AUTO ML

Praca domowa nr 2

Weronika Dyszkiewicz, Katarzyna Mamla

16.01.2024

1 Wstęp

Celem projektu jest zaproponowanie metody klasyfikacji, która pozwoli zbudować model o jak największej mocy predykcyjnej. Dysponujemy sztucznie wygenerowanym zbiorem danych *artificial*, w którym zostały ukryte istotne zmienne. Należy dokonać klasyfikacji do dwóch klas, {-1,1}. Dokładność modelu będzie mierzona za pomocą miary zrównoważonej dokładności **balanced accuracy**. Dane do projektu to sztucznie wygenerowany zbiór, który zawiera 500 zmiennych objaśniających, ale część z tych kolumn jest zbędna. Zbiór treningowy zawiera 2000 obserwacji, natomiast zbiór testowy 600.

1.1 Preprocessing

W projekcie skorzystałyśmy z dwóch metod selekcji zmiennych:

- `PCA` ,
- `SelectFromModel` poprzez `LinearSVC` .

Wymiarowość została redukowana przy użyciu `PCA` z tunowanym, przy użyciu `RandomizedSearchCV` , argumentem liczby komponentów do zachowania.
Implementacja `LinearSVC` wykorzystuje generator liczb losowych do wyboru cech podczas dopasowywania modelu. Ten klasyfikator wraz z użyciem `SelectFromModel` pozwala wybrać niezerowe współczynniki, aby zmniejszyć wymiarowość danych.
Przed tworzeniem modeli, przy pomocy `LabelEncoder` , przetransformowałyśmy klasy zmiennej objaśnianej z {-1,1} na {0,1}.
W każdym modelu zastosowałyśmy także skalowanie zmiennych korzystając z jednej z metod: `StandardScaler(), RobustScaler(), Normalizer(), MaxAbsScaler()` , którą wybrał nam `RandomizedSearchCV` .

2 Modele wykonane ręcznie

Każdą z dwóch metod wyboru zmiennych testowałyśmy na modelach

1. `RandomForestClassifier` ,
2. `KNeighborsClassifier` ,
3. `XGBClassifier` .

Modele były ewaluowane miarą balanced-accuracy. Dla każdego z rozważanych modeli ustaliłyśmy siatki hiperparametrów dane w tabeli 1.

Hiperparametry dla wybranych modeli		
Model	**Hiperparametry**	**Wartości hiperparametrów**
`RandomForestClassifier`	`n_estimators`	[i for i in range(50, 500, 25)]
	max_features	['auto', 'sqrt', 'log2']
	max_depth	[1,5,10,25]
	criterion	['gini', 'entropy']
`KNeighborsClassifier`	n_neighbors	[1, 3, 5, 7, 10]
	p	[1, 2]
	leaf_size	[1, 5, 10, 15]
	weights	['uniform','distance']
`XGBClassifier`	n_estimators	[100, 400, 800]
	max_depth	[3, 6, 9]
	learning_rate	[0.05, 0.1, 0.20]
	min_child_weight	[1, 10, 100]

Tabela 1: Rozważane modele i ich siatki hiperparametrów.

2.1 SelectFromModel

Przed przystąpieniem do tworzenia modeli z użyciem `SelectFromModel` stworzyłyśmy testowy klasyfikator na wszystkich 500 zmiennych. Biorąc pierwszy z naszej listy modeli tj. `RandomForestClassifier` uzyskałyśmy miarę balanced-accuracy równą 0.69403. Wynik ten potwierdził konieczność selekcji zmiennych. Dla pokazanej w Tabeli 2 ilości istotnych zmiennych uzyskałyśmy konfigurację optymalnych hiperparametrów i miary balanced-accuracy dane w Tabeli 3.

Liczba istotnych zmiennych dla modeli	
Model	**Liczba istotnych zmiennych**
`RandomForestClassifier`	156
`KNeighborsClassifier`	159
`XGBClassifier`	166

Tabela 2: Liczba istotnych zmiennych objaśniających uzyskanych przy użyciu `SelectFromModel` .

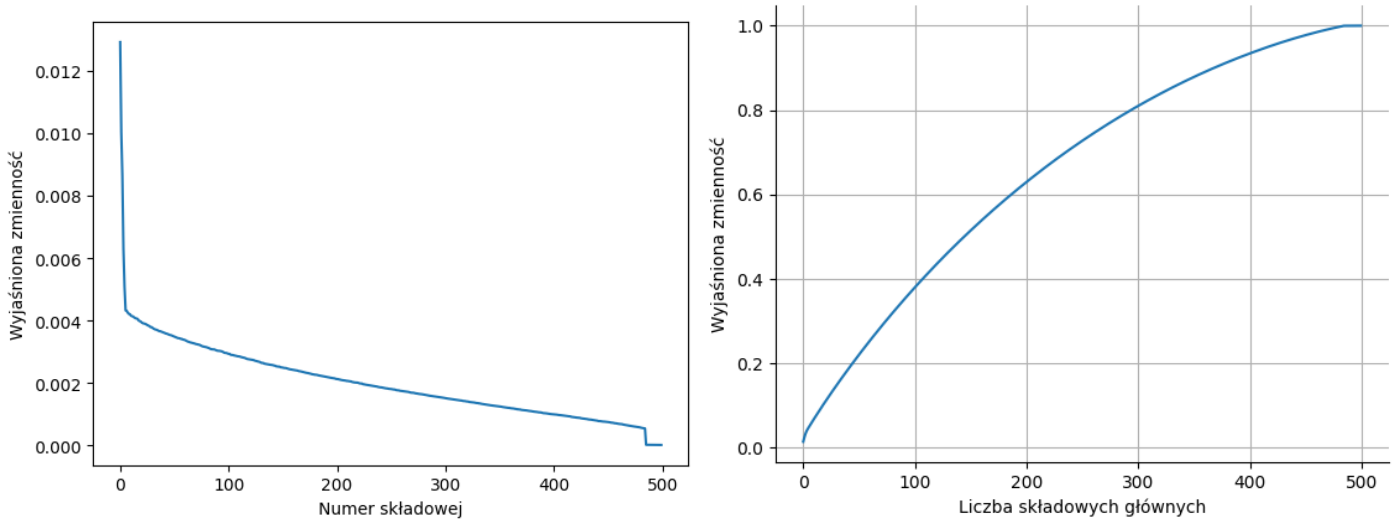
Wartości hiperparametrów przy `SelectFromModel`			
Model	**Hiperparametry**	**Wartości**	**Score**
`RandomForestClassifier`	`n_estimators`	175	0.74051
	max_features	sqrt	
	max_depth	10	
	criterion	entropy	
`KNeighborsClassifier`	n_neighbors	1	0.70715
	p	2	
	leaf_size	5	
	weights	distance	
`XGBClassifier`	n_estimators	400	0.78951
	max_depth	9	
	learning_rate	0.2	
	min_child_weight	1	

Tabela 3: Wartości hiperparametrów i wyniki dla najlepszych modeli przy wyborze zmiennych używając `SelectFromModel` .

Jak widać najlepszy score uzyskał `XGBClassifier` , a liczba zmiennych we wszystkich modelach wahała się między $150 - 170$.

2.2 PCA

Na początku zbadałyśmy poziom wyjaśnionej wariancji przez każdą z 500 zmiennych objaśniających w naszych danych. Jak widać na wykresach na Rysunku 1 mamy bardzo dużo zmiennych, które nie wnoszą wiele informacji (mają poziom wyjaśnionej zmienności > 0.003). Dlatego zawęziłyśmy liczbę możliwych składowych głównych do maksymalnie 18, czyli liczby składowych które wyjaśniają więcej niż 4% wariancji danych. W rzeczywistości, przeszukując siatki parametrów w każdym z 3 rozważanych klasyfikatorów, uzyskałyśmy tylko 5 głównych składowych bez względu na model. Taka liczba parametru `n_components` wydaje się optymalnym wyborem patrząc na lewy wykres na Rysunku 1, gdzie to właśnie 5 jest momentem wygięcia wykresu. Zgodnie z Tabelą 4 najlepszym modelem okazała się `KNeighborsClassifier` i to właśnie on został wybrany jako ostateczny, ręcznie wykonany model w projekcie.



Rysunek 1: Lewy wykres: Procent wyjaśnionej wariancji przez każdą z 500 zmiennych. Prawy wykres: skumulowany: procent wyjaśnionej wariancji przez zmienne.

Wartości hiperparametrów przy `PCA`			
Model	**Hiperparametry**	**Wartości**	**Score**
`RandomForestClassifier`	`n_estimators`	325	0.86392
	max_features	sqrt	
	max_depth	25	
	criterion	entropy	
`KNeighborsClassifier`	n_neighbors	7	**0.88029**
	p	2	
	leaf_size	10	
	weights	uniform	
`XGBClassifier`	n_estimators	800	0.86479
	max_depth	9	
	learning_rate	0.05	
	min_child_weight	1	

Tabela 4: Wartości hiperparametrów i wyniki dla najlepszych modeli przy wyborze zmiennych używając `PCA` .

3 Model AutoML

W projekcie został wykorzystany framework AutoMlowy *FLAML*. Wybrałyśmy akurat ten, ponieważ w nim łatwo odnaleźć wybrane hiperparametry oraz jest dosyć szybki. Pierwszy model tworzymy przy pomocy funkcji `AutoML()` , a następnie fitujemy z argumentem `task="classification"` oraz `time budget=200` . Najlepsza dokładność na walidacyjnym zbiorze puszczonym na 200 sekund wynosi 0.9489. Co ciekawe na podobnym modelu, jedynie ze zmianą na argument `time budget=1800` , najlepsza dokładność wynosi 0.9282. Zatem model puszczony na 200 sekund miał u nas lepszą dokładność niż puszczony na pół godziny. Być może dlatego, że najlepsze parametry wyszły na początku trenowania. Testowałyśmy te modele również z argumentem `ensemble=True` jednak tu wychodziły gorsze wyniki.

Wybrane wartości hiperparametrów przy `time budget=200`			
Model	**Hiperparametry**	**Wartości**	**Score**
`LGBM`	`n_estimators`	67	0.9489
	num_leaves	103	
	min_child_samples	10	
	learning_rate	0.016254	

Tabela 5: Wartości hiperparametrów i wyniki dla najlepszych modeli przy wyborze zmiennych używając funkcji `AutoML()` przy 200 sekundach.

Wybrane wartości hiperparametrów przy `time budget=1800`			
Model	**Hiperparametry**	**Wartości**	**Score**
`extra_tree`	`n_estimators`	503	0.9282
	max_features	1	
	max_leaves	598	
	criterion	entropy	

Tabela 6: Wartości hiperparametrów i wyniki dla najlepszych modeli przy wyborze zmiennych używając funkcji `AutoML()` przy 1800 sekundach.

4 Podsumowanie

Porównujący wyniki uzyskane na trzech różnych klasyfikatorach i dwóch metodach selekcji zmiennych jako najlepszy model wykonany ręcznie został wybrany `KNeighborsClassifier` , gdzie selekcja odbyła się za pomocą `PCA` . Tunowanie hiperparametrów tej metody wyboru zmiennych przez `RandomizedSearchCV` wskazało na użycie 5 głównych składowych. Podczas sprawdzenia w aplikacji model ten osiągnął wartość balanced-accuracy na poziomie 0.9333 na pięcioprocentowej próbce testowej. Model uzyskany za pomocą `AutoML` również osiągnął balanced-accuracy na poziomie 0.9333 na tej samej pięcioprocentowej próbce testowej. Warto zauważyć, że próbka testowa była stosunkowo mała, co może wpływać na stabilność wyników. Równe wyniki obu modeli mogą sugerować, że ręcznie przygotowany model może być przetrenowany na dostępnych danych treningowych. Model uzyskany z wykorzystaniem `AutoML` może lepiej radzić sobie z dostosowaniem się do różnorodności danych.