Analiza światowych ocen ramenu

Rafał Klifeld, Patryk Sadowski, Agata Wicikowska, Wydział Fizyki Technicznej i Matematyki Stosowanej, Fizyka Techniczna, spec. Informatyka Stosowana st II, sem I, Politechnika Gdańska. Gdańsk, Polska

1. Wstęp - motywacja, cele

Praca motywowana jest chęcią poznania lokalizacji, gdzie można zjeść najlepszy ramen. Grupa postanowiła wykorzystać te dane, żeby wybrać się do tego miejsca i zjeść wspólnie to wywodzące się z Azji danie.

2. Opis danych - struktura zbiorów, opis zmiennych, pochodzenie

W pracy zostały wykorzystane dane w postaci pliku ramen-ratings.csv, który jest eksportem danych z “The Big List”. Zawiera on recenzje ponad 2500 rodzajów ramenów z różnych miejsc na świecie. Każdy wpis w pliku jest recenzją jednej zupki z proszku. Każdy wiersz zawiera nazwę producenta, odmianę zupki z proszku, kraj, styl podania i ilość gwiazdek, które wskazują na ocenę jakości zupki z proszku. Im więcej gwiazdek tym jest ona wyższa. Maksymalna ocena to 5 gwiazdek.

- *Review #* - Numer recenzji
- *Brand* - Producent
- *Variety* - Odmiana
- *Style* - Styl podania
- *Country* - Kraj pochodzenia
- *Stars* - Liczba przyznanych gwiazdek od 0 do 5
- *Top Ten* - Zajęte miejsce w Top10 w danym roku

3. Opis procesu przygotowywania danych do analizy - kolejne kroki

3.1 Przygotowanie

Dane ramen-ratings.csv zawierają 2580 recenzji zupek z proszku z ocenami w granicach <0; 5>, w których znajduje się 2413 odmian ramenu z całego świata, 355 producentów oraz 7 sposobów serwowania.

```
data = pd.read_csv('ramen-ratings.csv')
data.head()
```

Rys. 3.1. Załadowanie danych ramen-ratings.csv poprzez bibliotekę Pandas.

	Review #	Brand	Variety	Style	Country	Stars	Top Ten
0	2580	New Touch	T's Restaurant Tantanmen	Cup	Japan	3.75	NaN
1	2579	Just Way	Noodles Spicy Hot Sesame Spicy Hot Sesame Guan...	Pack	Taiwan	1	NaN
2	2578	Nissin	Cup Noodles Chicken Vegetable	Cup	USA	2.25	NaN
3	2577	Wei Lih	GGE Ramen Snack Tomato Flavor	Pack	Taiwan	2.75	NaN
4	2576	Ching's Secret	Singapore Curry	Pack	India	3.75	NaN

Rys. 3.2. Wywołanie pięciu pierwszych rekordów w celu prezentacji danych.

```
data[['Brand', 'Variety', 'Style', 'Country', 'Top Ten']].nunique()
```

```
Brand       355
Variety    2413
Style         7
Country      38
Top Ten      38
dtype: int64
```

Rys. 3.3. Obliczanie unikalnych wartości kolumn Brand, Variety, Style, Country, Top Ten.

Dane posiadają 2580 wierszy, z podziałem na siedem kolumn “Review #”, “Brand”, “Variety”, “Style”, “Country”, “Stars”, “Top Ten”.

```
data.shape
```

```
(2580, 7)
```

Rys. 3.4. Wyświetlanie rozmiaru danych wiersze × kolumny.

Zestaw danych został następnie sprawdzony pod kątem brakujących danych, czyli pól z pustymi wartościami.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2580 entries, 0 to 2579
Data columns (total 7 columns):
Review #    2580 non-null int64
Brand       2580 non-null object
Variety     2580 non-null object
Style       2578 non-null object
Country     2580 non-null object
Stars       2580 non-null object
Top Ten     41 non-null object
dtypes: int64(1), object(6)
memory usage: 141.2+ KB
```

Rys. 3.5. Wyświetlanie informacji dotyczących danych.

Dane zostały testowo wczytane, a następnie sprawdzone zostały typy danych kolumn. Wszystkie kolumny poza id (int64) były typu obiekt.

```
data.dtypes
```

```
Review #     int64
Brand       object
Variety     object
Style       object
Country     object
Stars       object
Top Ten     object
dtype: object
```

Rys. 3.6. Wyświetlanie informacji dotyczącej typów danych kolumn.

3.1.1 Kolumna “Stars”

Kolumna z ocenami (Stars) została sprawdzona pod kątem unikalnych danych.

```
data.Stars.sort_values().unique()
```

```
array(['0', '0.1', '0.25', '0.5', '0.75', '0.9', '1', '1.1', '1.25',
       '1.5', '1.75', '1.8', '2', '2.1', '2.125', '2.25', '2.3', '2.5',
       '2.75', '2.8', '2.85', '2.9', '3', '3.0', '3.00', '3.1', '3.125',
       '3.2', '3.25', '3.3', '3.4', '3.5', '3.50', '3.6', '3.65', '3.7',
       '3.75', '3.8', '4', '4.0', '4.00', '4.125', '4.25', '4.3', '4.5',
       '4.50', '4.75', '5', '5.0', '5.00', 'Unrated'], dtype=object)
```

Rys. 3.7. Wyświetlanie unikalnych wartości dla kolumny “Stars”.

Procentowy wynik danych odnoszący się do wartości “Unrated” wynosi 0.12%, dlatego zostało zdecydowane, aby w celu integralności danych zastąpić je wartością 0.

```
print('Procentage of `Unrated` ramens : {}%'.format(round(np.sum(data.Stars.isin(['Unrated']))*100/len(data),2)))
```

```
Procentage of `Unrated` ramens : 0.12%
```

Rys. 3.8. Wyświetlanie procentowej wartości danych posiadających wartość “Unrated”.

Kolumna z ocenami została przekonwertowana do typu liczbowego funkcją `to_numeric()`[1] z biblioteki Pandas. Następnie dane tekstowe “Unrated” zostały zamienione na ocenę 0, w celu zachowania jednolitości danych, gdyż były tylko 3 takie wartości i zajmowały 0.12% wszystkich ocen.

```
# Setting all ratings to float and filling with zeros if 'Unrated' or missed value
data['Stars'] = pd.to_numeric(data.Stars, errors='coerce')
data['Stars'] = data.Stars.fillna(0)
data['Stars'] = data.Stars.astype('float')

# Rounding ratings to equal amount of decimal places
data['Stars'] = np.around(data.Stars,decimals=1)
```

Rys. 3.9. Konwersja kolumny do typu numerycznego - float oraz zamiana wartości “Unrated” na 0.

Dane tekstowe zostały zamienione na małe litery, w celu ujednolicenia użytego nazewnictwa.

```
# To lower case
data.Brand = data.Brand.str.lower()
data.Variety = data.Variety.str.lower()
data.Style = data.Style.str.lower()
data.Country = data.Country.str.lower()
```

Rys. 3.10. Konwersja wielkości liter na małe.

3.1.2 Kolumna “Style”

```
data.Style.unique()
```

```
array(['cup', 'pack', 'tray', 'bowl', 'box', 'can', 'bar', nan],
      dtype=object)
```

Rys. 3.11. Wyświetlanie unikalnych wartości dla kolumny “Style”.

```
data[data.Style.isnull()]
```

	Review #	Brand	Variety	Style	Country	Stars	Top Ten
2152	428	kamfen	e menm chicken	NaN	china	3.8	NaN
2442	138	unif	100 furong shrimp	NaN	taiwan	3.0	NaN

```
# Filling 2 missing ramen styles after online check
data['Style'] = np.where(data.Style.isnull(), 'pack', data.Style)
```

Rys. 3.12. Wyświetlanie wierszy w kolumnie “Style” posiadających wartości NULL i uzupełnienie ich.

Z racji małej ilości brakujących danych w kolumnie *Style*, dane zostały uzupełnione po uprzednim zgromadzeniu brakujących informacji w internecie dla ramenów Unif-100 Furong Shrimp oraz Kamfen E-Men Chicken.

3.2.3 Kolumna “Country”

Następnie zostały wyświetlone unikatowe wartości krajów, w celu sprawdzenia jednolitości nazewnictwa państw.

```
data.Country.unique()
```

```
array(['japan', 'taiwan', 'usa', 'india', 'south korea', 'singapore',
       'thailand', 'hong kong', 'vietnam', 'ghana', 'malaysia',
       'indonesia', 'china', 'nigeria', 'germany', 'hungary', 'mexico',
       'fiji', 'australia', 'pakistan', 'bangladesh', 'canada', 'nepal',
       'brazil', 'uk', 'myanmar', 'netherlands', 'united states',
       'cambodia', 'finland', 'sarawak', 'philippines', 'sweden',
       'colombia', 'estonia', 'holland', 'poland', 'dubai'], dtype=object)
```

Rys. 3.13. Wyświetlanie unikalnych wartości dla kolumny “Country”.

```
geolocator = Nominatim(timeout=20, user_agent="pg-ramen-ppir/1.0")
geocode = partial(geolocator.geocode, language="en")
```

Rys. 3.14. Inicjalizacja pakiety “geopy”.

Został użyty pakiet GeoPy do ujednolicenia nazw krajów oraz ich kodów ISO z zewnętrznej bazy danych. W celu pobrania danych została użyta funkcja `Nominatim()` wraz z parametrem `timeout = 20` w celu pobierania poszczególnych grup danych co 20 milisekund, aby przeciwdziałać zapchaniu serwera. Nazwa UserAgenta została ustawiona jako `pg-ramen-ppir/1.0`. Za pomocą funkcji `partial()` zostały wprowadzone domyślne parametry dla pakietu GeoPy i jego funkcji, między innymi nazwy państw w języku angielskim.

```
# Unify country names
countries = list(map(lambda c: geocode(c).address.split(', ')[-1], list(data.Country.unique())))

fix_dict = {k:v for k,v in zip(list(data.Country.unique()), countries) if k != v}
```

Rys. 3.15. Utworzenie słownika z poprawnymi nazwami krajów.

Funkcją lambda został utworzony słownik na podstawie unikalnych wartości z załadowanych danych z kolumny “Country”. Do zmiennej `countries` każda poszczególna unikalna wartość została zastąpiona ujednoliconą wartością, powodując spowolnienie czasu wykonania programu, jednak eliminując proces ręcznego poprawiania danych. Wyszukane zostały odpowiednie nazwy państw z GeoPy dla wszystkich wartości, gdyż dane posiadały wpisane państwa, miasta, skrótowe nazwy państw i miast oraz potoczne ich określenia. Dla przykładu wartości typu `USA` oraz `US` zostały zastąpione wyrażeniem `United States`, wartość `dubai` została zastąpiona wartością `United Arab Emirates`.

```
data.Country.replace(fix_dict,inplace=True)
```

Rys. 3.16. Zastąpienie wartości z kolumny “Country” wartościami ze słownika.

Przetworzone dane z poprawnymi ujednoliconymi nazwami zostały ponownie wprowadzone do oryginalnych danych zastępując poprzednią kolumnę “Country”

```
data.Country.unique()
```

```
array(['Japan', 'Taiwan', 'United States', 'India', 'South Korea',
       'Singapore', 'Thailand', 'China', 'Vietnam', 'Ghana', 'Malaysia',
       'Indonesia', 'Nigeria', 'Germany', 'Hungary', 'Mexico', 'Fiji',
       'Australia', 'Pakistan', 'Bangladesh', 'Canada', 'Nepal', 'Brazil',
       'United Kingdom', 'Myanmar', 'Netherlands', 'Cambodia', 'Finland',
       'Philippines', 'Sweden', 'Colombia', 'Estonia', 'Poland',
       'United Arab Emirates'], dtype=object)
```

Rys. 3.17. Wyświetlanie unikalnych wartości dla kolumny “Country” po obróbce.

W celu sprawdzenia wyniku ponownie zostały wyświetlone unikalne wartości dla kolumny “Country” w celu porównania wartości poprzednich z obecnymi.

4. Analiza danych - przyjęte założenia, krótki opis metod i obranej metodologii analizy

4.1 Założenia i ogólny proces

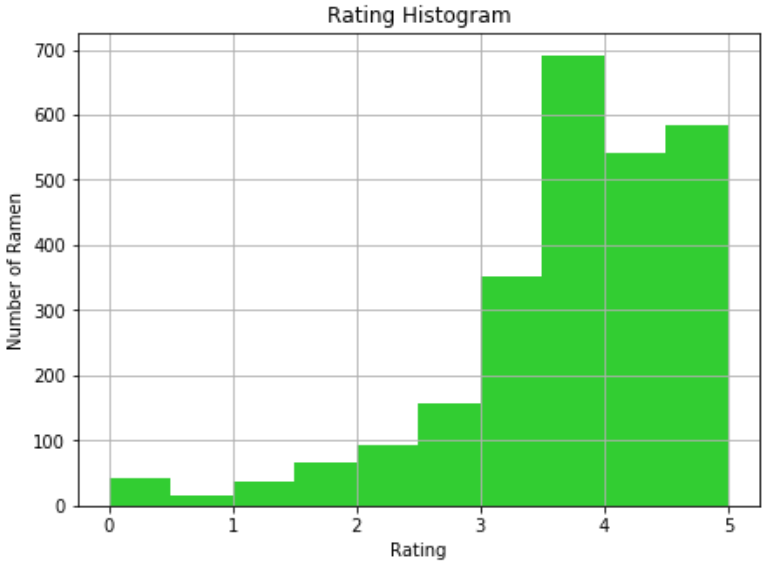
Po przygotowaniu danych do obróbki została rozpoczęta analiza danych. Metodą `describe()` zostały otrzymane dane typu: ilość ocen, średnia ocen, odchylenie standardowe ocen, minimalną ocenę, maksymalną ocenę. Za pomocą funkcji `median()` otrzymaliśmy medianę ocen. Dane następnie zostały zwizualizowane na wykresach dla:

- stosunku ilości ramenów do wystawionych ocen
- stosunku liczby zjedzonych ramenów do typów opakowań.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “pack”.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “bowl”.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “cup”.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “tray”.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “box”.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “bar”.
- stosunku wystawionych ocen do ilości ocen dla typu opakowania “can”.
- stosunku ilości ramenów do opakowań w danym kraju
- stosunku kraju do liczby osób uczestniczących w ankiecie.
- stosunku marki ramenu do ilości pozytywnych ocen.
- stosunku marki ramenu do ilości negatywnych ocen.
- stosunku procentowego ramenu ocenianego pozytywnie do ramenu ocenianego negatywnie.
- stosunku pozytywnych i negatywnych ocen ramenu w każdym stylu podania.
- stosunku rodzaju ramenu do ilości firm go produkujących.

4.2 Analiza danych na podstawie wykresów

```
rat = plt.subplots(figsize=(7, 5))
rat = data.Stars.hist(color='limegreen')
plt.xlabel('Rating')
plt.ylabel('Number of Ramen')
plt.title('Rating Histogram')
plt.show()
```

Rys. 4.1. Wyświetlanie wykresu stosunku ilości ramenów do wystawionych ocen z przedziału <0;5>.



Rys. 4.2. Wykres stosunku ilości ramenów do wystawionych ocen z przedziału <0;5>.

Na wykresie został przedstawiony empiryczny rozkład wysokości ocen ramenu.

```
data['Stars'].median()
```

```
3.8
```

```
round(data['Stars'].mean(),2)
```

```
3.65
```

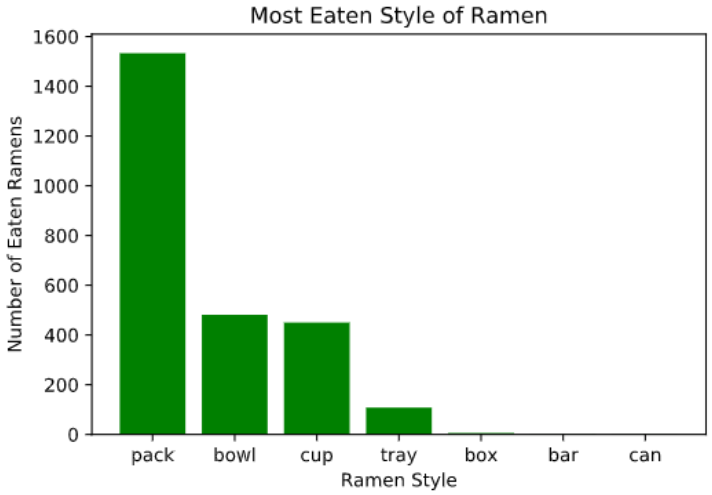
Rys. 4.3. Obliczanie mediany i średniej ocen ramenów.

Średnia ocen ramenu w pobranych danych z pliku ramen-ratings.csv wynosi 3.65 gwiazdki, zaś jej mediana 3.8. Odchylenie standardowe wynosi 1.02, minimalna ocena ramenu wynosi 0, zaś maksymalna 5.

Most eaten Style of Ramen

```
styles = dict(data['Style'].value_counts())
plt.bar(range(len(styles)), list(styles.values()), align='center', color='green')
plt.xticks(range(len(styles)), list(styles.keys()))
plt.ylabel('Number of Ratings')
plt.xlabel('Ramen Style')
plt.title('Most eaten Style of Ramen')
plt.show()
```

Rys. 4.4. Wyświetlanie wykresu stosunku liczby zjedzonych ramenów do typów opakowań.

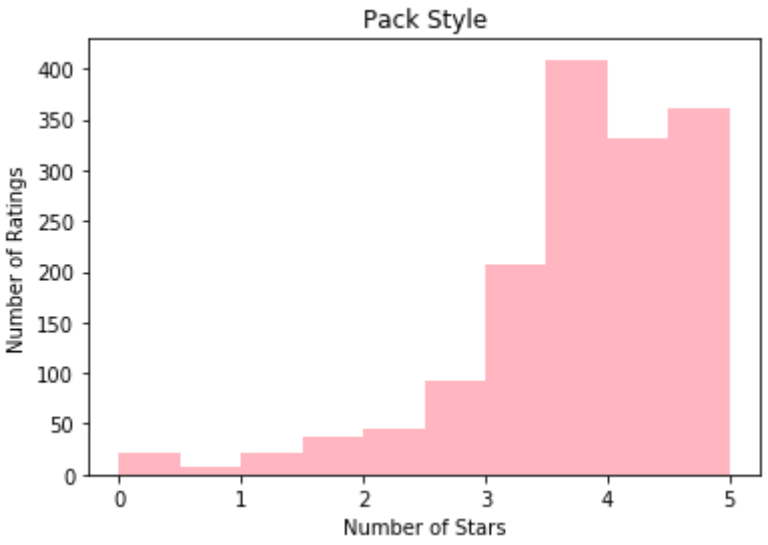


Rys. 4.5. Wykres stosunku liczby zjedzonych ramenów do typów opakowań.

Powyższy wykres przedstawia, że najczęściej jedzony ramen jest w opakowaniu typu “pack”.

```
pack_freq = data['Stars'].loc[data['Style'] == 'pack'].astype(float)
pack_freq.plot(kind='hist',color='lightpink')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
plt.title('Pack Style')
plt.show()
```

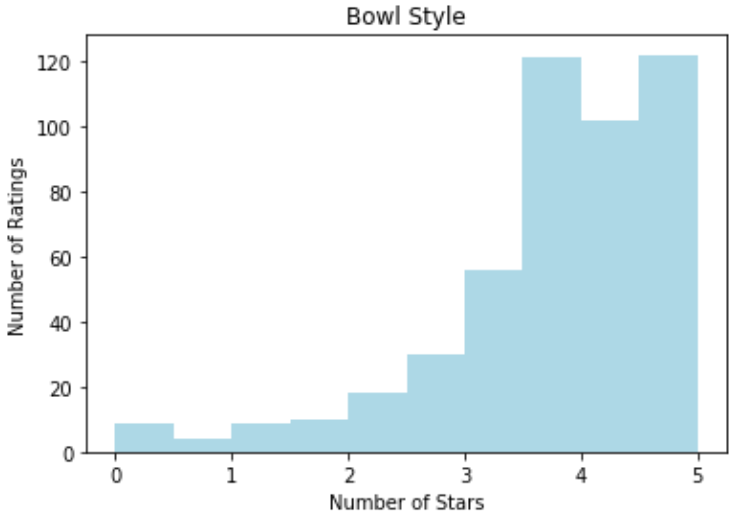
Rys. 4.6. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “pack”.



Rys. 4.7. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “pack”.

```
bowl_freq = data['Stars'].loc[data['Style'] == 'bowl'].astype(float)
bowl_freq.plot(kind='hist',color='lightblue')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
plt.title('Bowl Style')
plt.show()
```

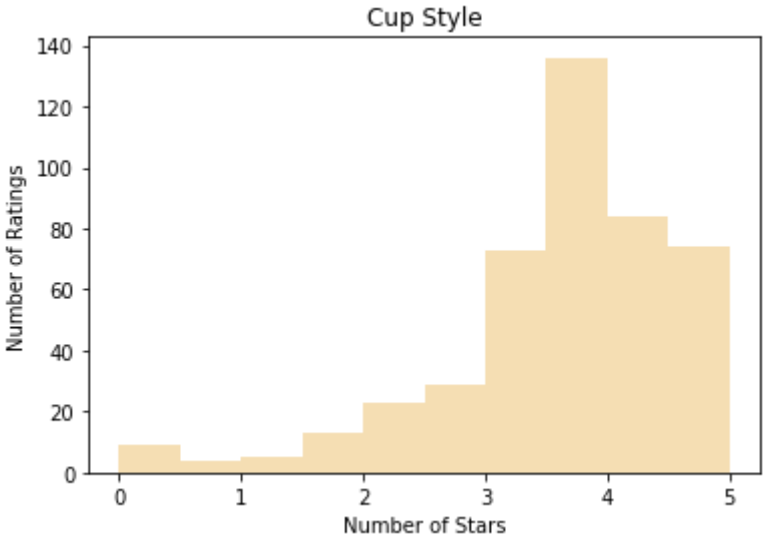
Rys. 4.8. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “bowl”.



Rys. 4.9. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “bowl”.

```
cup_freq = data['Stars'].loc[data['Style'] == 'cup'].astype(float)
cup_freq.plot(kind='hist',color='wheat')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
_ = plt.title('Cup Style')
plt.show()
```

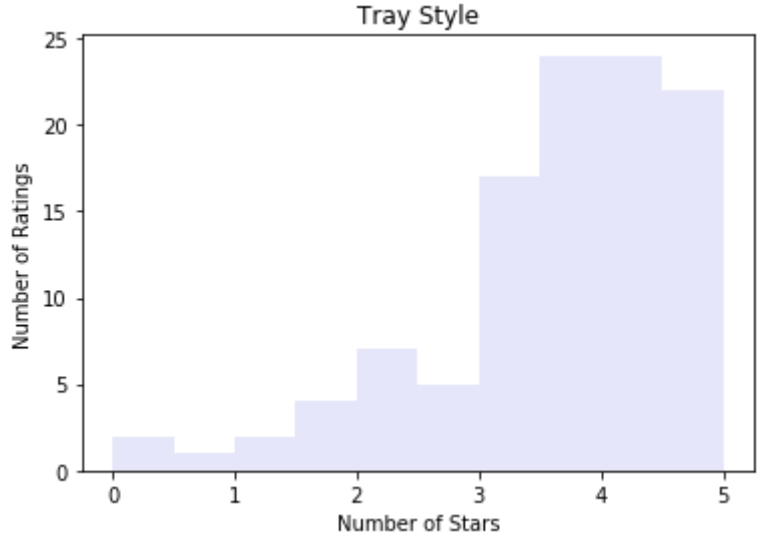
Rys. 4.10. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “cup”.



Rys. 4.11. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “cup”.

```
tray_freq = data['Stars'].loc[data['Style'] == 'tray'].astype(float)
tray_freq.plot(kind='hist',color='lavender')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
plt.title('Tray Style')
plt.show()
```

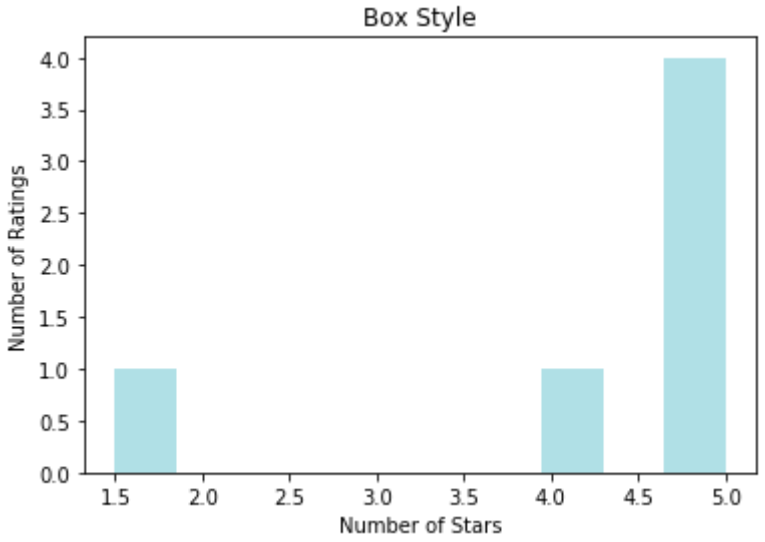
Rys. 4.12. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “tray”.



Rys. 4.13. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “tray”.

```
box_freq = data['Stars'].loc[data['Style'] == 'box'].astype(float)
box_freq.plot(kind='hist',color='powderblue')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
plt.title('Box Style')
plt.show()
```

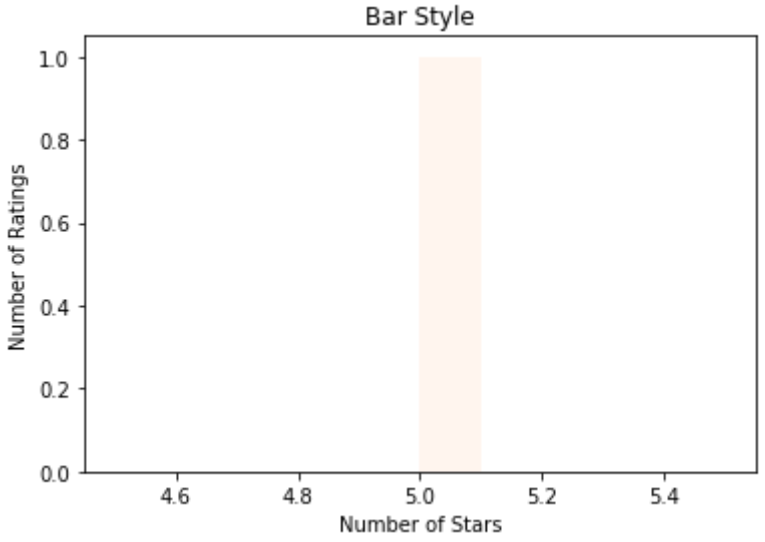
Rys. 4.14. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “box”.



Rys. 4.15. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “box”.

```
bar_freq = data['Stars'].loc[data['Style'] == 'bar'].astype(float)
bar_freq.plot(kind='hist',color='seashell')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
plt.title('Bar Style')
plt.show()
```

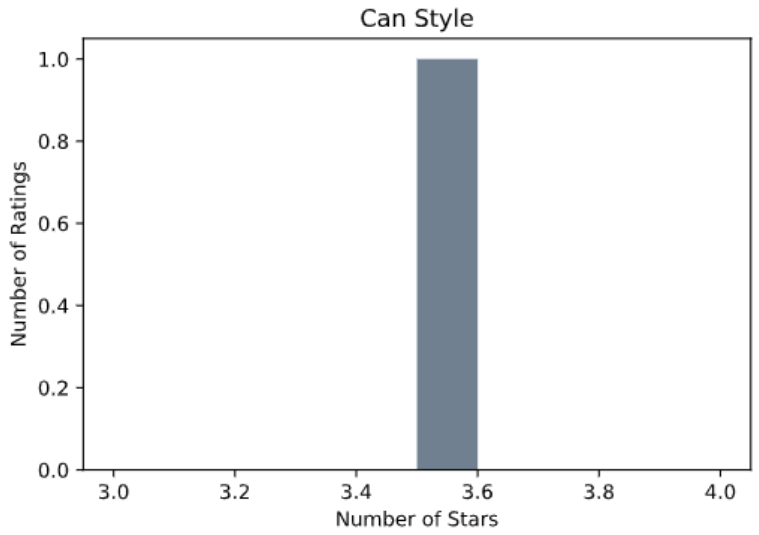
Rys. 4.16. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “bar”.



Rys. 4.17. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “bar”.

```
can_freq = data['Stars'].loc[data['Style'] == 'can'].astype(float)
can_freq.plot(kind='hist',color='slategray')
plt.xlabel('Number of Stars')
plt.ylabel('Number of Ratings')
plt.title('Bar Style')
plt.show()
```

Rys. 4.18. Wyświetlanie wykresu stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “can”.



Rys. 4.19. Wykres stosunku wystawionych ocen z przedziału <0;5> do ilości ocen dla typu opakowania “can”.

Frequency of stars given in each *country*

```
c_dicts = []
c_names = []
countries = data.Country.unique()
for c in countries:
    c_dicts.append(dict(data.Style.loc[data.Country == c].value_counts()))
    c_names.append(c)
```

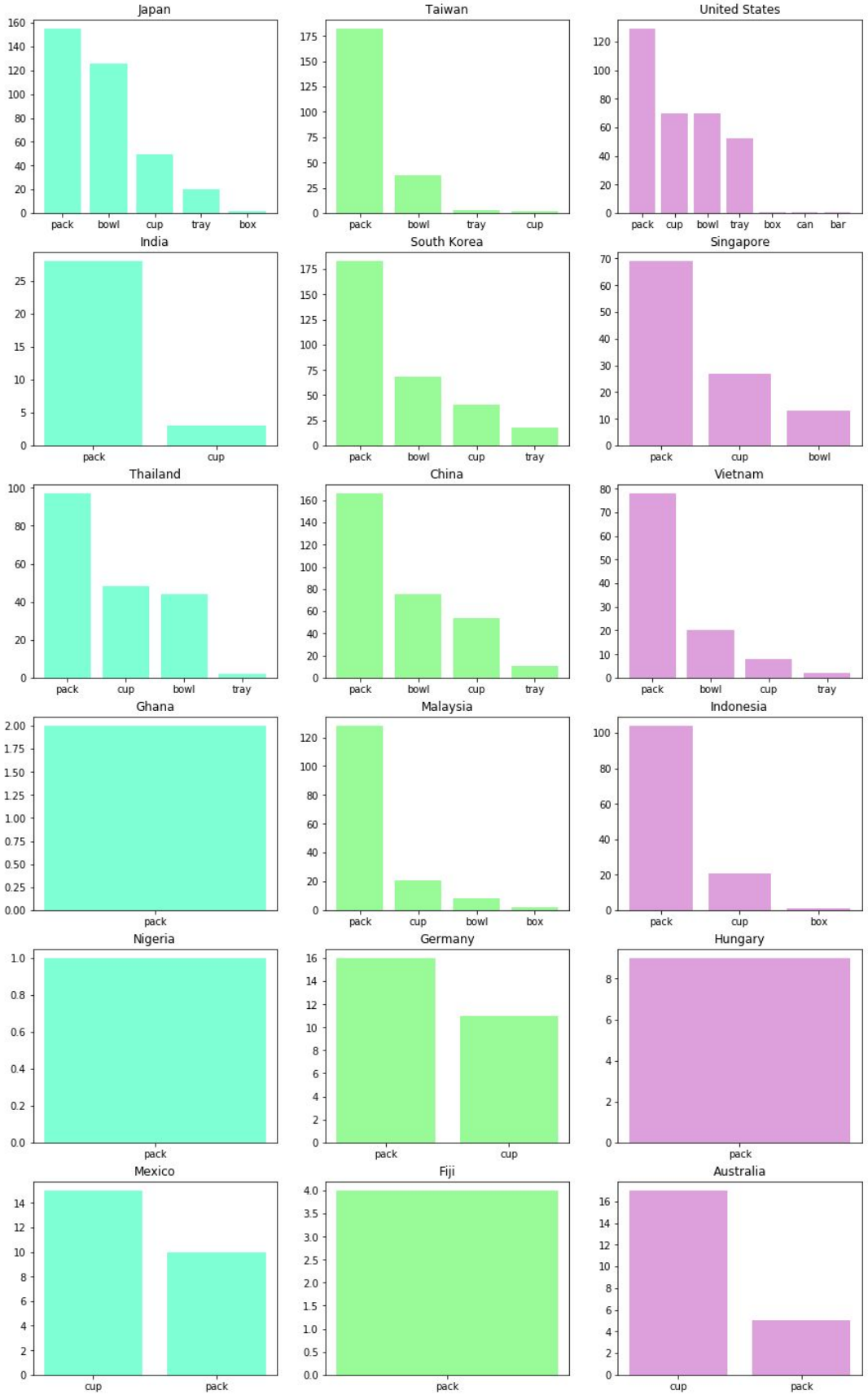
Rys. 4.20. Tworzenie słownika z danymi do wykresów.

Na potrzeby kolejnych wykresów został stworzony słownik zawierający styl ramenu oraz ilość recenzji dla danego kraju.

```
def get_x(country_dict):
    x = []
    for key,value in country_dict.items():
        x.append(key)
    return x

def get_y(country_dict):
    y = []
    for key,value in country_dict.items():
        y.append(value)
    return y

colors = ['aquamarine', 'palegreen', 'plum']

plt.figure(figsize=(15,25), facecolor='white')
plot_number = 1
for i in range(18):
    ax = plt.subplot(6, 3, plot_number)
    ax.bar(get_x(c_dicts[i]), get_y(c_dicts[i]), color=colors[i%3])
    ax.set_title(c_names[i])
    plot_number = plot_number + 1

plt.figure(figsize=(15,25), facecolor='white')
plot_number = 1
for i in range(19,len(data.Country.unique())):
    ax = plt.subplot(6, 3, plot_number)
    ax.bar(get_x(c_dicts[i]), get_y(c_dicts[i]), color=colors[i%3])
    ax.set_title(c_names[i])
    plot_number = plot_number + 1
```

Rys. 4.21. Generowanie wykresów stosunku ilości ramenów w danym kraju do rodzaju opakowania.

Następnie przy użyciu funkcji `get_x()` oraz `get_y()` są generowanie etykiety osi X oraz odpowiednie wartości przedstawione na osi Y.



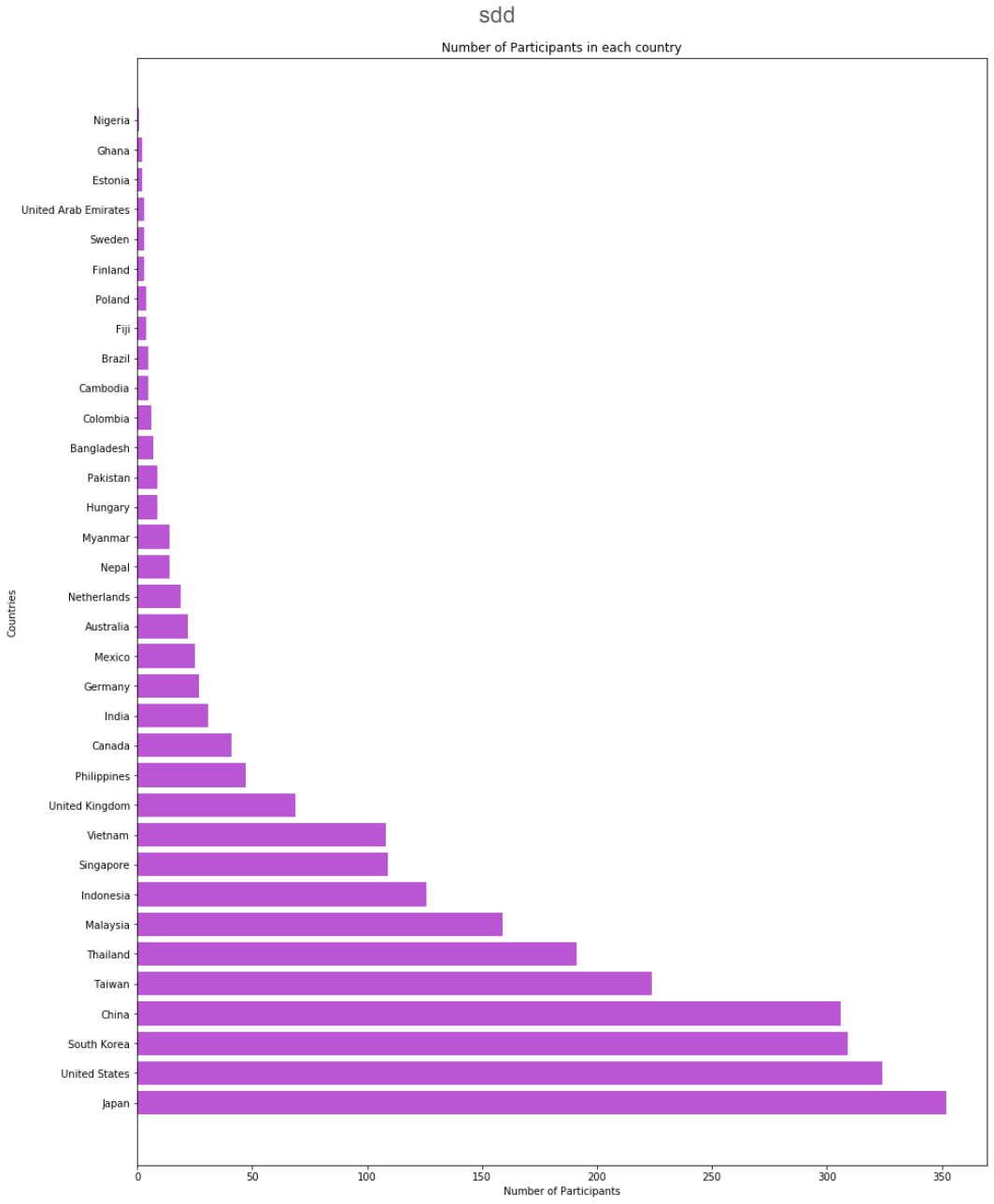
Rys. 4.22.Wykresy stosunku ilości ramenów do opakowań w danym kraju.

Powyższe wykresy przedstawiają popularność każdego ze stylów ramenu w danym kraju.

```
country_counts = dict(data.Country.value_counts())
countries = get_x(country_counts)
counts = get_y(country_counts)
```

```
plt.figure(figsize=(15, 20))
plt.barh(countries,counts, color='mediumorchid')
plt.xlabel('Number of Participants')
plt.ylabel('Countries')
_ = plt.title('Number of Participants in each country')
```

Rys. 4.23. Wyświetlanie wykresu stosunku kraju do liczby osób uczestniczących w ankiecie.



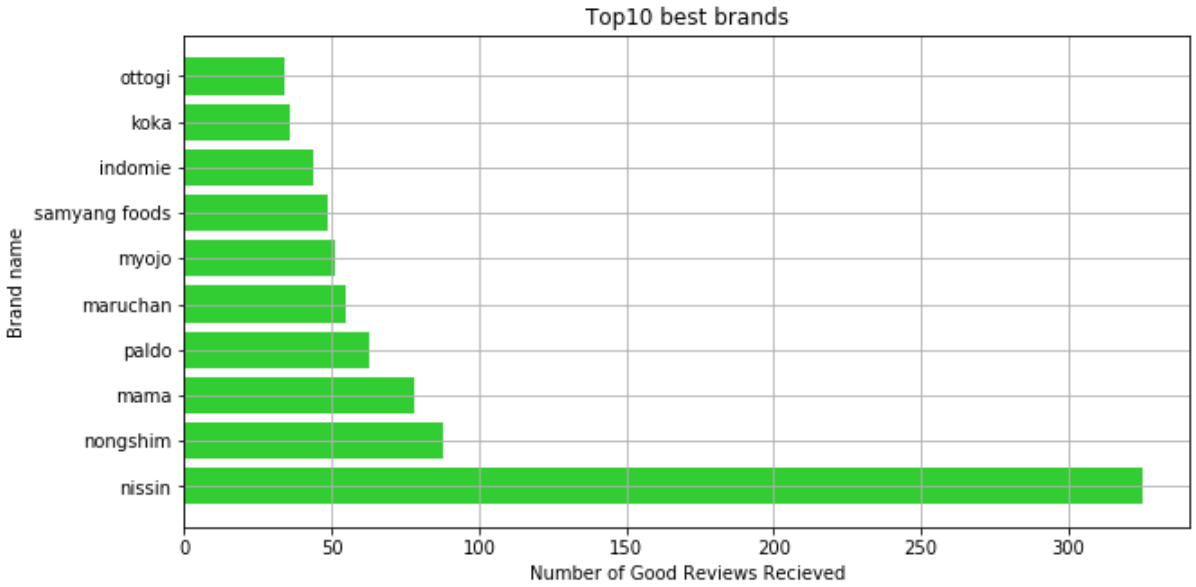
Rys. 4.24. Wykres stosunku kraju do liczby osób uczestniczących w ankiecie

Z wykresu można odczytać, że w rejonie powstania ramenu było najwięcej uczestników.

Top 10 best *Brands*

```
best_brands = dict(data.Brand.loc[data.Stars.astype(float) > 3.0].value_counts())
brand_names = list(best_brands.keys())
brand_apperance_count = list(best_brands.values())
plt.figure(figsize=(10, 5))
plt.barh(brand_names[:10],brand_apperance_count[:10], color='limegreen')
plt.xlabel('Number of Good Reviews Recieved')
plt.ylabel('Brand name')
plt.grid(True)
_ = plt.title('Top10 best brands')
```

Rys. 4.25. Wyświetlanie wykresu stosunku marki ramenu do ilości pozytywnych ocen.



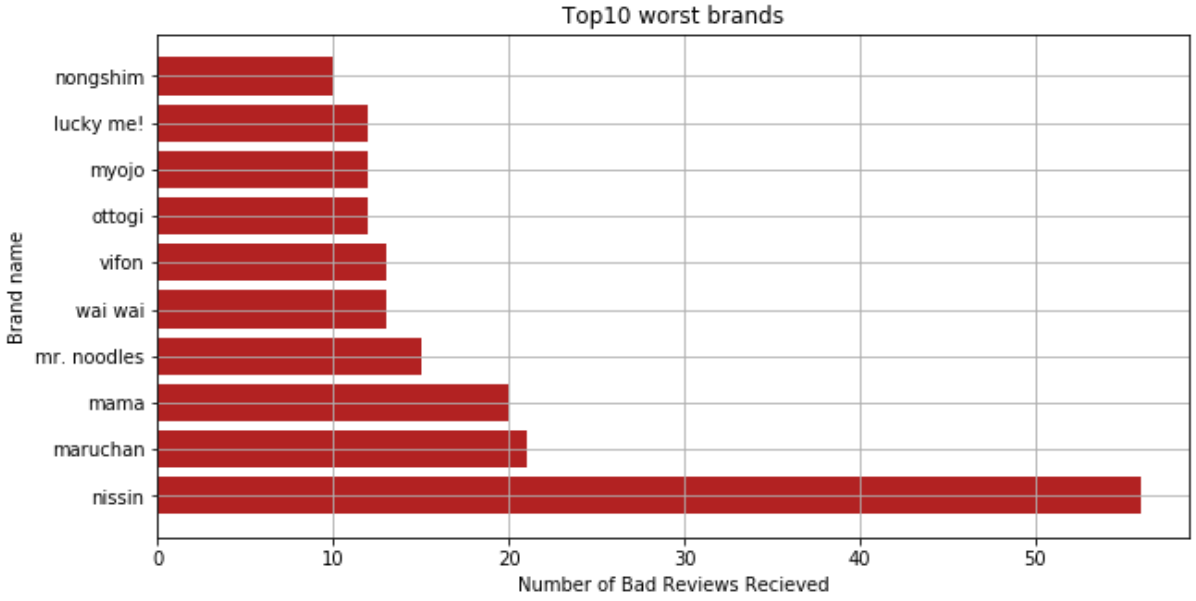
Rys. 4.26. Wykres stosunku marki ramenu do ilości pozytywnych ocen.

Z wykresu można odczytać, że największą popularnością pozytywnych ocen może się cieszyć ramen firmy Nissin. Wynoszącą ponad 300 pozytywnych ocen ich produktów.

Top 10 worst *Brands*

```
worst_brands = dict(data.Brand.loc[data.Stars.astype(float) <= 3.0].value_counts())
brand_names = list(worst_brands.keys())
brand_apperance_count = list(worst_brands.values())
plt.figure(figsize=(10, 5))
plt.barh(brand_names[:10],brand_apperance_count[:10], color='firebrick')
plt.xlabel('Number of Bad Reviews Recieved')
plt.ylabel('Brand name')
plt.grid(True)
_ = plt.title('Top10 worst brands')
```

Rys. 4.27. Wyświetlanie wykresu stosunku marki ramenu do ilości negatywnych ocen.



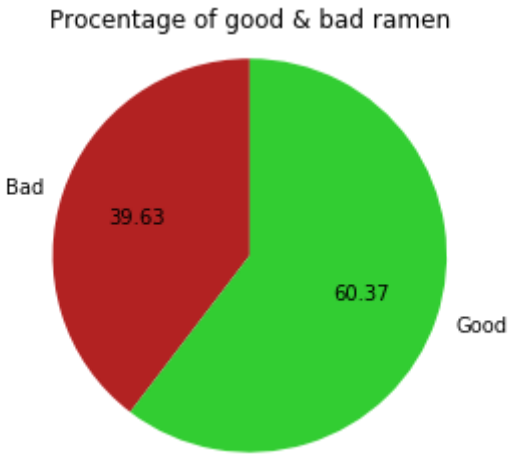
Rys. 4.28. Wykres stosunku marki ramenu do ilości negatywnych ocen.

Z wykresu wynika, że najwięcej negatywnych ocen miał ramen firmy Nissin wynoszących ponad 50.

Procentage of good and bad ramen

```
x = [len(data.Brand.loc[data.Stars.astype(float) <= 3.0].value_counts()),len(data.Brand.loc[data.Stars.astype(float) > 3.0].value_counts())]
plt.pie(x, labels=['Bad', 'Good'], autopct='%1.2f', startangle=90, colors=['firebrick','limegreen'])
plt.axis('equal')
_ = plt.title('Procentage of good & bad ramen')
```

Rys. 4.29. Wyświetlanie wykresu stosunku procentowego ramenu ocenianego pozytywnie do ramenu ocenianego negatywnie.



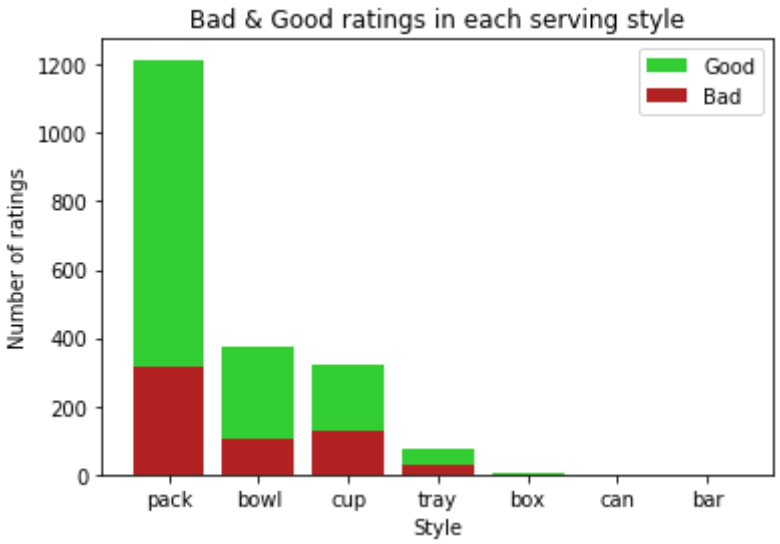
Rys. 4.30. Wykres stosunku procentowego ramenu ocenianego pozytywnie do ramenu ocenianego negatywnie.

Z wykresu wynika, że prawie 60% recenzowanych ramenów jest pozytywnie ocenianych, czyli z oceną wyższą niż 3 gwiazdki, a negatywnie tylko niecałe 40%, z oceną niższą niż 3 gwiazdki.

Number of Bad and Good ratings in each serving style

```
ax = plt.subplot(111)
good = ax.bar(get_x(data.Style.loc[data.Stars.astype(float) > 3.0].value_counts()),get_y(data.Style.loc[data.Stars.astype(float) > 3.0].value_
bad = ax.bar(get_x(data.Style.loc[data.Stars.astype(float) <= 3.0].value_counts()),get_y(data.Style.loc[data.Stars.astype(float) <= 3.0].value_
ax.legend((good, bad), ('Good', 'Bad'))
plt.xlabel('Style')
plt.ylabel('Number of ratings')
plt.title('Bad & Good ratings in each serving style')
plt.show()
```

Rys. 4.31. Wyświetlanie stosunku pozytywnych i negatywnych ocen ramenu w każdym stylu podania.



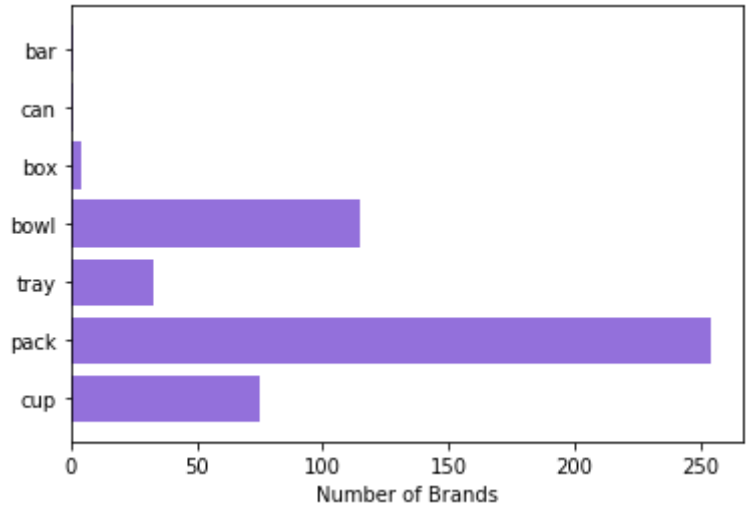
Rys. 4.32. Wykres stosunku pozytywnych i negatywnych ocen ramenu w każdym stylu podania.

Number of brands serving ramen in specyfic style

```
choices = {style: len(data.Brand.loc[data.Style == style].unique()) for style in data.Style.unique()}

plt.barh(get_x(choices),get_y(choices),color='mediumpurple')
plt.xlabel('Number of Brands')
plt.show()
```

Rys. 4.33. Wyświetlanie liczby firm produkujących dany rodzaj ramenu.

Na potrzeby poniższego wykresu został sporządzony słownik zawierający jako klucz styl opakowania, a przypisana jemu wartość odpowiada liczbie firm produkujących ramen w danym typie opakowania.



Rys. 4.34. Wykres stosunku rodzaju ramenu do ilości firm go produkujących.

Wykres przedstawia ilość firm produkujących ramen w danym opakowaniu. Biorąc pod uwagę dane z wykresu, wynika z niego, że najwięcej produkuje się ramenu w opakowaniu typu “pack”.

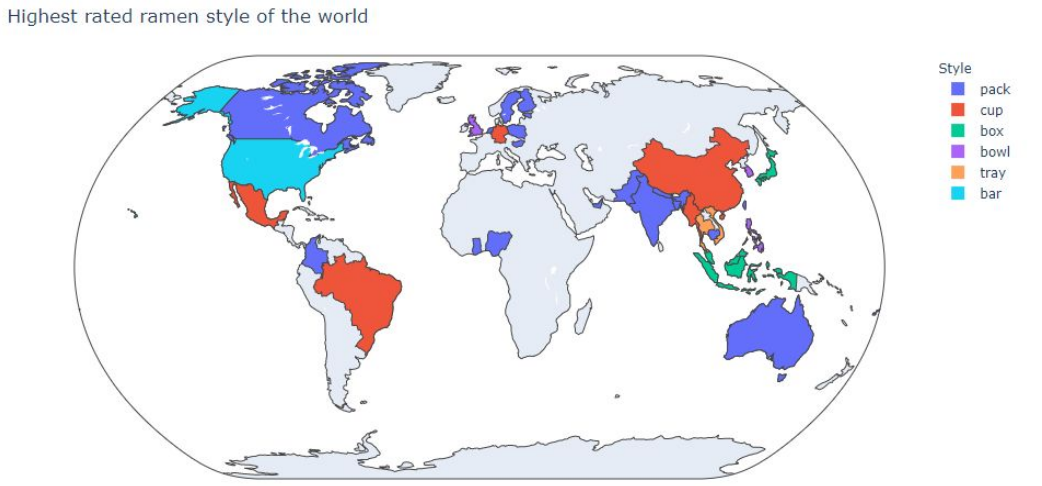
```
world_df = pd.DataFrame()
world_df['Country'] = data.Country.sort_values().unique()
world_df['Style'] = pd.Series(map(lambda x: data[data.Country.isin([x])].groupby('Style').Stars.mean().sort_values(ascending=Fal
world_df['Avg rating'] = pd.Series(map(lambda x: round(data[data.Country.isin([x])].groupby('Style').Stars.mean().sort_values(as
world_df['Popular style'] = pd.Series(map(lambda x : data.Style.where(data.Country == x).value_counts().idxmax(),data.sort_value
world_df['ISO'] = pd.Series(map(lambda x: pc.country_name_to_country_alpha3(x),data.Country.sort_values().unique()))
world_df['Continent'] = pd.Series(map(lambda x: pc.convert_continent_code_to_continent_name(pc.country_alpha2_to_continent_code(
```

Rys. 4.35. Tworzenie zestawu danych do wyświetlenia na mapie.

Na potrzeby wyświetlania informacji na mapie został stworzony specjalny zestaw danych zawierający między innymi ISO danego kraju czy średnią ocen danego stylu ramenu.

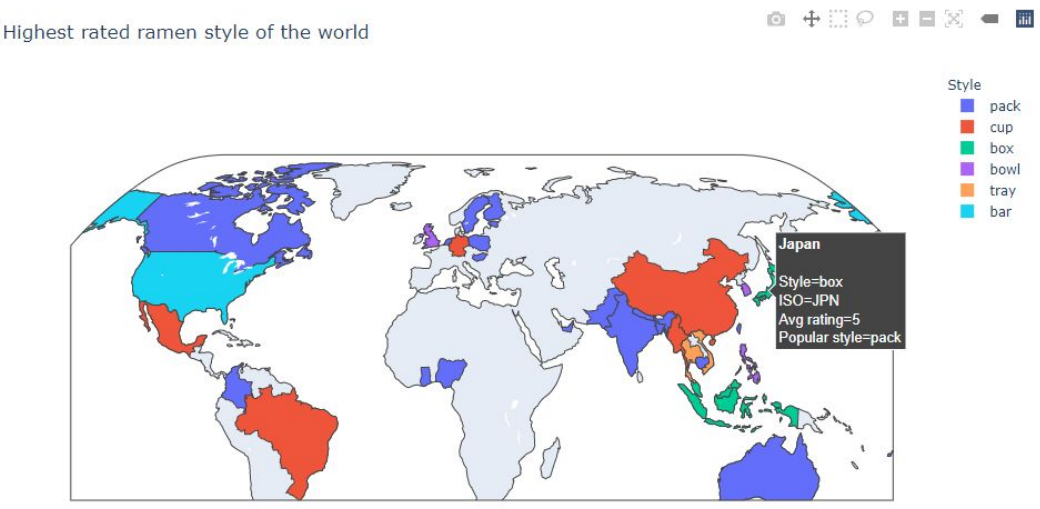
```
fig = px.choropleth(world_df, locations="ISO", color="Style", hover_name='Country',hover_data=['Style','Avg rating', 'Popular st
fig.update_layout(title_text='Highest rated ramen style of the world')
fig.show()
```

Rys. 4.36. Wyświetlenie mapy Świata z uprzednio przygotowanymi danymi.



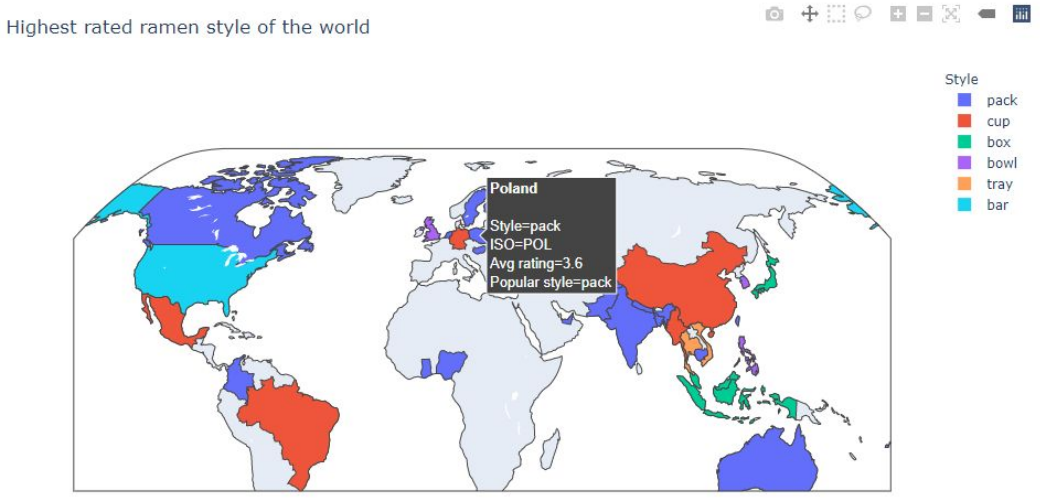
Rys. 4.37. Mapa świata przedstawiająca popularność danego stylu podania ramenu.

Wygenerowana mapa świata przedstawia podział świata ze względu na popularność danego stylu ramenu.



Rys. 4.38. Mapa świata przedstawiająca średnią ocenę dla Japonii.

Powyżej przedstawiona mapa świata pokazuje statystykę dla Japonii, w której najbardziej popularnym ramenem jest ramen w opakowaniu “pack”, zaś najwyższą ocenę dostał ramen w stylu “box” wynoszącą średnio 5 gwiazdek.



Rys. 4.39. Mapa Świata przedstawiająca średnią ocenę dla Polski.

Powyżej przedstawiona mapa świata pokazuje statystykę dla Polski, w której najbardziej popularnym ramenem, również i najwyżej ocenianym jest ramen w stylu “pack” a jego średnia ocen wynosi 3.6 gwiazdki.

4.3 Uczenie maszynowe

4.3.1 Regresja liniowa za pomocą PyTorch

PyTorch to biblioteka do uczenia maszynowego. Została wykonana prosta regresja liniowa za pomocą PyTorch.

Przekonwertowane zostały trzy kolumny: `Country`, `Brand`, `Style` na zmienne fikcyjne. Konwersja do zmiennej fikcyjnej została wykonana przy użyciu funkcji `get_dummies()`.

```
Country = pd.get_dummies(data['Country'], prefix='Country', drop_first=True)
Brand = pd.get_dummies(data['Brand'], prefix='Brand',drop_first=True)
Style = pd.get_dummies(data['Style'], prefix='Style',drop_first=True)
```

Rys. 4.40. Konwersja na zmienne fikcyjne kolumn “Country”, “Brand” oraz “Style”.

Cechy, które były danymi kategorycznymi, są reprezentowane przez zmienne fikcyjne (0 i 1). Jako ostatni krok wstępnego przetwarzania dane zostały podzielone na ilość cech (X) i cel (y).

Następnie PyTorch został użyty do wytrenowania modelu do analizy regresji wielokrotnej. Zaimportowany został moduł sieci neuronowej `torch.nn`, zaś do obsługi regresji liniowej został użyty `nn.Linear()`.

```
m = nn.Linear(391, 1)
```

Rys. 4.41. Użycie funkcji do obliczenia regresji liniowej.

`Linear()`, jako pierwszy argument przekazuje całkowitą liczbę danych uczących. Drugim argumentem jest rozmiar danych wyjściowych.

Następnie ustawiane są straty wynikające z regresji liniowej i funkcji optymalizacji. `MSELoss()` została użyta jako funkcja straty i `SGD()` jako funkcja optymalizacji. `MSELoss()` oznacza średni kwadrat błędu, a `SGD()` oznacza stochastyczne zejście w gradiencie.

```
loss = nn.MSELoss()

opt = torch.optim.SGD(m.parameters(), lr=0.6)
```

Rys. 4.42. Użycie funkcji straty oraz funkcji optymalizacji.

W argumencie funkcji `SGD()` znajduje się `lr`, jest to współczynnik maszynowego uczenia się modelu. Współczynnik maszynowego uczenia został ustawiony na 0,6.

Następnie model liniowy był uczony na podstawie danych wejściowych. Kod do wykonania szkolenia podzielony jest na 4 kroki:

```
for dataepoch in range(1000):
    datain = torch.from_numpy(X)
    datatrgs = torch.from_numpy(y)

    dataout = m(datain)
    cost = loss(dataout, datatrgs)

    opt.zero_grad()
    cost.backward()
    opt.step()

    if (dataepoch+1) % 100 == 0:
        print ('Epoch for ramen-ratings.csv [{}/{}]\nLoss: {:.4f}'.format(dataepoch+1, 1000, cost.item()))
```

Rys. 4.43. Trenowanie modelu za pomocą moduły torch.nn.

- Ustawiona została liczba iteracji (epoki) na 1000
- Została skonwertowana liczba cech i cel z tablicy Numpy na tensor za pomocą `from_numpy()`[2]
- Szacowana wartość wyjściowa (output), porównano z wartością rzeczywistą (celami) za pomocą funkcji kosztu
- Zostały wykonane obliczenia kosztów i przetwarzania wstecznej propagacji błędów.

```
Epoch for ramen-ratings.csv [100/1000]
Loss: 0.7978
Epoch for ramen-ratings.csv [200/1000]
Loss: 0.7490
Epoch for ramen-ratings.csv [300/1000]
Loss: 0.7194
Epoch for ramen-ratings.csv [400/1000]
Loss: 0.6989
Epoch for ramen-ratings.csv [500/1000]
Loss: 0.6838
Epoch for ramen-ratings.csv [600/1000]
Loss: 0.6719
Epoch for ramen-ratings.csv [700/1000]
Loss: 0.6623
Epoch for ramen-ratings.csv [800/1000]
Loss: 0.6544
Epoch for ramen-ratings.csv [900/1000]
Loss: 0.6477
Epoch for ramen-ratings.csv [1000/1000]
Loss: 0.6420
```

Rys. 4.44. Wyświetlenie funkcji straty dla poszczególnych epok.

Jak widać, pierwsze 100 strat (koszt) wyniosło 0,7978, ale koszt tysięcznej epoki został zmniejszony do 0,6420. Wprowadzone zostały cechy do wytrenowanego modelu i została sprawdzona szacowana wartość.

```
y_predict = m(torch.from_numpy(X)).data.numpy()

print(y_predict[0:5])
print(y[0:5])
```

Rys. 4.45. Wyświetlenie predykcji oraz wartości rzeczywistej.

`y_pred` to wynik oszacowania uzyskany przez model regresji liniowej skonstruowany przy użyciu PyTorch. Z drugiej strony y to poprawna etykieta danych, czyli rzeczywista wartość.

```
[[4.120036 ]
 [2.9402654]
 [3.4925938]
 [3.4295306]
 [3.7062385]]
[[3.8]
 [1. ]
 [2.2]
 [2.8]
 [3.8]]
```

Rys. 4.46. Wartość predykcji oraz wartości rzeczywistej.

Istnieją różne metody oceny modelu regresji, jednakże w tym badaniu został użyty jeden z najbardziej podstawowych wskaźników, „średni kwadrat błędu (MSE)".

```
metrics.mean_squared_error(y, y_predict)
```

```
0.6419483
```

Rys. 4.47. Wyświetlenie wartości MSE (Mean Square Error) - błąd średniokwadratowy.

MSE wynosi teraz 0,64. Krótko mówiąc, MSE jest wskaźnikiem błędu między wartością oszacowaną przez model a wartością rzeczywistą. Im niższa wartość MSE, tym większe prawdopodobieństwo, że model zgadnie.

4.3.2 AdaBoost, Metoda "łokcia" i algorytm centroidów

Dla porównania zostały użyte również klasyfikator AdaBoost oraz metoda "łokcia" (Eblow Method) do predykcji celu `y_pred` wraz z algorytmem centroidów (k-means clustering).

AdaBoost to podstawowy algorytm, w której z dużej liczby słabych klasyfikatorów można otrzymać jeden lepszy. AdaBoost działa w ten sposób, że w kolejnych iteracjach trenuje, a następnie mierzy błąd wszystkich dostępnych słabych klasyfikatorów. W każdej następnej iteracji "ważność" źle zakwalifikowanych obserwacji jest zwiększana, tak że klasyfikatory zwracają na nie większą uwagę.[3]

Metoda "łokcia" to metoda interpretacji i weryfikacji spójności wewnątrz analizy skupień zaprojektowany do pomocy w znalezieniu odpowiedniej liczby klastrów w zbiorze danych . W metodzie "łokcia" (elbow method), leży taki dobór klastrów, aby łączna wariancja wszystkich

klastrów była minimalna. Wariancję tę określa się poprzez sumę kwadratów odległości pomiędzy każdą z obserwacji, a środkiem danego klastra.[4]

Algorytm centroidów jest jednym z algorytmów stosowanym w analizie skupień, wykorzystywanym m.in. w kwantyzacji wektorowej. Algorytm nazywany jest także algorytmem klastrowym lub – od nazwisk twórców Linde, Buzo i Graya – algorytmem LBG. Celem algorytmu jest przypisanie do wektorów kodowych n-wymiarowych wektorów danych, przy jak najmniejszym średnim błędzie kwantyzacji.[5]

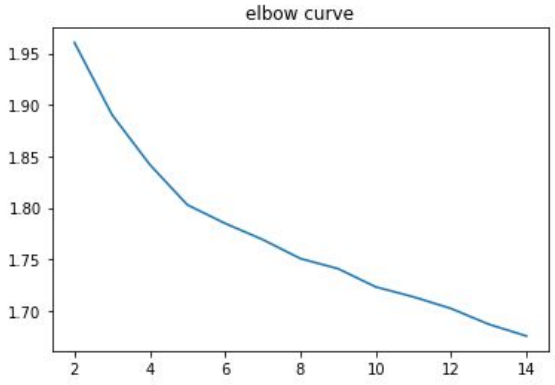
Na początek zostały utworzone zmienne fikcyjne dla kolumn `Brand`, `Variety`, `Style`, `Country` oraz dodana została kolumna `Stars`. Wykorzystując nienadzorowaną metodę k-średnich zostały obliczone wszystkie klastry.

```
dummies= pd.get_dummies(data[["Brand","Variety","Style","Country"]])

stars = data[["Stars"]]
data3 = pd.concat((dummies, stars), axis=1)
```

Rys. 4.48. Konwersja kolumn “Brand”, “Variety”, “Style” oraz “Country” do zmiennych fikcyjnych.

Algorytm centroidów wybrał N wektorów kodowych i określił maksymalny błąd kwantyzacji. W każdej kolejnej iteracji został uzyskany dokładniejszy średni błąd kwantyzacji, zaś centroidy i grupy wektorów zostały przypisane do wektorów kodowych, aż do ostatniej iteracji i uzyskania wymaganej dokładności.

```
res=list()
n_cluster = range(2,15)
for n in n_cluster:
    kmeans = KMeans(n_clusters = n)
    kmeans.fit(data3)
    res.append(np.average(np.min(cdist(data3, kmeans.cluster_centers_, 'euclidean'), axis=1)))

plt.plot(n_cluster, res)
plt.title('elbow curve')

kmeans_model=KMeans(n_clusters=7,init='k-means++', max_iter=300, n_init=10, random_state=0).fit(data3);
results = kmeans_model.predict(data3);
centroids = kmeans_model.cluster_centers_;
kmeans_n = pd.DataFrame(data = results);
data["Score"] = kmeans_n
res= kmeans_model.__dict__

res2 = pd.DataFrame.from_dict(res, orient='index')
```



Rys. 4.49. Użycie algorytmu centroidów i metody “łokcia”.

Następnie na podstawie klastrów została narysowana krzywa dla metody “łokcia”[2]

```
data4 = data3
H_cluster = linkage(data4,'ward')
cluster_2 = fcluster(H_cluster, 5, criterion='maxclust')
cluster_Hierarchical = pd.DataFrame(cluster_2)
data4.to_csv("tree.csv")
```

Rys. 4.50. Łączenie danych metodą linkage.

Dane zostały połączone metodą linkage z biblioteki SciPy w celu utworzenia hierarchicznego drzewa decyzyjnego. Następnie wygenerowane w ten sposób dane zostały zklasteryzowane i wyeksportowane do pliku `tree.csv`. Potem wszystko zostało zaimportowane do programu Excel. Po przejrzeniu danych uporządkowanych przy pomocy powyższego drzewa hierarchicznego zostały wyciągnięte następujące wnioski:

- najmniej gwiazdek otrzymała marka Nissin głównie w Kanadzie, Tajwanie, USA, Tajlandii, Wielkiej Brytanii i Hong Kongu, co potwierdza pierwotna analiza danych
- kilka gwiazdek <2,5;4> otrzymały produkty na rynku koreańskim i ich lokalne marki
- dużą liczbę gwiazdek <4;5> otrzymały produkty z Tajwanu, Japonii i Malezji, oraz marka Nissin w Korei Południowej

```
X = data3.drop(["Stars"], axis=1)
data3["Stars"] = pd.to_numeric(data3["Stars"], errors='coerce')
y = data3["Stars"].astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
abc = AdaBoostClassifier(n_estimators=50, learning_rate=1)
X_train = X_train.fillna(X_train.mean())
y_train = y_train.fillna(y_train.mean())
X_test = X_test.fillna(X_test.mean())
y_test = y_test.fillna(y_test.mean())
model = abc.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.42764857881136953
```

Rys. 4.51. Trenowanie pierwszego modelu.

Algorytm AdaBoost został użyty do przewidywania punktacji gwiazdek, ponieważ istnieje wiele cech do regresji. Dane zostały użyte w następujący sposób: w pierwszej próbie 70% do uczenia modelu oraz 30% do testowania modelu z użyciem 50 estymatorów. W tym przypadku została otrzymana 42% dokładność. ;-;

```
X = data3.drop(["Stars"], axis=1)
data3["Stars"] = pd.to_numeric(data3["Stars"], errors='coerce')
y = data3["Stars"].astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
abc = AdaBoostClassifier(n_estimators=500, learning_rate=1)
X_train = X_train.fillna(X_train.mean())
y_train = y_train.fillna(y_train.mean())
X_test = X_test.fillna(X_test.mean())
y_test = y_test.fillna(y_test.mean())
model = abc.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.55348837209302323
```

Rys. 4.52. Trenowanie drugiego modelu.

W drugim podejściu po użyciu 500 estymatorów i użyciu danych w następujący sposób: 90% do uczenia modelu oraz 10% do testowania modelu, otrzymana została 55% dokładność.

5. Modelowanie danych - przyjęte założenia, krótki opis metod i obranej metodologii budowania modeli

5.1 Model - PyTorch

Pierwszy model został wykonany z pomocą biblioteki PyTorch. Dane wejściowe zostały podzielone w stosunku 60% do uczenia się i 40% do testowania.

```
opt = torch.optim.SGD(m.parameters(), lr=0.6)
```

Rys. 5.1. Funkcja optymalizacji SDG ze stosunkiem 60% danych przeznaczonych do uczenia się i 40% danych do testowania.

Mean Squared Error (MSE) wyniósł w tym przypadku 0,64, im niższa wartość MSE, tym większe prawdopodobieństwo, że model przewidzi daną wartość.

5.2 Model - AdaBoost, algorytm centroidów

Drugi model został wykonany za pomocą biblioteki SciKit przy użyciu algorytmu boostowania AdaBoost oraz algorytmu centroidów (k-means clustering) oraz wykorzystując metodę "łokcia" (Elbow Method). Dane wejściowe w pierwszym podejściu zostały podzielone w stosunku 70% do uczenia się i 30% do testowania z obsługą 50 estymatorów.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
abc = AdaBoostClassifier(n_estimators=50, learning_rate=1)
```

Rys. 5.2. Funkcja uczenia się wraz z algorytmem boostowania ze stosunkiem 70% danych przeznaczonych do uczenia się i 30% danych do testowania.

Dokładność przewidywania modelu wyniosła 42%.

W drugim podejściu dane wejściowe zostały podzielone w stosunku 90% do uczenia się i 10% do testowania modelu z obsługą 500 estymatorów.

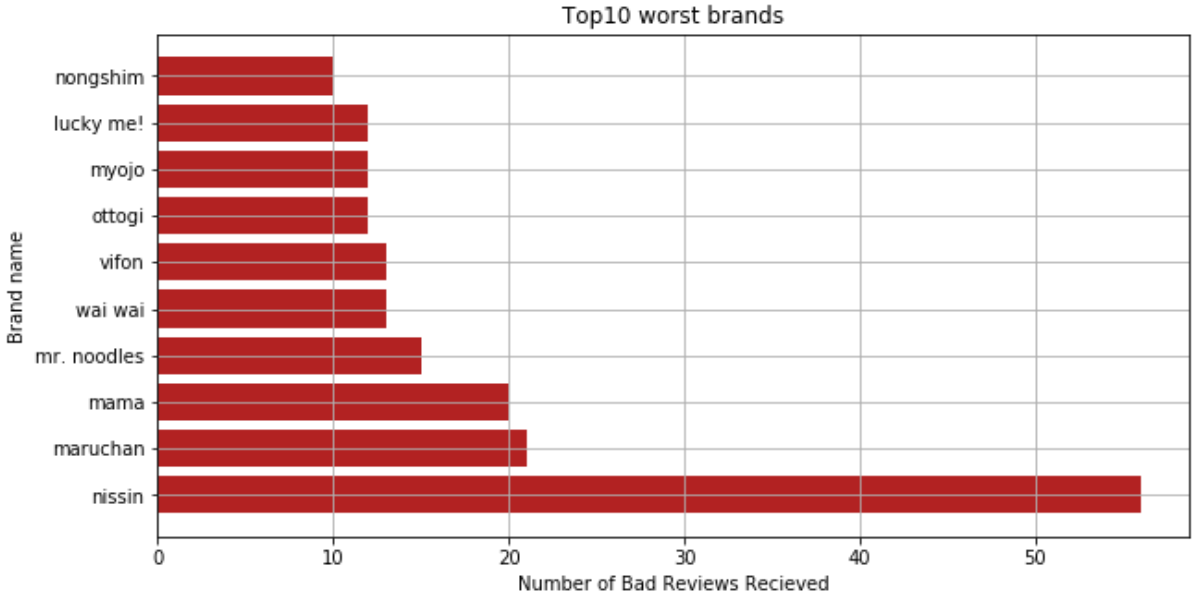
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
abc = AdaBoostClassifier(n_estimators=500, learning_rate=1)
```

Rys. 5.3. Funkcja uczenia się wraz z algorytmem boostowania ze stosunkiem 90% danych przeznaczonych do uczenia się i 10% danych do testowania.

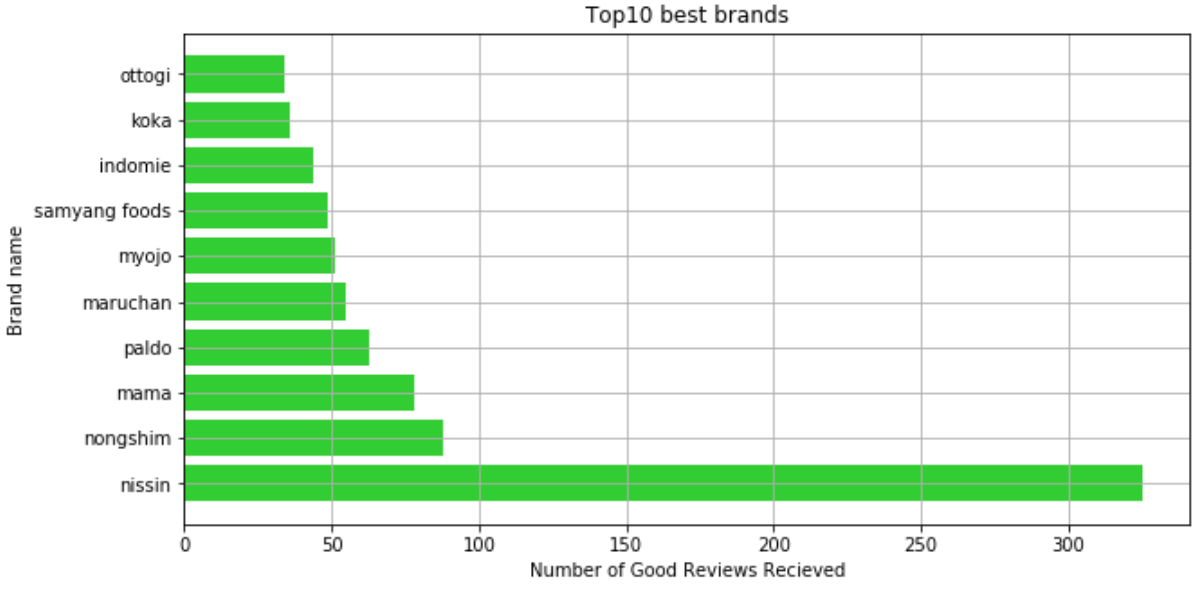
Dokładność przewidywania modelu wyniosła 55%.

6. Rezultaty, wnioski i ich dyskusja:

Z otrzymanych rezultatów wynika, że najlepsza i zarazem najgorsza zupka z proszku jest od producenta Nissin. Przez wszystkich oceniających najczęściej jedzona zupka z proszku była paczkowana w saszetkach (pack). Najczęstszą oceną było 3.5, co pokazuje, że recenzent ramenów ma neutralny stosunek do zupek z proszku.



Rys. 5.1. Wykres stosunku marki ramenu do ilości negatywnych ocen.



Rys. 5.2. Wykres stosunku marki ramenu do ilości pozytywnych ocen.

Średnia ocena ramenu w Polsce wynosi 3.6, zaś w kraju jego pochodzenia - Japonii - 5.0. Zaś średnia ocen dla wszystkich danych wynosi 3.65 gwiazdki. Mediana ocen wynosi 3.8

gwiazdek. Odchylenie standardowe wynosi 1.02 gwiazdki, minimalna ocena ramenu wynosi 0 gwiazdek, zaś maksymalna 5 gwiazdek.

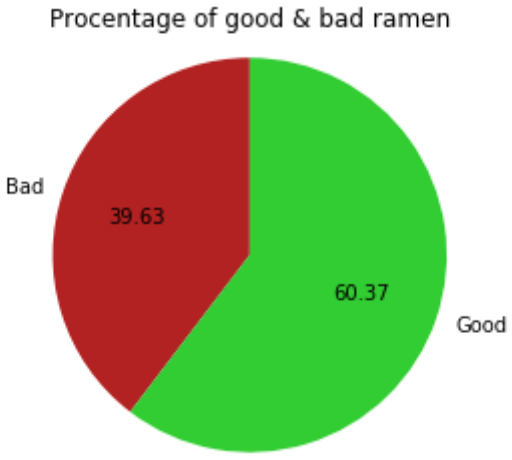
Najmniej gwiazdek otrzymała marka Nissin głównie w Kanadzie, Tajwanie, USA, Tajlandii, Wielkiej Brytanii i Hong Kongu, co potwierdza pierwotna analiza danych.

Kilka gwiazdek <2,5;4> otrzymały produkty na rynku koreańskim i ich lokalne marki.

Dużą liczbę gwiazdek <4;5> otrzymały opakowania w saszetkach (pack), w Tajwanie, Japonii, Malezji pod marką Nissin.

Możemy dzięki temu dojść do wniosków, że marka Nissin jest najpopularniejsza na rynku Azjatyckim, wyłączając z tego zestawienia Koreę Południową, gdzie dominują lokalne marki.

W gotowym przedstawieniu danych - ocen pozytywnych (powyżej 3 gwiazdek) jest 60,37% w porównaniu do ocen negatywnych (poniżej 3 gwiazdek) - 39,63%. Co oznacza, że ocen pozytywnych wystawiono nieznacznie więcej.



Rys. 5.3. Wykres stosunku procentowego ramenu ocenianego pozytywnie do ramenu ocenianego negatywnie.

Porównując trzy modele uczenia maszynowego najwyższą dokładność uzyskał algorytm centroidów wraz z algorytmem boostowania AdaBoost. W jego przypadku dokładność wyniosła 55%.

---

7. Przypisy

[1] pandas.to_numeric — pandas 1.1.1 documentation
[2] torch.from_numpy — PyTorch 1.6.0 documentation
[3] AdaBoost

[4] Optymalna liczba klastrów w metodzie k-średnich w R – QUANT Blog
[5] Algorytm centroidów – Wikipedia, wolna encyklopedia