Prognozowanie cen telefonów na podstawie ich parametrów technicznych

Urszula Szczęsna i Martyna Leśniak

22 kwietnia 2024

1 Wstęp

Celem naszego projektu było stworzenie modelu uczenia maszynowego, który wykorzystując metodę klasyfikacji pozwoliłby na precyzyjne przypisanie telefonów do odpowiednich przedziałów cenowych na podstawie ich różnorodnych parametrów technicznych. Dane, z których korzystałyśmy, zostały pozyskane z publicznego zbioru danych z platformy Kaggle dostępnego pod adresem https://www.kaggle.com/datasets/ahmedghonem01/phones-price-classification, zawierającego informacje na temat kategorii cen telefonów oraz ich cech.

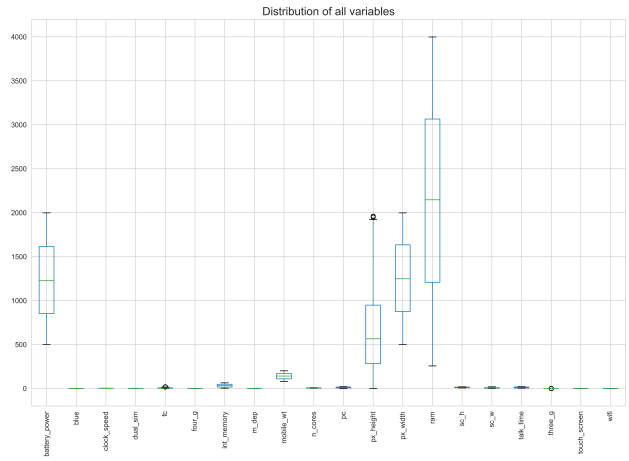
2 Przygotowanie danych

Pierwszym etapem naszego projektu było zapoznanie się z danymi i odpowiednie ich przygotowanie. Zaczęłyśmy od sprawdzenia jakie kolumny znajdują się w naszej ramce danych i jakie są ich typy oraz czy występują braki danych.

```
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  1600 non-null   int64
 1   blue           1600 non-null   int64
 2   clock_speed    1600 non-null   float64
 3   dual_sim       1600 non-null   int64
 4   fc             1600 non-null   int64
 5   four_g         1600 non-null   int64
 6   int_memory     1600 non-null   int64
 7   m_dep          1600 non-null   float64
 8   mobile_wt      1600 non-null   int64
 9   n_cores        1600 non-null   int64
 10  pc             1600 non-null   int64
 11  px_height      1600 non-null   int64
 12  px_width       1600 non-null   int64
 13  ram            1600 non-null   int64
 14  sc_h           1600 non-null   int64
 15  sc_w           1600 non-null   int64
 16  talk_time      1600 non-null   int64
 17  three_g        1600 non-null   int64
 18  touch_screen   1600 non-null   int64
 19  wifi           1600 non-null   int64
 20  price_range    1600 non-null   int64
dtypes: float64(2), int64(19)
```

Rysunek 1: Enter Caption

Ramka składała się z 21 kolumn o wartościach liczbowych oraz nie było żadnych braków danych. Kolumna price_range była naszą kategoryczną zmienną celu, zawierającą wartości z zakresu 0-3. Większość kolumn zawierała zmienne ciągłe, w 6 natomiast pojawiły się zmienne kategoryczne binarne. Nie wymagało to żadnych poprawek. Zbadałyśmy również jak wyglądają rozkłady wszystkich zmiennych. Przedstawione zostały na poniższym wykresie.



Rysunek 2: Enter Caption

Można zauważyć, że różnice pomiędzy zakresami wartości dla różnych zmiennych były bardzo duże, zdecydowałyśmy się więc przeskalować dane używając MinMaxScalera.

Na koniec tego etapu sprawdziłyśmy zależności pomiędzy poszczególnymi parametrami oraz naszą zmienną celu posługując się macierzą korelacji. TU WSTAWIE JA

Widać więc, że większość zmiennych jest bardzo słabo skorelowana z ceną. Główny wpływ na predykcję będzie mieć zmienna zawierająca informację dotyczące pamięci ram. Na tym etapie postanowiłyśmy jednak zostawić nasze dane w takiej postaci, dopiero przy tworzeniu naszego drugiego, bardziej zaawansowanego modelu skupiłyśmy się na zastosowaniu metod Feature Selection oraz grupowania zmiennych.

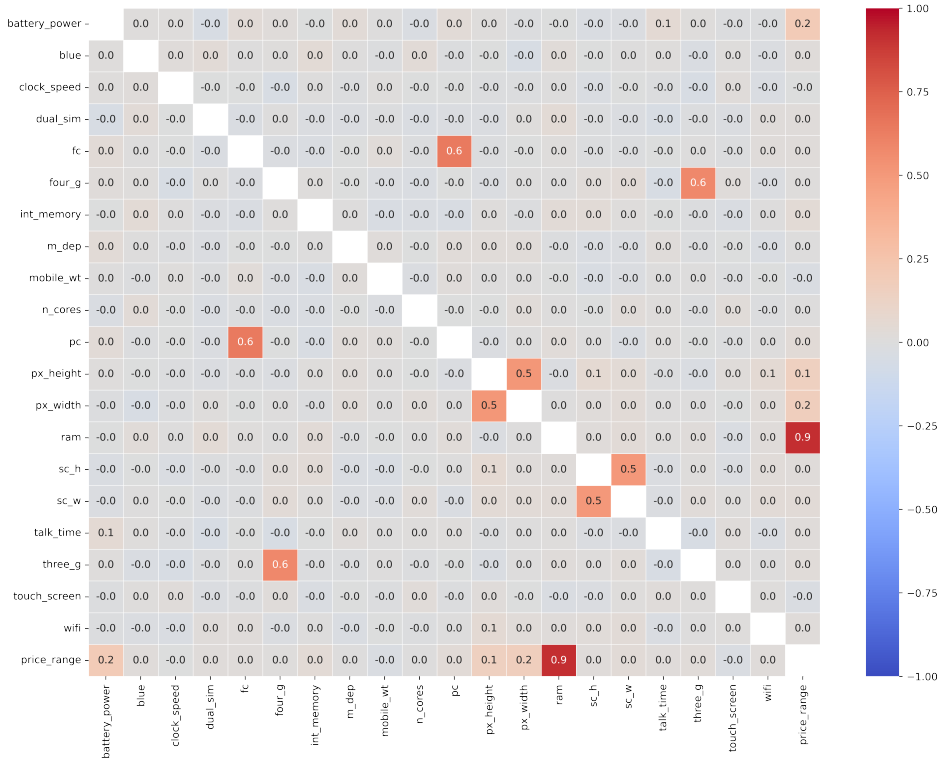
3 Inżynieria cech i wstępne modelowanie

Nasze dane były kompletnie i pozbawione braków, a wszystkie kolumny zawierały wartości numeryczne. Dodatkowo, klasy były zbalansowane, co pozwoliło nam za punkt odniesienia przyjąć DummyClassifier. Z uwagi na równoważność klas, naszym celem było maksymalizowanie dokładności (accuracy) przy jednoczesnym utrzymaniu wysokich wartości recall i precision.

Na początku postanowiłyśmy sprawdzić jak sprawdzą się proste modele:

- Decission Tree
- K- Nearest Neighbors
- Logistic Regression
- Supported Vector Machine
- Gaussian Naive Bayes

Po przeprowadzeniu analizy okazało się, że Logistic Regression sprawdza się najlepiej w kontekście naszego problemu klasyfikacji, osiągając imponującą dokładność na poziomie 98% w przypadku modelu bez dostrojonych hiperparametrów. Jednakże, pomimo tego obiecującego wyniku, postanowiłyśmy



Rysunek 3: Macierz korelacji

przeprowadzić bardziej wszechstronną analizę tych rezultatów poprzez zastosowanie kroswalidacji w celu oceny stabilności modeli.

Kroswalidacja, znana również jako cross-validation, jest techniką często wykorzystywaną do oceny wydajności modelu na różnych zestawach danych treningowych i testowych. Pozwala to na uzyskanie bardziej obiektywnej oceny wydajności modelu poprzez uwzględnienie różnorodności danych.

Poniżej przedstawiamy wykres z wynikami kroswalidacji, który pozwala porównać wydajność różnych modeli na różnych metrykach.

Analiza wyników kroswalidacji pokazuje, że Logistic Regression wciąż utrzymuje się jako lider na wszystkich metrykach, wyprzedzając inne proste modele. Wyniki te potwierdzają skuteczność tego modelu w naszym problemie klasyfikacji i sugerują, że jest to model wart dalszego zbadania i ewentualnego wdrożenia w praktyce.

Rysunek 4: Zbalansowanie klas

4 Pierwszy model

Za pierwszy finalny model przyjęłyśmy Logistic Regression. Przystapiłyhśmy do szukania najlepszych hiperparametrów za pomocą funkcji Grid Search. Parametry, na których się skupiłysmy to:

- parametr C, który określa siłę regularyzacji
- penalty, który określa rodzaj 'kary', regularyzacji
- l1-ratio

Paramtery finalnego modelu:

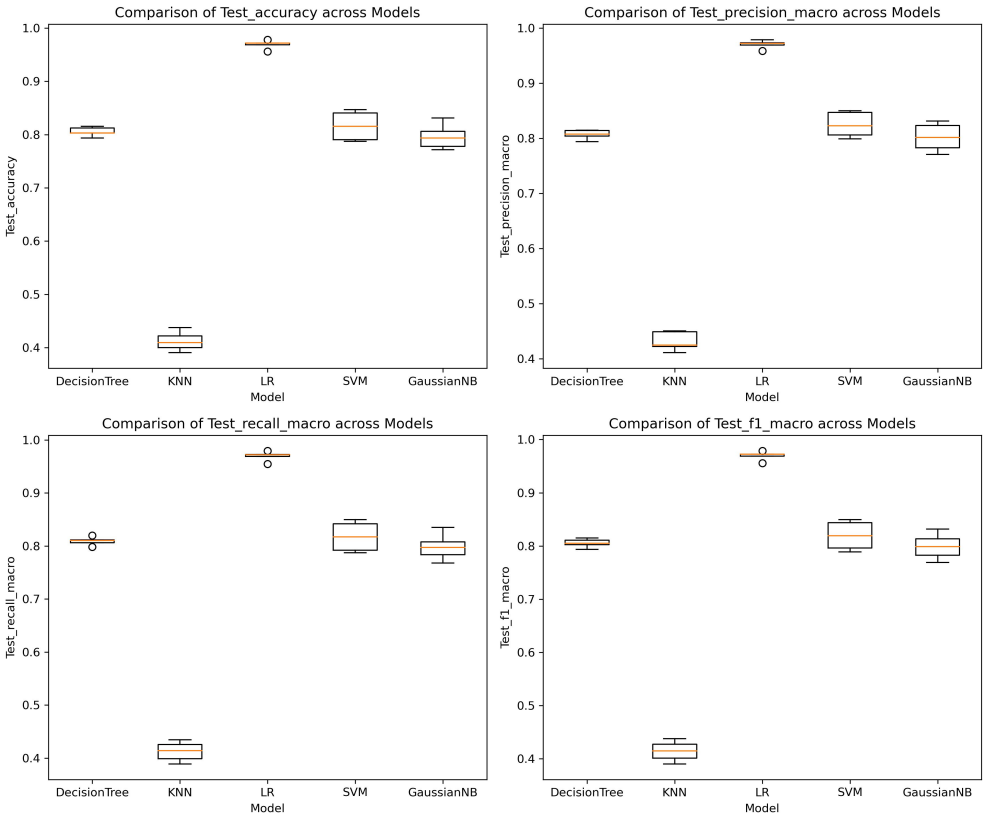
```
Logistic Regression Model:
– max_iter: 6000
– solver: 'saga'
– penalty: 'none'
– C: 0.001
– l1_ratio: 0.1
```

Wyniki modelu:
Wnioski:

- Precyzja (Precision) dla klasy 0 wynosi 1.00, co oznacza, że wszystkie przypadki sklasyfikowane jako klasa 0 są poprawne.
- Czułość (Recall) dla wszystkich klas recall wynosi powyżej 0.97, co oznacza, że model dobrze radzi sobie z wykrywaniem rzeczywiście pozytywnych przypadków każdej klasy.
- Ogólna dokładność modelu (accuracy) wynosi 0.98, co oznacza, że model poprawnie sklasyfikował 98% obserwacji.

Podsumowując raport klasyfikacji potwierdza wysoką skuteczność modelu w klasyfikacji dla wszystkich klas. Zarówno precyzja, jak i czułość są wysokie dla każdej klasy, co oznacza, że model jest dobrze zbalansowany i skutecznie radzi sobie z każdą klasą. Zatem zastosowany model wykazuje bardzo dobrą zdolność do poprawnego przewidywania klas.

5 Drugi model



Rysunek 5: Wyniki kroswalidacji

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        94
           1       0.96      1.00      0.98        98
           2       0.99      0.97      0.98        99
           3       0.98      0.99      0.99       109

    accuracy                           0.98       400
   macro avg       0.98      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

Rysunek 6: Logistic Regression model

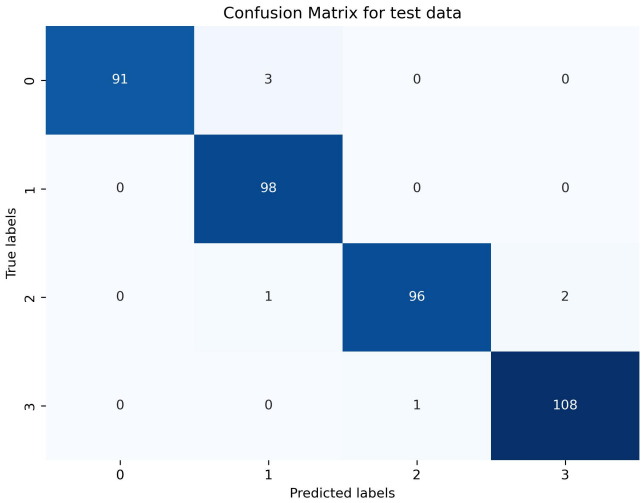
6 Porównanie modeli

Prowadząc porównanie efektywności działania modeli, zdecydowałyśmy się na wykorzystanie walidacji krzyżowej. Naszym celem było dokładne zbadanie osiągnięć obu modeli. Poniżej przedstawiamy wyniki krzyżowej walidacji oraz macierze pomyłek, które pomogą nam zrozumieć, jak dobrze modele radzą sobie z klasyfikacją danych.

Dla modelu Logistic Regression, średnia wartość accuracy dla krzyżowej walidacji wynosi 0.9715, co sugeruje, że model ten osiąga wysoką skuteczność w klasyfikacji danych. Wyniki krzyżowej walidacji dla poszczególnych foldów wahają się od 0.948 do 0.992, co wskazuje na konsystentną wydajność modelu na różnych podzbiorach danych.

Dla Stacked Classifier, średnia wartość accuracy dla krzyżowej walidacji wynosi 0.9385. Chociaż jest to nieco niższe niż dla pierwszego modelu, nadal wskazuje na stosunkowo wysoką skuteczność klasyfikacji. Wyniki krzyżowej walidacji dla poszczególnych foldów dla złożonego modelu wahają się od 0.904 do 0.98.

Wnioskiem jest to, że oba modele wykazują dobrą wydajność, ale Logistic Regression osiąga nieco lepsze wyniki niż Stacked Classifier. Jednak drugi model nadal jest skuteczną alternatywą.



Rysunek 7: Confusion matrix

7 Ewaluacja modelu z wykorzystaniem technik wyjaśnialnej sztucznej inteligencji

Na zakończenie naszej analizy wykorzystałyśmy pakiet dalex, aby zbadać wpływ poszczególnych zmiennych na wyniki modelu. W tym celu skorzystałyśmy z dwóch metod: model_parts oraz permutation_importance.

Metoda model_parts umożliwiła nam obliczenie ważności poszczególnych zmiennych w modelu. Przy użyciu tej metody otrzymałyśmy wyniki, które pokazały, jak każda zmienna przyczynia się do predykcji modelu.

Dodatkowo, dzięki metodzie permutation_importance, mogłyśmy ocenić wpływ poszczególnych zmiennych na spadek wydajności modelu. Ta technika polega na ocenie zmiany w dokładności modelu po losowym przemieszczeniu wartości danej cechy. Wyniki tej analizy pozwoliły nam zidentyfikować, które zmienne mają największy wpływ na dokładność modelu.

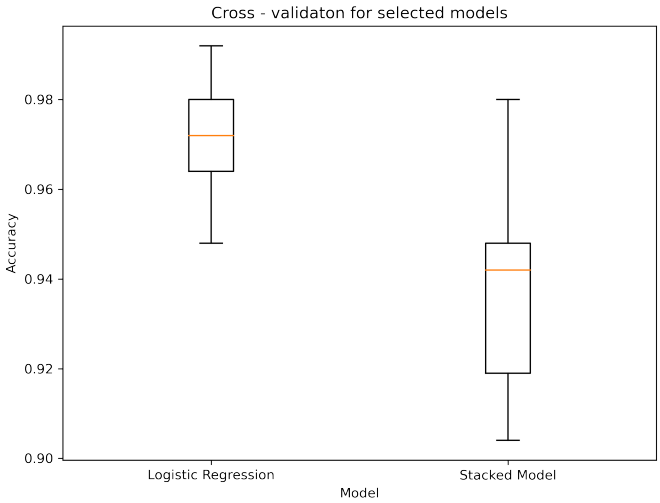
Poniżej przedstawiamy wyniki obu metod dla naszych modeli.

- Ram jest najistotniejszą zmienną, która najsilniej wpływa na predykcję modelu. Zarówno w modelu pierwszym, jak i drugim, ram odpowiada za około 70% accuracy modelu, co stanowi znaczącą część wyjaśnianej zmienności. Jest to zmienna, której wartość wyraźnie różnicuje klasy i stanowi kluczowy czynnik decydujący o predykcji.

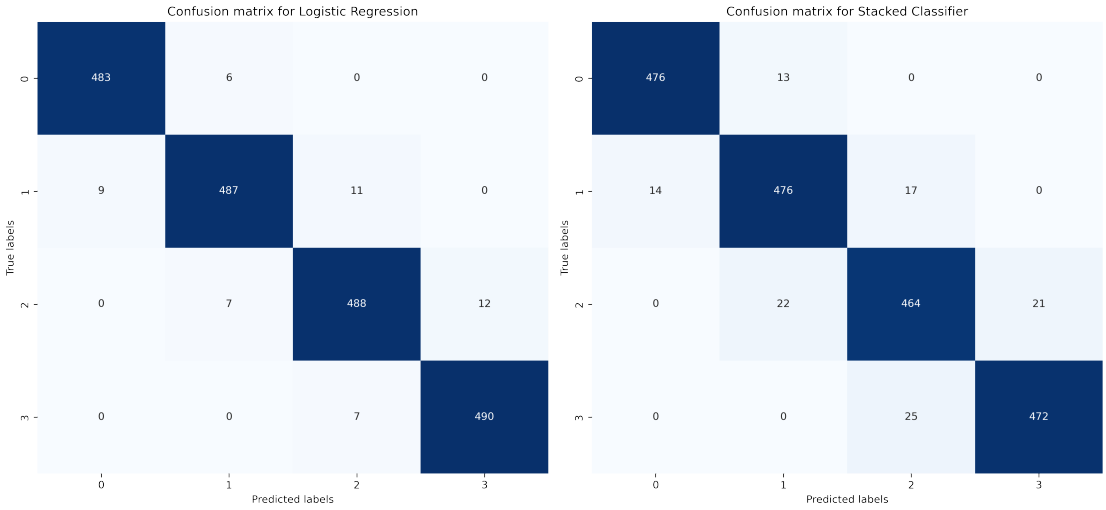
- Battery power również jest istotną zmienną, jednak jej wpływ jest znacznie mniejszy niż w przypadku ram. W obu modelach odpowiada za około 25% accuracy, co sugeruje, że jest to istotny czynnik, ale nie tak kluczowy jak ram.

- W pierszym modelu w miarę istotnymi zmiennymi są px_height i px_width, które w drugim modelu zostały przekształcone na zmienna px_area.

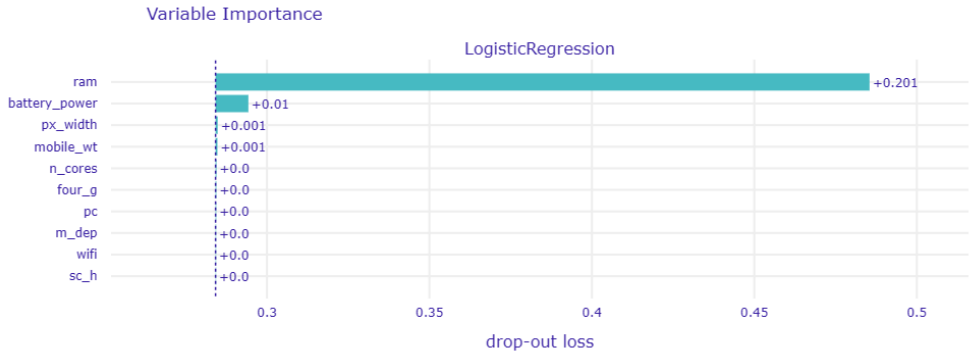
- Pozostałe zmienne mają znikomy wpływ na predykcję modelu. Ich udział w wyjaśnianiu zmienności jest minimalny, co sugeruje, że nie wniosły istotnego wkładu w poprawę dokładności modelu. Wartość tych zmiennych nie różnicuje wystarczająco mocno klas, aby miały znaczący wpływ na predykcje.



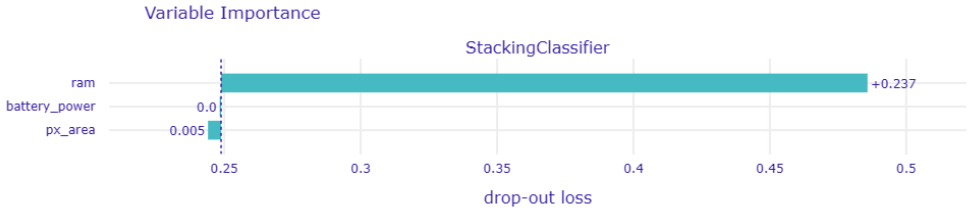
Rysunek 8: Crossvalidation



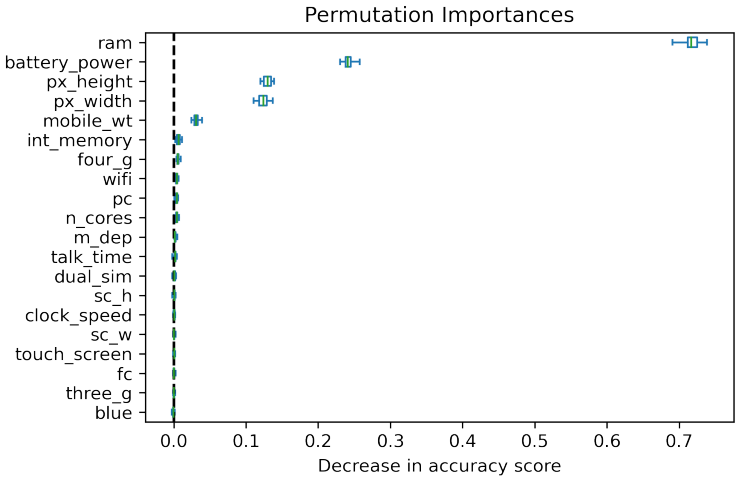
Rysunek 9: Confusion Matrix



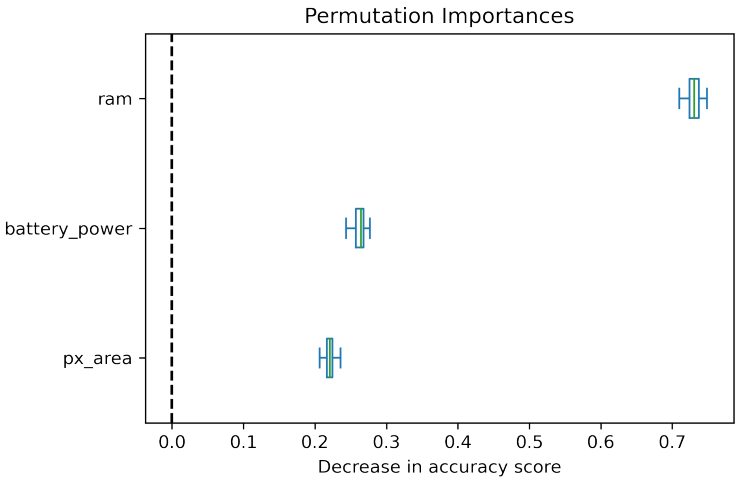
Rysunek 10: Model 1 - Logistic Regression



Rysunek 11: Model 2 - Stacking Classifier



Rysunek 12: Accuracy drop - Logistic Regression



Rysunek 13: Accuracy drop - Stacking Classifier