

# Przygotowanie danych

W pierwszej kolejności zajmę się przygotowaniem danych, tj. oczyszczeniem z brakujących wartości oraz zamianą typów do odpowiedniego formatu. Jednak zanim przejdę do tego kroku, szybkie objaśnienie zbioru danych na którym pracuje

## Zbiór

Dane przedstawiają efekty kampanii marketingowej banku. Telemarketerzy dzwoniли do potencjalnych klientów, próbując ich przekonać do założenia lokaty terminowej. Celem pracy jest stworzenie modelu który na podstawie danych będzie w stanie przewidzieć czy dana osoba zapisze skorzysta z usługi banku.

Krótki opis poszczególnych cech (cechy odnoszą się do stanu osoby do której został wykonany telefon):

1. age – wiek
2. job – wykony zawód
3. marital – stan cywilny
4. education – uzyskane wykształcenie
5. default – czy jest dłużnikiem
6. housing – czy ma kredyt hipoteczny
7. loan – czy ma kredyt konsumpcyjny
8. contact – czy telefon został wykonany na telefon stacjonarny lub komórkowy
9. month – miesiąc w którym została wykonana ostatnia rozmowa
10. day\_of\_week – dzień tygodnia w którym została wykonana ostatnia rozmowa
11. duration – czas rozmów. Zgodnie z sugestiami autorów ta cecha zostanie odrzucona. Jej wartość nie jest znana przed wykonaniem telefonu do klienta, więc nie pomaga w zdeterminowaniu docelowej grupy klientów.
12. campaign – liczba rozmów wykonana podczas analizowanej kampanii
13. pdays – liczba dni od ostatniego kontaktu z klientem podczas poprzedniej kampanii. W przypadku braku kontaktu w poprzedniej kampanii pole przyjmuje wartość 999
14. previous – liczba rozmów podczas poprzednich kampanii
15. poutcome – wynik poprzednich kampanii
16. emp.var.rate – wskaźnik zmienności zatrudnienia – wskaźnik kwartalny
17. cons.price.idx – indeks cen towarów i usług – wskaźnik miesięczny
18. cons.conf.idx – indeks ufności konsumentów – wskaźnik miesięczny
19. euribor3m – referencyjna, trzymiesięczna wysokość oprocentowania depozytów i kredytów na rynku międzybankowym - wskaźnik dzienny
20. nr.employed – wskaźnik zatrudnienia – wartość kwartalna
21. y – rezultat obecnej kampanii

## Poprawność danych

Tuż po załadowaniu danych przy użyciu biblioteki Pandas, dokonuje szybkich oględzin przy użyciu metod `info()` oraz `describe()`.

```
df.info()
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
age                41188 non-null int64
job                40858 non-null object
marital            41108 non-null object
education          39457 non-null object
default            32591 non-null object
housing            40198 non-null object
loan               40198 non-null object
contact            41188 non-null object
month              41188 non-null object
day_of_week        41188 non-null object
duration           41188 non-null int64
campaign           41188 non-null int64
pdays             41188 non-null int64
previous           41188 non-null int64
poutcome           41188 non-null object
emp.var.rate       41188 non-null float64
cons.price.idx     41188 non-null float64
cons.conf.idx      41188 non-null float64
euribor3m          41188 non-null float64
nr.employed        41188 non-null float64
y                  41188 non-null object
dtypes: float64(5), int64(5), object(11)
```

```
df.describe()

```

	age	duration	campaign	pdays	previous	\
count	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	
mean	40.02406	258.285010	2.567593	962.475454	0.172963	
std	10.42125	259.279249	2.770014	186.910907	0.494901	
min	17.000000	0.000000	1.000000	0.000000	0.000000	
25%	32.000000	102.000000	1.000000	999.000000	0.000000	
50%	38.000000	180.000000	2.000000	999.000000	0.000000	
75%	47.000000	319.000000	3.000000	999.000000	0.000000	
max	98.000000	4918.000000	56.000000	999.000000	7.000000	

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	1.570960	0.578840	4.628198	1.734447	72.251528
min	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

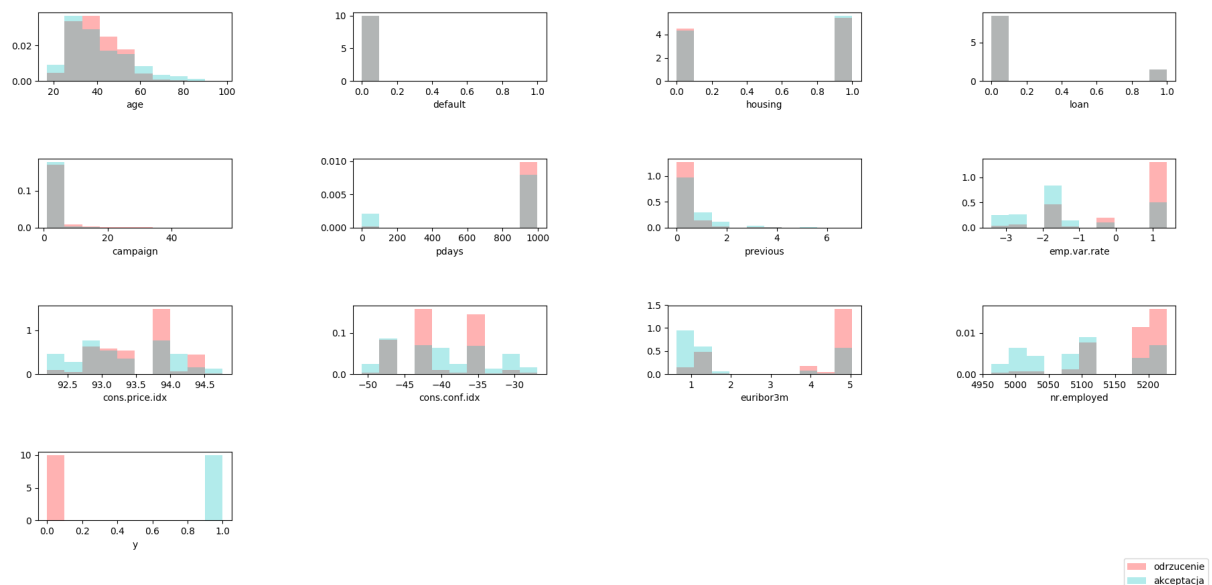
W większości przypadków, brakujące wartości stanowią niewielki procent całości. Postanawiam uzupełnić je wykorzystując w tym celu wartość najpopularniejszą (mod każdej kolumny). Następnie przekształcam typy danych. Tam gdzie mam do czynienia z wartościami „yes” lub „no”, zamieniam je na 1 i 0. Pozostałe nienumeryczne kolumny zamieniam na typ kategorialny.

# Wizualizacja oraz analiza

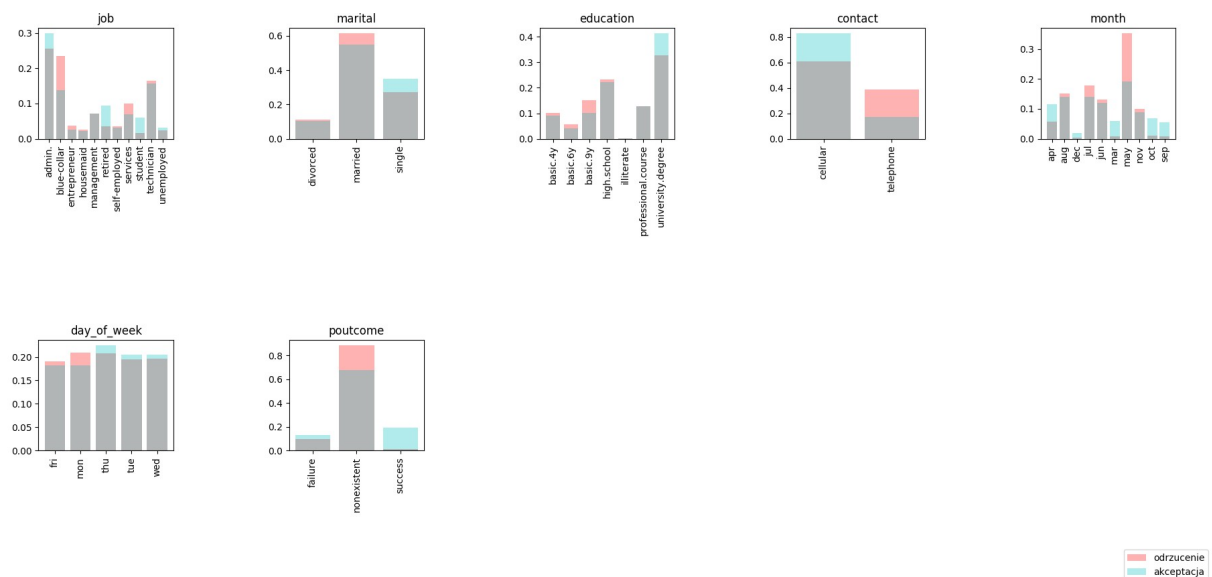
Ten etap rozpoczne od prezentacji danych w formie histogramów (dla typów numerycznych) oraz wykresów słupkowych (dla typów kategoryjnych). Każdy obszar wykresu reprezentuje jedną cechę. Dokonałem również podziału na klientów ze względu na efekt końcowy (skorzystanie z oferty banku lub nie). Ci drudzy są oznaczeni kolorem czerwonym oraz opisani jako „odrzućenie”. Będę używał tej konwencji w reszcie pracy.

```
histogramy = plt.figure('Histogramy')  
rysujHistogramy(df, histogramy)
```

```
słupki = plt.figure('Słupki')  
rysujSłupki(df, słupki)
```

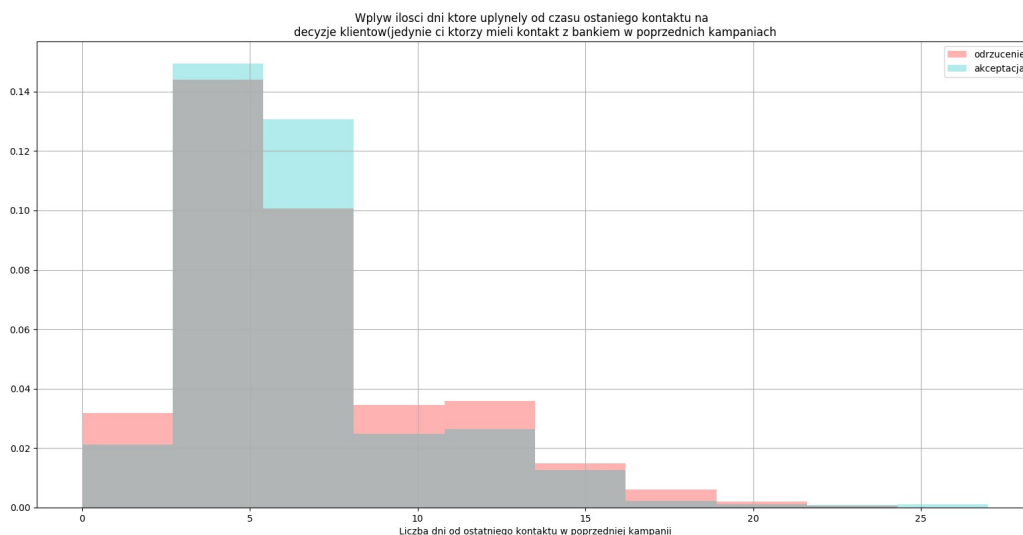


Ilustracja 1: Histogramy



Widać że rozkład cechy *pdays* jest mocno zdeformowany przez wartość 999, która to jest liczbą sztucznie dodaną do zbioru. Na późniejszym etapie będzie potrzebne podzielenie danych na zbiory-kategorie, odpowiadające poszczególnym zakresom wartości. Póki co wyrysuje ten sam histogram raz jeszcze, wykluczając tym razem rekordy z wartością 999, czyli ludzi którzy nie odbyli nigdy rozmowy z bankiem.

```
plt.figure('pdays bez 999')
df.pdays[(df.pdays<999) & (wyb_neg)].hist(bins=10,alpha=0.3, color='r',
      density = True, label='odrzućenie', range=(0,27))
df.pdays[(df.pdays<999) & (wyb_poz)].hist(bins=10,alpha=0.3, color='c',
      density = True, label='akceptacja')
```



### Ilustracja 3: Poprawiony histogram cechy *pdays*

Widać że w większości przypadków ponowny kontakt z klientem nastąpił po około 5 dniach. Po takim okresie skuteczność telemarketerów również była najwyższa. Nie jest to jednak przekonująca zależność. Sprawdzę czy na taki wynik nie mają wpływu inne czynniki. W tym celu policzę jaki procent klientów którzy zaakceptowali ofertę w obecnej kampanii, mieli już do czynienia z bankiem w poprzednich edycjach.

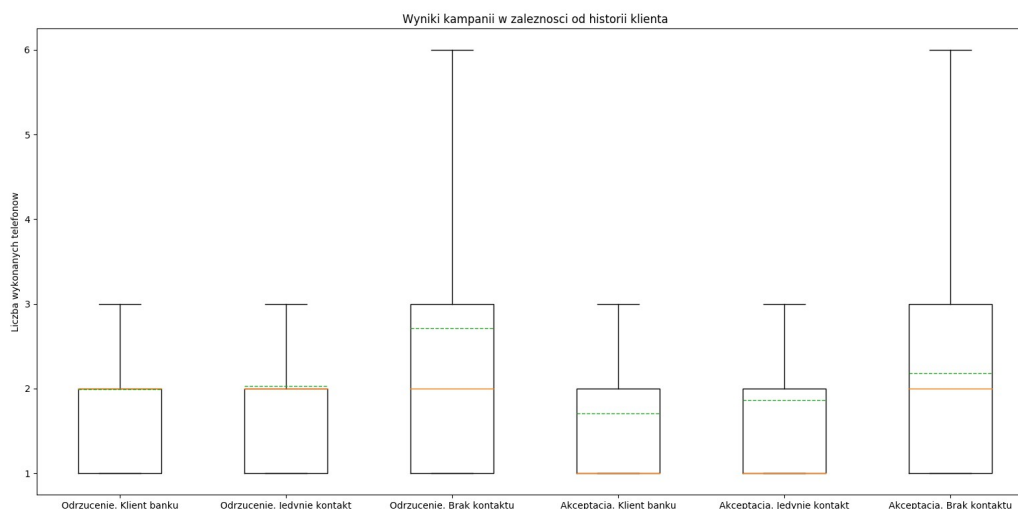
```
Procent klientów którzy zaakceptowali ofertę, mając do czynienia z bankiem
poprzednio: 26.65%
Procent klientów którzy zaakceptowali ofertę, nie mając do czynienia z bankiem
poprzednio: 8.83%
```

Dodatkowo sprawdzę czy sam fakt wcześniejszej styczności z ofertą banku jest wystarczający, a może istotniejszy jest wynik poprzedniej kampanii (czy klient założył lokatę).

```
Procent klientów którzy zaakceptowali ofertę, mając już wcześniej lokatę w
banku: 65.11%
Procent klientów którzy zaakceptowali ofertę, nie mając wcześniej lokaty w
banku, ale mieli kontakt w poprzednich kampaniach: 14.23%
```

Z powyższych wynika, że ponowny kontakt z uczestnikami poprzednich kampanii jest skuteczniejszy od rekrutacji nowych klientów. Rekordowa pod tym względem jest grupa ludzi która już skorzystała z oferty banku – szansa na założenie lokaty jest średnio ponad siedmiokrotnie większa aniżeli wśród grupy bez wcześniejszej historii kontaktów z bankiem.

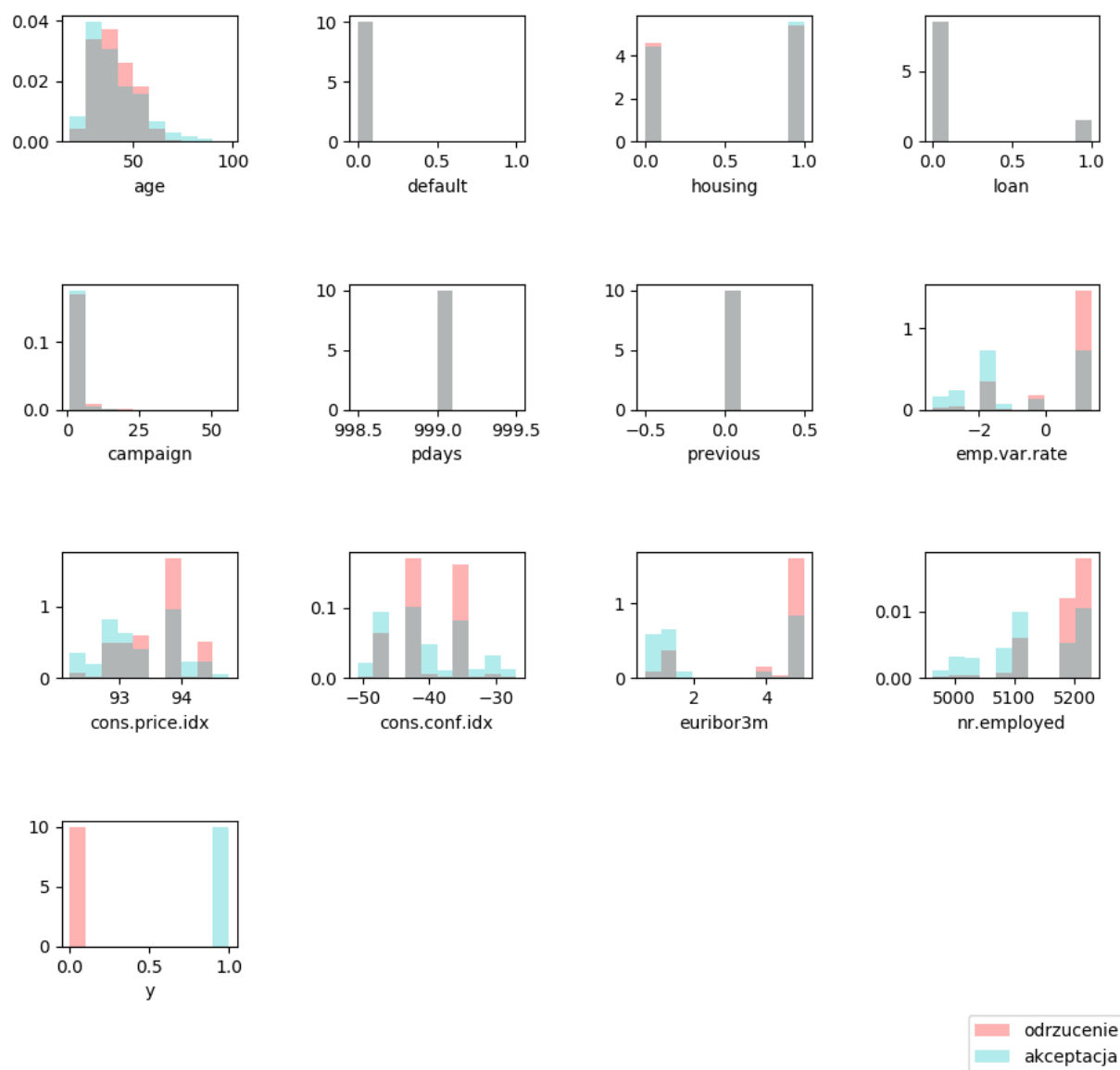
Następnie porównam liczbę telefonów wykonywanych do klientów w zależności od końcowego efektu. Dla lepszej czytelności wykresu, nie wyrysuję tzw. „outliers”



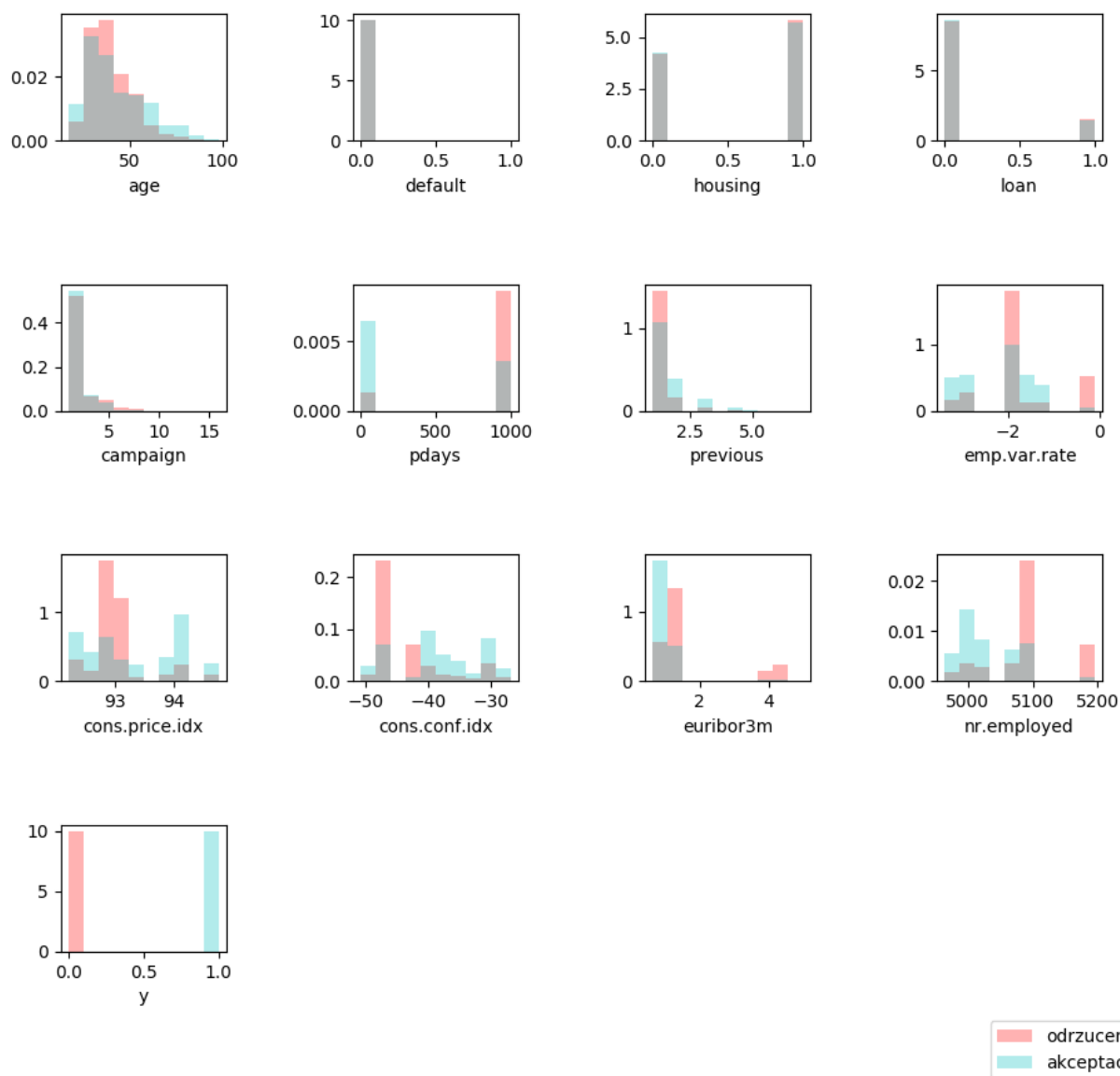
*Ilustracja 4: Liczba wykonanych telefonów w podziale na grupy*

Z powyższego wykresu widać, że najszybciej decyzję o otwarciu lokaty podejmują ludzie którzy mieli już do czynienia z bankiem. Mediana liczby telefonów dla tej grupy wynosi 1. Jednocześnie średnia liczba telefonów pozostaje stosunkowo wysoka, co wynika z istnienia kilku rekordów mocno odstających od pozostałych, przez co zaburzających wynik. Osoby dla których obecna kampania jest pierwszą w której biorą udział, wykazują znacząco szerszy rozkład. Oznacza to, że pozyskanie nowych klientów wymaga więcej pracy ze strony telemarketerów. Prawdopodobnie różnica wynika z faktu, że część informacji na temat obecnej oferty jest wspólna z poprzednimi kampaniami, więc nie ma potrzeby powtarzać szczegółowego opisu ludziom już z nimi zapoznanymi.

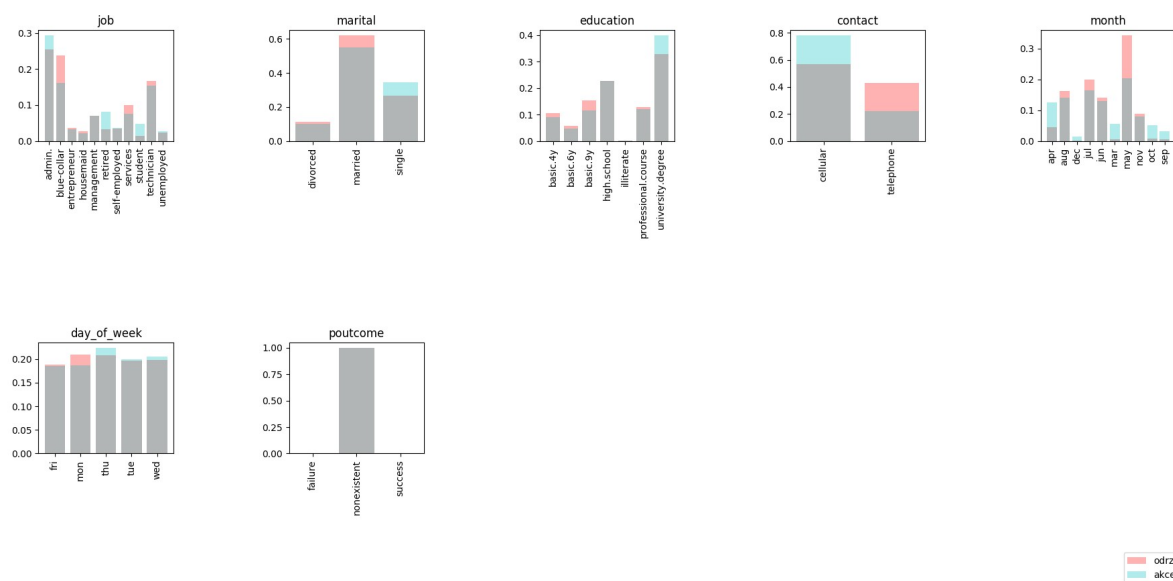
Poszukam teraz różnic pomiędzy dwoma grupami klientów – ci którzy mieli już kontakt z bankiem w poprzednich kampaniach oraz ci którzy nie mieli. Ustalę w ten sposób czy wyższa skuteczność telemarketerów wśród „starych” klientów nie wynika ze zmiany grupy wśród której jest prowadzona kampania. W tym celu ponownie wyrysuję histogramy oraz wykresy słupkowe, tym razem z dodatkowym podziałem na ludzi biorących udział w poprzednich kampaniach i nie biorących.



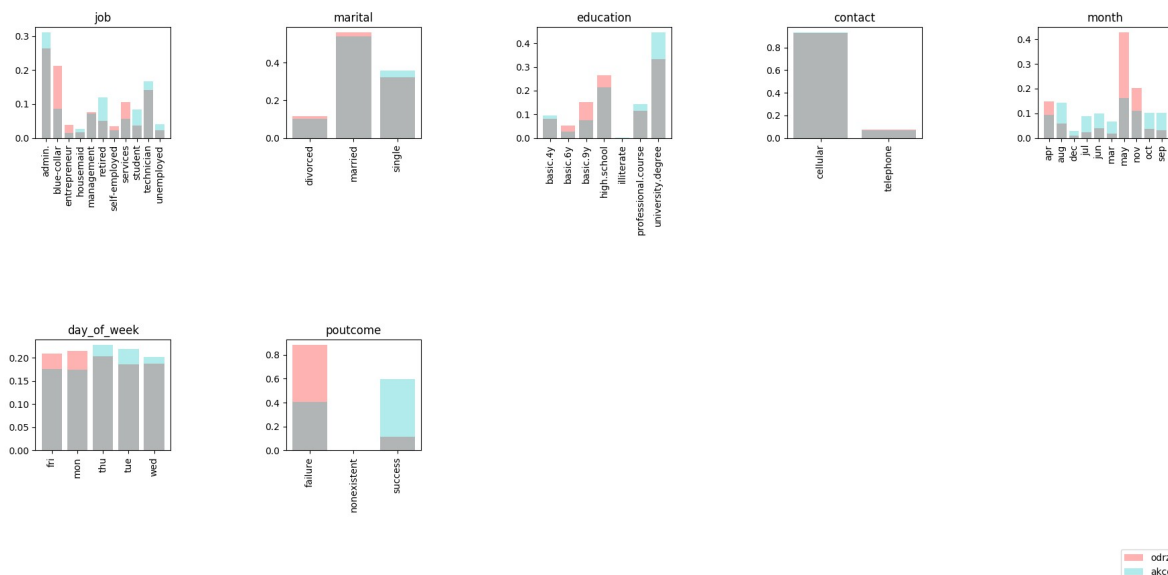
*Ilustracja 5: Histogramy dla klientów bez historii kontaktów z bankiem*



Ilustracja 6: Histogramy dla klientów z historią kontaktów z bankiem



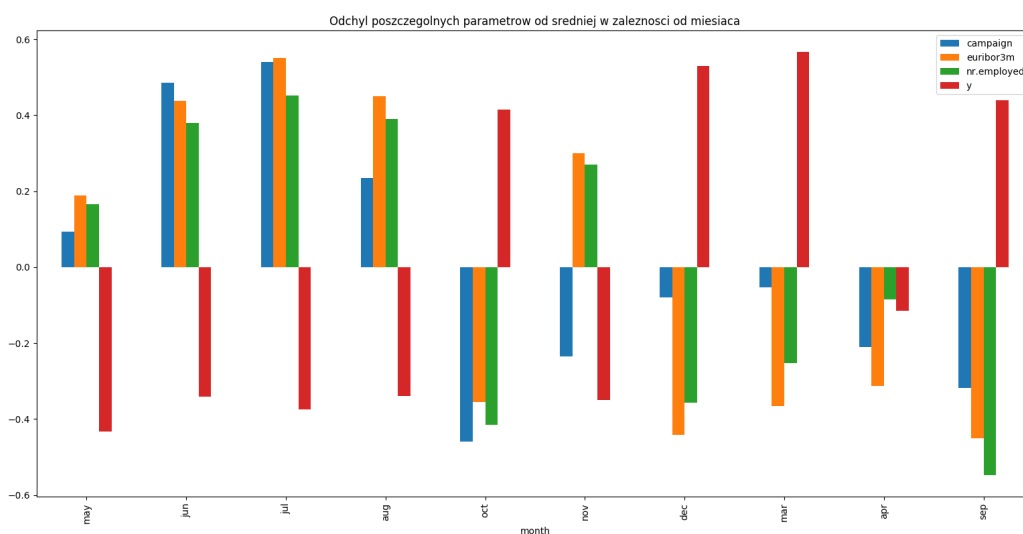
Ilustracja 7: Wykresy słupkowe dla klientów bez historii kontaktów z bankiem



Ilustracja 8: Wykresy słupkowe dla klientów z historią kontaktów z bankiem

Z powyższych wykresów nie wynika żadna istotna różnica pomiędzy charakterystykami klientów z poprzednich oraz obecnej kampanii. Zmienił się jedynie rozkład sposób w jaki dokonano kontaktu z klientem, tzn. telefon komórkowy czy stacjonarny. W obecnej kampanii przeważa ten drugi sposób, co nie odbija się pozytywnie na wynikach. Inna ciekawa rzecz która się ujawniła, to silny wpływ czynników makroekonomicznych na liczbę pozyskiwanych klientów. Aby dokładniej zbadać to zagadnienie, zilustruję zmiany wybranych wskaźników dla poszczególnych miesięcy, jako odchylenie od średniej wartości wyliczonej dla analizowanego okresu czasu.

```
grouped = df.groupby(['month'], sort=False).mean()
grouped = grouped[['campaign', 'euribor3m', 'nr.employed', 'y']]
#grouped.loc[:, ['nr.employed']] = grouped.loc[:, ['nr.employed']] / 1000
grouped = (grouped - grouped.mean()) / (grouped.max() - grouped.min())
grouped.plot.bar()
```



Ilustracja 9: Zmiany wartości wskaźników makroekonomicznych



Z powyższego wykresu widać silną zależność wysokości pomiędzy kolumną *euribor3m* – opisującą wysokość oprocentowania na rynku międzybankowym, a ilością pozyskanych lokat. Sugeruje to istotną rolę atrakcyjności oferty w procesie podejmowania decyzji o założeniu lokaty przez potencjalnych klientów. Bank nie ma wpływu na wysokość stawki euribor, ale może zintensyfikować działania marketingowe w okresie jej niskiej wartości. Poniżej liczbową reprezentacją tej zależności.

`grouped.corr()`

	campaign	euribor3m	nr.employed	y
campaign	1.000000	0.783063	0.790278	-0.604730
euribor3m	0.783063	1.000000	0.969371	-0.891991
nr.employed	0.790278	0.969371	1.000000	-0.916503
y	-0.604730	-0.891991	-0.916503	1.000000

Na koniec policzę procentowe udziały „idealnych” klientów w całości populacji

## Wnioski z analizy

Z powyższej analizy wynika, że działalność marketingowa banku jest najskuteczniejsza wśród obecnych klientów oraz ludzi mających już kontakt z telemarketerami analizowanej instytucji. Również przeciętna liczba telefonów (a więc zasoby) po której dana osoba podejmuje decyzję jest mniejsza. Skoncentrowanie się na tej grupie jest najlepszą strategią w sytuacji gdy bank potrzebuje zwiększyć liczebność swojego portfela klientów z lokatami.

- Procent klientów którzy zaakceptowali ofertę, mając do czynienia z bankiem poprzednio: 26.65%
- Procent klientów którzy zaakceptowali ofertę, nie mając do czynienia z bankiem poprzednio: 8.83%
- Procent klientów którzy zaakceptowali ofertę, mając już wcześniej lokatę w banku: 65.11%
- Procent klientów którzy zaakceptowali ofertę, nie mając wcześniej lokaty w banku, ale mieli kontakt w poprzednich kampaniach: 14.23%

Z kolei chcąc zdobyć nowych klientów, bank powinien skupić się na ludziach w wieku poniżej 30 lat lub powyżej 60, o statusie zawodowych studenta, emeryta lub pracownika administracyjnego. Preferowane powinno być wykształcenie wyższe. Są to cechy mające pozytywny wpływ na końcowy efekt, czyli założenie lokaty. Również posiadający pozytywny wpływ, ale o mniejszym znaczeniu cechy to stan cywilny – preferowane osoby żyjące samotnie oraz dzień tygodnia – preferowany kontakt w drugiej połowie, tj. środa, czwartek i piątek. Liczbowe przedstawienie powyższych danych:

- Udział ludzi spełniających idealne warunki w całkowitej liczbie lokat to 9.22%, przy całkowitym udziale w populacji 1.04%.
- Skuteczność rekrutacji wśród grupy spełniającej idealne warunki wyniosła 24.58%, przy ogólnej skuteczności 11.27%
- 7.29% ludzi posiadających przynajmniej jedną cechę ze zbioru idealnych, posiada również pozostałe cechy

Powyższe dane zostały wyliczone na podstawie poniższego kodu

```
idealni_poz = sum( ( (df.age<=30) | (df.age>=60)) &
                  ( (df.job=='student') | (df.job=='retired') |
(df.job=='admin.')) &
                  ( df.education == 'university.degree' ) &
                  ( df.y==1 ) )

idealni_wszystkie = sum( ( (df.age<=30) | (df.age>=60)) &
                        ( (df.job=='student') | (df.job=='retired') |
(df.job=='admin.')) &
                        ( df.education == 'university.degree' ) )

idealni_jedna_cecha = sum( ( (df.age<=30) | (df.age>=60)) |
                           ( (df.job=='student') | (df.job=='retired') |
(df.job=='admin.')) |
                           ( df.education == 'university.degree' ) )

print('''Udział ludzi spełniających idealne warunki w całkowitej liczbie lokat
to: {0:.2f}%, przy całkowitym udziale w populacji: {1:.2f}%. Skuteczność
rekrutacji wśród grupy wyniosła {2:.2f}%, przy ogólnej skuteczności
{3:.2f}%'''.
      format( idealni_poz/sum(df.y==1)*100, idealni_poz/len(df)*100,
              idealni_poz/idealni_wszystkie*100, sum(wyb_poz)/len(wyb_poz)*100 ) )

#w jakiej mierze są to ci sami ludzie (spełniający wszystkie wymagania)

print('''{0:.2f}% ludzi posiadających przynajmniej jedną cechę ze zbioru
idealnych, posiada również pozostałe cechy'''.format(
      idealni_wszystkie / idealni_jedna_cecha *100))
```

## Model

W pierwszym kroku zamienię wszystkie dane sprowadzę do reprezentacji liczbowej. W przypadku miesięcy, edukacji oraz dni tygodnia, możliwe jest wyróżnienie kolejności – odpowiednie wartości liczbowe tę kolejność zachowają. Dla kolumny *pdays* dokonam wcześniejszego podziału na kategorie – zakresy, a dopiero następnie sprowadzę do reprezentacji liczbowej. W pozostałych przypadkach powstaną nowe kolumny z wartościami 1 lub 0 odpowiadającymi istnieniu danej cechy.

```
df.month = df.month.map({'mar':3,'apr':4,'may':5,'jun':6,'jul':7,'aug':8,
                        'sep':9,'oct':10, 'nov':11, 'dec':12})

df.education = df.education.map({'illiterate':0,'basic.4y':1,'basic.6y':2,
                                'basic.9y':3, 'professional.course':4, 'high.school':5, 'university.degree':6})

df.day_of_week = df.day_of_week.map({'mon':1,'tue':2,'wed':3,'thu':4,'fri':5})

df.pdays=pd.cut(df.pdays,[0,5,10,15,20,25,999],[5,10,15,20,25,999]).cat.codes

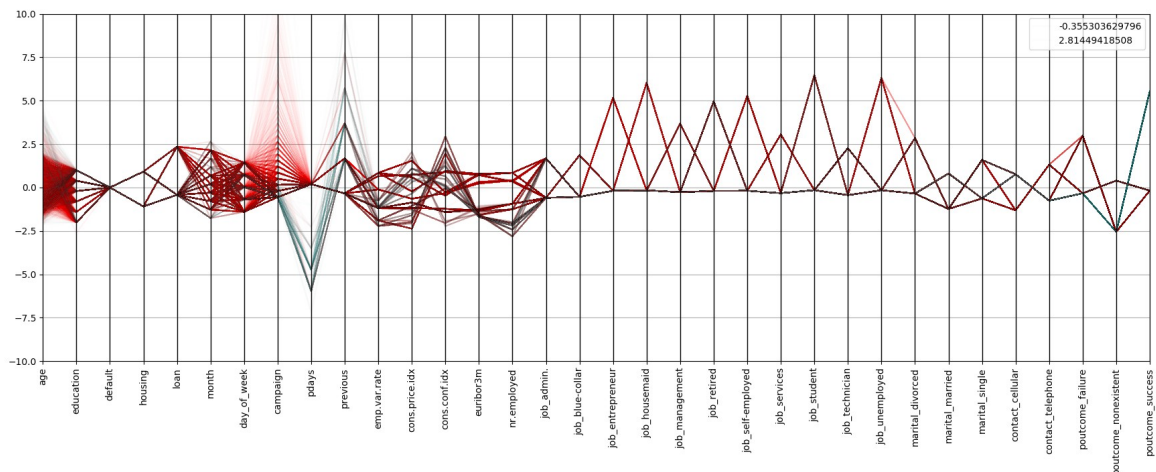
df = pd.get_dummies(df,columns=['job','marital','contact','poutcome'])
```

Przed przystąpieniem do kolejnego kroku, skontroluję jeszcze ogólny rozkład zmiennych przy użyciu wykresu ze współzrędnymi równoległymi. Dla celów wizualizacji, dane zostaną przeskalowane, a wykres powstanie na co trzeciej próbce.

```
plt.figure('wspolrzedne rownolegle')

pd.plotting.parallel_coordinates(pd.DataFrame(scale(df[:,3])),columns=df.columns,
                                , 'y', alpha=0.01, color=['r','c'])

plt.xticks(rotation='vertical')
plt.ylim((-10, 10))
plt.tight_layout()
```



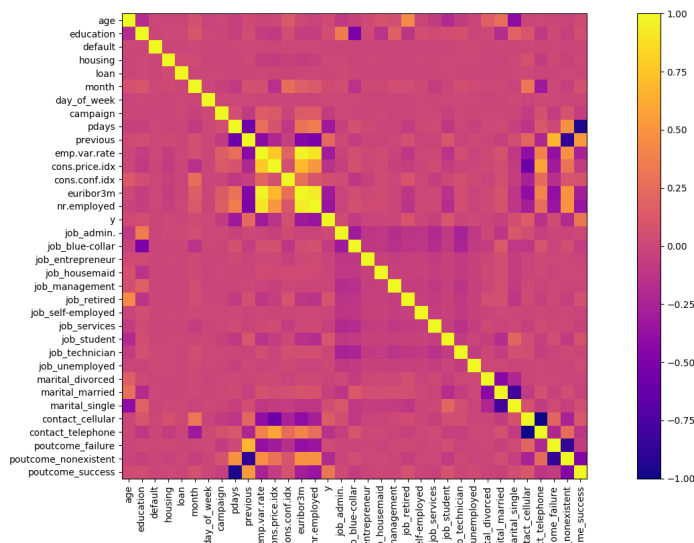
*Ilustracja 10: Współrzędne równoległe*

Na podstawie powyższego można dojść do wniosku że cecha *default* nie daje żadnego wkładu do rezultatów. Jednocześnie widać tendencje opisane w poprzedniej części pracy.

Kolejnym krokiem będzie eliminacja zmiennych które są ze sobą mocno skorelowane lub nie wnoszą do ostatecznego wyniku. Przydatne może się okazać wyznaczenie macierzy korelacji.

```
plt.figure('macierz korelacji')
plt.imshow(df.corr(), cmap=plt.cm.plasma, interpolation='nearest')
plt.colorbar()

tick_marks = [i for i in range(len(df.columns))]
plt.xticks(tick_marks, df.columns, rotation='vertical')
plt.yticks(tick_marks, df.columns)
```



Ilustracja 11: Ilustracja korelacji pomiędzy zmiennymi

Na podstawie dotychczasowych analiz zdecydowałem się pozbyć zmiennych: *default*, *campaign*, *loan*, *housing*, *nr.employed* oraz *emp.var.rate*

```
target = df.y
zbedne_cechy = ['y', 'default', 'campaign', 'loan', 'housing', 'nr.employed',
               'emp.var.rate']
df.drop(zbedne_cechy, axis=1, inplace=True)
```

Mając przygotowane dane, mogę przejść do ostatniej fazy, tj. trenowanie oraz wybór modelu klasyfikacyjnego. Jako finalny, wybiorę jeden spośród czterech: SVC, regresja logistyczna, lasy losowe lub drzewo decyzyjne. 20% danych przeznaczę na próbkę testową. Jako ostatni krok przed trenowaniem modeli, wyskaluję dane, tak aby miały średnią 0 oraz wariancję 1. Do wyznaczenia optymalnych hiperparametrów posłużę się funkcją GridSearchCV, która wybierze je na podstawie najlepszego wyniku na zbiorze walidacyjnym. Jako metrykę oceny jakości modelu wybrałem *F\_score*, który jest odpowiedni w przypadku danych o mocno nierównej liczebności przypadków pozytywnych oraz negatywnych.

```

modele = dict()

modele['SVC'] = [SVC(),{'C':[0.1,0.3,1,], 'kernel':['rbf']} ]

modele['LogisticRegression'] = [LogisticRegression(), {'C':[0.1,0.3,1,3,10],
'solver':['saga'], 'max_iter':[2000]}]

modele['RandomForestClassifier'] = [RandomForestClassifier(),{'n_estimators':
[10,20], 'criterion':['gini','entropy'], 'max_depth':[3,7,10],
'min_samples_split':[7,20], 'min_samples_leaf':[3,10]}]

modele['DecisionTreeClassifier'] = [DecisionTreeClassifier(), {'criterion':
['gini','entropy'], 'max_depth':[3,7,10], 'min_samples_split': 7,20],
'min_samples_leaf':[3,10]}]

X_train, X_test, y_train, y_test = train_test_split(df,target, test_size=0.2)

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

wyniki=dict()
best_model = 0
best_score = 0

for model in modele:
    print(model)

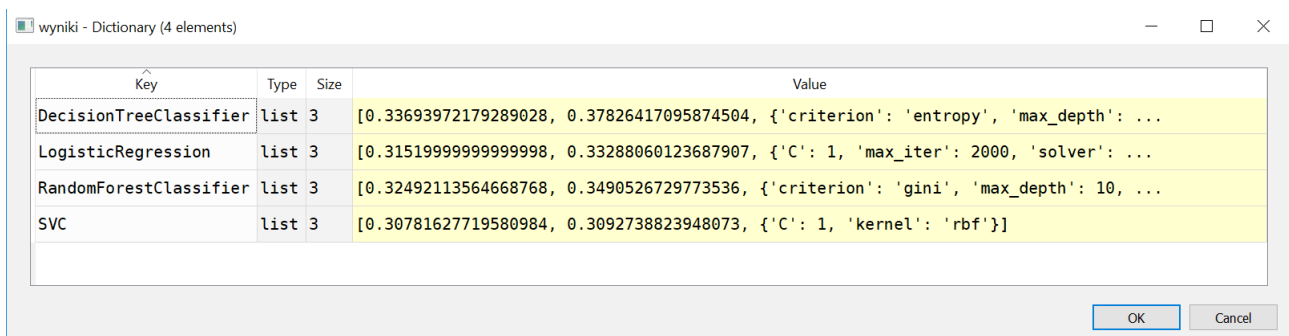
    classifier = GridSearchCV(modele[model][0], modele[model][1], scoring='f1',
                             verbose=True)
    classifier.fit(X_train, y_train)

    wyniki[model] = [classifier.score(X_test,y_test),classifier.best_score_,
                    classifier.best_params_]

    if wyniki[model][0] > best_score:
        best_score = wyniki[model][0]
        best_model = model

```

Najlepszy wynik na zbiorze testowym (pierwsza kolumna) uzyskał model drzewa decyzyjnego i wyniósł on 0.34 co nie jest wysokim rezultatem.



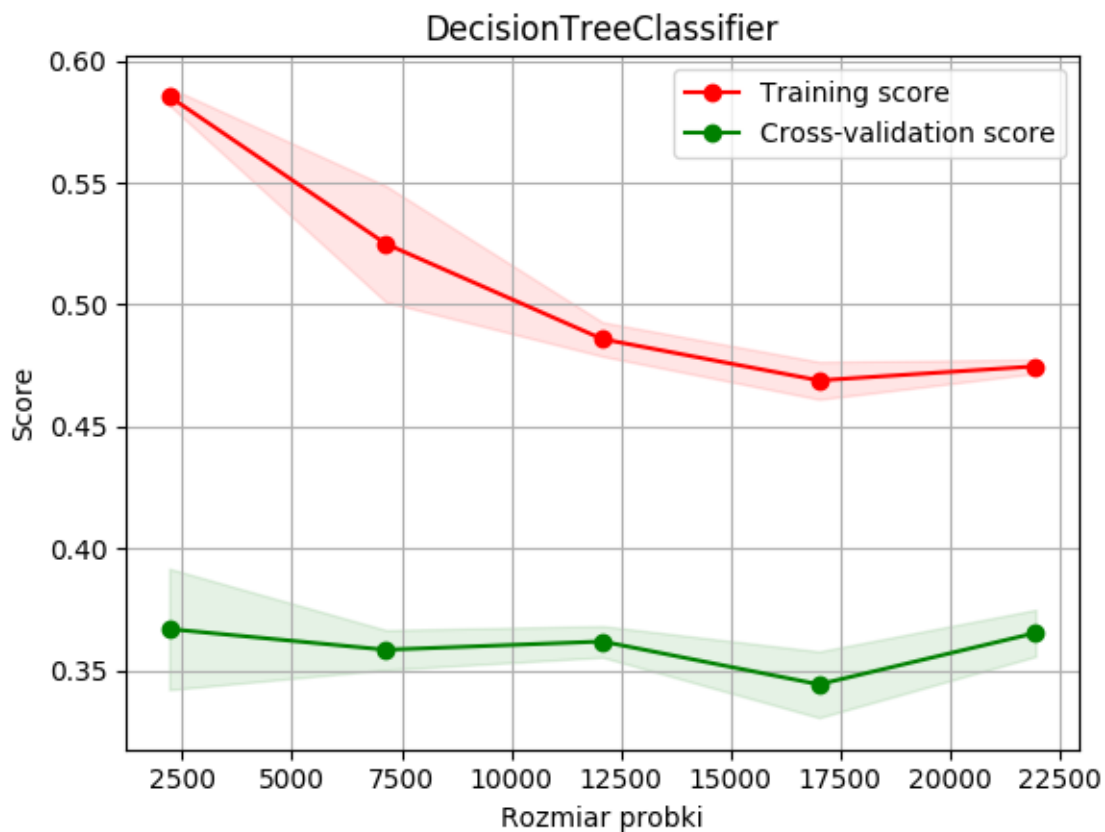
Key	Type	Size	Value
DecisionTreeClassifier	list	3	[0.33693972179289028, 0.37826417095874504, {'criterion': 'entropy', 'max_depth': ...
LogisticRegression	list	3	[0.31519999999999998, 0.33288060123687907, {'C': 1, 'max_iter': 2000, 'solver': ...
RandomForestClassifier	list	3	[0.32492113564668768, 0.3490526729773536, {'criterion': 'gini', 'max_depth': 10, ...
SVC	list	3	[0.30781627719580984, 0.3092738823948073, {'C': 1, 'kernel': 'rbf'}]

*Ilustracja 12: Wyniki poszczególnych modeli. Pierwsza kolumna reprezentuje rezultat dla zbioru testowego*

Drzewo decyzyjne zwraca również informację na temat istotności poszczególnych cech. W tym przypadku najważniejsze 3 to:

1. euribor3m z wartością 0.638
2. poutcome\_success z wartością 0.091
3. age z wartością 0.077

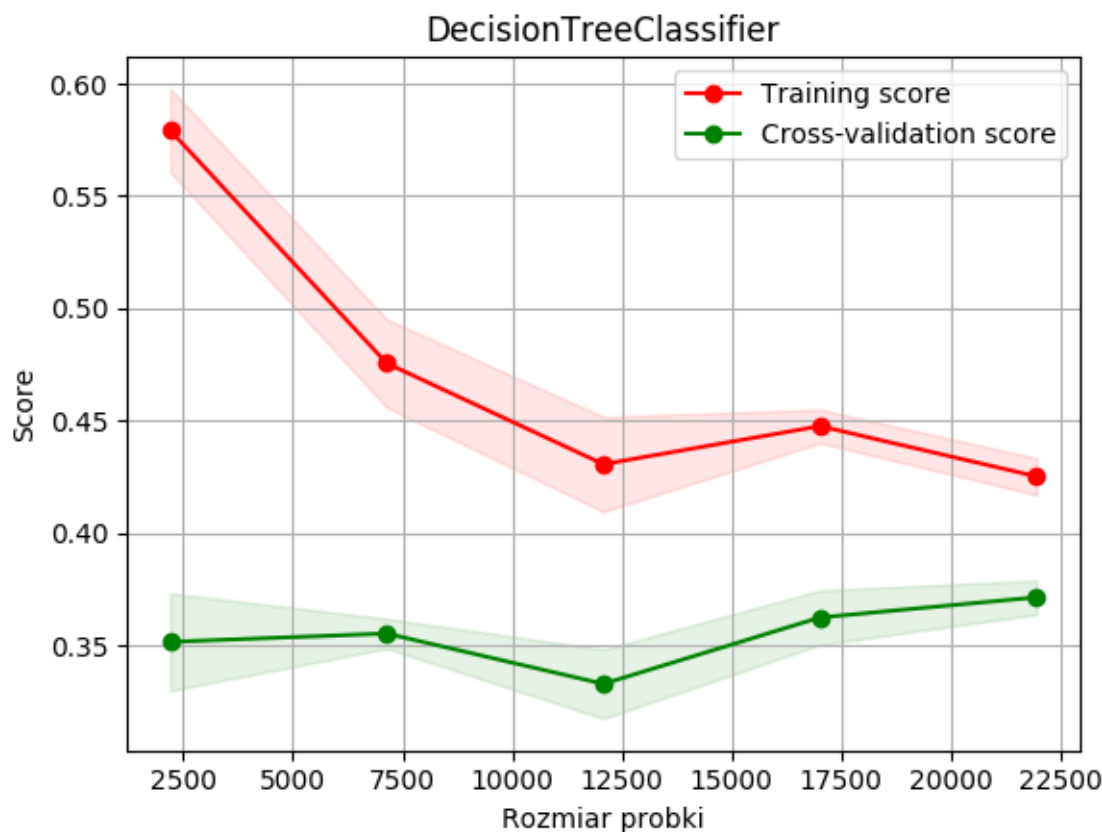
Aby zdiagnozować problem niskiej skuteczności algorytmu, posłużę się tzw. learning curves, czyli wykresem wyników modelu na zbiorze treningowym oraz walidacyjnym, w zależności od rozmiaru zbioru treningowego. Efekty poniżej.



Ilustracja 13: Krzywe uczenia

Kształt krzywych na wykresie może sugerować przeuczenie modelu. Spróbuj rozwiązać ten problem powtarzając proces trenowania modelu, lecz tym razem odrzucając więcej cech.

Niestety eliminacja kolejnych cech oraz próby prostszego przedstawienia danych (przykładowo bez użycia funkcji `get_dummies`) nie przyniosły oczekiwanych rezultatów.



*Ilustracja 14: Krzywa uczenia dla cech zakodowanych bez użycia `get_dummies()`*

## Wnioski

Na podstawie analizowanych danych udało się zbudować model predykcyjny, aczkolwiek jego skuteczność na poziomie 0.34 używając metryki F score, nie należy do najwyższych. Z analizy krzywych uczenia, można wywnioskować przeuczenie modelu. Jednym z możliwych rozwiązań jest zebranie większej ilości danych. Jednocześnie, cechy wybrane jako najistotniejsze przez model drzewa decyzyjnego, pokryły się z cechami zasugerowanymi w pierwszej części pracy.