< 중고차 가격 (시세) 예측 >

안현진
박재근
김 순

목 차

1. 프로젝트 개요

1.1 도메인

◆ 중고차 가격 예측

차를 바꿀 때, 기존에 타던 차는 마음대로 버릴 수 없다.
따라서 새 차를 구입하면 기존의 차는 폐차, 양도, 중고 판매 등의 방법으로 처리하게 된다.

보통 앞선 방법들은 처리 비용이 들어가고 경제적인 이익을 전혀 얻을 수 없기 때문에
대부분의 사람들은 자동차를 중고 시장에 내놓는 것을 선택하게 된다.

또한 최근에는 코로나-19 확산에 따른 경기 회복에 대한 불확실성으로 수입 감소 우려와 재택 근무 확산으로 새 차에 대한 수요가 줄어들고 대신 중고차 인기가 상승하게 되었다.

실제 GM과 토요타는 올해 2분기 판매 실적이 각각 34% 이상 감소했으며
중고차 가격 지수인 '맨하임 인덱스(Manheim index)'에 따르면, 전 세계 500만대 중고차의
매입-매도 현황을 분석한 결과 가격 지수가 162.3포인트로 역대 최대를 기록했다.

이렇게 중고차 시장은 성장하게 되었으며, 코로나로 인해 중개인(딜러)를 거치지 않고 온라인으로 거래하는 비대면 중고 거래 및 직거래가 유행하게 되었다.

문제는, 비전문인들이 직거래를 나서게 되면서 중고차 시세에 대해서 파악하기가 어렵다는 것이다.
비대면 판매 서비스를 제공하는 업체 'AJ셀카'가 이용객들을 대상으로 설문조사를 실시했는데,
'차량 판매가격이 적정 수준인지 궁금하다'는 답변이 44.9%로 가장 많았고
'딜러 거래 시 현장 감가 될까 불안하다'는 답변이 26.4%로 뒤를 이었다.

즉, 10명중 7명은 차량 판매 금액에 대한 불신과 불안을 갖고 있다는 통계가 나오게 되었다.

고로 이 프로젝트에서는 기존의 중고차 시세를 분석하여 사용자의 차 가격을 예측하려고 한다.
즉, 판매 가격을 예측하여 사용자 차량의 시장 가치에 대한 정보를 제공해주는 것이다.

코로나-19로 인해 성장하고 변화하는 중고차 시장에서 비전문인도 활용 가능한, 확인이 쉽고 신뢰도가 있는 시세 정보를 제공하는 것이 프로젝트의 목표이다.

1.2 기대 효과

프로젝트의 결과로 만들어진 모델의 활용 방안과 기대 효과를 정리하였다.

◆ 가격 책정에 대한 객관적인 지표 제시

① 판매자 입장 :

개인 중고 판매 과정이 복잡하고 전문적이기에 보통 판매자들은 중고차 매입 딜러들과 거래한다. 이 때, 모델로 만들어진 객관적인 지표를 활용한다면, 내 차의 시세 가치를 파악할 수 있으므로 딜러와의 가격 협상에서 유리한 위치를 점할 수 있으며, 판매 의사결정에 큰 도움이 된다.

또한 개인간의 직거래에서도 모델로 만들어진 지표를 토대로 가격을 책정한다면,
비전문가가 책정한 판매 가격이라도 해당 부분에 신뢰도를 부여해주는 효과를 얻을 수 있다.

② 구매자 입장 :
보통 중고차를 구입할 때는 발품을 팔며 여러 옵션을 따지고 중고 시세를 파악하여 구매할 중고차를 신중하게 결정하게 된다. 문제는 이 과정에서 허위 매물을 판매하는 판매자 및 딜러가 존재한다는 것이다.

이 때, 모델로 만들어진 객관적인 지표를 활용한다면, 구매하려고 하는 중고차의 시세 가치를 파악할 수 있으므로, 그들이 책정한 가격이 과연 합리적인 가격인지 판단할 수 있다.

③ 거래 서비스 운영자 :

지표를 제공하고, 사용 데이터 셋을 공개함으로써 이용자들에게 신뢰를 얻을 수 있다.
개선된 이미지와 신뢰도를 토대로 서비스 이용자들을 늘릴 수 있다.

④ 공통 : 모두의 입장에서, 시장에 매물이 전혀 없는 경우에도 가격을 예측할 수 있다.

◆ 특정 차량의 선호 정보 제공

① 제조사 :
중고 시장에서 가격은 곧 수요와 직결된다. 제조사는, 중고차 시장에서의 수요(선호도)를 바탕으로 보증기간 설정 및 신차 가격 책정 등 다양한 전략을 수정할 수 있다.

또한 제조사가 선호도 정보를 기반으로 직접 중고차 시장에 뛰어들 수 있다.
특정 차종에 대해 수요가 높다면 제조사가 직접 품질을 관리하여 양질의 중고 차량을 제공하고 혼탁한 기존 시장에 반해 믿을 수 있다는 신뢰적인 이미지와 함께 경제적인 효과를 누릴 수 있다.

실제 수입차 업체들은 품질 관리를 위해 중고차 시장에 관여하고 있으며, 해당 선호도 정보는 품질 관리 차종 선종에 도움이 될 수 있다.

1.3 입력 데이터

◆ Instance (Total 3000)

중고차 거래 사이트 Craigslist의 시세표를 기준으로 각 가격대마다 600개씩, 총 3000개 사용

◆ Attributes (Total 11)

Attribute Type	Attribute Name	Attribute Value	
Nominal	Manufacture	Chevrolet	Jeep
		Ram	Ponitac
		Ford	Mercedes-benz
		Honda	Volkswagen
		Subaru	Mercury
		Chrysler	Audi
		GMC	Acura
		Mitsubishi	Buick
		KIA	Jaguar
		Dodge	Lincoln
		Nissan	Rover
		Saturn	Infiniti
		Toyota	Lexus
		Cadillac	Mini
		Mazda	BMW
		Hyundai	Volvo
		Aston-martin	Fiat
		Ferrari	

Attribute Type	Attribute Name	Attribute Value	
Nominal	Type	Sedan	Pickup
		Hatchback	SUV
		Van	Truck
		Coupe	Mini-van
		Convertible	Wagon
		Offroad	Bus
		Other	

Attribute Type	Attribute Name	Attribute Value	
Nominal	Size	Mid-size	Full-size
		Compact	Sub-compact
Attribute Type	**Attribute Name**	**Attribute Value**	

Nominal	condition	Like New	Excellent
		Good	Fair
		New	

Attribute Type	Attribute Name	Attribute Value	
Nominal	Fuel	Gas	Hybrid
		Diesel	Electric
		Other	

Attribute Type	Attribute Name	Attribute Value	
Nominal	Cylinders	4 Cylinders	5 Cylinders
		6 Cylinders	8 Cylinders
		10 Cylinders	12 Cylinders

Attribute Type	Attribute Name	Attribute Value	
Nominal	Paint_color	Grey	Silver
		Brown	White
		Red	Black
		Bule	Green
		Yellow	Orange
		Purple	Custom

Attribute Type	Attribute Name	Attribute Value	
Nominal	Transmission	Automatic	Manual

Attribute Type	Attribute Name	Attribute Value		
Nominal	Drive	FWD	4WD	RWD

Attribute Type	Attribute Name	Attribute Value [MIN, MAX]
Numeric	Year	[2000 , 2020]
Numeric	Odometer	[0 , 650000] (km)

◆ Class

$1,000 ~ 650,000까지의 linear한 가격 데이터를 가진 Numeric Class를
5가지의 범주로 분류한 Nominal Class

Category 1	Category 2	Category 3	Category 4	Category 5
$0 ~ $8,000	$8,000 ~ $16,000	$16,000 ~ $30,000	$30,000 ~ $40,000	Over $40,000

2. 데이터 전처리 진행과정 (Preprocessing)

```java
import java.io.BufferedReader;

public class csvRandomExtract {

    static List<List<String>> allData = new ArrayList<List<String>>(); // csv원본을 저장할 리스트
    static List<List<String>> writeData = new ArrayList<List<String>>(); // 출력할 내용을 담은 리스트

    static void randomWrite(int startSize, int finishSize, int startRecord, int finishRecord) {
        // random 배열 생성
        // record : 입력파일의 레코드 행 번호
        int size = finishSize - startSize;
        int arrayR[] = new int[size];
        Random r = new Random();
        for (int i = 0; i < size; i++) {
            arrayR[i] = r.nextInt(finishRecord-startRecord) + startRecord;
            for (int j = 0; j < i; j++) {
                if (arrayR[i] == arrayR[j])
                    i--;
            }
        }
        Arrays.sort(arrayR);

        // writeData 채우기
        List<String> Line = new ArrayList<String>();
        int j = 0;
        for(int i = startRecord; i<finishRecord; i++) {
            if(i == arrayR[j]){
                Line = allData.get(i);
                List<String> writeLine = new ArrayList<String>();
                for (String data : Line) {
                    writeLine.add(data);
                }
                writeData.add(startSize+j, writeLine);
                j++;
            }
            if (j == size) break;
        }
    }


    public static void main(String[] args) {

        BufferedReader br = null;

        //원하는 만큼 수정
        int inputSize = 3001;   //입력파일의 전체 행
        int classIndex = 12;
        int classNum = 5;
        int OutputSize = 300;   //출력할 인스턴스 수
        String className = "price_class";
        int splitSize = OutputSize / classNum ;   //각 클래스마다 출력할 인스턴스 수


        List<Integer> classSplit = new ArrayList<Integer>(); //클래스 값이 달라지는 행
```

```
68          // load CSV
69          try {
70              br = Files.newBufferedReader(
71                      Paths.get("train3000.csv"));
72              Charset.forName("UTF-8");
73              String line = "";
74
75              int i =0;
76
77              // CSV파일의 매 행을 allData에 저장. 클래스 값이 달라지면 classSplit에 행 값 저장.
78              while ((line = br.readLine()) != null) {
79                  i++;
80                  List<String> tmpList = new ArrayList<String>();
81                  String array[] = line.split(",");
82                  tmpList = Arrays.asList(array);
83                  if(!className.equals(tmpList.get(classIndex))) {
84                      className = tmpList.get(classIndex);
85                      classSplit.add(i);
86                  }
87                  //System.out.println(tmpList);
88                  allData.add(tmpList);
89              }
90              classSplit.add(inputSize+1); // 마지막행+1까지 저장.
91          } catch (FileNotFoundException e) {
92              e.printStackTrace();
93          } catch (IOException e) {
94              e.printStackTrace();
95          } finally {
96              try {
97                  if (br != null)
98                      br.close();
99              } catch (IOException e) {
100                 e.printStackTrace();
101             }
102         }
103
104         // 실행
105         for(int i = 0; i<classNum; i++) {
106             randomWrite(splitSize*i, splitSize*(i+1), classSplit.get(i), classSplit.get(i+1)-1);
107         }
108
109
110         // write CSV
```

상단의 그림은 Instance를 추출하여 새로운 데이터 셋을 만드는 과정이 담긴 Java 실행 코드이다. 세부 과정은 다음과 같다. (Instance 3000개 기준)

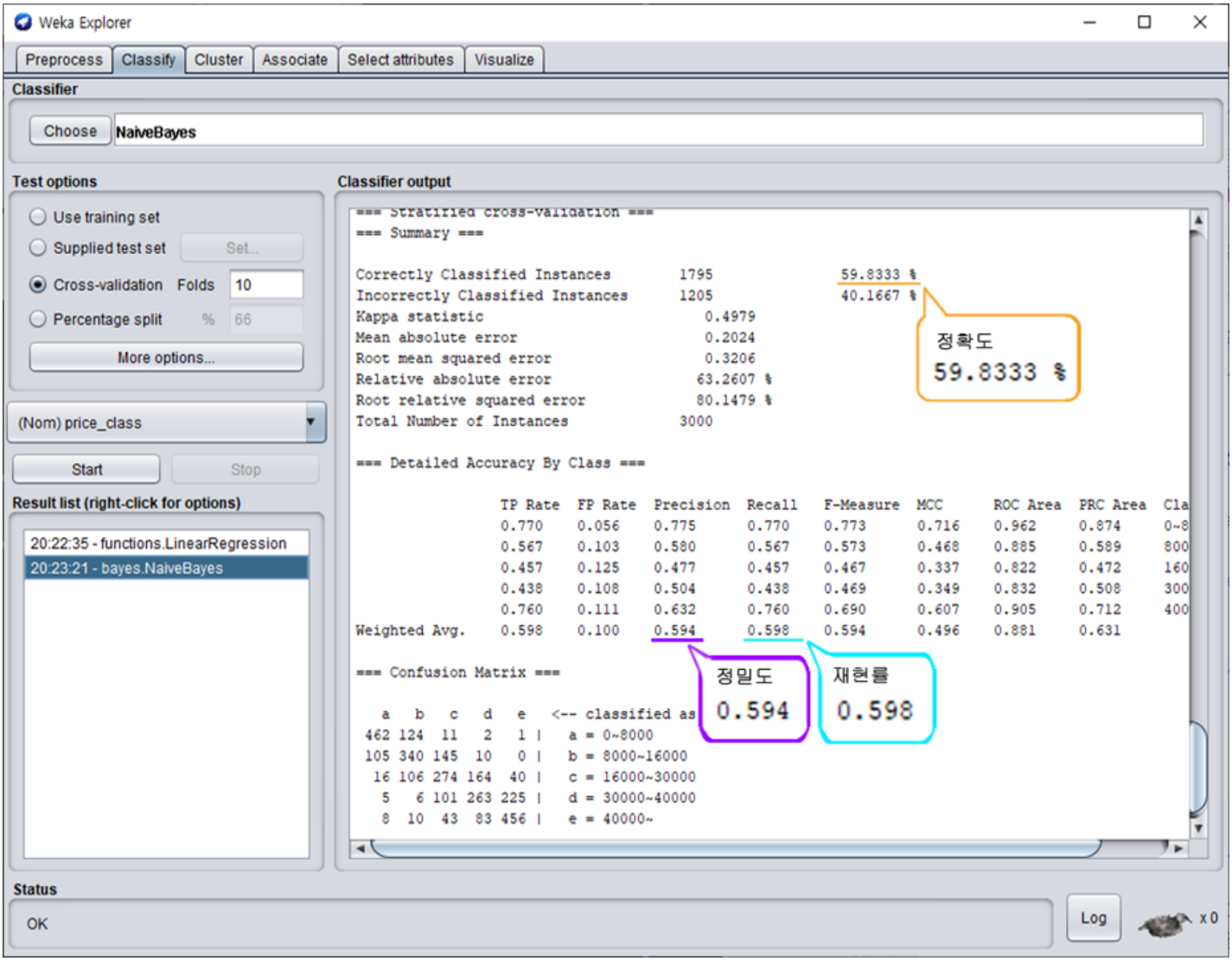
① 원본 데이터 셋 로드
② 원본 데이터 셋에서 결측 값(NULL)을 가진 Instance 삭제
③ 5가지 범주로 분류한 Nominal Class를 기준으로, 각 범주마다 Instance를 600개씩 추출
④ 추출한 총 3000개의 Instance로 새로운 데이터 셋을 만들어 저장

300~2700개의 Instance를 가진 데이터 셋을 만들 때도 같은 코드로 진행했다.

3. 프로젝트에 사용한 알고리즘 (Algorithm)

3.1 Naive Bayes

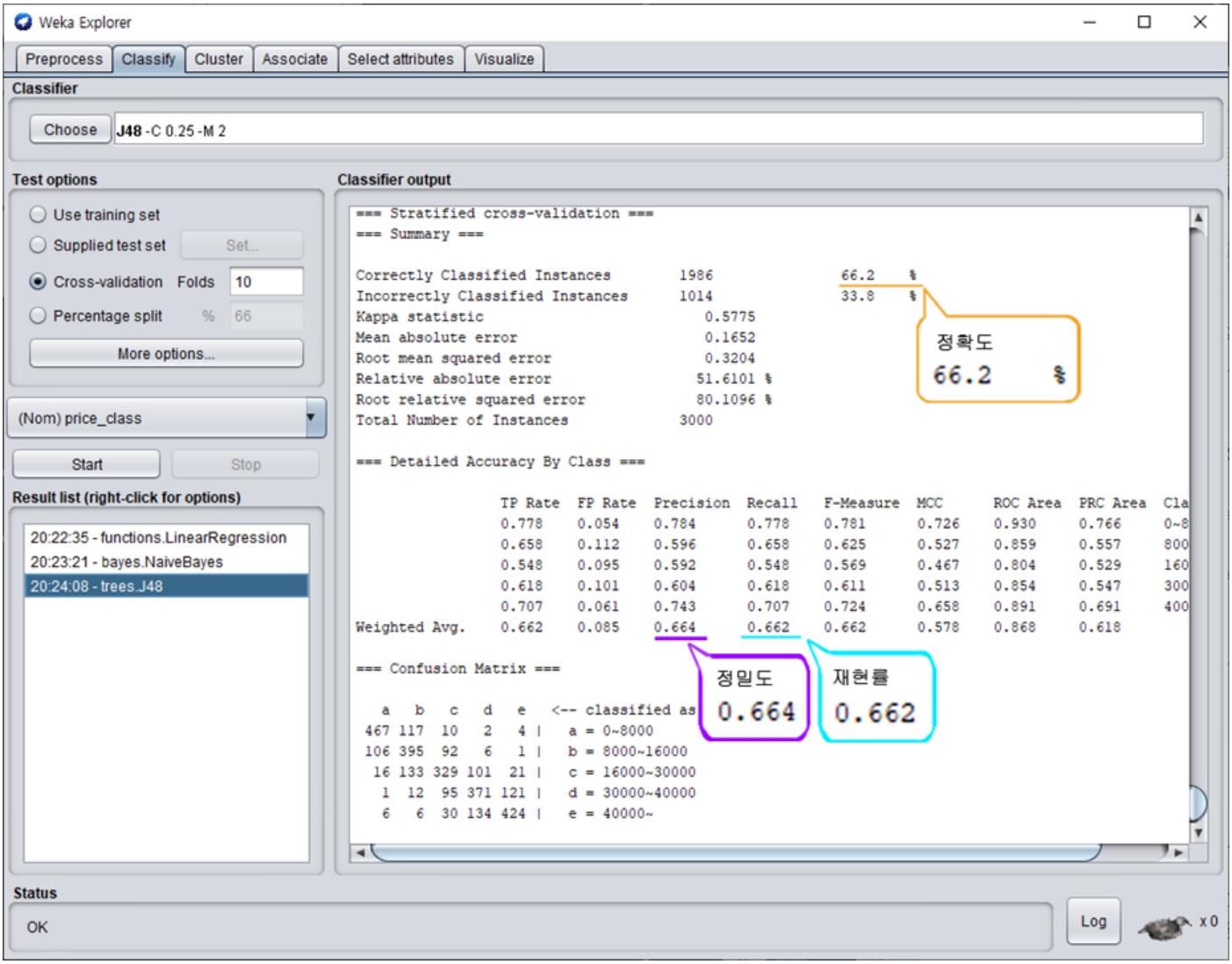
Naive Bayes는 Bayes 확률을 이용한 분류 모델이다. 데이터의 모든 속성들을 사용하는데, 이때, 각각의 속성들은 동등하게 중요하고 서로 독립적이다. 각각의 사건들이 발생할 때, 해당 클래스일 확률을 구하는 것이다. 가장 높은 확률의 클래스를 선택하는 방법이다.



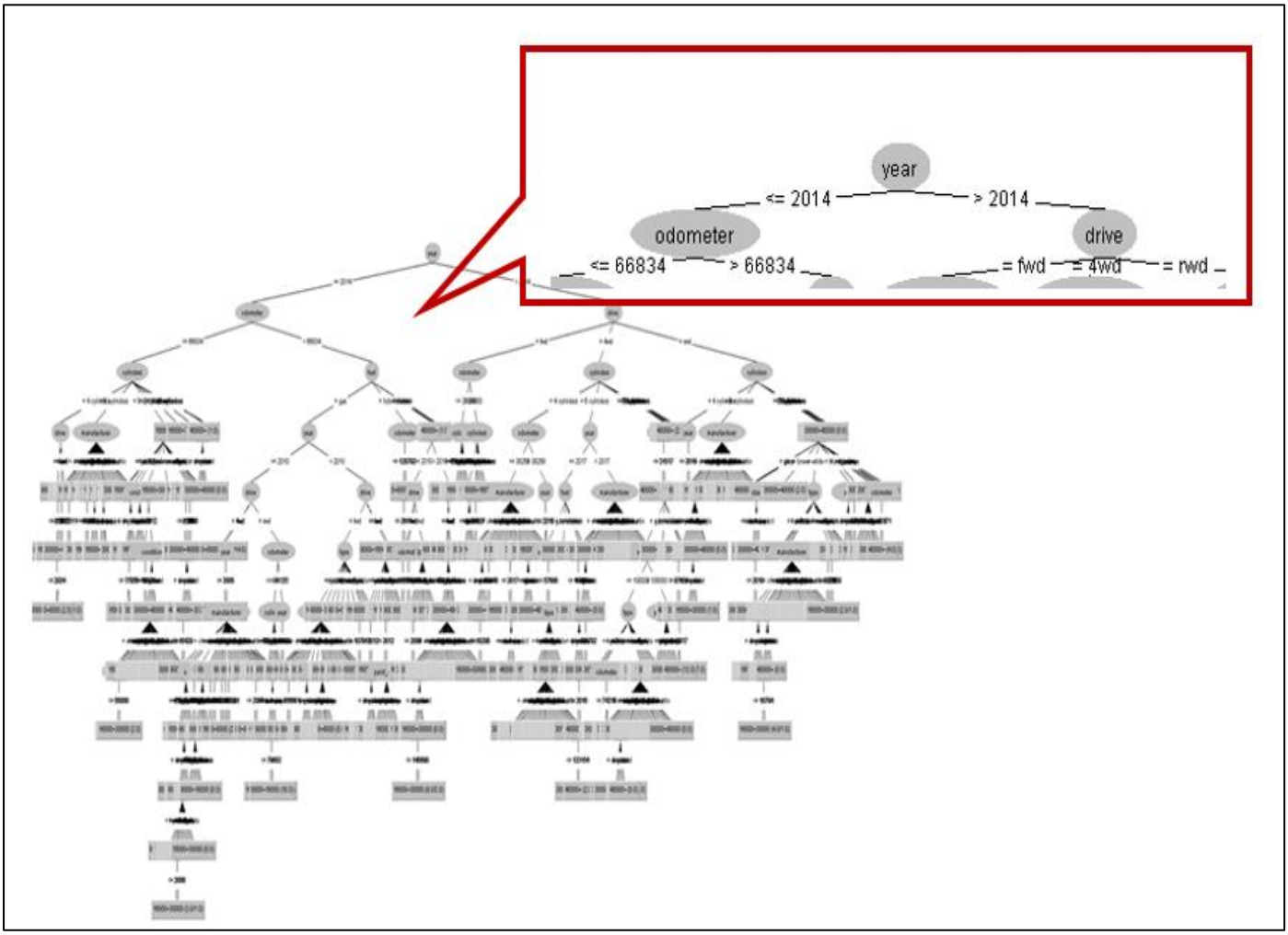
상단의 사진은 Naive Bayes로 학습시킨 모델의 결과이다.
정확도는 59.8%가 나왔고, 정밀도는 59.4%, 재현율은 59.8%가 나왔다.

3.2 J48 (Decision Tree)

Decision Tree는 결과와 속성들의 엔트로피를 이용한 Gain값 계산 과정을 통해, Gain 값이 가장 큰 속성을 정보 분류 능력이 가장 뛰어난 속성으로 결정한다. 그 후 해당 속성을 루트로 정하고 그것을 토대로 Instance들을 분류한다. 해당 과정은 모든 Instance가 완전히 분류될 때까지 반복된다,



상단의 사진은 J48 (Decision Tree)로 학습시킨 모델의 결과이다.
정확도는 66.2%가 나왔으며, 정밀도는 66.4%, 재현율은 66.2%로 결과 값이 도출되었다.



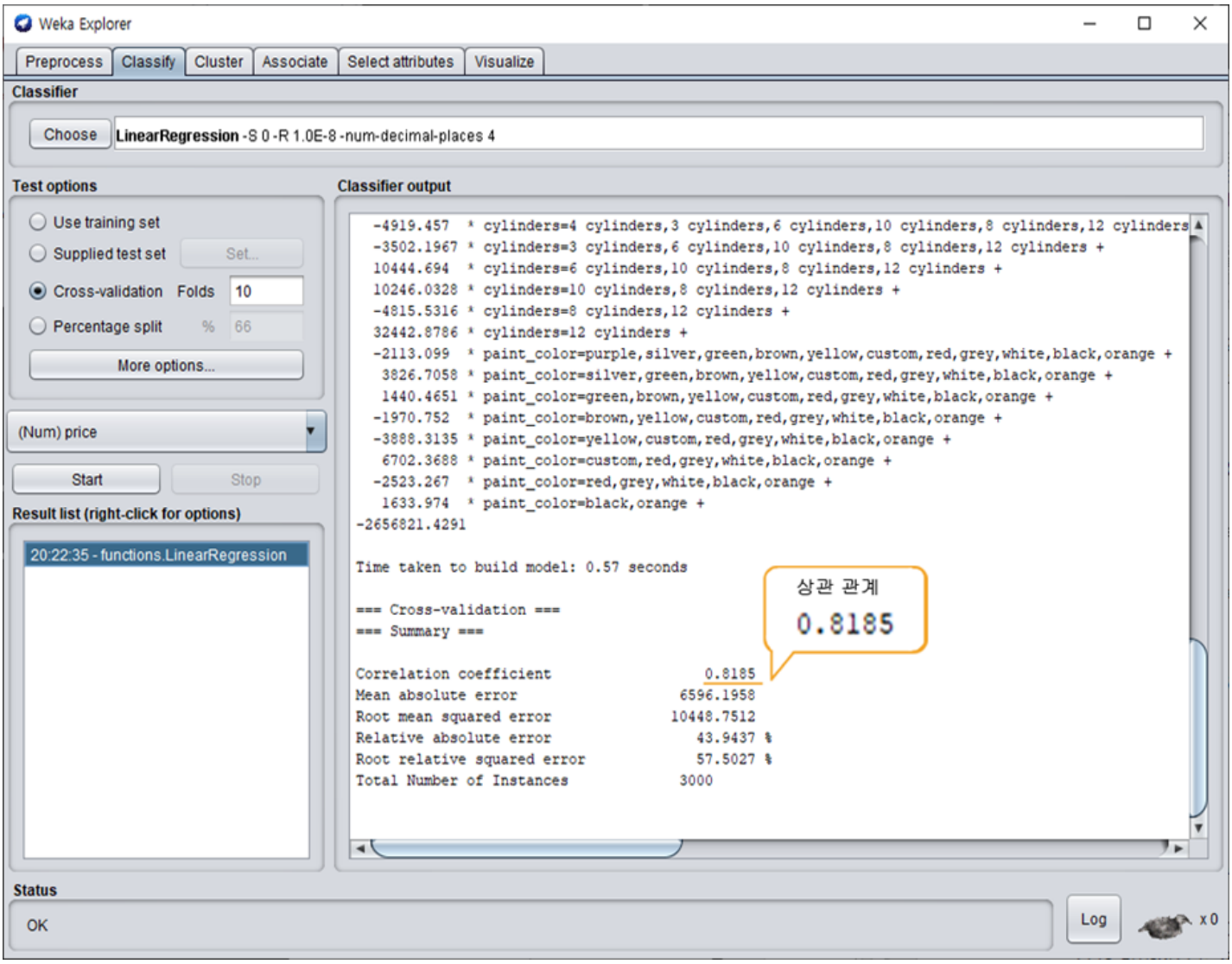
상단의 사진은 J48 (Decision Tree)의 Visual Tree이다.

위의 트리를 보면 이 모델에서 가장 정보 분류 능력이 뛰어난 속성은 'year'이고,
year <= 2014 일 경우 그 다음으로 정보 분류 능력이 뛰어난 속성은 odometer,
year > 2014 일 경우 그 다음으로 정보 분류 능력이 뛰어난 속성은 drive인 것을 알 수 있다.

3.3 Linear Regression (선형회귀)

Attribute N개에 대해서, 'N차원에 존재하는 모든 Instance (점)들과의 거리의 합'
을 최소로 하는 직선을 찾는 알고리즘이다. 각 분기마다 Attribute별로 가중치를 Adaptive하게 주어, 분기를 진행할수록 가장 적합한 직선을 찾는다.

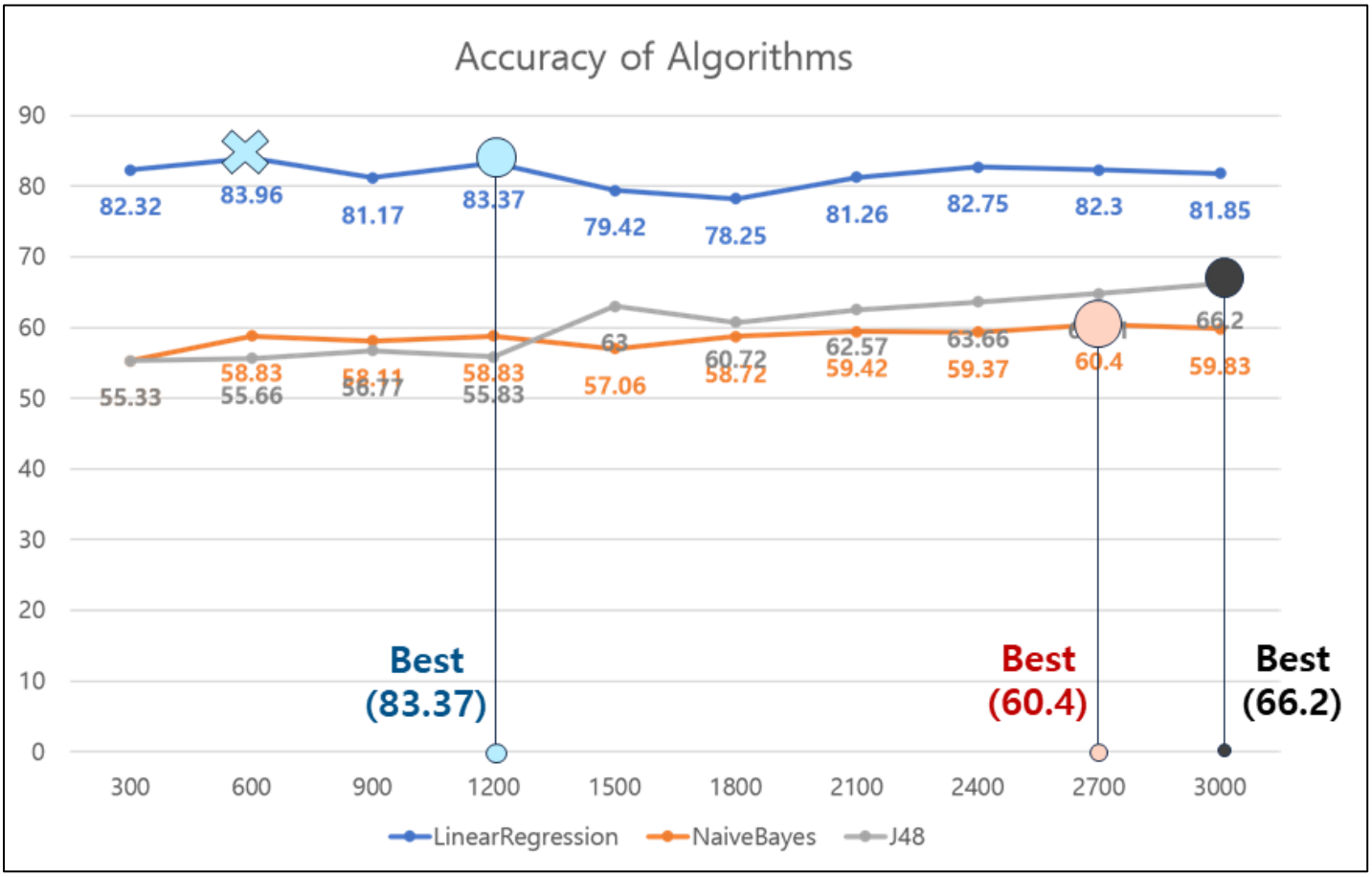
프로젝트의 모델의 클래스는 Numeric이기에 회귀알고리즘이 필요하여 채택하였다.



상단의 사진은 Linear Regression을 바탕으로 분석한 결과이다.
Correlation coefficient (상관관계)가 0.8185로, +1에 가깝게 측정되었다.

4. 성능 분석 (Performance Analyzing)

4.1 학습곡선 (Learning Curve)



상단의 사진은 각 알고리즘 별로 Instance를 개수를 작게 하여 300개 단위로 구간을 나누어 정확도를 측정하고 그 측정된 정확도를 토대로 그래프로 나타낸 결과이다.
또한 각 알고리즘 별로 Best라고 생각되는 지점을 사진에 표시하였다.

◆ Linear Regression (83.37%) :

Instance 수가 600일 때 베스트 퍼포먼스를 보였는데, 600개는 학습 데이터의 양이 부족하다고 판단하였고, 신뢰할 만한 수치로 그 다음 정확도를 보여주는 1,200개를 베스트로 정했다.

◆ Naive Bayes (60.4%) :

Instance 수가 2,700개일 때 가장 높은 정확도를 보여준다.

◆ J48 (66.2%) :

Instance 수가 23,000개일 때 가장 높은 정확도를 보여준다.

4.2 측정한 성취도의 예측 범위 계산 (Bernoulli Process)

베르누이 분포는 통계학에서 주로 사용되는 이론 중 하나다.
이항분포에 대해 독립 시행을 하는데, 충분히 큰 인스턴스 개수 N에 대해 정규분포를 이룬다. 아래 식을 계산하면 포준 정규 분포에서의 범위 P를 구할 수 있다.

$$P = (f + \frac{Z^2}{2N} \pm Z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{Z^2}{4N^2}}) \ / \ (1+\frac{z^2}{N})$$

f는 측정값이고, N은 인스턴스의 개수이다. z는 confidence에 따라서 정해져 있는 critical point이다.
N이 작을수록, confidence가 클수록 P의 범위가 커진다.

Naïve Bayes			
N	Accuracy	c = 80%	c = 90%
300	0.5833	[0.546, 0.619]	[0.535, 0.631]
600	0.5883	[0.562, 0.613]	[0.554, 0.621]
900	0.5811	[0.559, 0.602]	[0.553, 0.608]
1200	0.5883	[0.569, 0.606]	[0.564, 0.611]
1500	0.5706	[0.554, 0.586]	[0.549, 0.591]
1800	0.5872	[0.572, 0.601]	[0.567, 0.606]
2100	0.5942	[0.580, 0.607]	[0.576, 0.612]
2400	0.5937	[0.580, 0.606]	[0.577, 0.610]
2700	0.604	[0.591, 0.615]	[0.588, 0.619]
3000	0.5983	[0.586, 0.609]	[0.583, 0.613]

J48			
N	Accuracy	c = 80%	c = 90%
300	0.4066	[0.370, 0.443]	[0.360, 0.455]
600	0.5566	[0.530, 0.582]	[0.523, 0.590]
900	0.5677	[0.546, 0.588]	[0.540, 0.595]
1200	0.5583	[0.539, 0.576]	[0.534, 0.582]
1500	0.63	[0.613, 0.645]	[0.609, 0.650]
1800	0.6072	[0.592, 0.621]	[0.588, 0.626]
2100	0.6257	[0.612, 0.639]	[0.608, 0.643]
2400	0.6366	[0.623, 0.649]	[0.620, 0.652]
2700	0.6481	[0.636, 0.659]	[0.632, 0.663]
3000	0.662	[0.650, 0.672]	[0.647, 0.676]

Linear Regression			
N	Accuracy	c = 80%	c = 90%
300	0.7499	[0.716, 0.780]	[0.706, 0.791]
600	0.8396	[0.819, 0.857]	[0.813, 0.864]
900	0.8117	[0.794, 0.827]	[0.789, 0.833]
1200	0.8337	[0.819, 0.847]	[0.815, 0.851]
1500	0.7942	[0.780, 0.807]	[0.776, 0.811]
1800	0.7825	[0.769, 0.794]	[0.766, 0.798]
2100	0.8126	[0.801, 0.823]	[0.798, 0.826]
2400	0.8275	[0.817, 0.837]	[0.814, 0.840]
2700	0.823	[0.813, 0.832]	[0.810, 0.835]
3000	0.8185	[0.809, 0.827]	[0.806, 0.830]

상단의 사진은 프로젝트 모델에 대해 시행한 Bernoulli Process의 결과이다.
confidence가 80%일 때, z값은 1.282,
confidence가 90%일 때, z값은 1.645이다.

각각의 인스턴스 개수에서 각각 다른 측정값이 나오게 되는데,
이때 위의 z 값을 이용하여 정규 분포의 범위를 계산할 수 있다.

◆ Naive Bayes

Naive Bayes에서는 인스턴스 개수가 2700개 일 때 측정값이 60.4%로 가장 높았다.

confidence 80%일 때, 정규 분포의 범위는 59.1% ~ 61.5% 라는 것을 알 수 있다.
confidence 90%일 때, 정규 분포의 범위는 58.8% ~ 61.9% 라는 것을 알 수 있다.

◆ J48

J48에서는 인스턴스 개수가 3000개 일 때 측정값이 66.2%로 가장 높았다.

confidence 80%일 때, 정규 분포의 범위는 65.0% ~ 67.2% 라는 것을 알 수 있다.
confidence 90%일 때, 정규 분포의 범위는 64.7% ~ 67.6% 라는 것을 알 수 있다.

◆ Linear Regression

Linear Regression 에서는 인스턴스 개수가 900개 일 때 83.9%로 가장 높았다.
하지만 Instance의 개수가 적어 Overfitting으로 판단, 그 다음으로 높은 83.3%인 경우를 선택했다.

confidence 80%일 때, 정규 분포의 범위는 81.9% ~ 84.7% 라는 것을 알 수 있다.
confidence 90%일 때, 정규 분포의 범위는 81.5% ~ 85.1% 라는 것을 알 수 있다.

4.3 기계학습 알고리즘의 성능 비교 (분산분석: ANOVA)

◈ Best일 때를 기준으로 ±100 범위 내에서 10번 재측정한 결과

Linear Regression	Naïve Bayes	J48
84.11	62.23	64.70
83.31	61.21	65.42
83.20	60.34	64.90
82.34	60.07	65.42
84.05	60.92	65.35
83.37	60.40	66.20
83.35	60.47	65.46
79.16	60.43	65.72
78.53	60.5	65.45
83.46	60.64	65.19

요약표

인자의 수준	관측수	합	평균	분산
Linear Regression	10	824.88	82.488	3.942484
Naïve Bayes	10	607.21	60.721	0.38121
J48	10	653.81	65.381	0.170254

분산 분석

◈ 분산 분석 : 일원 배치법

변동의 요인	제곱합	자유도	제곱 평균	F 비	P-값	F 기각치
처리	2627.224	2	1313.612	876.9207	2.75E-25	3.354131
잔차	40.44554	27	1.497983			
계	2667.67	29				

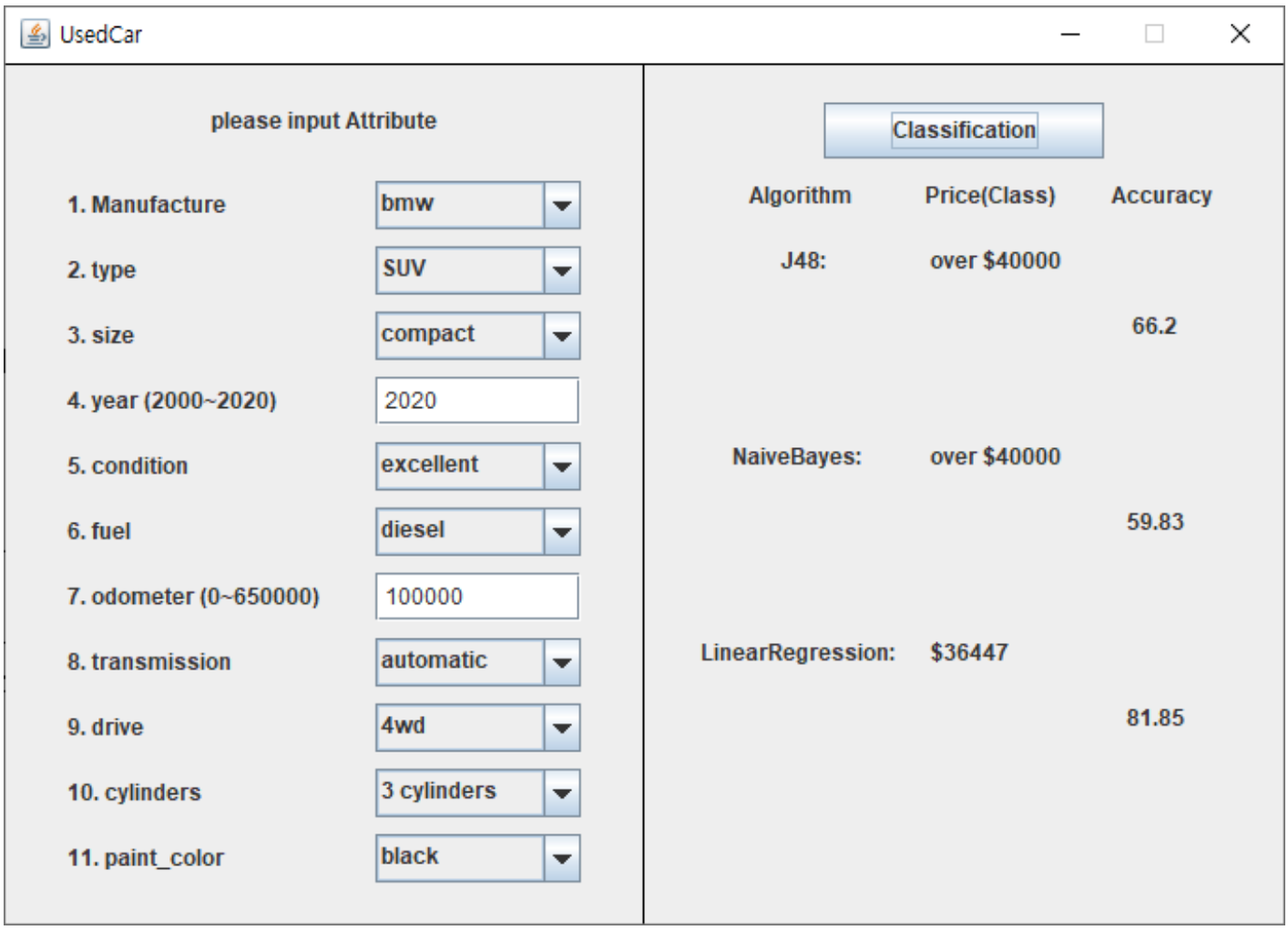
분산분석은 여러 집단의 평균과 분산의 차이에 의해 만들어진 F분포를 이용, 가설을 검정하는 방법이다. 학습에 사용된 3가지의 알고리즘을 통해 분석한 결과가 우연이 아닌 통계적으로 유의미한지 검증하기 위해 분산분석을 진행했다.

엑셀의 데이터분석 기능인 분산분석:일원 배치법을 사용하였고, 유의도는 0.05로 설정하였다.

결과로 계산된 F비 값이 876.92
같은 데이터의 종류와 수에 해당하는 F기각치는 3.35
876.92 > 3.35로, Null Hypothesis (귀무가설)를 기각할 수 있다.
따라서 분석내용은 통계적으로 유의미하다고 결론지었다.

5. 데모 프로그램 및 개선점

5.1 데모 프로그램



* 개발환경: JDK14

개발한 모델을 편하게 시연해볼 수 있도록 간단한 GUI로 구성한 프로그램이다.
좌측에서 11가지의 Attribute를 설정하면, 우측에서 예측된 가격 및 각 알고리즘별 정확도를 표시한다.

5.2 개선점

Multi-Layer Perceptron이나 Random Forest 알고리즘을 사용하여 데이터 분석을 진행해 보았으나 Instance 수가 3000개로 제한되어 정확도가 낮게 나온 것 같다.

Instance 개수를 늘린다면 좀 더 높은 정확도를 얻을 수 있을 것이라고 생각한다.
또한 지역에 대한 충분한 데이터가 있다면 모델을 더 개선 시킬 수 있을 것 같다.

6. 프로젝트 업무 분담 및 일정

6.1 프로젝트 업무 분담

박재근 : 데이터 학습, 성능분석, 데모 프로그램 제작
안현진 : 데이터 전처리, 성능분석, 데모 프로그램 및 GUI 제작.
김 순 : 데이터 수집 및 분석 내용 정리 PPT/보고서 제작

6.2 프로젝트 일정

팀 구성 및 역할 분담	(11.03 ~ 11.10)
데이터 수집	(11.10 ~ 11.17)
데이터 전처리	(11.17 ~ 11.20)
데이터 학습	(11.20 ~ 11.24)
성능 분석	(11.25 ~ 11.30)
데모 프로그램 및 GUI 제작	(11.25 ~ 12.02)
최종 보고서 작성	(12.03 ~ 12.10)

7. 참고 자료 및 출처

◆ 데이터셋 원본

https://www.kaggle.com/austinreese/craigslist-carstrucks-data

◆ Weka Library in Java

https://weka.sourceforge.io/doc.stable/

◆ 중고차 시장, 코로나19 여파로 오히려 호황…그 이유는?

http://www.dailycar.co.kr/content/news.html?type=view&autoId=37743 (데일리카)

◆ AJ셀카 설문조사

http://www.naeil.com/news_view/?id_art=355656 (내일신문)