Projektbericht

Credit Card Default als binäres Klassifikationsproblem

Projekt:	Alfa_ML2.0 DC
Voraussetzung:	Lastenheft
Autoren:	Alexander Alexandrov Mehmet Fatih Kaya
Datum :	24. Januar 2022

Inhaltsverzeichnis

1 Zielbestimmungen

1.1 Klärung der Aufgabenstellung

Ziel ist es anhand verschiedener Kundendaten (Geschlecht, Bildung, Alter, Kreditumfang) und Kreditstatistik der letzten sechs Monate den Kreditausfall vorherzusagen. Das heißt, es handelt sich um eine Klassifikation. Beim Data Mining können Entscheidungsbäume als mathematische Vehrfahren zur Beschreibung, Klassifizierung und Zusammenfassung eines Datensatzes verwendet werden, der wie folgt beschrieben werden kann: $(x, Y) = (x_1, x_2, x_3 \ldots x_k, Y)$

Die abhängige Variable Y ist die Zielvariable, die analysiert, klassifiziert und verallgemeinert werden soll. Der Vektor X besteht aus Eingangsvariablen X1, X2, X3 ... Xk, usw., die zur Erfüllung dieser Aufgabe verwendet werden.

Grundlegende Hypothese:

Wir vermuten, dass die Faktoren wie Geschlecht, Bildung, Alter, Kreditlimit , Kreditstatistik der letzten sechs Monate die wichtigsten Kriterien sein werden. Wir starten die Analyse mit allen Spalten und werden diese Hypothese anhand der Faktoren verifizieren

Als Scores stehen die Accuracy, die Precision, der Recall und der f1-Score in der engeren Auswahl. Da es am wichtigsten ist, dass Kreditausfall auch korrekt klassifiziert werden, entscheiden wir uns für den Recall dieser Klasse, behalten aber anfangs auch noch die Precision im Auge. Im Idealfall sollte sie ebenfalls hoch sein, damit möglichst wenig Kreditausfälle fälschlicherweise zugeordnet werden. Insgesamt ist eine niedrige Precission aber als weniger schwer zu betrachten als ein niedriger Recall. Erstrebenswert wäre ein Recall von 0.80, das heißt 80% der Kreditausfall sollten erkannt werden.

Machine Learning-Verfahren: **überwachtes, binäre unbalancierte Klassifizierung**

1.2 Beschreibung der Daten

Der Datensatz enthält Informationen über 30.000 Bankkunden (23 Spalten).

Gesamtgröße: 20.7012 MB

Die Zielspalte als Zielklasse kann 2 Werte haben:

1 - ja (Kreditkartenausfall ist eingetreten) **;**

0 - nein (Kreditkartenausfall ist nicht eingetreten)

Zielspalte: default_payment_next_month

Originaldatensatz: https://archive.ics.uci.edu/ml/machine-learning-databases/00350/clients.xls

1.3 Arbeitsschwerpunkte

Aufgabenzuweisung im Team und Implementierungen:

Alexander Alexandrov: EDA, Decision tree classifier + Hyperparameter und die Verfeinerung mittels Grid Search und Cross-Vlidation

Random Forest Pipeline, Gradient Boosting Trees Pipeline + Hyperparameter die Verfeinerung mittels Randomized Search.

Mehmet Fatih Kaya : LightGBM Pipeline, XGBoost Pipeline und die Verfeinerung mittels Randomized Search.

2 Beschreibung der Daten

2.1 Laden des Datensatzes und Verwaltung der Datentypen

s. Anhang **Credit_Card_Default_Part_1** und detaillierte Analyse, Abschnitt 1.1

2.2 EDA

s. Anhang **Credit_Card_Default_Part_1** und detaillierte Exploratory Data Analyse, Abschnitt 1.2

2.3 Aufteilung der Daten in Trainings- und Testsatz

s. Anhang **Credit_Card_Default_Part_1** und detaillierte Beschreibung, Abschnitt 1.3

2.4 Aufteilung in Trainings-, Validierungs- und Testsätze zur Evaluierung der Hyperparameter.

s. Anhang **Credit_Card_Default_Part_1** und detaillierte Beschreibung, Abschnitt 1.4.

2.5 Umgang mit fehlenden Werten

s. Anhang **Credit_Card_Default_Part_1** und detaillierte Beschreibung, Abschnitt 1.5.

2.6 Enkodierung kategorialer Variablen

s. Anhang **Credit_Card_Default_Part_1** und detaillierte Beschreibung, Abschnitt 1.6.

3 Datenvorbereitung

3.1 Datensatzes und Verwaltung der Datentypen

Der Entscheidungsbaum erfordert keine Skalierung und Normalisierung von Merkmalen, die Unterstützung numerischer und kategorialer Merkmale, die Handhabung von Nicht-Linearität in Daten.

4 Anwendung verschiedener Algorithmen

4.1 Analyse und Wahl des Algorithmus

Wenn die Ergebnisse in diskreter Form vorliegen bzw. sind die Werte qualitativ, spricht man von einem Klassifikationsproblem. Die Lösung für unser Problem ist ein binäre unbalancierte Klassifikationsproblem aus dem Bereich des maschinellen Lernens (Überwachtes Lernen).

Die Hauptklassen des Datensatzes sind kategoriale Variablen.
Der Datensatz enthält Informationen über 30.000 Bankkunden (23 Spalten).

Die Zielspalte als Zielklasse kann 2 Werte haben:

1 - ja (Kreditkartenausfall ist eingetreten) ;
0 - nein (Kreditkartenausfall ist nicht eingetreten)

Gewählter Algorithmus-Typ: **Entscheidungsbaum-Klassifikator**

Ein Entscheidungsbaum-Klassifikator ist ein relativ einfacher, aber sehr wichtiger Algorithmus für maschinelles Lernen, sowohl für Regressions- als auch für Klassifikationsprobleme. Die Entscheidungsbäume segmentieren den Merkmalsraum in eine Reihe kleinerer Regionen, indem sie die Merkmale wiederholt bei einem bestimmten Wert aufteilen. Dazu verwenden sie einen Greedy-Algorithmus (zusammen mit einigen Heuristiken), um eine Aufteilung zu finden, die die kombinierte Unreinheit der Kindknoten minimiert (gemessen anhand der Gini oder Entropie).

Vorteile von Entscheidungsbäumen:

- Leichte Visualisierung in Form eines Baumes - hohe Interpretierbarkeit
- Schnelle Trainings- und Vorhersagephasen
- Eine kleine Anzahl von Hyperparametern zum Abstimmen
- Unterstützung numerischer und kategorialer Merkmale
- Handhabung von Nicht-Linearität in Daten
- Erfordert keine Skalierung und Normalisierung von Merkmalen
- Nichtparametrisches Modell - keine Annahmen über die Verteilung der Merkmale/des Ziels.

Nachteile von Entscheidungsbäumen:

- Die Bäume reagieren sehr empfindlich auf das Rauschen in den Eingabedaten, eine kleine Änderung in den Daten kann das Modell erheblich verändern;
- Überanpassung - Wenn wir keine Maximalwerte oder Stoppkriterien vorgeben, neigen die Bäume dazu, sehr tief zu wachsen und sind nicht gut zu verallgemeinern.
- Informationsgewinn (eine Abnahme der Entropie) in einem Entscheidungsbaum mit kategorialen Variablen führt zu einem verzerrten Ergebnis für Merkmale mit einer höheren Anzahl von Kategorien.

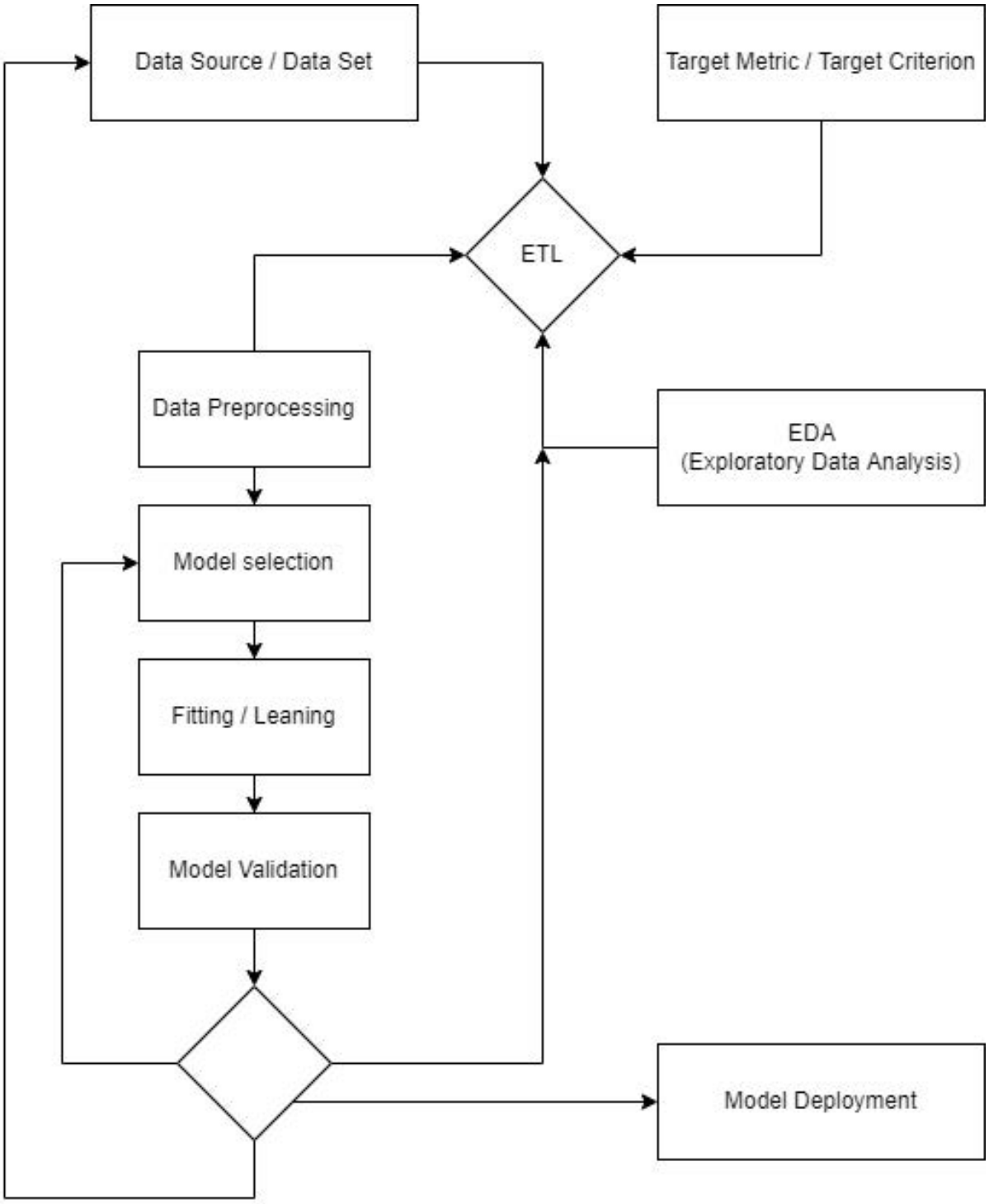
Entscheidungsbäume bilden die Grundlage für viele komplexe Algorithmen, die in dieser Studie werden verwendet:

Bootstrap aggregating (Bagging): **Random Forest, Random Forest für imbalancierten Daten (Over-/ Undersampling), Oversampling mittels SMOTE, Oversampling mittels ADASYN, Balanced RandomForest, Random Forest (sample weights)**

Boosting : **Gradient Boosted Trees, XGBoost, LightGBM**

Ensemblemethoden: **DecisionTree-LogisticRegression-KNeighborsClassifier-GaussianNB**

4.2 ML Model Prozess & Konzept



4.3 Metrikauswahl

Um die Qualität des Modells zu bewerten, müssen wir eine Metrik zur Bewertung der Qualität der Leistung des Modells auswählen.

Das Verständnis der Feinheiten hinter diesen Metriken ist sehr wichtig für die korrekte Bewertung der Leistung des Modells. Die Genauigkeit kann im Falle eines Klassenungleichgewichts sehr irreführend sein. Stellen Sie sich einen Fall vor, in dem 99 % der Daten nicht betrügerisch sind und nur 1 % betrügerisch sind. Dann erreicht ein naives Modell, das jede Beobachtung als nicht betrügerisch einstuft, eine Genauigkeit von 99 %, während es eigentlich wertlos ist. Deshalb sollten wir in solchen Fällen auf Precision oder Recall verweisen. Wenn wir versuchen, eine möglichst hohe Genauigkeit zu erreichen, erhalten wir weniger falsch positive Ergebnisse auf Kosten von mehr falsch negativen Ergebnissen. Bei der Optimierung des Rückrufs erzielen wir weniger falsch negative Ergebnisse auf Kosten von mehr falsch positiven Ergebnissen. Die Metrik, die wir zu optimieren versuchen, sollte basierend auf dem Anwendungsfall ausgewählt werden.

Die ROC-Kurve verliert ihre Glaubwürdigkeit, wenn es darum geht, die Leistung des Modells zu bewerten, wenn wir es mit einem Klassenungleichgewicht zu tun haben. Deshalb sollten wir in solchen Fällen eine andere Kurve verwenden - die **Precision-Recall-Kurve**. Denn bei der Berechnung von Precision und Recall werden die echten Negative nicht berücksichtigt, sondern nur die korrekte Vorhersage der Minderheitsklasse (der positiven Klasse).

Als zusammenfassende Metrik können wir die Fläche unter der **Precision-Recall curve** annähern, indem wir **metrics.auc(recall, precision)** aufrufen. Im Gegensatz zur ROC-AUC reicht die **PR-AUC von 0 bis 1**, wobei 1 für das perfekte Modell steht. Ein Modell mit einem PR-AUC von 1 kann alle positiven Beobachtungen identifizieren (perfekter **Recall**), ohne dass eine einzige negative Beobachtung fälschlicherweise als positiv eingestuft wird (perfekte **Precision**). Wir können Modelle, die sich dem **Punkt (1,1)** nähern, als geschickt betrachten.

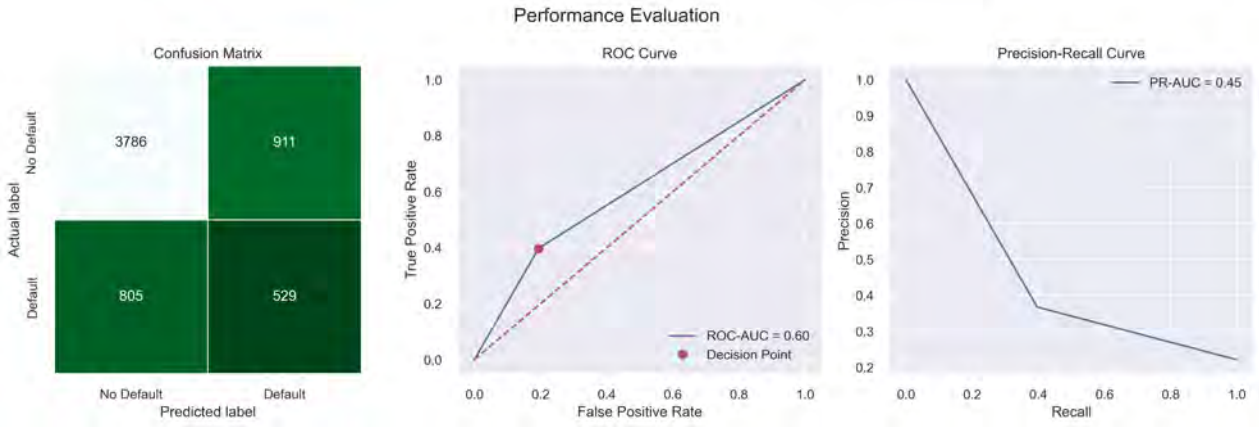
Recall ist eine Metrik, die die Anzahl der korrekten positiven Vorhersagen aus allen positiven Vorhersagen, die hätten gemacht werden können, quantifiziert.

Im Gegensatz zur **precision**, die sich nur auf die richtigen positiven Vorhersagen unter allen positiven Vorhersagen bezieht, gibt die Rückrufquote Aufschluss über fehlende positive Vorhersagen. Auf diese Weise liefert der Recall eine Vorstellung von der Abdeckung der positiven Klasse.

Metrik: Recall (Precision-Recall curve (PR-AUC)

4.4 Anpassung eines Entscheidungsbaum-Klassifikators / Decision tree

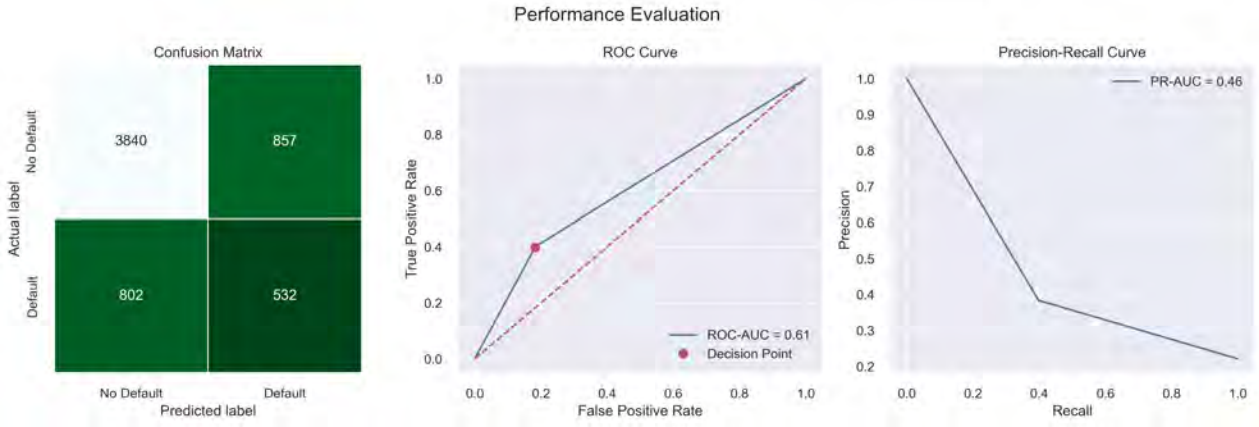
4.4.1 Decision tree implementation



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.7154700712982922,
'precision': 0.36736111111111114,
'recall': 0.39655172413793105,
'f1_score': 0.3813987022350397,
'roc_auc': 0.6016888830441072,
'pr_auc': 0.44881290572571897

4.4.2 Decision tree implementation mittels scikit-learn's Pipeline



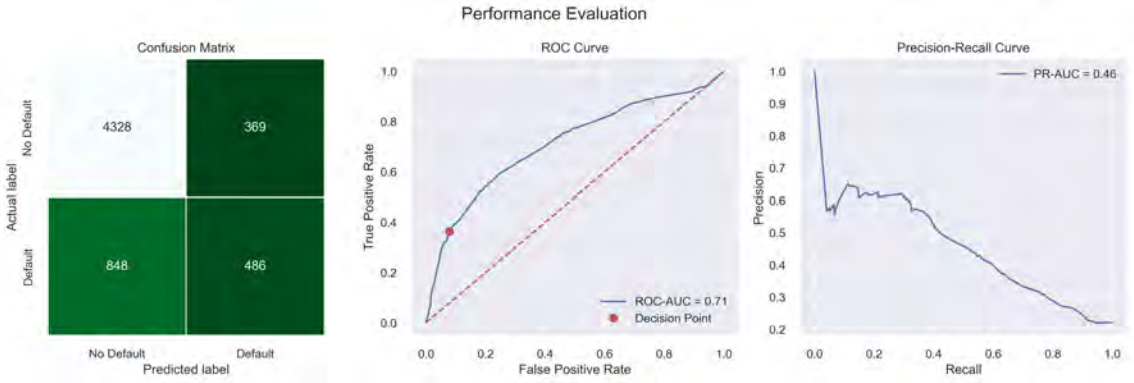
Bewertung unseres Algorithmus und Modells:

'accuracy': 0.7249212402586636,
'precision': 0.3830093592512599,
'recall': 0.3988005997001499,
'specificity': 0.8175431126250798,
'f1_score': 0.3907455012853471,
'cohens_kappa': 0.2131969484517079,
'roc_auc': 0.6088353789892365,
'pr_auc': 0.4576457222303991

4.4.2 Verfeinerung von Hyperparametern mittels Grid Search und Cross-Validatio

Bewertung mit Grid Search:

Best parameters: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_leaf': 5, 'preprocessor__numerical__outliers__n_std': 3}
Recall (Training set): 0.3959
Recall (Test set): 0.3643

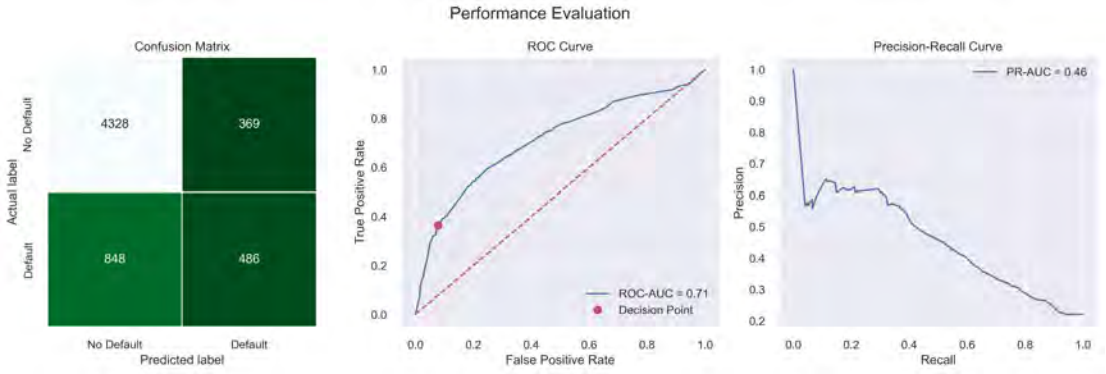


Bewertung unseres Algorithmus und Modells:

'accuracy': 0.7982092521969822,
'precision': 0.5684210526315789,
'recall': 0.36431784107946025,
'roc_auc': 0.7109416869168141,
'pr_auc': 0.45681381023741036

Bewertung mit RandomizedSearchCV:

Best parameters: {'preprocessor__numerical__outliers__n_std': 4, 'classifier__min_samples_leaf': 5, 'classifier__max_depth': 10, 'classifier__criterion': 'gini'}
Recall (Training set): 0.3946
Recall (Test set): 0.3643

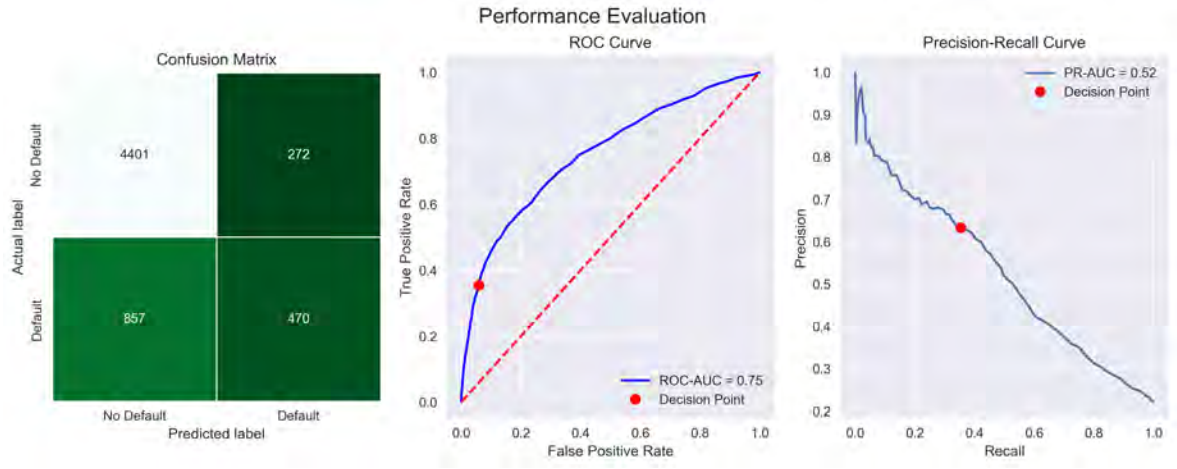


Bewertung unseres Algorithmus und Modells:

' accuracy': 0.7982092521969822,
'precision': 0.5684210526315789,
'recall': 0.36431784107946025,
'roc_auc': 0.7105784450759505,
'pr_auc': 0.4567479301300528}

4.5 Anpassung einer Random Forest Pipeline + Hyperparameter

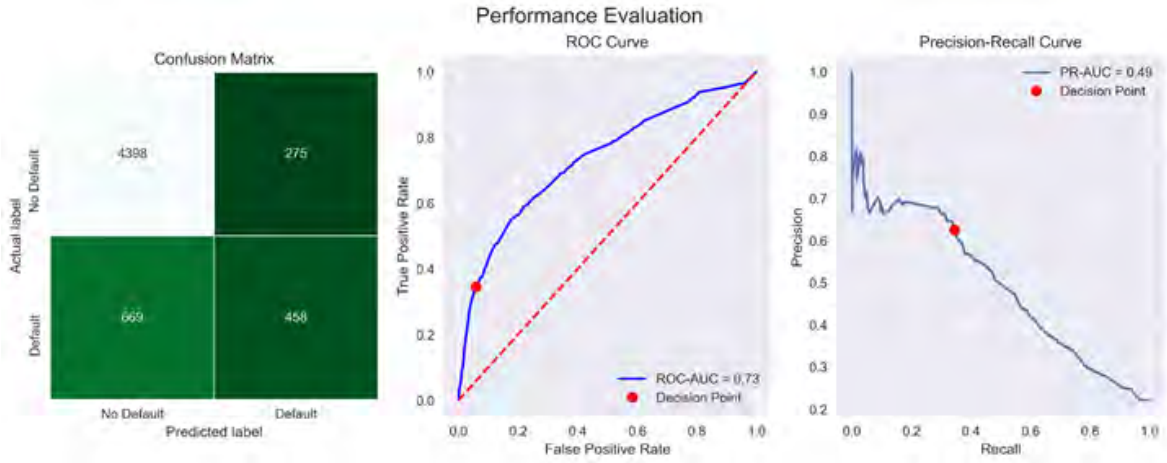
4.5.1 Random Forest Pipeline implementation mittels scikit-learn's Pipeline



Bewertung unseres Algorithmus und Modells:

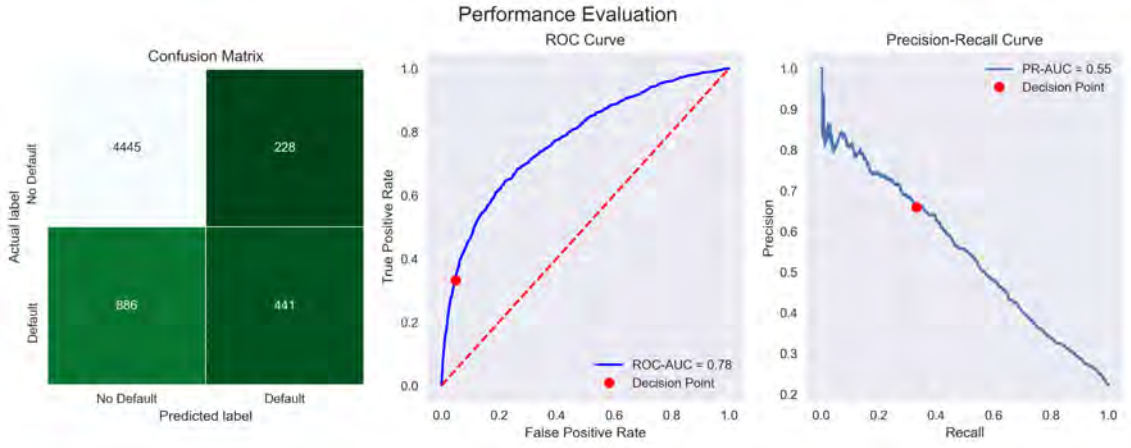
'accuracy': 0.8118333333333333,
'precision': 0.633423180592992,
'recall': 0.3541823662396383,
'f1_score': 0.4543257612373127,
'roc_auc': 0.7517763463762952,
'pr_auc': 0.5247244574365608}

4.5.2 Verfeinerung von Hyperparametern mittels Randomized Search



4.6 Anpassung eine Gradient Boosting Pipeline + Hyperparameter

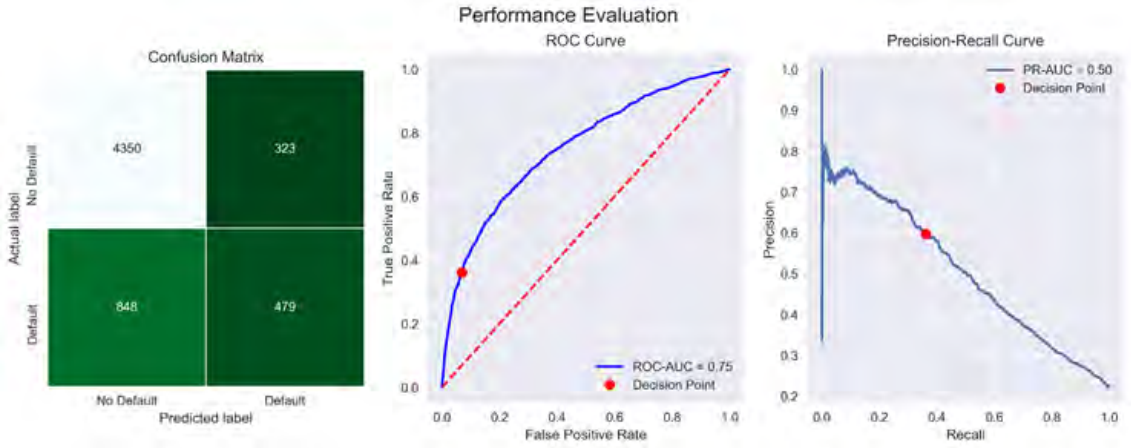
4.6.1 Gradient Boosting Pipeline implementation mittels scikit-learn's



Bewertung unseres Algorithmus und Modells:

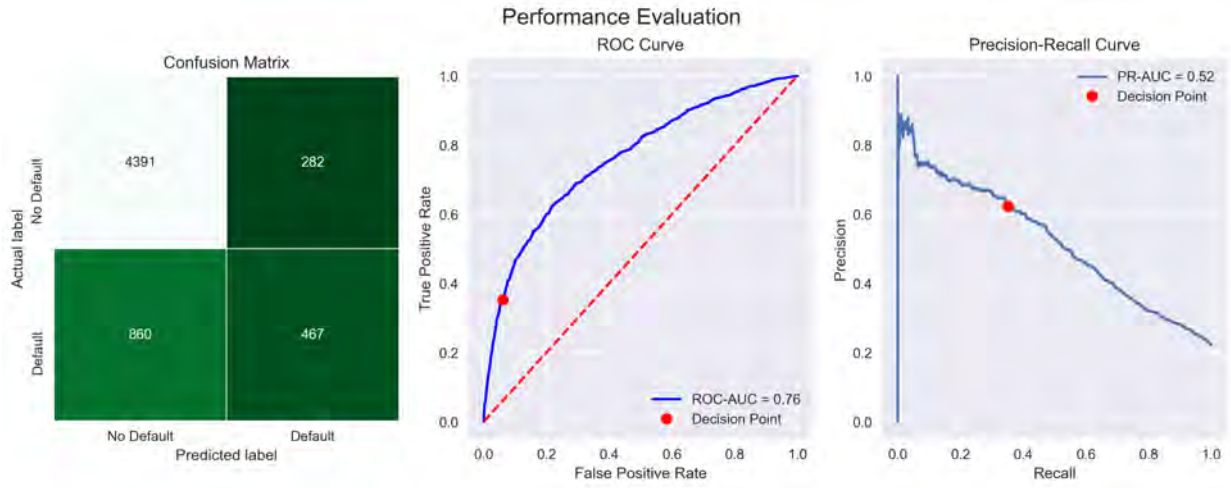
{'accuracy': 0.8143333333333334,
'precision': 0.6591928251121076,
'recall': 0.33232856066315,
'f1_score': 0.44188376753507014,
'roc_auc': 0.775486121671563,
'pr_auc': 0.5474742234717066}

4.6.2 Verfeinerung von Hyperparametern mittels Randomized Search

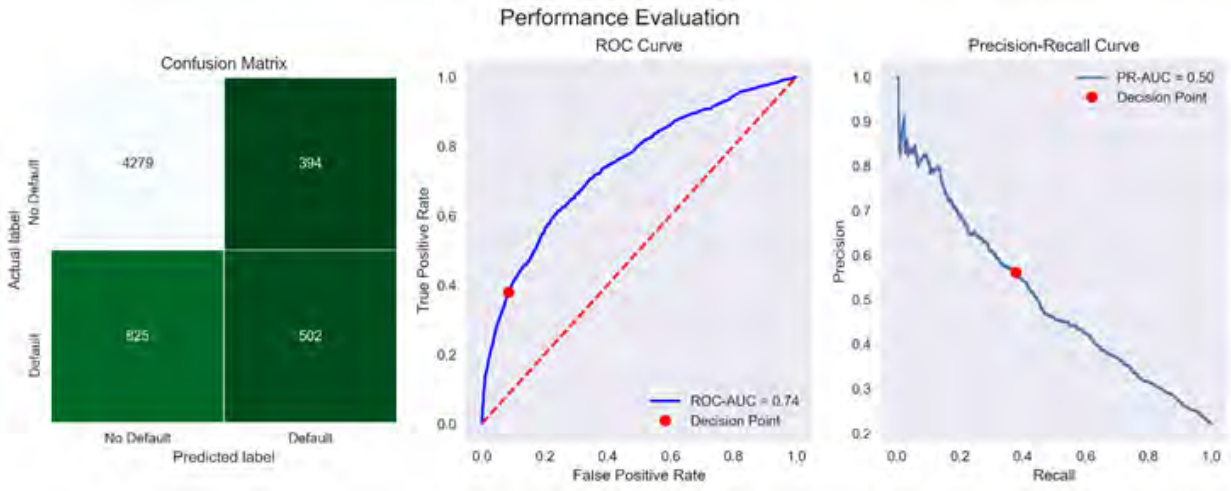


4.7 Anpassung eine XGBoost Pipeline + Hyperparameter

4.7.1 XGBoost Pipeline implementation mittels scikit-learn's Pipeline

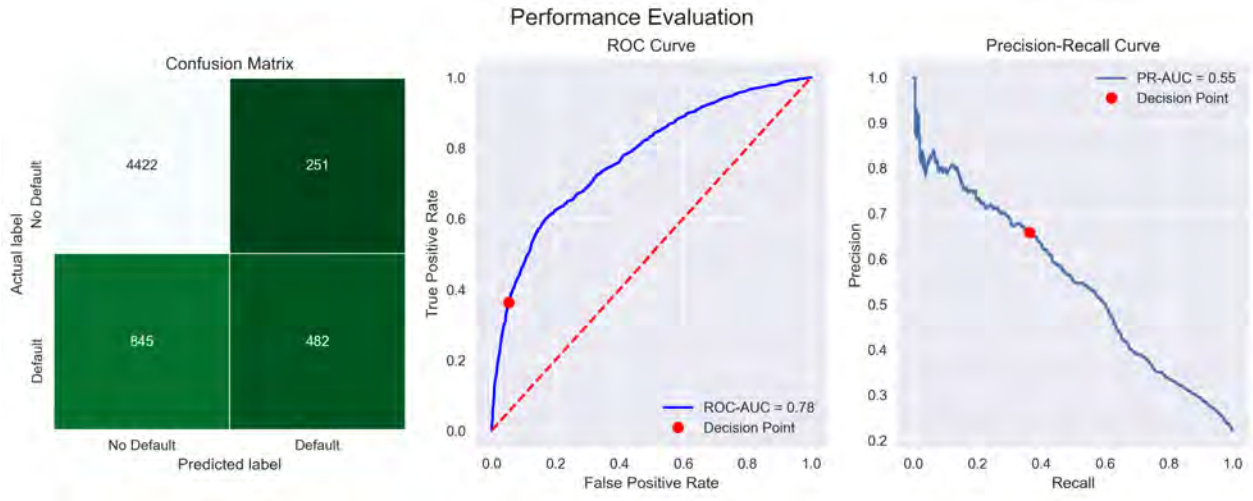


4.7.2 Verfeinerung von Hyperparametern mittels Randomized Search

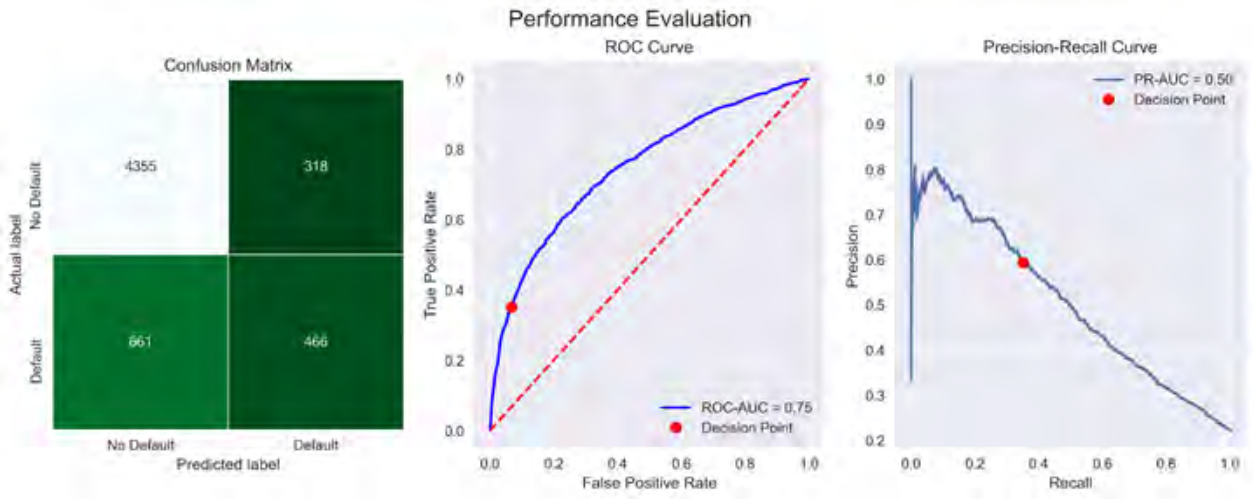


4.8 Anpassung eine LightGBM Pipeline + Hyperparameter

4.8.1 LightGBM Pipeline implementation with scikit-learn's Pipeline

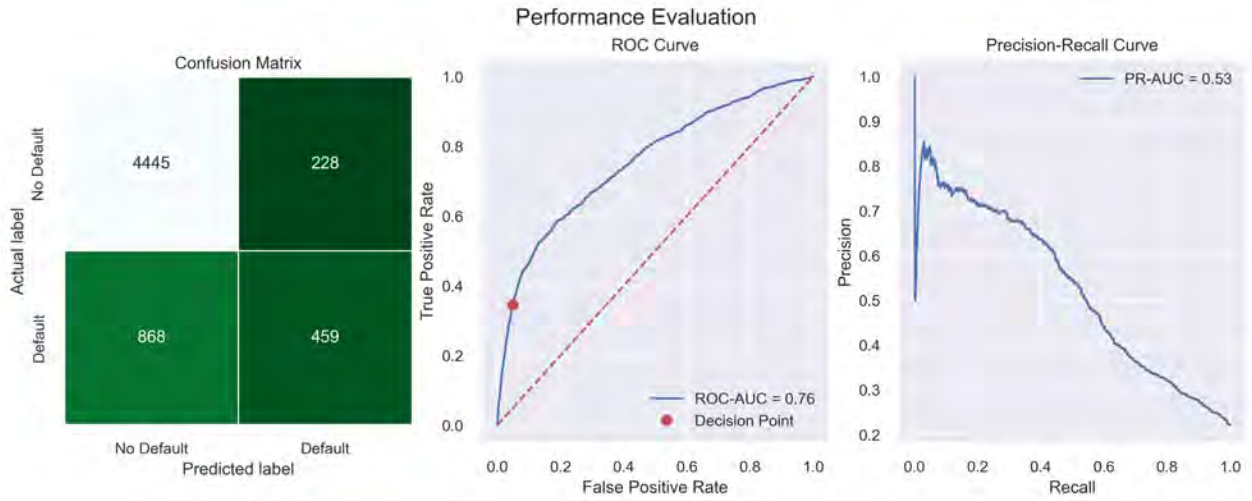


4.8.2 Verfeinerung von Hyperparametern mittels Randomized Search



4. 9 Anpassung eines Ensemble-StackingClassifier

4.9.1 Ensemble: DecisionTree-Logistic Regression-KNeighbors-GaussianNB:

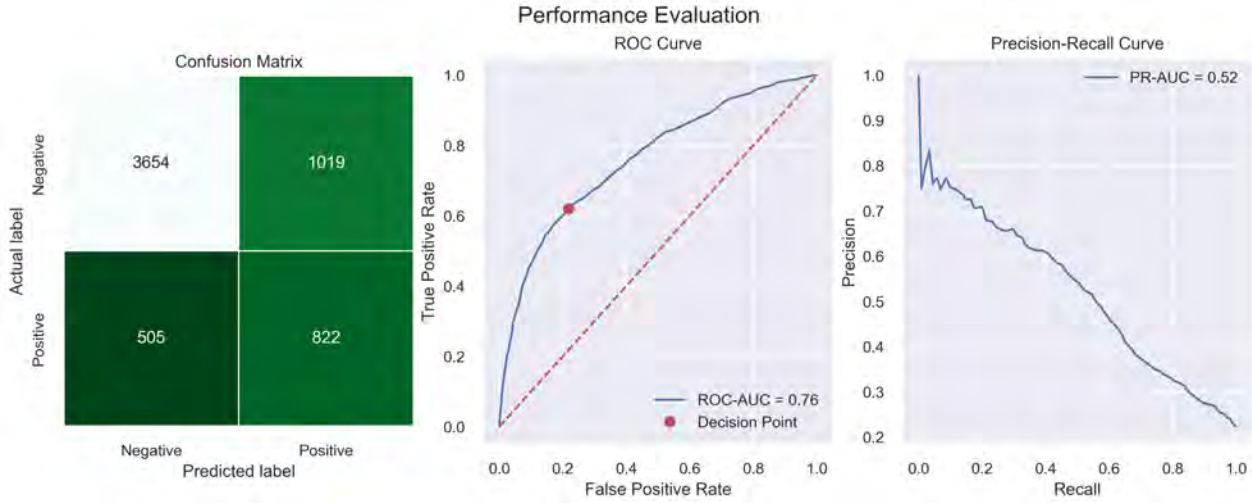


Bewertung unseres Algorithmus und Modells:

'accuracy': 0.8173333333333334,
'precision': 0.6681222707423581,
'recall': 0.3458929917106255,
'roc_auc': 0.7559731214172519,
'pr_auc': 0.5254897277576952}

4. 10 Anpassung eines RandomForest-Klassifikators für imbalancierten Daten (Oversampling - Undersampling)

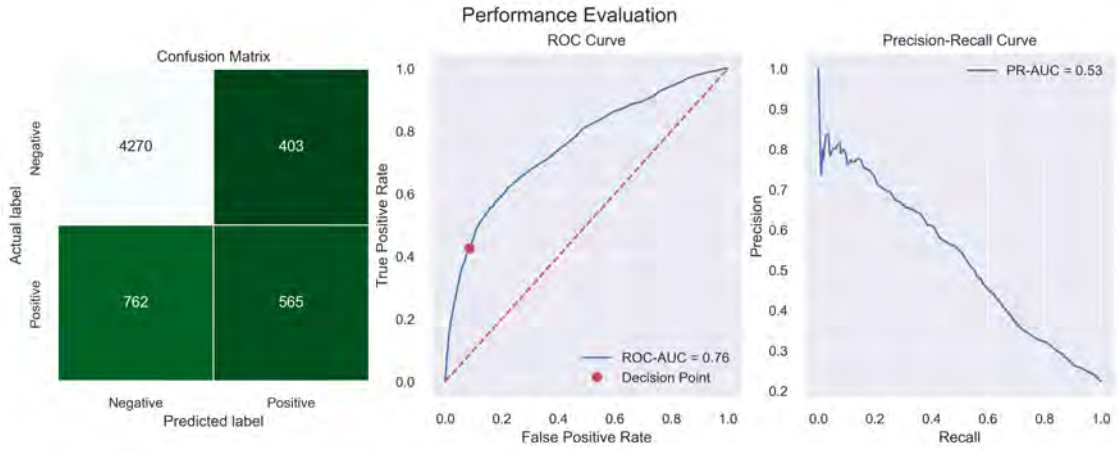
4.10.1 Undersample der Daten und Training eines Random Forest Klassifikators:



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.746,
'precision': 0.4464964693101575,
'recall': 0.6194423511680482,
'roc_auc': 0.7613026040179189,
'pr_auc': 0.5208692474960201}

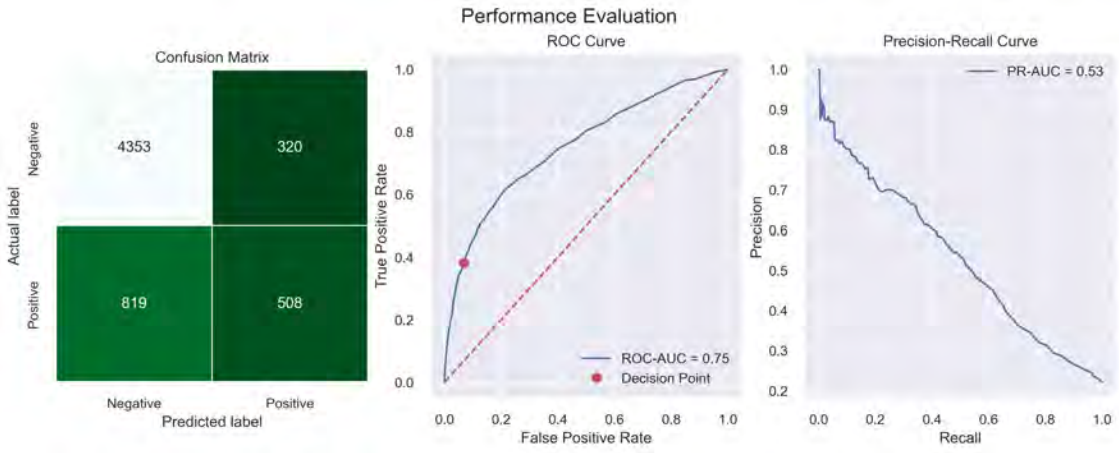
4.10.2 Oversample der Daten und Training eines Random Forest Classifiers:



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.8058333333333333,
'precision': 0.5836776859504132,
'recall': 0.42577241899020346,
'roc_auc': 0.755885475267095,
'pr_auc': 0.526641779935374}

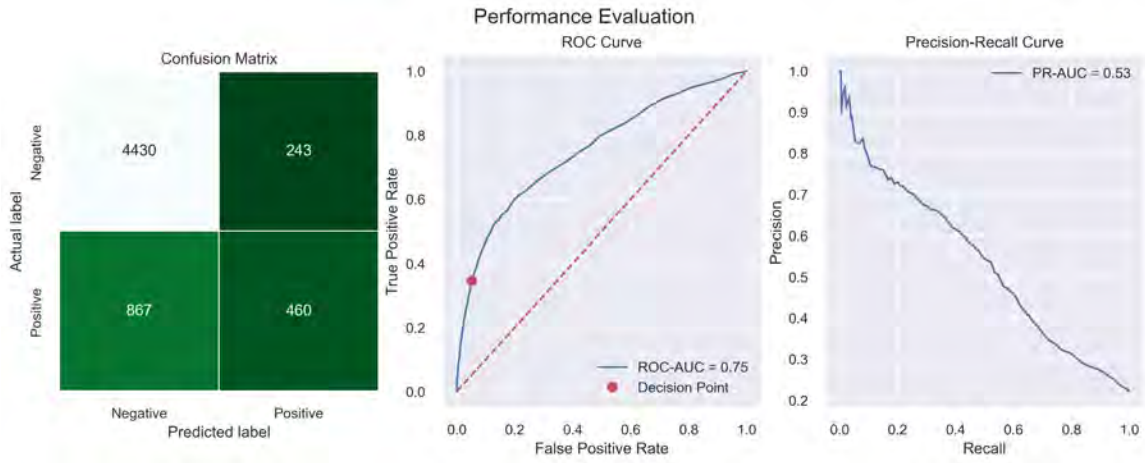
4.10.3 Oversample mittels SMOTE:



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.8101666666666667,
'precision': 0.6135265700483091,
'recall': 0.38281838733986434,
'roc_auc': 0.7524079308235626,
'pr_auc': 0.5306677739830754

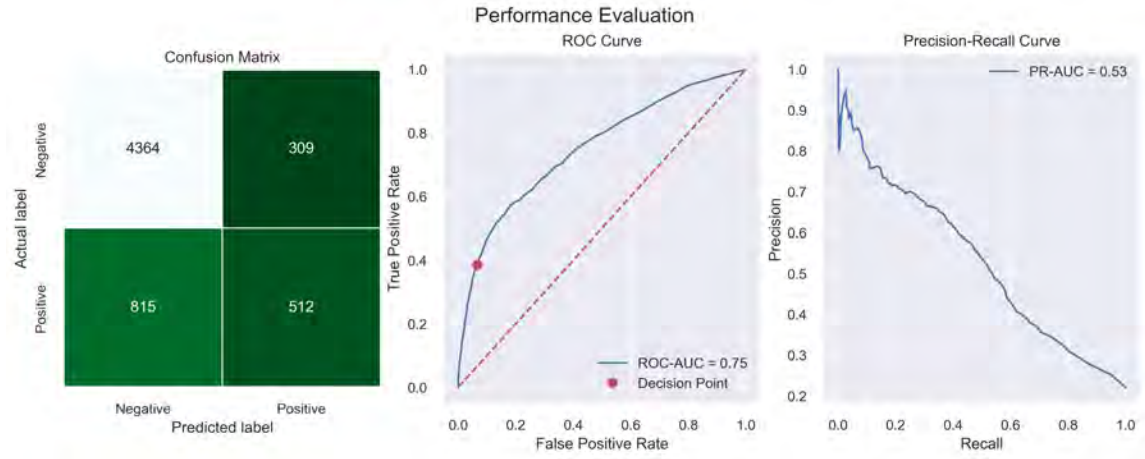
4.10.4 RandomForest-Klassifikator mittels "sample weights":



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.815,
'precision': 0.6543385490753911,
'recall': 0.346646571213263,
'f1_score': 0.4532019704433497,
'roc_auc': 0.7526559524959479,
'pr_auc': 0.5335040052024946

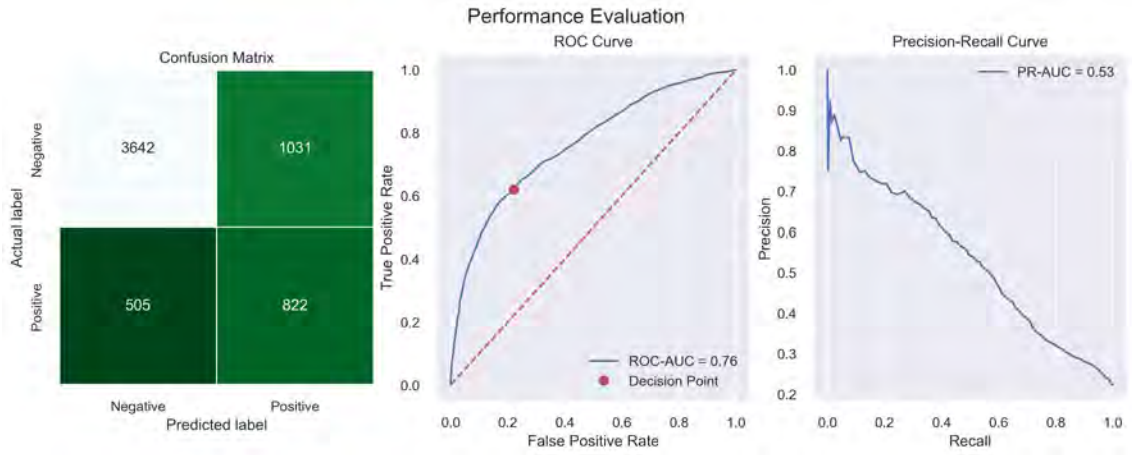
4.10.5 Oversample mittels ADASYN:



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.8126666666666666,
'precision': 0.6236297198538368,
'recall': 0.3858327053504145,
'roc_auc': 0.7510607441843514,
'pr_auc': 0.5288956224554586

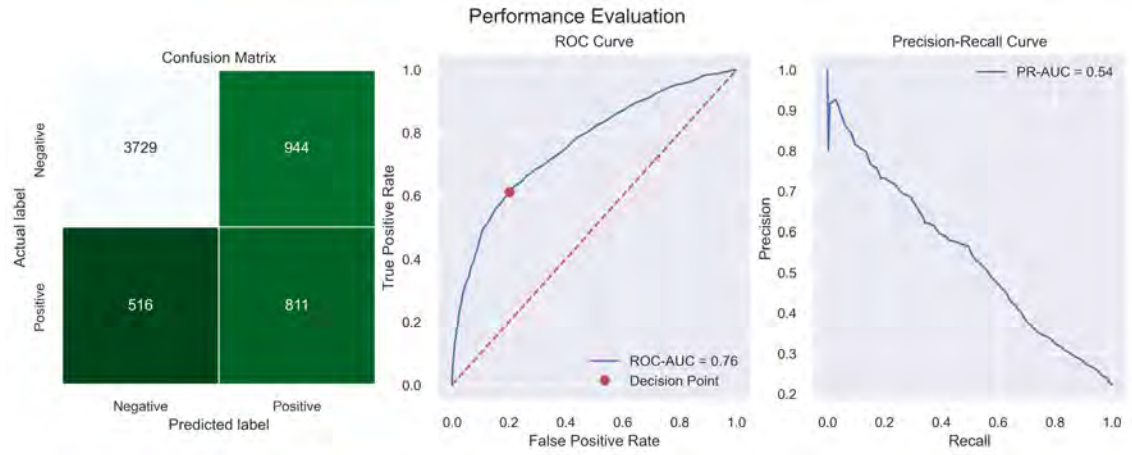
4.10.6 Balanced Random Forest :



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.744,
'precision': 0.4436049649217485,
'recall': 0.6194423511680482,
'f1_score': 0.5169811320754718,
'roc_auc': 0.7638698218420656,
'pr_auc': 0.5334272891290391

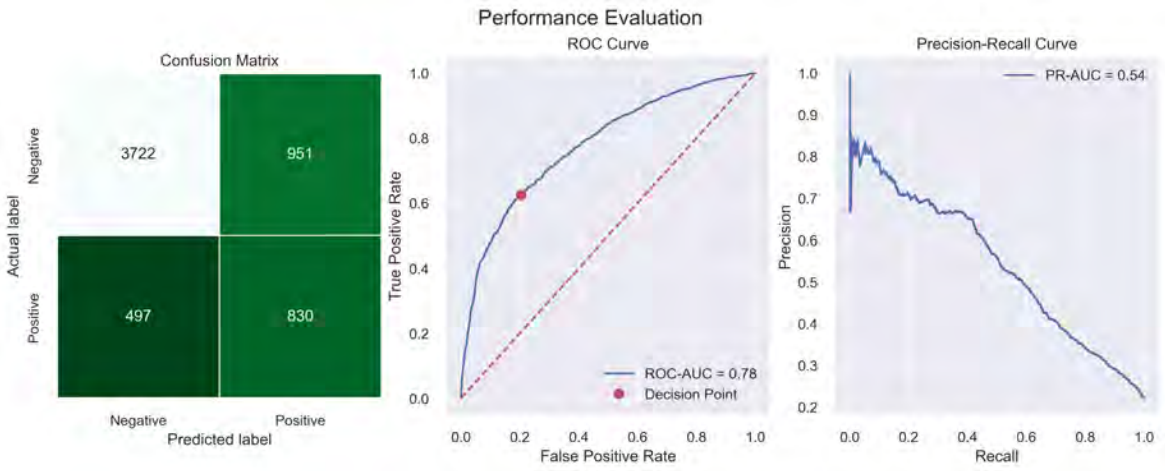
4.10.7 Balanced Random Forest mit balancierten Klassen:



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.7566666666666667,
'precision': 0.4621082621082621,
'recall': 0.6111529766390355,
'f1_score': 0.5262816353017522,
'roc_auc': 0.7644640740285025,
'pr_auc': 0.5408843641248839

4.10.8 Bayesian Hyperparameter Optimization :



Bewertung unseres Algorithmus und Modells:

'accuracy': 0.7586666666666667,
'precision': 0.46603032004491857,
'recall': 0.6254709871891485,
'f1_score': 0.5341055341055341,
'roc_auc': 0.7779632744085658,
'pr_auc': 0.5419185644746215

5 Kommentar des Endergebnisses

5.1 Bewertung unserer Algorithmen und Modelle

Im ersten Durchlauf entscheiden wir uns alle kennengelernten Algorithmen einmal zu testen. Dazu werden die Default-Parameter verwendet und sowohl Precision, Recall, als auch der f1-Score ausgeben gelassen. Die Ergebnisse sind in Tabelle wie folgt zusammengefasst.

	accuracy	precision	recall	specificity	f1_score	cohens_kappa	roc_auc	pr_auc
decision_tree_baseline	0.723333	0.381663	0.404672	0.813824	0.392831	0.213880	0.609528	0.458925
random_forest	0.811833	0.633423	0.354182	0.941793	0.454326	0.351443	0.751776	0.524724
random_forest_rs	0.809333	0.624829	0.345139	0.941151	0.444660	0.340926	0.732860	0.492833
gradient_boosted_trees	0.814333	0.659193	0.332329	0.951209	0.441884	0.344736	0.775486	0.547474
gradient_boosted_trees_rs	0.804833	0.597257	0.360965	0.930880	0.449977	0.340002	0.751316	0.503885
xgboost	0.809667	0.623498	0.351922	0.939653	0.449904	0.345443	0.761328	0.520238
xgboost_rs	0.796833	0.560268	0.378297	0.915686	0.451642	0.332665	0.742675	0.502855
light_gbm	0.817333	0.657572	0.363225	0.946287	0.467961	0.368580	0.775395	0.547444
light_gbm_rs	0.803500	0.594388	0.351168	0.931949	0.441497	0.331712	0.748019	0.504453

	accuracy	precision	recall	specificity	f1_score	cohens_kappa	roc_auc	pr_auc
random_forest	0.813000	0.641379	0.350414	0.944361	0.453216	0.351935	0.753151	0.534631
undersampled rf	0.746000	0.446496	0.619442	0.781939	0.518939	0.352499	0.761303	0.520869
oversampled_rf	0.805833	0.583678	0.425772	0.913760	0.492375	0.375945	0.755885	0.526642
smote	0.810167	0.613527	0.382818	0.931522	0.471462	0.363242	0.752408	0.530668
adasyn	0.812667	0.623630	0.385833	0.933875	0.476723	0.370253	0.751061	0.528896
random_forest_cw	0.815000	0.654339	0.346647	0.947999	0.453202	0.354291	0.752656	0.533504
balanced_random_forest	0.744000	0.443605	0.619442	0.779371	0.516981	0.349251	0.763870	0.533427
balanced_random_forest_cw	0.756667	0.462108	0.611153	0.797988	0.526282	0.366788	0.764464	0.540884
Bayesian Hyperparameter Optimization	0.758667	0.466030	0.625471	0.796490	0.534106	0.375917	0.777963	0.541919

Schlussfolgerung:

Das Projekt im Bereich der binären unbalancierten Klassifikation hat die optimalsten Modelle identifiziert, die auf den folgenden Algorithmen und Verfahren basieren :

- **Undersampled Random Forest Classifier (Recall 0.62);**
- **Balanced Random Forest Classifier (Recall 0.62);**
- **Bayesian Hyperparameter Optimization (Recall 0.63).**

Eine genauere Sensitivität / Recall ist möglich, wenn bestimmte Anforderungen an die Datenqualität erfüllt sind.

Können Sie mit dem Ergebnis zufrieden sein?
Recall war 0,63 erreicht. Zweifellos ist es uns gelungen, den Entscheidungsbaum auf unbalancierten Daten zu trainieren.

Welche Lösung brachte den Erfolg?
Der Erfolg basiert auf neuen Konzepten und modernen Verfahren und Konzepten zur Gestaltung und Symbiose mit anderen Verfahren.

Welche Faktoren haben den Erfolg behindert?
Hauptproblem ist die Zeit und das Wissen. Man muss sehr hart und kontinuierlich arbeiten, um Ergebnisse zu erreichen.

Kann man dem Ergebnis trauen? (Kann es auf zukünftige Daten repliziert werden?)
Das Ergebnis ist vertrauenswürdig und kann angewendet werden

Welche andere Methode sollten Sie in Zukunft ausprobieren, wenn Sie mehr Zeit haben?
Adaptive Synthetic Sampling Approach for Imbalanced Learning wie Adasyn, Imbalanced Random Forrest, Smote etc.

5.2 Die Klassen, die sich gut für eine Wiederverwendbarkeit eignen

In unserem Projekt haben wird die benutzerdefinierte Funktion performance.py verwendet, die alle Metrik-Ergebnisse (Konfusionsmatrix), alle möglichen Kombinationen der vorhergesagten Werte im Vergleich im Gegensatz zum tatsächlichen Ziel, ausplotten. Die Funktion ziemlich standardisiert ist und Funktionen aus dem Metrik-Modul von scikit-learn verwendet.

Das zweite Diagramm enthält die Receiver Operating Characteristic (ROC)-Kurve. Die ROC-Kurve zeigt einen Kompromiss zwischen der wahr-positiven Rate und der falsch-positiven Rate für verschiedene Wahrscheinlichkeitsschwellenwerte. Ein Wahrscheinlichkeitsschwellenwert bestimmt die vorhergesagte Wahrscheinlichkeit, oberhalb derer wir entscheiden, dass die Beobachtung zur positiven Klasse gehört (standardmäßig beträgt sie 50 %). Der ideale Punkt ist (0, 1), und die Kurve eines geschickten Modells würde so nah wie möglich an diesem Punkt liegen. Bei einem Modell ohne Fähigkeiten liegt die Kurve dagegen nahe an der Diagonalen (45°).