Aleksandra Biedrzycka
pd3817@pjwstk.edu.pl

Projekt końcowy
Studia podyplomowe PJATK Big Data
Predykcja cen samochodów BMW na podstawie ogłoszeń motoryzacyjnych

Link do projektu:
https://github.com/Aleksandra-Biedrzycka/Cars_prices_predictions

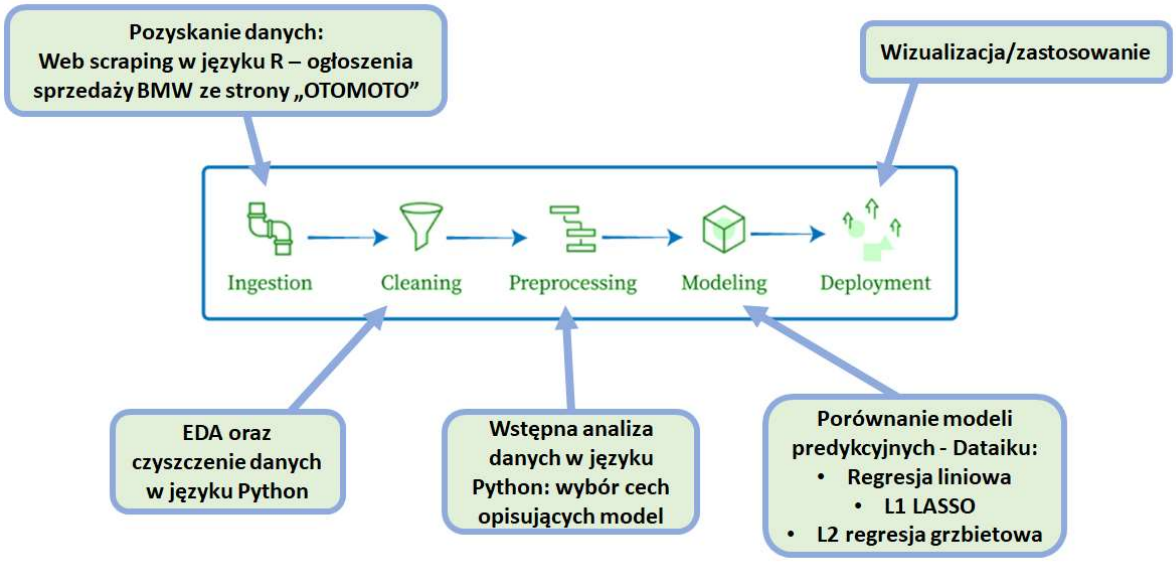
Spis treści

1. Wstęp

Projekt miał na celu zbudowanie pipeline'u z użyciem dowolnych narzędzi. W projekcie został podjęty temat prognozowania cen za pomocą modeli regresji liniowej.

Pipeline składał się z następujących etapów przedstawianych na rys 1:

- Pozyskanie danych – web scraping w języku R.
  Skrypt napisany w języku R znajduje się w załączniku A.
- EDA, czyszczenie, imputacja i export danych w języku Python.
- Wstępna analiza danych z decyją o wyborze cech opisujących model w języku Python.
  Skrypt napisany w języku Python znajduje się w załączniku B.
- Modele predykcyjne regresji liniowej wykonane w Dataiku.
- Wizualizacja wyniku, ewaluacja modeli.



1 Pipeline.

2. Pozyskiwanie danych - web scraping (język R)

Dane zostały pozyskane pozyskane metodą Web Scrapingu ze strony z ogłoszeniami motoryzacyjnymi „OTOMOTO".

Analizowane były ogłoszenia o sprzedaży samochodów marki BMW.

W pierwszej kolejności strony w formacie .html zostały zescrapowane. Następnie wyniki zostały zapisanie do ramki danych i wyeksportowane do pliku .csv.

Zbiór danych składał się 688 rekordów oraz z cech: wiek auta, rok produkcji, pojemność silnika, przebieg oraz kolumny zmiennej celu jaką była cena.

Skrypt napisany w języku R znajduje się w załączniku A.

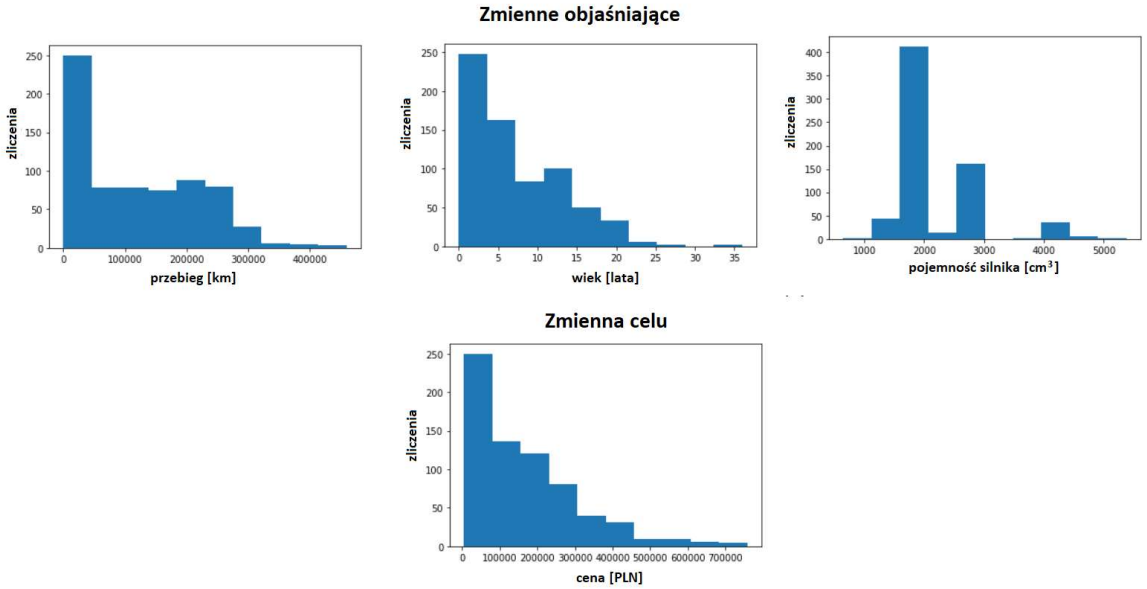
3. EDA (język Python)

Plik wyeksportowany w języku R został następnie wyeksportowany do ramki danych w języku Python tam została podjęta analiza EDA (Explanatory Data analysis) oraz przygotowanie danych do dalszej analizy.

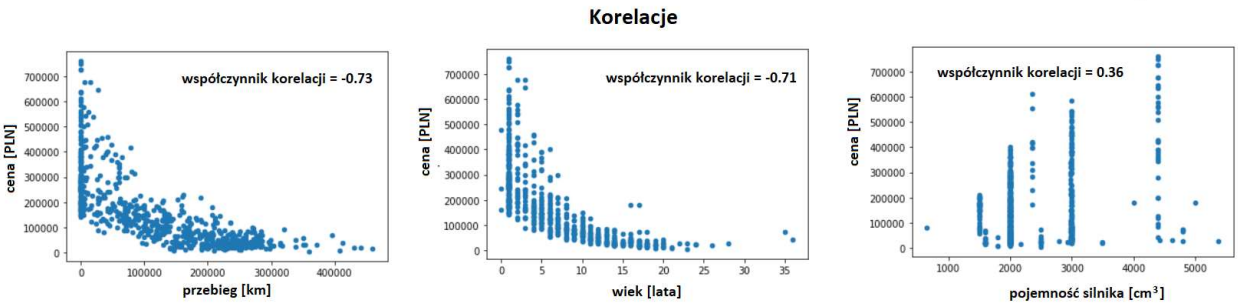
3.1. EDA – eksploracyjna analiza danych.

Podczas analizy EDA wykonane zostały m.in.:

- sprawdzenie typów danych, braków danych
- statystyki dotyczące danych
- histogramy,
- obliczenie współczynnika korelacji



2 Histogramy częstości występowań danych wartości (zmienna celu: cena).

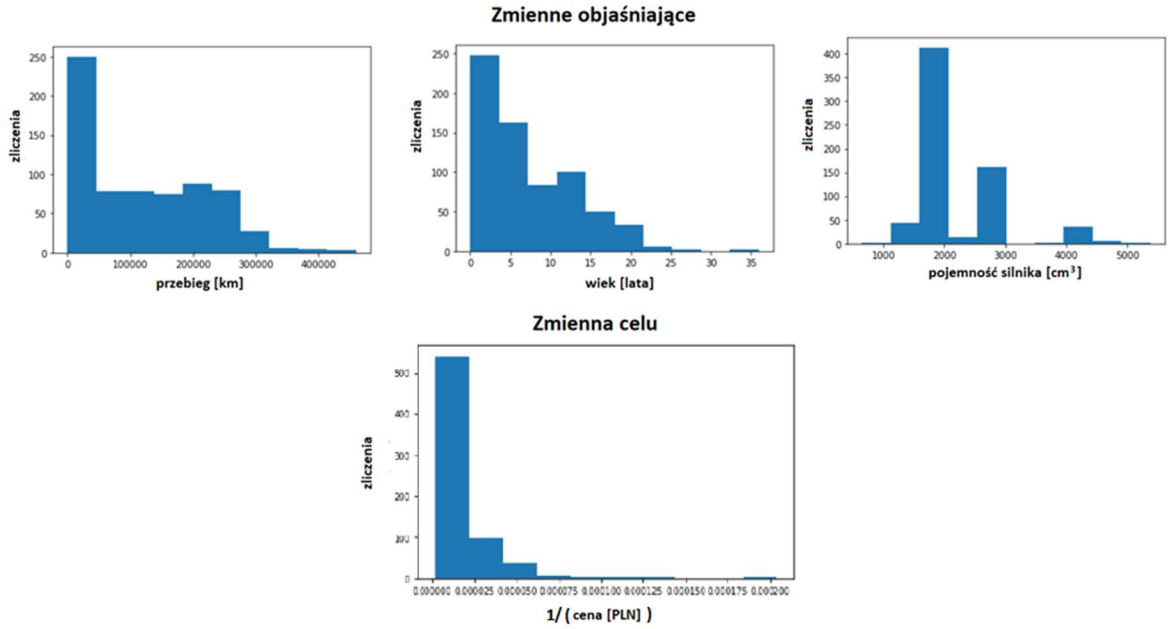


3 Korelacje pomiędzy zmiennymi objaśniającymi a zmienną celu (zmienna celu: cena).

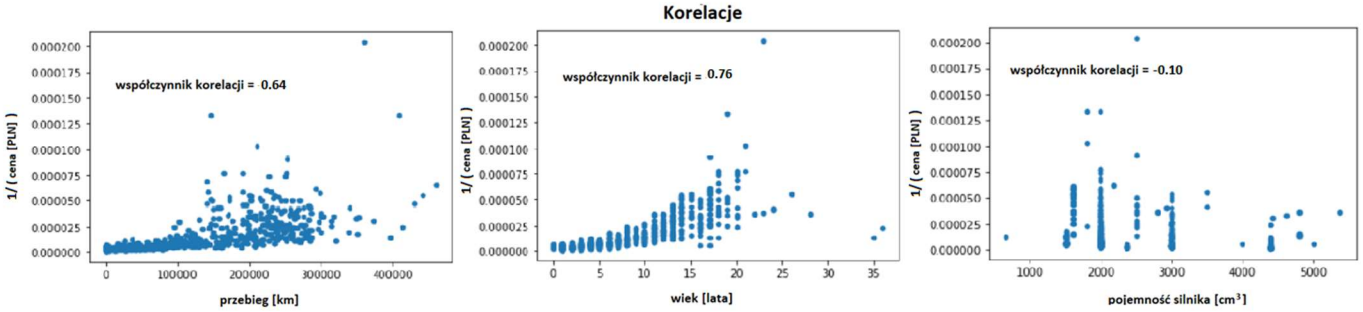
Z wykresów znajdujących się na rysunkach: 2 oraz 3 wynika, że najsilniej skorelowany z ceną jest przebieg samochodu.

3.2. Feature engineering

Obserwując wykresy na rysunku 3 można dokonać niewielkiego feature engineering (inżynieria cech) i przekształcić zmienną celu z ceny na jej odwrotność (1/cena) – następnie ponownie przeanalizować wykresy i histogramy.



4 Histogramy częstości występowań danych wartości (zmienna celu: 1/cena).



5 Korelacje pomiędzy zmiennymi objaśniającymi a zmienną celu (zmienna celu: 1/cena).

Z wykresów znajdujących się na rysunkach: 4 oraz 5 wynika, że najsilniej skorelowany z ceną jest wiek samochodu.

3.3. Przygotowanie danych.

Brakujące dane występowały jedynie w kolumnie dotyczącej pojemności silnika. Braki zostały uzupełnione poprzez imputację średniej wartości.

Została usunięta kolumna dotycząca roku produkcji ze względu na dużą korelację z kolumną dotyczącą wieku pojazdu.

4. Przewidywanie cen - budowa modelu

4.1. Modelowanie – porównanie modeli regresji liniowej

Dane wstępnie przygotowana w języku Python zostały zaimportowane do Dataiku, gdzie jako zmienna celu wskazana została cena BMW.

Ten element pipelinu znajduje się pod linkiem
http://dataiku.pjwstk.edu.pl:11000/projects/CARSPRICES/

Parametry modeli zostały przyjęte poprzez auto-dopasowanie w narzędziu Dataiku.

Stosunek modelów treningowego i testowego wynosił 0.8 we wszystkich 3 modelach.

Regresja liniowa

Dopasowanie liniowe wyraża się wzorem:

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n$$

gdzie:
y – zmienna objaśniana (zależna)
$x_1, x_2, \dots, x_n$ – zmienne objaśniające (niezależne)
$a_0, a_1, a_2, \dots, a_n$ – współczynniki regresji liniowej

Celem regresji liniowej jest dopasowanie linii regresji do zaobserwowanego zbioru danych empirycznych, a więc odpowiednie dopasowanie współczynników $a_0, a_1, a_2, \dots, a_n$.
Model jest tym lepiej dopasowany im mniejsza jest odległość wartości teoretycznych $\hat{y}$ od wartości zaobserwowanych y dla zmiennej zależnej [4].

Zostały porównane 3 modele regresji liniowej, różniące się funkcją strat podlegającą minimalizacji.

Regresja liniowa klasyczna – metoda najmniejszych kwadratów

Minimalizowana funkcja strat wyraża się wzorem:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

gdzie:
y_i – i-ta wartość rzeczywista
$\hat{y}_i$ – i-ta wartość teoretyczna (z predykcji)

Regresja liniowa LASSO (L1)
Minimalizowana funkcja strat wyraża się wzorem:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{n} |a_i|$$

Regresja liniowa grzbietowa (L2)
Minimalizowana funkcja strat wyraża się wzorem:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{n} (a_i)^2$$

gdzie:
y_i – i-ta wartość rzeczywista
$\hat{y}_i$ – i-ta wartość teoretyczna (z predykcji)
a_i – współczynniki regresji liniowej
λ – parametr regularyzacji

4.2. Ewaluacja modeli.

Głównym miernikiem jakości modelu jest wartość R^2 opisana wzorem:

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \geq 0$$

gdzie:
y_i – i-ta wartość rzeczywista
$\hat{y}_i$ – i-ta wartość teoretyczna (z predykcji)
$\bar{y}$ – średnia wartość rzeczywista

Dopasowanie modelu jest tym lepsze, im wartość R^2 jest bliższa jedności.

Współczynnik determinacji R^2, inaczej zwany współczynnikiem określoności lub R-kwadrat jest miarą tego, jaki procent zmienności zmiennej zależnej (objaśnianej) jest wyjaśniany za pomocą zmiennej niezależnej (czynnik zmienna objaśniająca, predyktor) bądź modelu statystycznego. Innymi słowy, współczynnik determinacji informuje nas, ile nasz model, nasz badany czynnik wyjaśnia zgromadzone dane pomiarowe (zmienną zależną) [3].

Dopasowanie modelu jest tym lepsze, im wartość R^2 jest bliższa jedności.

Dla zmiennej celu: cena, według tego miernika najbardziej w przypadku naszych danych i auto-dopasowanych przez Dataiku parametrów sprawdziła się klasyczna regresja liniowa. Wyniki znajdują się w tabeli 1.

1 Ewaluacja modeli dla zmiennej celu: cena.

Model Evaluation	metrics
Name	R2 Score
Ordinary Least Squares (predictions)	0.722
Lasso (L1) regression (predictions)	0.720
Ridge (L2) regression (predictions)	0.708

Dla zmiennej celu: 1/cena, według miernika R^2 najbardziej (nieznacznie) w przypadku naszych danych i auto-dopasowanych przez Dataiku parametrów sprawdziła się regresja L1. Wyniki znajdują się w tabeli 2.

2 Ewaluacja modeli dla zmiennej celu: 1/cena.

Model Evaluation	metrics
Name	R2 Score
Lasso (L1) regression (predictions)	0.619
Ordinary Least Squares (predictions)	0.618
Ridge (L2) regression (predictions)	0.584

5. Podsumowanie i konkluzje

Celem projektu było wykonanie pipeline'u w oparciu o dane dotyczące ogłoszeń sprzedaży samochodów marki BMW, zescrapowane ze strony internetowej „OTOMOTO".

Zadanie zostało wykonane w języku R (web-scraping), języku Python (EDA oraz przygotowanie danych) oraz w narzędziu Dataiku (uczenie maszynowe: modele predykcji regresji liniowej).

Pierwszym ważnym wnioskiem było stwierdzenie, że ceny pojazdów najsilniej zależą od ich przebiegów (na podstawie współczynnika korelacji w przypadku zmiennej celu: cena). Dla zmiennej celu 1/cena najbardziej skorelowany był wiek pojazdu.

Drugi wniosek wynika z porównania modeli regresji liniowej. Tu najskuteczniejszy (przy auto-dopasowaniu parametrów) okazał się klasyczny model regresji liniowej – w przypadku zmiennej celu: cena i (nieznacznie) model L1 dla zmiennej celu 1/cena. Kryterium w ewaluacji modeli był współczynnik R^2.

W przyszłych pracach nad podobnym projektem należałoby wykonać usprawnienie modeli (dopasowywanie ich parametrów) a także spróbować bardziej zaawansowanych metod uczenia maszynowego takich jak np. sieci neuronowe.

6. Bibliografia

[1] https://academy.dataiku.com/machine-learning-basics

[2] https://academy.dataiku.com/

[3] https://www.naukowiec.org/wiedza/statystyka/wspolczynnik-determinacji_736.html

[4] Paweł Strawiński „Notatki do ćwiczeń z ekonometrii”

Załącznik A: „Skrypt w języku R – web scraping”

```
##############################################################################################
###

## Element pipeline'u do projektu "Projekt własny- predykcja cen BMW:
## Pozyskanie danych metodą webscrapin

##############################################################################################
###

library(rvest)
library(stringr)
library(progress)
library(ggplot2)
library(dplyr)

# Scraping
N = 500
link = 'https://www.otomoto.pl/osobowe/bmw?page='
for(i in 1:N){
  download.file(paste0(link, i), destfile = paste0("scrapedpage-", i,".html"), quiet=TRUE)
  page <- read_html(paste0("scrapedpage-", i,".html"))
}

# Data
N = 250
results <- data.frame('price'=integer(), 'year'=integer(), 'mileage'=integer(), 'engine'=integer())
pb <- progress_bar$new(total=N)
for(i in 1:N){
  page <- read_html(paste0("scrapedpage-", i,".html"))

  price <- page %>% html_nodes(xpath = '//span[@class="ooa-1bmnxg7 e1b25f6f11"]') %>%
html_text()
  eur_index <- str_detect(price, "EUR")
  price <- as.integer(str_replace_all(price, " |PLN|EUR", ""))
  price[eur_index] <- price[eur_index]*4.7
  print(price)

  year <- page %>% html_nodes(xpath = '//article/div/div/ul[1]/li[1]') %>% html_text() %>%
str_replace(., " ", "") %>% as.integer()

  mileage <- page %>% html_nodes(xpath = '//article/div/div/ul[1]/li[2]') %>% html_text() %>%
str_replace_all(., " |km", "") %>% as.integer()

  engine <- page %>% html_nodes(xpath = '//article/div/div/ul[1]/li[3]') %>% html_text() %>%
str_replace_all(., " |cm3", "") %>% as.integer()
```

```
  if(length(price) == 0) price <- NA
  if(length(year) == 0) year <- NA
  if(length(mileage) == 0) mileage <- NA
  if(length(engine) == 0) engine <- NA

  print(price)
  print(year)

  results <- rbind(results, data.frame('price'=price, 'year'=year, 'mileage'=mileage, 'engine'=engine))
  print(dim(results))
  pb$tick()
}
summary(results)
hist(results$year)

results$age <- 2023-results$year

write.csv(results, "bmw.csv")

#opcionalne analizy wstępne:

#results <- read.csv("bmw.csv")

plot(x=results$year, y=results$price, pch=19)
summary(lm(price ~ year, data=results))
abline(lm(price ~ year, data=results))

lm(price ~ age, data=results)
plot(x=results$age, y=results$price, pch=19, xlab="Car age", ylab="Price")
abline(lm(price ~ age, data=results), lwd=2, col="dark red")


fit_p2 <- lm(price ~ age + I(age^2), data=results)
summary(fit_p2)
points(x = 0:50, y=predict(fit_p2, data.frame('age'=0:50)), lwd=2, col="pink", type='l')

fit_p3 <- lm(price ~ age + I(age^2) + I(age^3), data=results)
AIC(fit_p3)
summary(fit_p3)
points(x = 0:50, y=predict(fit_p3, data.frame('age'=0:50)), lwd=2, col="navy blue", type='l')

fit_log <- lm(price ~ age + I(log(age+1)), data=results)
AIC(fit_log)
summary(fit_log)
points(x = 0:50, y=predict(fit_log, data.frame('age'=0:50)), lwd=2, col="darkgoldenrod1", type='l')
```

Załącznik B: „Skrypt w języku Python – EDA”

```
# -*- coding: utf-8 -*-
"""bmw.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1k0YLOhI5cj0_YD91wwOjlUAORlMgB7YN

---

# **Elementy pipeline'u do projektu "Projekt własny- predykcja cen BMW:**

*   **EDA (exploratory data analysis) oraz czyszczenie danych**

*   **wstępne przetwarzanie danych**

---
"""

from google.colab import drive
drive.mount('/content/drive')

#import bibliotek

import numpy as np
import pandas as pd
import scipy.stats
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import os

#wczytanie pliku do ramki danych

bmw=pd.read_csv('/content/drive/MyDrive/projekt_wlasny/bmw.csv')

#podgląd pliku

bmw

#typy danych: mamy tylko liczbowe

bmw.dtypes

#opis pliku (w tak wykonanym poleceniu zostaną opisane tylko dane liczbowe,
ale w trym przypadku tylko takie mamy)
```

```
bmw.describe()

"""
Funkcja describe() wykonana osobno dla każdej zmiennej numerycznej pokazuje
kolejno:

* count -zliczenia
* mean - średnią
* std -odchylenie standardowe
* min -minimum
* 25% -dolny kwartyl
* 50% -mediana
* 75% -górny kwadrtyl
* max -maksimum

"""

#braki danych

bmw.columns[bmw.isna().sum(axis=0) > 0]

#histogramy

bmw.price.plot.hist()

bmw.year.plot.hist()

bmw.mileage.plot.hist()

bmw.engine.plot.hist()

bmw.age.plot.hist()

"""# Kurtoza i skośność

*(źródło: https://www.ibm.com/docs/pl/spss-statistics/SaaS?topic=descriptives-
options)*

**Rozkład:** Kurtoza i skośność są to statystyki charakteryzujące kształt i
symetrię rozkładu. Te statystyki wyświetlane są wraz ze swoimi błędami
standardowymi.

**Kurtoza:** Miara zakresu, do którego występują wartości odstające. W
przypadku rozkładu normalnego wartość statystyki kurtozy wynosi zero. Kurtoza
dodatnia wskazuje, że w danych istnieje więcej dodatnich wartości odstających
niż w przypadku rozkładu normalnego. Kurtoza ujemna wskazuje, że w danych
istnieje mniej dodatnich wartości odstających niż w przypadku rozkładu
normalnego.
```

```
**Skośność:** Miara asymetrii rozkładu. Rozkład normalny jest symetryczny i ma skośność równą 0. Rozkład z dużą skośnością dodatnią ma długi ogon prawostronny. Gdy zaś współczynnik skośności jest ujemny, rozkład ma długi kraniec z lewej strony. Jako wytyczna, wartość skośności przekraczająca dwukrotnie swój błąd standardowy na ogół oznacza odstępstwo od symetrii rozkładu.

Zobaczmy co można powiedzieć o rozkładzie np. cen na podstawie kurtozy i skośności:
"""

bmw.price.plot.hist()

scipy.stats.kurtosis(bmw['price'])

"""Kurtoza jest dodatnia, zatem istnieje więcej dodatnich wartości odstających niż w przypadku rozkładu normalnego (wysokie ceny odstające)."""

scipy.stats.skew(bmw['price'])

"""Widać dodatnią skośność, zatem wykres posiada ogon prawostronny, co znajduje potwierdzenie na histogramie."""

#kasowanie niepotrzebnych danych: kolumny year powiązanej z kolumną 'age', oraz kolumny 'Unnamed: 0'

bmw.drop(['year', 'Unnamed: 0'], axis=1 ,inplace=True)

bmw

#zastąpienie brakujących danych numerycznych średnią

bmw.columns[bmw.isna().sum(axis=0) > 0]

imputer = SimpleImputer(strategy='mean', missing_values=np.nan)
imputer = imputer.fit(bmw[['engine']])
bmw['engine'] = imputer.transform(bmw[['engine']])

bmw.columns[bmw.isna().sum(axis=0) > 0]

#odstające dane (powyżej 98%)

print("98% samochodów ma cenę niższą niż {0:
.2f}".format(np.percentile(bmw.price, 98)))
price = bmw[(bmw.price <= np.percentile(bmw.price, 98)) & (bmw.price > 0)]

print("98% samochodów ma przebieg niższy niż {0:
.2f}".format(np.percentile(bmw.mileage, 98)))
```

```
mileage = bmw[(bmw.mileage <= np.percentile(bmw.mileage, 98)) & (bmw.mileage >
0)]

print("98% samochodów ma silnik mniejszy niż {0:
.2f}".format(np.percentile(bmw.engine, 98)))
engine = bmw[(bmw.engine <= np.percentile(bmw.engine, 98)) & (bmw.engine > 0)]

print("98% samochodów ma wiek niższy niż {0:
.2f}".format(np.percentile(bmw.age, 98)))
age = bmw[(bmw.age <= np.percentile(bmw.age, 98)) & (bmw.age > 0)]

#zależnosci pomiędzy cechami modelu w stosunku do ceny: macierze korelacji

"""Współczynnik korelacji ([-1,1]) określa, jaka jest zależność pomiędzy dwoma
zmiennymi. Im wyższa jest wartość bezwzględna tego współczynnika, tym zmienne
silniej od siebie zależą.

Sprawdźmy więc jak zależy cena samochodów od wieku, przebiegu i silnika.
"""

price_age = np.corrcoef(bmw['price'], bmw['age'])
price_age

price_mileage = np.corrcoef(bmw['price'], bmw['mileage'])
price_mileage

price_engine = np.corrcoef(bmw['price'], bmw['engine'])
price_engine

"""Cena najbardziej powiązana jest z przebiegiem, najmniej z silnikiem."""

bmw.plot.scatter(x='age', y='price')

bmw.plot.scatter(x='mileage', y='price')

bmw.plot.scatter(x='engine', y='price')

bmw

os.makedirs('folder/subfolder', exist_ok=True)
bmw.to_csv('/content/drive/MyDrive/projekt_wlasny/bmw_preproc.csv',
index=False)
```

Załącznik C: „Skrypt w języku Python – zmienna celu: 1/cena”

```
# -*- coding: utf-8 -*-
"""bmw_feature_engineering.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1l30iqW-kRiKDg7rflkb1DTcNvnXIu5KR

---

# **Feature engineering: predykcja odwrotności cen BMW**
# **Elementy pipeline'u do projektu "Projekt własny - predykcja cen BMW":**

*   **EDA (exploratory data analysis) oraz czyszczenie danych**

*   **wstępne przetwarzanie danych**

---
"""

from google.colab import drive
drive.mount('/content/drive')

#import bibliotek

import numpy as np
import pandas as pd
import scipy.stats
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import os

#wczytanie pliku do ramki danych

bmw=pd.read_csv('/content/drive/MyDrive/projekt_wlasny/bmw.csv')

#podgląd pliku

bmw

#zastępujemy cenę odwrotnością ceny
bmw.price = 1/bmw.price

bmw

#typy danych: mamy tylko liczbowe
```

```
bmw.dtypes

#opis pliku (w tak wykonanym poleceniu zostaną opisane tylko dane liczbowe,
ale w trym przypadku tylko takie mamy)

bmw.describe()

"""
Funkcja describe() wykonana osobno dla każdej zmiennej numerycznej pokazuje
kolejno:

* count -zliczenia
* mean - średnią
* std -odchylenie standardowe
* min -minimum
* 25% -dolny kwartyl
* 50% -mediana
* 75% -górny kwadrtyl
* max -maksimum

"""

#braki danych

bmw.columns[bmw.isna().sum(axis=0) > 0]

#histogramy

bmw.price.plot.hist(fontsize = 7.5)

bmw.year.plot.hist()

bmw.mileage.plot.hist()

bmw.engine.plot.hist()

bmw.age.plot.hist()

"""# Kurtoza i skośność

*(źródło: https://www.ibm.com/docs/pl/spss-statistics/SaaS?topic=descriptives-
options)*

**Rozkład:** Kurtoza i skośność są to statystyki charakteryzujące kształt i
symetrię rozkładu. Te statystyki wyświetlane są wraz ze swoimi błędami
standardowymi.
```

```
**Kurtoza:** Miara zakresu, do którego występują wartości odstające. W przypadku rozkładu normalnego wartość statystyki kurtozy wynosi zero. Kurtoza dodatnia wskazuje, że w danych istnieje więcej dodatnich wartości odstających niż w przypadku rozkładu normalnego. Kurtoza ujemna wskazuje, że w danych istnieje mniej dodatnich wartości odstających niż w przypadku rozkładu normalnego.

**Skośność:** Miara asymetrii rozkładu. Rozkład normalny jest symetryczny i ma skośność równą 0. Rozkład z dużą skośnością dodatnią ma długi ogon prawostronny. Gdy zaś współczynnik skośności jest ujemny, rozkład ma długi kraniec z lewej strony. Jako wytyczna, wartość skośności przekraczająca dwukrotnie swój błąd standardowy na ogół oznacza odstępstwo od symetrii rozkładu.

Zobaczmy co można powiedzieć o rozkładzie np. cen na podstawie kurtozy i skośności:
"""

bmw.price.plot.hist()

scipy.stats.kurtosis(bmw['price'])

"""Kurtoza jest dodatnia, zatem istnieje więcej dodatnich wartości odstających niż w przypadku rozkładu normalnego (wysokie ceny odstające)."""

scipy.stats.skew(bmw['price'])

"""Widać dodatnią skośność, zatem wykres posiada ogon prawostronny, co znajduje potwierdzenie na histogramie."""

#kasowanie niepotrzebnych danych: kolumny year powiązanej z kolumną 'age', oraz kolumny 'Unnamed: 0'

bmw.drop(['year', 'Unnamed: 0'], axis=1 ,inplace=True)

bmw

#zastąpienie brakujących danych numerycznych średnią

bmw.columns[bmw.isna().sum(axis=0) > 0]

imputer = SimpleImputer(strategy='mean', missing_values=np.nan)
imputer = imputer.fit(bmw[['engine']])
bmw['engine'] = imputer.transform(bmw[['engine']])

bmw.columns[bmw.isna().sum(axis=0) > 0]

#odstające dane (powyżej 98%)
```

```
print("98% samochodów ma przebieg niższy niż {0:
.2f}".format(np.percentile(bmw.mileage, 98)))
mileage = bmw[(bmw.mileage <= np.percentile(bmw.mileage, 98)) & (bmw.mileage >
0)]

print("98% samochodów ma silnik mniejszy niż {0:
.2f}".format(np.percentile(bmw.engine, 98)))
engine = bmw[(bmw.engine <= np.percentile(bmw.engine, 98)) & (bmw.engine > 0)]

print("98% samochodów ma wiek niższy niż {0:
.2f}".format(np.percentile(bmw.age, 98)))
age = bmw[(bmw.age <= np.percentile(bmw.age, 98)) & (bmw.age > 0)]

#zależnosci pomiędzy cechami modelu w stosunku do ceny: macierze korelacji

"""Współczynnik korelacji ([-1,1]) określa, jaka jest zależność pomiędzy dwoma
zmiennymi. Im wyższa jest wartość bezwzględna tego współczynnika, tym zmienne
silniej od siebie zależą.

Sprawdźmy więc jak zależy cena samochodów od wieku, przebiegu i silnika.
"""

price_age = np.corrcoef(bmw['price'], bmw['age'])
price_age

price_mileage = np.corrcoef(bmw['price'], bmw['mileage'])
price_mileage

price_engine = np.corrcoef(bmw['price'], bmw['engine'])
price_engine

"""Cena najbardziej powiązana jest z wiekiem, najmniej z silnikiem."""

bmw.plot.scatter(x='age', y='price')

bmw.plot.scatter(x='mileage', y='price')

bmw.plot.scatter(x='engine', y='price')

bmw

os.makedirs('folder/subfolder', exist_ok=True)
bmw.to_csv('/content/drive/MyDrive/projekt_wlasny/bmw_preproc_fe.csv',
index=False)
```