10 KROKÓW DO LEPSZEGO ZROZUMIENIA DANYCH

Mateusz Grzyb

MATEUSZGRZYB.PL

Wprowadzenie

Niezależnie od tego, czy pracujesz jako *data scientist*, analityk danych, czy statystyk, **zrozumienie danych jest kluczem do sukcesu**. Nie ma większego znaczenia, jaki jest szczegółowy cel zadania, które wykonujesz.

Jeśli Twoim celem jest wnioskowanie parametryczne, to **musisz najpierw się nieco „pobrudzić" danymi**. Jeśli budujesz model uczenia maszynowego, to będzie on tak dobry, jak dane, którymi go zasilisz.

Co więcej, odpowiednie zrozumienie danych przydaje się również na etapach pośrednich:

- Zanim przejdziesz do **czyszczenia danych**, musisz mieć informacje na temat brakujących wartości.
- Nim rozpoczniesz **transformację danych**, powinieneś wiedzieć jakiego typu zmienne występują w zbiorze.
- Przed rozpoczęciem **procesu modelowania**, musisz sprawdzić, czy w zbiorze znajdują się obserwacje odstające i zmienne o nietypowych rozkładach.

Całość można zatem opisać za pomocą dwóch reguł:

1. Lepsze zrozumienie danych → lepsze przygotowanie danych.
2. Lepsze przygotowanie danych → lepsze rezultaty, które osiągniesz.

Skąd pomysł na 10 kroków?

Zdobywając doświadczenie na kolejnych projektach, zacząłem zauważać dwie rzeczy:

1. Istnieje pewien **zestaw powtarzających się kroków**, które wykonuję każdorazowo, zaczynając pracę z nowym zbiorem danych.
2. Zestaw ten stanowi (w myśl zasady Pareto) **20% czynności, które przynoszą 80% efektów** w eksploracyjnej analizie danych.

Jestem wielkim fanem automatyzacji i uwielbiam budować *check-listy*. Postanowiłem zatem spisać swoje wnioski w postaci listy składającej się 10 kroków, które w mojej ocenie wnoszą największą wartość na etapie wczesnej analizy danych.

Większość z opisywanych przeze mnie technik pochodzi z działu statystyki, który nazywany jest **statystyką opisową** (ang. *descriptive statistics*). Techniki te są codziennie wykorzystywane przez specjalistów na całym świecie, co samo w sobie może świadczy o ich wartości i użyteczności.

Zanim przejdziesz do dalszej części dokumentu, pamiętaj proszę, że mechaniczne wykonywanie poszczególnych kroków, bez jednoczesnego zrozumienia procesu, w którym dane zostały wygenerowane, nie przyniesie Ci oczekiwanych rezultatów.

Wyniki, które uzyskasz z pomocą tej listy „10 kroków..." traktuj zatem jako materiał pomocniczy, który ułatwi Ci zrozumienia danych i zapewni wsad do procesów: transformacji danych i modelowania.

Krok 1. Poznaj rozmiar analizowanego zbioru

Co zrobić?

Pierwszym i absolutnie podstawowym krokiem jest poznanie rozmiaru zbioru, który będziesz analizować. Niezależnie od tego, gdzie znajdują się dane, sprawdź zarówno liczbę obserwacji, jak i liczbę zmiennych, które je opisują.

Dlaczego?

- Podejmiesz decyzję m.in. co do narzędzi i sprzętu, który wykorzystasz przy dalszej analizie.
- Oszacujesz czasochłonność procesu analizy.

Przykład w Python:

```
1. df.shape
```

Przykład w R:

```
1. dim(df)
```

Krok 2. Rzuć okiem na dane

Co zrobić?

Znając wielkość zbioru, możesz wyświetlić na ekranie swojego komputera przynajmniej próbkę „surowych" danych. Chyba nic nie buduje w taki sposób wyobrażenia o zbiorze jak podgląd danych w postaci tabelarycznej.

Dlaczego?

- Poznasz strukturę analizowanego zbioru.
- Ocenisz wstępną przydatność poszczególnych zmiennych w procesie modelowania.

Przykład w Python:

```
1. df.head(20)
```

Przykład w R:

```
1. head(df, 20)
```

Krok 3. Zweryfikuj typy poszczególnych zmiennych

Co zrobić?

Poprzedni krok dał Ci ogólne spojrzenie na to, z jakimi zmiennymi będziesz pracować. Podgląd pierwszy kilkudziesięciu obserwacji, nie musi oznaczać, że cały zbiór ma identyczną strukturę. Dlatego teraz upewnij się co do typów poszczególnych zmiennych.

W językach programowania takich jak R, czy Python jest to możliwe z użyciem dosłownie jednej linijki kodu. Jeśli korzystasz z SQL-a, to możesz podejrzeć po prostu strukturę tabeli (ew. tabel).

Dlaczego?

- Zweryfikujesz jakiego typu zmienne występują w zbiorze (zmienne całkowite, zmiennoprzecinkowe, kategoryczne porządkowe, kategoryczne nominalne, zmienne typu logicznego, daty).
- Znając typy zmiennych, będziesz wiedzieć, jak podejść do ich ewentualnej transformacji.

Przykład w Python:

```
1. df.dtypes
```

Przykład w R:

```
1. sapply(df, typeof)
```

Krok 4. Zbuduj podstawowe podsumowanie zbioru

Co zrobić?

Zbuduj podsumowanie zmiennych opisujących zbiór, w postaci jednej tabelki. Posłuży Ci ono jako 'wsad' do dalszej analizy. Podsumowanie powinno zawierać podstawowe informacje o zmiennych numerycznych, takie jak:

- Wartości minimalne.
- Wartości maksymalne.
- Średnia.
- Mediana.
- Drugi (dolny) kwartyl.
- Trzeci (górny) kwartyl.
- Odchylenie standardowe.

Dlaczego?

- Podsumowanie to będzie to punktem wejściowym do dalszej analizy.
- Dzięki niemu będziesz wiedzieć, na które zmienne należy mieć oko w dalszych krokach.

Przykład w Python:

```
1. df.describe()
```

Przykład w R:

```
1. summary(df)
2. df.numeric <- df[,sapply(df, is.numeric)]
3. sapply(df.numeric, sd)
```

Krok 5. Sprawdź czy w zbiorze występują braki danych

Co zrobić?

Zbuduj jeszcze jedno podsumowanie. Tym razem skup się na poszukiwaniu brakujących wartości w zbiorze. Sprawdź jakie zmienne zawierają braki i jaka jest ich liczba. Zastanów się również z czego mogą one wynikać. Pamiętaj przy tym, że w niektórych przypadkach braki danych również mogą być wartościową informacją.

Dlaczego?

- Niektóre algorytmy są wrażliwe na brakujące wartości. Przed ich użyciem konieczne będzie usunięcie obserwacji zawierających braki lub przeprowadzenie procesu imputacji.
- Wiedząc ile braków zawiera dana zmienna, podejmiesz decyzję, czy włączyć ją do modelu. Zmienne zawierające zbyt dużo braków nie będą podlegać imputacji.

Przykład w Python:

```
1. nulls_summary = pd.DataFrame(df.isnull().any(), columns=['Nulls'])
2. nulls_summary['Num_of_nulls [qty]'] = pd.DataFrame(df.isnull().sum())
3. nulls_summary['Num_of_nulls [%]'] = round((df.isnull().mean()*100),2)
4. print(nulls_summary)
```

Przykład w R:

```
1. nulls.summary <- data.frame("is.null" = apply(df, 2, function(x) any(is.na(x))),
2.                             "num.Of.nulls" = colSums(is.na(df)),
3.                             "per.of.nulls" = colMeans(is.na(df))*100)
```

Krok 6. Zbadaj skośność rozkładów

Co zrobić?

Najprostsza reguła dotycząca rozpoznawania skośności mówi o różnicy pomiędzy średnią a medianą. Daje to jednak jedynie ogólne spojrzenie, dlatego warto wyznaczyć współczynniki skośności poszczególnych zmiennych numerycznych.

Jeśli nie korzystasz z języka programowania używanego w analizie danych (np. R lub Python) i nie masz dostępu do specjalistycznego oprogramowania, to w możesz samodzielnie wyznaczyć współczynniki skośności w oparciu o informacje, które uzyskałeś w poprzednich krokach. Współczynnik skośności można obliczyć za pomocą prostego wzoru:

$$współczynnik\ skośności = 3 * \frac{średnia - mediana}{odchylenie\ standardowe}$$

Uzyskane wyniki interpretuj w sposób następujący:

- Współczynnik o wartości 0, to rozkład symetryczny.
- Współczynnik o wartości ujemnej to **rozkład lewostronnie skośny** (wydłużone lewe ramię rozkładu; średnia mniejsza od mediany).

- Współczynnik o wartości dodatniej to **rozkład prawostronnie skośny** (wydłużone prawe ramię rozkładu; średniej większa od mediany).

Dlaczego?

- Wysnujesz pierwsze wnioski na temat rozkładu. Nie zawsze jest możliwe narysowanie histogramu dla wszystkich obserwacji i wszystkich zmiennych. Obliczenie współczynnika skośności to proces o mniejszej złożoności obliczeniowej.

Przykład w Python:

```
1.  df.skew()
```

Przykład w R:

```
1.  library(e1071)
2.  df.numeric <- df[,sapply(df, is.numeric)]
3.  sapply(df.numeric, skewness)
```

Krok 7. Sprawdź rozkład zmiennych numerycznych

Co zrobić?

W poprzednich krokach sprawdziłeś wartości poszczególnych kwartyli i skośności. Dodaj teraz do tego warstwę wizualizacyjną. Dla każdej zmiennej numerycznej, z którą będziesz pracować, narysuj histogram i spróbuj rozpoznać rozkład. Jeśli masz wątpliwości, to możesz dodatkowo zastosować test statystyczny, by upewnić się, że dany rozkład jest np. rozkładem normalnym.

Dlaczego?

- Wnioski z tego punktu, będziesz mógł użyć, np. w procesie imputacji zmiennych numerycznych.
- Niektóre techniki statystyczne mają założenia, co do rozkładu zmiennych (np. w korelacji Pearsona wskazane jest, by zmienne miały rozkład normalny).

Przykład w Python:

```
1.  df.hist(bins=12) # histogram dla wszystkich zmiennych
2.  from scipy import stats  # test na normalność rozkładu
3.  df.select_dtypes([float, int]).apply(stats.normaltest) # p-value to wartość
```

Przykład w R:

```
1.  df.numeric <- df[,sapply(df, is.numeric)]
2.  sapply(df.numeric, hist)
```

Krok 8. Zidentyfikuj obserwacje odstające

Co zrobić?

Zidentyfikuj '*outliery*'. Najprostszym sposobem jest narysowanie wykresu pudełkowego dla zmiennych numerycznych. W większości narzędzi obserwacje odstające będą zaznaczane na wykresie jako kropki.

Zgodnie z regułą, obserwacją odstającą jest obserwacja, która przyjmuje wartość:

- Większą niż półtora odstępu międzykwartylowego (Q3 – Q1), od trzeciego kwartyla (Q3).
- Mniejszą niż półtora odstępu międzykwartylowego (Q3 – Q1), od pierwszego kwartyla (Q1).

Dlaczego?

- Niektóre algorytmy są wrażliwe na występowanie obserwacji odstających. Zanim przejdziesz do modelowania z użyciem, np. regresji liniowej, powinieneś się pozbyć obserwacji nietypowych.
- Niektóre metody stosowane w statystyce (np. korelacja Pearsona), są wrażliwe na występowanie obserwacji odstających. Jeśli nie usuniesz obserwacji odstających ze zbioru, to wyniki, które zaobserwujesz, mogą być przekłamane.

Przykład w Python:

```
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
iqr = q3-q1

low_boundary = (q1 - 1.5 * iqr)
upp_boundary  = (q3 + 1.5 * iqr)
num_of_outliers_L = (df[iqr.index] < low_boundary).sum()
num_of_outliers_U = (df[iqr.index] > upp_boundary).sum()

outliers = pd.DataFrame({'low_boundary':low_boundary, 'upp_boundary':upp_boundary,'
num_of_outliers_L':num_of_out_L, 'num_of_outliers_U':num_of_out_U})
print(outliers)
```

Przykład w R:

```
outlr = function(x){
  lower.boundry <- quantile(x, 0.25) - IQR(x) * 1.5
  upper.boundry <- quantile(x, 0.75) + IQR(x) * 1.5
  num.of.outliers.u <- sum(x>upper.boundry)
  num.of.outliers.l <- sum(x<lower.boundry)
  return(data.frame(lower.boundry, upper.boundry, num.of.outliers.l, num.of.outlier
s.u))
}
df.numeric <- df[,sapply(df, is.numeric)]
outliers.summary <- data.frame(sapply(df.numeric, outlr))
print(outliers.summary)
```

Krok 9. Sprawdź liczności zmiennych kategorycznych

Co zrobić?

Do tej pory skupiałeś się na zmiennych numerycznych. Przejdź teraz do zmiennych kategorycznych i odpowiedz na kilka pytań:

1. Ile jest zmiennych kategorycznych?
2. Ile kategorii wchodzi w skład poszczególnych zmiennych?
3. Jaka jest liczność kategorii w poszczególnych zmiennych kategorycznych (pokrycie zbioru przez daną kategorię)?
4. Jakie jest procentowe pokrycie zbioru przez poszczególne kategorie.

Dlaczego?

- Jeśli zmienna celu jest zmienną kategoryczną (ew. logiczną), to po sprawdzeniu liczności poszczególnych kategorii będziesz wiedzieć, czy zbiór jest odpowiednio zbalansowany (posiada względnie równy podział wg wystąpień danej wartości zmiennej celu).
- Może się okazać, że któraś ze zmiennych posiada 40 kategorii. Część algorytmów może sobie nie poradzić z taką liczbą kategorii. Znając „pokrycie zbioru" przez kategorie, możesz dojść do wniosku, że 5% wszystkich kategorii pokrywa 95% zbioru. W takim przypadku będziesz mógł rozważyć ujęcie 95% kategorii pod nazwą „*Other*" i skupienie się na tych najistotniejszych.

Przykład w Python:

```
for col in df.select_dtypes(['object', 'category']):
    print(df[col].value_counts())
```

Przykład w R:

```
sapply(df[,sapply(df, is.factor)], table)
```

Krok 10. Zbadaj zależności pomiędzy zmiennymi

Co zrobić?

W tym kroku zweryfikuj poziomy współczynników:

- Korelacji pomiędzy zmiennymi numerycznymi (współczynnik korelacji rang Spearmana lub współczynnik Pearsona).
- Zależności pomiędzy zmiennymi kategorycznymi (współczynnik V Cramméra).
- Zależności pomiędzy zmiennymi kategorycznymi i numerycznymi (współczynnik R modelu liniowego z jedną zmienną kategoryczną, która objaśnia zmienną numeryczną).

Dlaczego?

- Krok ten pozwoli Ci odkryć związki pomiędzy poszczególnymi zmiennymi. Informacje te będziesz mógł użyć, np. na etapie transformacji zmiennych.

- Na podstawie analizy korelacji możesz decydować o wyborze zmiennych do modelu (szczególnie istotne w modelach liniowych).

Przykład w Python:

```
# Przypadek A - nie występują wartosci odstające, rozkład normalny.
np.corrcoef(df.select_dtypes(['float', 'int']), rowvar=0)
# Przypadek B - mogą występować obserwacje odstające, dowolny rozklad.
stats.spearmanr(df.select_dtypes(['float', 'int']))[0]
```

Przykład w R:

```
df.numeric <- df[,sapply(df, is.numeric)]
# Przypadek A - nie występują wartości odstające, rozkład normalny.
cor(df.numeric, method = 'pearson')
# Przypadek B - mogą występować obserwacje odstające, dowolny rozklad.
cor(df.numeric, method = 'spearman')
```

Podsumowanie

Mam nadzieję, że opisane przeze mnie kroki okażą się dla Ciebie przydatne i będziesz z nich regularnie korzystać przy wykonywanych przez Ciebie projektach :)

Jeśli spodobała Ci się lista i opisane w niej wskazówki, to zapraszam Cię na mojego bloga. Znajdziesz na nim więcej wartościowych wskazówek i porad. Będę również bardzo wdzięczny, jeśli podzielisz się linkiem ze swoimi znajomymi na Facebook, Twitter lub LinkedIn.