Rafał Wójcik, 277366

Finalna dokumentacja projektu nr 6

Regresyjny las losowy

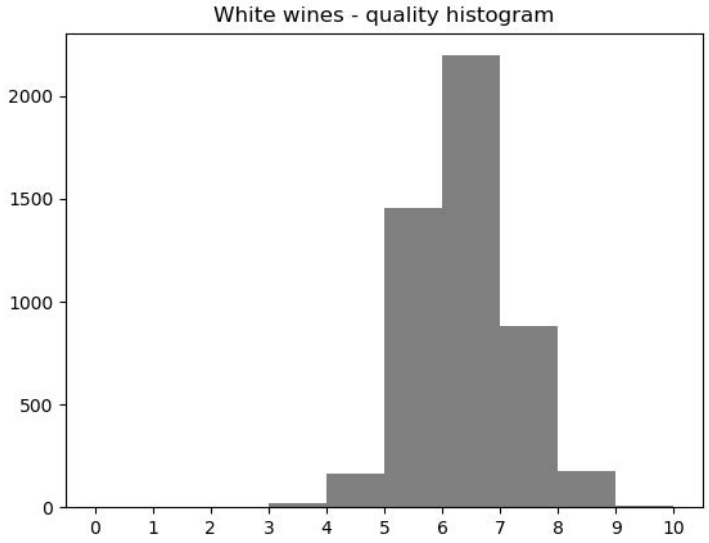
1. Treść projektu

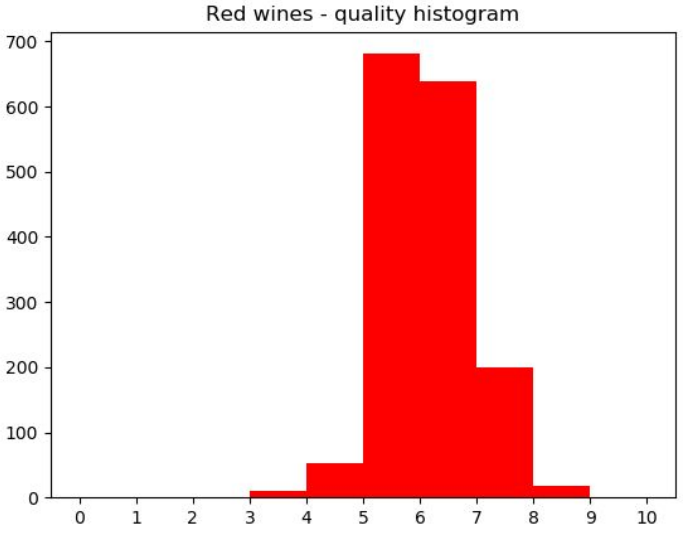
Zaimplementować zmodyfikowaną wersję algorytmu generowania lasu losowego regresji, w której do generowania kolejnych drzew losowane są częściej elementy ze zbioru uczącego, na których dotychczasowy model popełniał większe błędy. W eksperymentach należy wykorzystać zadanie Wine Quality.

2. Wstępna analiza danych

Zadanie zawiera dwa zbiory danych wiążące chemiczne właściwości win z ich jakością dla win czerwonych oraz białych. Wina posiadają 11 cech oraz przyporządkowaną im klasę oznaczającą jakość w skali od 0 do 10.

W pierwszym kroku przeanalizujmy rozkład win pod względem grup jakości:

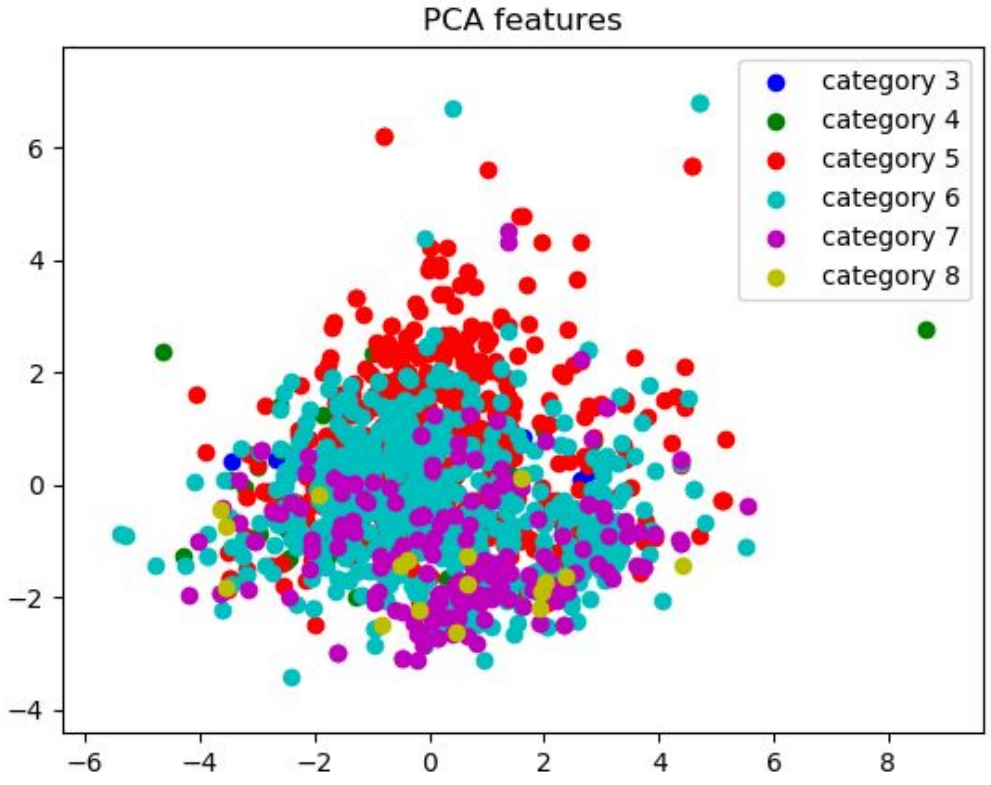


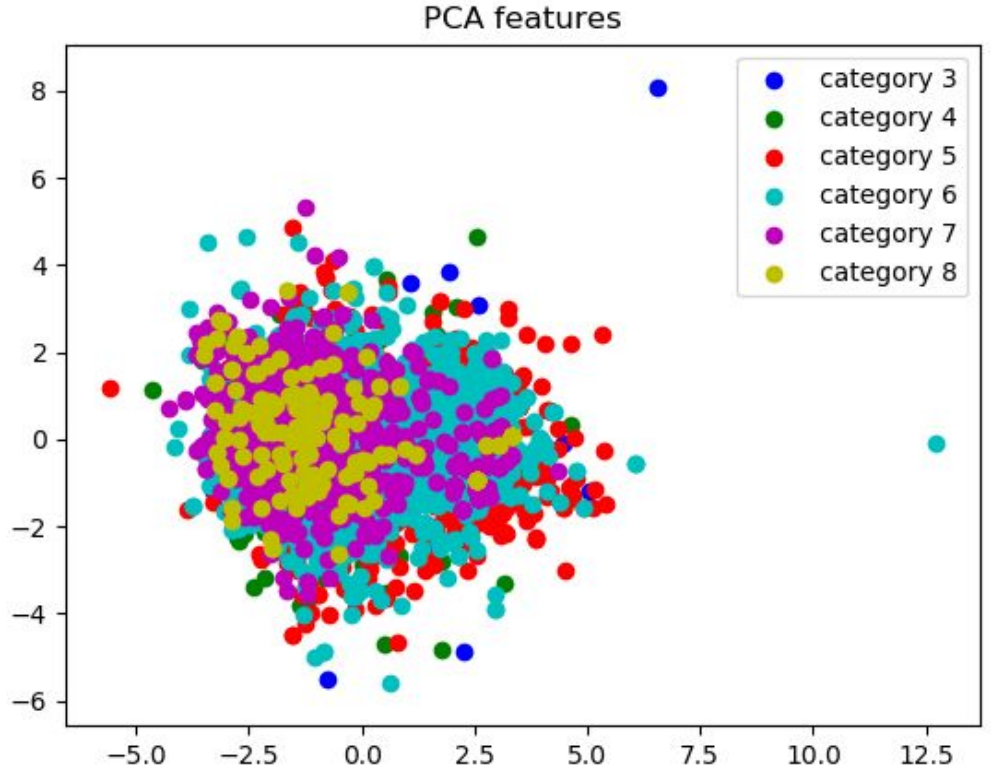


Jak widać zgodnie z opisem zawartym na stronie udostępniającej dane - jest znacznie mniej win wybitnych oraz słabych, a przeważają te znajdujące się w klasach 5-7.

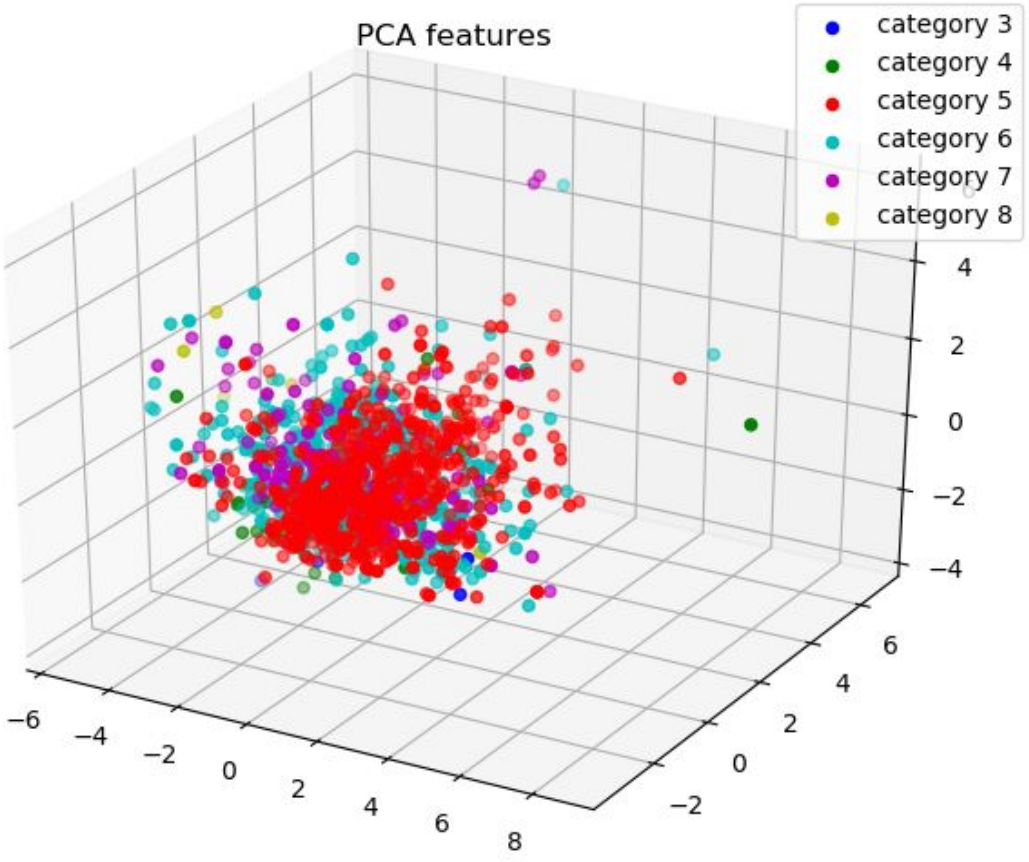
Na razie rozważać będziemy zbiory danych oddzielnie. Aby lepiej zrozumieć dane spróbujmy narysować je na wykresie. W celu zredukowania liczby cech skorzystamy

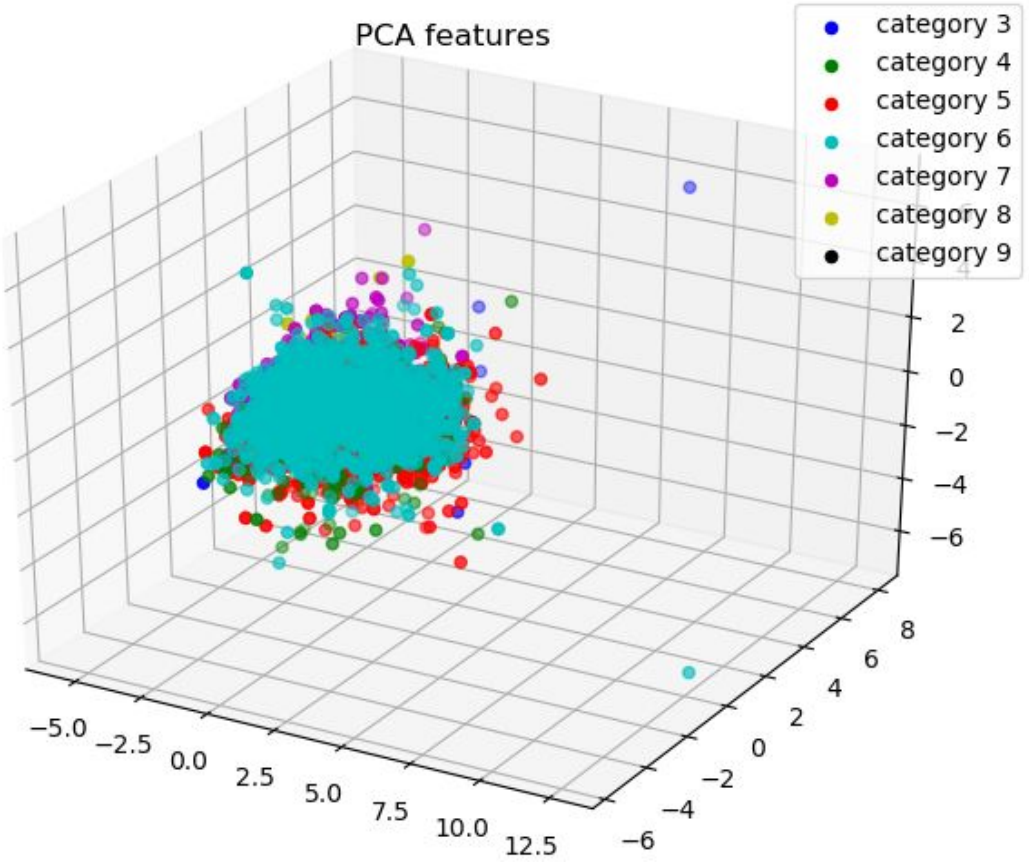
wstępnie z algorytmu PCA, który próbuje znaleźć taką podprzestrzeń rozciągniętą przez wektory będące kombinacją liniową pierwotnych zmiennych, aby odległości w rzucie ortogonalnym próbek na tę podprzestrzeń były jak najmniejsze. Zatem stara się znaleźć liniowe zależności między zmiennymi. Po sporządzeniu wykresów dla 2 cech o największej wariancji otrzymujemy (odpowiednio dla win czerwonych i białych):



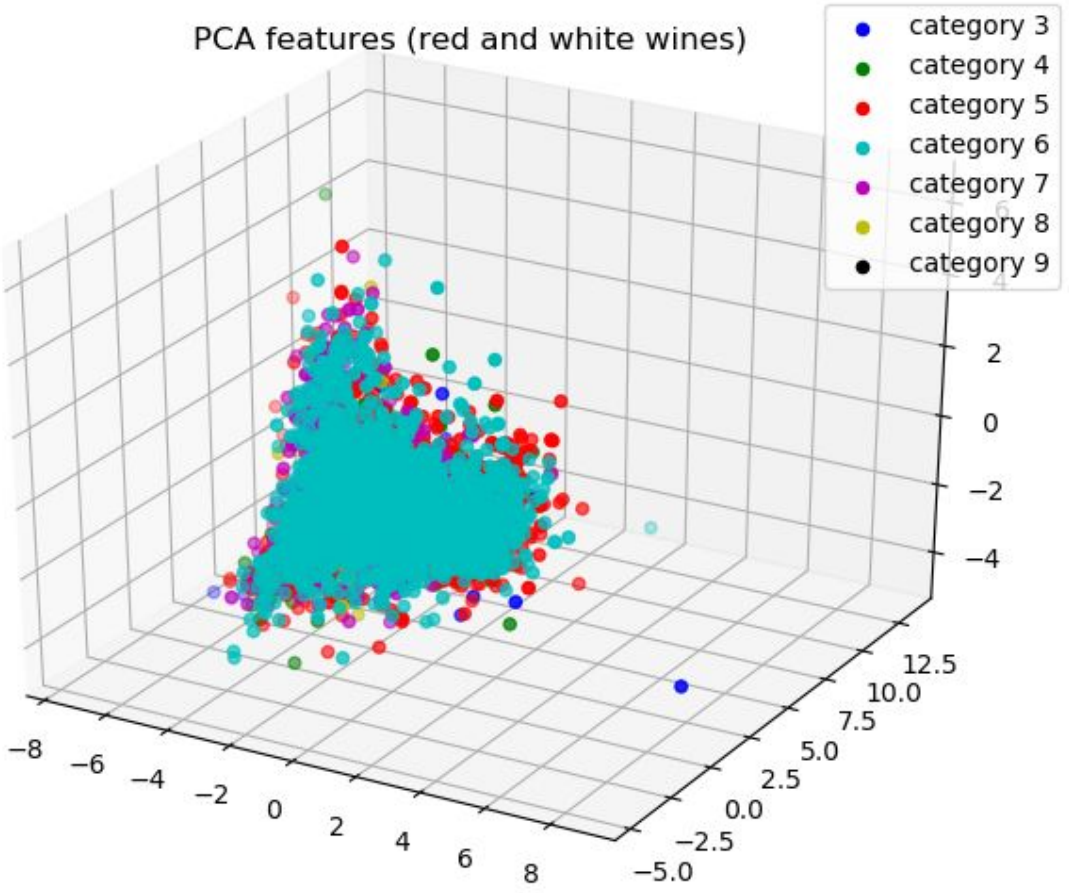


Natomiast dla wykresów zawierających o jedną cechę więcej, otrzymujemy następujące wykresy w przestrzeni trójwymiarowej:





Przeanalizujmy jeszcze wykres dla połączonych danych (wina czerwone + białe):



Tak więc okazuje się, że trudno dostrzec w danych wzorce, badając je jedynie pod kątem maksymalnie 3 cech otrzymanych z algorytmu PCA.

Powyższe wykresy wskazują jednak na pewne próbki wyraźnie odstające od innych. Mimo tego trudno dostrzec w nich jakieś powiązania klasowe. Z kolei po przyjrzeniu się danym z plików rzeczywiście niektóre próbki w ramach pewnych cech posiadają wartości lekko odstające (czasem nawet w wielu kolumnach jednocześnie).

Zbadajmy tym razem dane ściśle pod kątem znaczenia cech próbek. Sprawdźmy, czy wszystkie w równie istotny sposób wpływają na wynik. W tym celu użyjemy biblioteki sklearn i algorytmu opartego o klasyfikator Extra Trees (link do dokumentacji: http://scikit-learn.org/stable/modules/feature_selection.html#tree-based-feature-selection).

Po zastosowaniu algorytmu otrzymujemy procentowe wartości wpływu cechy na wynik:
0.07822734964819192, 0.09932466302436282, 0.08116938577860763,
0.08366978435806197, 0.08951895619957766, 0.08669808642672229,
0.08557114933894594, 0.08698026803106719, 0.07983210270460799,
0.08665724351539048, 0.14235101097446407.
Zatem na pierwszy rzut oka dane nie zawierają żadnych wyróżniających się cech.

3. Implementacja

```
n_trees        <- liczba drzew składających się na model
n_samples      <- liczba danych treningowych wybieranych dla każdego drzewa
n_features     <- liczba cech losowanych dla każdego z drzew
```

1. i := 0, M := empty set // M - zbiór drzew po i iteracjach
2. jeśli i >= n_trees: przejdź do kroku 9.
3. jeśli i = 0: S := zbiór n_samples losowo wybranych próbek, w przeciwnym wypadku: S := zbiór n_samples próbek, dla których błąd dla M był największy
4. wylosuj n_features cech
5. stwórz drzewo regresji D na podstawie danych składających się z próbek z pkt 3. z wyselekcjonowanymi cechami z pkt 4.
6. do zbioru drzew M dołącz drzewo D
7. i := i+1
8. przejdź do kroku 2.
9. zwróć model składający się z drzew z M

Dla zbioru drzew M odpowiedź dla danej próbki będzie liczona jako średnia po odpowiedziach dla każdego drzewa ze zbioru.

4. Struktura projektu

W katalogu src znajdują się pliki źródłowe projektu podzielone na katalogi: *random_forest* oraz *utils* zawierające odpowiednio implementację algorytmu drzewa losowego i narzędzi wykorzystywanych do parsowania oraz modyfikacji danych. Ponadto w tym samym folderze znajdują się dwa notatniki *jupyter* służące do testów. Notatnik *data_analyzer* zawiera podstawowe skrypty służące do analizy danych, natomiast notatnik *random_forest_test* sprawdza skuteczność zaimplementowanego algorytmu lasu losowego.

5. Dokładniejszy opis implementacji

5.1. Struktura klas

Algorytm posługuje się trzema klasami:

- *RandomForest* - klasa implementująca las losowy, zawiera w sobie instancję klasy *ClassifierSet,*
- *ClassifierSet* - klasa, która zawiera zbiór klasyfikatorów; implementuje metodę *predict,* która zwraca średnią z rezultatów wywołań metod *predict* skladowych klasyfikatorów,
- *PartialModel* - klasa opakowująca model, która oprócz modelu zawiera zbiór cech (indeksy), na podstawie których model będzie uczony oraz na podstawie których będzie dokonywał predykcji.

5.2. Las losowy

Za model drzewa wykorzystywanym w lesie losowym służy model z biblioteki sklearn *DecisionTreeRegressor* (http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html).

5.3. Sposób pomiaru efektywności modelu

Do pomiaru jakości modelu wykorzystywana jest metoda błędu średniokwadratowego.

6. Opis stosowanej procedury eksperymentalnej i danych używanych do eksperymentów

W doświadczeniach wykorzystywane są zbiory danych z winami czerwonymi oraz białymi - rozważane łącznie oraz oddzielnie. W związku z wynikiem wstępnej analizy, z której wynikało, że zbiór danych nie zawiera znaczących “outlierów” czy cech, które byłyby redundantne, dane nie były w żaden sposób dalej modyfikowane.

Przeprowadzane eksperymenty polegają na szukaniu efektywnych parametrów dla algorytmu lasu losowego oraz sprawdzeniu efektywności znalezionych modeli (z wykorzystaniem MSE). Niestety z powodu braku regularności w zbiorze danych, które dałoby się przedstawić w 3 wymiarach (co poświadczył we wstępnej analizie algorytmy PCA oraz ExtraTrees), trudno o przedstawienie sensownych wykresów z badań.

7. Wyniki/Wnioski

W trakcie uczenia dostępne są 3 parametry strojenia: n_trees, samples oraz n_features (częściowo opisane w pkt. 3). Ze względów praktycznych początkowo przyjęte n_samples oznaczające liczbę próbek zostało wymienione na parametr samples, który jest liczbą rzeczywistą z przedziału [0; 1] oznaczająca, jaka część próbek z całego zbioru danych ma zostać wykorzystana do nauki pojedynczego drzewa.

1. Rezultaty dla połączonych zbiorów win białych i czerwonych

Algorytm wykorzystujący jedynie jedno drzewo (dla n_trees = 1) zgodnie z oczekiwaniami dostarcza model, który ma tendencje do przeuczenia, ale w ogólności jego efektywność bywa bardzo niestabilna (czasami wypada lepiej, czasami gorzej). Ilościowo jakość takiego modelu można umieścić w granicach 0.7-0.8 w funkcji MSE.

Przy testach na większej ilości drzew wykorzystywana jest klasa *GridSearchCV* z biblioteki sklearn (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.htm), która daje możliwość wygodnego przeszukiwania przestrzeni parametrów w celu optymalizacji modelu. Klasa ta ocenia wewnętrznie modele na podstawie algorytmu kroswalidacji oraz podanej jako argument funkcji błędu (w tym przypadku MSE). Zwraca między innymi parametry, dla których znaleziono najlepszy model czy też jego błąd.
Klasa *GridSearchCV* wymaga przez model implementacji interfejsu *BaseEstimator* (sam w sobie zawiera przydatne dla klasy *GridSearchCV* metody).

W przypadku doświadczeń na lasie losowym składającym się z większej ilości drzew (eksperymenty były przeprowadzane do maksymalnie 30 drzew) rzeczywiście sprawdziło się

przypuszczenie, że model złożony z wielu drzew jest bardziej stabilny, a także ma mniejszą tendencję do przeuczenia. Przy 15-20 drzewach błędy maleją nieznacznie, są na poziomie 0.65 - jednocześnie przy parametrach samples = 0.8 oraz n_features = 7.

2. Rezultaty dla zbiorów win białych i czerwonych oddzielnie

Modele uczone na zbiorach win białych i czerwonych oddzielnie okazują się być nieco lepsze od modeli uczonych na połączonych danych. Dla parametrów n_trees = 20, samples = 0.8 oraz n_featuers = 7 udało się osiągnąć wynik 0.54 (w obydwu przypadkach) - co daje poprawę o około ⅛ poprzednich wyników modeli.