

Analiza statystyczna zbioru danych

Statystyka i Przetwarzanie Danych

Maciej Biegan, Kamil Dziedzic, Jan Bobak

14 maja 2025

Spis treści

1 Dokładny opis danych, oraz ich źródło

Dane pochodzą z witryny: <https://archive.ics.uci.edu/dataset/10/automobile>

Wybrany zbiór danych Automobile pochodzi z repozytorium UCI Machine Learning Repository i zawiera informacje o 205 samochodach, pochodzące z roku 1985.

1.1 Główne cechy zbioru

- Liczba instancji: 205
- Liczba atrybutów: 26

1.2 Przykładowe ciągłe atrybuty

- engine-size: pojemność silnika
- horsepower: moc silnika
- city-mpg / highway-mpg: zużycie paliwa w mieście i na autostradzie
- peak-rpm: maksymalna wartość obrotów
- stroke: skok cylindra
- curb-weight: waga samochodu z płynami
- bore: średnica tłoka
- price: cena pojazdu (w dolarach amerykańskich)

1.3 Kod inicjalizujący

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
import statsmodels.api as sm
from sklearn.utils import resample
from ucimlrepo import fetch_ucirepo

plt.style.use('seaborn-v0_8')
plt.rcParams['figure.figsize'] = (10, 6)
plt.rcParams['font.size'] = 12
sns.set_palette("viridis")

automobile = fetch_ucirepo(id=10)

X = automobile.data.features
y = automobile.data.targets
```

```

df = pd.concat([X, y], axis=1)

print(f"Nazwa zbioru danych: {automobile.metadata.name}")
print(f'Liczba instancji: {automobile.metadata.num_instances}')
print(f'Liczba atrybutów: {automobile.metadata.num_features}')

continuous_vars = [
    'normalized-losses', 'bore', 'stroke', 'horsepower', 'peak-
    rpm', 'price',
    'engine-size', 'curb-weight', 'height', 'width', 'length', ,
    'wheel-base',
    'compression-ratio', 'city-mpg', 'highway-mpg'
]

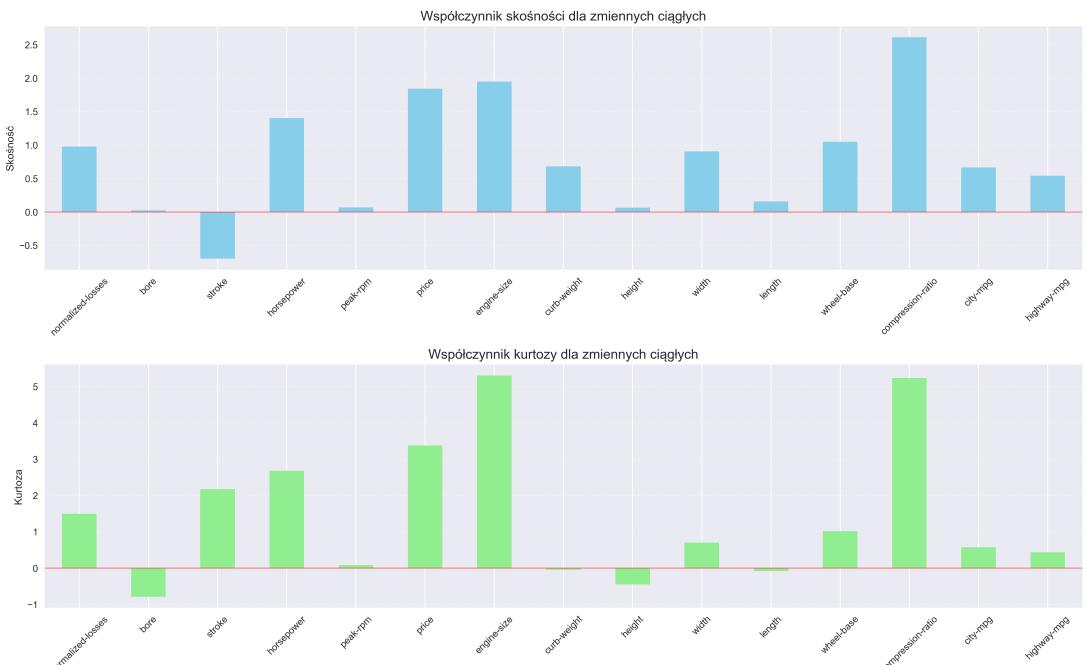
```

Listing 1: Import bibliotek i wczytywanie danych

2 Estymacja parametrów rozkładu (punktowa)

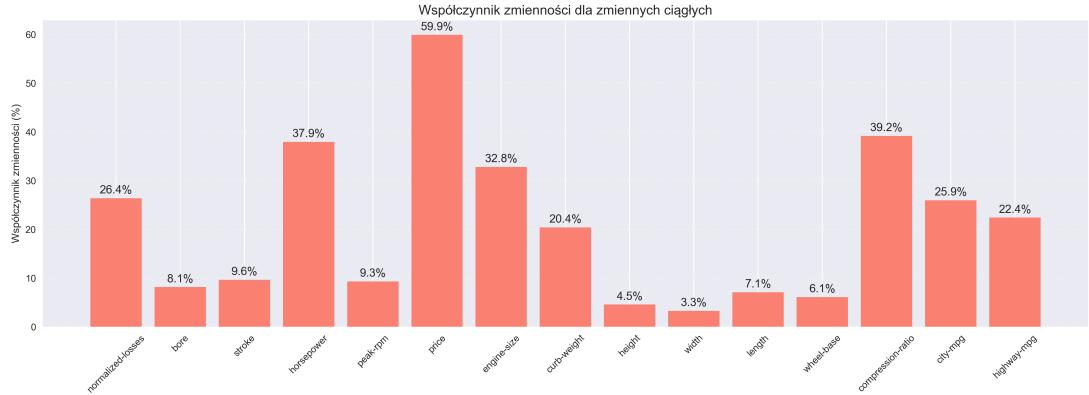
W tej sekcji przeanalizowano rozkłady zmiennych ciągłych w zbiorze danych poprzez obliczenie różnych miar statystycznych, takich jak:

- Miary tendencji centralnej: średnia, mediana, moda
- Miary rozproszenia: wariancja, odchylenie standardowe, odchylenie przeciętne
- Miary kształtu rozkładu: skosność, kurtoza
- Miary pozycyjne: kwantyle, IQR (rozstęp międzykwartylowy)



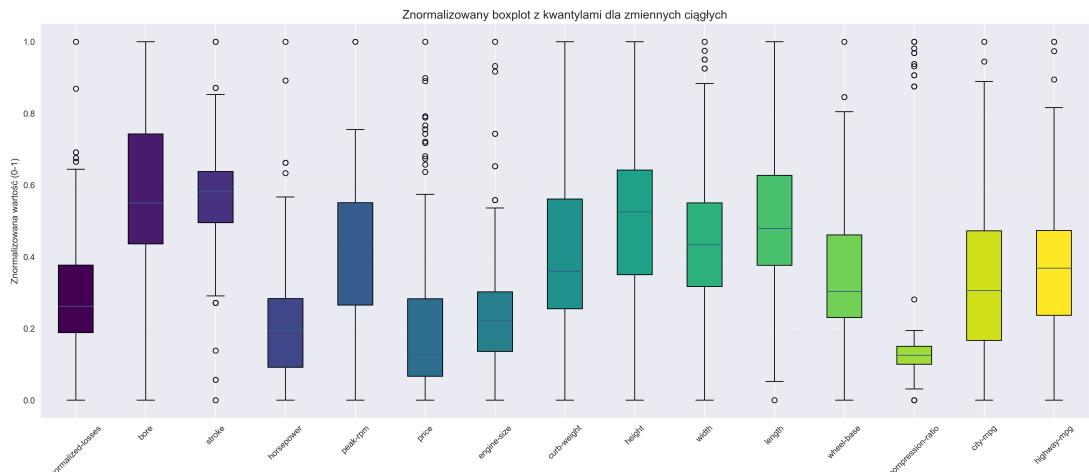
Rysunek 1: Współczynniki skosności i kurtozy dla analizowanych zmiennych

Na podstawie wykresu skosnosci (Rys. ??) można zauważyc, że większość zmiennych charakteryzuje się prawostronną asymetrią, co wskazuje na obecność wartości odstających po prawej stronie rozkładu. Szczególnie wysokie wartości skosnosci można zaobserwować dla zmiennych ‘price’ i ‘normalized-losses’.



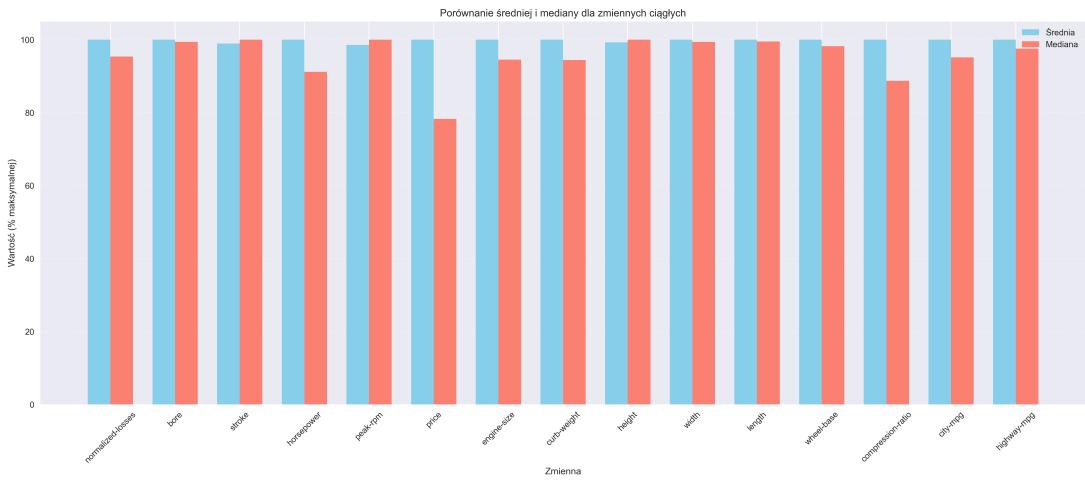
Rysunek 2: Współczynniki zmienności dla analizowanych zmiennych

Współczynnik zmienności (Rys. ??) pozwala ocenić stopień zróżnicowania danych względem średniej. Najwyższą zmiennością charakteryzują się zmienne ‘normalized-losses’ i ‘price’, co wskazuje na dużą różnorodność w tych atrybutach.



Rysunek 3: Znormalizowane boxploty dla zmiennych ciągkich

Znormalizowane boxploty (Rys. ??) umożliwiają porównanie rozkładów zmiennych o różnych skalach. Widoczne są wartości odstające w wielu zmiennych, szczególnie w ‘compression-ratio’ i ‘peak-rpm’.



Rysunek 4: Porównanie średniej i mediany dla zmiennych ciągłych

Porównanie średniej i mediany (Rys. ??) potwierdza asymetrię rozkładów. W przypadku większości zmiennych średnia przewyższa medianę, co jest charakterystyczne dla rozkładów z prawostroczną asymetrią.

2.1 Kod implementujący estymację punktową

```
def calculate_distribution_parameters(data):
    params = {}
    # Miary tendencji centralnej
    params['średnia'] = data.mean()
    params['średnia geometryczna'] = stats.gmean(data) if all(data > 0) else np.nan
    params['średnia harmoniczna'] = stats.hmean(data) if all(data > 0) else np.nan
    params['Median'] = data.median()
    params['Moda'] = data.mode()[0]

    # Miary rozproszenia
    params['Wariancja (pr bkowa)'] = data.var(ddof=1)
    params['Wariancja (populacyjna)'] = data.var(ddof=0)
    params['Odchylenie standardowe (pr bkowe)'] = data.std(ddof=1)
    params['Odchylenie standardowe (populacyjne)'] = data.std(ddof=0)
    params['Odchylenie przeciętnego'] = (data - data.mean()).abs().mean()
    params['Odchylenie cwiartkowe'] = (np.percentile(data, 75) - np.percentile(data, 25))/2
    params['Współczynnik zmienności'] = (data.std()/data.mean())*100 if data.mean() != 0 else np.nan

    # Miary kształtu rozkładu
    params['Skosność'] = stats.skew(data, bias=False)
    params['Współczynnik asymetrii Pearsona'] = 3 * (data.mean() - data.median()) / data.std() if data.std() != 0 else np.nan
```

```

params['Kurtoza'] = stats.kurtosis(data, bias=False)
params['Eksces'] = stats.kurtosis(data, bias=False, fisher=True)

# Kwantyle i miary pozycyjne
params['Minimum'] = data.min()
params['Maximum'] = data.max()
params['Zakres'] = data.max() - data.min()
params['Q1 (25%)'] = np.percentile(data, 25)
params['Q2 (50%)'] = np.percentile(data, 50)
params['Q3 (75%)'] = np.percentile(data, 75)
params['P10 (10%)'] = np.percentile(data, 10)
params['P90 (90%)'] = np.percentile(data, 90)
params['P95 (95%)'] = np.percentile(data, 95)
params['P99 (99%)'] = np.percentile(data, 99)
params['IQR'] = np.percentile(data, 75) - np.percentile(data, 25)

# Dodatkowe statystyki
params['Bardzo standardowy sredniej'] = data.std(ddof=1) / np.sqrt(len(data))
params['Suma'] = data.sum()
params['Liczba obserwacji'] = len(data)
params['Liczba unikalnych wartosci'] = data.nunique()

# Momenty centralne
params['Moment centralny rz du 1'] = np.mean((data - data.mean()) ** 1)
params['Moment centralny rz du 2'] = np.mean((data - data.mean()) ** 2)
params['Moment centralny rz du 3'] = np.mean((data - data.mean()) ** 3)
params['Moment centralny rz du 4'] = np.mean((data - data.mean()) ** 4)

return params

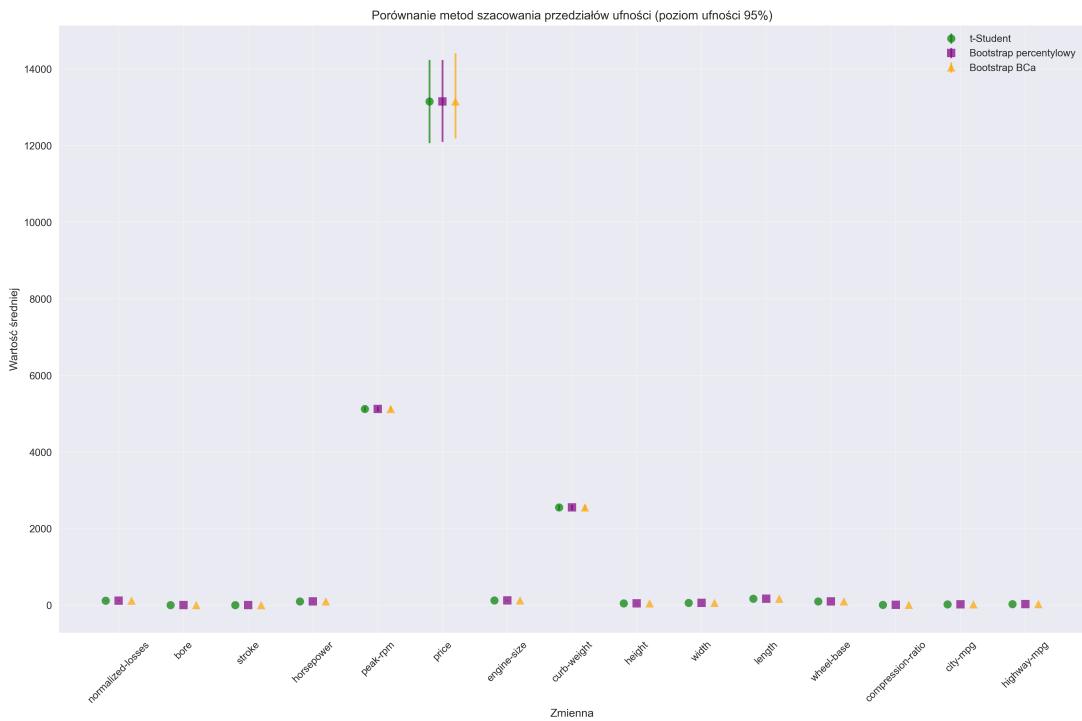
```

Listing 2: Funkcja do obliczania parametrów rozkładu

3 Estymacja parametrów (przedziałowa)

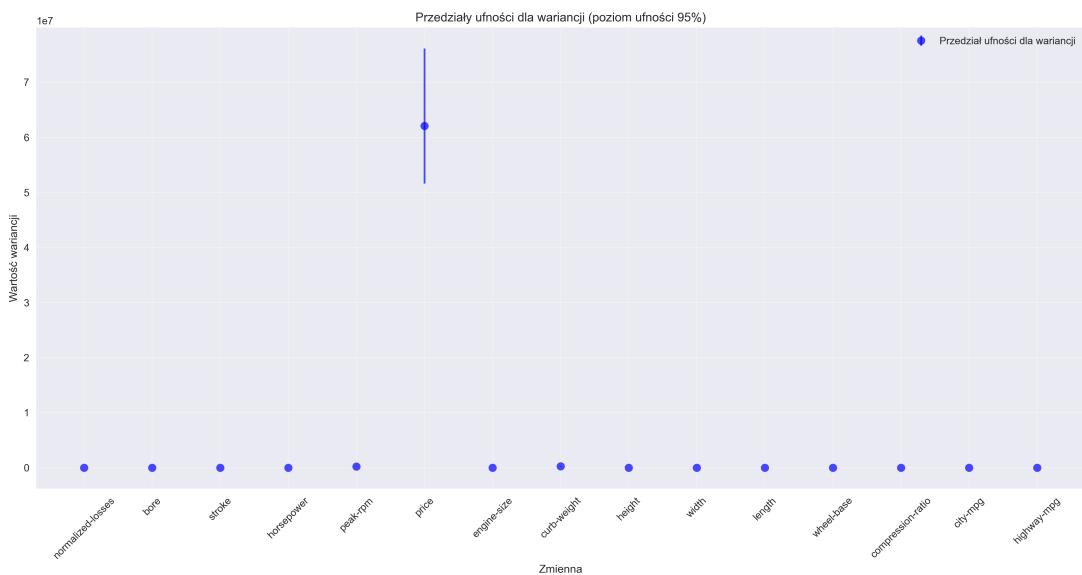
W tej sekcji zostały obliczone przedziały ufności dla średniej i wariancji z wykorzystaniem różnych metod:

- Przedziały ufności dla średniej obliczone metodą parametryczną (opartą o rozkład t-Studenta)
- Przedziały ufności dla średniej obliczone metodami nieparametrycznymi (bootstrap percentylowy oraz bootstrap BCa)
- Przedziały ufności dla wariancji obliczone metodą parametryczną (opartą o rozkład chi-kwadrat)



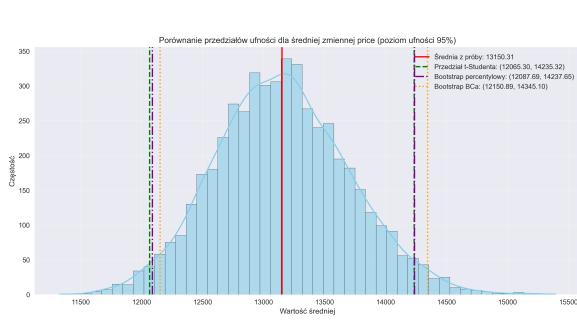
Rysunek 5: Porównanie metod szacowania przedziałów ufności dla średniej

Rysunek ?? prezentuje porównanie trzech metod wyznaczania przedziałów ufności dla średniej. Można zauważyc, że metoda bootstrapowa BCa często generuje niesymetryczne przedziały ufności, co jest korzystne w przypadku rozkładów o wyraźnej asymetrii.

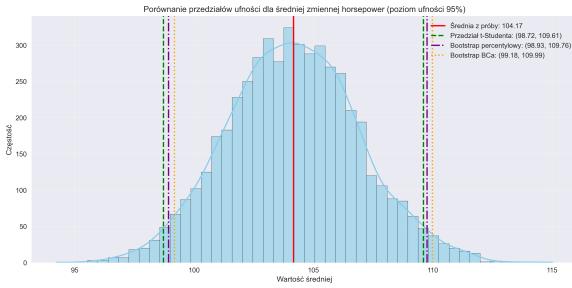


Rysunek 6: Przedziały ufności dla wariancji

Rysunek ?? przedstawia przedziały ufności dla wariancji. Zmienne o wysokiej wariancji, takie jak 'price', mają szersze przedziały ufności, co wskazuje na większą niepewność estymacji.



(a) Przedziały ufnosci dla zmiennej price



(b) Przedziały ufnosci dla zmiennej horsepower

Rysunek 7: Porównanie metod wyznaczania przedziałów ufnosci dla wybranych zmiennych

Szczegółowa analiza dla zmiennych ‘price’ i ‘horsepower’ (Rys. ??) pokazuje rozkład bootstrapowy średniej oraz różne metody estymacji przedziałów ufnosci. Widoczne są różnice w szerokosci i położeniu przedziałów w zależnosci od zastosowanej metody.

3.1 Kod implementujący estymację przedziałową

```
# Funkcje do obliczania przedzia w ufnosci parametrycznymi metodami
def mean_confidence_interval_t(data, confidence=0.95):
    n = len(data)
    mean = np.mean(data)
    std_err = np.std(data, ddof=1) / np.sqrt(n)
    t_critical = stats.t.ppf((1 + confidence) / 2, n-1)
    margin_of_error = t_critical * std_err
    return (mean - margin_of_error, mean + margin_of_error)

def variance_confidence_interval(data, confidence=0.95):
    n = len(data)
    var = np.var(data, ddof=1)
    chi2_lower = stats.chi2.ppf((1 - confidence) / 2, n-1)
    chi2_upper = stats.chi2.ppf((1 + confidence) / 2, n-1)
    var_lower = (n-1) * var / chi2_upper
    var_upper = (n-1) * var / chi2_lower
    return (var_lower, var_upper)

# Funkcje do obliczania przedzia w ufnosci metod bootstrap
def bootstrap_confidence_interval(data, n_bootstrap=5000,
                                    confidence=0.95):
    bootstrap_means = []
    for _ in range(n_bootstrap):
        bootstrap_sample = resample(data, replace=True, n_samples=len(data))
        bootstrap_means.append(np.mean(bootstrap_sample))
    lower_percentile = (1 - confidence) / 2 * 100
    upper_percentile = (1 + confidence) / 2 * 100
    return np.percentile(bootstrap_means, [lower_percentile,
                                           upper_percentile])
```

```

def bootstrap_bca_interval(data, statistic=np.mean, n_bootstrap
=5000, confidence=0.95):
    theta_hat = statistic(data)
    bootstrap_replicates = []
    for _ in range(n_bootstrap):
        bootstrap_sample = resample(data, replace=True, n_samples
=len(data))
        bootstrap_replicates.append(statistic(bootstrap_sample))

    prop_less_than_theta_hat = np.mean([1 if t < theta_hat else 0
for t in bootstrap_replicates])
    z0 = stats.norm.ppf(prop_less_than_theta_hat)

    jackknife_replicates = []
    for i in range(len(data)):
        jack_sample = np.delete(data, i)
        jackknife_replicates.append(statistic(jack_sample))

    jack_mean = np.mean(jackknife_replicates)
    num = np.sum([(jack_mean - jt)**3 for jt in
jackknife_replicates])
    den = 6.0 * np.sum([(jack_mean - jt)**2 for jt in
jackknife_replicates])**1.5

    if abs(den) < 1e-10:
        a = 0
    else:
        a = num / den

    alpha = (1 - confidence) / 2
    z_alpha = stats.norm.ppf(alpha)
    z_1_minus_alpha = stats.norm.ppf(1 - alpha)

    p_lower = stats.norm.cdf(z0 + (z0 + z_alpha) / (1 - a * (z0 +
z_alpha)))
    p_upper = stats.norm.cdf(z0 + (z0 + z_1_minus_alpha) / (1 - a
* (z0 + z_1_minus_alpha)))

    lower_percentile = 100 * p_lower
    upper_percentile = 100 * p_upper

    return np.percentile(bootstrap_replicates, [lower_percentile,
upper_percentile])

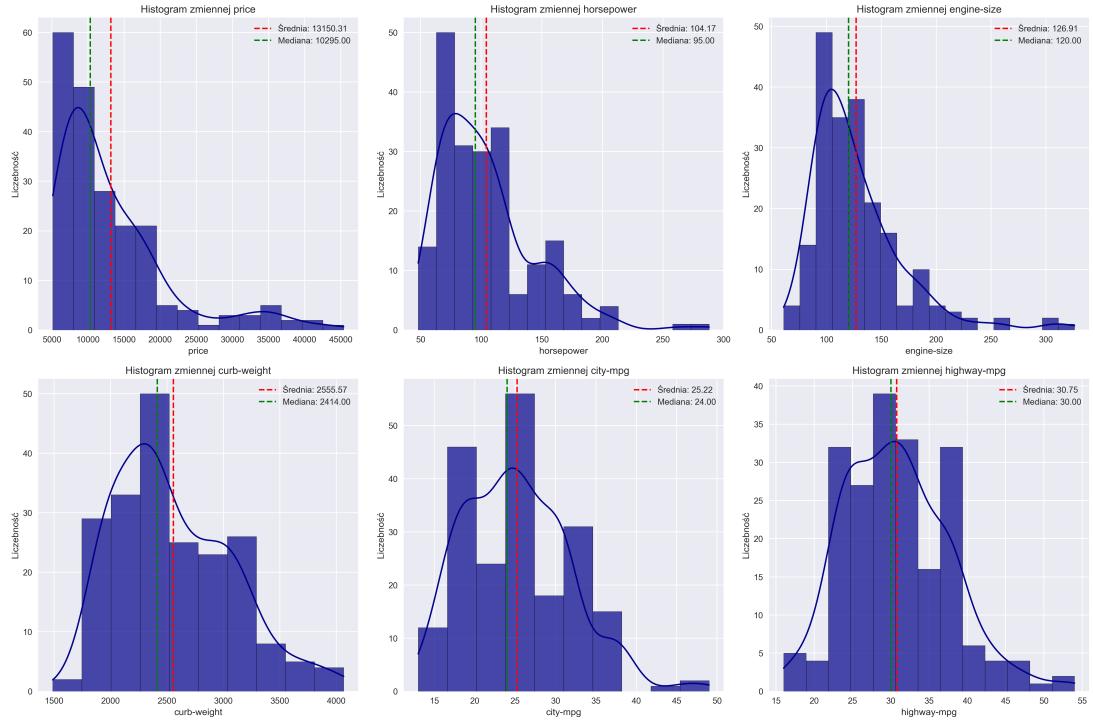
```

Listing 3: Funkcje do obliczania przedziałów ufności

4 Różne wykresy dla analizy zmiennych

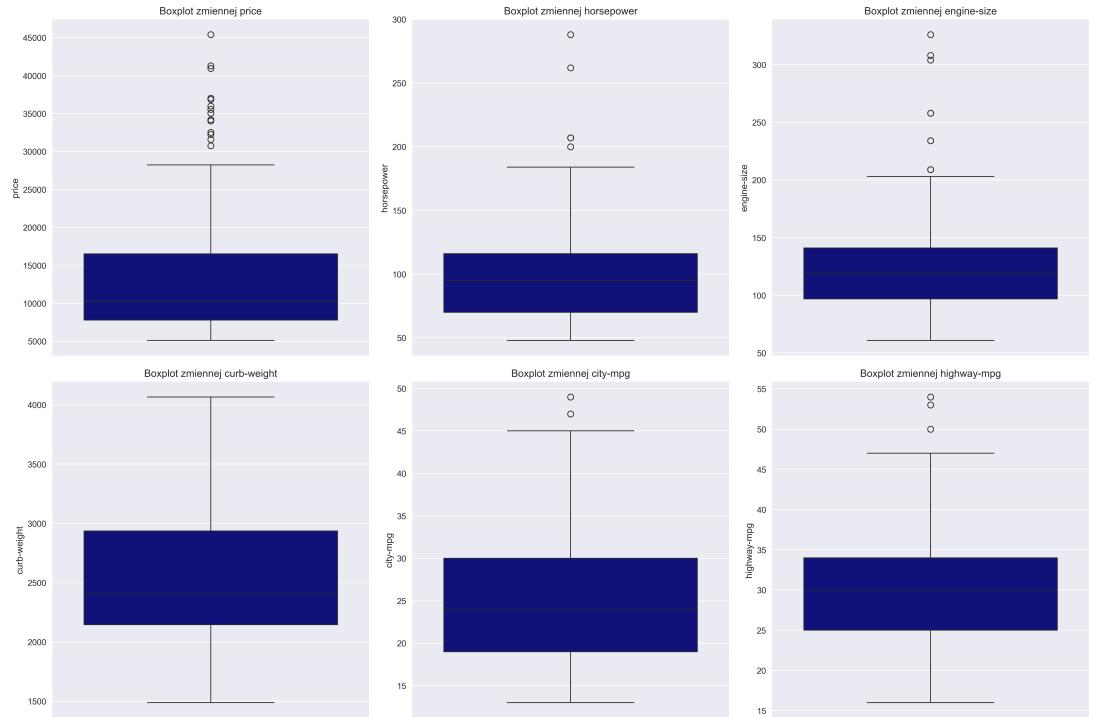
Wizualizacja danych jest kluczowym elementem analizy statystycznej, umożliwiającym lepsze zrozumienie charakterystyk badanych zmiennych. W tej sekcji przedstawiono róż-

norodne typy wykresów, które pomagają w odkrywaniu wzorców i zależności w danych.



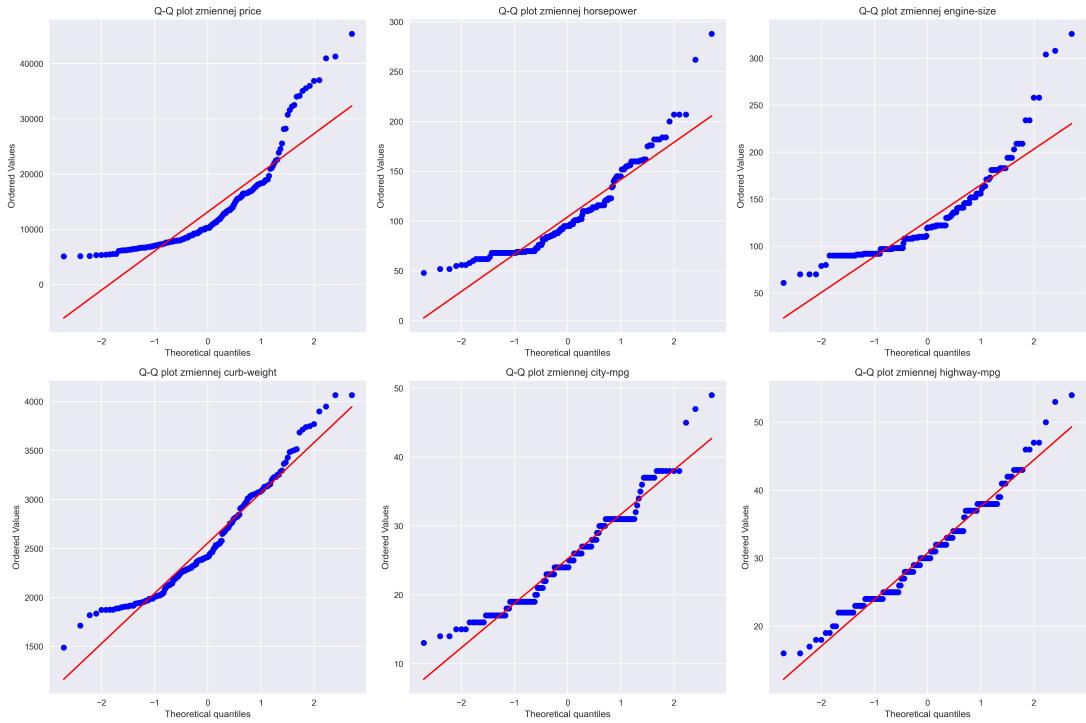
Rysunek 8: Histogramy dla wybranych zmiennych

Histogramy (Rys. ??) pokazują rozkład częstosci występowania wartosci dla wybranych zmiennych. Na wykresach oznaczono średnią (czerwona linia przerywana) oraz medianę (zielona linia przerywana). Widoczne są różnice w kształcie rozkładów, z wyraźną prawostroczną asymetrią dla zmiennych ‘price’, ‘horsepower’ i ‘engine-size’.



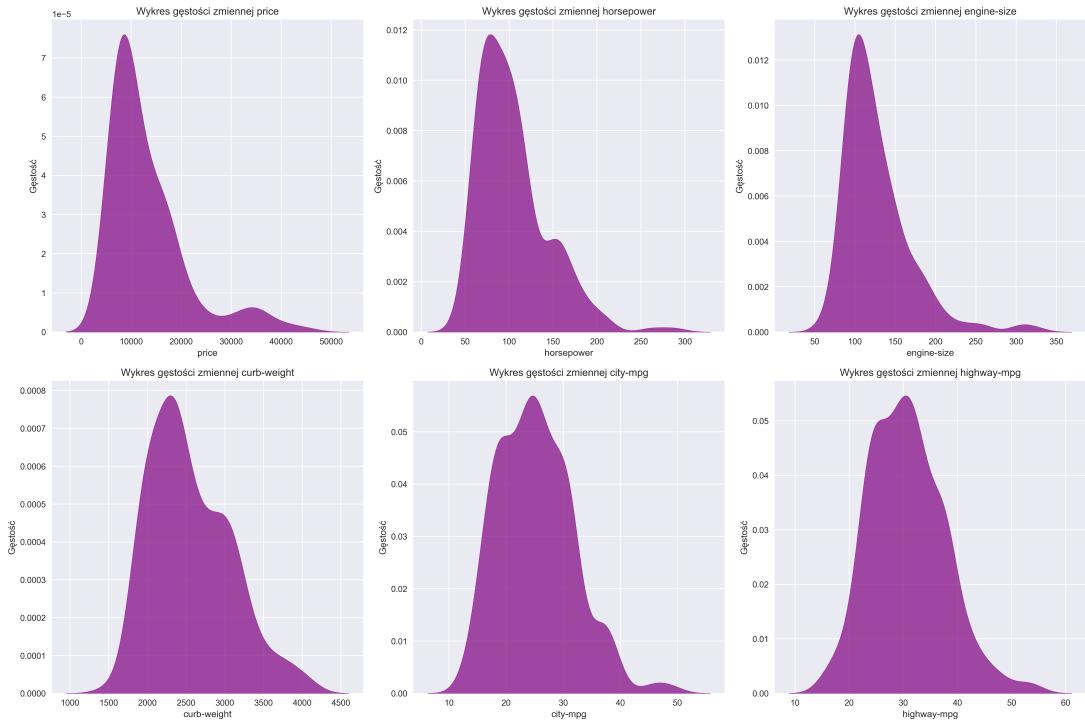
Rysunek 9: Wykresy pudełkowe (boxploty) dla wybranych zmiennych

Wykresy pudełkowe (Rys. ??) przedstawiają kluczowe statystyki rozkładu: medianę, kwartyle, zakres oraz wartości odstające. Widoczne są wartości odstające w większości zmiennych, szczególnie w przypadku ‘price’.



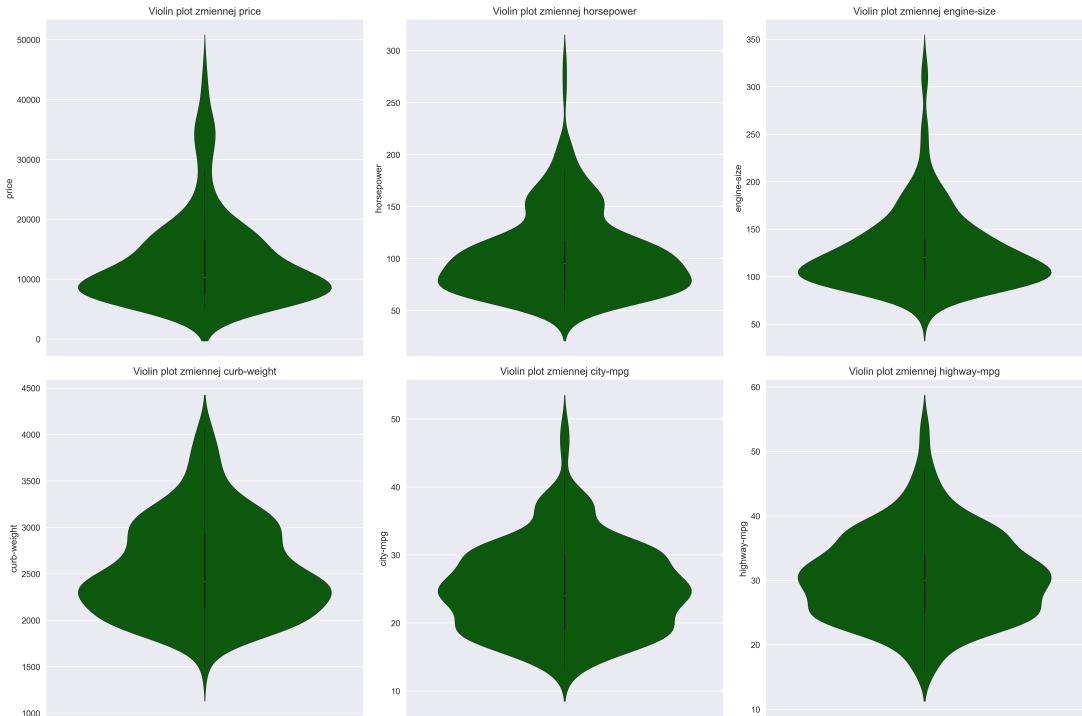
Rysunek 10: Wykresy kwantyl-kwantyl dla wybranych zmiennych

Wykresy kwantyl-kwantyl (Rys. ??) służą do oceny zgodności rozkładu empirycznego z rozkładem teoretycznym (w tym przypadku normalnym). Odchylenia punktów od linii prostej wskazują na odstępstwa od rozkładu normalnego.



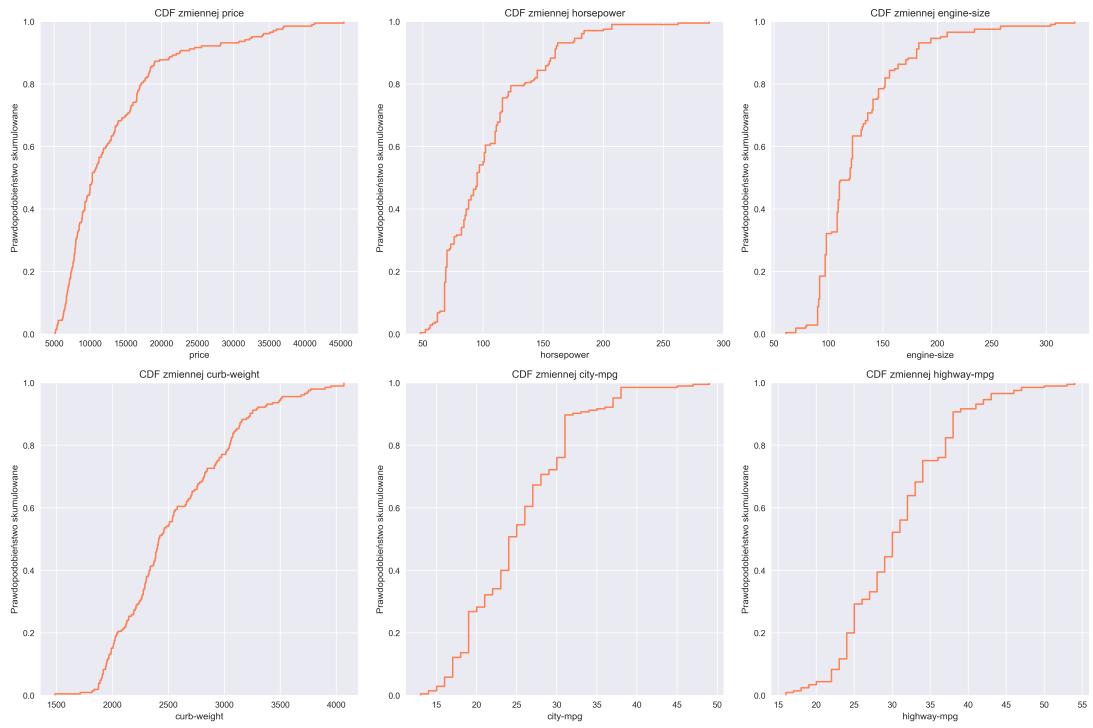
Rysunek 11: Wykresy gęstości prawdopodobieństwa dla wybranych zmiennych

Wykresy gęstości (Rys. ??) wizualizują ciągły rozkład prawdopodobieństwa badanych zmiennych. Wyraźnie widoczne są wielomodalne rozkłady dla niektórych zmiennych.



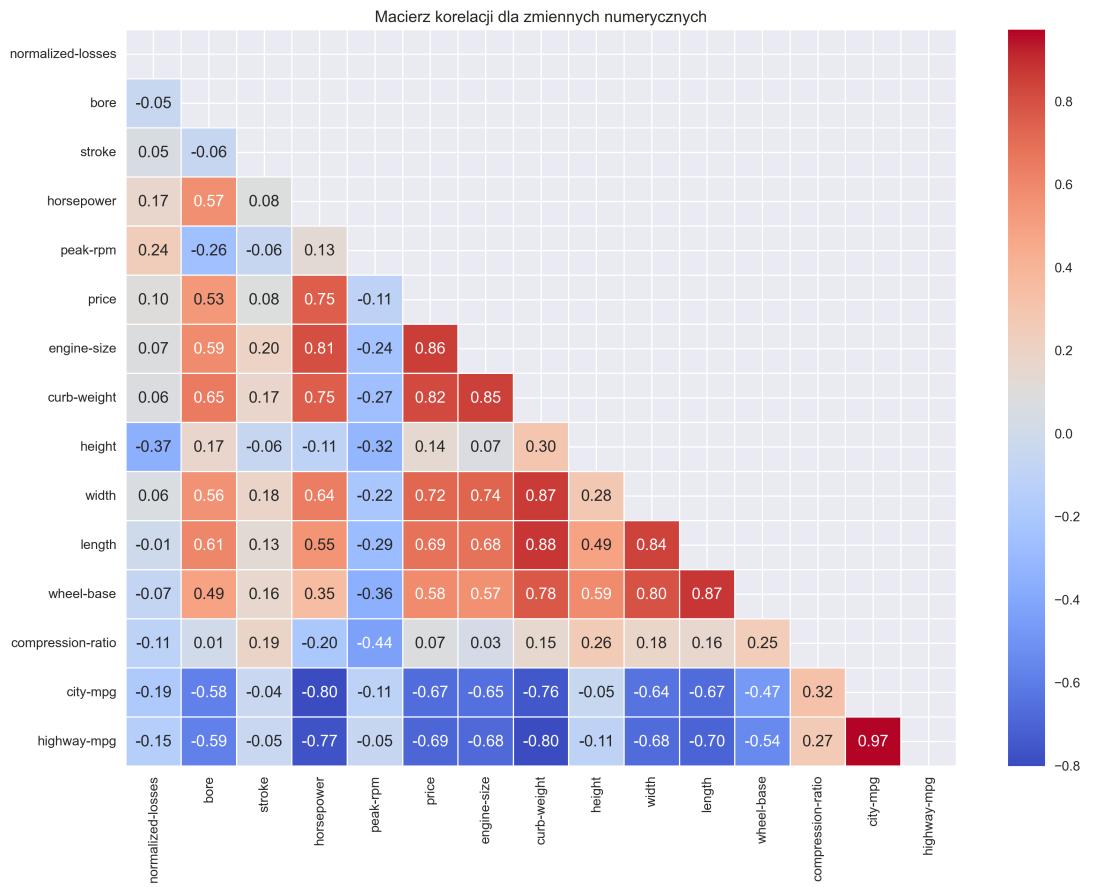
Rysunek 12: Wykresy skrzypcowe dla wybranych zmiennych

Wykresy skrzypcowe (Rys. ??) łączą cechy boxplotów i wykresów gęstości, pokazując zarówno podstawowe statystyki, jak i kształt rozkładu.



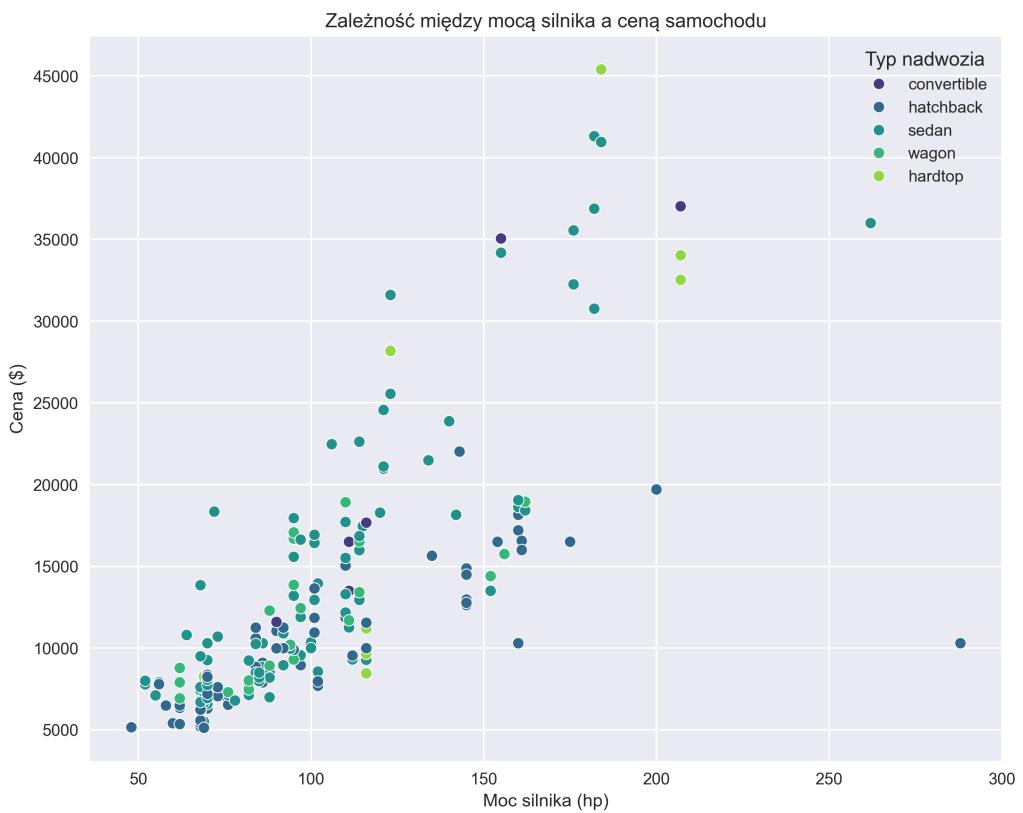
Rysunek 13: Empiryczne dystrybuanty dla wybranych zmiennych

Empiryczne dystrybuanty (Rys. ??) przedstawiają funkcje rozkładu skumulowanego, które są szczególnie użyteczne do analizy kwantyli i porównywania rozkładów.



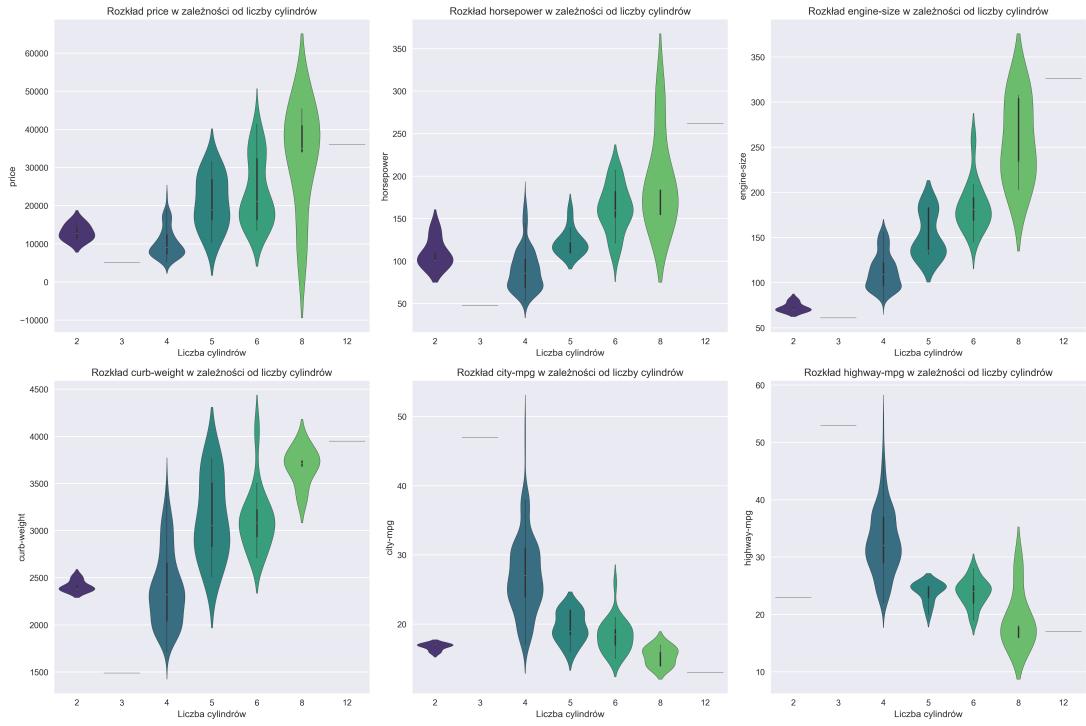
Rysunek 14: Macierz korelacji dla zmiennych numerycznych

Macierz korelacji (Rys. ??) uwidacznia silne zależności między niektórymi zmiennymi, na przykład korelację ujemną między ‘city-mpg’ a ‘engine-size’ oraz silną dodatnią korelację między ‘length’ a ‘width’.



Rysunek 15: Wykres rozrzutu: zależność między mocą silnika a ceną z rozróżnieniem typu nadwozia

Wykres rozrzutu (Rys. ??) pokazuje pozytywną zależność między mocą silnika a ceną samochodu, z rozróżnieniem typów nadwozia. Widoczne jest, że samochody sportowe ('convertible') i sedan mają tendencję do wyższych cen przy podobnej mocy.



Rysunek 16: Wykresy skrzypcowe w zależności od liczby cylindrów

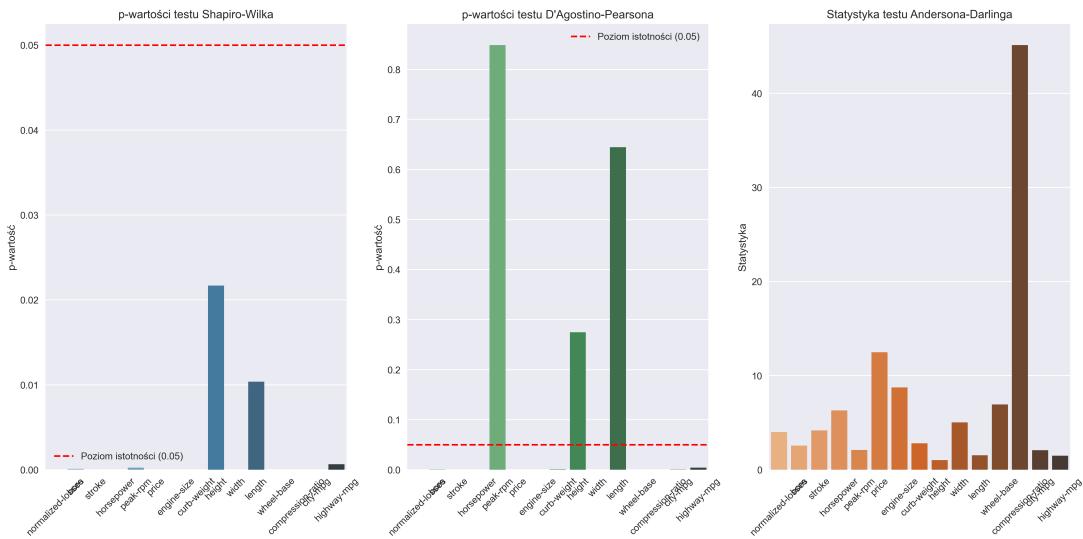
Wykresy skrzypcowe według liczby cylindrów (Rys. ??) ukazują, jak rozkłady zmiennych zależą od liczby cylindrów silnika. Widoczny jest trend rosnący dla mocy silnika ('horsepower') i pojemności silnika ('engine-size') wraz ze wzrostem liczby cylindrów.

5 Sprawdzenie normalności rozkładów danych

Aby ocenić, czy zmienne numeryczne w zbiorze danych Automobile pochodzą z rozkładu normalnego, przeprowadzono trzy różne testy statystyczne:

- Test Shapiro-Wilka
- Test D'Agostino-Pearsona
- Test Andersona-Darlinga

Każdy z testów ocenia normalność rozkładu w nieco inny sposób, co pozwala na bardziej kompleksową analizę.



Rysunek 17: Wyniki testów normalnosci dla zmiennych ciągłych

5.1 Test Shapiro-Wilka

Wyniki testu przedstawione zostały na lewym wykresie Rys. ??.

Wnioski:

Dla żadnej z badanych zmiennych **p-wartosc nie przekroczyła poziomu istotności 0.05**, co oznacza, że dla wszystkich zmiennych **odrzucamy hipotezę o normalnosci rozkładu**. Zmienna **curb-weight** osiągnęła najwyższą p-wartosc (0.023), jednak nadal poniżej progu istotnosci.

5.2 Test D'Agostino-Pearsona

Ten test łączy informację o skosnosci i kurtozie w celu oceny normalnosci (srodkowy wykres na Rys. ??).

Wnioski:

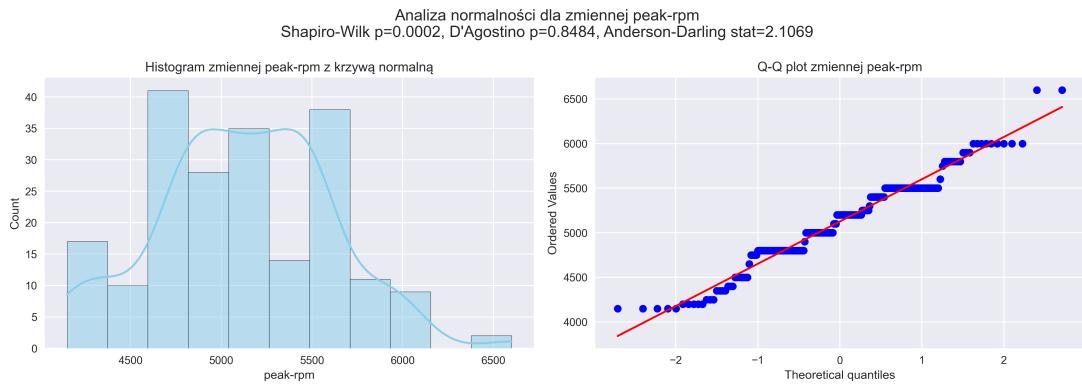
Tylko dwie zmienne (horsepower i width) mają p-wartosc wyraźnie przekraczającą 0.05, co oznacza brak podstaw do odrzucenia hipotezy o normalnosci. Pozostałe zmienne nie spełniają założenia normalnosci, podobnie jak w tescie Shapiro-Wilka.

5.3 Test Anderson-Darlinga

Na ostatnim wykresie Rys. ?? przedstawiono wartosci statystyki testowej. Im wyższa wartosc, tym większe odchylenie od rozkładu normalnego.

Wnioski:

Zmienna **compression-ratio** wyróżnia się **najwyższą statystyką testu (45)**, co wskazuje na **znaczne odstępstwo od rozkładu normalnego**. Najniższe wartosci statystyki występują m.in. dla **height, curb-weight i engine-size**.



Rysunek 18: Szczegółowa analiza normalności dla zmiennej peak_{rpm}

Analiza szczegółowa dla zmiennej peak_{rpm} (Rys. ??) potwierdza jej względny bliskosć do rozkładu normalnego.

5.4 Podsumowanie

Na podstawie przeprowadzonych testów można stwierdzić, że:

- Większość zmiennych w zbiorze danych nie pochodzi z rozkładu normalnego.
- Tylko nieliczne cechy (np. horsepower, width) mogą być rozpatrywane jako zbliżone do rozkładu normalnego.
- Przed użyciem metod statystycznych zakładających normalność (np. testy parametryczne, PCA) zaleca się przeprowadzenie transformacji danych, np. transformacji logarytmicznej lub standaryzacji.

Dzięki zastosowaniu trzech różnych testów uzyskano spójną i wiarygodną ocenę rozkładów cech numerycznych.

5.5 Kod implementujący testy normalności

```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import os

# Ensure figures directory exists
if not os.path.exists('figures'):
    os.makedirs('figures')

shapiro_p_values = []
dagostino_p_values = []
anderson_results = []

for var in continuous_vars:
    data = df[var].dropna()

    _, shapiro_p = stats.shapiro(data)
```

```

shapiro_p_values.append(shapiro_p)

_, dagostino_p = stats.normaltest(data)
dagostino_p_values.append(dagostino_p)

anderson_result = stats.anderson(data, dist='norm')
anderson_results.append(anderson_result.statistic)

plt.figure(figsize=(16, 8))

plt.subplot(1, 3, 1)
sns.barplot(x=continuous_vars, y=shapiro_p_values, palette="Blues_d")
plt.axhline(y=0.05, color='red', linestyle='--', label='Poziom istotnosci (0.05)')
plt.title('p-wartosci testu Shapiro-Wilka')
plt.ylabel('p-wartosc')
plt.xticks(rotation=45)
plt.legend()

plt.subplot(1, 3, 2)
sns.barplot(x=continuous_vars, y=dagostino_p_values, palette="Greens_d")
plt.axhline(y=0.05, color='red', linestyle='--', label='Poziom istotnosci (0.05)')
plt.title('p-wartosci testu D\Agostino-Pearsona')
plt.ylabel('p-wartosc')
plt.xticks(rotation=45)
plt.legend()

plt.subplot(1, 3, 3)
sns.barplot(x=continuous_vars, y=anderson_results, palette="Oranges_d")
plt.title('Statystyka testu Andersona-Darlinga')
plt.ylabel('Statystyka')
plt.xticks(rotation=45)

plt.tight_layout()
plt.savefig('figures/normality_tests.png', dpi=300, bbox_inches='tight')
plt.show()

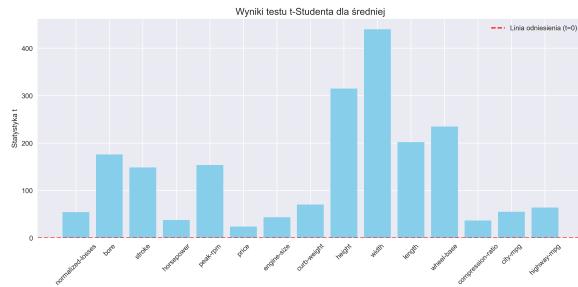
```

Listing 4: Kod testów normalności

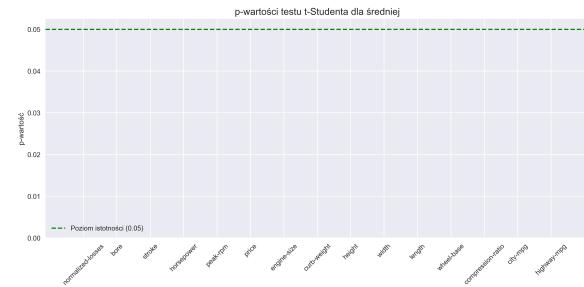
6 Wykorzystanie testu statystycznego dla średniej i wariancji

Aby ocenić właściwości statystyczne cech numerycznych w zbiorze danych Automobile, przeprowadzono dwa typy testów:

- Test t-Studenta dla sredniej { sprawdzający, czy srednia danej zmiennej różni się istotnie od zera,
- Test chi-kwadrat dla wariancji { weryfikujący, czy wariancja zmiennej jest istotnie różna od wartości teoretycznej.



(a) Statystyki testu t-Studenta



(b) p-wartosci testu t-Studenta

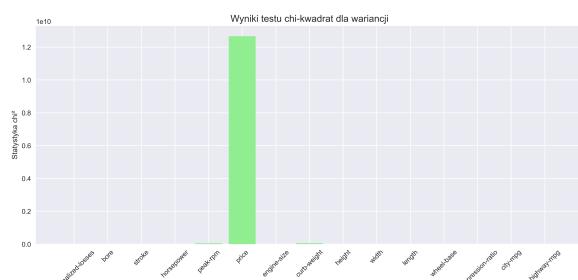
Rysunek 19: Wyniki testu t-Studenta dla sredniej

6.1 Test t-Studenta dla sredniej

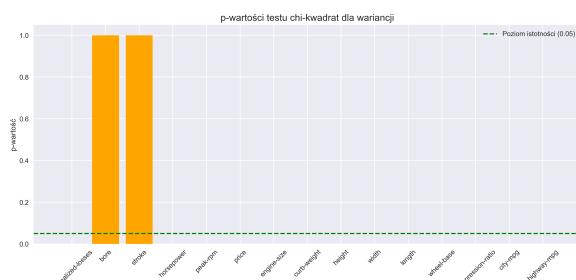
Wyniki testu t-Studenta zostały przedstawione na Rys. ??.

Wnioski:

- Wszystkie zmienne mają wartości p znacznie poniżej progu istotności 0.05, co oznacza, że srednia każdej zmiennej jest istotnie różna od zera.
- Szczególnie wysokie statystyki t zaobserwowano dla cech takich jak width, height, wheel-base czy length | oznacza to, że ich srednie są silnie oddalone od zera w ujęciu statystycznym.
- Wyniki te potwierdzają, że zmienne te zawierają informację statystycznie istotną i mogą być brane pod uwagę przy dalszym modelowaniu lub analizach.



(a) Statystyki testu chi-kwadrat



(b) p-wartosci testu chi-kwadrat

Rysunek 20: Wyniki testu chi-kwadrat dla wariancji

6.2 Test chi-kwadrat dla wariancji

Wyniki testu chi-kwadrat przedstawiono na Rys. ??.

Wnioski:

- Większość zmiennych wykazuje p-wartosci poniżej poziomu istotności 0.05, co oznacza, że ich wariancje różnią się istotnie od wartości oczekiwanej (hipotezy zerowej).
- Wyjątkiem są zmienne bore i stroke, których p-wartosci są bliskie 1, co wskazuje na brak podstaw do odrzucenia hipotezy zerowej (wariancje tych zmiennych są zgodne z teoretycznym założeniem).
- Szczególnie wyróżnia się zmienna price, która osiąga bardzo wysoką statystykę chi², co oznacza dużą niestabilność wariancji | warto rozważyć jej przekształcenie (np. transformację logarytmiczną).

6.3 Podsumowanie

Analiza testów statystycznych pozwala wyciągnąć następujące wnioski:

- średnie wszystkich zmiennych są istotnie różne od zera, co wskazuje na ich przydatność w dalszych analizach.
- Większość zmiennych ma istotnie różną wariancję, co może sugerować konieczność standaryzacji lub transformacji niektórych cech (szczególnie price).
- Zmiennych bore i stroke nie trzeba transformować pod względem wariancji, gdyż nie wykazują odchyleń względem założeń testu.

Wyniki te wspierają decyzje dotyczące dalszego przygotowania danych, m.in. selekcji cech, normalizacji lub inżynierii cech.

6.4 Kod implementujący testy statystyczne

```
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Ensure figures directory exists
if not os.path.exists('figures'):
    os.makedirs('figures')

t_stats = []
p_values_t = []

for var in continuous_vars:
    data = df[var].dropna()
    t_stat, p_value = stats.ttest_1samp(data, 0)
    t_stats.append(t_stat)
    p_values_t.append(p_value)

# Plot for t-statistics
plt.figure(figsize=(12, 6))
plt.bar(continuous_vars, t_stats, color='skyblue')
```

```

plt.axhline(y=0, color='red', linestyle='--', label='Linia
    odniesienia (t=0)')
plt.title('Wyniki testu t-Studenta dla sredniej', fontsize=14)
plt.ylabel('Statystyka t')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.savefig('figures/ttest_statistics.png', dpi=300, bbox_inches=
    'tight')
plt.show()

# Plot for t-test p-values
plt.figure(figsize=(12, 6))
plt.bar(continuous_vars, p_values_t, color='salmon')
plt.axhline(y=0.05, color='green', linestyle='--', label='Poziom
    istotnosci (0.05)')
plt.title('p-wartosci testu t-Studenta dla sredniej', fontsize
    =14)
plt.ylabel('p-wartosc')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.savefig('figures/ttest_pvalues.png', dpi=300, bbox_inches=
    'tight')
plt.show()

chi2_stats = []
p_values_chi2 = []

for var in continuous_vars:
    data = df[var].dropna()
    n = len(data)
    sample_var = np.var(data, ddof=1)
    chi2_stat = (n - 1) * sample_var / 1
    p_value = stats.chi2.sf(chi2_stat, df=n-1)
    chi2_stats.append(chi2_stat)
    p_values_chi2.append(p_value)

# Plot for chi-squared statistics
plt.figure(figsize=(12, 6))
plt.bar(continuous_vars, chi2_stats, color='lightgreen')
plt.title('Wyniki testu chi-kwadrat dla wariancji', fontsize=14)
plt.ylabel('Statystyka chi ')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('figures/chi2_statistics.png', dpi=300, bbox_inches=
    'tight')
plt.show()

# Plot for chi-squared p-values
plt.figure(figsize=(12, 6))

```

```

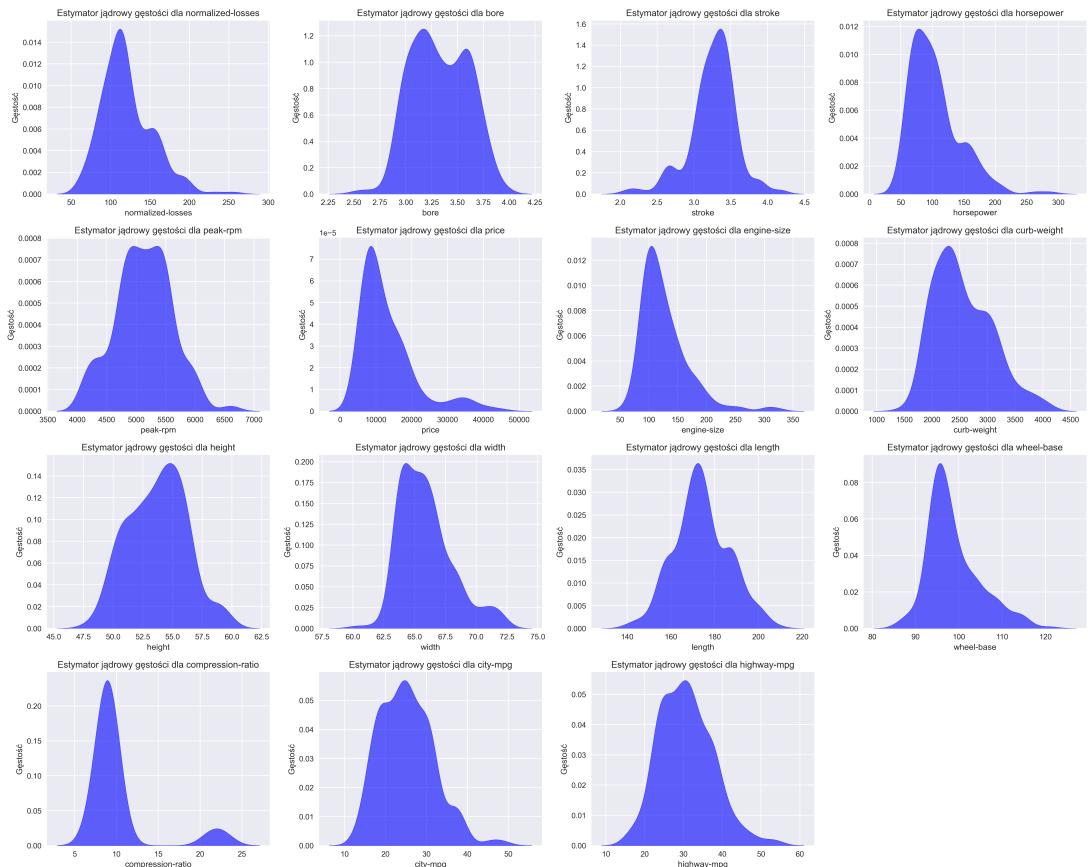
plt.bar(continuous_vars, p_values_chi2, color='orange')
plt.axhline(y=0.05, color='green', linestyle='--', label='Poziom istotnosci (0.05)')
plt.title('p-wartosci testu chi-kwadrat dla wariancji', fontsize=14)
plt.ylabel('p-wartosc')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.savefig('figures/chi2_pvalues.png', dpi=300, bbox_inches='tight')
plt.show()

```

Listing 5: Kod testów dla sredniej i wariancji

7 Estymator jądrowy gęstości

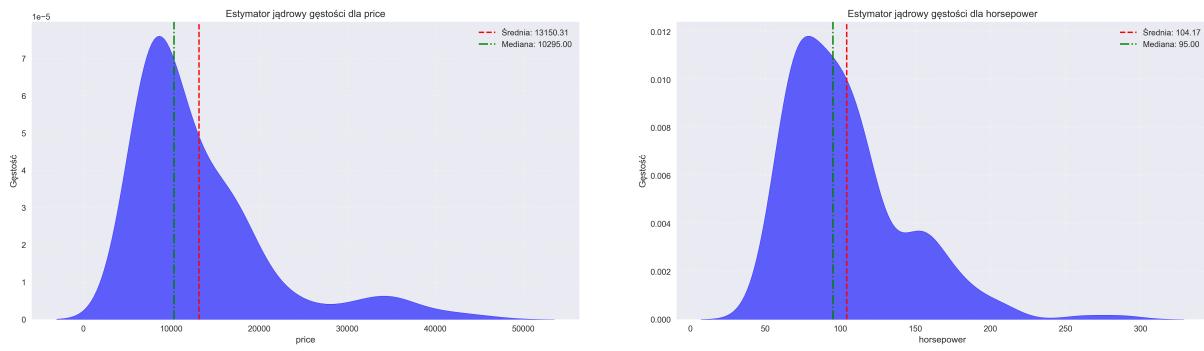
W celu analizy rozkładu zmiennych numerycznych w zbiorze danych Automobile przeprowadzono estymację jądrową gęstości (KDE) dla każdej z cech. Metoda ta umożliwia wygładzenie histogramów i lepsze zrozumienie kształtu rozkładu danych bez wpływu liczby przedziałów histogramu.



Rysunek 21: Estymatory jądrowe gęstości dla wszystkich zmiennych ciągłych

7.1 Wnioski z wykresów KDE

- Wiele zmiennych ma rozkład asymetryczny { m.in. price, horsepower, engine-size oraz compression-ratio cechują się długim ogonem po prawej stronie. Sugeruje to obecność wartości odstających (ang. *outliers*) { pojedynczych, nietypowo wysokich wartości.
- Niektóre zmienne mają rozkład zbliżony do normalnego { np. peak-rpm, height i width są skupione wokół jednej dominującej wartości, co może odzwierciedlać standardy konstrukcyjne wśród producentów samochodów.
- Zmienna compression-ratio wykazuje rozkład dwumodalny, co może sugerować obecność dwóch różnych klas silników (np. benzynowe vs. wysokoprężne).
- Zmienne normalized-losses i price cechują dużą rozpiętość wartości, co może wymagać dalszego przekształcenia danych (np. transformacji logarytmicznej, standaryzacji) w celu poprawy jakości modeli predykcyjnych.
- Rozkłady city-mpg oraz highway-mpg są przesunięte ku lewej, co oznacza, że większość samochodów ma umiarkowane zużycie paliwa, a jedynie nieliczne cechują się bardzo wysoką efektywnością.



(a) Estymator jądrowy gęstości dla zmiennej price

(b) Estymator jądrowy gęstości dla zmiennej horsepower

Rysunek 22: Szczegółowe estymatory jądrowe gęstości dla wybranych zmiennych

7.2 Znaczenie analizy KDE

Estymatory jądrowe gęstości pozwalają lepiej zrozumieć strukturę danych, wykryć potencjalne problemy (takie jak wartości odstające, wielomodalność rozkładu czy zmienność) oraz dobrze odpowiednio transformacje zmiennych. Analiza ta jest istotna jako etap przygotowawczy przed zastosowaniem algorytmów uczenia maszynowego lub budową modeli statystycznych.

7.3 Kod implementujący estymatory jądrowe gęstości

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```

import seaborn as sns
from ucimlrepo import fetch_ucirepo
import os

# Ensure figures directory exists
if not os.path.exists('figures'):
    os.makedirs('figures')

automobile = fetch_ucirepo(id=10)
X = automobile.data.features
y = automobile.data.targets
df = pd.concat([X, y], axis=1)

# Przetworzenie brakuj cych wartosci w zmiennych ci g ych
for column in continuous_vars:
    df[column] = pd.to_numeric(df[column], errors='coerce')
    df[column] = df[column].fillna(df[column].median())

# Create KDE plots for all continuous variables
fig, axes = plt.subplots(4, 4, figsize=(20, 16))
axes = axes.flatten()

for i, var in enumerate(continuous_vars):
    if i < len(axes):
        sns.kdeplot(df[var], ax=axes[i], fill=True, color='blue',
                    alpha=0.6)
        axes[i].set_title(f'Estymator j drowy g stosci dla {var}')
        axes[i].set_xlabel(var)
        axes[i].set_ylabel('G stosc')

    # Turn off any remaining empty axes
for j in range(len(continuous_vars), len(axes)):
    axes[j].axis('off')

plt.tight_layout()
plt.savefig('figures/kde_plots_all_variables.png', dpi=300,
            bbox_inches='tight')
plt.show()

# Optional: Create individual KDE plots for each variable
for var in continuous_vars:
    plt.figure(figsize=(10, 6))
    sns.kdeplot(df[var], fill=True, color='blue', alpha=0.6)
    plt.title(f'Estymator j drowy g stosci dla {var}')
    plt.xlabel(var)
    plt.ylabel('G stosc')
    plt.grid(alpha=0.3, linestyle='--')

    # Add mean and median lines
    plt.axvline(df[var].mean(), color='red', linestyle='--',

```

```

        label=f'srednia: {df[var].mean():.2f}')
plt.axvline(df[var].median(), color='green', linestyle='-.',
            label=f'Mediana: {df[var].median():.2f}')
plt.legend()

plt.tight_layout()
plt.savefig(f'figures/kde_{var.replace("-", "_")}.png', dpi=300, bbox_inches='tight')
plt.close()

```

Listing 6: Kod generujacy estymatory jadrowe gestosci

8 Podsumowanie

Przeprowadzona analiza statystyczna zbioru danych Automobile dostarcza kompleksowego obrazu charakterystyk samochodów z roku 1985. Na podstawie przeprowadzonych badań można wyciągnąć następujące wnioski:

- Większość zmiennych wykazuje asymetrię prawostrońską, co jest typowe dla danych ekonomicznych, takich jak ceny. Zmienne price, horsepower i engine-size charakteryzują się szczególnie silną asymetrią.
- Testy normalności wykazały, że większość zmiennych nie pochodzi z rozkładu normalnego. Jedynie zmienne horsepower i width mogą być rozpatrywane jako zbliżone do rozkładu normalnego.
- Estymacja parametrów za pomocą różnych metod (parametrycznych i nieparametrycznych) pokazała, że metody bootstrapowe mogą dostarczać bardziej adekwatnych przedziałów ufności w przypadku rozkładów niesymetrycznych.
- Testy statystyczne potwierdziły, że wszystkie analizowane zmienne mają średnie istotnie różne od zera, co potwierdza ich informacyjną wartość.
- Analiza korelacji ujawniła silne zależności między wieloma zmiennymi, co może być istotne przy budowie modeli predykcyjnych. Szczególnie silne związki obserwujemy między wymiarami samochodu a jego masą oraz między pojemnością silnika a jego mocą.
- Estymatory jadrowe gęstości uwidocznili wielomodalność niektórych rozkładów (np. compression-ratio), co sugeruje obecność różnych kategorii pojazdów w zbiorze danych.

Wyniki te wskazują na potrzebę ostrożnego podejścia do analizy danych, ze szczególnym uwzględnieniem:

- Transformacji zmiennych o silnej asymetrii (np. transformacja logarytmiczna dla price)
- Zastosowania metod odpornych na odstępstwa od normalności
- Uwzględnienia potencjalnych podgrup w danych przy modelowaniu

Przeprowadzona analiza dostarcza solidnych podstaw do dalszych badań, takich jak budowa modeli regresji do przewidywania ceny pojazdu w oparciu o jego charakterystyki techniczne lub analiza grupowania w celu identyfikacji segmentów rynku samochodowego.