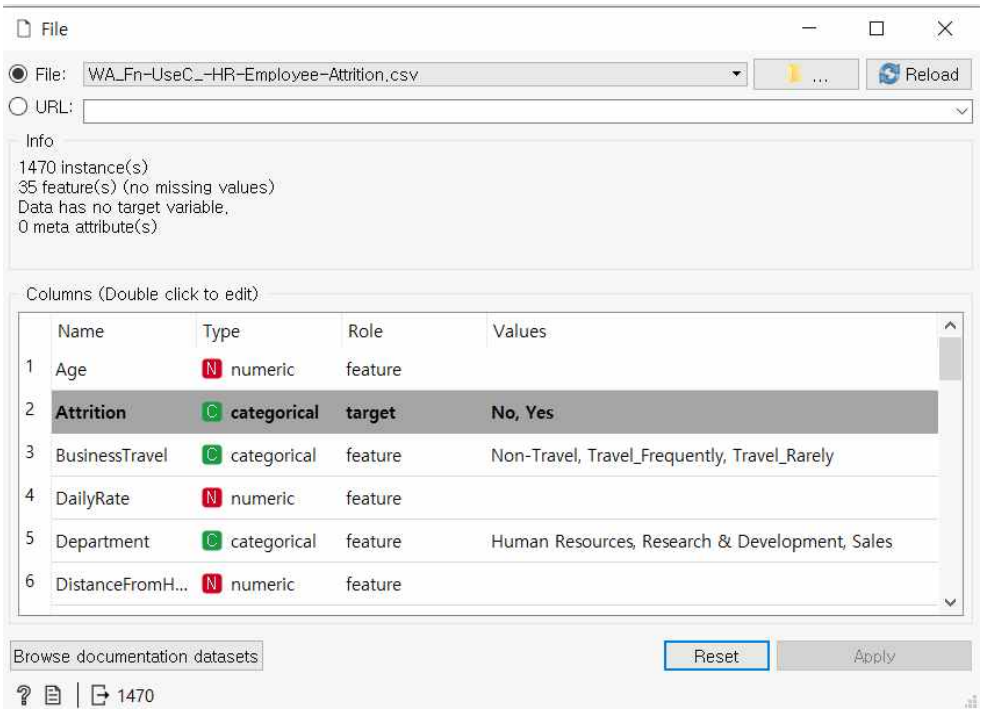
데이터 마이닝 기말 과제

빅데이터전공_2019380610_황현수

1. 주제 : HR(Human Resource) 데이터를 이용해 이직 가능성을 예측하는 분류모델 만들기

2. 데이터 셋 확인



1) 1470개의 행과 35개의 변수가 있다. 결측치는 없다.

3. 데이터 셋에대한 설명(데이터 출처, 데이터의 특성)

1) 데이터 출처 : Kaggle

[WA_Fn-UseC_-HR-Employee-Attrition.csv] : https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

2) 데이터 특성

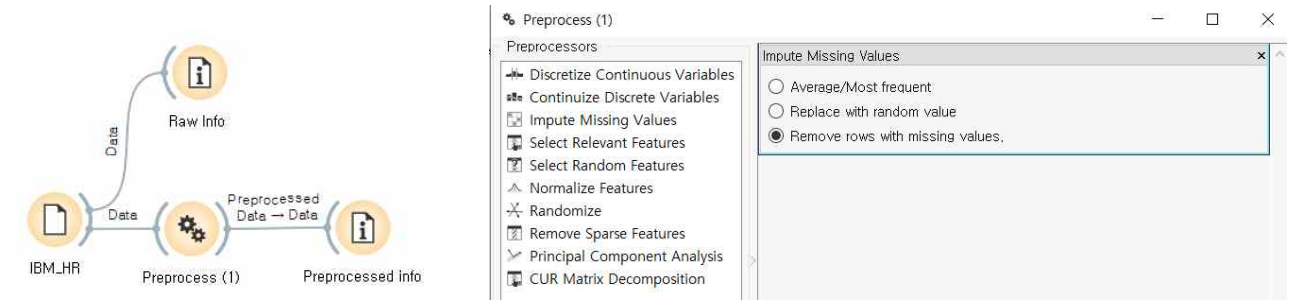
IBM에서 제공하는 가상의 HR 데이터 세트이다. 여기에는 1470명의 직원 항목(row)과 35개의 변수(나이, 급여 수준, 성별 등)가 포함되어 있다. 그중 하나는 목표 변수인 Attrition(이직 여부 -> Yes / No) 변수이다.

변수명	변수 설명(Type)	변수명	변수 설명(Type)
1. Age	나이(N)	19. MonthlyIncome	월 소득(N)
2. Attrition	이직 여부(C)	20. MonthlyRate	월급 수준(N)
3. Business Travel	출장 빈도(C)	21. NumCompaniesWorked	일한 회사의 수(N)
4. DailyRate	일일 급여 수준(N)	22. Over18	18세 이상(C)
5. Department	소속 부서(C)	23. OverTime	규정 외 노동시간(C)
6. DistanceFromHome	출퇴근거리(N)	24. PercentSalaryHike	임금 상승률(N)
7. Education	교육 수준(N)	25. PerformanceRating	업무 평가 수준(N)
8. EducationField	전공(C)	26. RelationshipSatisfaction	대인관계 만족도(N)
9. EmployeeCount	직원숫자	27. StandardHours	표준 시간(N)
10. EmployeeNumber	직원ID(N)	28. StockOptionLevel	스톡옵션 수준(N)
11. EnvironmentSatisfaction	업무 만족도(N)	29. TotalWorkingYears	경력 기간(N)
12. Gender	성별(C)	30. TrainingTimesLastYear	교육시간(N)
13. HourlyRate	시급 수준(N)	31. WorkLifeBalance	일과 생활의 균형(N)
14. JobInvolvement	업무 참여도(N)	32. YearsAtCompany	근속 연수(N)
15. JobLevel	업무 수준(N)	33. YearsInCurrentRole	현재 업무 근로 기간(N)
16. JobRole	업무 유형(C)	34. YearsSinceLastPromotion	마지막 프로모션 지급 연도(N)
17. JobSatisfaction	업무 만족도(N)	35. YearsWithCurrManager	현재 관리자와 협업 기간(N)
18. MaritalStatus	결혼 여부(C)		

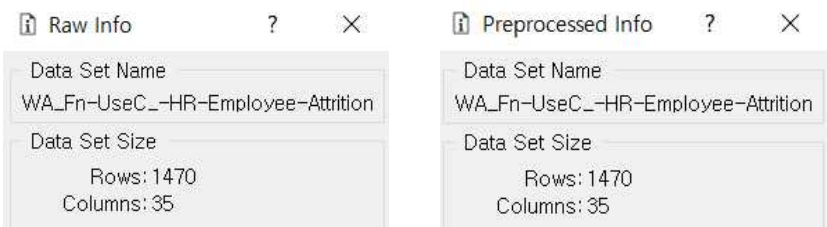
Features -> Categorical : 9, Numeric : 25

4. 분석을 위한 데이터 가공(결측치 점검 및 처리, 불필요 변수 제거, 질적 변수의 수량화, 오버 샘플링)

1) 결측치 점검 및 처리

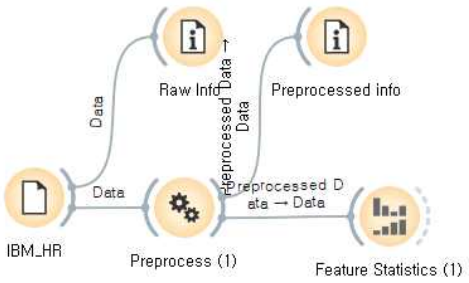


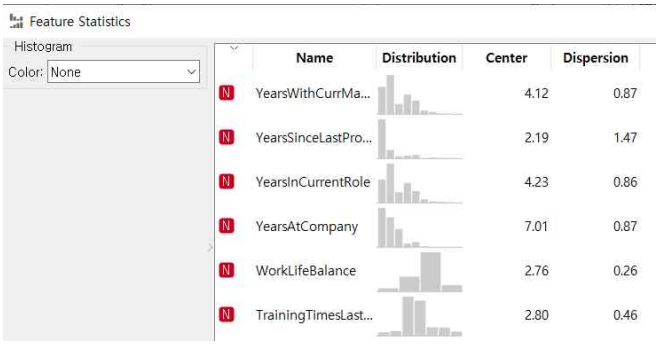
Preprocess 위젯의 Remove rows with missing values를 활용하여 결측치 행을 제거했다.



결측치 행을 제거하고 난 후의 Preprocessed data의 행이 기존 데이터의 행 개수와 동일하게 1470개인 것을 확인할 수 있다. 결측치가 없다고 볼 수 있다.

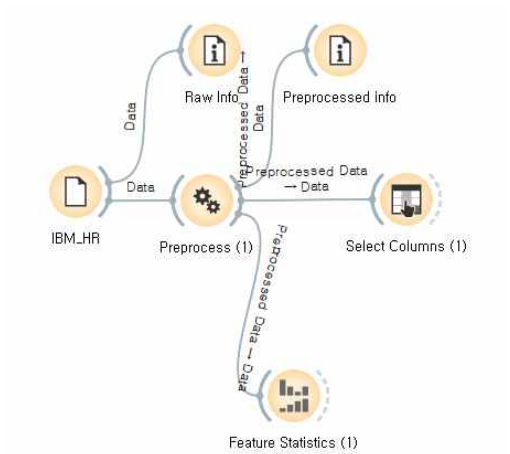
2) 불필요 변수 제거

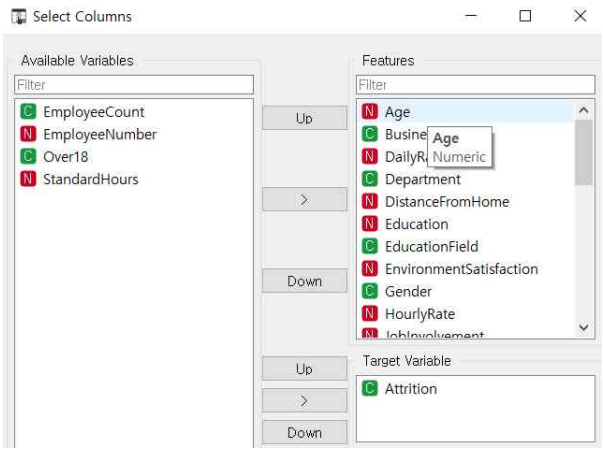




Feature Statistics 위젯을 통해 변수의 분포를 확인했다.
모든 직원이 같은 값을 가진 변수나 직원들의 고유번호 같은 분석에 의미 없는 변수를 발견할 수 있었다.

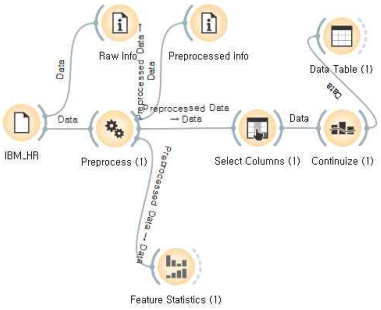
EmployeeCount(직원 숫자) -> 같은 회사이기에 모두 같음, Over18(18세 이상) -> 직원들 모두 성인이기에 모두 Y.
StandardHours(표준 시간) -> 평균 노동시간으로 모두 동일, EmployNumber(직원ID) -> 1470개 직원 고유의 값.

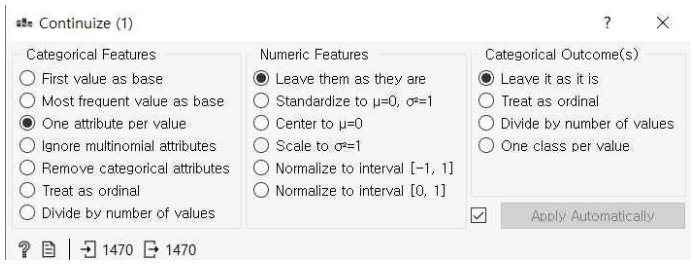




Select Columns 위젯을 통해 불필요한 변수를 제외하고 변수를 선택했다.

3) 질적 변수의 수량화(One-Hot Ecoding)





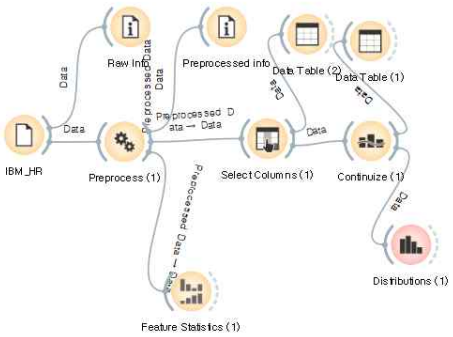
Continuize 위젯을 통해 범주형 텍스트 데이터를 숫자 데이터로 변환했다(컴퓨터가 이해할 수 있는 데이터로 변경).

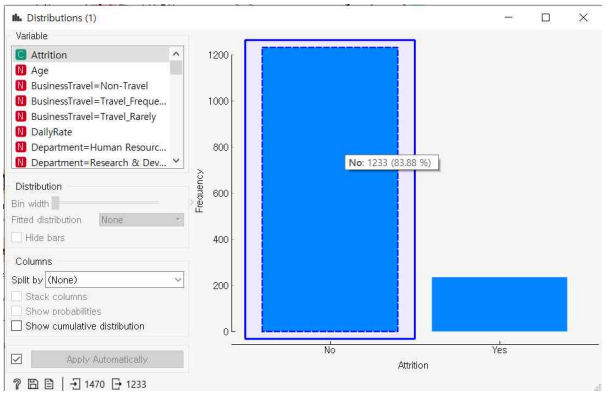
	Attrition	Age	BusinessTravel
1	Yes	41	Travel_Rarely
2	No	49	Travel_Frequently
3	Yes	37	Travel_Rarely
4	No	33	Travel_Frequently
5	No	27	Travel_Rarely
6	No	32	Travel_Frequently
7	No	59	Travel_Rarely
8	No	30	Travel_Rarely
9	No	38	Travel_Frequently
10	No	36	Travel_Rarely
11	No	35	Travel_Rarely

⇨

	Attrition	Age	BusinessTravel=Non-Travel	BusinessTravel=Travel_Frequently	BusinessTravel=Travel_Rarely
1	Yes	41	0	0	1
2	No	49	0	1	0
3	Yes	37	0	0	1
4	No	33	0	1	0
5	No	27	0	0	1
6	No	32	0	1	0
7	No	59	0	0	1
8	No	30	0	0	1
9	No	38	0	1	0
10	No	36	0	0	1
11	No	35	0	0	1

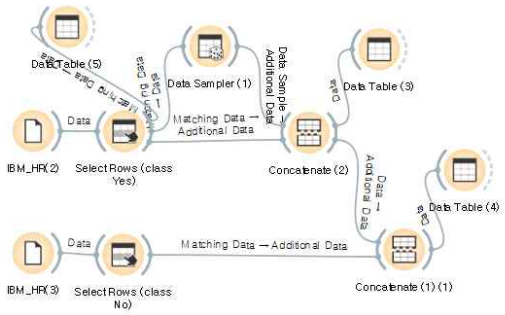
4) 오버 샘플링(Over Sampling)

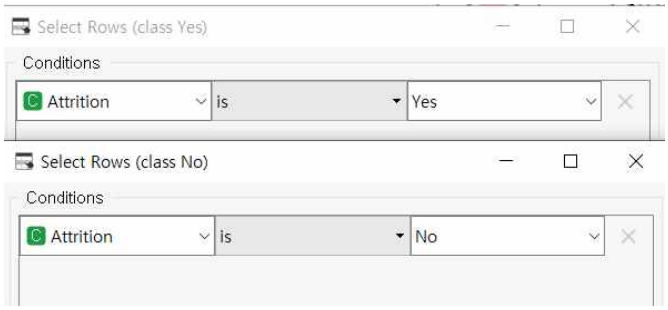




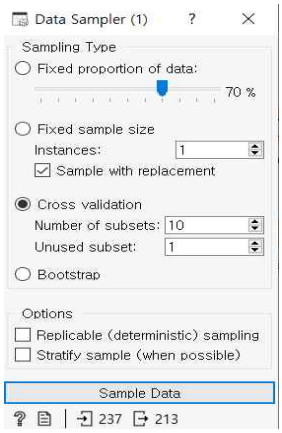
Distributions 위젯을 통해 목표 변수의 분포를 확인했다.
클래스 **Yes / No** 중 No의 비중이 83.88%로 상당히 불균형 데이터인 것을 확인할 수 있었다.

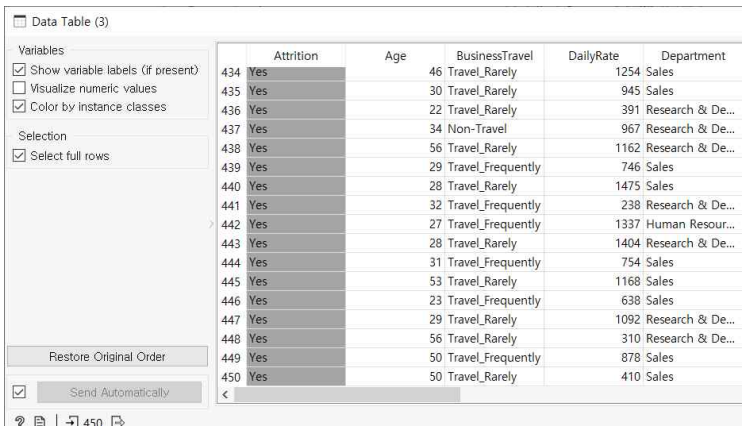
모델을 학습시킴에 있어 클래스 간 샘플 수의 차이가 너무 크게 되면 분류기는 더 많은 샘플이 존재하는 클래스로 편향(bias)된다. 이 경우 전체적인 정확도는 높게 나올지라도 샘플 수가 적은 클래스에 대한 재현율(recall)이 작아지게 된다(즉, 특정 클래스에 오버피팅 된다.). 이러한 문제를 데이터 불균형 문제 혹은 비대칭 문제라고 부른다.
불균형 문제를 해결하는 방법은 대표적으로 언더 샘플링과 오버 샘플링이 있는데 정보 손실이 없고 언더 샘플링보다 보통 분류 정확도가 높은 오버 샘플링을 선택해 클래스 간 데이터 불균형을 해결하려고 했다.
이후 모델 성능 평가 단계에서 오버 샘플링을 하기 전후 모델의 성능을 비교하고 더 좋은 정확도와 재현율을 가지는 모델을 최종 모델로 선택했다.



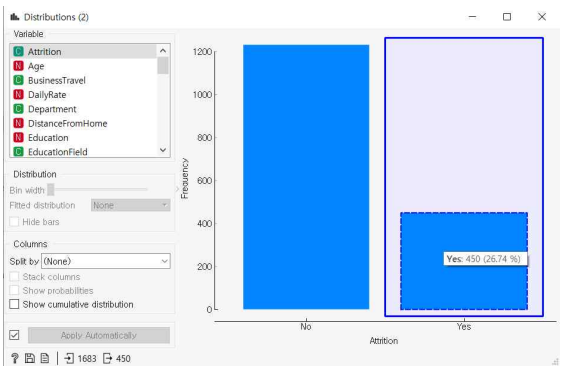


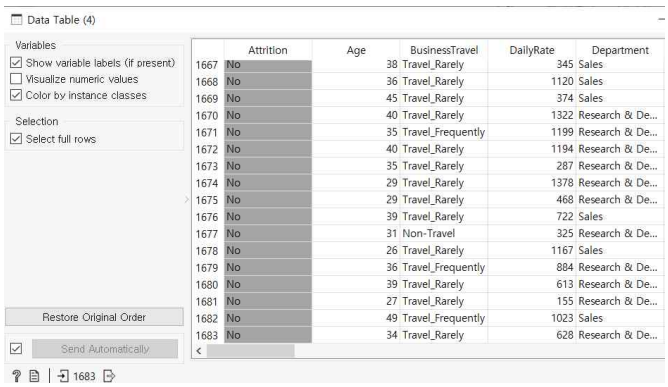
먼저 Select Rows 위젯을 통해 Yes / No 클래스를 분리했다.





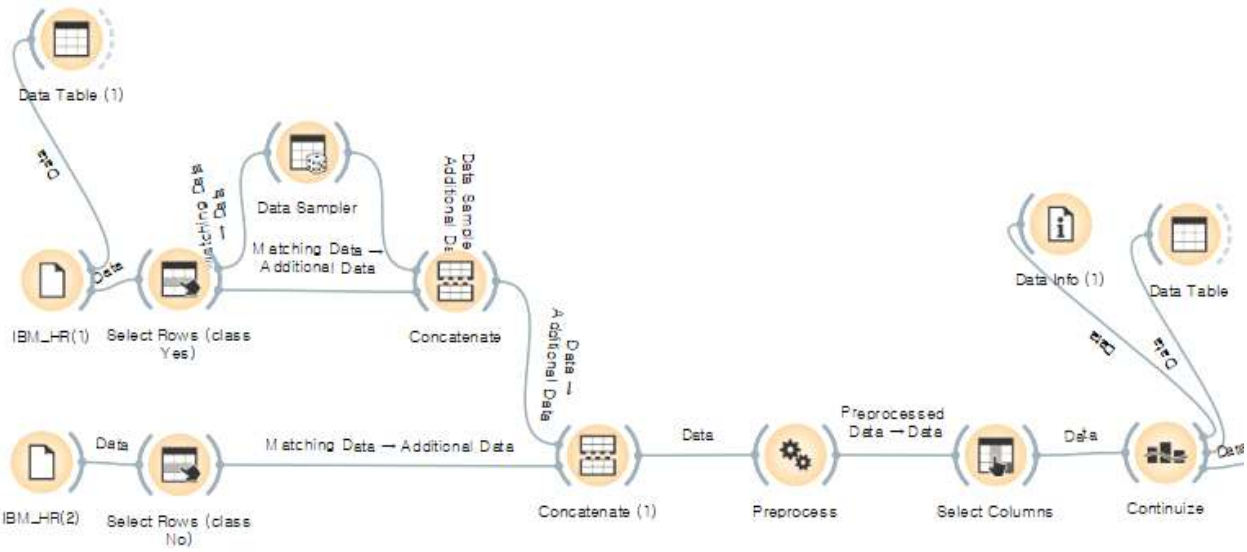
그런 다음 Data Sampler 위젯을 통해 소수 클래스인 Yes 클래스의 데이터를 Cross validation으로 랜덤 샘플링 했고 Concatenate 위젯을 통해 기존의 Yes 클래스 데이터와 합쳤다. 최종적으로 Yes 클래스 데이터를 450개로 증가시켰다.
237개(기존 데이터) + 213개(랜덤 샘플링 데이터) = 450개

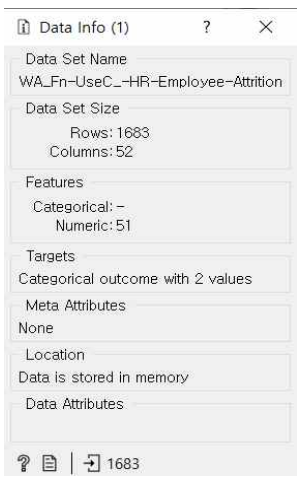


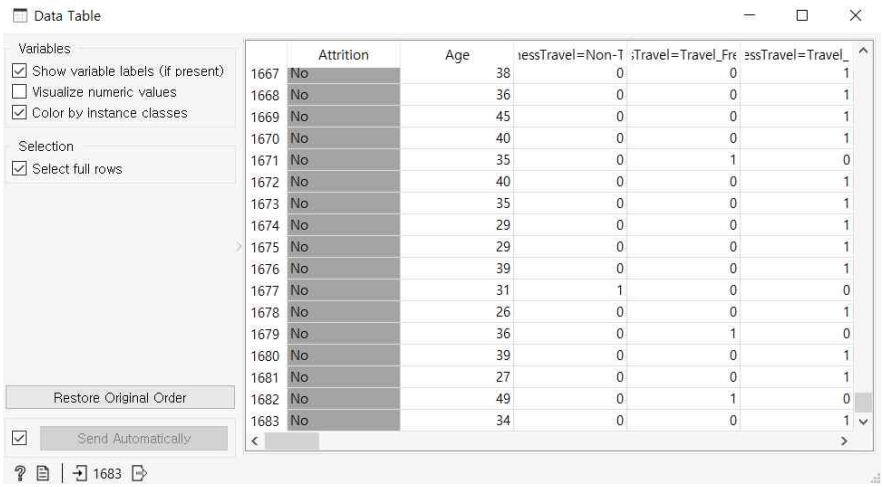


마지막으로 분리해 뒀던 No 클래스의 데이터를 Concatenate 위젯을 통해 증가시킨 Yes 클래스와 다시 한번 합쳤다. 데이터 불균형이 완벽히 해결되지는 않았지만, 성능 향상에 도움이 될 것으로 예상된다.
1233개(기존 No 클래스 데이터) + 450개(증가시킨 Yes 클래스 데이터) = 1683개

5. 완성된 데이터 셋에 대한 설명



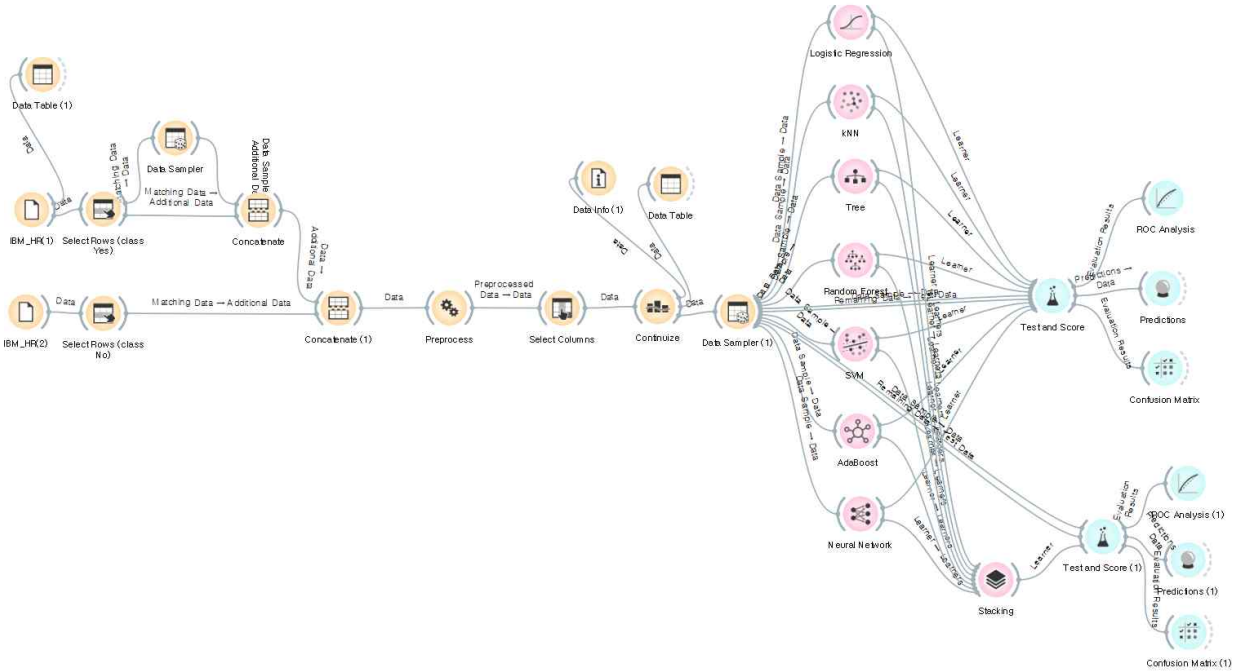




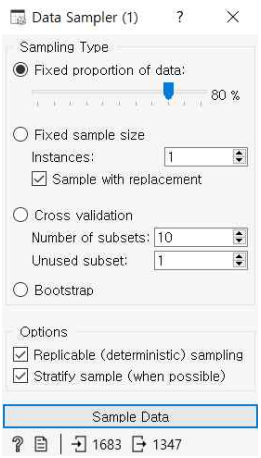
Rows는 데이터 불균형을 해결하기 위해 오버 샘플링 했고 결측치 행을 제거했다. 그 과정에서 총 1683개가 되었다.
(사용 위젯 : 오버 샘플링 -> Select Rows, Data Sampler, Concatenate, 결측치 행 제거 -> Preprocess)

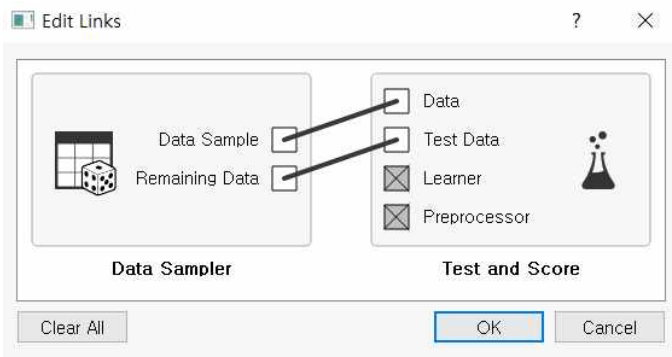
Columns는 불필요한 변수를 제거했고, 범주형 변수를 수량화시켰다. 그 과정에서 총 52개가 되었다.
(사용 위젯 : 불필요한 변수 제거 -> Select Columns, 범주형 변수의 수량화 -> Continuize)

6. 다양한 모형의 적합 및 모형별 성능의 확인



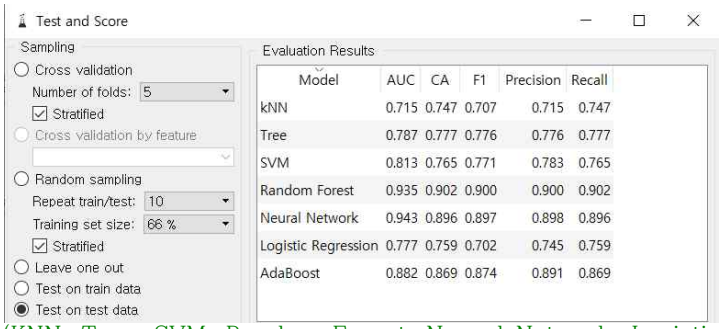
1) Train-Test 데이터 나누기

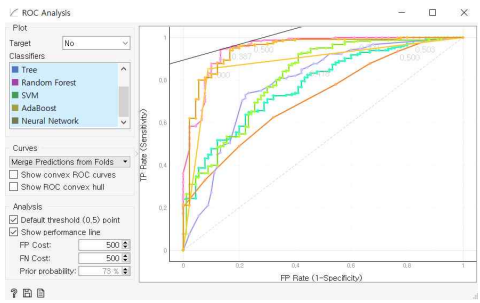




먼저 Data sampler 위젯과 Test and Score 위젯을 연결해서 Train-Test 데이터를 나눴다.
Data sampler 위젯에서 데이터를 80% 샘플링하고 Test and Score와 연결할 때 두 줄을 연결한다. Edit Links에서 80%의 샘플링 데이터(Data Sample)를 Train 데이터로 남은 20%의 데이터(Remaining Data)를 Test Data로 연결해 주면 Train-Test 데이터 나누기가 완료된다.

2) 모형 적합 및 모형별 성능의 확인

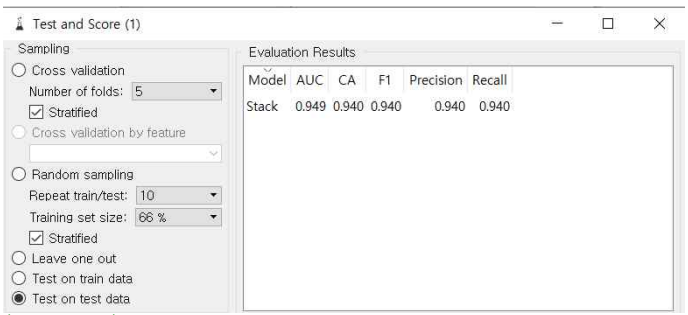


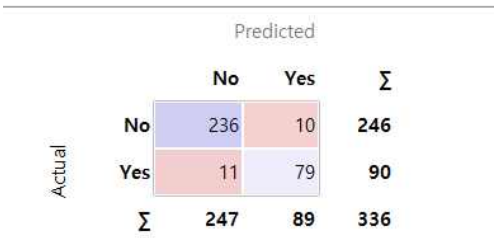


(KNN, Tree, SVM, Random Forest, Neural Network, Logistic Regression, AdaBoost)

모델 위젯들을 데이터 위젯과 Test and Score 위젯을 연결하면 모델을 적합하고 성능을 확인할 수 있다.
이때 Test and Score에서 Sampling 옵션을 Test on test data를 선택해야 미리 나눈 Train 데이터로 모델 학습을 하고 Test 데이터로 성능 평가를 할 수 있다.

분류모델의 성능은 AUC와 F1값이 **높을수록 좋은 성능**이라고 할 수 있다.
성능 평가 결과 **Random Forest**와 **Neural Network** 모델이 비등하게 가장 좋은 성능을 보였다. 또한,
ROC 커브는 왼쪽 위에 그래프가 붙어 있을수록 좋은 모델이라고 평가하는데 마찬가지로 두 모델이 가장 좋았다.





(Stacking)

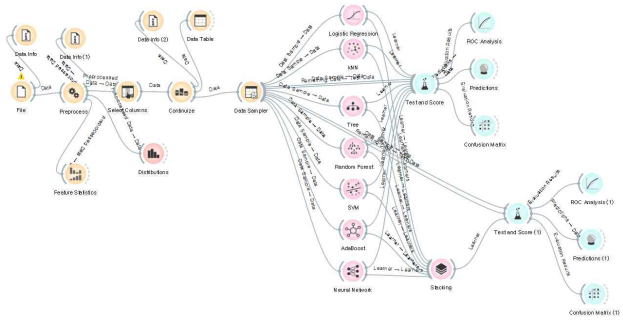
스태킹은 여러 모델을 활용해 각각의 예측 결과를 도출한 뒤 그 예측 결과를 결합해 최종 예측 결과를 만들어내는 모델링 방법이다.
기존의 8개 모델을 Stacking 위젯과 연결하고 Test and Score를 진행하면 스태킹 모델을 사용할 수 있다.

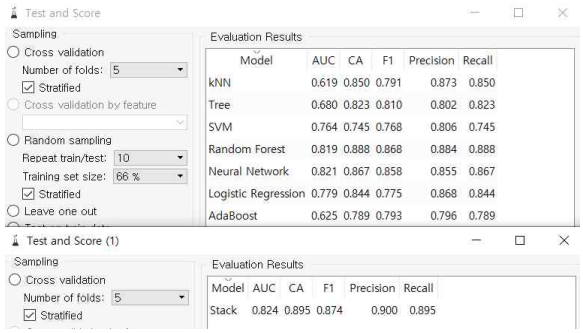
성능은 AUC = 0.949, F1 = 0.940으로 기존 개별 모델들보다 좋은 성능을 보였다.
또한, Confusion Matrix 확인결과 분류를 잘하고 있는 것으로 확인했다.

7. 최종 모형의 선택 및 이유

성능 평가 결과 스태킹(Stacking) 모델의 성능이 가장 좋았다. 마지막으로 오버 샘플링 하기 전 기본 데이터로 학습시킨 모델과 오버 샘플링 처리 후 학습시킨 모델의 성능을 비교해 보고자 한다.

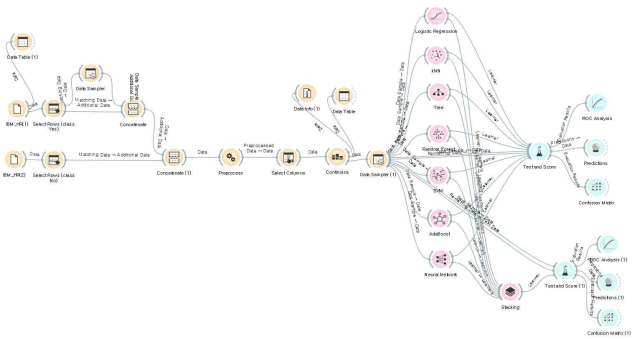
1) 기본 데이터로 학습시킨 모델(오버 샘플링 전)

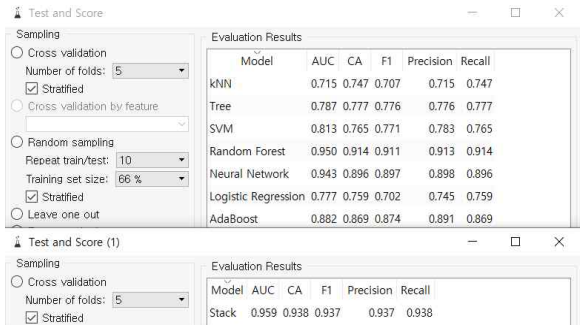




성능 평가 결과 스태킹(Stacking) 모델이 AUC = 0.842, F1 = 0.874로 가장 좋은 성능을 보였다.

2) 오버 샘플링 처리 후 학습시킨 모델





성능 평가 결과 스태킹(Stacking) 모델이 AUC = 0.959, F1 = 0.937로 가장 좋은 성능을 보였다.

3) 최종 모형 선택

다른 조건을 모두 같게 하고 오버 샘플링 처리 전후 모델을 비교한 결과 모든 모델에서 오버 샘플링 처리 후 모델이 좋은 성능을 보였다.
이것으로 불균형 데이터는 오버 샘플링으로 어느 정도 클래스의 균형을 맞춰 준다면 성능이 향상된다는 것을 확인했다.

최종적으로 AUC = 0.959, F1 = 0.937의 가장 좋은 성능을 보인
오버 샘플링 처리 후 학습시킨 스태킹(Stakinng) 모델로 최종 모형을 선택했다.