Accuracy: 0.8186195826645265
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.82      0.82      1532
           1       0.82      0.82      0.82      1583

    accuracy                           0.82      3115
   macro avg       0.82      0.82      0.82      3115
weighted avg       0.82      0.82      0.82      3115

AUC-ROC: 0.9027039910009912
```

Confusion Matrix for SVC

- **True Positives (TP):** 1301
- **False Positives (FP):** 283
- **True Negatives (TN):** 1249
- **False Negatives (FN):** 282

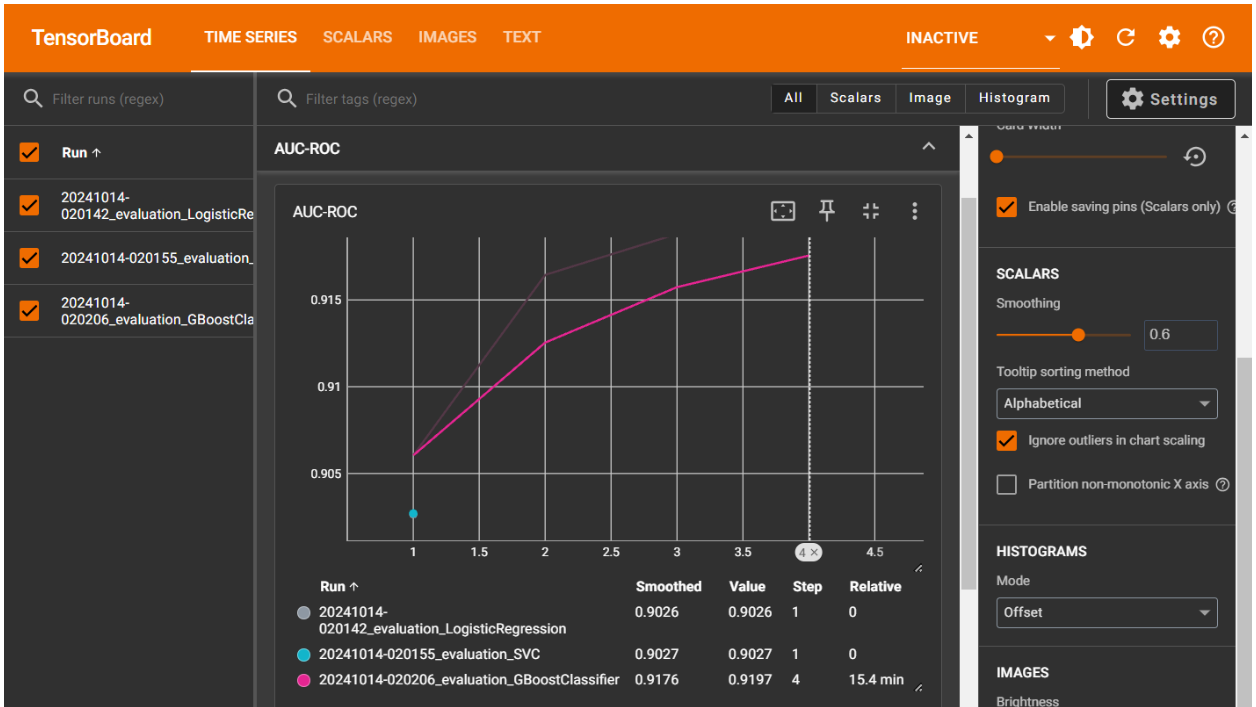


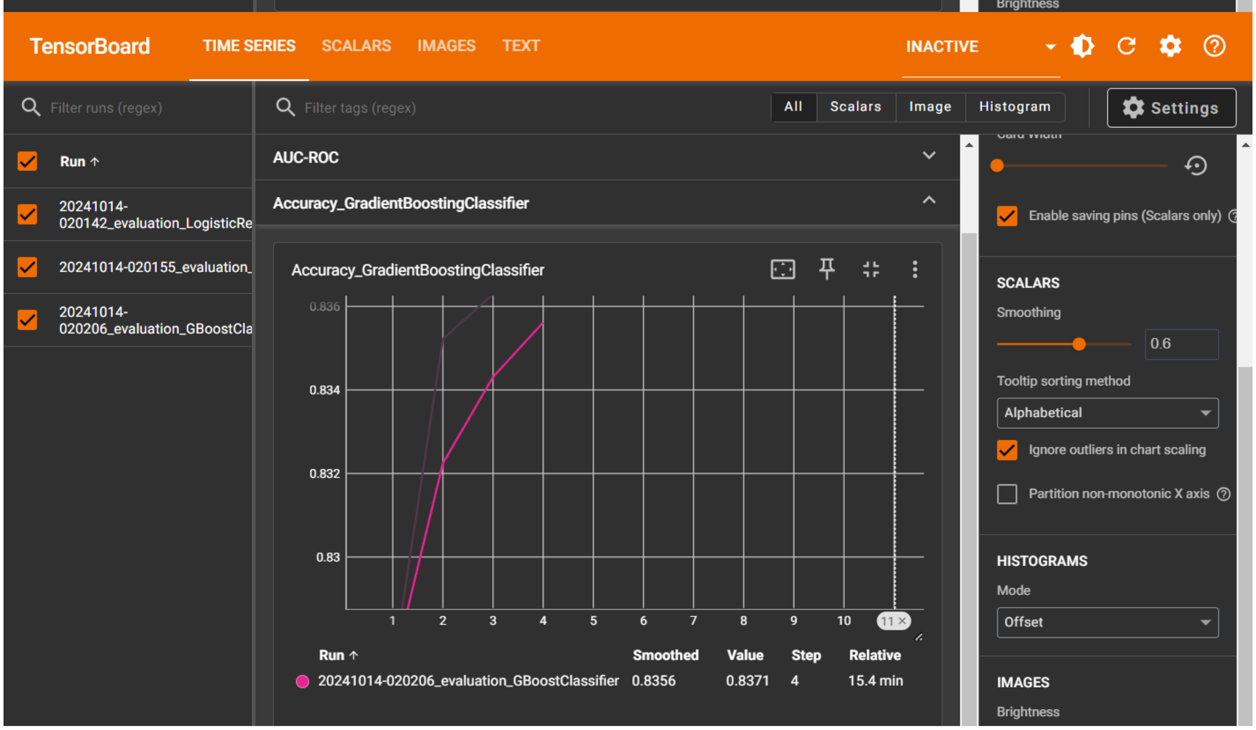
This confusion matrix illustrates the model's performance in classifying the churn variable.

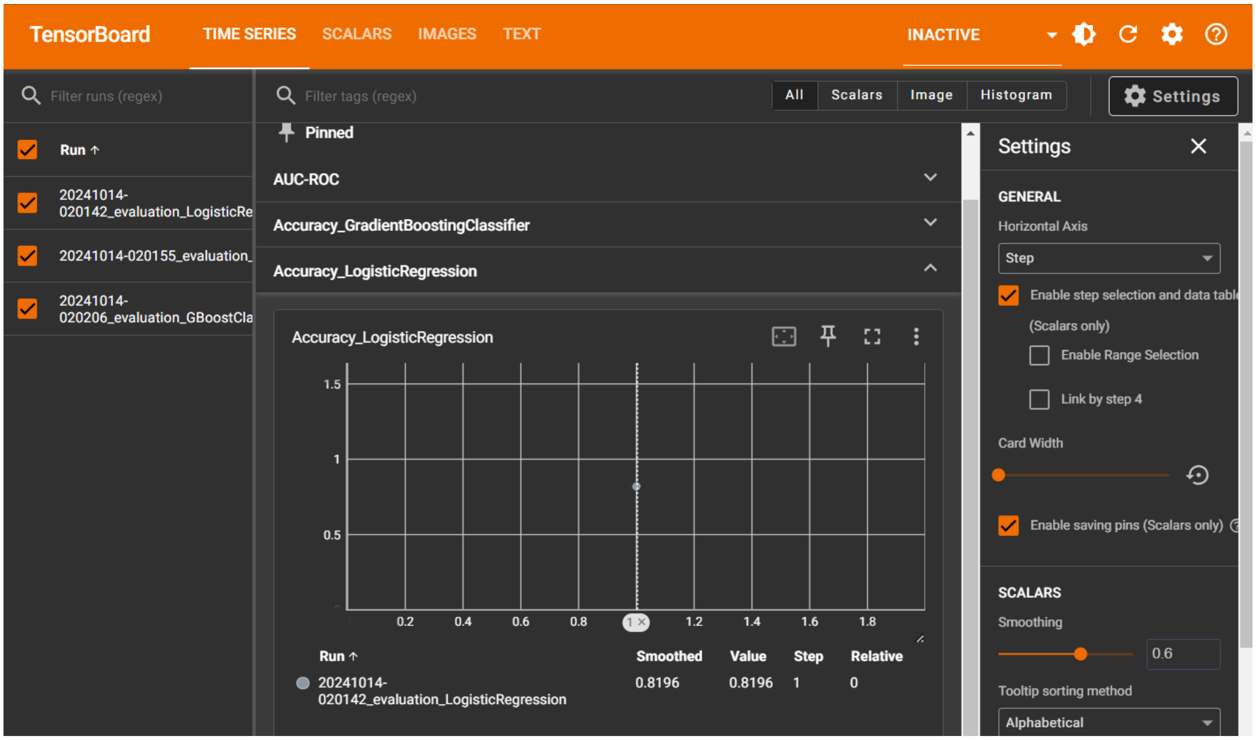
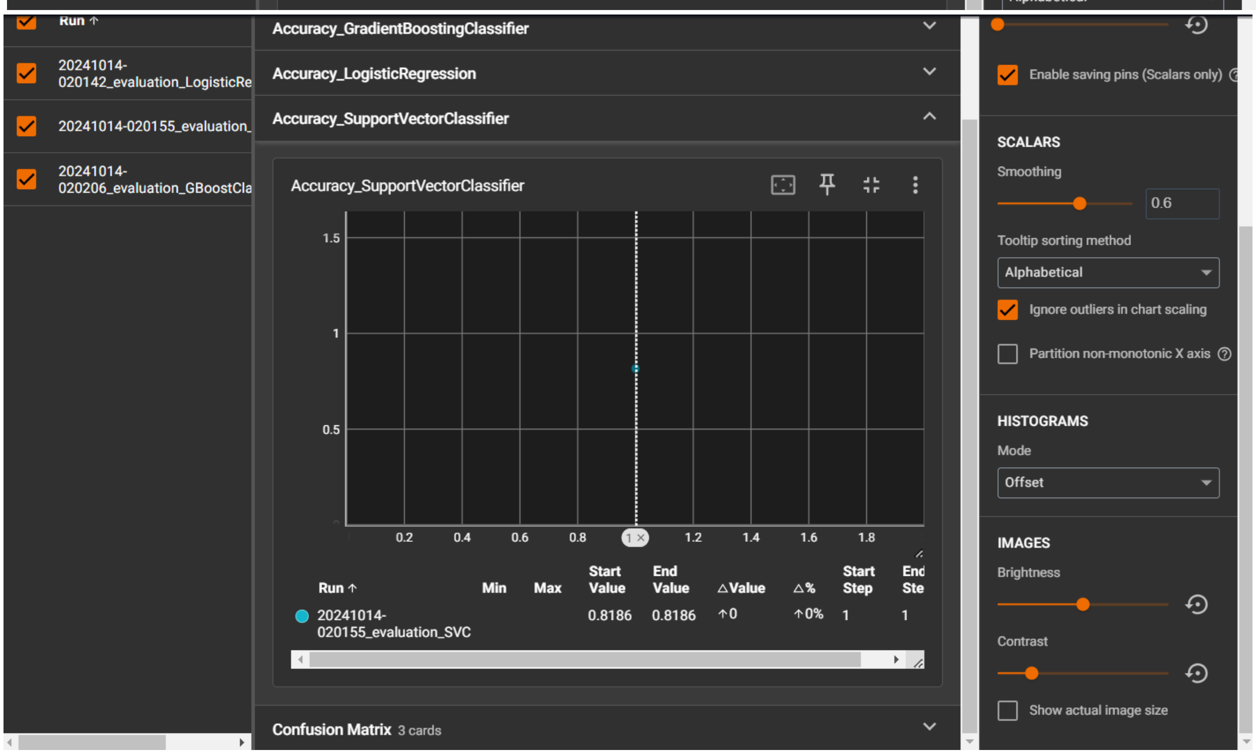
5. TensorBoard for Workflow Visualization

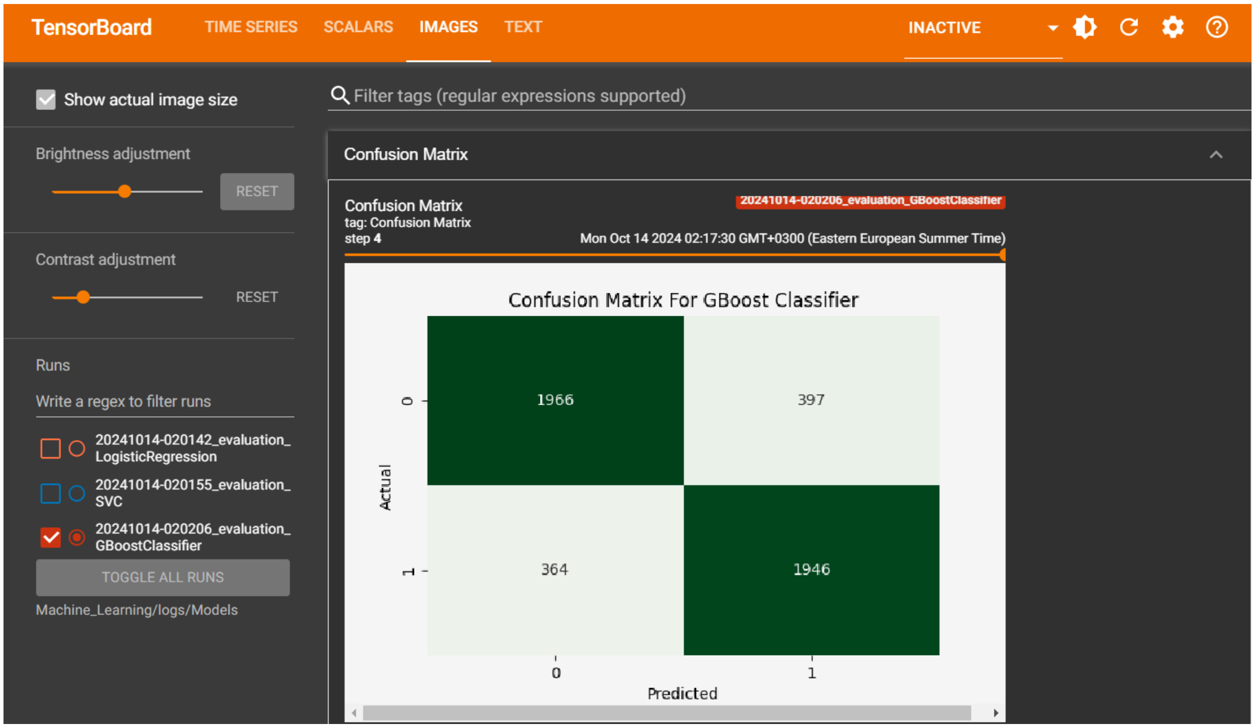
TensorBoard was used to visualize the model training process, including metrics such as accuracy, AUC-ROC scores, and confusion matrices for each model. It helped in monitoring the workflow and provided real-time insights into model performance.

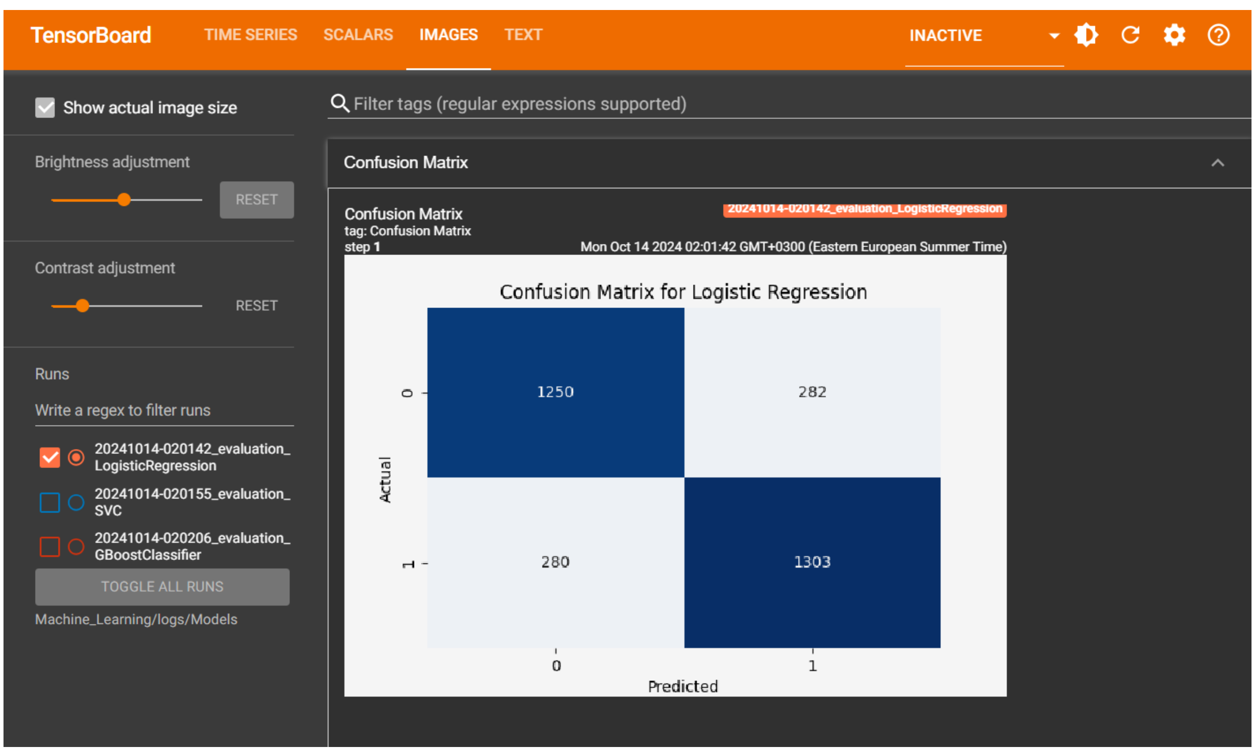
TensorBoard Logs Created At: log_dir

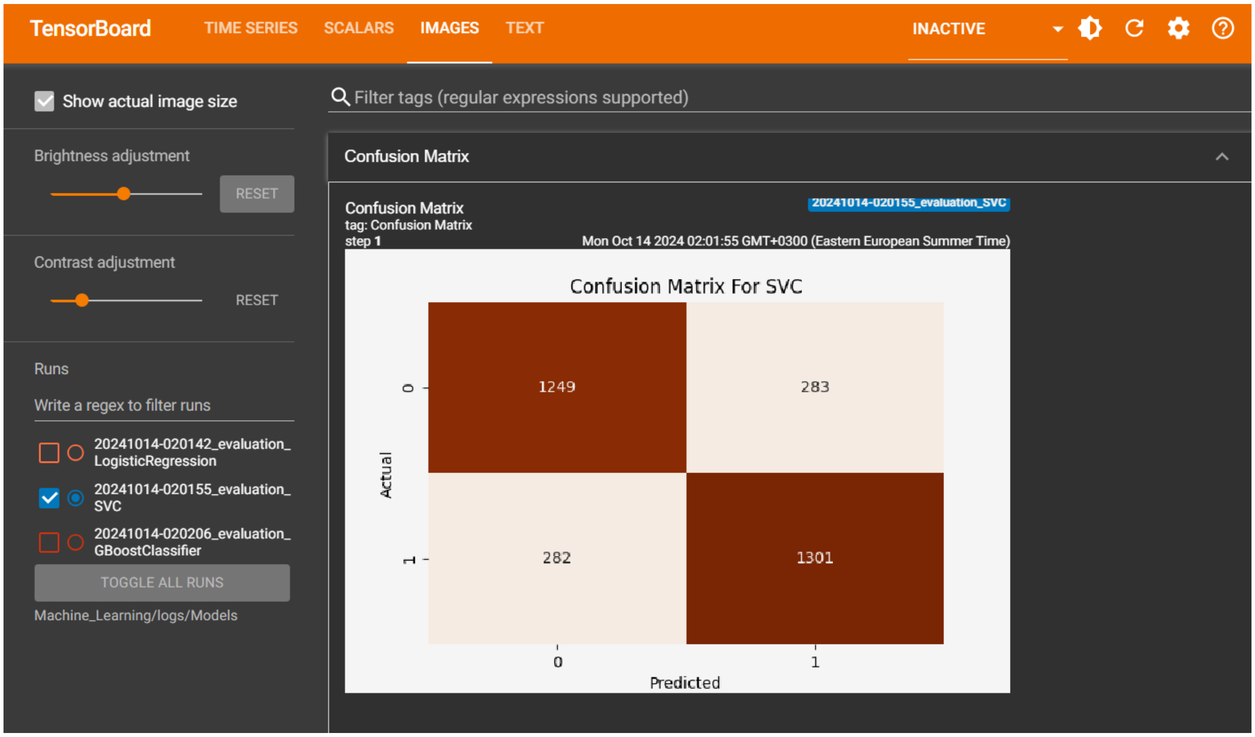


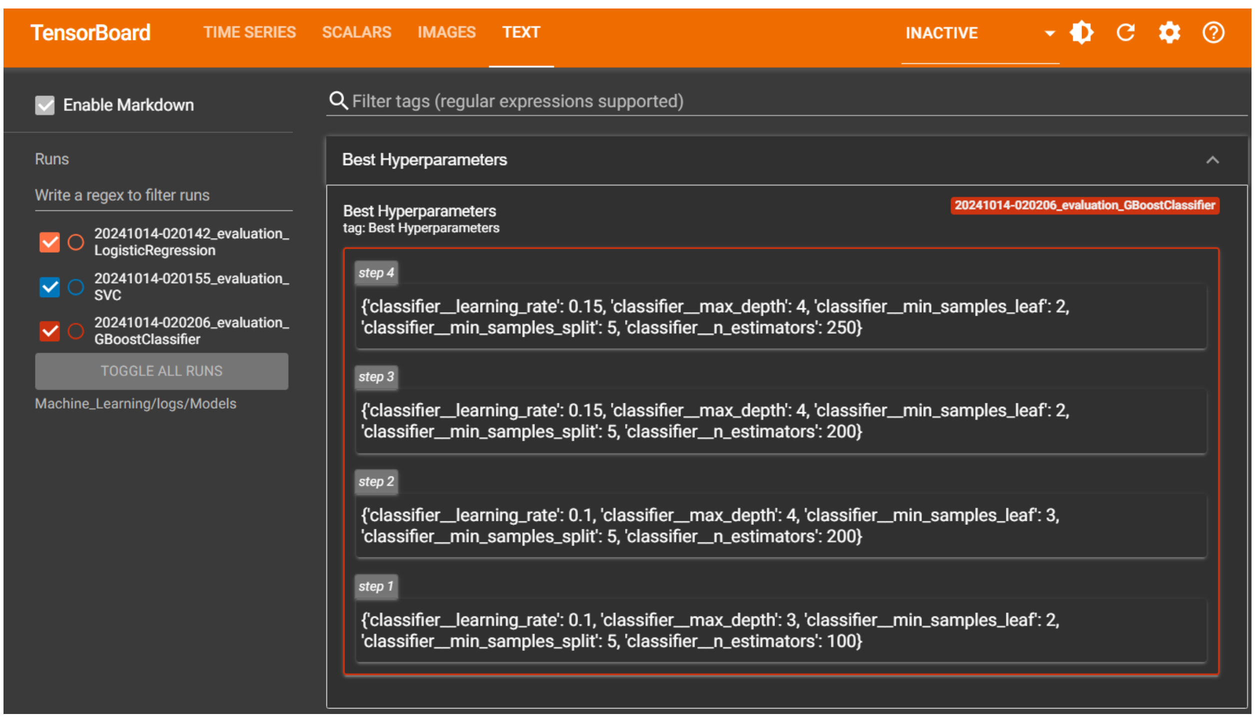



TensorBoard
TIME SERIES
SCALARS
IMAGES
TEXT
INACTIVE
Filter runs (regex)
Filter tags (regex)
All
Scalars
Image
Histogram
Settings
Run
20241014-020142_evaluation_LogisticRe
20241014-020155_evaluation_
20241014-020206_evaluation_GBoostCla
Pinned
AUC-ROC
Accuracy_GradientBoostingClassifier
Accuracy_LogisticRegression
Accuracy_LogisticRegression
Run
Smoothed
Value
Step
Relative
20241014-020142_evaluation_LogisticRegression
0.8196
0.8196
1
0
GENERAL
Horizontal Axis
Step
Enable step selection and data tabl
(Scalars only)
Enable Range Selection
Link by step 4
Card Width
Enable saving pins (Scalars only)
SCALARS
Smoothing
0.6
Tooltip sorting method
Alphabetical

Run
20241014-020142_evaluation_LogisticRe
20241014-020155_evaluation_
20241014-020206_evaluation_GBoostCla
Accuracy_GradientBoostingClassifier
Accuracy_LogisticRegression
Accuracy_SupportVectorClassifier
Accuracy_SupportVectorClassifier
Run
Min
Max
Start Value
End Value
△Value
△%
Start Step
End Ste
20241014-020155_evaluation_SVC
0.8186
0.8186
↑0
↑0%
1
1
Confusion Matrix 3 cards
Enable saving pins (Scalars only)
SCALARS
Smoothing
0.6
Tooltip sorting method
Alphabetical
Ignore outliers in chart scaling
Partition non-monotonic X axis
HISTOGRAMS
Mode
Offset
IMAGES
Brightness
Contrast
Show actual image size


TensorBoard
TIME SERIES
SCALARS
IMAGES
TEXT
INACTIVE
Show actual image size
Filter tags (regular expressions supported)
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter runs
20241014-020142_evaluation_LogisticRegression
20241014-020155_evaluation_SVC
20241014-020206_evaluation_GBoostClassifier
TOGGLE ALL RUNS
Machine_Learning/logs/Models
Confusion Matrix
Confusion Matrix
tag: Confusion Matrix
step 4
20241014-020206_evaluation_GBoostClassifier
Mon Oct 14 2024 02:17:30 GMT+0300 (Eastern European Summer Time)
Confusion Matrix For GBoost Classifier
1966
397
364
1946
Actual
0
1
Predicted


TensorBoard
TIME SERIES
SCALARS
IMAGES
TEXT
INACTIVE
Show actual image size
Filter tags (regular expressions supported)
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter runs
20241014-020142_evaluation_LogisticRegression
20241014-020155_evaluation_SVC
20241014-020206_evaluation_GBoostClassifier
TOGGLE ALL RUNS
Machine_Learning/logs/Models
Confusion Matrix
Confusion Matrix
tag: Confusion Matrix
step 1
20241014-020142_evaluation_LogisticRegression
Mon Oct 14 2024 02:01:42 GMT+0300 (Eastern European Summer Time)
Confusion Matrix for Logistic Regression
1250
282
280
1303
Actual
0
1
Predicted


TensorBoard
TIME SERIES
SCALARS
IMAGES
TEXT
INACTIVE
Show actual image size
Filter tags (regular expressions supported)
Brightness adjustment
RESET
Contrast adjustment
RESET
Runs
Write a regex to filter runs
20241014-020142_evaluation_LogisticRegression
20241014-020155_evaluation_SVC
20241014-020206_evaluation_GBoostClassifier
TOGGLE ALL RUNS
Machine_Learning/logs/Models
Confusion Matrix
Confusion Matrix
tag: Confusion Matrix
step 1
20241014-020155_evaluation_SVC
Mon Oct 14 2024 02:01:55 GMT+0300 (Eastern European Summer Time)
Confusion Matrix For SVC
Actual
Predicted
0
1
1249
283
282
1301


TensorBoard
TIME SERIES
SCALARS
IMAGES
TEXT
INACTIVE
Enable Markdown
Filter tags (regular expressions supported)
Runs
Write a regex to filter runs
20241014-020142_evaluation_LogisticRegression
20241014-020155_evaluation_SVC
20241014-020206_evaluation_GBoostClassifier
TOGGLE ALL RUNS
Machine_Learning/logs/Models
Best Hyperparameters
Best Hyperparameters
tag: Best Hyperparameters
20241014-020206_evaluation_GBoostClassifier
step 4
{'classifier__learning_rate': 0.15, 'classifier__max_depth': 4, 'classifier__min_samples_leaf': 2, 'classifier__min_samples_split': 5, 'classifier__n_estimators': 250}
step 3
{'classifier__learning_rate': 0.15, 'classifier__max_depth': 4, 'classifier__min_samples_leaf': 2, 'classifier__min_samples_split': 5, 'classifier__n_estimators': 200}
step 2
{'classifier__learning_rate': 0.1, 'classifier__max_depth': 4, 'classifier__min_samples_leaf': 3, 'classifier__min_samples_split': 5, 'classifier__n_estimators': 200}
step 1
{'classifier__learning_rate': 0.1, 'classifier__max_depth': 3, 'classifier__min_samples_leaf': 2, 'classifier__min_samples_split': 5, 'classifier__n_estimators': 100}

6. Conclusion

This report covered the workflow for building and evaluating machine learning models using Gradient Boosting Classifier, Logistic Regression, and SVC. Each model provided valuable insights, with Gradient Boosting emerging as the most accurate model. TensorBoard facilitated real-time monitoring of the workflow, contributing to better decision-making during model development. Future improvements could include exploring more advanced models and fine-tuning the existing ones to enhance accuracy further.

Summary of Findings

Throughout the process of building a machine learning model to predict customer churn, several key steps were undertaken, including data preparation, feature engineering, and the use of **PyCaret** for algorithm comparison. The analysis provided important insights into the performance of various models, particularly in terms of their ability to accurately predict which customers are likely to leave the bank.

- **Data Preparation and Augmentation**: The dataset was balanced using data augmentation techniques, ensuring fair representation between the "Exited" and "Not Exited" classes.
- **Handling Outliers**: Outliers were detected and handled in key variables such as **credit_score** and **age**, improving the robustness of the model.
- **Algorithm Comparison**: Using PyCaret, multiple machine learning algorithms were tested, including **Gradient Boosting Classifier (GBC)**, **Random Forest (RF)**, **Logistic Regression (LR)**, and **Support Vector Classifier (SVC)**. Among them, **Gradient Boosting Classifier** outperformed all others with the highest accuracy and AUC score.
- **Model Metrics**: The models were evaluated based on various metrics such as **Accuracy**, **AUC**, **Recall**, **Precision**, and **F1-Score**. The GBC model demonstrated the best balance between precision and recall, making it the most suitable model for this churn prediction task.