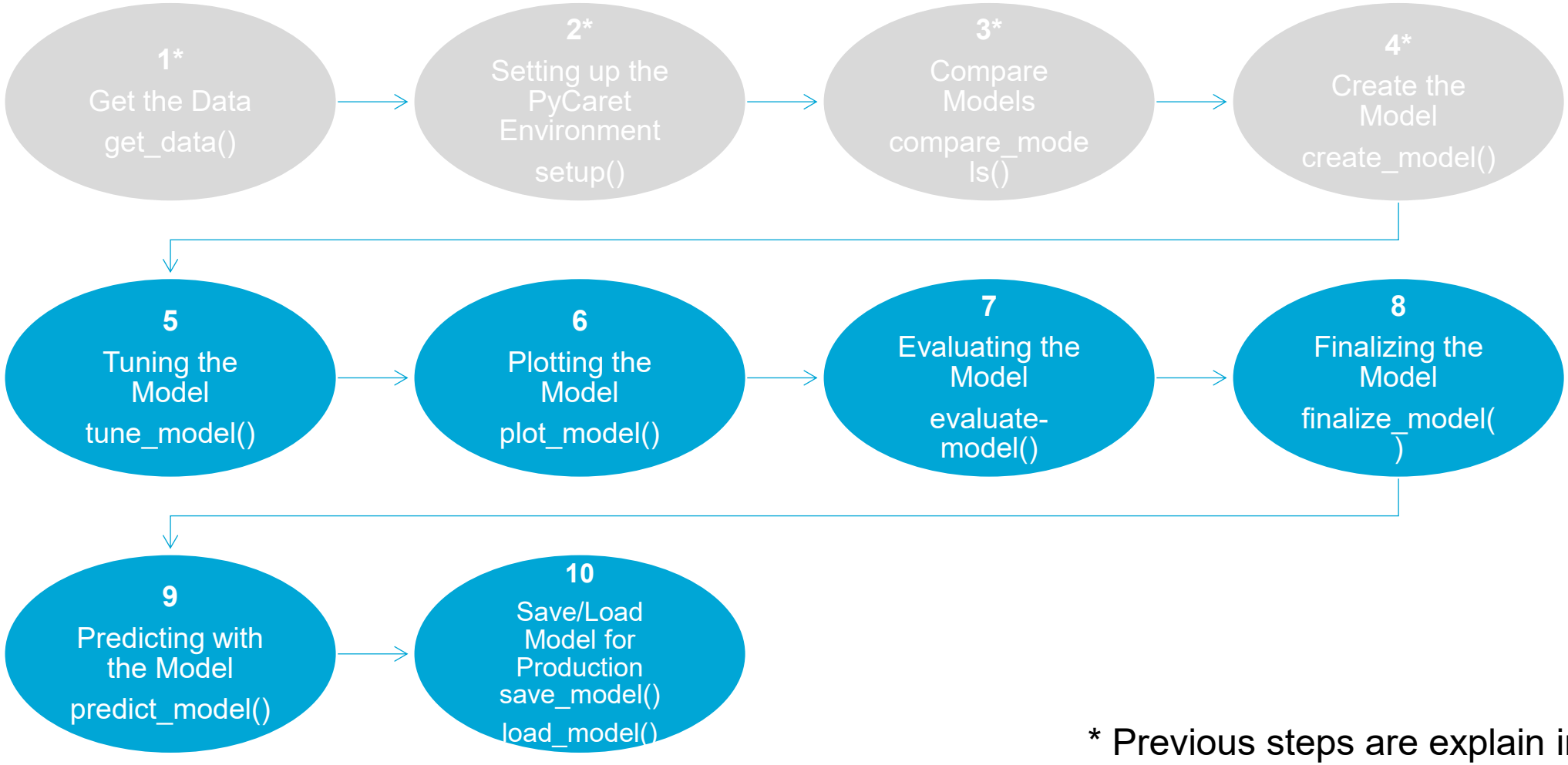
MachineLearnAthon - Microlecture
Low code II – Model evaluation

MachineLearnAthon
A project Co-funded by the Erasmus+ programme of the European Union

Machine Learning Pipeline with PyCaret



* Previous steps are explain in first microlecture

Step 5 Tuning the Model

- Optimizes a model's hyperparameters for better performance.

Default Behavior

- Uses Random Grid Search within a predefined search space. Optimizes for Accuracy by default.

Key Features of tune_model

- Custom Optimization Metrics: Change the metric using the optimize parameter.

Output of tune_model

- Prints a scoring grid with metrics: Accuracy, AUC, Recall, Precision, F1, Kappa, and MCC.
- Compares fold-wise performance to identify the best hyperparameters.

```
[47]  tuned_gbc = tune_model(gbc)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8239	0.7842	0.2975	0.7609	0.4277	0.3464	0.4001
1	0.8133	0.7993	0.2691	0.7037	0.3893	0.3042	0.3534
2	0.8183	0.8217	0.3003	0.7114	0.4223	0.3350	0.3790
3	0.8158	0.7743	0.2975	0.6954	0.4167	0.3275	0.3693
4	0.8233	0.7832	0.2946	0.7591	0.4245	0.3433	0.3972
5	0.8208	0.8014	0.2946	0.7376	0.4211	0.3374	0.3873
6	0.8164	0.7579	0.2861	0.7113	0.4081	0.3221	0.3690
7	0.8158	0.7962	0.2578	0.7398	0.3824	0.3026	0.3611
8	0.8164	0.7665	0.2833	0.7143	0.4057	0.3203	0.3684
9	0.8082	0.7828	0.2805	0.6556	0.3929	0.3000	0.3383
Mean	0.8172	0.7867	0.2861	0.7189	0.4091	0.3239	0.3723
Std	0.0045	0.0176	0.0132	0.0302	0.0153	0.0162	0.0183

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

```
[48]  #tuned model object is stored in the variable 'tuned_dt'.
      print(tuned_gbc)
```

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='log_loss', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=100, n_iter_no_change=None,
                           random_state=123, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

Co-funded by the Erasmus+ programme of the European Union
Machine LearnAthon

Step 6 Plotting the Model - AUC Plot (classification)

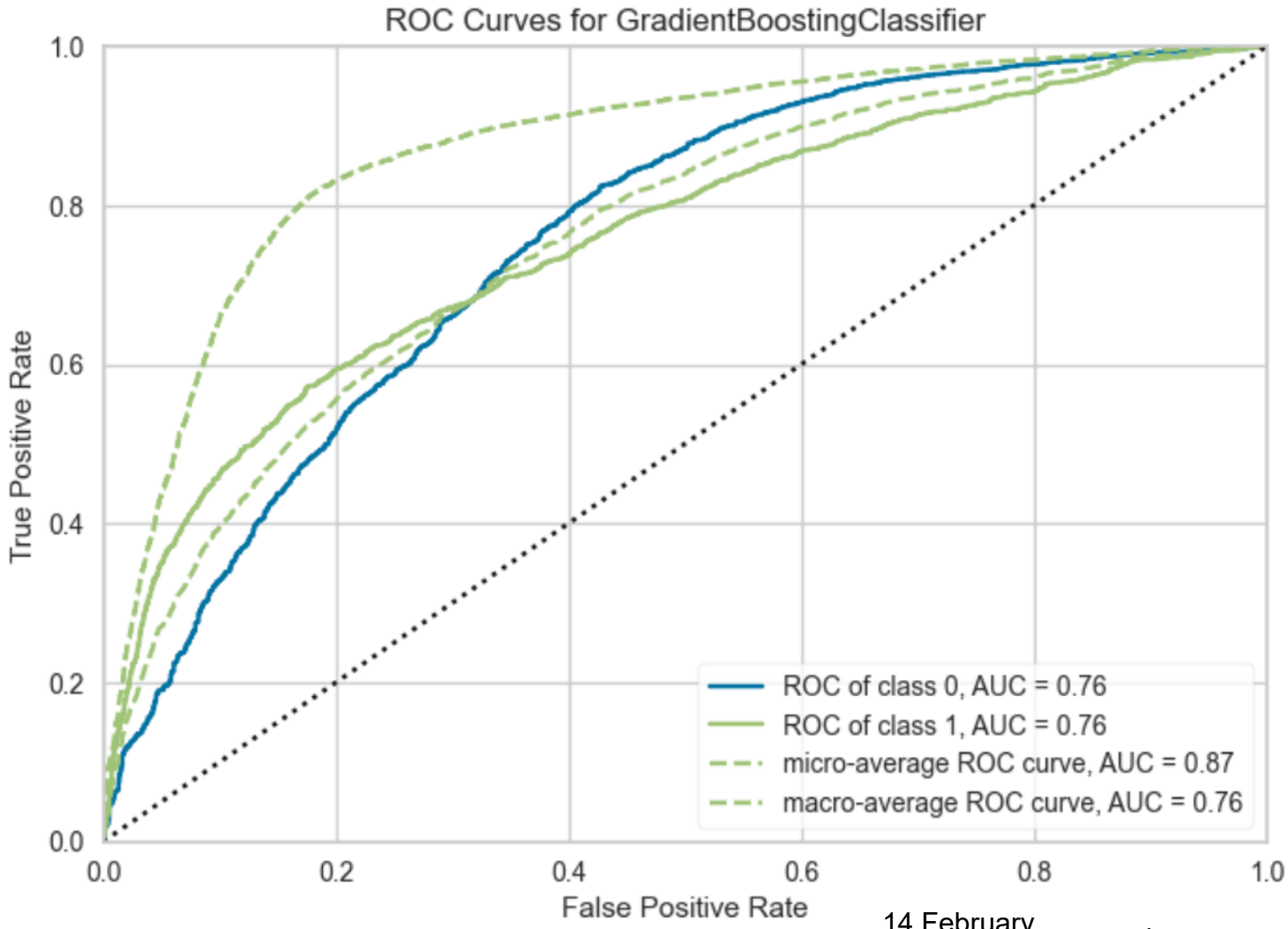
```
## AUC Plot
plot_model(tuned_gbc, plot = 'auc')
```

✓ 0.8s

- ROC curves (Receiver Operating Characteristic) for the Gradient Boosting Classifier model. The ROC curve shows the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) at different decision thresholds of the model.

- The **AUC** represents the area under the curve and indicates how well the model discriminates between classes.
- AUC = 1.0 is a perfect model, AUC = 0.5 is a random model
- AUC = 0.76 for each class means that the model has a moderate ability to discriminate between classes.

- **Micro-average AUC** = 0.87 means that the model shows good overall performance when all classes are combined.

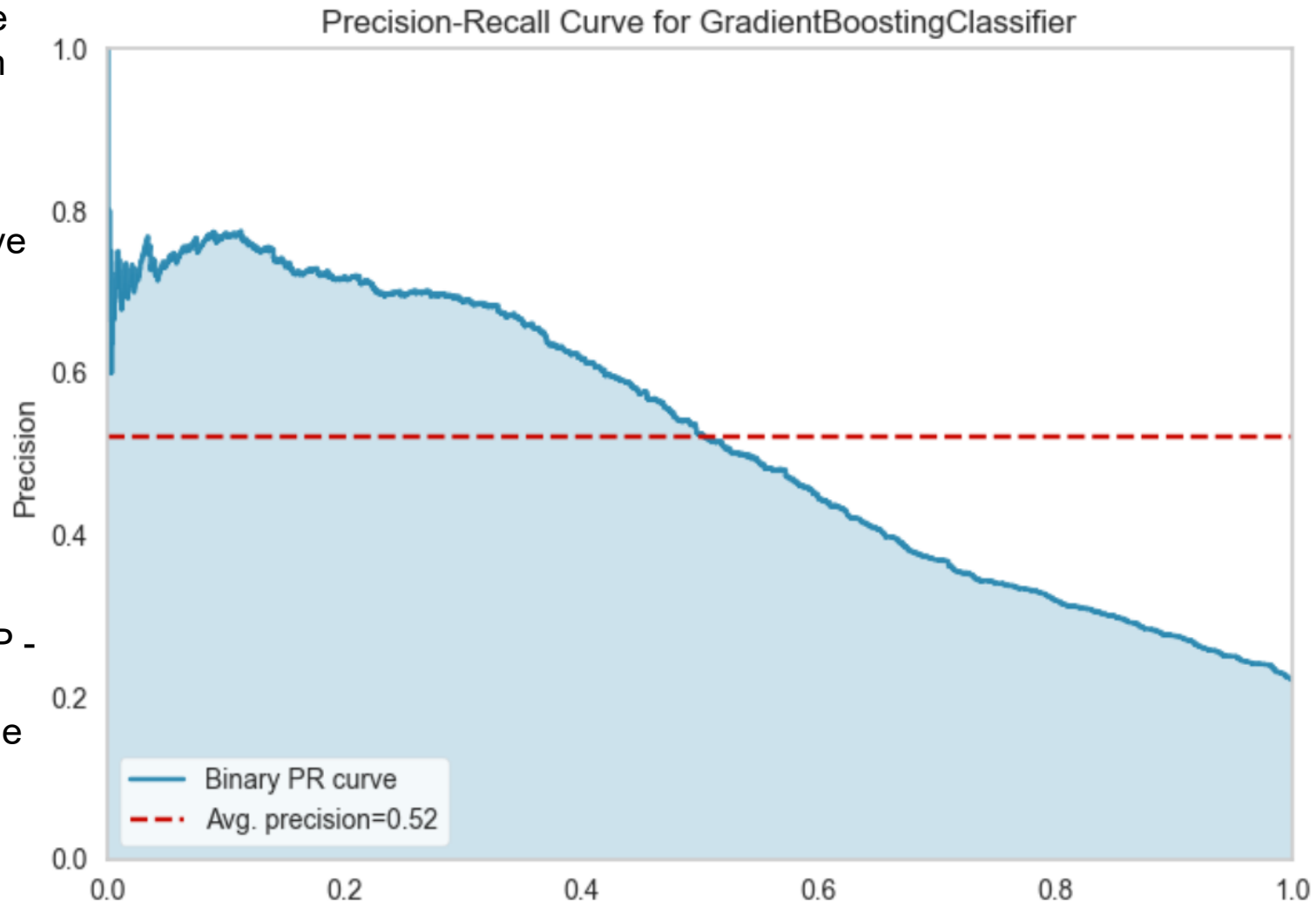


Step 6 Plotting the Model - Precision-recall curve

```
## Precision-recall curve
plot_model(tuned_gbc, plot = 'pr')
```

✓ 0.8s

- The Precision-Recall curve is used to evaluate the performance of classification models, especially in the case of non-equilibrium datasets (when one class is significantly less frequent than the other).
- **Blue curve:** Represents the Precision-Recall curve for the model. It shows how precision changes at different levels of recall.
- **Red dotted line:** Represents the Average Precision, summarizing the performance of the model over the entire PR curve.
- **Area Under the Curve (Precision-Recall AUC):** The area under the PR curve (often also called AP - Average Precision) represents the overall performance of the model. The larger this area, the better the model.

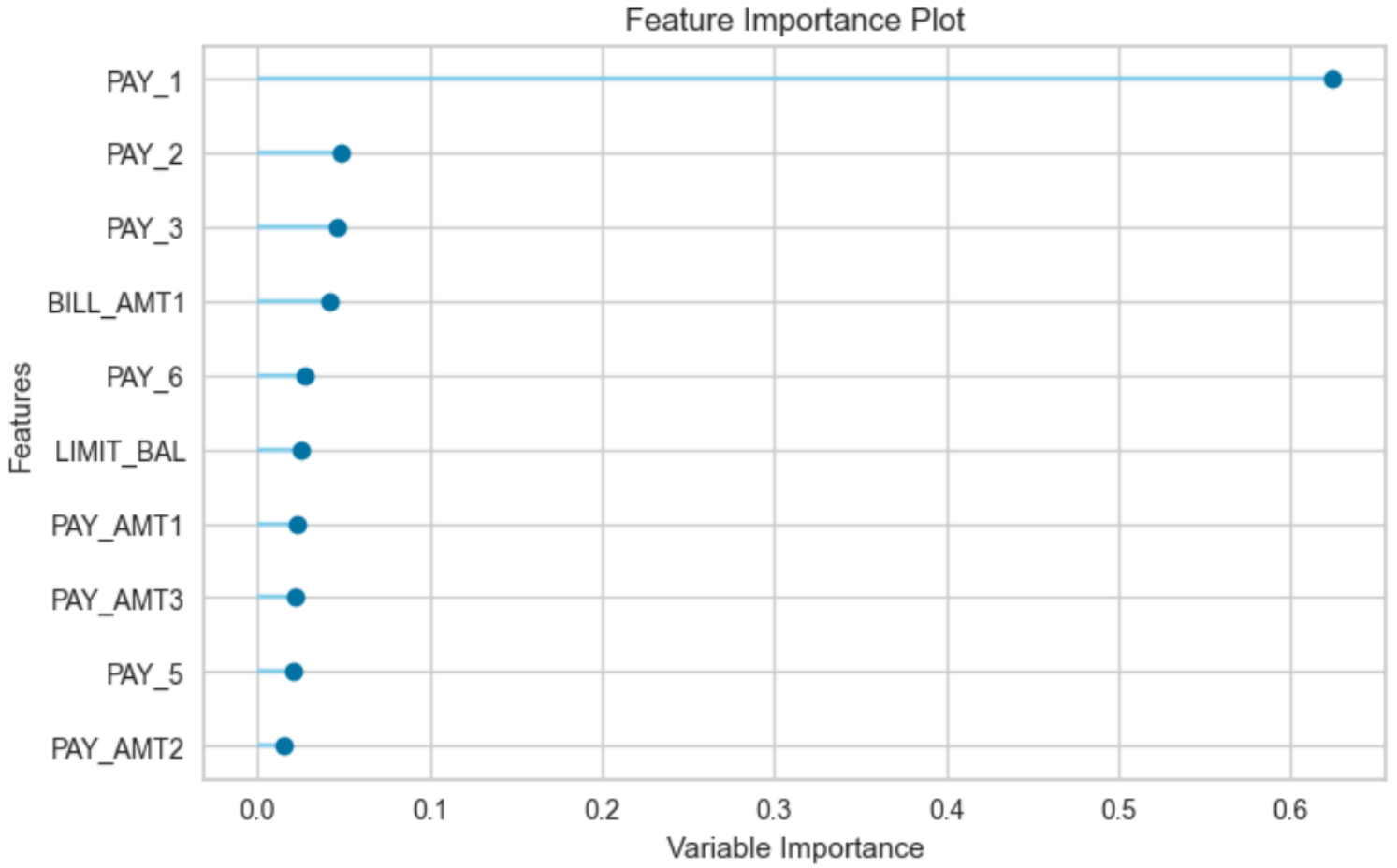


Step 6 Plotting the Model - Feature importance

```
## feature importance
plot_model(tuned_gbc, plot='feature')
```

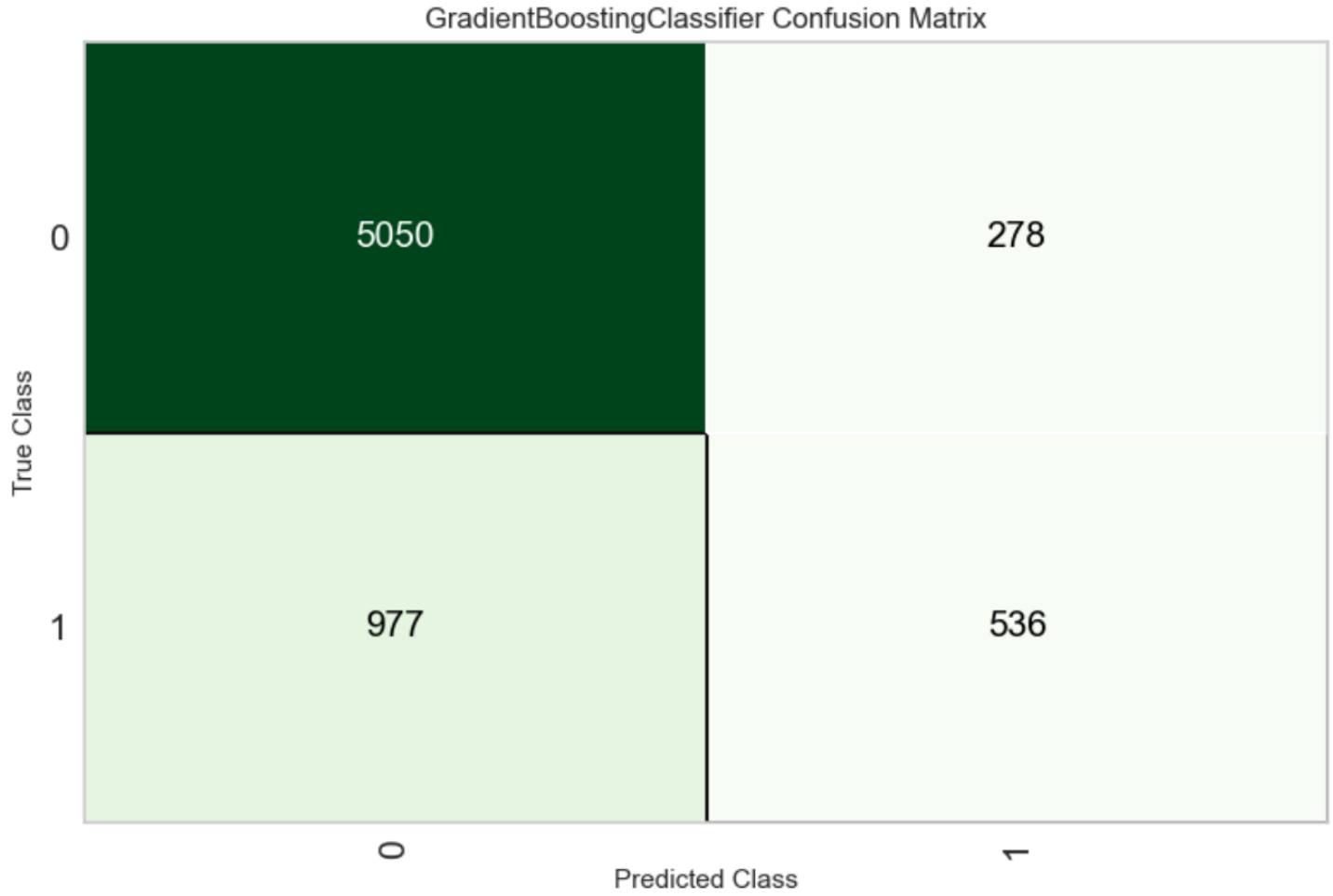
✓ 0.5s

- The graph shows the Feature Importance for the Gradient Boosting Classifier model. Variable Importance determines how much contribution each input variable makes to the prediction of the model.
- **X axis (Variable Importance):** shows the relative importance of variables. Values are normalized to a range of 0 to 1, where 1 represents the most important variable.
- The model relies most heavily on variables with higher importance, so these factors could be key for decision making or prediction of the target variable.



Step 6 Plotting the Model - Consufion matrix

- The confusion matrix is used to evaluate the performance of the classification model by showing the numbers of correct and incorrect predictions for each class.
- **The rows** represent the true classes (True Class).
- **The columns** represent the predicted classes (Predicted Class).
- **True Positives (TP):** the model correctly predicted the positive class (536).
- **True Negatives (TN):** The model correctly predicted the negative class (5050).
- **False Positives (FP):** The model incorrectly labeled the negative case as positive (278).
- **False Negatives (FN):** The model incorrectly flagged the positive case as negative (977).



Step 7 Evaluation the Model

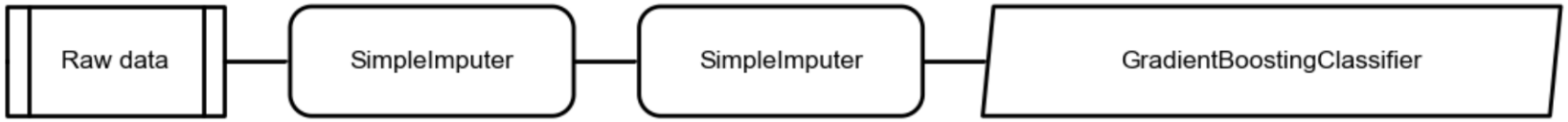
- evaluate_model() function displays a user interface for all available graphics for a given model.

```
evaluate_model(tuned_gbc)
```

[53] ✓ 0.4s Python

Plot Type:

Pipeline Plot	Hyperparameters	AUC	Confusion Matrix	Threshold	Precision Recall	Prediction Error	Class Report	Feature Selection	Learning Curve
Manifold Learning	Calibration Curve	Validation Curve	Dimensions	Feature Importance	Feature Importance…	Decision Boundary	Lift Chart	Gain Chart	Decision Tree
KS Statistic Plot									



Step 8 Finalizing the Model

Objective

- Fits the model to the entire dataset (including the test set).

Why

- Ensures the model learns from all available data before production.

Order of Execution

- Best Practice: Use predict_model() to evaluate the model before finalizing.

```
final_gbc = finalize_model(tuned_gbc)
#Final Random Forest model parameters for deployment
print(final_gbc)
```

[54] ✓ 13.7s

```
Pipeline(memory=Memory(location=None),
         steps=[('numerical_imputer',
                 TransformerWrapper(exclude=None,
                                    include=['LIMIT_BAL', 'SEX', 'EDUCATION',
                                             'MARRIAGE', 'AGE', 'PAY_1',
                                             'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5',
                                             'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
                                             'BILL_AMT3', 'BILL_AMT4',
                                             'BILL_AMT5', 'BILL_AMT6',
                                             'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',
                                             'PAY_AMT4', 'PAY_AMT5',
                                             'PAY_AMT6'],
                                    transform...
                                            criterion='friedman_mse', init=None,
                                            learning_rate=0.1, loss='log_loss',
                                            max_depth=3, max_features=None,
                                            max_leaf_nodes=None,
                                            min_impurity_decrease=0.0,
                                            min_samples_leaf=1,
                                            min_samples_split=2,
                                            min_weight_fraction_leaf=0.0,
                                            n_estimators=100,
                                            n_iter_no_change=None,
                                            random_state=123, subsample=1.0,
                                            tol=0.0001, validation_fraction=0.1,
                                            verbose=0, warm_start=False))],
         verbose=False)
```


technische universität dortmund
TU Delft
Co-funded by the Erasmus+ programme of the European Union
Machine LearnAthon

Step 9 Predicting with the Model

```
predict_model(final_gbc)
```

[55] ✓ 0.5s Python

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default	prediction_label	prediction_scor
576	30000	1	1	1	38	8	7	6	5	4	...	31085.0	0.0	0.0	0.0	0.0	0.0	0.0	1	1	0.672
2833	110000	1	3	1	43	-1	-1	-1	-1	0	...	31073.0	390.0	5050.0	63032.0	1100.0	1100.0	1000.0	0	0	0.828
22415	290000	2	1	2	27	2	0	0	0	2	...	278260.0	16500.0	15000.0	20000.0	10000.0	9701.0	0.0	0	1	0.701
440	110000	2	3	0	31	0	0	0	0	0	...	63208.0	4000.0	5000.0	3000.0	3000.0	3000.0	8954.0	0	0	0.936
4236	360000	1	3	2	34	0	0	0	0	0	...	235916.0	15000.0	9221.0	9225.0	8112.0	8369.0	9000.0	0	0	0.881
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
18193	50000	1	2	1	37	2	0	0	0	0	...	19799.0	2306.0	2155.0	1086.0	19001.0	906.0	1500.0	0	1	0.691
5717	280000	2	1	1	36	-1	-1	-1	-1	-1	...	14106.0	33840.0	8689.0	41652.0	9031.0	14106.0	6680.0	0	0	0.948
21799	10000	1	5	1	43	-1	0	0	0	-2	...	0.0	2537.0	1000.0	0.0	0.0	0.0	0.0	1	0	0.765
7661	280000	2	2	2	27	-1	0	0	0	2	...	117635.0	10077.0	27785.0	30000.0	23.0	20000.0	3406.0	0	0	0.943
6412	190000	2	1	2	31	0	0	0	0	0	...	125841.0	7000.0	7000.0	8500.0	5000.0	6000.0	5000.0	0	0	0.917

6841 rows × 26 columns

```
unseen_predictions = predict_model(final_gbc, data=data_unseen)
unseen_predictions.head()
```

[56] ✓ 0.4s Python

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Gradient Boosting Classifier	0.8175	0.7898	0.3802	0.6410	0.4773	0.3754	0.3942

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default	prediction_label	prediction_score
0	100000	2	2	2	23	0	-1	-1	0	0	...	567.0	380.0	601.0	0.0	581.0	1687.0	1542.0	0	0	0.8716
1	380000	1	2	2	32	-1	-1	-1	-1	-1	...	11873.0	21540.0	15138.0	24677.0	11851.0	11875.0	8251.0	0	0	0.9633
2	200000	2	2	1	32	-1	-1	-1	-1	2	...	3151.0	5818.0	15.0	9102.0	17.0	3165.0	1395.0	0	0	0.8806
3	200000	1	1	1	53	2	2	2	2	2	...	149531.0	6300.0	5500.0	5500.0	5500.0	5000.0	5000.0	1	1	0.7901
4	240000	1	1	2	41	1	-1	-1	0	0	...	1737.0	2622.0	3301.0	0.0	360.0	1737.0	924.0	0	0	0.6919

5 rows × 26 columns

Step 10 Save/Load Model for Production

Purpose

- Retrieve a previously saved model for predictions.

Function

- load_model().

Using the Loaded Model for Predictions

- Apply the loaded model to new or unseen data

  predict_model(loaded_model, data=data_unseen)

Output

- Predictions are concatenated with the original dataset. Automatically applies saved transformations from the pipeline.

```
save_model(final_gbc, "C:/Users/mirka/Desktop/Final RF Model 19Nov2020.xls")
```

[57] ✓ 0.3s

```
Transformation Pipeline and Model Successfully Saved

(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['LIMIT_BAL', 'SEX', 'EDUCATION',
                                              'MARRIAGE', 'AGE', 'PAY_1',
                                              'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5',
                                              'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
                                              'BILL_AMT3', 'BILL_AMT4',
                                              'BILL_AMT5', 'BILL_AMT6',
                                              'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',
                                              'PAY_AMT4', 'PAY_AMT5',
                                              'PAY_AMT6'],
                                     transform...
                                             criterion='friedman_mse', init=None,
                                             learning_rate=0.1, loss='log_loss',
                                             max_depth=3, max_features=None,
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100,
                                             n_iter_no_change=None,
                                             random_state=123, subsample=1.0,
                                             tol=0.0001, validation_fraction=0.1,
                                             verbose=0, warm_start=False))],
                                     verbose=False),
 'C:/Users/mirka/Desktop/Final RF Model 19Nov2020.xls.pkl')
```

Thank you very much for your attention!

Literature

- Alamin, M. A. A., & Uddin, G. (2018). Challenges and barriers of using low code software for machine learning. *ACM Transactions.*
- Data Valley. (n.d.). *Kaggle survey insights and analysis*. Retrieved December 1, 2024, from https://datavalley.technology/kaggle/
- Hasim, N., & Haris, N. A. (2015). A study of open-source data mining tools for forecasting. *Proceedings of the ACM International Conference.* https://doi.org/10.1145/2701126.2701152
- Kaggle. (2021). *Kaggle State of Machine Learning and Data Science Report 2021*. Retrieved from https://storage.googleapis.com/kaggle-media/surveys/Kaggle%20State%20of%20Machine%20Learning%20and%20Data%20Science%20Report%202021.pdf
- Kaggle. (2022). *Kaggle State of Machine Learning and Data Science Report 2022*. Retrieved from https://storage.googleapis.com/kaggle-media/surveys/Kaggle%20State%20of%20Machine%20Learning%20and%20Data%20Science%20Report%202022.pdf

technische universität dortmund
TU Delft
Co-funded by the Erasmus+ programme of the European Union
Machine LearnAthon

Literature (cont.)

- Naik, A., & Samant, L. (2016). Correlation review of classification algorithm using data mining tools: WEKA, Rapidminer, Tanagra, Orange, and Knime. *Procedia Computer Science, 85,* 662-668. https://doi.org/10.1016/j.procs.2016.05.251
- Ng Mh. (2024). A comparison of the top 3 tools for machine learning beginners: Orange vs PyCaret vs Scikit. *Tinkercademy Build Log.* Retrieved from https://blog.tinkercademy.com/orange-v-s-pycaret-v-s-scikit-a-comparison-of-beginner-machine-learning-libraries-2d79c0e43e3f
- Rexer Analytics. (2023). *Data Science Survey 2023: Highlights from the TechCast (July 27, 2023)*. Retrieved December 1, 2024, from https://andouc.org/wp-content/uploads/2023/10/DSS-2023-Highlights-REXER-TechCast-7-27-23-CDChanges.pdf
- Tolios, G. (2023). *Simplifying Machine Learning with PyCaret: A Low-code Approach for Beginners and Experts!* Leanpub.
- Wahbeh, A. H., Al-Radaideh, Q. A., Al-Kabi, M. N., & Al-Shawakfa, E. (2011). A comparison study between data mining tools over some classification methods. *International Journal of Advanced Computer Science and Applications.* https://doi.org/10.14569/SpecialIssue.2011.010304
- Yan, Z. (2022). The impacts of low/no-code development on digital transformation and software development. *University of Toronto.* Retrieved from https://arxiv.org/pdf/2112.14073v1.pdf
- Yeh, I.-C. (2016). *Default of credit card clients dataset*. UCI Machine Learning Repository. University of California, Irvine. https://doi.org/10.24432/C55S3H

The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.