Przykłady dzialania biblioteki SAS SWAT dla Python

*Opracowano przez Mykyta Yehorov s19313*

**Połączenie**

Aby umożliwić programowi Python współpracę z SAS Cloud Analytic Services, musimy nawiązać połączenie z serwerem.

Robimy to w trzy kroki

1. Połączenie binarne

import swat

s = swat.CAS("*cloud.example.com*", *5570*)

SWAT to nazwa pakietu używanego do łączenia się z serwerem. Pakiet zawiera klasy i metody, które są udokumentowane w innych częściach tego dokumentu oraz w dokumentacji pakietu.

2. Połączenie z serwerem HTTP

s = swat.CAS("https://webserver.example.com/cas-shared-default-http/")

3. Połączenie bezpośrednio z CAS

s = swat.CAS("http://cloud.example.com:8777", protocol='http')

**Load CAS Action Sets**

Procedury analityczne w SAS Viya są znane jako działania CAS i są zorganizowane w zestawy działań CAS. Te zestawy akcji CAS reprezentują modułową funkcjonalność (np. Przetwarzanie danych, modele budynków) i muszą zostać załadowane po połączeniu się z SAS Viya.

s.loadactionset(action\_set\_name)

Pryzkladz setów : aggregation, cardinality, dataStep, mixed itd.

**Python**

**pandas, matplotlib** - są to pakiety Pythona do manipulacji danymi i tworzenia wykresów.

import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline

**Actions Sets**

Każda akcja CAS wymaga obiektu połączenia (w tym przypadku `` s '') z przodu, a następnie następuje według określonej kolejności - zestaw akcji, akcja, parametr, opcja:

connectionObject.actionSetName.actionSet(

parameter1={

option1=True,

option2=True,

},

parameter2=5

)

Na przykład czynnością trenującą model drzewa decyzyjnego jest:

s.decisionTree.dtreeTrain(

table = dict(name= input\_data),

target = target,

inputs = input\_variables,

nominals = nominal\_variables,

casOut = dict(name='DecisionTreeModelOutput', replace=True),

code = dict(casout=dict(name='DecisionTreeModelCode', replace=True)

)

**Dostęp do danych w SAS Viya**

Wszystkie dane w SAS Viya znajdują się w pamięci, aby wydajniej przeprowadzać analizy. Oznacza to, że po połączeniu musisz załadować zestaw danych, którego chcesz użyć, do pamięci SAS Viya. Gdy dane znajdują się w pamięci w SAS Viya, są przedstawiane jako tabela CAS.

table\_name = "looking\_glass\_v4"

castbl = s.load\_path(

table\_name+'.sas7bdat',

casOut=dict(name=table\_name, replace=True)

)

castbl.head()

Modeli

1. **Algorytm drzewa decyzyjnego dła pociąga**

table = dict(name= table\_name, where='\_partind\_ = 1'),

target = target,

inputs = total\_inputs,

nominals = target, *# Encode target as a category, since this is a binary classifcation problem,*

casOut = dict(name='DT\_model', replace=True),

code = dict(casout=dict(name='DT\_model\_code', replace=True)),

encodeName=True

)

s.CASTable('DT\_model').head()

Po wytrenowaniu modelu drzewa decyzyjnego możemy teraz ocenić go w naszym zbiorze walidacyjnym, aby to zrobić, odwołujemy się do tabeli CAS zawierającej nasz model „DT\_model”, który utworzyliśmy w akcji uczącej:

score\_model = s.decisionTree.dtreeScore(

encodeName = True,

table = dict(name = table\_name, where='\_partind\_=0'),

modelTable = 'DT\_model',

copyVars = [target, '\_partind\_'],

casOut = dict(name = '\_scored\_DT', replace = True)

)

score\_model

**2.Oceńianie na zestawie testowym**

Teraz, gdy model został wytrenowany, możemy ocenić go na zbiorze testowym i utworzyć nasze zgłoszenie. Pierwszym krokiem jest ocena naszego modelu na zestawie danych do oceny - który znajduje się w ACADCOMP Caslib.

Jak podano w sekcji Dane konkursu Kaggle https://www.kaggle.com/c/sasviyaforlearners2020/data, tworzenie pliku CSV zgłoszenia składa się z dwóch części:

1.Generowanie kolumny prognoz

2.Dołączanie kolumny prognozy do pliku CSV zawierającego identyfikatory klientów zestawu testowego (submissionsId.csv)

Dzieje się tak, ponieważ Kaggle wymaga, aby wszystkie zgłoszenia zawierające prognozy były sortowane według określonej kolejności identyfikatora klienta.

W pozostałej części tego notesu omówiono sposób generowania kolumny prognozy. Aby rozpocząć proces oceniania w celu wygenerowania kolumny predykcji, ładujemy zestaw testów do pamięci, a następnie oceniamy model. Będziemy musieli imputować nasze dane testowe, ponieważ model drzewa decyzyjnego oczekuje przypisanej kolumny i użyjemy akcji dtScore, aby następnie ocenić model

castbls = s.load\_path(

'KAGGLETEST\_LOOKING\_GLASS\_1\_V3.sas7bdat',

caslib='ACADCOMP',

casOut=dict(name='testset', replace=True)

)

s.dataPreprocess.impute(

table = castbls,

outVarsNamePrefix = 'IMP',

methodContinuous = 'MEDIAN',

inputs = 'bill\_data\_usg\_m02',

copyAllVars = True,

casOut = dict(caslib= 'DLUS34', name='testset', replace=True)

)

eval\_model = s.decisionTree.dtreeScore(

encodeName = True,

table = 'testset',

modelTable = 'DT\_model',

copyVars = ['Customer\_ID'],

casOut = dict(name = '\_eval\_DT', replace = True)

)

eval\_model

Output i print jako CAS Table

keepcolumns = ['I\_upsell\_xsell']

evaluation = s.CASTable('\_eval\_DT').loc[:,keepcolumns]

evaluation.head()

Poprzedne dzialanie daje możliwość Output jako CSV

swat.options.cas.dataset.max\_rows\_fetched=60000

evaluation.to\_csv('predictionColumn\_Python.csv', index=False, float\_format='**%.12g**')

Pliki kodu wyniku modelu wyjściowego

Oprócz przesłania pliku CSV, aby móc odtworzyć wyniki, wymagane jest przesłanie plików wyników modelu.

Wyszkolone modele SAS Viya są reprezentowane przez:

Kod wyniku SAS (modele statystyczne - np. Regresja liniowa, regresja logistyczna, drzewo decyzyjne)

Kod wyniku SAS i ASTORE (modele ML - np. Losowy las, wzmocnienie gradientu, sieć neuronowa)

Kod oceny SAS to pliki tekstowe składające się z reguł IF-ELSE-THEN napisanych w języku programowania SAS (niezależnie od tego, jaki inny język programowania został użyty do trenowania modelu). ASTORE to obiekty binarne zawierające skompresowaną logikę modelu skomplikowanych modeli ML, aby umożliwić większą przenośność tych plików. W tym prostym przykładzie wytrenowaliśmy model drzewa decyzyjnego, abyśmy mogli wyprowadzić kod oceny z tabeli CAS, w której model był przechowywany.

pd.set\_option("display.max\_colwidth", -1)

file=open('DT\_score.sas', 'w')

file.write(s.CASTable('DT\_model\_code')['DataStepSrc'].str.strip().to\_string(index=False).replace("**\\**n",""))

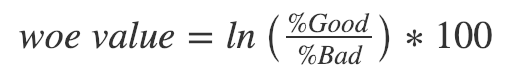
file.close()

**3.Tworzenie kart kredytowych za**

Niezależnie od tego, czy ubiegasz się o pierwszą kartę kredytową, czy kupujesz drugi dom - czy gdzieś pomiędzy - prawdopodobnie napotkasz proces aplikacyjny. W ramach tego procesu banki i inni pożyczkodawcy używają karty wyników, aby określić prawdopodobieństwo spłaty pożyczki.

Oczywiście oznacza to, że scoring kredytowy jest ważnym tematem nauki o danych dla banków i każdej firmy, która prowadzi interesy z sektorem bankowym.

Najpierw przekształcam dane za pomocą metody Weighted Evidence (WOE). Ta metoda próbuje znaleźć monotoniczną zależność między cechami wejściowymi a zmienną docelową, dzieląc każdą cechę na komórki i przypisując wagę do każdej komórki.



Zastosujmy tę transformację do całego naszego zbioru danych. Użycie akcji wstępnego przetwarzania danych ustawionej w SAS upraszcza tworzenie potoków danych .

Aby zbudować pojedynczy potok, wystarczy kilka kroków:

1.Przypisz zmienne do ról.

2.Skonstruuj zmienne transformacyjne.

3.Dodaj transformacje, aby zastosować je później



Następnie tworzymy pierwszą transformację o nazwie req\_pack1, która jest skrótem od partii żądania i jest parametrem znajdującym się w akcji datapreprocess.transform. Nadaję tej transformacji nazwę, przekazuję listę funkcji i celów oraz określam interesujące zdarzenie, które w tym przypadku jest „złe”.

Druga konwersja, którą oznaczam jako req\_pack2, jest prawie identyczna, z tym wyjątkiem, że konwertuję na nominalne wejścia i dlatego zamiast próbkowania muszę używać cattrans. Parametr cattrans oznacza transformację kategorialną.

Następnie dodajemy te listy razem, abyśmy mogli później przekazać schemat transformacji do naszej akcji transformacji.

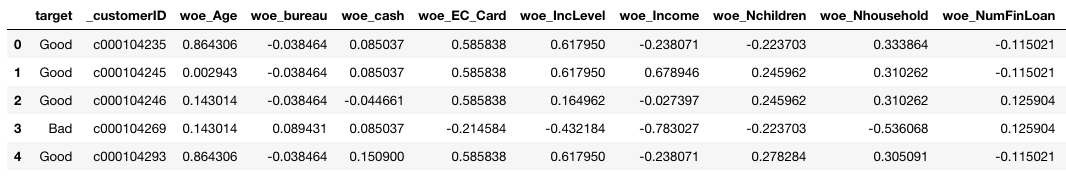


Podaję tabelę wyjściową (casout), która będzie się nazywać woe\_transform. Następnie używam copyVars do kopiowania funkcji target i \_customerID do naszej nowej przekonwertowanej tabeli z oryginalnej tabeli. Następnie daję globalny prefiks „woe” wszystkim naszym nowo przekonwertowanym funkcjom.

Na koniec przyjrzymy się podglądowi naszego nowego stołu WOE



Zaleca się, aby spojrzeć na IV, aby nasze funkcje zrozumiały ich moc predykcyjną i zdecydowały, czy musimy uwzględnić te funkcje w naszym modelu. Poniżej znajduje się wyliczenie dla IV. Rysunek 4 przedstawia wykres tych wartości IV dla każdej z naszych funkcji. Silne statystyki zwykle mają IV> 0,3, słabe <0,02, a wszystko> 0,5 może być podejrzane i wymaga dokładnego przyjrzenia się. Obliczanie wartości informacyjnej (IV) pokazuje, że nasze funkcje są prawie podzielone na pół między silnymi, a następnie gdzieś pomiędzy.



*Obliczanie wartości informacyjnej (IV)*

Następnym krokiem jest dopasowanie modelu regresji logistycznej przy użyciu naszego nowo przekształconego zbioru danych WOE.

1.Przypisz dane wejściowe do modelu i połącz „woe\_” ze wszystkimi oryginalnymi nazwami kolumn, tak aby poprawnie odwoływał się do zestawu danych woe\_transform.

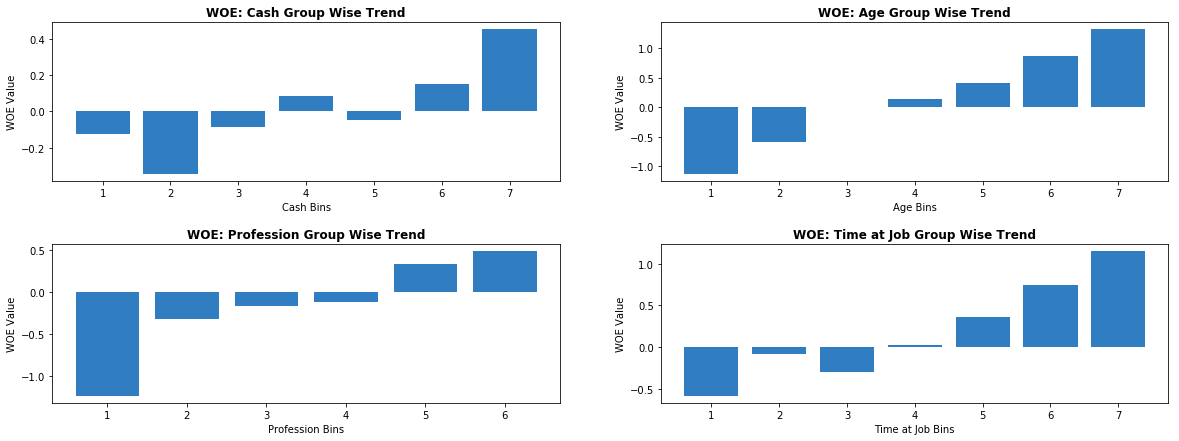
2.Odwoływanie się do zestawu danych woe\_transform przy użyciu parametru tabeli

3.Podaj zarówno cel, jak i łącze (łącze „dobre” lub „złe” w odniesieniu do modelu).

4.Następnie określ metodę bezpośredniego wyboru do wybierania zmiennych. Wybór bezpośredni rozpoczyna się od pustego modelu i dodaje jedną zmienną w każdej iteracji w oparciu o określone kryterium (AIC, AICC, SBC itp.).

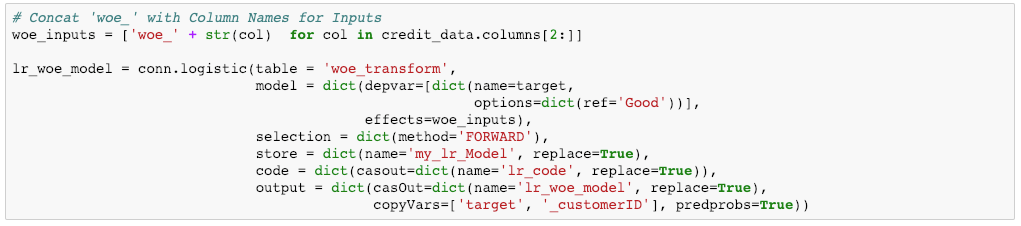
5.Użyj parametru code, aby zapisać nasz kod regresji logistycznej

6.Utwórz nazwę nowej tabeli wyjściowej za pomocą funkcji casout, a następnie ponownie skopiuj zmienne target i \_customerID.

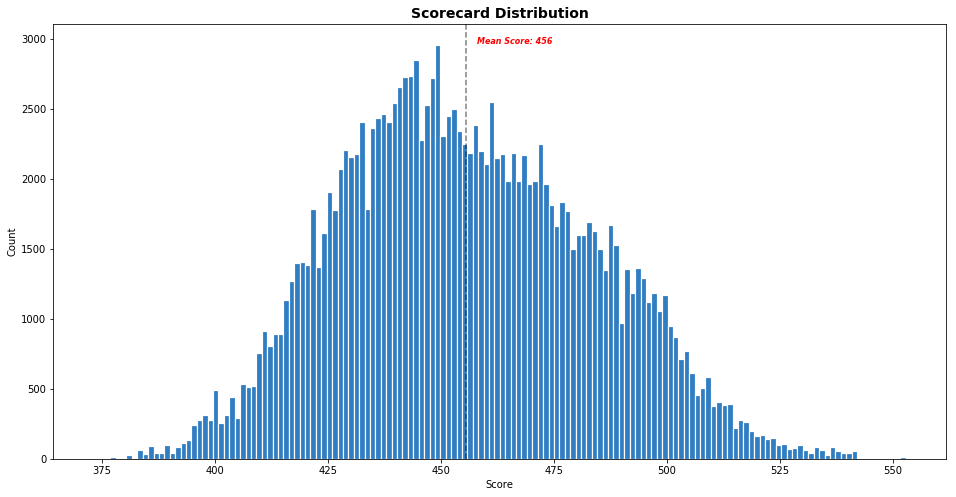


Ostatnim krokiem jest przeskalowanie modelu do karty wyników. Użyjemy ogólnej metody skalowania. Będziemy potrzebować zarówno naszych współczynników regresji logistycznej, które uzyskaliśmy podczas dopasowywania naszego modelu, jak i naszego zbioru danych WOE z przekształconymi wartościami WOE.

Pierwsza wartość to wynik docelowy. Można to uznać za wynik wyjściowy. Zwiększyliśmy wynik dla tej karty wyników do 600. Docelowy wynik równy 600 odpowiada dobrej / złej szansie na osiągnięcie celu z 30 do 1 (target\_odds = 30)

Następna zmienna nazywa się pts\_double\_odds, co oznacza podwojenie kursów. Oznacza to, że wzrost o 20 punktów podwaja nasze szanse na to, że wnioskodawca jest dobry pod względem niewykonania zobowiązań. Na przykład, jeśli zdobyłeś 600 punktów, masz 30 do 1 prawdopodobieństwa, że ​​będziesz dobrym kandydatem na pożyczkę. Ale wynik 620 daje szansę na bycie dobrym 60: 1.





*Wykres oszacowań według przewidywanego prawdopodobieństwa*

Żródła :

1. <https://www.kaggle.com/yjching/sas-viya-python-starter>
2. <https://go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.5&docsetId=allprodsactions&docsetTarget=actionSetsByName.htm&locale=en>
3. <https://go.documentation.sas.com/?cdcId=pgmcdc&cdcVersion=8.11&docsetId=caspg3&docsetTarget=titlepage.htm&locale=en>
4. <https://www.machinelearningmastery.ru/building-credit-scorecards-using-sas-and-python-cf9c3ecc30bc/>