Raport Końcowy
Bank Marketing Classification Task

Bartosz Jezierski i Michał Iwaniuk

21 kwietnia 2024

1 Wstęp

Celem tego projektu było wprowadzenie nas w świat ML i procesu tworzenia modeli maszynowych. Został on zrealizowany jako część kursu "Wstęp do uczenia maszynowego" na kierunku Inżynieria i Analiza Danych na wydziale MINI na Politechnice Warszawskiej. Całość została wykonana w głównej mierze za pomocą pakietu sklearn.
Link do repozytorium projektu - here

1.1 Zadania

Nasze zadanie zostało podzielone na kolejne podzadania:
- EDA (Exploatory Data Analysis) - wejść w nasz dataset, zrozumieć jego cechy i znaleźć relacje je łączące za pomocą wizualizacji,
- Feature Engineering - przetworzyć oraz ewentualnie wyczyścić i wyselekcjonować cechy,
- Stworzenie modelu - przejście od prostych modeli do tych bardziej zaawansowanych oraz dobranie hiperparametrów w celu maksymalizacji metryk oceny modelu.

1.2 Walidacja

Przez cały okres tworzenia projektu walidowaliśmy projekt naszych kolegów oraz byliśmy przez kolejnych walidowani. Zadaniem walidatorów było ciągłe sprawdzanie czy np. nie został popełniony gdzieś błąd logiczny w trakcie procesu oraz ewentualna pomoc w całym zagadnieniu. Ponadto otrzymali oni od nas 30% wszystkich danych, do których później nie mogliśmy zaglądać. Dane te miały posłużyć do ostatecznego testu naszego modelu.

1.3 Dane i cel

Naszą bazą były dane pochodzące z kampanii bezpośredniego marketingu prowadzenego przez portugalski bank. Oparto je o rozmowy telefoniczne prowadzone z potenc-

jalnymi przyszłymi klientami.
Celem zadania była estymacja czy dany klient będzie chętny złożyć lokatę w banku czy nie (problem klasyfikacji binarnej).
Link do danych - here.

2 EDA

Nasza część datasetu składa się z 3164 obserwacji. Oto pierwszych pięć:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	37	entrepreneur	single	tertiary	no	1467	yes	yes	cellular	17	nov	349	1	-1	0	unknown	no
1	25	unemployed	single	secondary	no	34	no	yes	unknown	4	jul	316	1	-1	0	unknown	no
2	53	retired	married	secondary	no	1278	yes	no	telephone	15	jul	174	3	-1	0	unknown	no
3	31	housemaid	married	unknown	yes	-6	no	yes	telephone	7	jul	94	2	-1	0	unknown	no
4	43	unemployed	married	secondary	no	3529	no	no	cellular	5	feb	169	2	-1	0	unknown	no

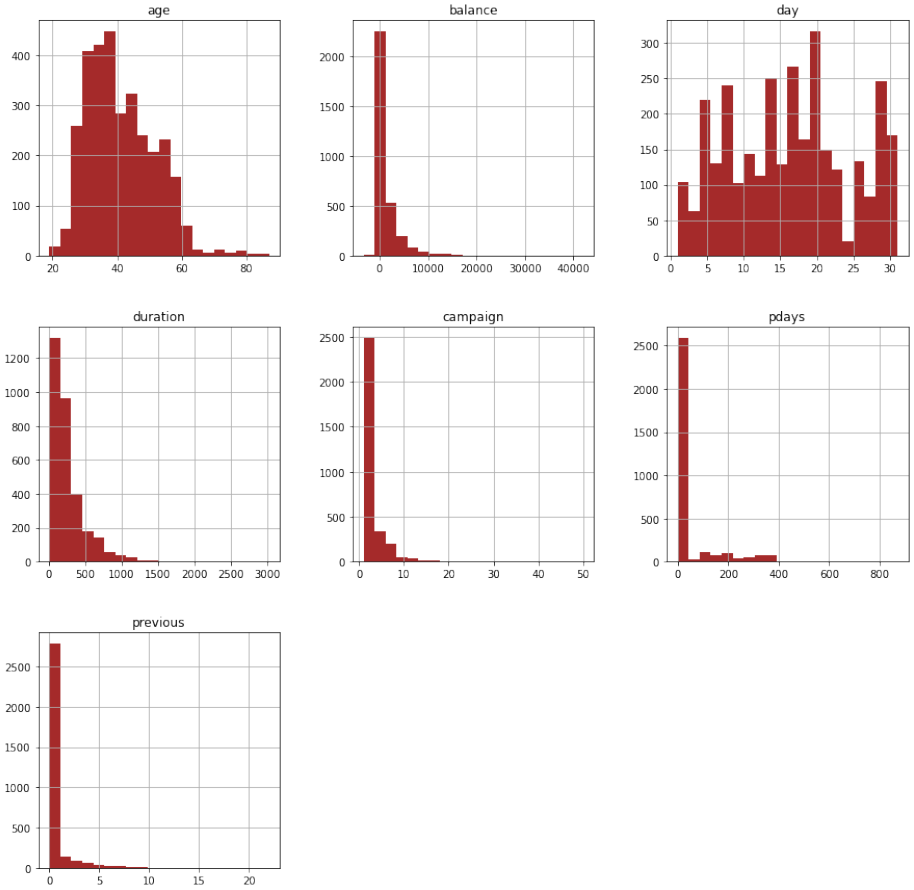
2.1 Znaczenia cech

Łącznie jest 16 cech:
- **age** - wiek klienta
- **job** - praca lub kategoria pracy jaką zajmuje się klient. Możliwości:
- **maritial** - stan cywilny. Możliwości:
- **education** - poziom edukacji
- **default** - czy klient ma niespłacony kredyt
- **balance** - średni bilans roczny
- **housing** - czy klient ma kredyt hipoteczny
- **loan** - czy klient ma prywatną pożyczkę
- **contact** - w jaki sposób skontatkowano się z klientem
- **day** - jakiego dnia miesiąca odbył się ostatni kontakt
- **month** - jakiego miesiąca odbył się ostatni kontakt
- **duration** - ile sekund trwała rozmowa
- **campaign** - ile łącznie razy kontaktowano się z klientem w czasie tej kampanii
- **pdays** - liczba dni, które minęły od kontaktu podczas poprzedniej kampanii
- **previous** - ile łacznie razy kontaktowano się z klientem w czasie poprzednich kampanii
- **poutcome** - jak zakończyła się ostatnia kampania

Oraz nasz target:
- **y** - czy klient skusił się na ofertę

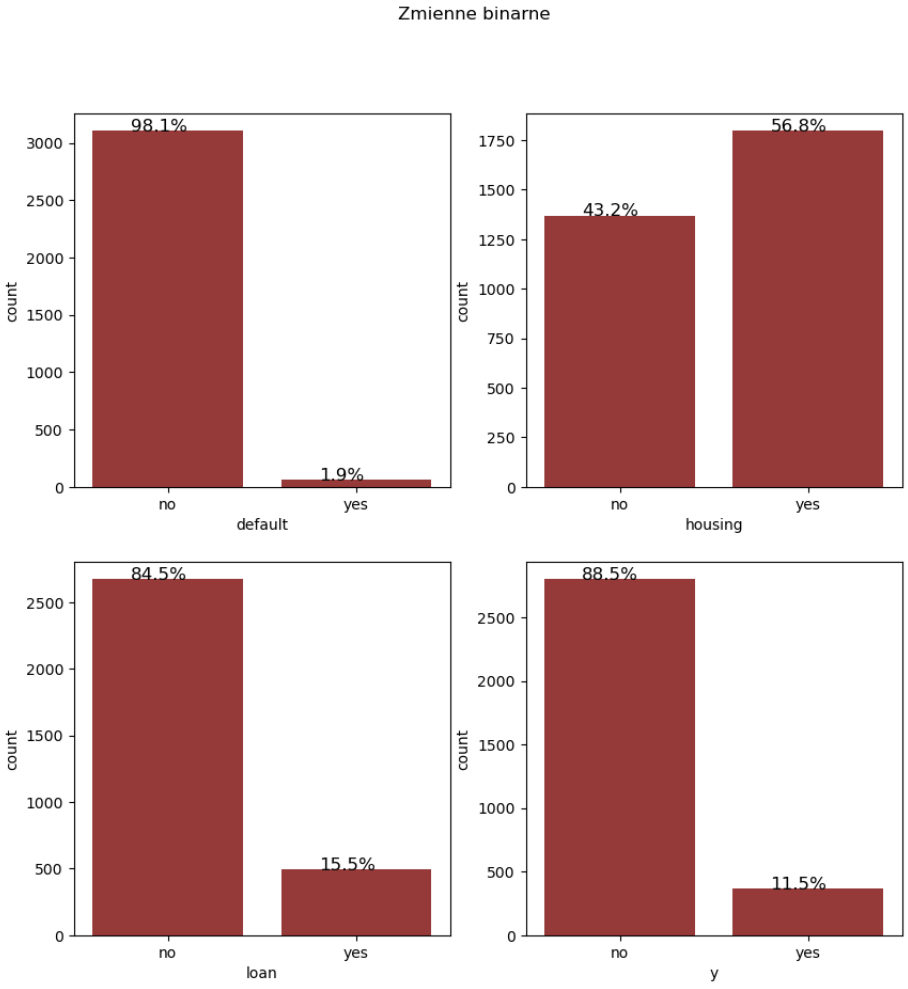
2.2 Analiza jednej zmiennej

2.2.1 Numerycznej

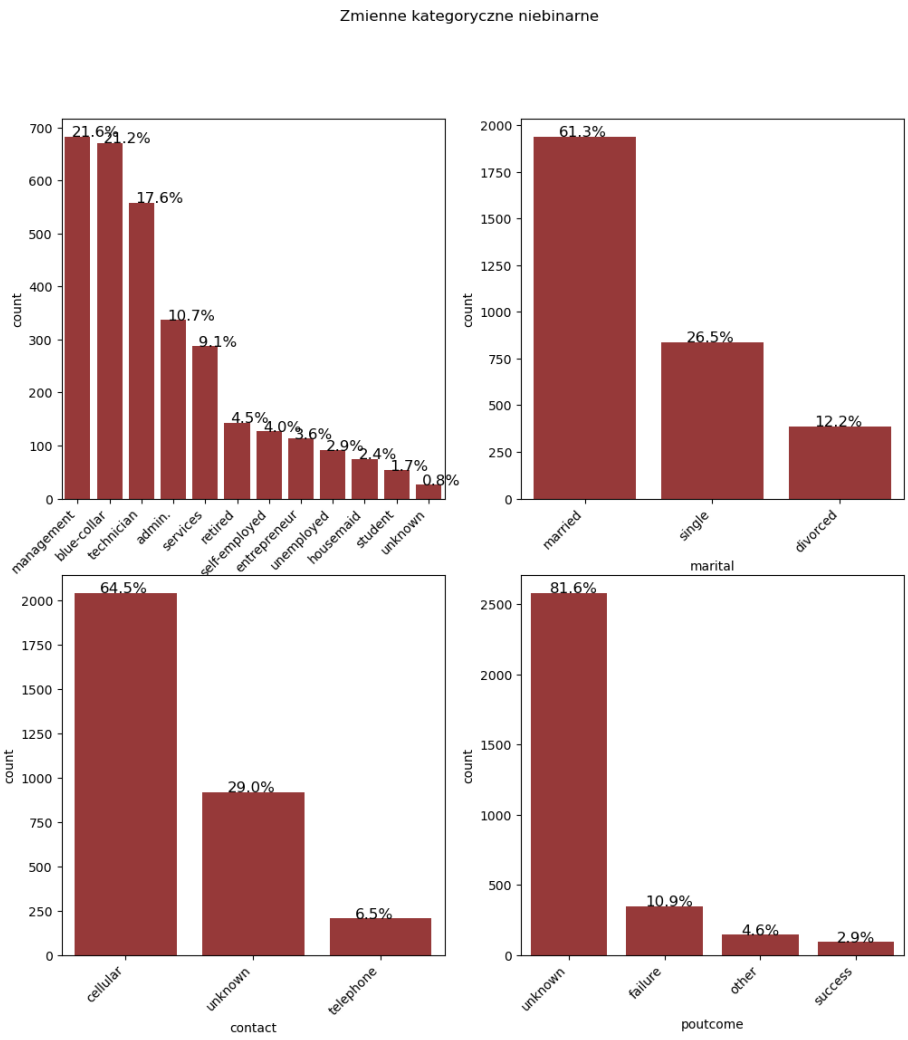


Obserwacje:
- Ludzie w wieku średnim głównie
- Większość ludzi jest na niewielkim plusie r/r (balance) - jeżeli na minusie to niewielkim
- Rozmowy trwają zwykle krótko (duration), wygląda na rozklad wykladniczy
- Większość ludzi ma pojedynczą styczność z kampanią (previous/pdays)

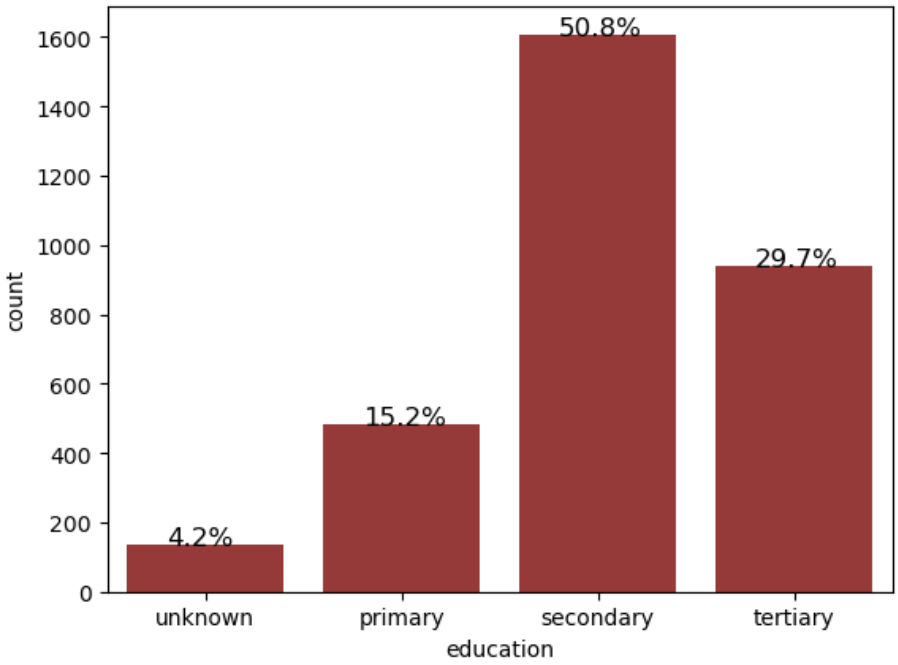
2.2.2 Kategorycznej

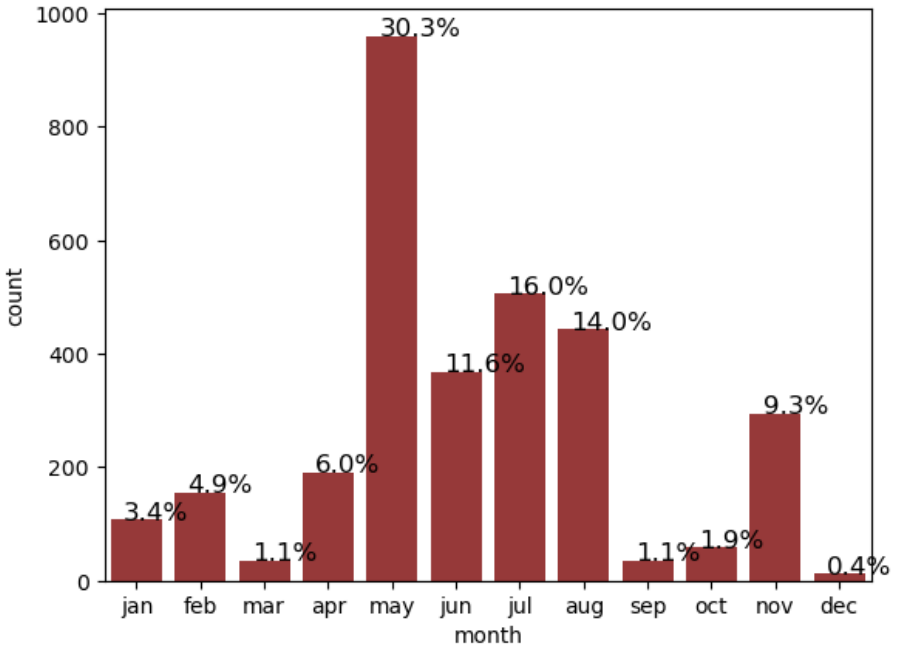


Obserwacje:
- Prawie nikt nie ma zaległych płatności kredytowych (credit)
- Mniej więcej tyle samo ma i nie ma kredytu mieszkaniowego (housing)
- Większość nie ma pożyczek indywidualnych (loan)
- Jedna na dziewięć z osób zgadza się na złożenie depozytu (y)



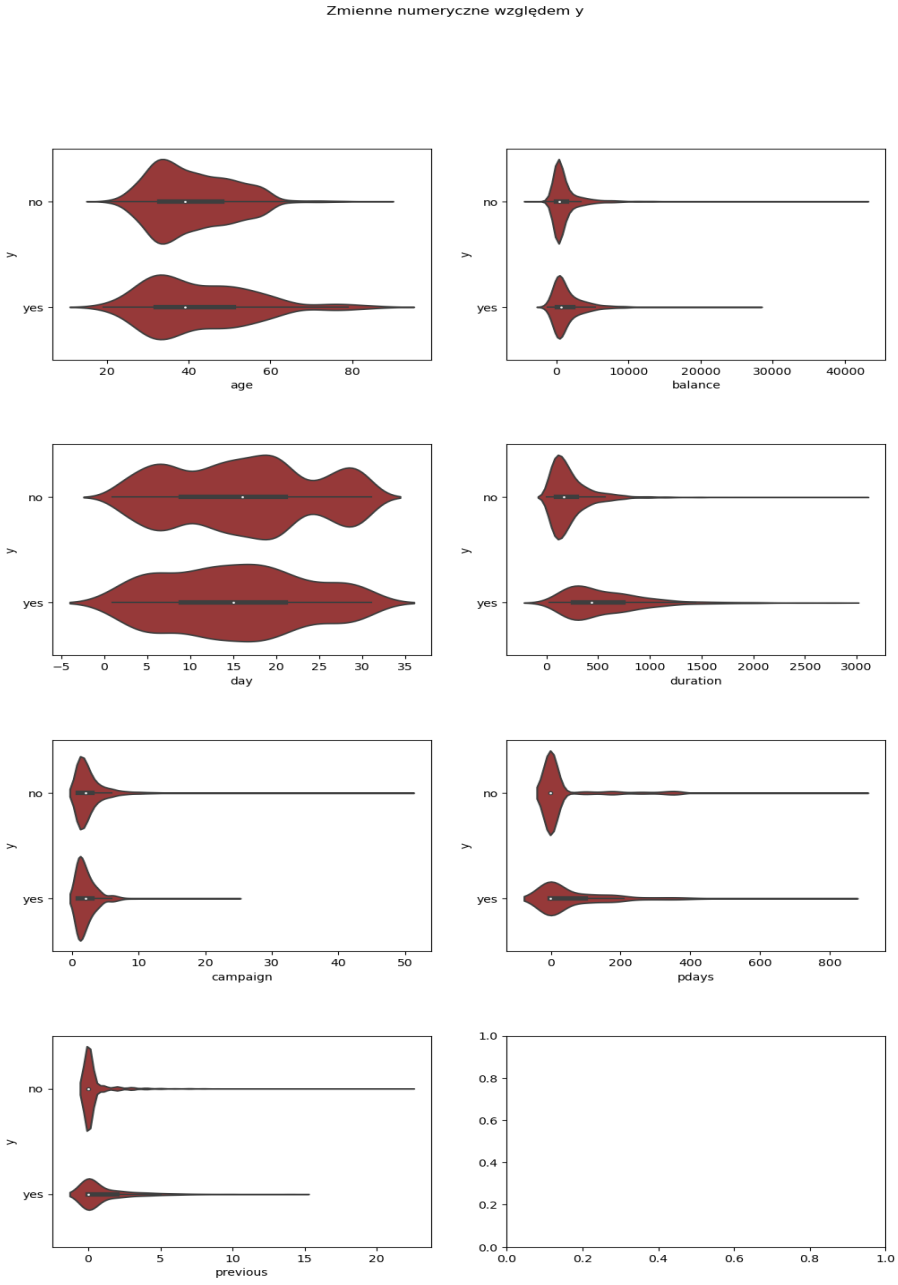
Obserwacje:
- typowe zawody klasy średniej
- większość zamężna
- kontakt przez rozmowy telefoniczne/zdalne
- dla większości osób jest to pierwsza kampania przeprowadzona przez ten bank


1600
1400
1200
1000
800
600
400
200
0
count
50.8%
29.7%
15.2%
4.2%
unknown
primary
secondary
tertiary
education


1000
800
600
400
200
0
count
30.3%
16.0%
14.0%
11.6%
9.3%
6.0%
4.9%
3.4%
1.1%
1.1%
1.9%
0.4%
jan
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
month

2.3 Analiza ze względu na y

2.3.1 Numeryczne



2.3.2 Kategoryczne

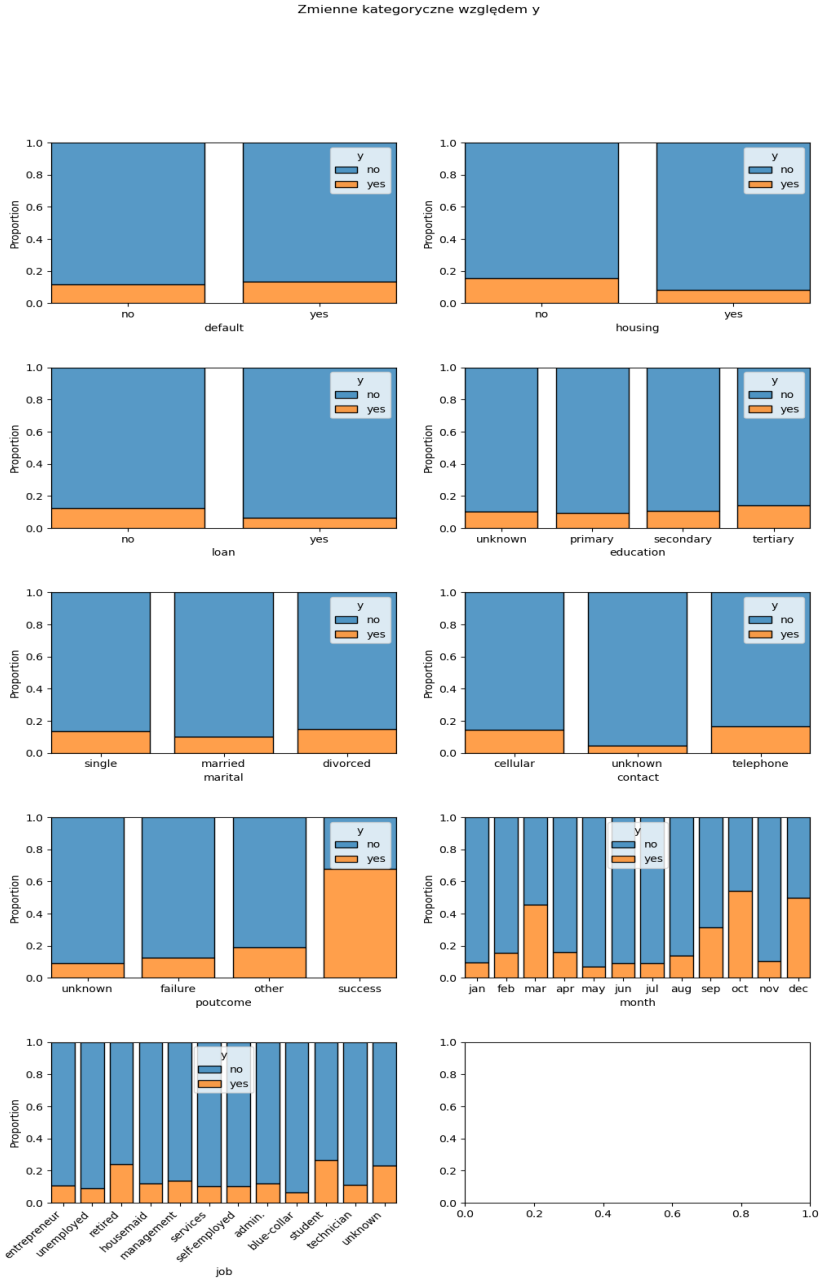
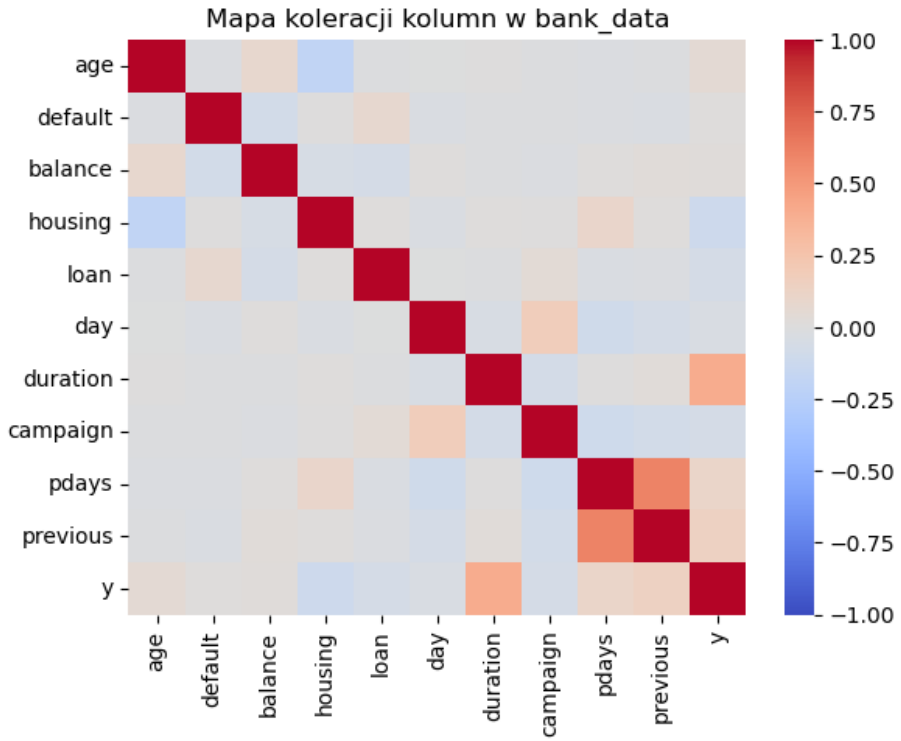


Figure 1: Słupki ukazują proporcję cechy y pod warunkiem wartości cechy napisanej pod słupkiem

Wydaje się, że:
- Ludzie starsi częściej (proporcjonalnie) mogą zostać namówieni
- Długość trwania rozmowy ma ogromny wpływ na to czy dany człowiek zdecyduje się na założenie lokaty czy nie
- Ludzie są chętniejsi do założenia lokaty, jeśli wcześniej mieli już styczność z jakąś kampanią prowadzoną przez ten bank(previous/pdays)
- Jeśli jesteśmy bez kredytu to jesteśmy chętniejsi powierzyć bankowi pieniądze
- Wydaje się, że im mamy wyższe wykształcenie tym chętniej założymy lokatę
- Ewidentnie jeśli w poprzedniej kampanii się udało klienta namówić to tym razem dużo łatwiej nam zaufa (ale jest to 3% obserwacji jedynie)
- Wydaje się cześciej udawać namówić klienta w marcu, wrześniu, październiku i grudniu, jednak obserwacje pochodzące z tych miesięcy stanowią ledwie 4.5% wszystkich obserwacji, tak naprawdę im częściej w danym miesiącu bombarduje się klientów kampanią tym rzadziej się udaje

2.3.3 Korelacje zmiennych



- oczywista koleracja między previous/pdays
- wcześniej zauważony wpływ długości rozmowy na target

2.4 Analiza ze względu na dwie zmienne

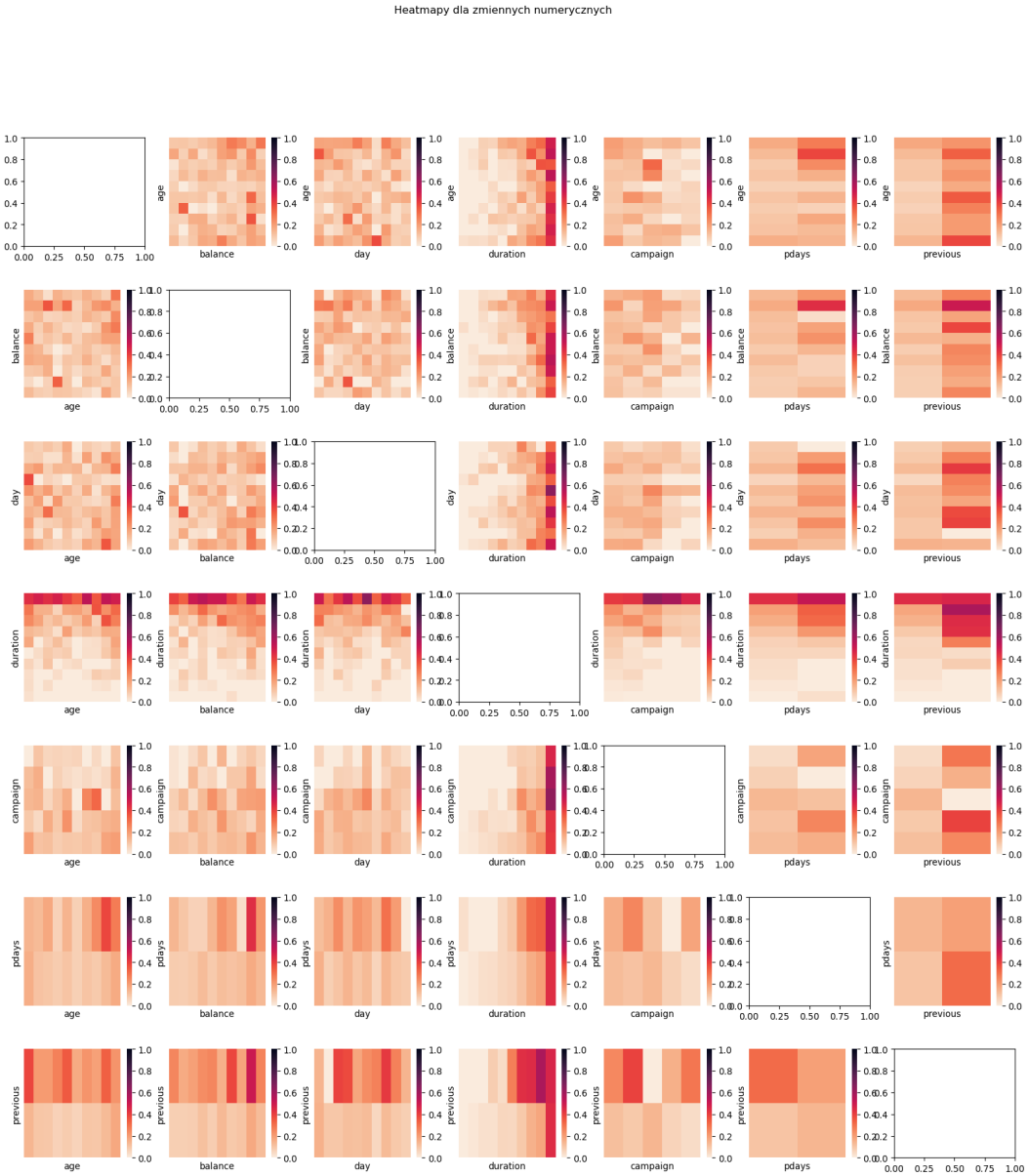


Figure 2: Wykresy przedstawiają proporcje y pod warunkiem przebywania w danym przedziale wartości. Każdy kwadrat/prostokąt zawiera tyle samo obserwacji. Im ciemniejszy obszar tym więcej jest obserwacji, gdzie y=1.

- Widać ewidentną zależność dla cechy duration, że ma ona wysoki wpływ prawdopodobieństwo założenia depozytu nie zależnie od drugiej cechy
- W takiej reprezentacji można zauważyć też, że previous mocno wpływa na prawdopodobieństwo przy uwzględnieniu duration. Oczywiście można to wytłumaczyć tym, że człowiek, który miał już styczność poprzednio z tym bankiem chętniej dowie się więcej o ofercie i końcowo zdecyduje się na proponowany produkt/założenie depozytu.
- Oprócz tego okazuje się, że zmienna balance ma wpływ tzn, że ludzie, którzy wychodzą na dużym plusie r/r chętniej założą konto jeśli wcześniej mieli kontakt z danym bankiem (plot previous/balance)

3 Feature Engineering

W tej części zaczynamy stosować proste modele i sprawdzamy jak różne transformacje zmiennych na nie wpływają.

3.1 Wybór metryki

Wcielając się w rolę pracodawcy, który zleca stworzenie modelu przewidującego czy klient złoży depozyt czy nie, uznaliśmy, że:

- Pracodawca (bank) oczekuje, że model będzie bardzo rzadko się mylił co do ocenienia, że klient zdecyduje się na lokatę (czyli jeśli już model przewiduje, że target=1 to rzeczywiście prawie na pewno tak jest). Oznacza to, że podczas oceny modeli jedną z metryk będzie **precision** = TP/(TP+FP), bo chcemy jak najbardziej uniknąć FalsePositives. Dlaczego? Jeśli bank potrzebuje takiego modelu, to bardzo możliwe, że po to, aby móc spróbować wcześniej podjąć jakieś ryzyko z zainwestowaniem środków, zanim te środki otrzyma. Źle by się stało, gdyby bank użył pieniędzy, których końcowo nie dostał. Naraziłby się wtedy na ryzyko płynności jego środków.

- Jednak pojawił się tu problem - już proste modele potrafiły uzyskać dosyć wysoki precision_score. Stało się tak dlatego, że mamy tu do czynienia z niezbalansowanym zbiorem danych, gdzie tylko w jednym na dziewięć przypadków nasz target wynosi 1 (czyli prawie 90% klientów nie złożyło końcowo lokaty). Stąd modelom "opłacało się" zakładać, że klient nie założy lokaty i ewentualnie przy bardzo pewnych przypadkach próbować przyjmować inne zdanie.

- Stąd w prostych modelach można zobaczyć, że o ile precision_score i **balanced_accuracy_score** jest na relatywnie wysokich poziomach o tyle **recall_score** jest bardzo niski. Dlatego głównym kryterium oceny stają się dwie metryki:
- **gmean** - dla niezbalansowanych danych
- **fbeta_score** - kompromis między recall i precision.

- W całym procesie, mimo wszystko będziemy moniotorować wszystkie pogrubione wyżej metryki, kładąc główny nacisk na dwie ostatnie.

$$gmean = \sqrt{sensitivity * specificity}$$

$$fbeta\_score = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + FP + \beta^2 FN}$$

, gdzie TP - true positives, FP - false positives, FN - false negatives.
Za parametr β przyjęliśmy wartość $\beta = \mathbf{0.5}$, która według dokumentacji sklearn ma kłaść większy nacisk na metrykę precision.

3.2 Transformacje zmiennych

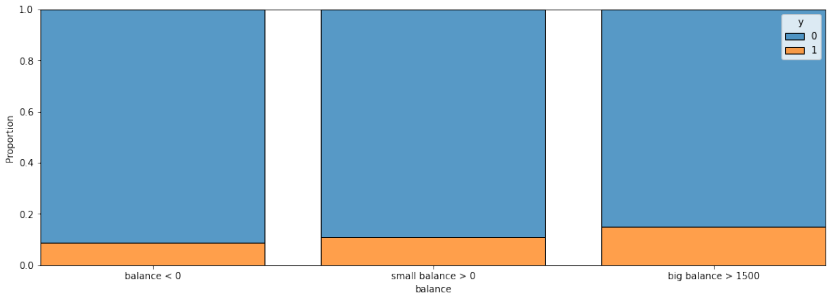
Do transformacji zmiennych utworzyliśmy specjalnie do tego stworzone klasy, dzięki którym dużo łatwiej było zarządzać modelem. Klasy te zostały następnie połączone w pipeline z pakietu sklearn. Próbowaliśmy na różne sposoby transformować zmienne, ale poniżej zapisane będą te, które według nas najlepiej wypadły i zostały zastosowane w ostatecznym modelu.

3.2.1 Numerycznych

- **duration, age, campaign**:
Zastosowaliśmy tutaj zwykłe liniowe przekształcenie na przedział [0, 1].

- **pdays, previous**:
Jako, że pdays było mocno skolerowane z previous (współczynnik korelacji pearsona wynosił tu około 0.6), a previous było mocniej skolerowane z naszym targetem niż pdays to wyrzuciliśmy zmienną pdays.
Ponadto dla previous uznaliśmy, że najlepiej będzie ją zbinnować, ponieważ wartość większości obserwacji dla tej cechy wynosiła 0. Stąd przetransfomowaliśmy tę zmienną tak, że:
- przyjmuje 0 dla wartości previous = 0
- przyjmuje 1 w przeciwnym przypadku.

- **balance**:
Na początku nie było widać zależności target od tej zmiennej. Jednak po zasosowaniu odpowiedniego kubełkowania tej zmiennej uzyskaliśmy następujący rezultat:



Uznaliśmy, że możemy zatem zastować zatem kubełkowanie tej zmiennej:
- 0, gdy balance ≤ 0
- 1, gdy balance > 0 i balance ≤ 1500
- 2, gdy balance > 1500

- **day**:
Zastosowaliśmy tu tzw. "transformację na koło", które ma ukazywać to, że pierwszy dzień miesiąca jest blisko ostatniego dnia miesiąca. Próbka kodu:
*bank_data["day"].apply(lambda x: np.sin(x * (2 * np.pi / 31)))*

3.2.2 Kategoryczne

- **default, housing, loan** (binarne):
Zwykła transformacja no = 0, yes = 1.

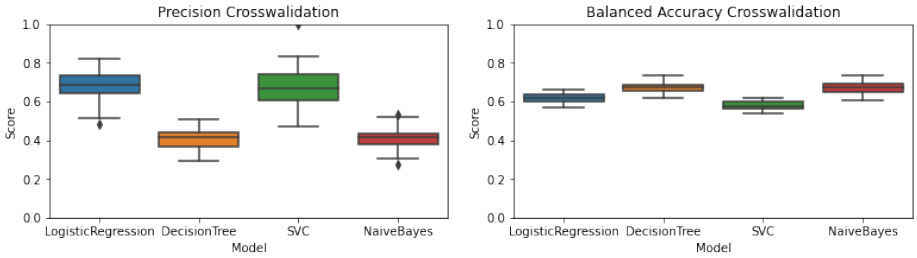
- **default**:
Zmienna binarna, która wykazywała bardzo niską wariancję (1.9% obserwacji yes). Uznaliśmy, że należy ją wyrzucić.

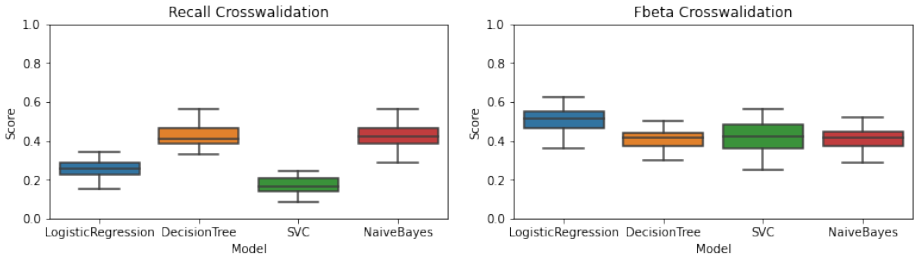
- **education**
Oridinal encoding (zmienna wykazywała porządek):
bank_data['education'].map('unknown': 0, 'primary': 1, 'secondary': 2, 'tertiary': 3)

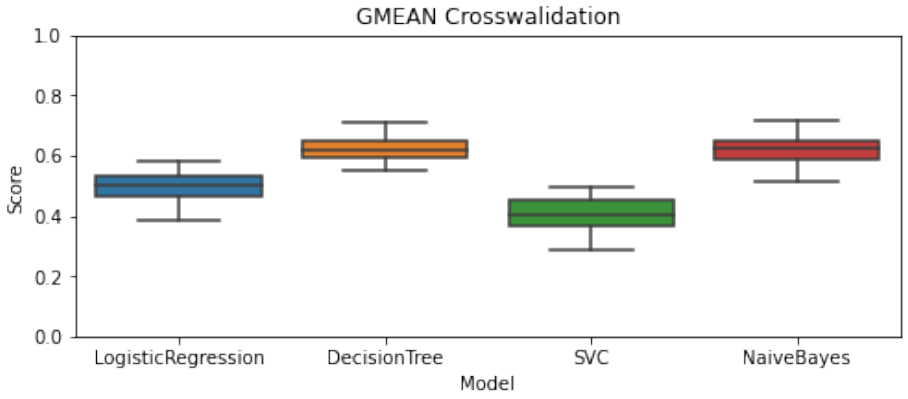
- **month**
Również oridinal encoding, ale z transformacją kołową:
bank_data['month'] = bank_data["month"].map("jan": 0, "feb": 1, "mar": 2, "apr": 3, "may": 4, "jun": 5, "jul": 6, "aug": 7, "sep": 8, "oct": 9, "nov": 10, "dec": 11)
*bank_data['month'] = bank_data["month"].apply(lambda x: np.sin(x * (2 * np.pi / 12)))*

- **job, contact, marital,poutcome**
Zastosowano tu zwykły OneHotEncoding, ale z wyrzuceniem niektórych niezidentyfikowanych kategorii: 'poutcome_unknown', 'poutcome_other', 'contact_unknown', 'job_unknown'

3.3 Wyniki prostych modeli po preprocessingu







4 Ostateczny model i hiperparametry

Hiperparametrów szukaliśmy na dwa sposoby. W przypadku RandomForest posłużyliśmy się BayesSearchSV z pakietu scikit-optimize , natomiast w przypadku reszty modeli szukaliśmy używając GridSearchCV. Jako metrykę, którą powyższe "maszyny" miały optymalizować wybraliśmy fbeta i gmean. Stąd dla każdego modelu wybraliśmy dwie kombinacje najlepszych parametrów: jedną dla przeszukiwania fbeta, drugą dla gmean.

4.1 Siatki parametrów

Poniżej zamieszczamy przestrzenie parametrów jakie przeszukiwaliśmy dla konretnych modeli:

- **RandomForest** (BayesSearchCV):
"n_estimators": Integer(50, 150),
"criterion": Categorical(["gini"]),
"max_depth": Integer(4, 21),
"min_samples_split": Integer(2, 8),
"max_features": Integer(2, 10)

- **GradientBooster** (GridSearchCV):
"learning_rate": [0.05, 0.1, 0.2],
"max_iter": [1000],
"max_depth": [2, 3, 5, 8, 30],
"max_leaf_nodes": [10, 30, None],
"l2_regularization": [0, 0.1, 0.5, 1, 10]

- **XGBooster** (GridSearchCV):
"eta": [0.05, 0.1, 0.2],
"max_depth": [3, 6, 12, 30],
"gamma": [0, 5, 10, 30, None],
"min_child_weight": [1, 5, 10, 20],
"lambda": [0.1, 0.5, 1, 5],
"alpha": [0, 0.5, 1, 5],
"scale_pos_weight": [7.67]

- **SVC** (GridSearchCV):
"kernel": ["linear", "poly", "rbf", "sigmoid"],
"degree": [1, 3, 5, 7],
"max_iter": [150000],
"probability": [True]

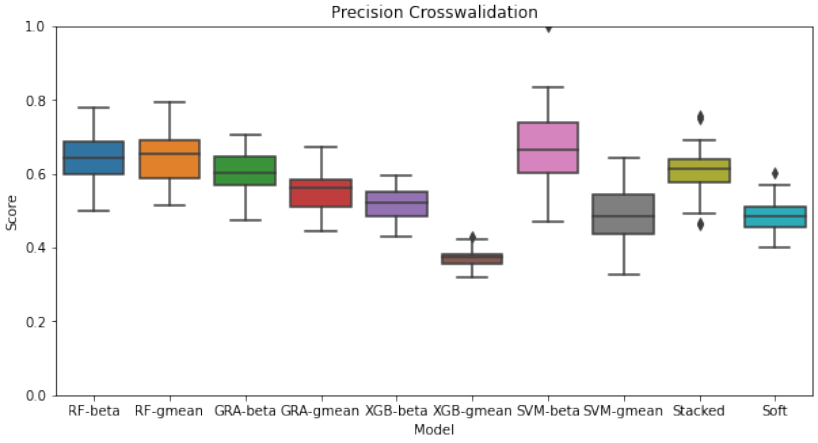
4.2 Modele łączone

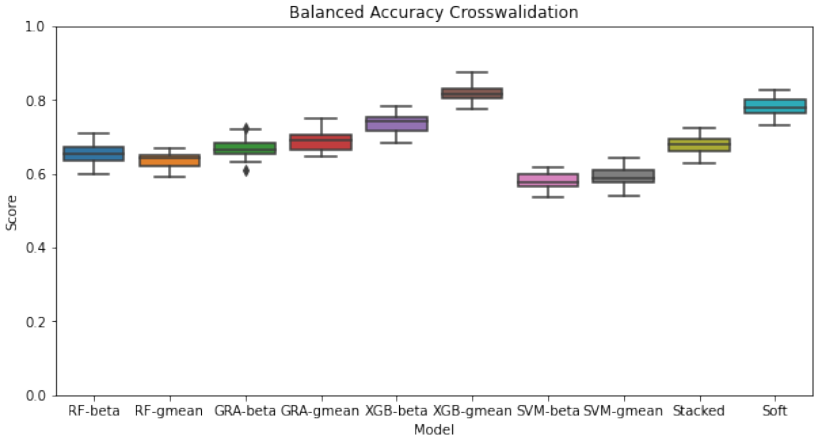
Po zobaczeniu, że najlepiej działają modele XGBoost, postanowaliśmy, że sprawdzimy jak działają ich łączone wersje, czyli:

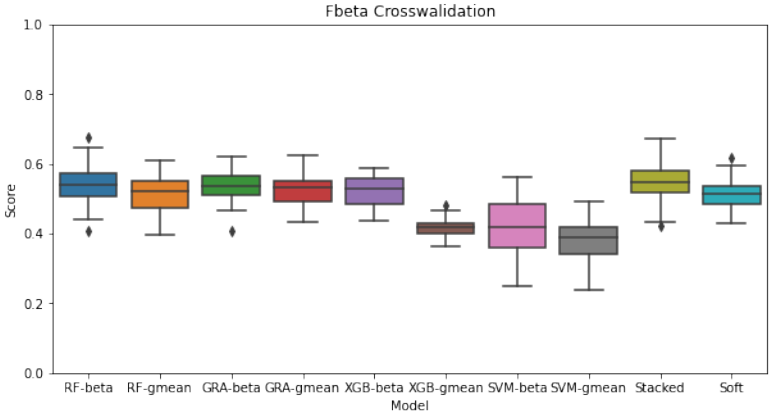
- **stacked** - model łączący za pomocą StackingClassifier dwa estymatory XGBoost: jeden z najlepszymi parametrami dla fbeta i jeden z najlepszymi dla gmean. Jako estymator łączący te dwa przyjęliśmy estymator LogisticRegression

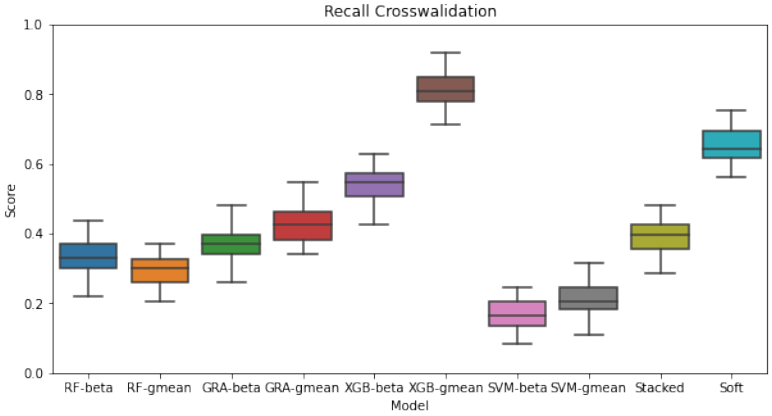
- **soft** - model również łączący XGBoost'y takie jak poprzednio, ale za pomocą VotingClassifier z paramerem voting="soft"

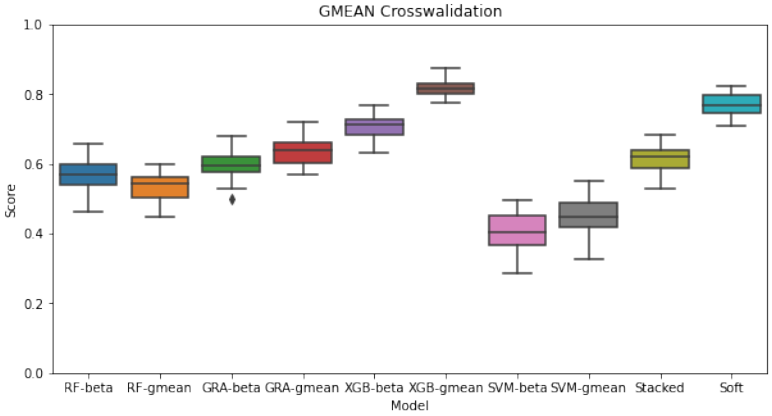
4.3 Ewaluacja modeli zaawansowanych






Fbeta Crosswalidation
Score
Model
RF-beta
RF-gmean
GRA-beta
GRA-gmean
XGB-beta
XGB-gmean
SVM-beta
SVM-gmean
Stacked
Soft


Recall Crosswalidation
Score
Model
RF-beta
RF-gmean
GRA-beta
GRA-gmean
XGB-beta
XGB-gmean
SVM-beta
SVM-gmean
Stacked
Soft


GMEAN Crosswalidation
Score
Model
RF-beta
RF-gmean
GRA-beta
GRA-gmean
XGB-beta
XGB-gmean
SVM-beta
SVM-gmean
Stacked
Soft

Na podstawie powyższych wykresów uznaliśmy, że najlepszym modelem przez nas stworzonym jest model **soft**, który jest kompromisem między naszymi głównymi metrykami: gmean i fbeta. To właśnie ten model przekazaliśmy walidatorom.

4.4 Wynik walidacji

Wytrenowaliśmy zatem model **soft** na naszych danych. Warto zaznaczyć jakie parametry miały modele XGBoost w tym modelu łączonym:

dict_xgboost_fbeta =
'alpha': 1,
'eta': 0.2,
'gamma': 0,
'lambda': 5,
'max_depth': 12,
'min_child_weight': 1,
'random_state': 17,
'scale_pos_weight': 7.67

dict_xgboost_gmean =
'eta': 0.1,
'max_depth': 3,
'gamma': 5,
'min_child_weight': 20,
'lambda': 0.5,
'alpha': 5,
'scale_pos_weight': 7.67,
'random_state': 17

Walidatorzy na swoim zbiorze danych uzyskali następujące wyniki:
- precision: 0.49
- gmean: 0.7581900507921896
- fbeta: 0.5125523012552301
- balanced accuracy: 0.7716379512799163
- recall: 0.6282051282051282

Wyniki mieszczą się w przedziałach, które wyznaczyły boxploty crosswalidacyjne z naszych danych.

5 Dodatki

5.1 AutoML

W celu weryfikacji sprawności naszego modelu postanowiliśmy sprawdzić, jak poradziłby sobie model wytrenowany przez narzędzie AutoML. W tym celu skorzystaliśmy z pakietu TPOT i pozwoliliśmy obiektowi TPOTClassifier przez godzinę sprawdzać najlepszy pipeline dla naszego zbioru danych w celu optymalizacji metryki **gmean**. Niestety nawet po przeszukaniu portalu Stack Overflow nie byliśmy w stanie znaleźć jak znaleźć wszystkie parametry i całą logikę pipelinu, który tpot znalazł. Stąd jedyne co mogliśmy tu zrobić, to porównać wyniki pipelinu i naszego modelu wytrenowanego na części naszego zbioru na zbiorze testowym. Oto one:

Wynik modelu soft
- Precision score: 0.4888888888888889
- G-mean score: 0.7437936291862172
- Accuracy score: 0.8815165876777251
- Fbeta_score: 0.5080831408775981
- Recall score: 0.6027397260273972

Wynik modelu wytrenowanego przez tpot
- Precision score: 0.3333333333333333
- G-mean score: 0.8084376466811949
- Accuracy score: 0.7883096366508688
- Fbeta_score: 0.37888198757763975
- Recall score: 0.8356164383561644

W tpot nie spawdzaliśmy jednocześnie wszystkich metryk, a tylko gmean. Widać, że lepiej sobie w tym polu poradził. Jednak patrząc ogólnie na nasze metryki wydaję się, że nasz model poradził sobie całkiem porównywalnie (a nawet lepiej) niż ten z tym wyszkolonym przez TPOT. Zatem raczej wycisnęliśmy z danych tyle, ile się dało.

6 Bibliografia

https://scikit-learn.org/stable/
https://scikit-optimize.github.io/stable/
https://epistasislab.github.io/tpot/
https://www.kaggle.com/datasets/adilashrafi/bank-marketing-classification-task
https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/