Klasyfikacja na zbiorze Home Credit Default Risk

Projekt nr 1

Wstęp do Uczenia Maszynowego
Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

Aleksandra Wichrowska, Karol Pysiak
Grupa F5

30 kwietnia 2019

Streszczenie

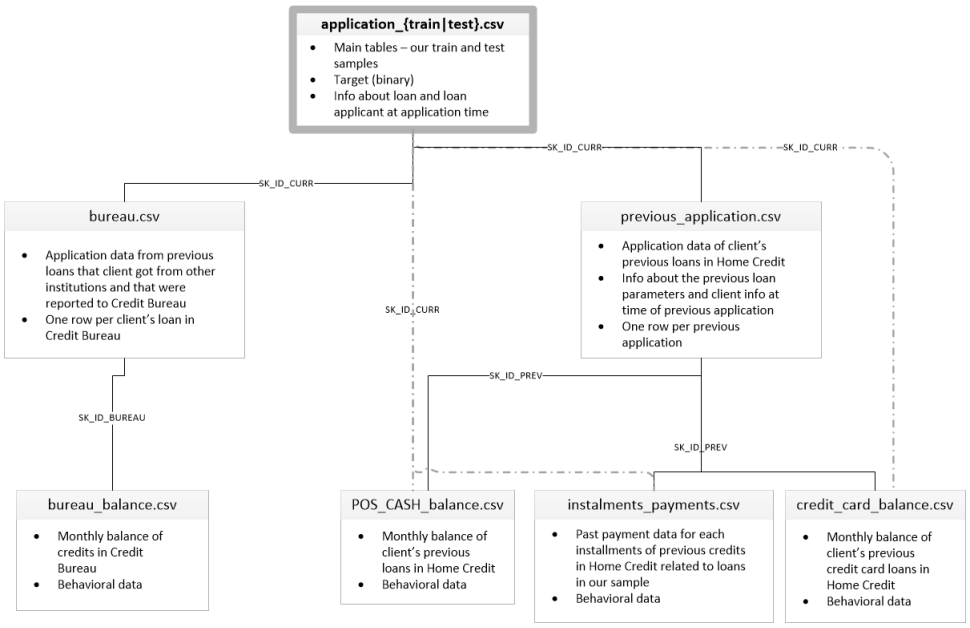
Celem naszego projektu była klasyfikacja klientów pod względem problemów ze spłatą kredytu. Przed rozpoczęciem zaawansowanej pracy z danymi przeprowadziliśmy wstępną eksplorację danych, aby poznać z jakimi danymi dokładnie mamy doczynienia, jak są one zbudowane oraz co możemy z nich uzyskać. Następnie przeszliśmy do inżynierii cech, która pozwoliła nam wyodrębnić najważniejsze zmienne oraz stworzyć nowe, bardziej adekwatne do problemu zmienne. Dalej przeszliśmy do wybrania odpowiednich modeli klasyfikacyjnych, by na końcu dostroić parametry optymalnym modelom.

Spis treści

1 Wstępna eksploracja danych

1.1 Zależnosci między zbiorami danych

Pierwszym krokiem do zrozumienia dostępnych danych było przyjrzenie się zależnosciom między poszczególnymi zbiorami danych. Prezentuje to poniższy schemat:



Rysunek 1: Schemat zależności tabel w zbiorze danych

1.2 Krótki opis zbiorów danych

Zbiór *application_train* jest główną składową dostępnych danych. Zawiera dane o wszystkich bieżących aplikacjach kredytowych do "naszego banku".

Zbiór *bureau* zawiera informacje o poprzednich kredytach klientów "naszego banku" w innych placówkach. Dodatkowo zbiór *bureau_balance* zawiera dane o miesięcznym bilansie kredytowym.

Zbiór *previous_application* zawiera w sobie informacje o poprzednich kredytach klientów w 'naszym banku'. Zależne od niego zbiory to *credit_card_balance* oraz *installments_payments*. Dla poszczególnych kredytów zawierają one odpowiednio informacje o bilansie karty kredytowej w kolejnych miesiącach oraz o spłacanych ratach.

1.3 Podstawowe informacje o zbiorach danych

nazwa zbioru	wiersze	kolumny	kolumny numeryczne	kolumny kategoryczne
application_train	184507	123	106	19
bureau	1716428	17	14	3
bureau_balance	27299925	3	2	1
previous_application	1670214	37	21	16
credit_card	3840312	23	22	1
installment_payments	13605401	8	8	0

1.4 Wartosci NA

1.4.1 application_train

Sumarycznie zbiór danych zawiera około 24% wartosci NA.

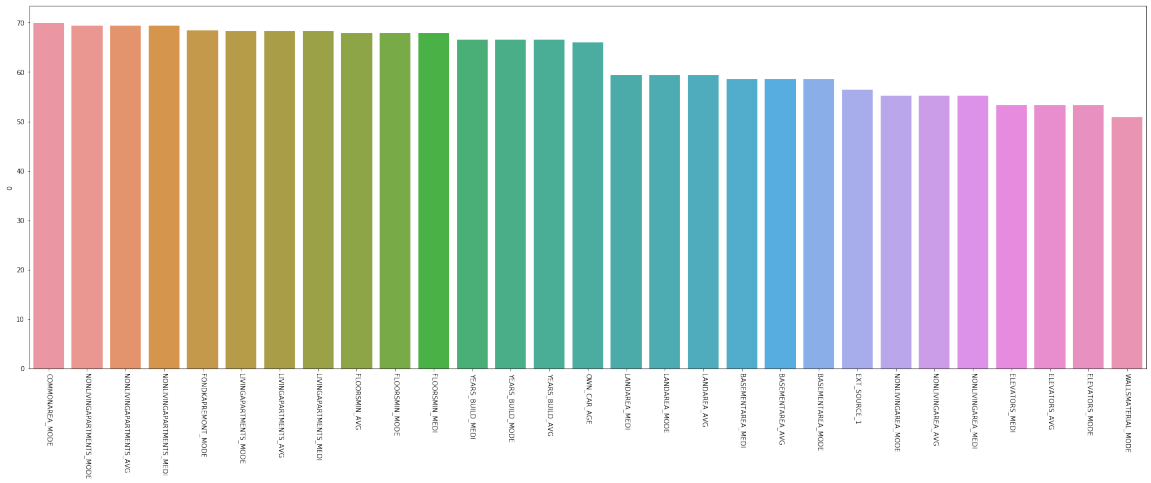
41 kolumn zawiera ponad połowę brakujących wartosci.

Poniżej fragment tabeli pokazującej, jaki procent wartości w kolumnie stanowią NA:

COMMONAREA_AVG	69.872297
COMMONAREA_MODE	69.872297
NONLIVINGAPARTMENTS_MODE	69.432963
NONLIVINGAPARTMENTS_AVG	69.432963
NONLIVINGAPARTMENTS_MEDI	69.432963
FONDKAPREMONT_MODE	68.386172
LIVINGAPARTMENTS_MODE	68.354953
LIVINGAPARTMENTS_AVG	68.354953
LIVINGAPARTMENTS_MEDI	68.354953
FLOORSMIN_AVG	67.848630
FLOORSMIN_MODE	67.848630
FLOORSMIN_MEDI	67.848630
YEARS_BUILD_MEDI	66.497784
YEARS_BUILD_MODE	66.497784
YEARS_BUILD_AVG	66.497784
OWN_CAR_AGE	65.990810
LANDAREA_MEDI	59.376738

Rysunek 2: Tabela przedstawiająca procent brakujących wartości w kolumnach *application*

oraz wykres



Rysunek 3: Wykres przedstawiający stosunek brakujących wartości w kolumnach *application*

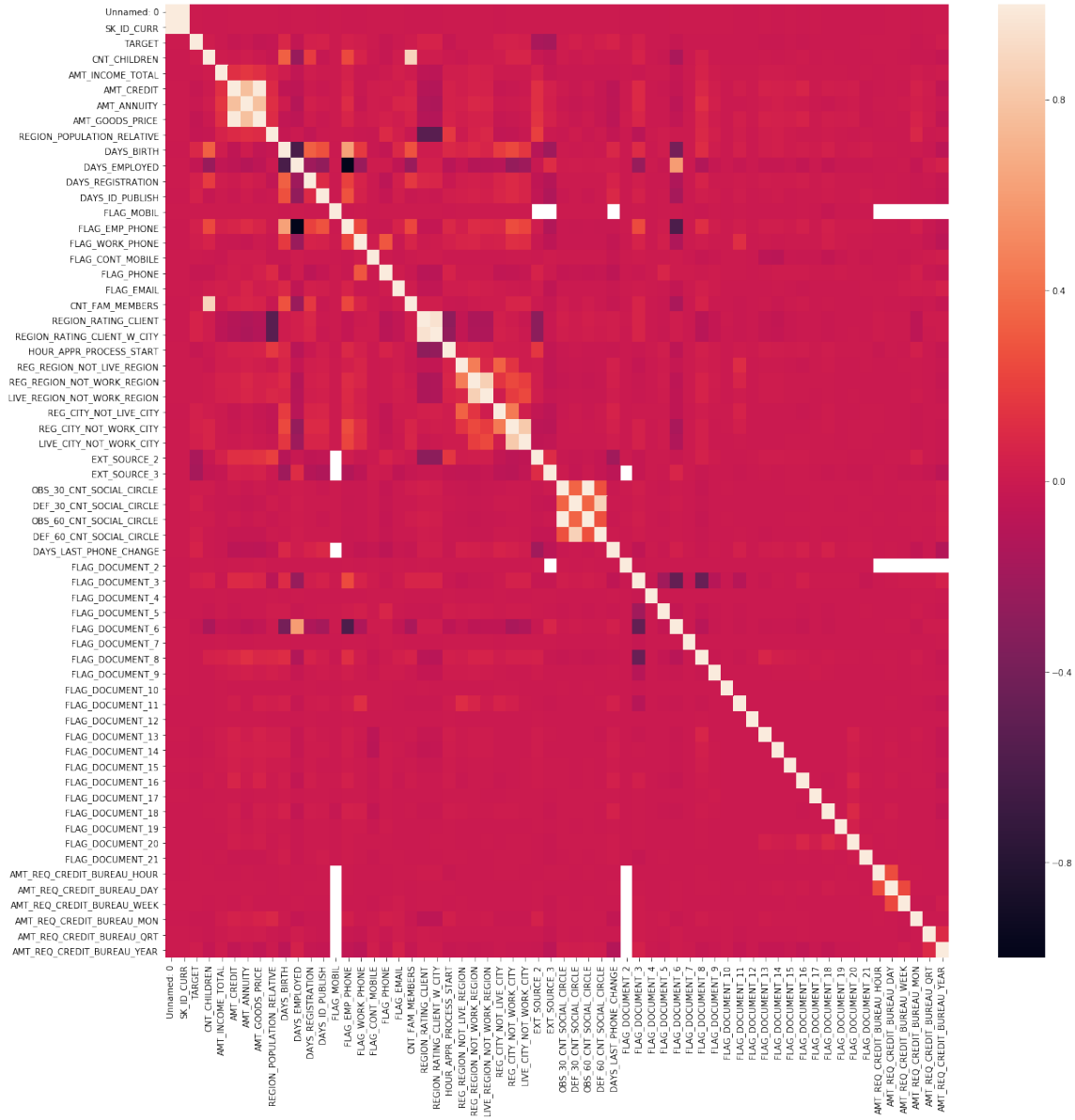
Kolumny z ponad połową brakujących wartości zdecydowaliśmy się od razu odrzucić, uznając je za bezużyteczne dla naszego modelu. Pozostałe braki danych będziemy starali się uzupełnić w kolejnej fazie projektu.

1.4.2 Pozostałe zbiory

nazwa zbioru	% NA	kolumny z NA	kolumny z ponad połową NA
bureau	13.5	7	3
bureau_balance	0	0	0
previous_application	18	16	4
credit_card	6.5	9	0
installment_payments	0	2	0

1.5 Analiza zależnosci z targetem (application_train)

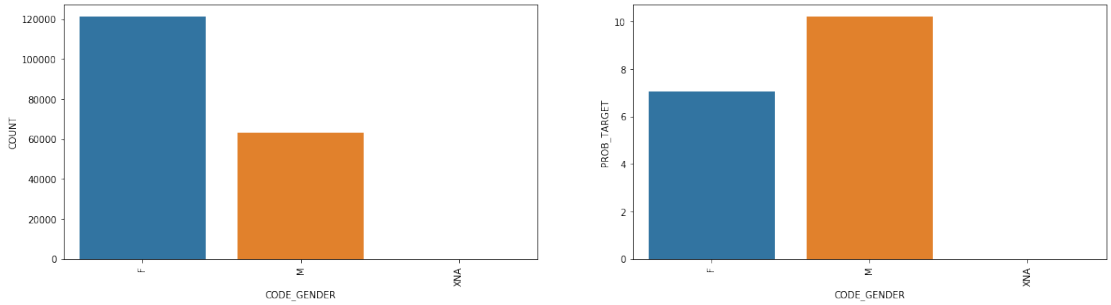
Wykres korelacji zmiennych:



Rysunek 4: Wykres korelacji zmiennych w *application*

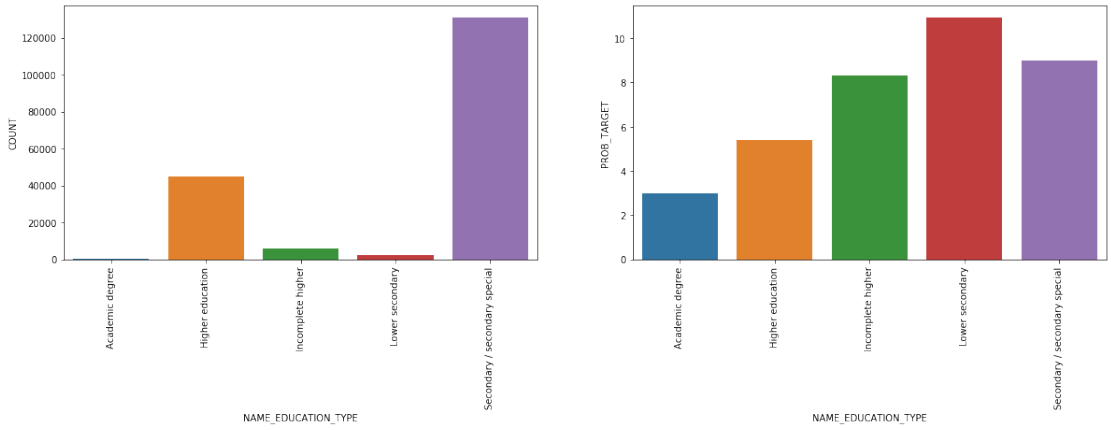
1.5.1 Zależnosci z poszczególnymi zmiennymi kategorycznymi

Dla poszczególnych zmiennych kategorycznych po lewej stronie przedstawiamy liczności wystąpień poszczególnych kategorii, a po prawej liczności tych kategorii dla TARGETU równego 1.



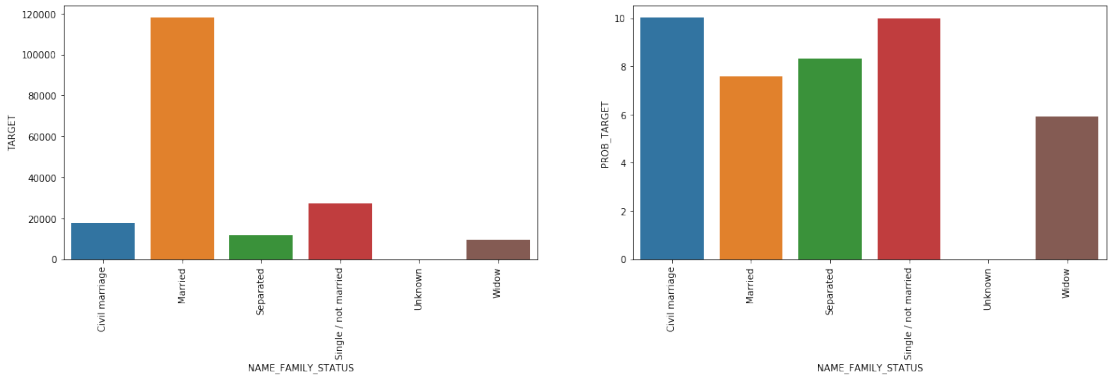
Rysunek 5: CODE_GENDER

Zdecydowanie więcej kobiet składało aplikacje kredytowe, ale więcej mężczyzn nie miało problemów ze spłatą kredytu.



Rysunek 6: NAME_EDUCATION_TYPE

Najwięcej aplikacji kredytowych wpłynęło od osób w wykształceniem średnim lub wyższym.



Rysunek 7: NAME_FAMILY_STATUS

2 Inżynieria cech

- Krok 1 - usunięcie braków danych

  W poprzedniej fazie projektu usunęliśmy już kolumny mające więcej niż 50% wartości NA. W pozostałych kolumnach chcemy uzupełnić braki danych. Dla zmiennych numerycznych zdecydowaliśmy się wstawić w miejsca braków danych medianę wartości poszczególnych kolumn. Dla zmiennych kategorycznych w miejsce wartości NA wstawiamy status "Unknown" - tworzy to odrębną kategorię tej zmiennej.

- Krok 2 - EXT-SOURCE

  W zbiorze danych mieliśmy dwie zmienne oznaczające ocenę klienta wydaną przez zewnętrzne źródła: EXT_SOURCE_2 i EXT_SOURCE_3. Jedna z nich miała dużo brakujących wartości, więc postanowiliśmy uśrednić ich wyniki. Dało nam to nową zmienną, która była lepiej skorelowana ze zmienną TARGET, a także pozbyliśmy się wielu brakujących wartości.

- Krok 3 - ograniczenie się do najważniejszych zmiennych

  Patrząc na korelację poszczególnych kolumn z targetem postanowiliśmy zostawić jedynie kilka najważniejszych zmiennych ze zbioru application_train:

  - TARGET
  - SK_ID_CURR
  - EXT_SOURCE
  - DAYS_BIRTH
  - REGION_RATING_CLIENT_W_CITY
  - FLAG_EMP_PHONE
  - AMT_GOODS_PRICE
  - CNT_CHILDREN
  - FLAG_DOCUMENT_3
  - CODE_GENDER

- Krok 4 - One Hot Encoding

  Do wszystkich zmiennych kategorycznych zastosowaliśmy One Hot Encoding.

- Krok 5 - Przygotowanie i dołączenie danych ze zbioru previous_application

  W zbiorze *installments_payments* dla każdego kredytu mamy informacje o spłacie kolejnych rat. Na podstawie wymaganych oraz rzeczywistych terminów spłaty rat utworzyliśmy nową kolumnę DAYS_DELAY (liczba dni opóźnienia w spłacie). Następnie podsumowaliśmy wartości w tej kolumnie dla każdego kredytu - kolumna IS_DELAYED oznaczająca liczbę opóźnionych spłat kredytu.

Dodatkowo powstała kolumna COUNT_INST oznaczająca łączną liczbę rat kredytu.

Tak utworzone kolumny dołączyliśmy do zbioru *previous_application* (dla odpowiednich SK_ID_ PREV).

W zbiorze *previous_application* dokonaliśmy One Hot Encodingu na wszystkich zmiennych kategorycznych, zwiększając tym samym liczbę kolumn do ok. 170.

Aby zbiór danych można było dołączyć do głównego zbioru *application* zagregowaliśmy dane o tych samych klientach (według SK_ID_CURR) za pomocą funkcji sum(). Korzystając z kolumn IS_DELAYED oraz COUNT_INST utworzyliśmy kolumnę PERCENT_DELAY - procent rat kredytów, które były opóźnione w spłacie.

- Krok 6 - Przygotowanie i dołączenie danych ze zbioru bureau

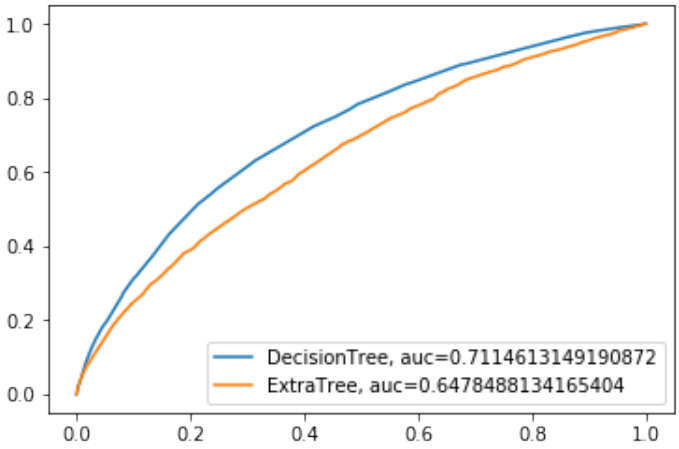
  Przygotowanie zbioru *bureau* sprowadziło się do usunięcia kolumn, w których zawarta była duża liczba brakujących wartości. Wyrzuciliśmy także kolumnę DAYS_CREDIT_UPDATE, ponieważ była ona silnie skorelowana z kolumną DAYS_CREDIT. Taki zbiór dołączyliśmy do naszego bazowego zbioru po kolumnie SK_ID_CURR. Ponieważ niewszyscy klienci, którzy znajdowali się w zbiorze *application* mieli historię kredytową w *bureau*, to powstało w ten sposób wiele brakujących wartości. Skupiliśmy się więc na klientach, których dane były zawarte w *bureau*, tzn. zostawiliśmy w zbiorze danych te osoby, które pojawił się w zbiorze *bureau*.

- Krok 7 - Ostateczne usunięcie braków danych

  Braki danych, które pojawiły się po joinie wszystkich tabel zastąpiliśmy wartością -1.

3 Wybór modeli klasyfikacyjnych

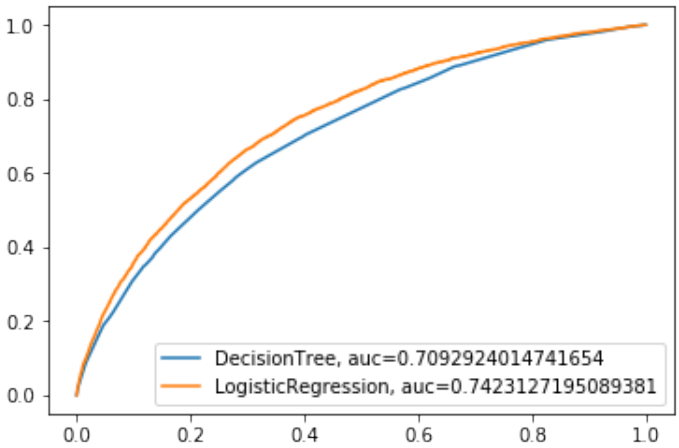
Po zapoznaniu się ze zbiorem danych postanowiliśmy wybrać kilka modeli klasyfikacyjnych do przetestowania. Postanowiliśmy spojrzeć najpierw na struktury drzewiaste. W naszym przypadku na początku wybraliśmy dwie: `DecisionTreeClassifier` oraz `ExtraTreeClassifier`.



Rysunek 8: Wykres AUC DecisionTree i ExtraTree

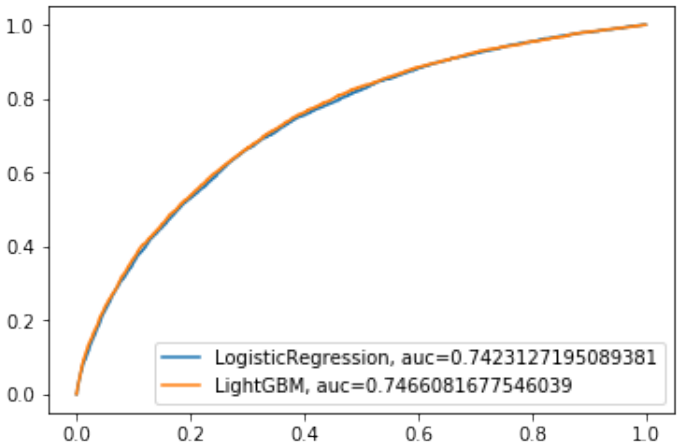
Po wstępnym strojeniu parametrów `DecisionTree` ma dużo lepsze właściwości predykcyjne niż `ExtraTree`, więc Jak narazie `ExtraTree` możemy odrzucić.

Następnie spojrzeliśmy na najprostsze modele. Naszą uwagę zwróciła regresja logistyczna. W tym przypadku na domyślnych parametrach uzyskaliśmy AUC rzędu 0.74, gdzie AUC `DecisionTree` waha się w okolicach 0.71.



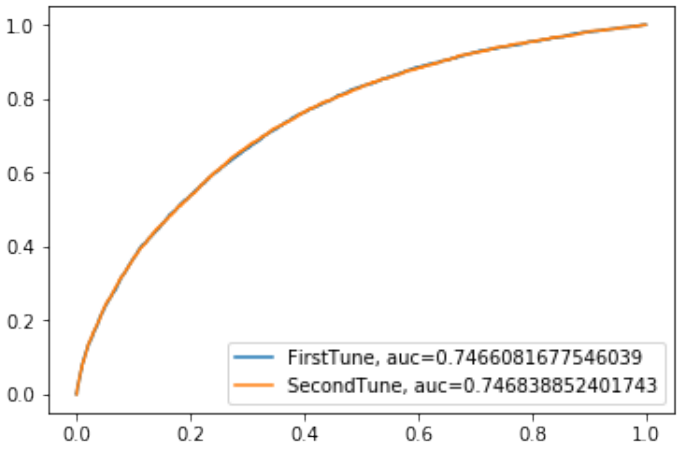
Rysunek 9: Wykres AUC DecisionTree i LogisticRegression

Na koniec przyjrzeliśmy się modelom boostingowym. Wybraliśmy `LightGBM`, ponieważ dawał on najwyższe AUC podczas testów oraz bardzo szybko się liczył. Na tym etapie odrzuciliśmy `DecisionTree`, gdyż w `LightGBM` boosting jest robiony m.in. na drzewach decyzyjnych, więc jeśli drzewa decyzyjne dawałyby najlepsze wyniki to powinno to nam wyjść przy strojeniu parametrów.



Rysunek 10: Wykres AUC LogisticRegression i LightGBM

Ponieważ podstawowy model regresji logistycznej nie ma parametrów, to postanowiliśmy, że dokładne strojenie hiperparametrów przeprowadzimy tylko na `LightGBM`. Rezultaty były gorsze niż się spodziewaliśmy, ponieważ AUC podniosło się jedynie o około 0.0002.



Rysunek 11: Wykres AUC LightGBM po wstępnym strojeniu hiperparametrów oraz po dokładnym strojeniu

4 Podsumowanie i wnioski

Ocena zdolności kredytowej klientów banku to bardzo złożony problem. Mamy tutaj doczynienia z wieloma, często enigmatycznymi, zmiennymi. Pojawia się także wiele brakujących wartości, które trudno w jakiś sposób zastąpić. Staraliśmy się w tym projekcie usuwać jak najmniej informacji. Dzięki takiemu podejściu na końcu uzyskaliśmy prawie 80 zmiennych. W tej sytuacji najlepiej zachowały się struktury drzewiaste oraz regresja liniowa. O ile te pierwsze nie są wcale zaskoczeniem co do swojej skuteczności, o tyle prosty model liniowy uzyskujący wyniki zbliżone do boostingu potrafi być zaskoczeniem dla początkujących analityków danych. LightGBM, który osiągał tutaj najlepsze wyniki nie jest bazowym modelem w bibliotece scikit-learn, a to uczy, że warto spojrzeć na algorytmy, które nie są dołączone w podstawowych wersjach bibliotek. Udało nam się uzyskać AUC w okolicach 0.75 co jest zbliżone, a jednocześnie dalekie od osiąganych przez specjalistów w tej dziedzinie 0.81. Na pewno prześledzimy jeszcze wiele razy cały nasz proces tworzenia tego projektu, żeby odnaleźć momenty, w których mogliśmy poprawić nasze AUC, ale i tak jesteśmy zadowoleni oraz dumni z tego co udało nam się uzyskać.