Machine-Learning-Feature-Selection

Project wykonany na przedmiot Zaawansowane Metody Uczenia Maszynowego na wydziale MINI PW

Cel projektu

Nalezy zaproponowac metody selekcji zmiennych oraz klasyfikacji które umozliwiaja zbudowanie modelu o duzej mocy predykcyjnej przy uzyciu mozliwie małej liczby zmiennych.

Zbiór danych

Zbiór *artificial* to sztuczny zbiór w którym sa ukryte istotne zmienne (pliki: *artificial train.data, artificial train.labels, artificial valid.data*).

Dane	Liczba zmiennych	Liczba obserwacji (treningowy)	Liczba obserwacji (walidacyjny)
artificial	500	2000	600

Dane potrzebne do wykonania projektu znajduja sie na stronie https://home.ipipan.waw.pl/p.teisseyre/TEACHING/ZMUM/index.html.

Testowane metody selekcji cech

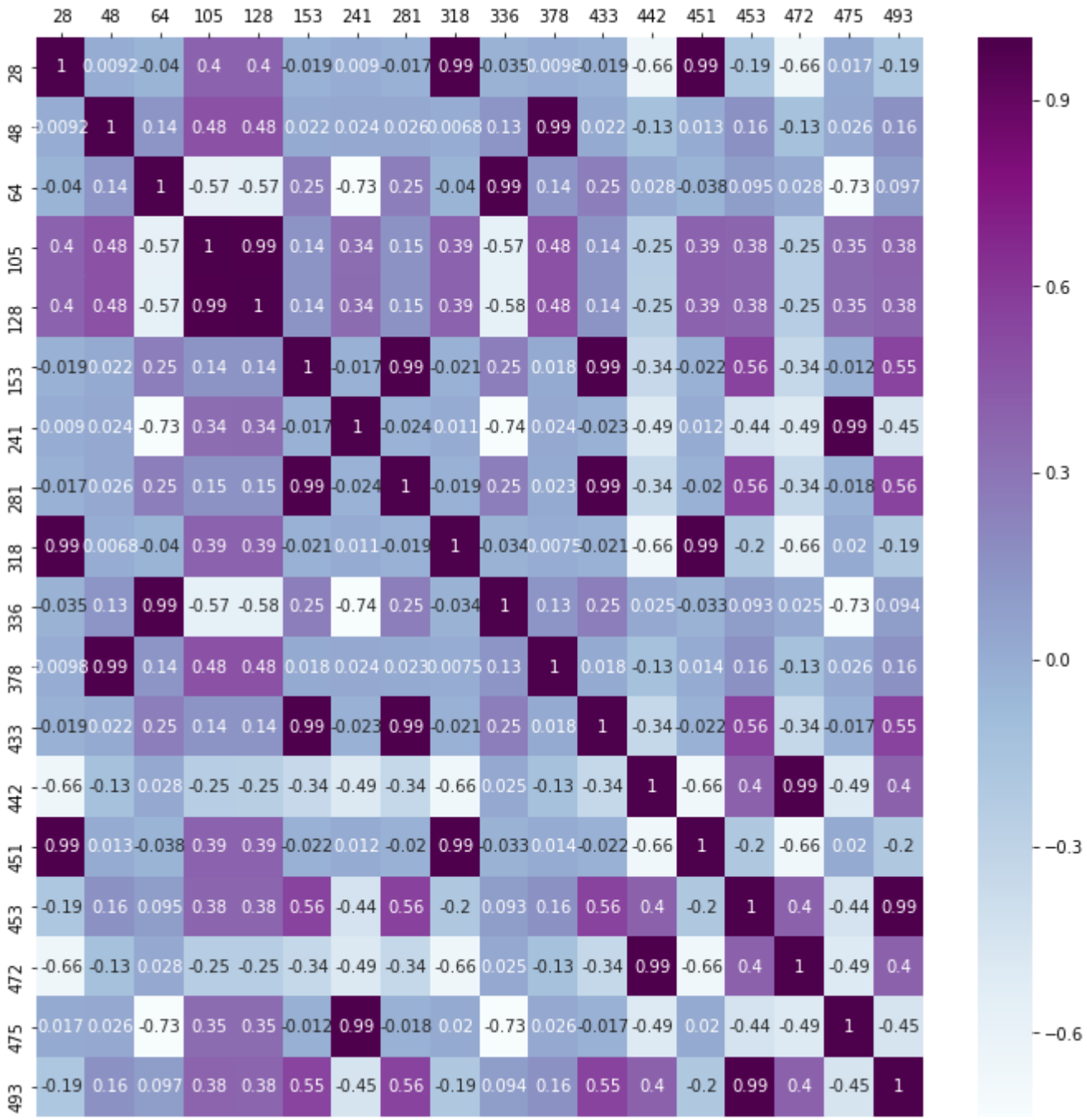
- R^2 denoisser
- Boruta

Testowane modele

- Linear Regression
- K-Neighbours Classifier
- Decision Tree Classifier
- Random Forest Classifier

Korelacje danych

Najbardziej skorelowane kolumny:



Widać że istnieją wysoko skorelowane cechy - selekcja jest niezbędna, aby uzyskać zoptymalizowane pod kątem uczenia maszynowego dane.

Metody selekcji zmiennych

Metoda R^2 Score Denoisser

Algorytm polega na wytrenowaniu modelu dla każdej zmiennej z osobna. Następnie wybranu tych zmiennych dla których funkcja `score()` odpowiadającego im modelu zwraca wartość metryki $R^2 > 0$.

Wewnątrz algorytmu umożliwiłem wybór jednego z dwóch modeli regresyjnych do wyliczenia wartości R^2.

Cechy jakie zostały wybrane przez te modele:

DecisionTreeRegressor

```
R² Score Denoisser
Regressor: DecisionTreeRegressor
Number of Selected Features: 18
Selected Feaures:
28 48 64 105 128 153 241 281 318 336 378 433 442 451 453 472 475 493
```

KNeighborsRegressor

```
R² Score Denoisser
Regressor: KNeighborsRegressor
Number of Selected Features: 21
Selected Feaures:
4 59 64 105 128 153 156 241 280 281 295 336 338 392 433 442 453 455 472 475 493
```

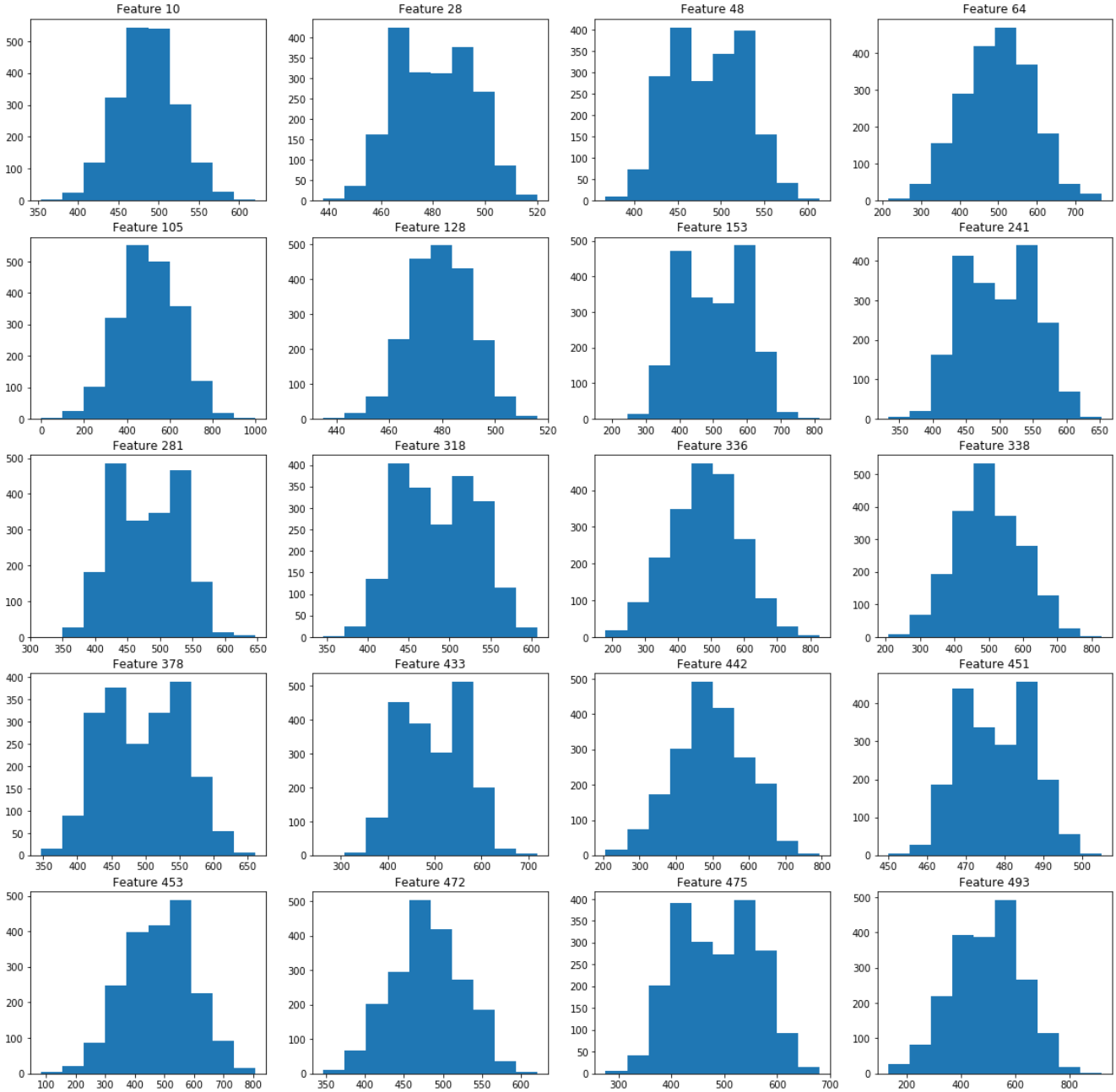
Metoda Boruta

"Matoda Boruta, polega na dodawaniu dodatkowych, mało istotnych cech (tzw. cieni) do istniejącego zbioru cech w danym systemie informacyjnym. Następnie są one wykorzystywane jako odniesienie dla oceny istotności oryginalnych atrybutów w kontekście pełnej struktury analizowanych danych, przy czym atrybuty są oceniane w oparciu o miarę przydatności obliczoną na podstawie lasów losowych wuczanych podczas treningu na danych. Na podstawie tej miary metoda Boruta prowadzi selekcję iteracyjnie, usuwając z systemu informacyjnego stopniowo cechy uznane za nieistotne, przy czym dzięki dodanym cieniom i dobranej mierze przydatności atrybutów, selekcja jest bardziej stabilna i niepodatna na przeuczene czy też szumy w danych." ~ Stabilne i wydajne metody selekcji cech z wykorzystaniem systemów uczących się - Miron Kursa

Cechy jakie zwrócił algorytm:

```
Boruta
Confirmed:  19
Tentative:  1
Number of Selected Features: 20
Selected Features:
10 28 48 64 105 128 153 241 281 318 336 338 378 433 442 451 453 472 475 493
```

Poniżej rozkłady cech wybranych przez algorytm boruta.



Część z nich ma rozkład zbliżony do Gausowskiego, część wykazuje lekki charakter dwumodalny. Uznałem za konieczne użycie jedynie standaryzacji.

W ostatecznym rozrachunku to właśnie cechy wybranę przez Borutę posłużyły do wyterenowania modelu, dlatego zaprezentuję działania modeli klasyfikatorów właśnie na tych cechach.

Modele Klasyfikacyjne

Wszystkie testowane modele miały parametry dobrane na podstawie zachłannego poszukiwania w przestrzeni parametrów. Miarą oceny jest *accuracy* ponieważ klasy są równomiernie reprezentowane.

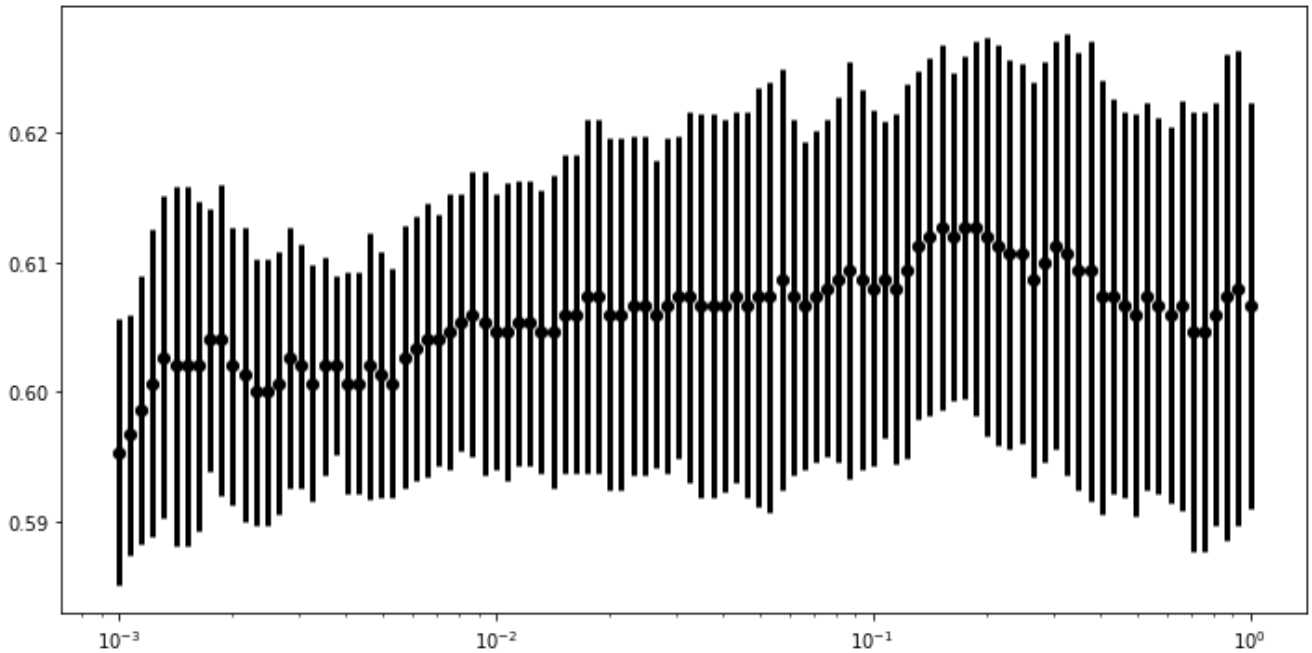
Regresja Logistyczna

Ten algorytm prezentuje podejście naiwne w problemie predykcji jednej dwóch klas.

Przestrzeń parametrów:

```
logregParam = {'logreg__C': np.logspace(-3, 0, 100)}
```

Wynik poszukiwania:



Wybrane Parametry:

```
logreg__C           0.151991
```

Wynik:

```
accuracy on the test set:  0.60
accuracy on the train set:  0.63
```

Macierz pomyłek na zbiorze testowym:

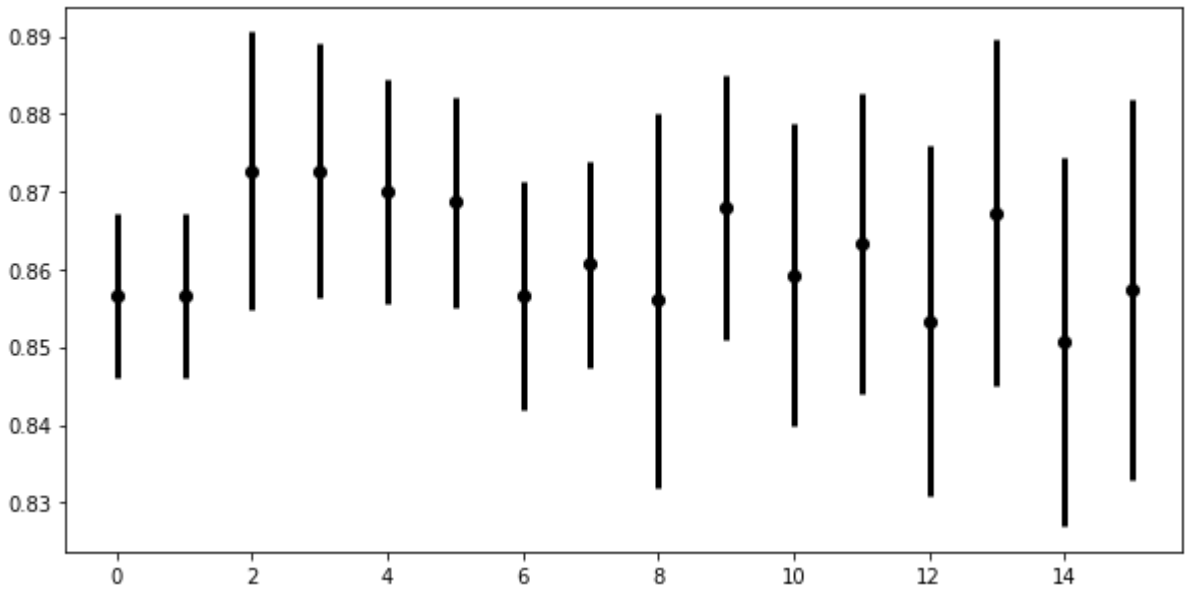
	Predicted -1	**Predicted 1**
Actual -1	115	123
Actual 1	134	128

K-Neighbors Classifier

Przestrzeń parametrów:

```
knnParam = {
    'knn__n_neighbors': [1, 3, 5, 7, 8, 9, 10, 11],
    'knn__weights': ['uniform', 'distance'],
}
```

Wynik poszukiwania:



Wybrane Parametry:

```
knn__n_neighbors                 3
knn__weights               uniform
```

Wynik:

```
accuracy on the train set:  0.94
accuracy on the test set: 0.88
```

Macierz pomyłek na zbiorze testowym:

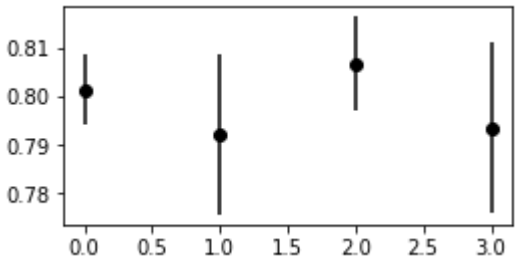
	Predicted -1	Predicted 1
Actual -1	211	27
Actual 1	33	229

DecisionTree Classifier

Przestrzeń parametrów

```
dectreeParam = {
    'dectree__criterion': ['gini', 'entropy'],
    'dectree__class_weight': ['balanced', None]
}
```

Wynik poszukiwania:



Wybrane Parametry:

```
dectree__class_weight              None
dectree__criterion                 gini
```

Wynik:

```
accuracy on the train set:  1.00
accuracy on the test set:  0.79
```

Macierz pomyłek na zbiorze testowym:

	Predicted -1	Predicted 1
Actual -1	108	130
Actual 1	125	137

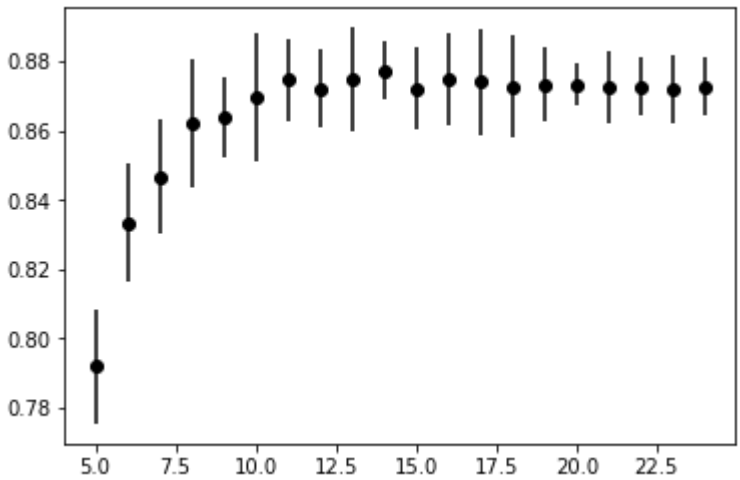
Random Forest

Ostatecznie ten algorytm poradził sobie najlepiej i został wybrany do finalnej predykcji.

Przestrzeń parametrów

```
randtreeParam = {'randtree__max_depth': range(5, 25)}
n_estimators=100
bootstrap=False
```

Wynik poszukiwania:



Wybrane Parametry:

```
randtree__max_depth                14
```

Wynik:

```
accuracy on the train set:  1.00
accuracy on the test set:  0.896
```

Macierz pomyłek na zbiorze testowym:

	Predicted -1	Predicted 1
Actual -1	117	121
Actual 1	129	133

Porównanie jakości

Dokładność na zbiorze testowym:

```
Logistic Regression: 0.60
K-Neighbors Classifier: 0.88
Decision Tree Classifier: 0.79
RandomForestClassifier: 0.89
```

Wybór modelu i metody selekcji danych

Dla każdego modelu uruchamiałem trening z wykorzystaniem wszystkich trzech zestawów cech, dopasowując ostateczne parametry jeszcze raz za pomocą poszukiwania zachłannego oraz mierząc accuracy na zbiorze testowym z wykorzystaniem pięciokrotnej walidacji krzyżowej (ang. Cross Validation).

Zadbałem też o równomierny podział klas w podzbiorach CV za pomocą funkcji `StratifiedKFold`.

```
Logistic Regression
     features_KNR 21        CV accuracy: 62.8 %
     features_DTR 18        CV accuracy: 62.6 %
     features_boruta 20     CV accuracy: 61.6 %
```

```
K-Neighbors Classifier
     features_KNR 21        CV accuracy: 66.6 %
     features_DTR 18        CV accuracy: 85.3 %
     features_boruta 20     CV accuracy: 83.0 %
```

```
Decision Tree Clasifier
     features_KNR 21        CV accuracy: 63.6 %
     features_DTR 18        CV accuracy: 71.4 %
     features_boruta 20     CV accuracy: 71.0 %
```

```
Random Forest Classifier
     features_KNR 19        CV accuracy: 71.6 %
     features_DTR 16        CV accuracy: 81.0 %
     features_boruta 20     CV accuracy: 86.6 %   <---
```

Wyniki

Wybór pada na **Random Forest Classifier** i 20 cech wybranych przez algorytm **Boruta** który osiągnął wynik: **86.6%** na zbiorze testowym stanowiącym 25% zbioru wejściowego.

Finalne predykcja została wykonana modelu wytrenowanym na pełnym zbiorze wejściowym (połączony zbiór treningowy i testowy z powyższych eksperymentów)