Testy poprawności funkcji `knn` i funkcji agregujących

Emil Dragańczuk

17 maja 2020

Wprowadzenie

W poniższym raporcie sprawdzamy poprawność funkcji `knn` poprzez sprawdzenie warunku koniecznego dla różnych wywołań oraz analizując procent poprawnie odgadniętych etykiet i błędy bezwzględne na wykresach.

Sprawdzenie warunku koniecznego poprawności funkcji `knn`

Poniżej sprawdzimy czy dla funkcji `knn` zachodzi warunek konieczny poprawności, czyli czy dla wywołania funkcji z parametrami `knn(trainingSet, trainingLabels, testSet = trainingSet, k = 1, p = p)` wynikowe etykiety są równe etykietom ze zbioru treningowego. Będziemy rozważać $p \in \{1, 2, \infty\}$. Użyjemy do tego poniższej funkcji:

```
necessaryConditionForKnn <- function(dataSet, p = 2) {
  labels <- dataSet[,1]
  variables <- data.matrix(dataSet[, 2:ncol(dataSet)])

  resultLabels <- knn(variables, labels, variables, 1, p)

  print(paste(
    "Warunek konieczny dla p=", p, " spełniony: ",
    all(labels == resultLabels), sep = ""))
}
```

Zbiór danych `abalone.csv`

```
for (p in c(1, 2, Inf)){
  necessaryConditionForKnn(head(abalone, 1000), p)
}
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = 1  spelniony:  NA"
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = 2  spelniony:  NA"
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = Inf  spelniony:  NA"
```

Zbiór danych californiahousing.csv

```
for (p in c(1, 2, Inf)){
  necessaryConditionForKnn(head(californiahousing, 1000), p)
}
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = 1  spelniony:  NA"
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = 2  spelniony:  NA"
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = Inf  spelniony:  NA"
```

Zbiór danych winequality-red.csv

```
for (p in c(1, 2, Inf)){
  necessaryConditionForKnn(head(redwine, 1000), p)
}
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = 1  spelniony:  NA"
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = 2  spelniony:  NA"
```

```
## Warning in all(labels == resultLabels, sep = ""): przekształcenie argumentu typu
## 'character' w logiczny
```

```
## [1] "Warunek konieczny dla p = Inf  spelniony:  NA"
```

Wykresy błędów bezwzględnych dla różnych konfiguracji danych

Rozważymy procent poprawnych predykcji oraz błędy bezwzględne dla różnych wywołań funkcji knn. W wywołaniach funkcji uwzględnimy różne zbiory danych oraz 3 z dostępnych agregatorów: moda, srednia_a i wazonyrand. Testy zostaną przeprowadzone dla zbiorów treningowych o wielkości 1000, 100 danych testowych, $k = 10$ oraz $p = 2$. Wykresy będziemy rysować następującą funkcją:

```
plotAbsoluteErrors <- function(dataSet, n, m, k, aggregator = moda, p = 2){
  nm <- n + m
  sampledData <- dataSet[sample(nm, size = nm, replace = FALSE), 1:3]
  dataNames <- names(sampledData)
  trainingSet <- data.matrix(sampledData[1:n, -1])
  trainingLabels <- sampledData[1:n, 1]
  testSet <- data.matrix(sampledData[n+1:m, -1])
  correctLabels <- sampledData[n+1:m, 1]

  resultLabels <- aggregator(knn(trainingSet, trainingLabels, testSet, k, p))
```

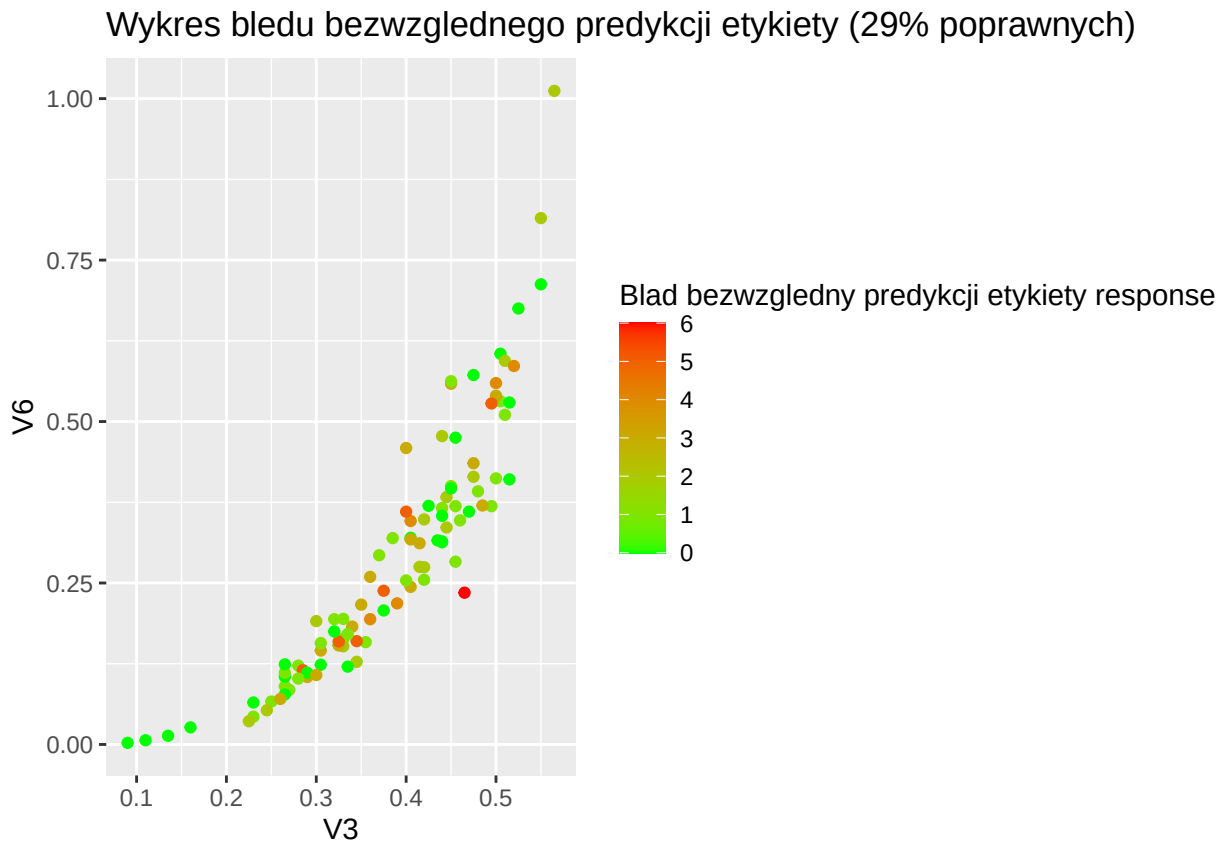
```
  correctPercent <- round(sum(resultLabels == correctLabels) / m * 100, 1)
  result <- data.frame(cbind(abs(correctLabels - resultLabels), testSet))

  ggplot(result, aes(x=result[, 2],y=result[, 3], col=result[, 1])) +
    geom_point() +
    ggtitle(paste("Wykres bledu bezwzglednego predykcji etykiety (", correctPercent, "% poprawnych)", s
    xlab(dataNames[2]) +
    ylab(dataNames[3]) +
    scale_color_gradient(low="green", high="red", name = paste("Blad bezwzgledny predykcji etykiety", da
}
```

Zbiór danych `abalone.csv`

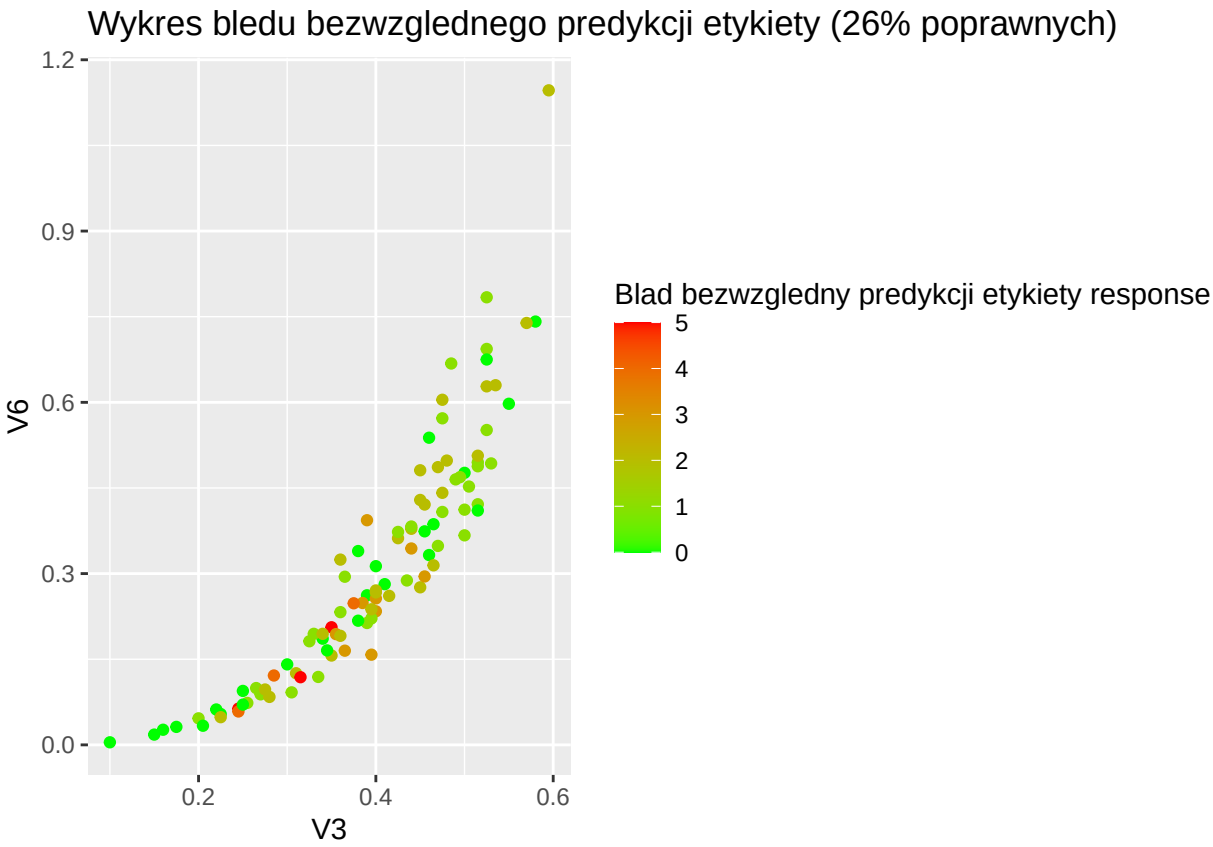
Agregator `moda`

```
plotAbsoluteErrors(abalone[,c(1, 3, 6)], 1000, 100, k = 10, aggregator = moda, p = 2)
```



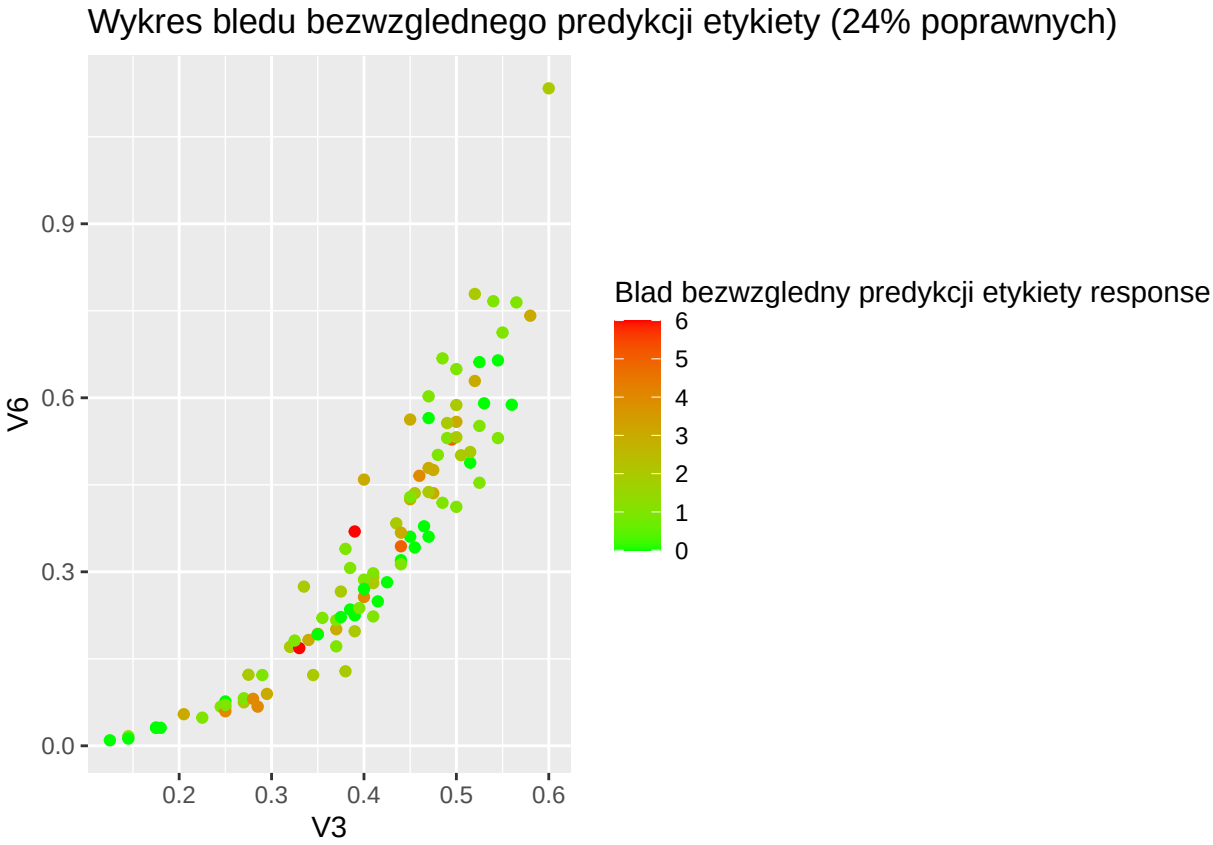
Agregator `srednia_a`

```
plotAbsoluteErrors(abalone[,c(1, 3, 6)], 1000, 100, k = 10, aggregator = srednia_a, p = 2)
```



Agregator wazonyrand

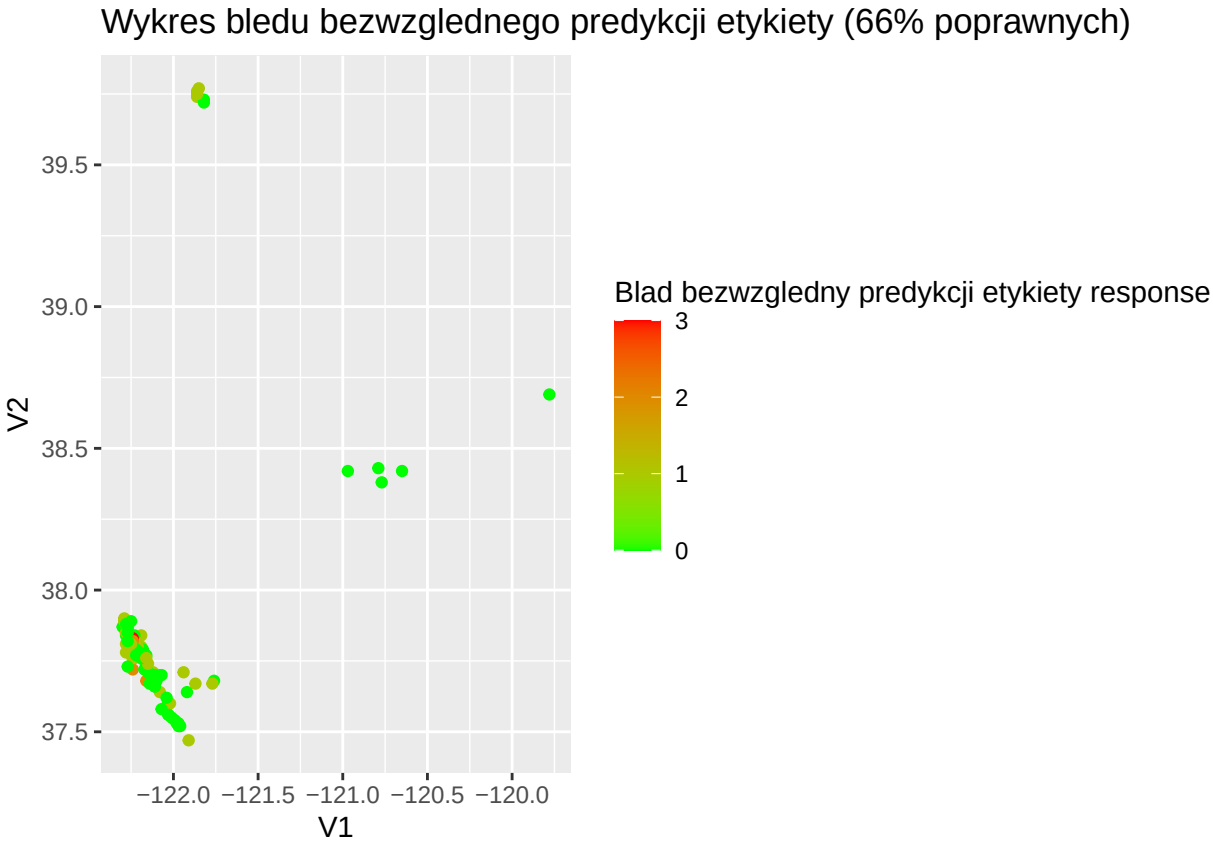
```
plotAbsoluteErrors(abalone[,c(1, 3, 6)], 1000, 100, k = 10, aggregator = wazonyrand, p = 2)
```



Zbiór danych `californiahousing.csv`

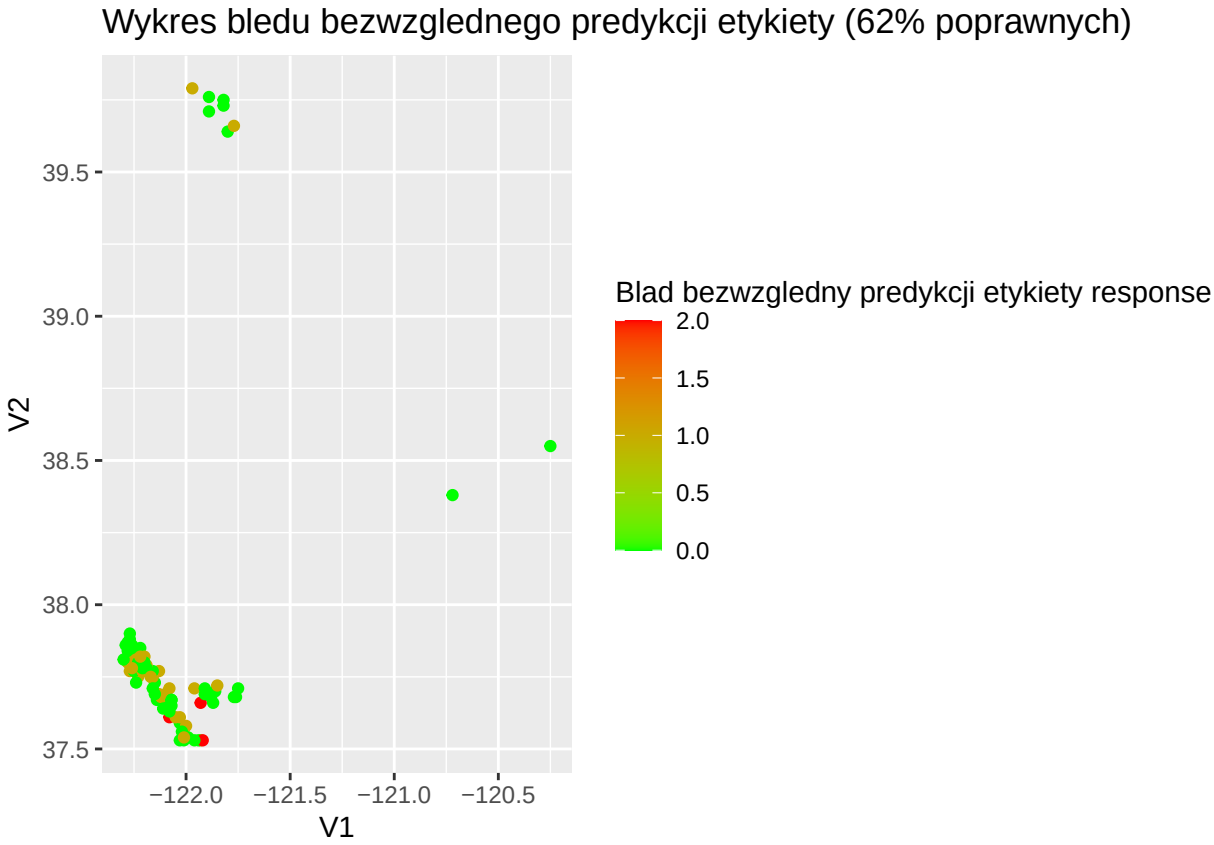
Agregator `moda`

```
plotAbsoluteErrors(californiahousing, 1000, 100, k = 10, aggregator = moda, p = 2)
```



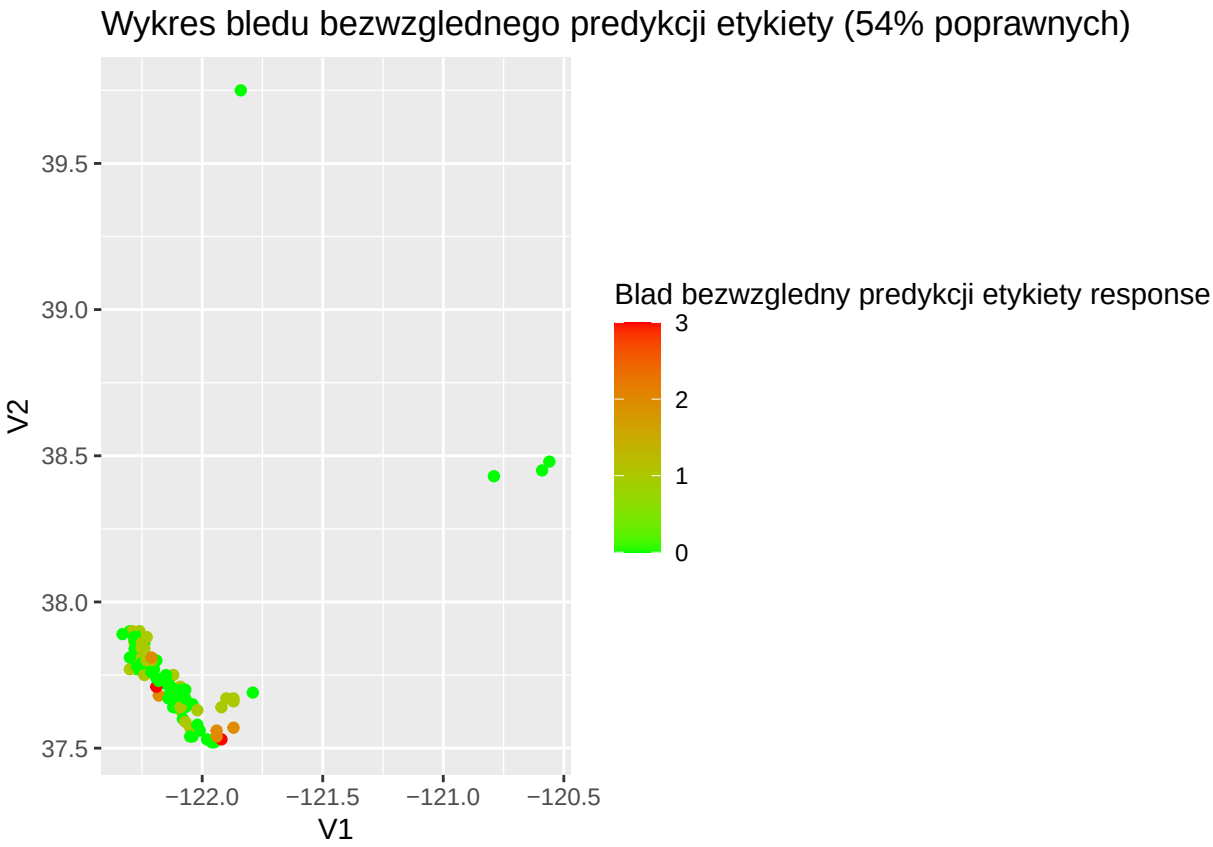
Agregator `srednia_a`

```
plotAbsoluteErrors(californiahousing, 1000, 100, k = 10, aggregator = srednia_a, p = 2)
```



Agregator `wazonyrand`

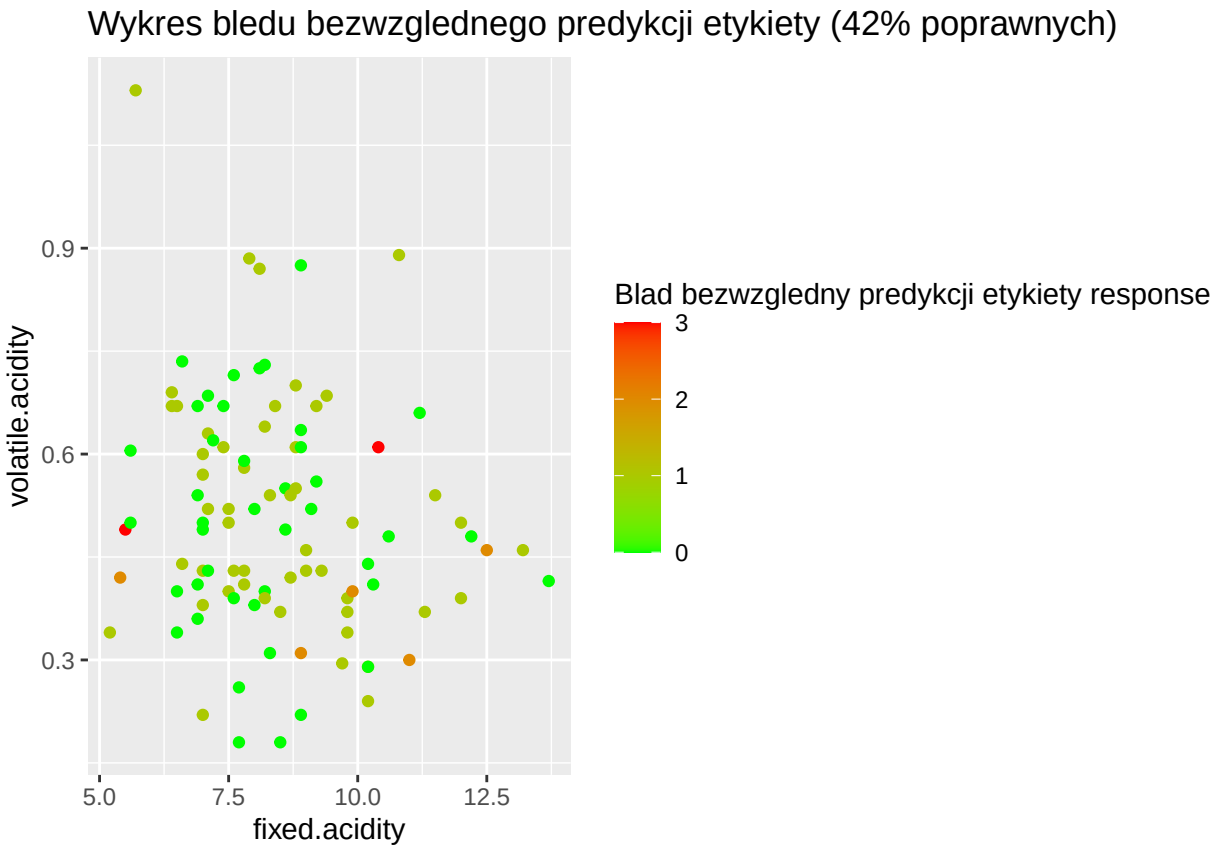
```
plotAbsoluteErrors(californiahousing, 1000, 100, k = 10, aggregator = wazonyrand, p = 2)
```



Zbiór danych `winequality-red.csv`

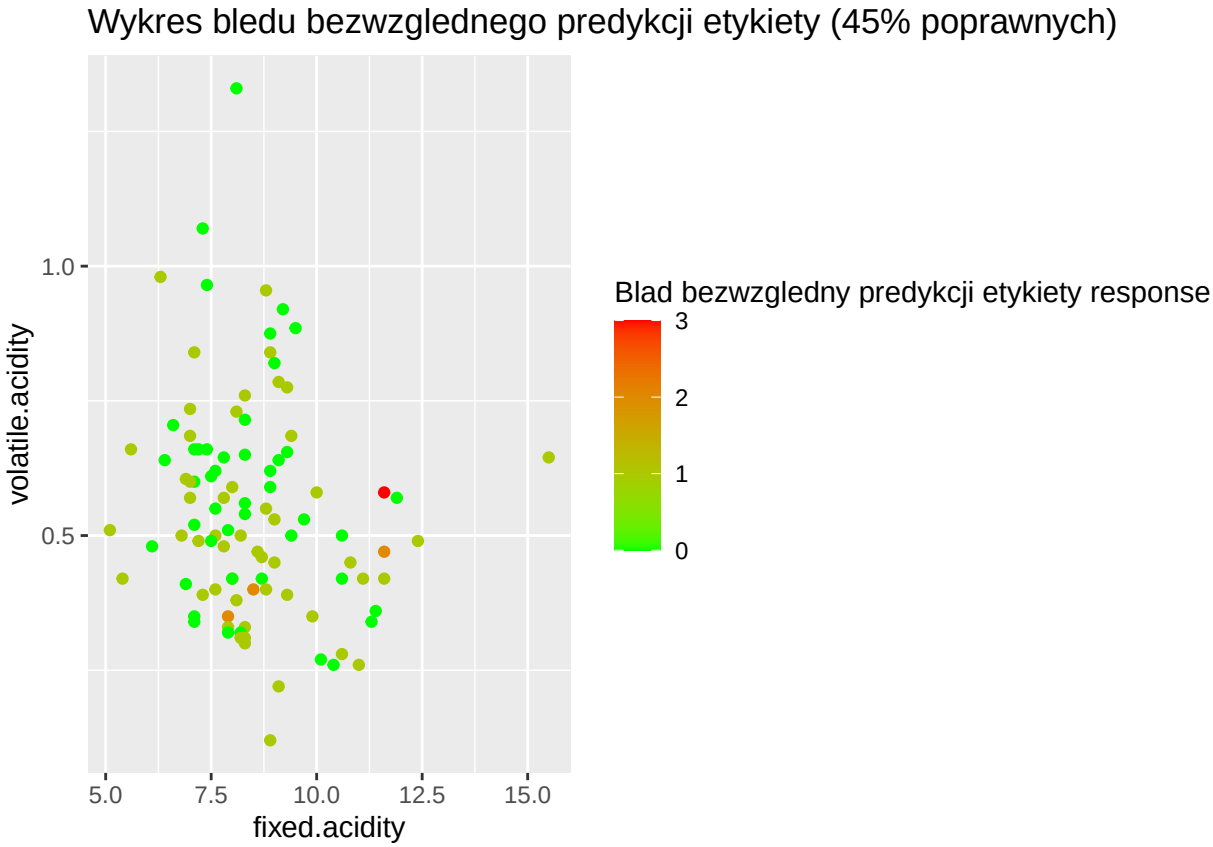
Agregator `moda`

```
plotAbsoluteErrors(redwine, 1000, 100, k = 10, aggregator = moda, p = 2)
```



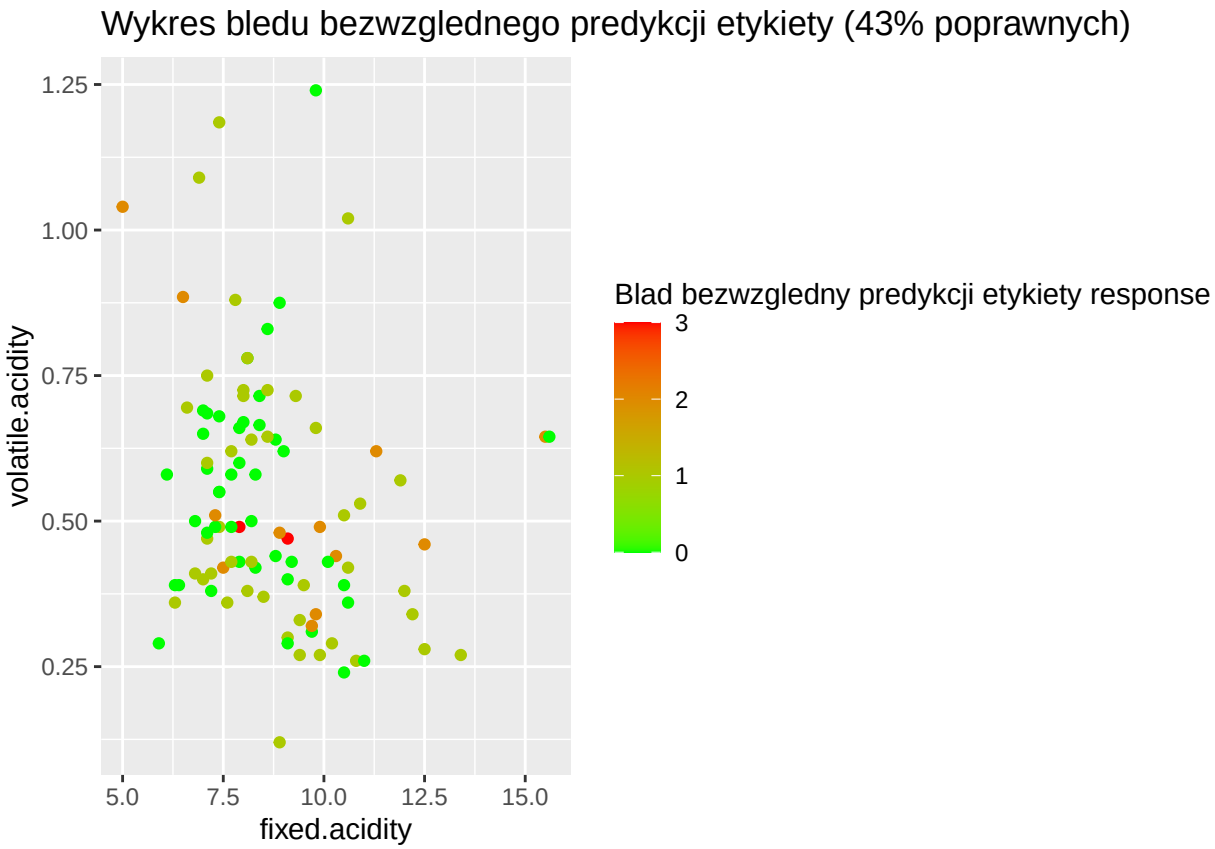
Agregator srednia_a

```
plotAbsoluteErrors(redwine, 1000, 100, k = 10, aggregator = srednia_a, p = 2)
```



Agregator `wazonyrand`

```
plotAbsoluteErrors(redwine, 1000, 100, k = 10, aggregator = wazonyrand, p = 2)
```



Podsumowanie wykresów

Procenty poprawnie przewidzianych etykiet są w każdym przypadku znacznie wyższe niż jakby były nadawane losowo. Ponadto na wykresach można zobaczyć, że znaczna część etykiet jest minimalnie oddalona od poprawnych wyników, tylko pojedyncze wartości są zupełnie niepoprawne. Poza poprawnością mimo wszystko nie da się zauważyć jak bardzo zbiór treningowy wpływa na efektywność algorytmu.