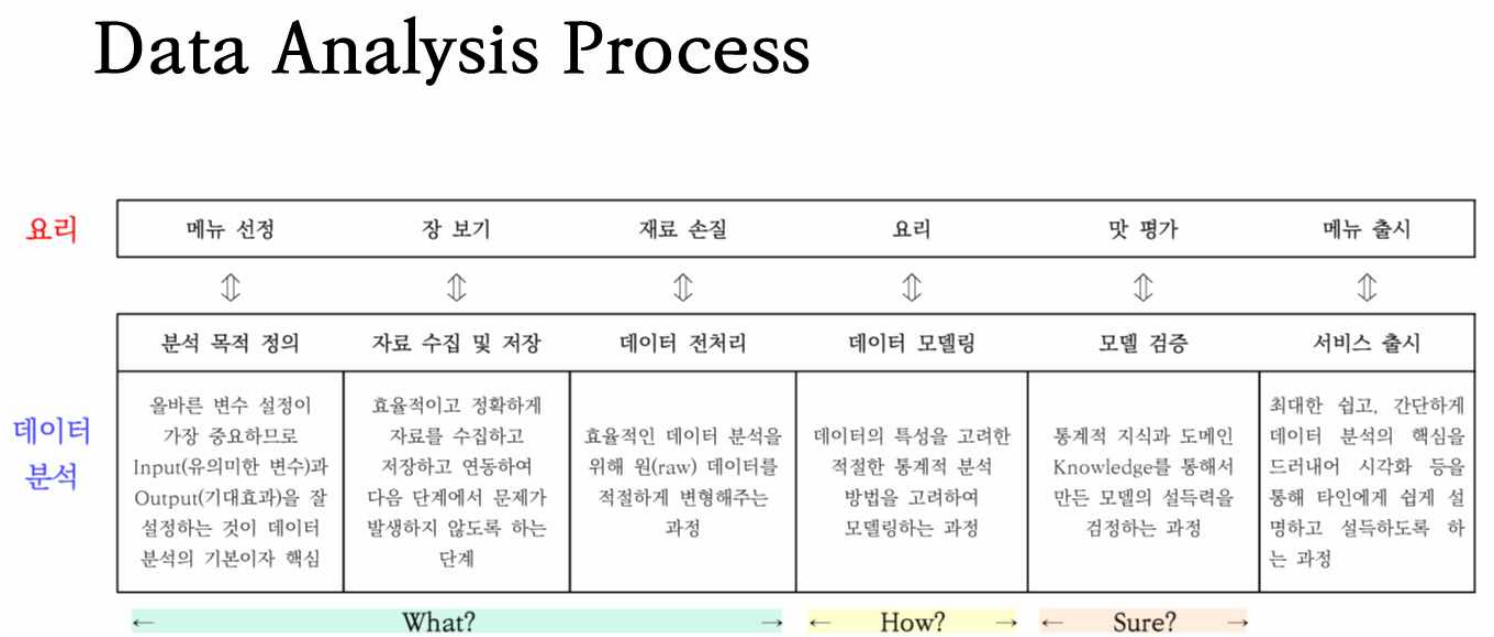


단계(Step)			내용	비고
1	Design	분석목적 정의	Ouput(기대효과)을 설정하고 이를 알아내기 위한 Input(유의미한 변수)을 적절하게 설정	3주차
2	Preprocessing	자료 수집 및 저장	Kaggle 출처의 csv 자료	3, 4주차
3		데이터 파악 및 가공	오타, 결측값, 명목형 변수 변형, 훈련 데이터와 검정 데이터로 분리	
4	Modeling	적절한 분석 도구 개발	데이터의 특성에 따른 방법론을 적용하여 신용 등급을 평가할 수 있는 예측 모델 개발	5, 6주차
5	Validation	예측 모델 검정	Test 데이터를 통하여 예측 모델의 예측정확성 검증	
6	Insight	평가, 결론, 시각화	자료에 대해 잘 모르는 사람도 이해할 수 있도록 분석의 핵심을 쉽고 간결하게 요약 및 시각화	

✔ 4단계와 5단계를 다시 Two Track(4.1, 5.1 / 4.2, 5.2)으로 나누어 각 트랙마다 서로 다른 모델을 만들고 평가할 예정이다.

Step_3.1 데이터 전처리 ① - 변수 파악

<신용 등급을 평가하는 데 영향을 주는 변수들의 의미>

ID	Customer_ID	Month	Name	Age
SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts
Num_Credit_Card	Interest_Rate	Num_of_Loan	Type_of_Loan	Delay_from_due_date
Num_of_Delayed_Payment	Changed_Credit_Limit	Num_Credit_Inquiries	Credit_Mix	Outstanding_Debt
Credit_Utilization_Ratio	Credit_History_Age	Payment_of_Min_Amount	Total_EMI_per_month	Amount_invested_monthly
Payment_Behaviour	Monthly_Balance	Target: Credit_Status ('Good', 'Standard', 'Poor')		

⇓

ID(인당 여러 개)	Customer_ID(인당 1개)	등급 평가를 받은 달	이름	나이
사회보장번호	직업	연 소득	월급	보유한 계좌 수
보유한 신용 카드 수	이자율 (개인별 가산금리)	대출 건수	대출 유형	연체일(납입일로부터 연체 일수)
연체 건수	변경된 신용한도	신용 조회 수	신용 조합(카드, 대출 등)	미지불 채무액
$\frac{\text{현재까지 사용한 카드값}}{\text{신용 카드 총 한도값}}$	신용 개설 후 기간	해당 납입일 납부 금액	등가 월간 할부	월 투자 금액
지불 행동 패턴	매월 지불해야 하는 총 금액	타겟 변수: 신용등급 (좋음, 보통, 나쁨)		

- 데이터 및 변수 특징 파악 -

✔ 신용 등급 평가를 받은 달: 인당 총 8회씩(1월 ~ 8월) → 월별 12500개의 데이터 존재
✔ 결측치 중 일부는 당연하게 알 수 있는 부분이므로 결측치를 채워서 보존할 수 있는 데이터들은 최대한 가져가는 방향으로 전처리
✔ 대략 24개 정도의 변수가 신용등급 평가에 영향을 미칠 것으로 판단됨
✔ Domaine Knowledge에 의해 신용등급 평가에 영향을 미치는 요인들을 조사할 수 있어서 참고 가능

설명변수 (24)	**수치형 (17)**	특이 사항: Credit_History_Age: 수치형으로 변환함 (예: 2 Years 3 Months → 27개월)	
	명목형 (7)	Month (신용등급 평가月)	수치형으로 전환 가능 (예: November → 11)
		Name (이름)	ID, Customer ID, SSN과 동일하다고 판단
		Occupation (직업)	Name이 같으면 Occupation도 같음
		Type of Loan (이용 중인 대출 종류)	9가지로 그룹화 가능
		Credit Mix (이용 중인 대출 종류 간 비율 안정성)	Good, Standard, Poor
		Payment of Min Amount (매월 납부해야하는 이자 포함 최소한의 돈)	No, Yes, NM (판단 불가)
		Payment Behaviour (지출 습관(패턴))	6가지 종류 가능 (spent: High, Low / value: High, Medium, Low)

Step_3.2 데이터 전처리 ② - 전처리 Sketch

변수	전처리 요소	변수	전처리 요소
Age	이상치* 제거, 하이픈* 제거, 음수* 제거, 동질화 작업*	Changed_Credit_Limit	하이픈 제거, 음수 제거, 평균화 작업
Occupation	하이픈 제거, 동질화 작업	Num_Credit_Inquiries	이상치 제거, 중위화 작업*
Annual_Income	하이픈 제거	Credit_Mix	하이픈 제거, 등차화 작업*
Monthly_Inhand_Salary	동질화 작업	Outstanding_Debt	하이픈 제거
Num_Bank_Accounts	이상치 제거, 동질화 작업	Credit_Utilization_Ratio	전처리 요소 없음
Num_Credit_Card	이상치 제거, 동질화 작업	Credit_History_Age	수치화 작업
Interest_Rate	이상치 제거, 동질화 작업	Payment_of_Min_Amount	NM 파악 불가로 인한 변수 폐기
Num_of_Loan	음수 제거, 이상치 제거, 하이픈 제거, 동질화 작업	Total_EMI_per_month	전처리 요소 없음
Type_of_Loan	9가지 경우의 수로 분류	Amount_invested_monthly	이상치 제거, 평균화 작업
Delay_from_due_date	전처리 요소 없음	Payment_Behaviour	일부 데이터에 대한 임의적 해석의 위험성에 따라 변수 폐기
Num_of_Delayed_Payment	음수 제거, 하이픈 제거, 평균화* 작업	Monthly_Balance	평균화 작업

✔ 상단 표에 sketch한 전처리 요소에 대해 변수 별 R을 활용해 직접 코딩하여 실행한 작업 결과물은 4주차에 '전처리 결과물 html 파일'로 이미 별도로 첨부함

✔ 상단 표에 사용된 용어(*) 설명은 다음 페이지에 기술함

✷ 전처리 용어 설명

(※ 공식적인 용어가 아니며 임의로 만든 용어)

※ 동질화: Name 혹은 Customer_ID가 동일한 데이터의 경우 모두 같은 변수값을 갖도록 변형하는 과정

① 결측치(빈칸)를 동일한 변수값을 갖도록 빈칸 채우기

② 음수 또는 이상치를 제거하고, 동일한 변수값을 갖도록 변형

→ 이상치의 경우 이상치 자체에 focus가 된다기보다 같은 사람에 대해서 1월~8월 데이터의 값이 모두 동일해야 하는데 그렇지 않은 경우 해당값을 동일한 값으로 바꿔서 넣어준다는 의미로 받아들이고 코딩하면 된다.

※ 평균화: Name 혹은 Customer_ID가 동일한 데이터가 변수에 대해 서로 다른 값을 갖을 때,

① 결측치(빈칸)는 나머지 1월~8월 데이터의 평균값으로 대체한다.

② 이상치의 경우도 나머지 1월~8월 데이터의 평균값으로 대체한다.

※ 중위화: Name 혹은 Customer_ID가 동일한 데이터가 변수에 대해 서로 다른 값을 갖을 때,

① 결측치(빈칸)는 나머지 1월~8월 데이터의 중위수(median)로 대체한다.

② 이상치의 경우도 나머지 1월~8월 데이터의 중위수(median)로 대체한다.

※ 등차화: Name 혹은 Customer_ID가 동일한 데이터는 Month가 1씩 증가할 때마다 해당 변수의 데이터 값도 1씩 증가 시킨다.

※ 이상치 제거: 해당 변수의 데이터 값이 물리적으로 불가능한 값을 갖는 수준으로 크거나 작은 값을 가질 때 해당 데이터를 삭제하고 결측치 상태로 두거나 동질화, 평균화, 중위화 등의 처리를 한다.

※ 음수 제거: 해당 변수의 데이터 값이 물리적으로 음수값을 가지지 못할 때 음수를 갖는 데이터 값을 삭제하고 결측치 상태로 두거나 동질화, 평균화, 중위화 등의 처리를 한다.

※ 하이픈 제거: 하이픈(_ 또는 ____)과 같은 불필요한 표기를 제거하여 수치화 시킨다.

① 회귀분석 (Regression Analysis)

: 어떠한 대상 혹은 현상에 관련된 변수들간에 상호 관련성을 찾는 것은 당연하고도 중요한 과정이다. 이 변수들간에 어떠한 수학적 연관성을 갖고 있다고 한다면 이들 간에 함수관계를 규명하는 것은 매우 의미있는 작업일 수 있다. 변수에는 크게 두 가지 종류가 있다. 다른 변수들에 영향을 받지 않는, 말 그대로 독립적인 지위에 있는 변수를 **독립변수**(independent variable)라고 하고, 독립변수의 영향을 받아 결정되는 변수를 **종속변수**(dependent variable)라고 한다. 독립변수는 종속변수를 설명할 수 있다는 의미에서 **설명변수**(explanatory variable)로, 종속변수는 독립변수에 의해 반응한다는 의미에서 **반응변수**(response variable)라고도 한다.

: 다시 말해 독립변수와 종속변수 사이의 함수관계를 규명해내기 위해 투자하는 시간과 노력은 가치가 있다. 이렇게 변수들간에 함수관계를 알아내는 통계적인 방법을 **회귀분석**이라고 한다. '회귀'라는 용어는 영국의 우생학자 F.Galton이 아버지와 아들의 키에 대한 관계를 조사하며 최초로 사용했다.

② 다중회귀분석 (Multiple Regression Analysis)

: 다중선형회귀분석의 목적은 두 개 이상의 변수 사이의 관계를 선형식으로 나타내는 것이다. 현실적으로 두 변수 사이의 관계가 선형적인 경우는 많지 않으므로 독립변수군과 종속변수 간에 직선회귀모형(straight line regression model)을 적합시키기 위해서는 다음과 같은 가정을 전제 조건으로 해야 한다.

(예시는 x와 y의 관계를 나타내는 단순선형회귀 조건으로 대신하여 설명한다.)

① 모형의 선형성

: 변수 x와 y 간에 존재하는 관련성은 주어진 x에 대한 y의 기댓값을 $\mu_{y\cdot x}$라고 할 때, 다음과 같은 선형식$\mu_{y\cdot x}=\beta_0+\beta_1 x$으로 표현될 수 있다.

② 정규성 가정

: 주어진 x의 값에서 변수 y는 정규분포(normal distribution)를 하고, y의 측정오차항 $\epsilon \sim N(0,\sigma^2)$이다.

② 독립성 가정과 등분산성 가정

: x는 오차 없이 측정할 수 있는 수학변수(mathematical variable)이며, 종속변수 y는 측정오차를 수반하는 확률변수 (random variable)이다. 이때 y의 측정오차들은 서로 독립 즉, $Cov(\epsilon_i,\epsilon_j)=0,\ i\neq j$이다. 평균은 $\mu_{y\cdot x}=\beta_0+\beta_1 x$로 x에 따라서 변하지만 분산은 x의 값과 관계없이 일정하며, $\epsilon \sim N(0,\sigma^2)$이다.

위의 3가지 가정을 간단히 나타내면 $y_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$ 또는 $\epsilon_i \overset{iid}{\sim} N(0,\sigma^2)$라고 하고, 회귀모형으로 나타내면 다음과 같다.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \ (iid : \text{independent identically distributed})$$

③ 분류 (Classification)

: 대표적인 데이터 분석기법으로서 분류는 해당 데이터가 각각 어떤 그룹에 속하는지 예측하는데 사용된다. 우리가 하려는 분석의 경우 신용등급에 영향을 미치는 다양한 요소들을 통해 각각 'Good', 'Standard', 'Poor' 3가지 등급 중 어떤 신용등급을 부여받는지 효율적으로 분류하는 모델을 만든다.

④ 로지스틱회귀분석 (Logistic Regression Analysis)

: ②에서 설명한 것과 달리 실제 많은 사회현상에서 특정 변수에 대한 확률값은 선형식의 형태가 아닌 형태를 따르는 경우가 많다. 이러한 비선형 함수를 표현해내는 함수를 로지스틱 함수라고 하는데, 이 로지스틱 함수는 출력 결과가 항상 0과 1 사이다. 확률값도 언제나 0과 1 사이이므로 나온 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 알고리즘을 로지스틱회귀모델이라고 한다.

⑤ 기계학습 (Machine Learning)

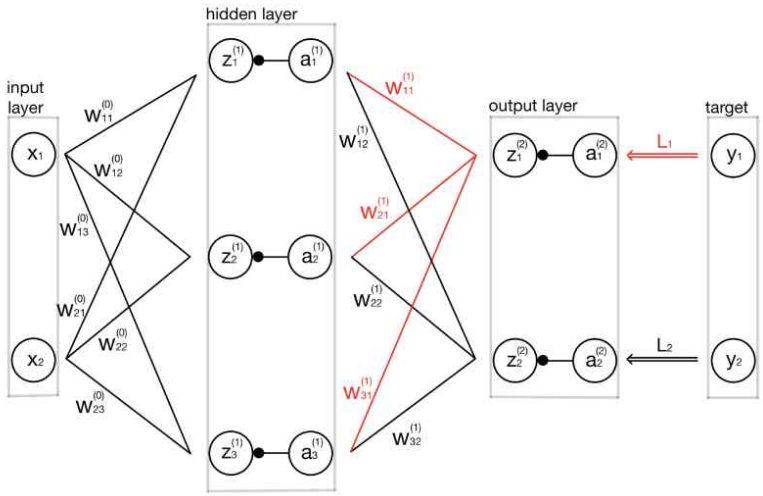
: 데이터 속에 숨겨진 패턴을 인식하거나, 그 패턴을 이용해 불확실성을 가진 대상을 예측하는데 사용되는 자동화된 학습 알고리즘을 통칭한다.

⑥ 지도학습 (Supervised Learning)

: 기계학습은 크게 지도학습, 비지도학습, 지도/비지도학습으로 나뉜다. 지도학습은 음성 및 안면 인식, 질병의 진단 등과 같이 정답이 라벨링 되어 있는 데이터를 분석한 모형을 이용하는 학습기법으로 새로운 입력 데이터가 들어왔을 때 대응되는 라벨값을 정확히 예측할 수 있도록 학습시키는 방법이다.

⑦ 인공신경망모형 (Artificial NeuralNet Model)

: 순방향 신경망 모형은 입력벡터 $x=(x_1, x_2, \cdots)^\top$와 출력벡터 $y=(y_1, y_2, \cdots)^\top$사이의 관계를 네트워크 형태로 연결하는 모형으로 특히 은닉층을 여럿 쌓아서 층의 깊이를 깊게 만든 신경망 모형을 심층신경망 모형 또는 딥러닝 모형이라고 한다.



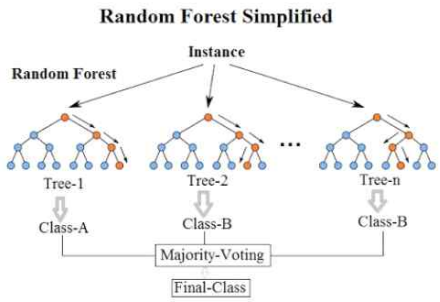
<그림_1>

✔ 간단한 신경망모델인 〈그림_1〉을 살펴보자. input layer에 데이터를 입력하게 되면 가중치를 얻고 hidden layer를 통해서 output layer에 해당 데이터에 대한 예측값이 출력되고, 이 예측값과 실제 라벨값(정답)을 비교하여 그 차이(오차)를 계산하며, 이 오차를 줄이는 방향으로 계속 학습해 나아가는 것이 인공신경망 모형의 기본적인 흐름이다.

⑧ 배깅 알고리즘(Bootstrap Aggregating Algorithm)과 랜덤포레스트 기법(Random Forest)

: 배깅 알고리즘은 분류 알고리즘의 하나로 Bootstrap Aggregating 알고리즘이라고도 하며, Bootstrap Aggregating의 앞글자와 뒷글자를 붙여서 네이밍 되었다. 하나의 나무를 사용하는 의사결정나무(Decision Tree)와 다르게 여러 개의 Tree를 앙상블하여(합하여) 최종 Voting(결과)를 결정한다. 배깅의 가장 큰 특징이라고 한다면 주어진 데이터 셋에서 샘플링을 할 때 무작위 복원 추출을 진행한다는 점이다. 때문에 각각의 Tree에 대한 학습은 독립적이고, 병렬적으로 진행이 되므로 학습 속도가 빠르다.

: 배깅의 일종인 랜덤 포레스트는 배깅과 방식이 모두 동일하나, 모든 변수를 사용하여 트리를 구성하는 배깅과 다르게 시작점에서 사용할 변수의 개수에 제한을 두어 특정한 변수들끼리 연관이 되어 왜곡된 결과를 낳지 않도록 제한된 수의 변수도 랜덤 샘플링을 하는 점이 다른 알고리즘이다.



⑨ 부스팅 알고리즘(Boosting Algorithm)과 XGBoosting Tree

: 부스팅 알고리즘은 앞서 설명한 배깅 알고리즘과 유사하나 피드백을 하여 데이터에 가중치를 준다는 점에서 큰 차이를 보인다. 부스팅은 첫 번째 트리에서 오분류된 데이터에 대하여 가중치를 부여하여 다음 순서의 트리에서 데이터를 샘플링할 때 더 높은 확률로 선택될 수 있는 장치를 마련하여 학습이 진행될수록 오분류될 데이터를 줄이게 해주는 알고리즘인 것이다.

: 부스팅 알고리즘에서 오분류를 줄이는 방향으로 학습을 경사하강법(Gradient Method)을 이용하는 알고리즘을 그레디언트 부스팅(Gradient Boosting)이라고 하는데 XGBoosting Tree는 이러한 Gradient 방식을 극한의 수준으로 끌어올린 알고리즘으로 생각할 수 있다. 그래서 이름 또한 Extreme Gradient Boosting의 줄임말이다. 하드웨어적 요소와 소프트웨어적 요소가 모두 결합된 최신의 알고리즘으로 최근 다양한 분야에서 가장 각광받는 알고리즘이다. 이전의 알고리즘에 비해 속도와 예측 성능이 상당히 높으며 데이터를 병렬처리할 수 있다는 큰 장점이 있다.

Step_4.1 & 5.1
모델링: 다중선형회귀 및 다중로지스틱회귀모형

- 분석 과정 -

종속 변수가 범주형이고, 독립 변수가 연속형이거나 둘 이상일 때 로지스틱 회귀분석을 수행한다. 이분형 로지스틱 회귀분석은 종속 변수의 범주가 2개인 경우, 다중명목형 로지스틱 회귀분석은 종속 변수의 범주가 3개 이상인 경우, 순서형 로지스틱 회귀분석은 종속 변수가 서열을 나타낼 때 사용한다.

1. 데이터 확인 및 추가 가공

✔ 앞서 전처리리 한 데이터를 순서형 로지스틱 회귀분석으로 돌리기 위해 추가적으로 전처리를 진행하였다.
✔ 범주형 변수를 Factor 형으로 변환하여 설명변수에 넣을 수 있도록 가공하였으며, 전처리가 아직 잘 안 된 'Annual_Income' 변수는 다시 전처리하여 숫자형 변수로 만들었다.
✔ 또한 숫자형이지만 아직 숫자형으로 인식되지 않은 변수 또한 변환하여 준비하였다고, 무한소수로 입력되어 있는 변수들을 반올림하여 정리하였다.

전처리 다시하기

```
# 범주형 변수 factor형으로 변환
train$Credit_Score <- as.factor(train$Credit_Score)
train$Payment_Behaviour <- as.factor(train$Payment_Behaviour)
train$Payment_of_Min_Amount <- as.factor(train$Payment_of_Min_Amount)
train$Credit_Mix <- as.factor(train$Credit_Mix)
train$Type_of_Loan <- as.factor(train$Type_of_Loan)
train$Annual_Income <- as.numeric(gsub("_","",train$Annual_Income))
train$Occupation <- as.factor(train$Occupation)

# 반올림하여 정리
x <- c()
y <- c()
for(i in 1 : 38){
  x[i] <- is.integer(train[,i])
  y[i] <- is.factor(train[,i])
}
x
```

```
##  [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [13]  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [37]  TRUE  TRUE
```

```
y
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [25] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE
```

```
train$Credit_Utilization_Ratio <- round(train$Credit_Utilization_Ratio, 2)
train$Total_EMI_per_month <- round(train$Total_EMI_per_month, 2)
train$Amount_Invested_monthly <- round(train$Amount_Invested_monthly, 2)
train$Monthly_Balance <- round(train$Monthly_Balance, 2)
train$Changed_Credit_Limit <- round(train$Changed_Credit_Limit, 2)
train$Annual_Income <- round(train$Annual_Income, 3)
train$Monthly_Inhand_Salary <- round(train$Monthly_Inhand_Salary, 4)
train$Num_of_Delayed_Payment <- round(train$Num_of_Delayed_Payment, 0)
train$Num_Credit_Inquiries <- round(train$Num_Credit_Inquiries, 0)
```

2. 메모리 사이즈 확대

✔ 우리가 다루는 데이터는 100,000개의 데이터로 꽤 많은 양에 속한다. 따라서 코딩을 돌릴 때에 메모리가 부족하다는 warning이 계속하여 발생하였다. 메모리 사이즈를 확인하고 추가 메모리를 확대할 수 있도록 코딩을 작성하였다.

2. 모델링(Modeling)

✔ 우리가 다루는 신용등급 평가 데이터에서 종속 변수는 Credit_Score이고, 그 값은 Good, Standard, Poor 3종류로 서열을 나타낸다고 볼 수 있다.
따라서 순서형 로지스틱 회귀분석을 수행하는 polr 함수를 사용해 확인해 본 결과, 모든 변수를 다 넣었을 때, 대출 유형 별 개수를 제거했을 때, 수치형 자료들만 넣었을 때, 이 3가지 경우 중 모든 변수를 다 넣었을 때 AIC 값이 가장 작은 것을 확인하였다.

✔ plor 함수를 사용하여 결과를 확인하려면 많은 시간이 소요되었다 팀원 곽효리와 이윤지는 각자 모델을 돌려봄으로써 두 팀원이 만든 모델 중 가장 AIC가 낮은 모델을 찾으려고 하였다. 필요한 패키지는 “MASS”와 “AER”이다. 우리가 다루는 변수의 총 개수는 38개이었다. 너무 많은 변수를 집어넣었기 때문에 warning에서는 계속 설명변수를 줄이라고 경고가 나왔다. 그러나 우리는 함부로 변수를 줄일 수 없어서 그대로 진행하기로 결정했다. 다만, 월별로 나눠서 모델링 해보기도 하고, 대출유형만 설명변수에 넣어서(다중회귀_윤지_모델2) 신용등급을 찾는 모델을 만들려고 하는 등 여러 가지 방안을 시도해보았다.

✔ AIC 값이 가장 작은 최적의 모델을 찾기 위해 step 함수를 사용하여 단계별 선택을 진행하였다. 단계별 선택은 독립변수를 연속적으로 추가/삭제하면서 AIC 값이 낮아지는 모델을 찾는 방법이다.

✔ 그 결과 29개의 독립 변수 중 19개의 독립 변수로 이루어진 모델이 AIC 값 177492.11로 가장 작은 것을 확인하였다.

(19개의 독립 변수)
: Credit_Mix, Interest_Rate, Changed_Credit_Limit, Outstanding_Debt, Credit_History_Age, Month, Monthly_Balance, Num_Bank_Accounts, Num_Credit_Card, Delay_from_due_date, Num_of_Loan, Occupation, StudentLoan, PersonalLoan, Age, Annual_Income, NotSpecified, CreditBuilderLoan, Credit_Utilization_Ratio)

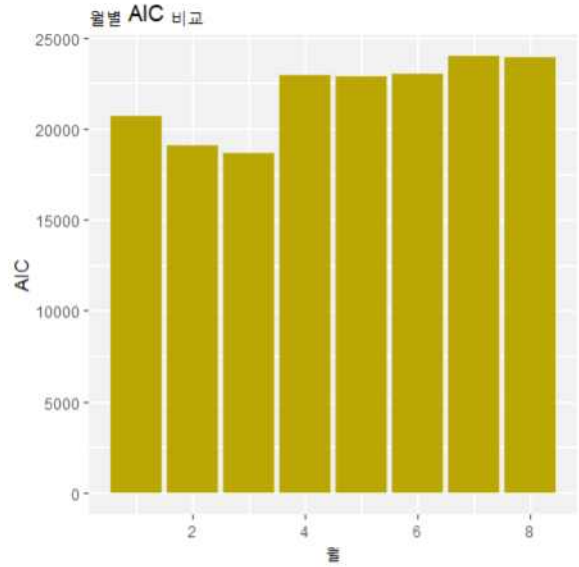
✔ 마지막으로 독립 변수가 n개 있을 때, 각 변수를 추가하거나 뺀 2n개의 회귀 모델을 만들고 이들 모두를 비교해보는 방법을 사용해보았다. 이때 사용한 함수는 regsubsets()라는 함수로 모든 가능한 모델과 단계적 알고리즘을 사용해 모델을 선택한다.

✔ regsubsets()가 반환한 객체를 summary()함수에 넘겨주면 BIC, 수정 결정 계수 등의 값을 얻을 수 있다.

✔ plot()함수를 이용해 BIC 기준으로 모델을 살펴보았고, BIC가 가장 좋은 모델은 절편, Credit_Mix, Interest_Rate, Outstanding_Debt, Month, Changed_Credit_Limit, Num_Bank_Accounts, Monthly_Balance를 포함한 모델로 이때 BIC 값은 -23166.77이다.

✔ 마지막으로 이 자료의 특성상 12500명의 고객들의 정보가 1월부터 8월까지 데이터가 누적되어 있는 형태이므로 월별로 나눠 각 고객들의 데이터에 대한 모델을 만들어 보았다. 100,000명의 데이터를 사용한 것보다 확실하게 AIC가 줄어든 것을 확인할 수 있었다. 1월의 AIC는 20697.78이고, 2월은 19087.18, 3월은 18665.11, 4월은 22938.73, 5월은 22904.33, 6월은 23013.04, 7월은 23989.97, 8월은 23956.78로 나타났다.

```
month <- c(1,2,3,4,5,6,7,8)
aic <- c( 20697.78,  19087.18,  18665.11,  22938.73,  22904.33,
23013.04, 23989.97, 23956.78)
last <- data.frame(month, aic)
last
library(ggplot2)
ggplot(last, aes(x=month, y=aic)) +
  geom_bar(stat="identity", fill="#B6A204") +
  labs(x="월", y="AIC", title="월별 AIC 비교")
```



위 결과를 통해 변수에 개수에 따라서도 AIC가 변화되지만, 데이터의 수에 따라 AIC가 좌지우지된다는 걸 확인할 수 있었다.

Step_4.2 & 5.2
모델링: 지도학습 기반의 분류 모형

- 데이터 추가 가공 -

✔ 앞서 지난주(5주차) pdf 첨부 자료-① '5주차_R을 활용한 데이터분석_Team_B_GuideLine'에서 설명한 바와 같이 Team_B의 데이터 분석에서는 명목형 변수는 제외하고 수치형 변수만을 가지고 분석을 실시하기로 했기 때문에 하단 표에 정리한 명목형 변수들을 모두 제외하였다.

제거한 변수					
ID	Customer_ID	Name	SSN	Occupation	Type_of_Loan
Payment_of_Min_Amount	Payment_Behaviour	AutoLoan	PersonalLoan	HomeEquityLoan	CreditBuilderLoan
NotSpecified	MortgageLoan	StudentLoan	DebtConsolidationLoan	PaydayLoan	

✔ 'Month' 변수의 특징을 살펴보면, 12,500명이 고객이 1월부터 8월까지 매달 신용 등급 평가를 받았으므로 특정한 달(Month)을 선택해 고정시킨 후 나머지 변수들에 대하여 예측 모델을 만드는 것이 전혀 문제가 없으므로 이번 분석에서는 1월(January)에 대한 신용등급평가 예측모델을 만드는 것으로 하고, 100,000개의 데이터 중 1월 데이터인 12,500개만 추출하도록 한다.

✔ 수치형 변수들을 관찰해보면 서로 단위가 다르기 때문에 10보다 작은 데이터부터 10,000,000이 넘는 데이터까지 다양하게 존재한다. 그러니 'ANN' 모형과 'Random Forest' 모형의 경우 이러한 자료를 그대로 사용하게 되면 특정 변수들은 신용등급을 평가하는데 거의 영향을 미치지 않는 요소로 평가가 될 수 있는 왜곡이 발생할 수 있으므로 모든 변수의 범위를 같은 범위로 만들어주기 위하여 정규화 데이터 스케일링(Normalizaition Data scaling)을 진행한다. 대표적으로 'Z-Score Normalization'과 'min-max Nornalization'이 있는데 우리의 데이터는 'min-max Nornalization'을 사용해 모든 수치형 자료들을 0과 1 사이의 값을 갖도록 바꾸어 주기로 한다.

- 1. 인공신경망 모형 (ANN Model) -

✔ 1.1 Class Imbalance (데이터 불균형)을 해결하기 위한 Stratified Sampling(층화추출법)

: 데이터 불균형이란 어떤 데이터에서 각 타겟 변수(범주)가 갖는 데이터의 양에 차이가 큰 경우에 발생하는 문제를 말한다. 우리의 데이터로 설명을 해보자면 신용등급은 총 3가지 'Good', 'Standard', 'Poor' 등급으로 분류가 되는데 각각의 'Good', 'Standard', 'Poor' 라벨이 되어 있는 데이터의 개수는 각각 다음과 같다.

▶ 'Good': 1928개 ▶ 'Standard': 7024개 ▶ 'Poor': 3548개

: 위의 결과에서 알 수 있듯이 'Good'의 개수와 'Standard'의 데이터 수 차이가 3배 이상의 차이가 발생하므로 이 경우 우리가 만든 모델이 'Standard'에 편향된 학습을 할 가능성이 커지고, 결과적으로 편향된 모델이 나오게 되면서 그 모형의 예측 정확도를 신뢰할 수 없게 된다는 문제점이 발생할 수 있다. 따라서 편향되지 않고, 높은 예측 정확도를 갖는 모델을 만들기 위해서 가장 데이터 수가 적은 범주에 맞추어 나머지 범주의 크기(데이터 수)를 통일하고 그 안에서 단순무작위표본추출법(simple random sampling)을 사용하는 층화추출법(stratified sampling)을 사용하도록 한다. 우리의 데이터의 경우 'Good'의 데이터 개수가 가장 작으므로 'Standard'와 'Poor'에 속하는 데이터의 수를 모두 1928개로 줄인다. 이 과정에서 각각의 1928개의 데이터는 단순무작위 원리로 추출되어 독립성을 보장한다.

1.2) Stratified extraction(층화추출)

```
#Standard_7024, Poor_3548, Good_1928
#install.packages("sampling")
library(sampling)
stratified_sampling <- strata(data_1, stratanames = c("Credit_Score"), size =c(1928, 1928, 1928),
                              method="srswor")

data_2 <- getdata(data_1, stratified_sampling)
table(data_2$Credit_Score)
```

```
##
##     Good     Poor Standard
##     1928     1928     1928
```

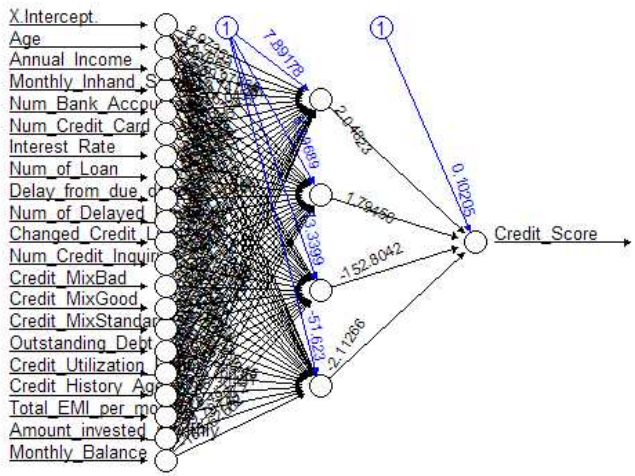
: R의 'sampling' 라이브러리에서 층화추출 옵션을 선택하여 코딩을 하게 되면 어렵지 않게 층화추출한 데이터를 얻을 수 있는데 이 과정에서 csv 파일의 열(column)에 분석에 필요없는 3가지 열(column)이 추가로 생성되므로 데이터 분할을 한 뒤 마지막 세 열(column)에 해당하는 'ID_unit', 'Prob', 'Stratum' 데이터를 지워주기로 한다.

✔ 1.2 데이터 분할 (Data Split)

: 위의 과정을 진행하면 현재 우리가 보유한 데이터 수는 $1928 \times 3 = 5784$개다. 이 데이터를 모두 모델의 학습에 사용하는 것이 아니라 일정 비율로 다시 나누어 학습에 사용할 train Data(학습 데이터)와 그러한 학습을 통해 만들어진 모델의 예측 정확성을 평가하는데 쓸 수 있는 test data(평가 데이터)로 나누어야 한다. 우리의 분석에서 5784개의 데이터를 train:test = 80 : 20의 비율로 나누어 사용하도록 한다.

✔ 1.3 모델링 (Modeling)

: R의 'neuralnet' 패키지의 'neuralnet' 라이브러리를 이용하여 인공신경망 모형을 만들도록 한다. 이때 가장 중요한 Node와 Layer의 수를 결정할 수 있는데 2개 이상의 Layer를 설정하는 딥러닝 모형의 경우 정확한 원인을 알기 어려운 사유로 학습 과정에서 오류가 발생을 하기에 1개의 Layer에 4개의 Node로 구성된 인공신경망으로 학습시키도록 한다. 학습시킨 모델을 시각화하면 다음과 같다.



✔ 1.4 모델 평가 (Validation)

: 이제 만든 신경망 모델의 예측정확도를 평가해보자. 예측정확도 평가에는 혼동행렬(Confusion Matrix)를 활용한다. 혼동행렬(Confusion Matrix)에 대한 설명은 아래와 같다.

✷ 혼동행렬(Confusion Matrix)

: 혼동행렬은 예측값이 실제 관측값을 얼마나 정확히 예측했는지 보여주는 행렬로 모델의 성능을 평가할 때 자주 사용되는 지표이다.

##	Class: 0	Class: 0.5	Class: 1
## Sensitivity	0.8350	0.5913	0.6456
## Specificity	0.8635	0.7592	0.9198
## Pos Pred Value	0.7598	0.5332	0.8070
## Neg Pred Value	0.9101	0.7997	0.8333
## Prevalence	0.3408	0.3175	0.3417
## Detection Rate	0.2846	0.1877	0.2206
## Detection Prevalence	0.3746	0.3521	0.2734
## Balanced Accuracy	0.8493	0.6752	0.7827

<그림_3>

✔ 혼동 행렬을 통해 계산한 지표들을 나타내는 **<그림_3>**을 살펴보자. “Class 0”, “Class 0.5”, “Class 1”은 각각 신용등급이 ‘Poor’, ‘Standard’, ‘Good’를 의미하고, 이때 각 클래스에 대한 Blanced Accuracy 값을 보면 0.8464, 0.7067, 0.8271로 각각 약 84%, 70%, 82% 정도의 예측정확성을 보이는 것으로 나온다.

- 2. 랜덤 포레스트 (Random Forest) -

✔ 2.1 모델링 (Modeling)

: R의 'randomForest' 패키지의 'randomForest' 라이브러리를 활용한다.

✔ 2.2 모델 평가 (Validation)

: 랜덤포레스트 역시 인공신경망 모형과 동일하게 혼동행렬(Confusion Matrix)을 사용하여 예측 정확도를 평가해본다. 혼동 행렬을 통해 계산한 지표들을 나타내는 **<그림_4>**를 살펴보자

##	Class: Good	Class: Poor	Class: Standard
## Sensitivity	0.9899	0.9721	0.8610
## Specificity	0.9671	0.9659	0.9810
## Pos Pred Value	0.9399	0.9364	0.9547
## Neg Pred Value	0.9946	0.9853	0.9382
## Prevalence	0.3417	0.3408	0.3175
## Detection Rate	0.3382	0.3313	0.2734
## Detection Prevalence	0.3599	0.3538	0.2863
## Balanced Accuracy	0.9785	0.9690	0.9210

<그림_4>

✔ 각 클래스에 대한 Blanced Accuracy 값을 보면 0.9785, 0.9690, 0.9210로 각각 약 98%, 97%, 92% 정도의 예측정확성을 보이는 것으로 나온다.

- 3. XGBoosting Tree -

✔ 3.1 모델링 (Modeling) 및 모델 평가(Validation)

: R의 'xgboost' 패키지의 'xgboost' 라이브러리를 활용하여 모델의 예측력을 구하면 다음과 같다.

3.6) KS statistics for train_set & test_set

```
#install.packages("MLmetrics")
library(MLmetrics)
```

```
##
## 다음의 패키지를 부착합니다: 'MLmetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```
KS_Stat(train_y_pred, train_y)
```

```
## [1] 58.72795
```

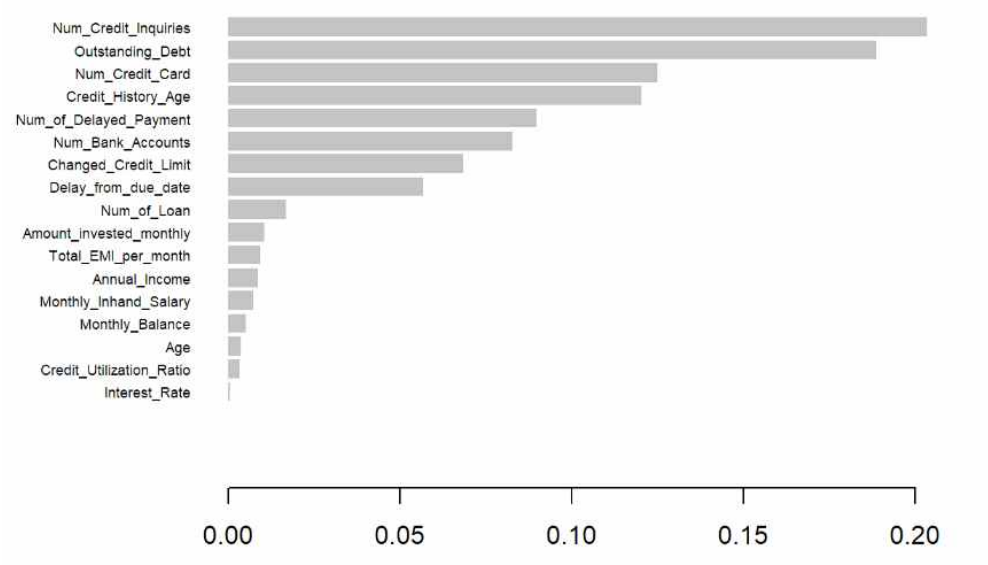
```
KS_Stat(test_y_pred, test_y)
```

```
## [1] 53.73812
```

<그림_5>

✔ **<그림_5>**를 살펴보면 예측 정확도가 약 약 50%대 초중반 정도를 보인다.

✔ XGB를 통해 어떠한 변수가 신용등급 평가에 가장 큰 영향을 주었는지 **<그림_6>**을 통해 파악할 수 있는데 신용 조회 횟수, 큰 빚, 신용카드의 개수, 신용이 발생한 이후 지난 시간 순으로 큰 영향을 미친다는 사실을 알 수 있다.



<그림_6>

✔ 신용 등급 평가 예측모델의 예측정확도를 높일 수 있는 3가지 알고리즘인 '심층인공신경망(딥러닝)(Deep Artificial NeuralNetwork)', '랜덤포레스트(Random Forest)', 'XGBoosting Tree'을 기반으로 만든 모델을 서로 비교하는 작업을 진행했고, 그 결과 예측정확도는 랜덤포레스트, 인공신경망, XGBoosting Tree 순으로 좋았다. 각 알고리즘마다 주어진 옵션을 달리하면 예측정확도가 변화할 수 있는 만큼 절대적인 순서라기 보다는 대략적인 분석 결과로 받아 들이는 것이 좋을 것 같다. XGBoosting Tree의 결과가 예상보다 낮아 해당 모델에 대한 개선을 추가로 해보고 싶다는 생각이 든다.