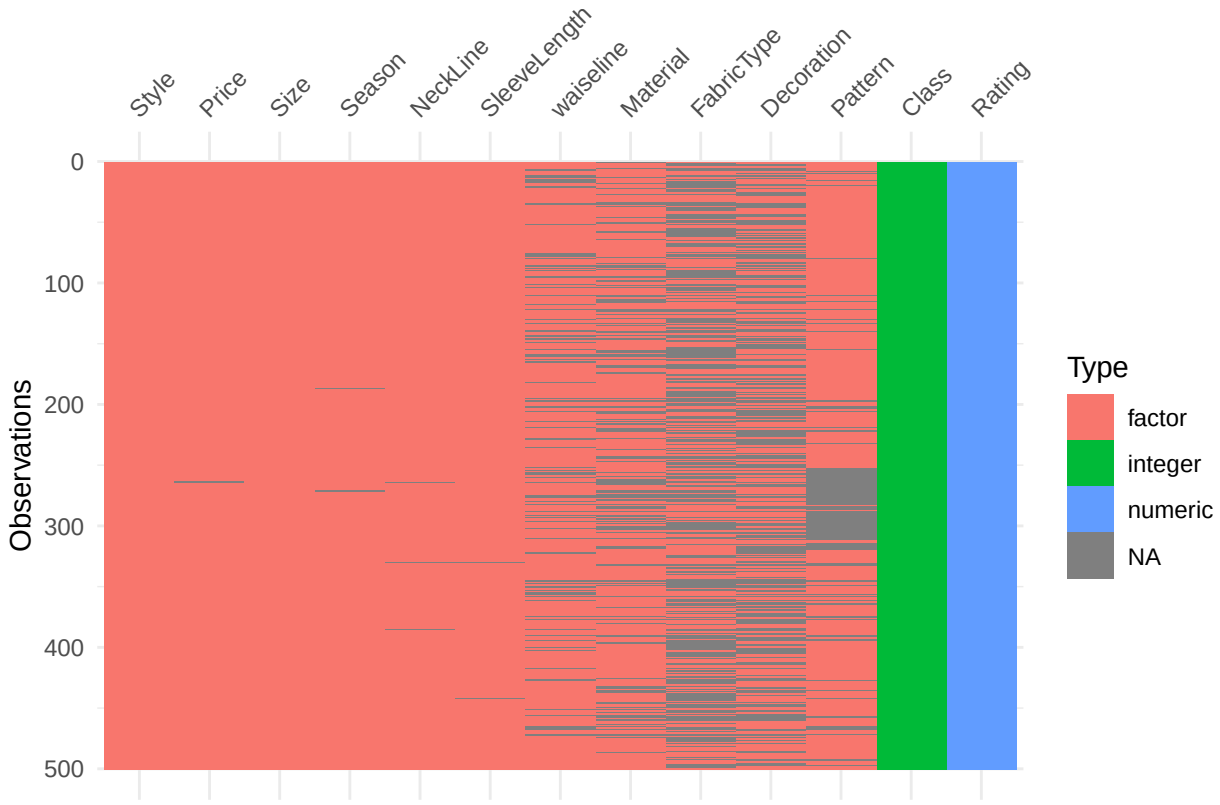
Projekt 1

Jan Borowksi

15 03 2020

Eksploracja Danych

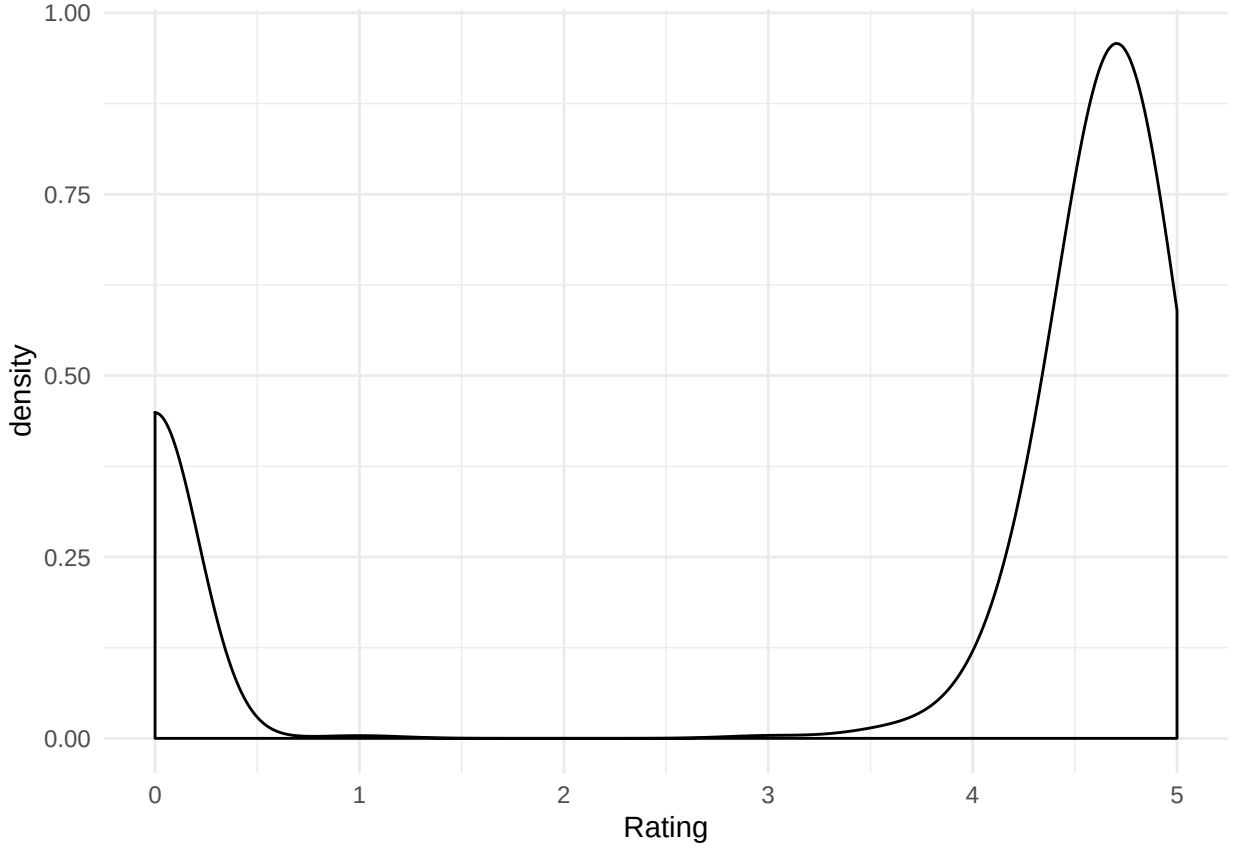
Zaczniemy od eksploracji danych ze zbioru dresses-sales z OpenML. W pierwotnym zbiorze występują braki danych przedstawione jako "?" zostały one już zamienione na **NA**. Najpierw przyjrzyjmu się typom zminnych w zbiorze:



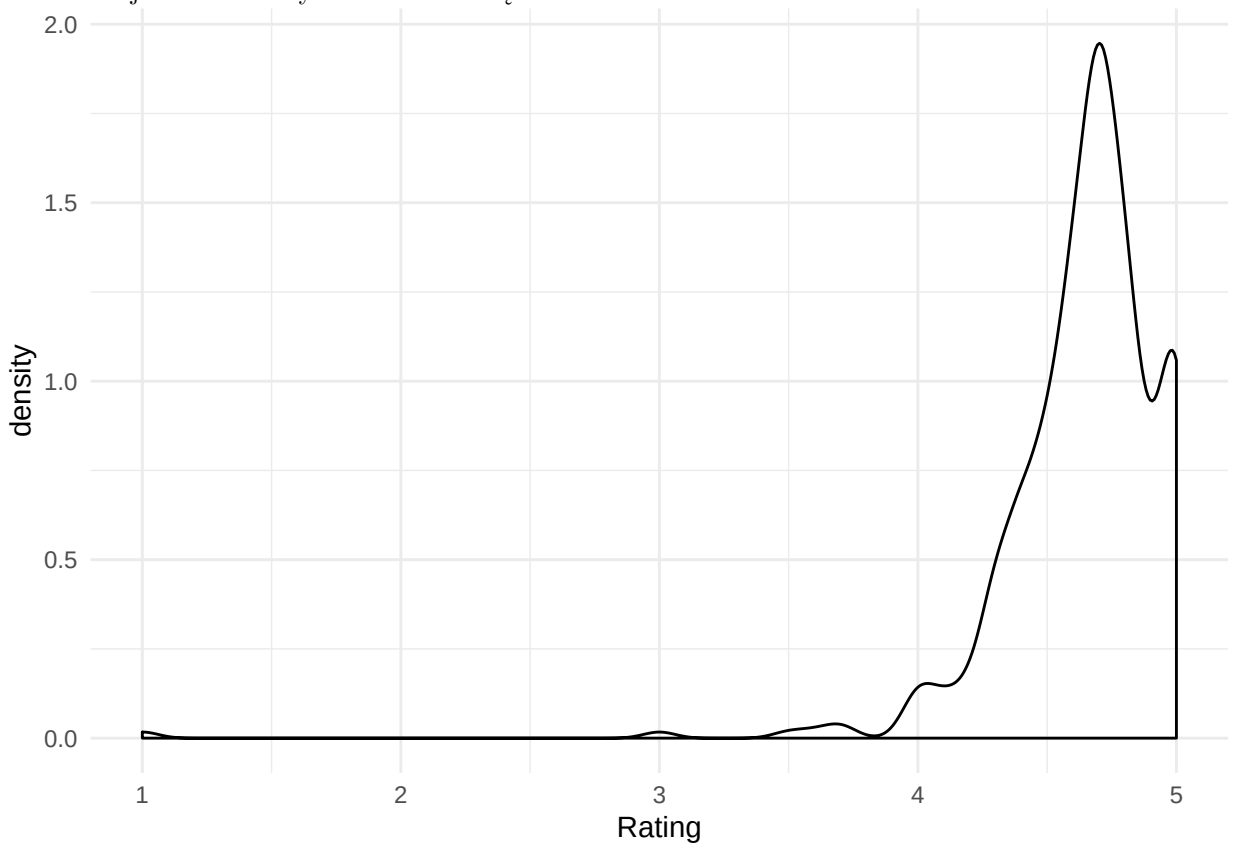
Widzimy ,że zdecydowana większośc zmiennych to zmienne kategoryczne. Przyjrzyjmy się im bliżej:

```
##       Style           Price        Size         Season          NeckLine
##  Casual  :232   Average  :252   free :173   Summer :159   o-neck    :271
##  Sexy    : 69   Low      :129   L    : 96   Spring :122   v-neck    :124
##  party   : 51   low      : 45   M    :177   Winter : 99   slash-neck: 25
##  cute    : 45   Medium   : 30   s    :  1   Automn : 61   boat-neck : 19
##  vintage : 25   very-high: 21   S    : 37   winter : 46   Sweetheart: 14
##  bohemian: 24   (Other)  : 21   small:  1   (Other): 11   (Other)   : 44
##  (Other) : 54   NA's     :  2   XL   : 15   NA's   :  2   NA's      :  3
##        SleeveLength    waiseline           Material        FabricType
##  sleevless   :223   ?       :  0   cotton       :152   chiffon   :135
##  full        : 97   dropped :  4   polyster     : 99   broadcloth: 31
##  short       : 96   empire  :104   silk         : 26   worsted   : 19
##  halfsleeve  : 35   natural :304   chiffonfabric: 25   jersey    : 12
##  threequarter: 17   princess:  1   mix          : 12   shiffon   :  9
##  (Other)     : 30   NA's    : 87   (Other)      : 58   (Other)   : 28
##  NA's        :  2                  NA's         :128   NA's      :266
##      Decoration       Pattern
##  lace     : 70   solid    :203
##  sashes   : 42   print    : 71
##  beading  : 22   patchwork: 48
##  applique : 21   animal   : 21
##  hollowout: 21   striped  : 17
##  (Other)  : 88   (Other)  : 31
##  NA's     :236   NA's     :109
```

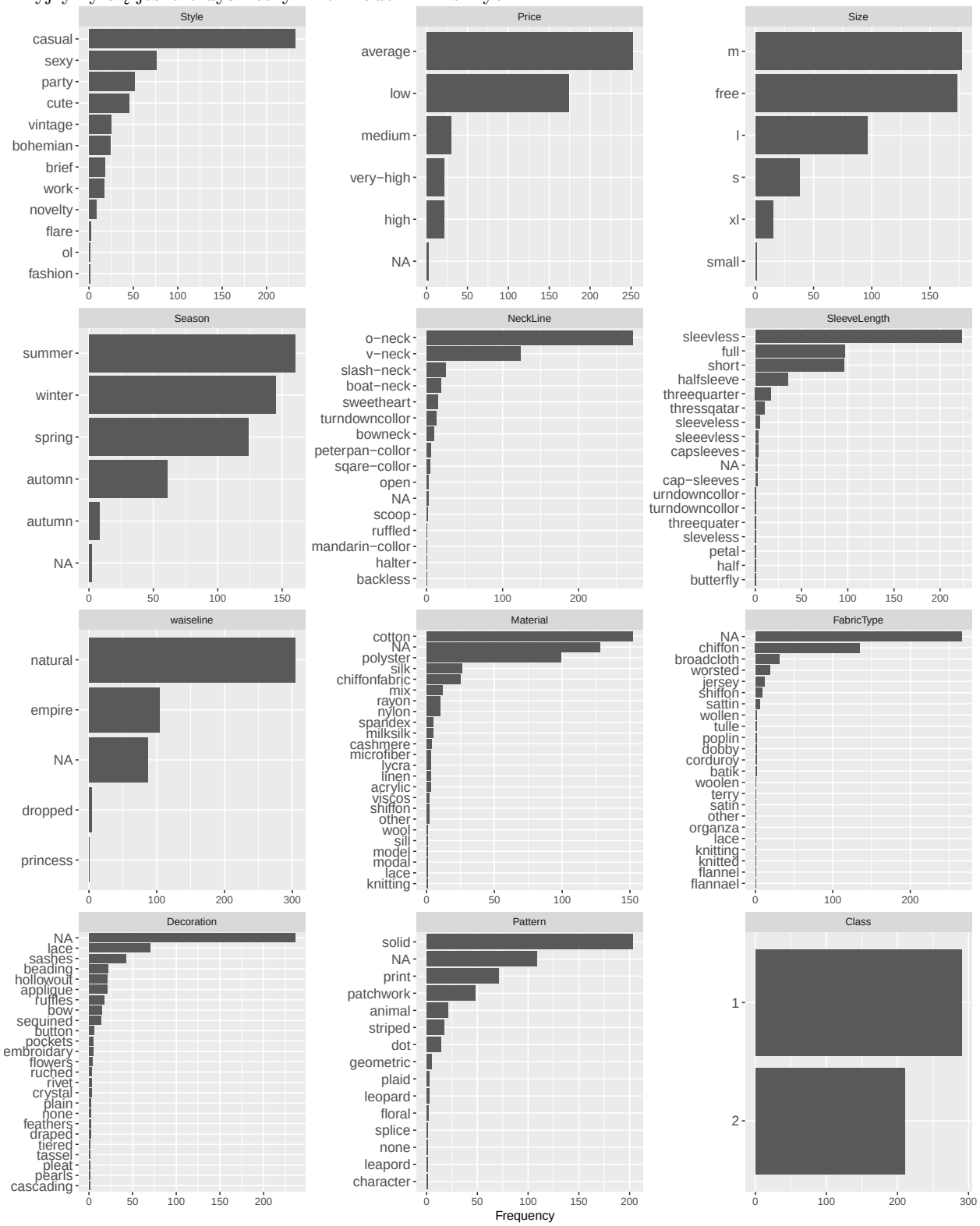
Widzimy pojawiające się sytuację jak w zmiennej **Price**, gdzie mamy "low" i "Low" pozbędę się takich błędów przed dalszą eksploracją. Nie ma większego sensu przygladać się macierzy korelacji ponieważ nie działa ona zbyt dobrze dla zmiennych kategorycznych. Sprawdzę, więc rozkład zmiennej ciagłej "Rating":



Widzimy rozkład bimodalny ale z opisu wiemy ,że Rating przyjmuje wartości z zbioru 1-5 więc 0 należy traktować jako braki danych. Po ich usunięciu:

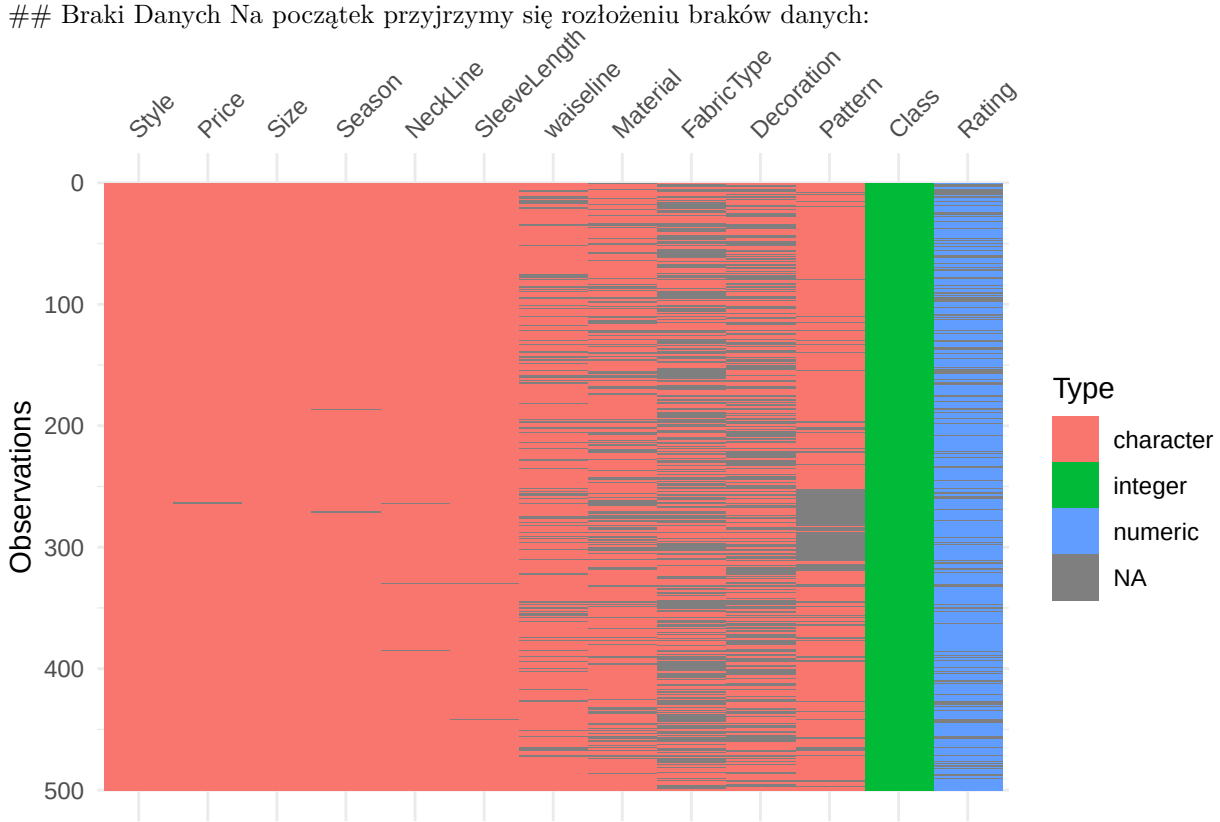


Przyjżymy się jeszcze dyskretnym rozkładom zmiennych:



Widzimy ,że w wiekszości są to zmienne o rozkładie wykładniczym. Warto też zauważyć dość równy podział klas.

Braki Danych Na początek przyjrzymy się rozłożeniu braków danych:



Widzimy ,że większe ilości braków danych występują w przypadku zmiennych walseline,Material,FabricType,Decoration,Pattern oraz Rating jeśli jako braki traktować 0. Oprócz tego widać pojedyncze braki w pozostałych zmiennych. Z nimi poradzę sobie poprzez usunięcie 7 wierszy z całej ramki jeszcze przed rozpoczęciem testów.

Uzupełnanie braków danych

W tej sekcij przygotuję dane uzupełnione na różne sposoby:

```
#Zaczniemy od usunięcia kolumn zawierających braki
dresses_sales_col_remove <- dresses_sales[,-c(3,8:12)]
#Usunięcia wierszy zawierających braki
dresses_sales_row_remove<- dresses_sales%>%drop_na()
#uzupełnienie medianą,średnia,modą) potrzebny będzie podział zbioru na testowy i treningowy.

#Do pozostałych technik
# Wykorzystamy stały podział na zbiory testowy i trenignowy
train_set = sample(length(row.names(dresses_sales)), 0.8 * length(row.names(dresses_sales)))
test_set = setdiff(seq_len(length(row.names(dresses_sales))), train_set)

# Uzupełnię osobno w zbiorze testowym i treningowym
columns_to_imput <- c('waiseline','Material','FabricType','Decoration','Pattern')

# Uzupełnanie modą
Mode <- function(x) {
  ux <- unique(x[!is.na(x)])
  ux[which.max(tabulate(match(x, ux)))]
}



dresses_sales_mode <- dresses_sales
for (i in columns_to_imput){
  mode <- Mode(dresses_sales_mode[train_set,i])
  NA_position <- ifelse(is.na(dresses_sales_mode[train_set,i]),TRUE,FALSE)
  dresses_sales_mode[train_set,i][NA_position] <- mode

}
dresses_sales_mode_mean <- dresses_sales_mode
dresses_sales_mode_median <- dresses_sales_mode

mode <- Mode(dresses_sales_mode[train_set,'Rating'])
NA_position <- ifelse(is.na(dresses_sales_mode[train_set,'Rating']),TRUE,FALSE)
dresses_sales_mode[train_set,'Rating'][NA_position] <- mode

# Uzupełnaianie inaczej jest możliwe tylko dla kolumny Rating w innych
# wypadkach średnia ani mediana nie ma sensu

# Uzupełneinie średnią zmiennej Rating
mean_ <- mean(dresses_sales_mode_mean[train_set,'Rating'],na.rm=TRUE)
NA_position <- ifelse(is.na(dresses_sales_mode_mean[train_set,'Rating']),TRUE,FALSE)
dresses_sales_mode_mean[train_set,'Rating'][NA_position] <- mean_
# Uzupełnianie zmiennej Rating medianą
media <- median(dresses_sales_mode_median[train_set,'Rating'],na.rm=TRUE)
NA_position <- ifelse(is.na(dresses_sales_mode_median[train_set,'Rating']),TRUE,FALSE)
dresses_sales_mode_median[train_set,'Rating'][NA_position] <- media
```

Po przygotowaniu danych można przejść do trenowania algorytmu.

Przygotowanie algorytmu

Użylem krosvalidacji do znaleziena najlepszych parametrów dla wybranego uzupełnienia (będę uzywał drzewa decyzyjnego). Funkcja znajdująca najlepsze parametry:

```
# Krosvalidacja

# funkcja zwraca algorytm z optymalnymi parametrami
# do funkcji podajemy zbiór treningowy
param_search <- function(df){
task = TaskClassif$new(id = "col_remove", backend = df, target = "Class")
cv = rsmp("cv", folds = 5)
heat_map <- matrix(nrow=20,ncol=30)
rownames(heat_map) <- 1:20




for (i in 1:20){
 for (j in  1:30){
   learner = mlr_learners$get("classif.rpart")
   learner$param_set$values = mlr3misc::insert_named(
  learner$param_set$values,
  list(cp = i/20, minsplit = j)
)


   invisible(capture.output(rr <-  resample(task, learner, cv)))

   heat_map[i,j] <- mean(rr$score(msr("classif.acc"))$classif.acc)

 }

}
# Zwraca algorytm z najlepszymi parametrami
a<- which(heat_map == max(heat_map), arr.ind = TRUE)
best_lerner = mlr_learners$get("classif.rpart")
learner$param_set$values = mlr3misc::insert_named(
  learner$param_set$values,
  list(cp = a[1]/20, minsplit = a[2]))
return(best_lerner)
}

# Funkcja przeprowadzająca test
accuracy <- msr("classif.acc")
precision <- msr("classif.precision")


test <-  function(alg,train_s,test_s,df){
  task <- TaskClassif$new(id = "some", backend = df, target = "Class")

 # Trenowanie
  alg$train(task, row_ids = train_s)

  # Predykcja
```

```
  prediction <- alg$predict(task, row_ids = test_s)

  # Miary
  acc <- prediction$score(accuracy)
  pr <- prediction$score(precision)

 return(c(acc,pr))
  }
```

Porównanie technik imputacij danych

W każdym wypadku postepuję według schematu:
1. Przy pomocy kroswalidacji znajduję najlpesze wartości parametrów na zbiorze treningowym,
2. Trenuję algorytm na zbiorze treningowym z ustalonymi parametrami,
3. Stosuję wybraną technikę imputacji dla zbioru testowego,
4. Testuję wytrenowany algorytm na zbiorze testowym.

Zaczniemy od usuwania kolumn:

```
## [1] "Dokładność :0.697"
```

```
## [1] "Precyzja :0.7333"
```

Usuwanie wierszy:

```
## [1] "Dokładność :0.75"
```

```
## [1] "Precyzja :0.5714"
```

Dalej w analogiczy sposób używając wcześniej przygotowanego podziału.
Uzupełnianie modą:

```
## [1] "Dokładność :0.5859"
```

```
## [1] "Precyzja :0.5385"
```

Uzupełnianie zmiennych kategorycznych modą i ciągłych średnią:

```
## [1] "Dokładność :0.5859"
```

```
## [1] "Precyzja :0.5385"
```

Uzupełnianie zminnych kategorycznych modą i ciągłych medianą:

```
## [1] "Dokładność :0.5859"
```

```
## [1] "Precyzja :0.5385"
```

Ponieważ drzewa decyzjne dopuszczają taką możliwość sprawdźmy jaki będzie wynik bez usuwania braków:

```
## [1] "Dokładność :0.6061"
```

```
## [1] "Precyzja :0.561"
```

Zastosuje jescze jedną technike polegającą na zamianie kolumny z brakami na kolune 0,1 gdzie 0 oznacza brak danych. Rating zastąpimy średnią :

```
## [1] "Dokładność :0.7071"
```

```
## [1] "Precyzja :0.7273"
```

Podsumowanie

Porównanie wyników

Table 1: Wyniki testów

	Dokładność	Precyzja
Usuwanie kolumn	0.697	0.733
Usuwanie wierszy	0.750	0.571
Uzupełnanie modą	0.586	0.538
Uzupełnanie modą i średnią	0.586	0.538
Uzupełnianie modą i medianą	0.586	0.538
Brak uzupełniania	0.606	0.561
Kolumny 0-1	0.707	0.727

Po pierwsze należy zauważyć ,że wyniki są ogólnie słabe może to wynikać z danych zawierających prawie same zmienne kategoryczne (sytuację mógł by poprawić ich encoding).
Porównanie technik imputacij:
1.**Usuwanie kolumn** - dobry wynik na tle pozostałych , ale prowadzi do usunięcia połowy danych nie jest to najlepsze rozwiązanie w ogólnym przypadku,
2.**Usuwanie wierszy** - wynik pozornie dobry, ale należy pamiętać ,że w pozostawiło to niecałe 20% danych czyli 75 obserwacij. Ciężko wyciągać jakieś wnioski na podstawie tak małej próbki, ale napewno prowadzi do utraty sporej części danych,
3.**Uzupełnianie modą** - wynik nie zbyt dobry bliski klasyfikatora przypadkowego ,
4.**Uzupełnianie modą i średnią** - wynik taki sam jak poprzednio,
5.**Uzupełnianie modą i medianą** - wynik taki jak dwa poprzednie, wypełnienie kolumny Rating nie ma większego znaczenia,
6.**Brak uzupełnienia** - wynik nieco lepszy od poprzednich choć również bardzo słaby, w praktyce miało to być coś w rodzaju próby kontrolnej,
7.**Kolumna 0-1** - Najlepszy ze sposobów ,który choć traci nieco danych prowadzi do najlepszych wyników.