Projekt

Julia Girtler

Wstęp

Celem projektu było stworzenie modelu uczenia maszynowego, który uzyska najwyższą zbalansowaną dokładność.

Dane

Udostępnione dane miały 29 kolumn i odpowiednio 2000 wierszy - zbiór X oraz 600 wierszy - zbiór X testowy.

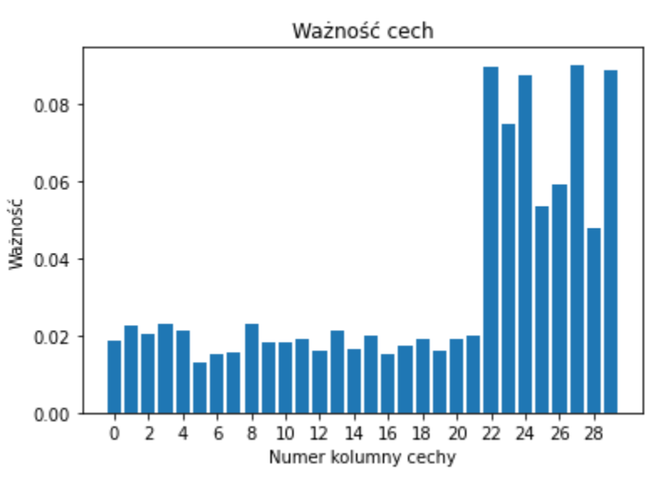
Każda kolumna podanego zbioru danych była kolumną numeryczną. Zbiór nie zawierał braków danych.

Outliery zostały wykrtyte za pomocą pętli przy wykorzystaniu IQR oraz zastępione średnią wartością kolumny.

Inżynieria cech

Na podstawie XGboostClassifier została zbadana ważność danej cechy w predykcji modelu. Wartości ważności sumują się do 1. Możemy wyróznić 6 znacząco ważnych kolumn, Na ich podstawie zostały stworozne nowe cechy takie jak suma oraz produkt wartości.

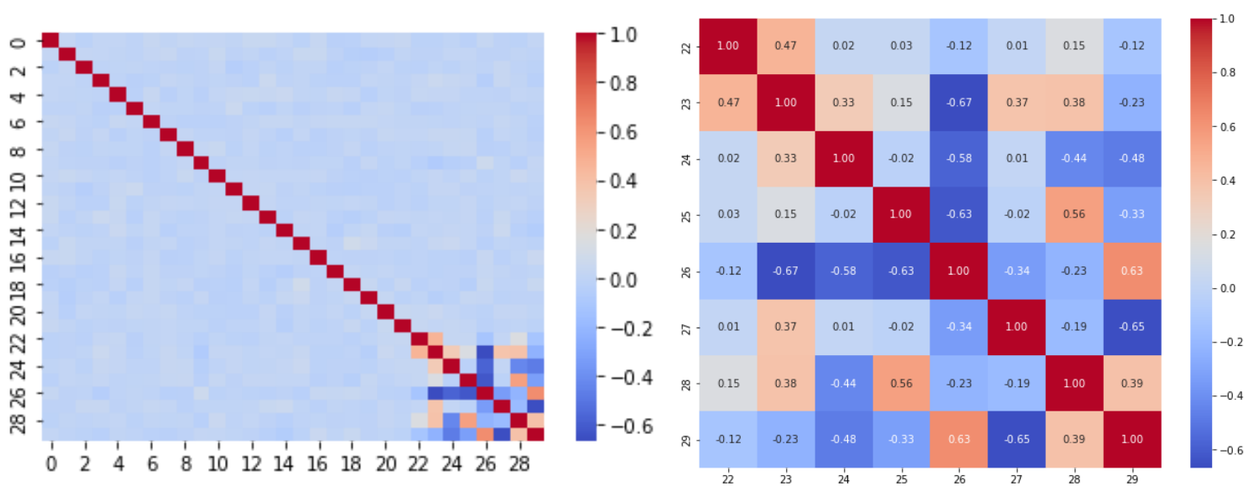
XGBoost to implementacja algorytmu gradient boosting, który jest znany ze swojej skuteczności w zadaniach klasyfikacji i regresji. Algorytm ten buduje sekwencję słabych modeli, a każdy nowy model jest dostosowany w celu poprawienia błędów poprzednich.



Została również zbadana korelacja i ukazana na mapie ciepła. Możemy zauważyć, że kolumny od 0 do 22 nie są ze sobą skorelowane, natomiast w dalszych kolumnach korelacje występują.

Dane zostały również przeskalowane za pomocą StandardScalera.

Na podstawie tych map PCA zostało przeprowadzone tylko na poszczegołnych kolumnach (22:29), ponieważ reszta kolumn nie była skorelowana.



Podział danych

Dane (sam X bez zbioru X testowego) zostały podzielone na zbiór treningowy i walidacyjny za pomocą train_test_split o parametrze test_size = 0.2.

Zostało to wykonane w celu możliwości kontroli predykcji na zbiorze walidacyjnym przy budowie naszego modelu.

Przeprowadzone eksperymenty

Przetestowane zostały modele takie jak: DecisionTreeClassifier, LogisticRegression, KNeighborsClassifier, RandomForestClassifier, BaggingClassifier, SVC, XGBoost, VotingClassifier. Dla każdego z algorytmów została stowrzona przestrzeń hiperparametrów, a następnie przeprowadzony gridsearch z potrójną kroswalidacją.

W poniższej tabeli zostały przedstawione zbalansowanę dokładności dla każdego z testowanych modeli.

Decision Tree Classifier	Logistic Regression	K Neighbors Classifier	Random Forest Classifier	Bagging CLassifier	SVC	XGBoost	Voting CLassifier
0.78	.6275	0.845	0.845	0.855	0.83	0.875	0.9025

Ostateczny model

Zastosowany został model klasyfikacji VotingClassifier.

Voting Classifier to metoda ensemble learning, która łączy predykcje wielu algorytmów uczenia maszynowego, aby dokonać ostatecznej prognozy. Ideą stojącą za Voting Classifierem jest wykorzystanie mocnych stron różnych indywidualnych modeli, aby stworzyć bardziej stabilny i dokładny model predykcyjny.

W projekcie wybrane zostały następujące algorytmy: RandomForestClassifier, XGboost, BaggingClassifier, SVC. Osobno zostało przeprowadzone 5 gridsearchy, w których parametry zostały dobrane tak, aby zmaksymalizować dokładność.

Zdecydowałam się na podane wyżej salgorytmy, ponieważ komitety w machine learning są skuteczne ze względu na zwiększenie stabilności, poprawę wydajności, redukcję nadmiernego dopasowania oraz uniwersalność. SVM są efektywne w wielowymiarowych przestrzeniach, obsługują złożone granice decyzyjne oraz stosują jądra dla danych nieliniowych.

Istnieją dwie główne odmiany Voting Classifiera: Hard Voting (głosowanie twarde) i Soft Voting (głosowanie miękkie). Nasz model wykorzystuje opcję Soft Votingu. W głosowaniu miękkim zamiast liczenia większości klas, modele dostarczają prawdopodobieństwa dla każdej klasy,które zostają uśredniane. Klasa z najwyższym średnim prawdopodobieństwem jest wybierana jako przewidziana.

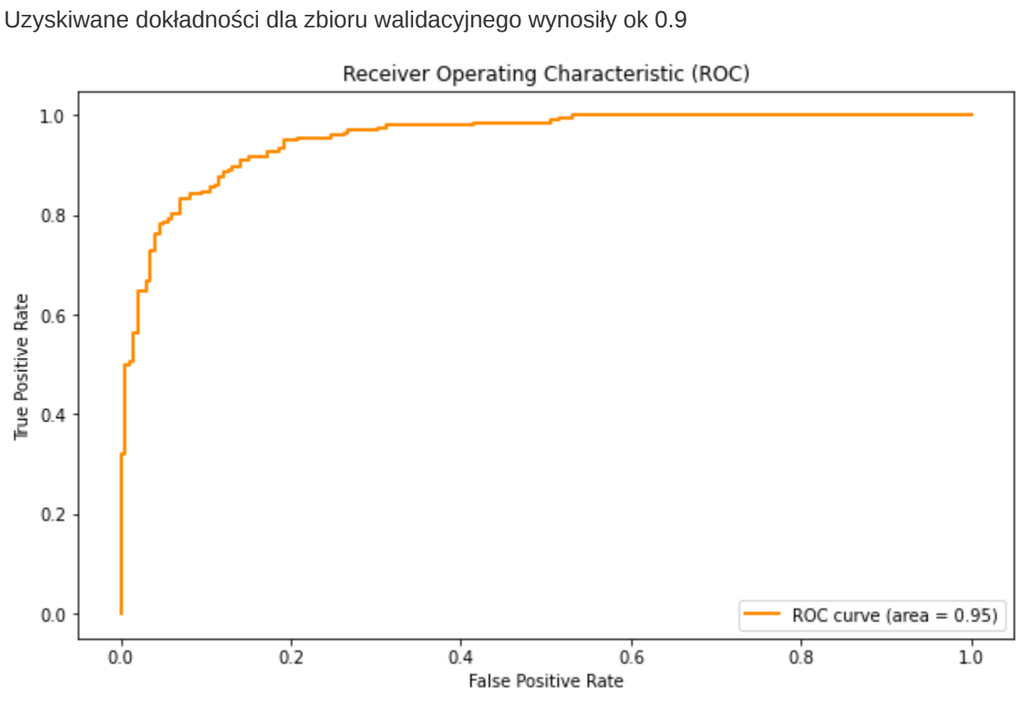
Na koniec został przeprowadzony gridsearch dla całego VotingClassifier dla parametru weigths.

Ostateczne hiperparametry naszego modelu:

- RandomForestClassifier(max_depth=28, min_samples_leaf=1, min_samples_split=5, n_estimators=150)
- XGBClassifier(max_depth=20, learning_rate=0.05, n_estimators=200, min_child_weight=1, subsample=1, colsample_bytree= 0.8)
- BaggingClassifier(n_estimators = 200,random_state = 0,max_features = 0.5)
- SVC(probability = True,C=1.25)
- weights = [3,2,3,1]

Testowanie przygotowanego modelu na zbiorze uczącym i walidacyjnym

Uzyskiwane dokładności dla zbioru walidacyjnego wynosiły ok 0.9



Wnioski

- przed rozpoczęciem budowy modelu należało dokładnie przyjrzeć się danym, ponieważ brak preprocesssingu wpływał na wartości dokładności, w szczególności duża zmiana zachodziła gdy outliery nie były zastępowane średnimi z kolumn
- z przetestowanych modeli najgorsza okazała się regresja logistyczna, natomiast najlepszy VotingClassifier
- Koniecznym również okazało się dostosowywanie parametrów, w szczególności RandomForestClassifier, gdzie drzewa musiały zostać odpowiednio przycięte aby uniknąć overfittingu
- Dla naszego problemu mało skutecznym okazał się dobór parametru voting = ‘hard’. Działo się tak, ponieważ głosowanie twardo traktuje każdy model jako równie ważny, niezależnie od pewności, z jaką dokonuje predykcji.