Projekt WUM

Krzysztof Maślak

1 Przygotowanie danych

W kodzie zawarłem już tylko kwintesencję procesu przygotowywania danych. W przeciwnym razie notebook mógłby być bardzo długi. Wczytujemy zatem dane. Zamieniamy wartości -7, -8, -9 na Nany. Następnie usuwamy wiersze, które składają się tylko z wartości Nan, bowiem nie wnoszą one jakości predykcyjnej. Dalej zauważamy, że możemy ręcznie uzupełnić dużą część kolumny X9. W istocie, wiele tych Nan jest po prostu 0, bo patrząc na kolumnę X8, która mówi, że ktoś ma 100 procent braku opóźnień w płatnościach to w szczególności minęło 0 miesięcy od ostatniego opóźnienia w płatnościach. Z wykresu korelacji widzimy, że kolumna X16 to praktycznie to samo co X17. Usuwamy tę pierwszą, gdyż z opisu X17 wydaje się być rzetelniejsza. Dalej zajmujemy się znormalizowaniem rozkładów cech. Na tyle ile dało się to zrobić, zrobiłem to metodą prób i błędów. Zauważamy, że rozkład targetu y jest w miarę zrównoważony.

2 Eksperymenty

Będziemy chcieli optymalizować nasz model pod względem metryki balanced accuracy. Będziemy poszukiwać jak najlepszych parametrów modeli posiłkując się kroswalidacją, pipelinemi oraz funkcją GridSearchCV. Osobiście, bardzo lubię SVM. Jednakże oczywiście faworyzowanie go byłoby nie fair w stosunku do innych modeli. Postaram się eksperymenty przeprowadzać obiektywnie.

2.1 Regresja logistyczna z regularyzacją L2

Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę 100 różnych parametrów C rozmieszczonych równomiernie w przedziale [0.001, 10]; maksymalną liczbę iteracji: 150, 300. Oprócz tego w KNNImputer bierzemy pod uwagę liczby sąsiadów: 3, 5, 7.

Z pomocą 5-krotnej kroswalidacji wybieramy najlepsze parametry: C : 0.1, maxiter : 150, liczba sąsiadów : 5.

2.2 Gradient Boosting

Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę 30 różnych parametrów learning rate rozmieszczonych równomiernie w przedziale [0.005, 2]; liczbę estymatorów: 50, 100, 200; maksymalna

głębokość: 3, 5, 7, 9; maksymalna ilość cech: pierwiastek, logarytm o podstawie 2; parametr subsample: 0.8, 1.

Z pomocą 3-krotnej kroswalidacji wybieramy najlepsze parametry: learning rate: 0.74; liczbę estymatorów: 100; maksymalna głębokość: 3; maksymalna ilość cech: pierwiastek; parametr subsample: 1.

Bonusowo sprawdzamy jak sobie poradzi Gradient Boosting z domyślnymi parametrami. Nieznacznie gorzej.

2.3 Random Forest

Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę liczbę estymatorów: 50, 75, 100, 200; maksymalna głębokość: 3, 5, 7; minimalną ilość w liściu do podziału: 8, 12, 18; parametr classweight: balanced, None; parametr bootstrap: True, False.

Z pomocą 4-krotnej kroswalidacji wybieramy najlepsze parametry: liczbę estymatorów: 50; maksymalna głębokość: 5; minimalną ilość w liściu do podziału: 12; parametr classweight: None; parametr bootstrap: True.

2.4 Bagging KNN

Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę liczbę estymatorów: 5, 10, 20, 30, 50; ilość sąsiadów w każdym estymatorze: 3, 5, 7, 9; ilość sąsiadów w imputerze: 3, 5, 7, 9.

Z pomocą 5-krotnej kroswalidacji wybieramy najlepsze parametry: liczbę estymatorów: 30; ilość sąsiadów w każdym estymatorze: 9; ilość sąsiadów w imputerze: 5.

Bonusowo sprawdzamy jak sobie poradzi model taki jak wyszedł jako z najlepszymi parametrami, tylko zamienimy KNNImputer na SimpleImputer. Wyszło nieznacznie lepiej, więc to on nam w przyszłości posłuży jako głosujący w modelach Voting Classifier.

2.5 Extra Tree

Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę liczbę estymatorów: 50, 75, 100, 150; maksymalna głębokość: 3, 5, 7; minimalną ilość w liściu do podziału: 10, 15, 19; parametr bootstrap: True, False.

Z pomocą 4-krotnej kroswalidacji wybieramy najlepsze parametry: liczbę estymatorów: 150; maksymalna głębokość: 7; minimalną ilość w liściu do podziału: 15; parametr bootstrap: False.

2.6 SVM z Regresję logistyczną jako selekcja cech

Bardzo temu modelowi kibicuję. Muszę się pilnować, żeby zachować umiar w rozmiarze przestrzeni parametrów. Ale chyba nic się nie stanie, jeśli mu trochę pomogę dodając skromną selekcję cech za pomocą regresji logistycznej...

Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę parametr C w regresji: 0.1, 0.5, 1, maksymalna liczba

wybranych cech: 5, 8, 12, 18. Do tego parametr C w SVM o wartościach: 0.01, 0.1, 0.2, 0.5, 0.7, 1, 3, 6, 10, 100.

Z pomocą 3-krotnej kroswalidacji wybieramy najlepsze parametry: parametr C w regresji: 0.1; maksymalna liczba wybranych cech: 5. Do tego parametr C w SVM o wartości: 1.

2.7 Voting Classifier

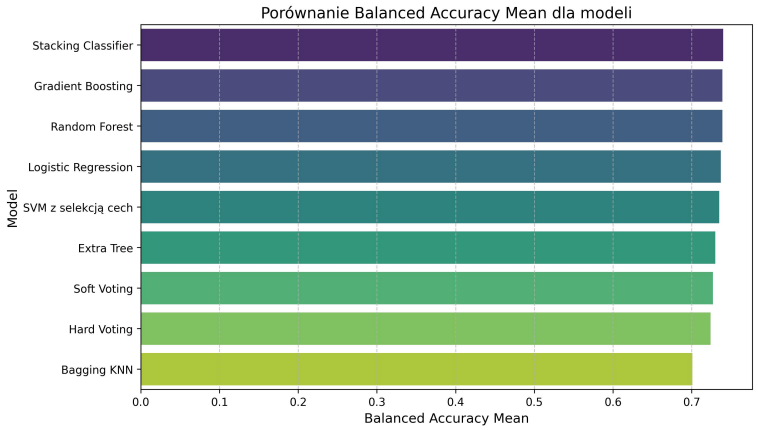
Stworzymy model składający się z trzech modeli, których parametry zostały wybrane przez wcześniejsze eksperymenty. Będą to Bagging 30 KNN, Extra Tree oraz SVM z selekcją cech za pomocą regresji logistycznej. Będąc dokładniejszym zbudujemy dwa medele głosujące. Jeden z nich będzie miał parametr voting ustawiony na soft, a drugi na hard.

2.8 Stacking Classifier

Stworzymy model StackingClassifier. Estymatory pomocnicze to będą regresja logistyczna oraz Random Forest ze znalezionymi wcześniej parametrami. Finalnym estymatorem będzie Gradient Boosting. Przeszukujemy przestrzeń parametrów w celu znalezienia jak najlepszego modelu. Bierzemy pod uwagę parametr learning rate: 0.005, 0.01, 0.1, 0.2, 0.5, 0.7, 1, 3, 10; maksymalną głębokość: 3, 5, 7; liczbę estymatorów: 50, 75, 100, 150; minimalną ilość w liściu do podziału: 8, 15, 20; parametr subsample: 0.8, 1.

Z pomocą 3-krotnej kroswalidacji wybieramy najlepsze parametry: learning rate: 0.01; maksymalną głębokość: 3; liczbę estymatorów: 50; minimalną ilość w liściu do podziału: 15; parametr subsample: 0.8.

3 Wyniki



Rysunek 1: Porównanie Balanced Accuracy Mean dla modeli.

Widzimy, że wybrane przez modele wykręcają stosunkowo zbliżone wyniki. Trochę gorzej wypada Bagging KNN, jednak jest on modelem w pewien sposób innym od pozostałych, więc postanowiłem mu tak czy siak dać szansę jako głosujący w Voting Classifierach. Poza tym, niezależnie czy mamy do czynienia z poczciwą regresją logistyczną, czy też złożonym Stacking Classifierem, to wyniki są podobne. Nadmieńmy jednak, że najlepiej sobie poradził właśnie wspomniany Stacking Classifier notując wynik 0.74 jako średnia balanced accuracy ze wszystkich foldów. Świetnie w rywalizacji odnalazł się również Gradient Boosting oraz Random Forest. Zauważmy, że w Stacking Classifierze estymatorem wieńczącym model był właśnie Gradient Boosting, co zdaje się go windować w notowaniach buchmakerów.

Model	**Balanced Accuracy Mean**	**Balanced Accuracy Std**
Logistic Regression	0.737	0.010
Gradient Boosting	0.739	0.008
Random Forest	0.739	0.014
Bagging KNN	0.701	0.013
Extra Tree	0.730	0.014
SVM z selekcją cech	0.735	0.008
Soft Voting	0.727	0.015
Hard Voting	0.724	0.015
Stacking Classifier	0.740	0.006

Tabela 1: Porównanie modeli pod względem średniej i odchylenia standardowego balanced accuracy ze wszystkich foldów

Druga tabelka pokazuje nam wyniki Gradient Boosting oraz Stacking Classifiera na 5 procentach zbioru testowego. Stacking Classifier jest odrobinę lepszy. Oczywiście chciałem jeszcze sprawdzić SVM, ale niestety pomyliłem się w aplikacji i przetestowałem drugi raz Gradient Boosting. Trochę to smutne, ale może tak miało być. Mimo tego, że obawiam się, że Stacking Classifier może się okazać zbyt skomplikowany i nie będzie się dobrze uogólniał na cały zbiór testowy, to wybieram właśnie jego. On po prostu sobie na to zasłużył. Dzięki ciężkiemu treningowi wykręcił rekordowe wyniki. Do tego jest ciekawszy.

Model	**Balanced Accuracy**
Gradient Boosting	0.7759
Stacking Classifier	0.7831

Tabela 2: Test Balanced Accuracy dla Gradient Boosting i Stacking Classifier