Zaawansowane metody uczenia maszynowego
Projekt 1

Mikołaj Małkiński
malkinskim@student.mini.pw.edu.pl

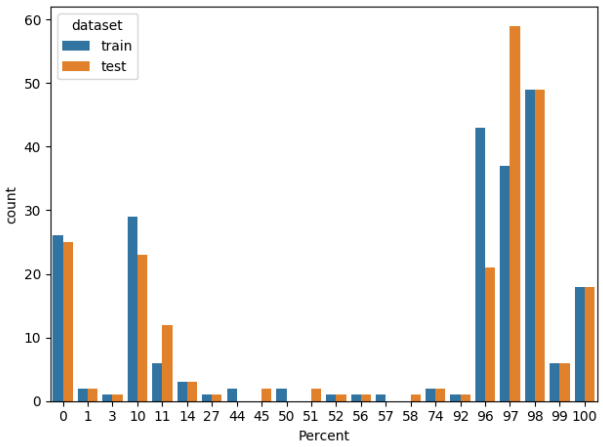
29 kwietnia 2019

1 Przygotowanie danych

1.1 Brakujące dane

Zbiór danych zawiera wiele kolumn które nie są w pełni wypełnione danymi. Analiza wykazała, że tylko 20 kolumn posiadały wszystkie wartości. Rysunek 1 przedstawia zależność między stopniem braku danych a liczbą kolumn. Istnieje kilka możliwych podejść które można zastosować w tym przypadku. Po pierwsze, można kompletnie zignorować i wybrać model który jest w stanie sam odpowiednio obsłużyć luki w zbiorze. Przykładem takiego modelu jest *XGBoost*. Jednakże, nie wszystkie algorytmy do klasyfikacji są przygotowane na braki w danych, Dlatego, aby móc porównać działanie kilku modeli na tym samym zbiorze podjęto decyzję o wypełnieniu brakujących danych.

Niektóre z kolumn posiadały nawet ponad 90% brakujących danych. Nie mając żadnych informacji o zbiorze danych oraz nie wiedząc co dana kolumna reprezentuje, ciężko stwierdzić z czego wynika taki duży brak. Może to oznaczać zwyczajnie brak pomiaru, wartość ważną równie dobrze jak ta która istnieje w danej kolumnie lub w danym przypadku wartość w tej kolumnie może nie mieć żadnego znaczenia dla konkretnego wiersza. Ze względu na duży rozmiar zbioru danych, podjęto decyzję o kompletnym usunięciu części z takich kolumn, które mają więcej braków niż 90%. Resztę poddano imputacji.



Rysunek 1: Liczba kolumn posiadających dany procent brakujących danych z podziałem na zbiór treningowy i testowy

Kolumny w zbiorze można podzielić na 2 rodzaje: *numeryczne* i *kategoryczne*. Aby wypełnić pierwsze z nich, brakujące dane w każdej kolumnie wypełniono jej medianą. Oczywiście w innych przypadkach mogłyby zostać

użyte także inne funkcje, takie jak średnia lub moda. W przypadku kolumn kategorycznych, dodano nową kategorię: *unknown*, która wypełniła braki w tych kolumnach.

1.2 Unikalność danych

Następnie, analizie poddano liczbę unikalnych wartości dla każdej z kolumn. Zbiór zawierał kolumny wypełnione tylko jedną tą samą wartością. Takie kolumny nie niosą ze sobą żadnej wartości więc zostały one usunięte Dodatkowo, część kolumn kategorycznych, posiadały bardzo dużo unikalnych wartości. Można przypuszczać że są to dane tekstowe, które także nie przyniosą pozytywnych efektów w klasyfikacji. Dlatego, kolumny mające więcej niż 100 unikalnych wartości zostały usunięte.

1.3 Kolumny kategoryczne

Niektóre algorytmy klasyfikacji wymagają aby zbiory na których zostaną użyte posiadały tylko cechy numeryczne. Z tego powodu, cechy kategoryczne musiały zostać przekształcone w liczbowe. Najprostszym sposobem jest zamienienie każdej z kategorii na unikalną liczbę. Jednakże to implikuje pewien porządek w tej kolumnie, który tak na prawdę nie występuje. Dlatego, każdą kolumnę przekształcono w n nowych kolumn, gdzie n to liczba kategorii dla danej kolumny. Proces ten nazywany jest *One-hot encoding*.

1.4 Porównawcza analiza danych

Kolejnym krokiem aby lepiej zrozumieć zbiór danych było porównanie rozkładów cech ze względu na zbiór. Wizualizacja zaprezentowana w Dodatku A wykazała że dane niewątpliwie pochodzą z tego samego rozkładu. Porównano także rozkłady zmiennych z podziałem na klasę. Wyniki są przedstawione w Dodatku B. Można wyraźnie zaobserwować że zmienne takie jak Var38, Var73, Var126, Var153 różnią się w zależności od klasy.

2 Klasyfikacja

2.1 Użyte modele

Po odpowiednim przygotowaniu danych, w prosty sposób można sprawdzić jakość różnych modeli z domyślnymi parametrami. Zbiór danych został podzielony na treningowy i walidacyjny w stosunku 4:1. Zbadano jakość następujących klasyfikatorów: *XGBoost* [4], *XGBoost balanced* - XGBoost biorący pod uwagę balans klas, *LightGBM* [9], *LightGBM balanced* - LightGBM biorący pod uwagę balans klas, *CatBoost* [5], *CatBoost balanced* - CatBoost biorący pod uwagę balans klas, *AdaBoost* [6], *GradientBoosting* [7], *RandomForest* [3], *BalancedRandomForest* [1], *ExtraTrees* [8].

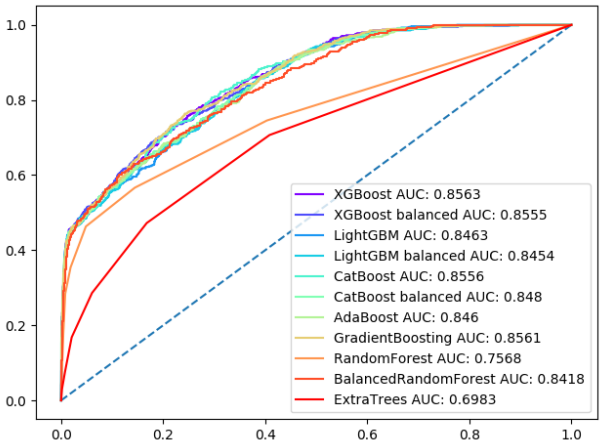
Krzywe ROC tych modeli zaprezentowane są na Rysunku 2. Wyraźnie widać że wszystkie modele z wyjątkiem RandomForest (las losowy) i ExtraTrees (ekstremalnie losowe drzewa) osiągnęły porównywalne wyniki. Dokładniejsze statystki przedstawione są w Tablicy 1. Porównano w niej takie metryki jak: AUC, dokładność, precyzja@10 (procent obserwacji poprawnie zklasyfikowanych jako należące do klasy 1 z 10% najwyżej ocenionych obserwacji),

	AUC	Dokładność	Precyzja@10	F1	Precyzja	Czułość
XGBoost	**0.8563**	**0.9516**	**0.3600**	**0.5442**	**0.7524**	0.4262
XGBoost balanced	0.8555	0.7742	0.3588	0.3043	0.1923	0.7288
LightGBM	0.8463	0.9508	0.3575	0.5310	0.7483	0.4114
LightGBM balanced	0.8454	0.8654	0.3488	0.3631	0.2672	0.5664
CatBoost	0.8556	0.9514	0.3500	0.5418	0.7492	0.4244
CatBoost balanced	0.8480	0.8450	0.3563	0.3467	0.2426	0.6070
AdaBoost	0.8460	0.9505	0.3550	0.5308	0.7417	0.4133
GradientBoosting	0.8561	0.9510	0.3588	0.5410	0.7404	0.4262
RandomForest	0.7568	0.9404	0.3250	0.2891	0.7519	0.1790
BalancedRandomForest	0.8418	0.7332	0.3575	0.2707	0.1661	**0.7306**
ExtraTrees	0.6983	0.9328	0.2162	0.0561	0.5714	0.0295

Tablica 1: Metryki podstawowych modeli klasyfikacyjnych

F1, precyzja oraz czułość. Należy pamiętać że analizowany zbiór danych ma silnie niezbalansowane klasy, dlatego porównując modele trzeba wziąć pod uwagę wszystkie przedstawione metryki.

Warto porównać wyniki modeli XGBoost, LightGBM i CatBoost w standardowej konfiguracji z konfiguracją mająca na celu zmniejszyć problem nie zbalansowanych klas (oznaczone przez sufiks *balanced*). Dla każdego z tych modeli, konfiguracja zbalansowana znacząco poprawiła czułość modelu, kosztem precyzji, ale pogorszyła też wartość metryki F1. Nie można jednak jednoznacznie stwierdzić która z tych konfiguracji jest lepsza dla omawianego zbioru danych ponieważ metryka która służy do ostatecznej ewaluacji (precyzja@10) jest niemalże identyczna w wszytkich przypadkach.



Rysunek 2: Porównanie krzywych ROC podstawowych modeli klasyfikacyjnych

2.2 Balansowanie klas

Podczas eksperymentów można było zaobserwować skłonność modeli do klasyfikowania większości obserwacji do klasy 0. Było to spowodowane nierównym stosunkiem liczby elementów pomiędzy klasami. Zbiór danych posiadał niecałe 8% obserwacji należących do klasy 1. Aby rozwiązać ten problem, przeanalizowano następujące metody:

- zmniejszenie ilości obserwacji z klasy 0,
- zwiększenie ilości obserwacji z klasy 1,
- dodanie sztucznych obserwacji z klasy 1 używając techinki SMOTE [2].

Niestety, żadne rozwiązanie nie przyniosło pozytywnych rezultatów.

	AUC	Dokładność	Precyzja@10	F1	Precyzja	Czułość
Próba 1	0.8605	0.7779	0.4000	0.3210	0.2071	0.7131
Próba 2	0.8757	0.7960	0.4263	0.3625	0.2395	0.7448
Próba 3	0.8702	0.7770	0.4075	0.3283	0.2099	0.7530
Próba 4	0.8621	0.7758	0.3987	0.3215	0.2055	0.7378
Próba 5	0.8569	0.7755	0.3812	0.3087	0.1971	0.7110
Średnia	0.8651	0.7804	**0.4027**	0.3284	0.2118	0.7319

Tablica 2: Metryki zoptymalizowanego modelu XGBoost używając kroswalidacji

2.3 Wybór cech

Przygotowany zbiór danych posiadał 470 różnych kolumn. Aby ograniczyć ich liczbę, spróbowano wybrać te, które są najważniejsze do dokonania poprawnej klasyfikacji. W tym celu, po wytrenowaniu modeli XGBoost, LightGBM oraz CatBoost (w domyślnej konfiguracji oraz w konfiguracji biorącej pod uwagę balans klas), dla każdego z nich wzięto po 30 cech które okazały się najważniejsze w treningu. Po złączeniu tych zbiorów otrzymano nowy zbiór zawierający 57 różnych cech. Następnie, wszystkie wyżej wspomniane modele zostały ponownie wytrenowane na zmniejszonym zbiorze danych. Niestety, osiągnięte wyniki nie przyniosły żadnych pozytywnych rezultatów.

3 Wybrany model

3.1 Dobór hiperparametrów

Na podstawie analizy wyników podstawowych modeli klasyfikacyjnych, zdecydowano o dokładniejszym zbadaniu modelu XGBoost. Aby osiągnąć wyższe rezultaty, wykorzystano metodę *Grid Search* do wyszukania optymalnych parametrów. Wzięto pod uwagę następujące parametry modelu: stosunek klas, maksymalna głębokość drzew wchodzących w skład komitetu (base learners), współczynnik uczenia, liczbę członków komitetu oraz współczynniki próbkowania. Dla każdej kombinacji tych parametrów sprawdzono jakość modelu metodą kroswalidacji na 3 różnych próbach. Aby osiągnąć bardziej wiarygodne wyniki warto zwiększyć liczbę prób do co najmniej 5, jednak ze względu na długi czas obliczeń nie było takiej możliwości. Zostały wybrane te parametry, dla których model osiągnął najwyższą wartość metryki precyzja@10.

3.2 Kroswalidacja

Dla dobranych hiperparametrów została przeprowadzona ostateczna ocena jakości modelu. Ponownie użyto metody kroswalidacji dla 5 różnych prób. Wyniki modelu dla każdej próby były do siebie zbliżone. Dokładne wyniki są przedstawione w Tablicy 2. Średnia wartość metryki precyzja@10 wyniosła 40.27%.

4 Podsumowanie

Osiągnięty wynik (40.27%) znacznie odbiega od idealnego (70%). Jednakże, biorąc pod uwagę fakt silnej anonimizacji zbioru danych, jest to wynik satysfakcjonujący. Warto zauważyć, że wyniki zastosowanych modeli klasyfikacyjnych były do siebie zbliżone, a optymalizacja hiperparametrów dla wybranego modelu (XGBoost) przyniosła poprawę jego jakości zbliżoną do 4%. Dlatego przypuszcza się że główną przeszkodą w osiągnięciu wyższej dokładności jest słaba znajomość zbioru danych i ograniczone możliwości jego analizy.

Literatura

[1] A. M. Yağcı , T. Aytekin , and F. S. Gürgen . Balanced random forest for imbalanced data streams. In *2016 24th Signal Processing and Communication Application Conference (SIU)*, pages 1065–1068, May 2016.

[2] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[4] Tianqi Chen and Carlos Guestrin. XGBoost : A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[5] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, and Aleksandr Vorobev. Fighting biases with dynamic boosting. *CoRR*, abs/1706.09516, 2017.

[6] Yoav Freund and Robert E Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780, 10 1999.

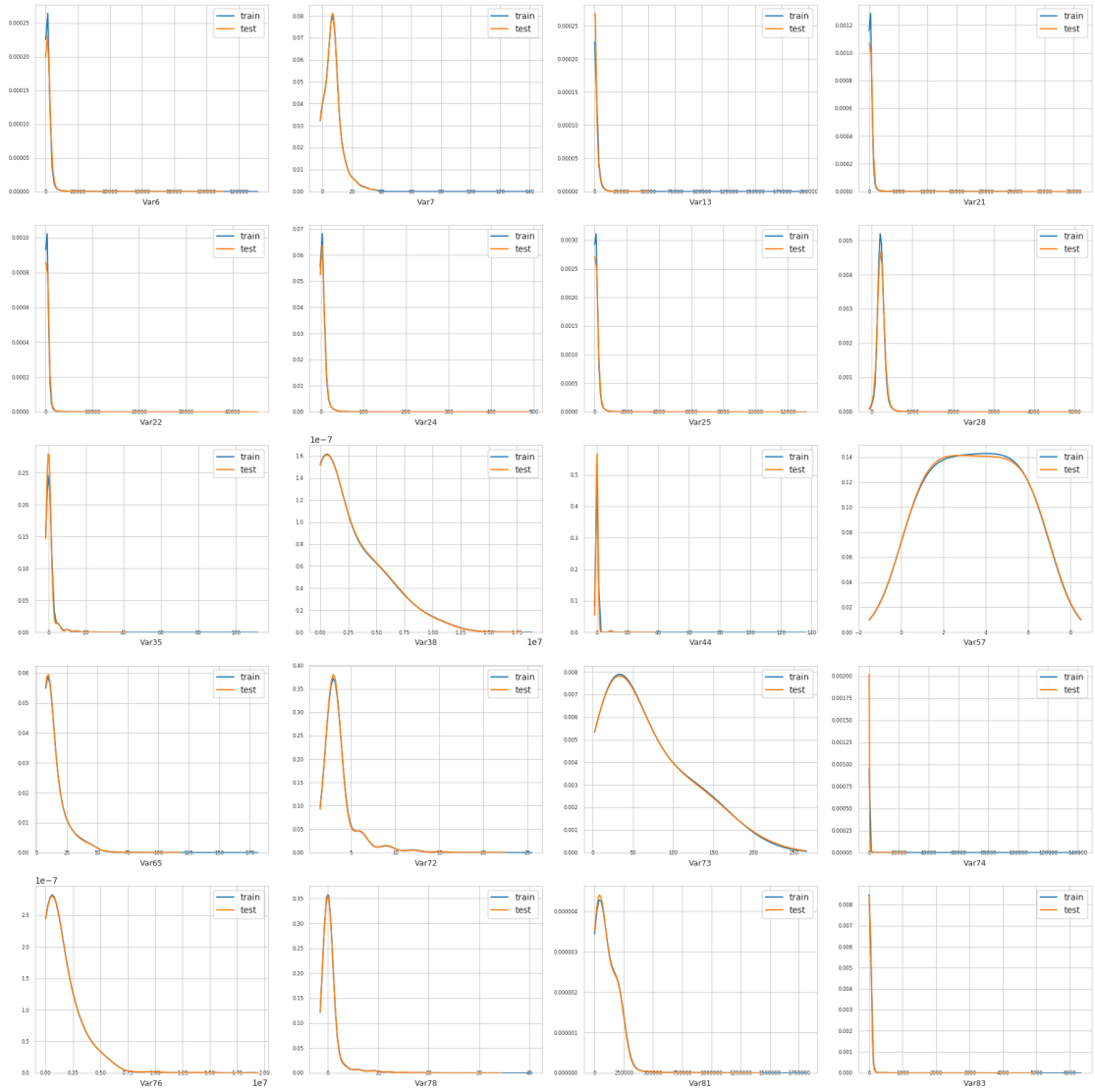
[7] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232, 10 2001.

[8] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, Apr 2006.

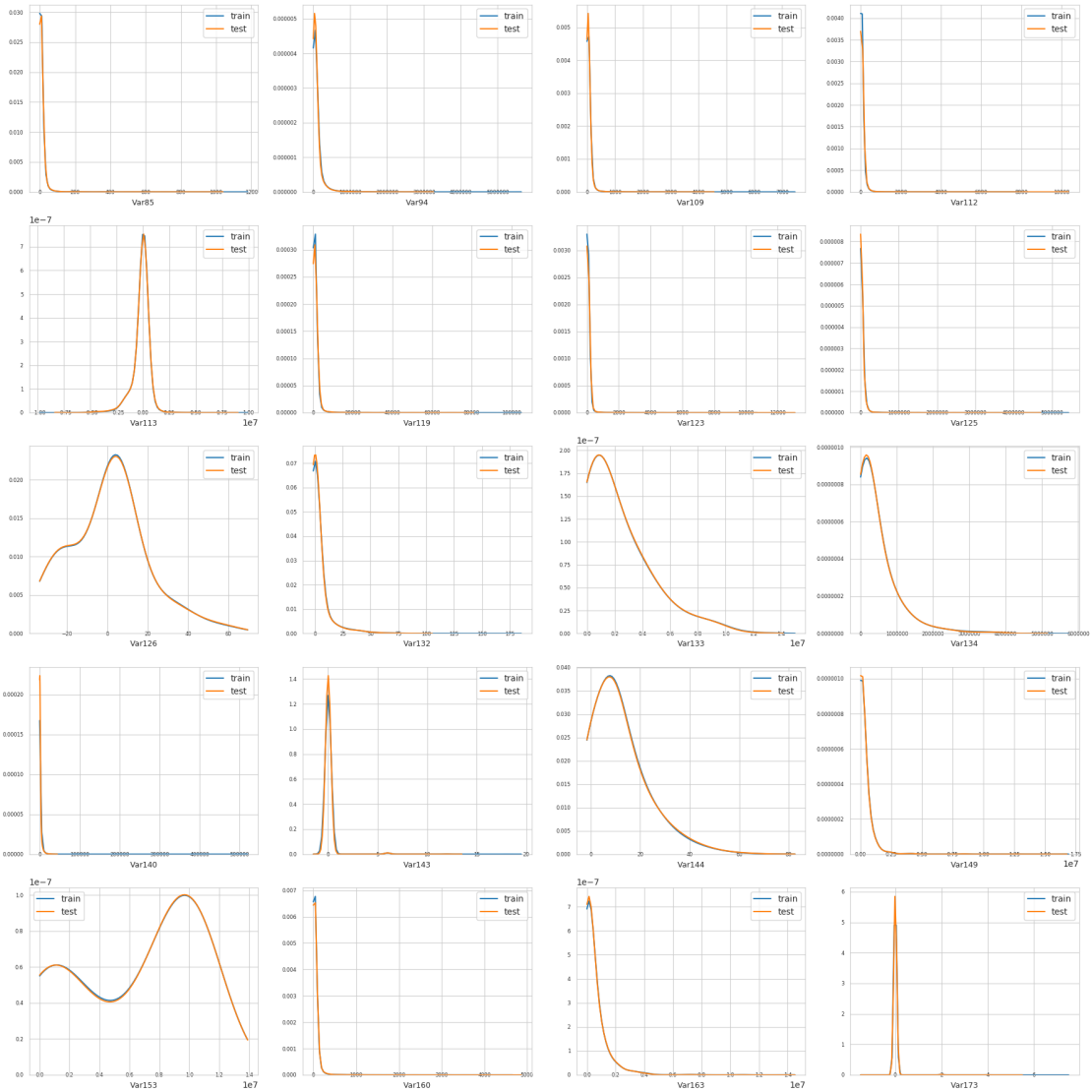
[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017.

Dodatki

A Rozkład cech numerycznych z podziałem na zbiór

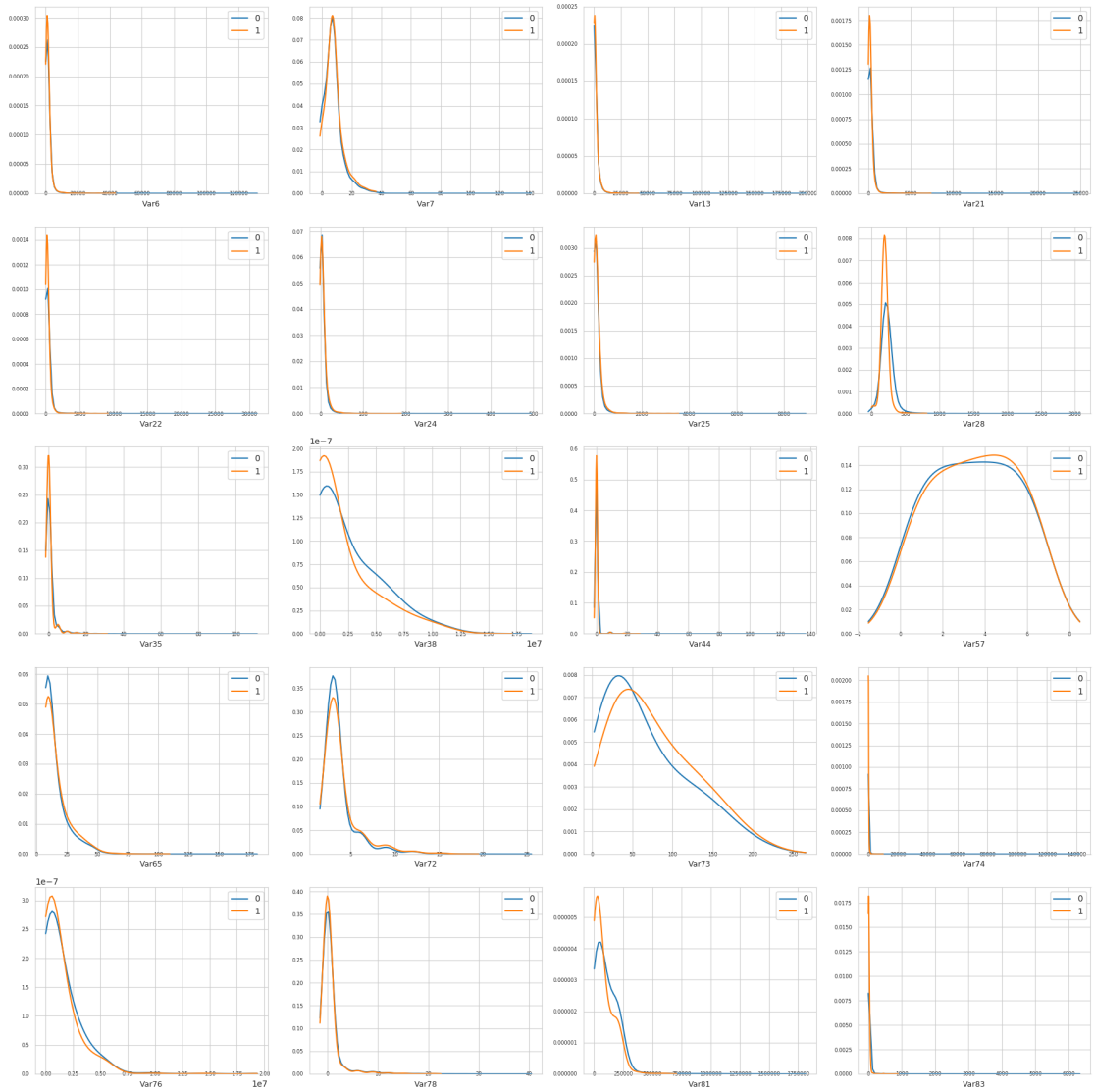


Rysunek 3: Rozkład cech numerycznych z podziałem na zbiór

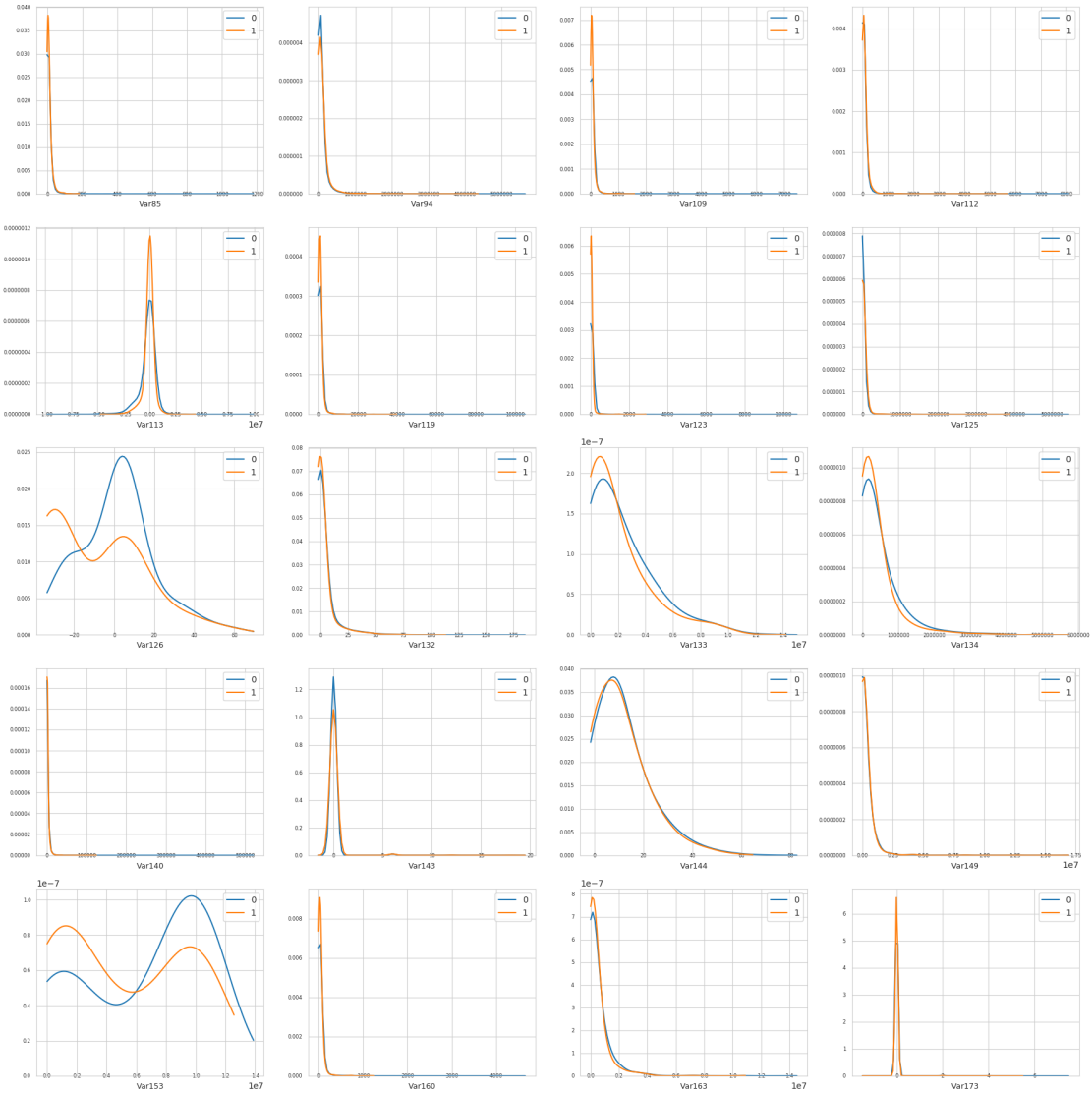


Rysunek 4: Rozkład cech numerycznych z podziałem na zbiór

B Rozkład cech numerycznych z podziałem na klasę



Rysunek 5: Rozkład cech numerycznych z podziałem na klasę



Rysunek 6: Rozkład cech numerycznych z podziałem na klasę