

Fast Campus 157|

# INSURANCE FRAUD DETECTION

2조 김준성 방희란 정해주

## 0. 쏘카 소개

쏘카 소개

쏘카,  
누구나  
들어가고 싶은  
꿈의 직장^^

급격하게 성장중인 쏘카

## 카셰어링 1위

약 1만 2천대의 차량 운영  
누적 580만 회원 사용

6.5M 앱 다운로드 수

5.8M 누적 유저 수

200K 월간 사용자 수(MAU)

45% 브랜드 최초 상기도



# 1. 문제 정의

문제 정의

보험사기로  
골머리를  
앓는다는데..

보험사기에 멍드는 카셰어링

“ㄷㅋ구함”

\* 뒷쿵

보험사기에 멍드는 카셰어링

문제 정의

**보험사기를  
근절하기 위해  
각분야전문가가  
모였다!!**

전문가 3인 소개

---



문제 정의

보험사기를  
근절하기 위해  
각분야전문가가  
모였다!!

전문가 3인 소개



튜닝의 신!  
**제이 킴**

그의 손만 거치면,  
죽은 모델도  
살려낸다는데!!

판다스가 낳은 괴물!  
**희란 팡**

EDA는  
내가  
책임진다!!

구글링 마스터!  
**엠버 정**

내게  
불가능이란  
없다!!

대장님♥  
**핑크윙크**

마지막 앞새는  
끝까지 떨어지지  
않았다!!

## 2. 데이터 탐색



## 데이터 개요

## Fraud 검출 분류 데이터

25 columns

16000 rows

	fraud_YN	car_model	sharing_type	age_group	has_previous_accident		police_site_aid_YN	total_prsn_cnt	test_set
15995	0	2	0	2	0	...	0	-1	0
15996	0	2	0	2	1		0	-1	0
15997	0	2	1	2	0		0	-1	1
15998	0	2	0	2	0		0	-1	0
15999	0	2	0	2	0		0	-1	0

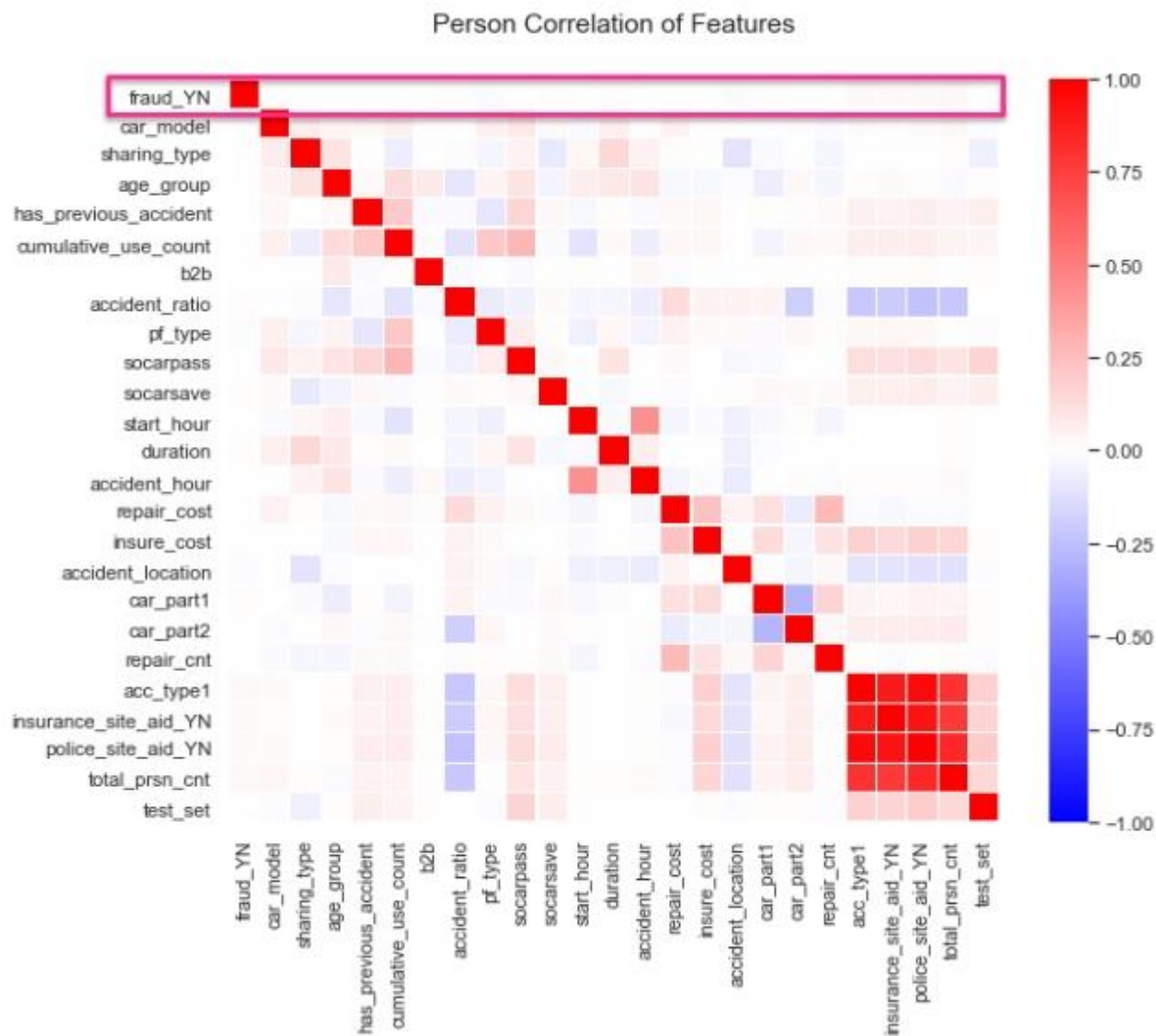
- fraud\_YN 컬럼의 불균형이 극심하여 전체 데이터 中 fraud 0.26%

fraud가 총 41 건 中 train 셋에는 34 건, test 셋에는 7 건

- fraud\_YN 컬럼이 우리가 예측하고자 하는 라벨
- 미기록으로 기입된 결측치 데이터 多

## Column 소개

- |                         |                     |                         |
|-------------------------|---------------------|-------------------------|
| • fraud_YN              | • socarpass         | • car_part1             |
| • car_model             | • socarsave         | • car_part2             |
| • sharing_type          | • start_hour        | • repair_cnt            |
| • age_group             | • duration          | • acc_type1             |
| • has_previous_accident | • accident_hour     | • insurance_site_aid_YN |
| • cumulative_use_count  | • repair_cost       | • police_site_aid_YN    |
| • b2b                   | • insure_cost       | • total_prsn_cnt        |
| • accident_ratio        | • accident_location | • test_set              |
| • pf_type               |                     |                         |



### 3. 평가 기준

Binary  
Classification

사기 유무 예측

극심한  
데이터 불균형

fraud의 비율  
전체 데이터 대비

**0.26% !!**

데이터  
부족

fraud 총 41 건  
test 셋에는 7 건

평가기준

accuracy만으로  
성능을평가하기어렵다

평가항목 정하기

15,595 건 (99.74%)

No Fraud

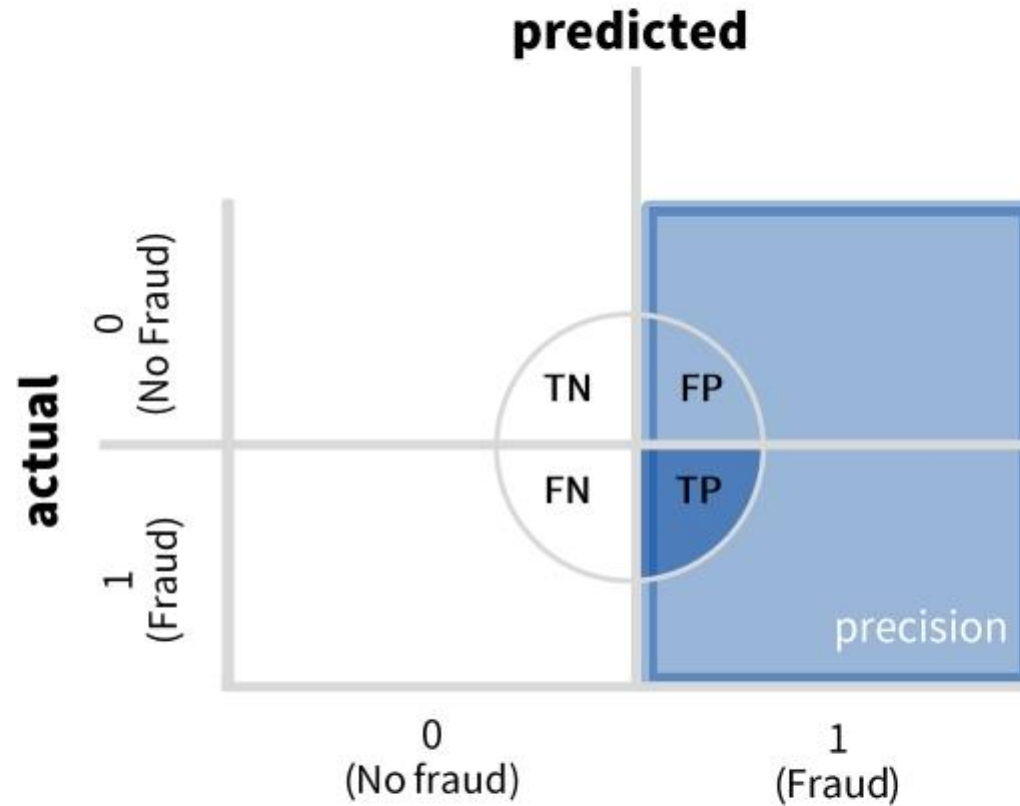
Fraud

전부 0-Class (No Fraud)로 예측



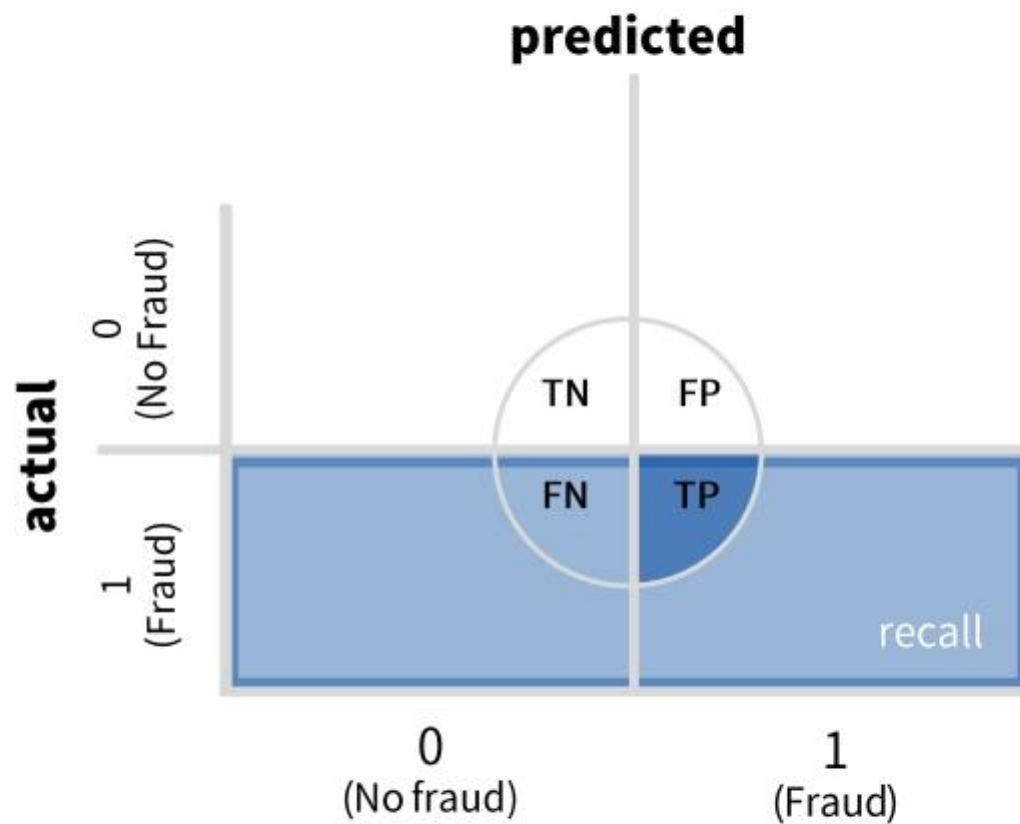
**accuracy = 0.99 !!**

쏘카 고객이라면  
precision이  
중요하지 않을까?



**precision ↑** ▶ **오판위험율 ↓** ▶ **고객불만 ↓**

쏘카관계자라면  
recall이 중요하지  
않을까?



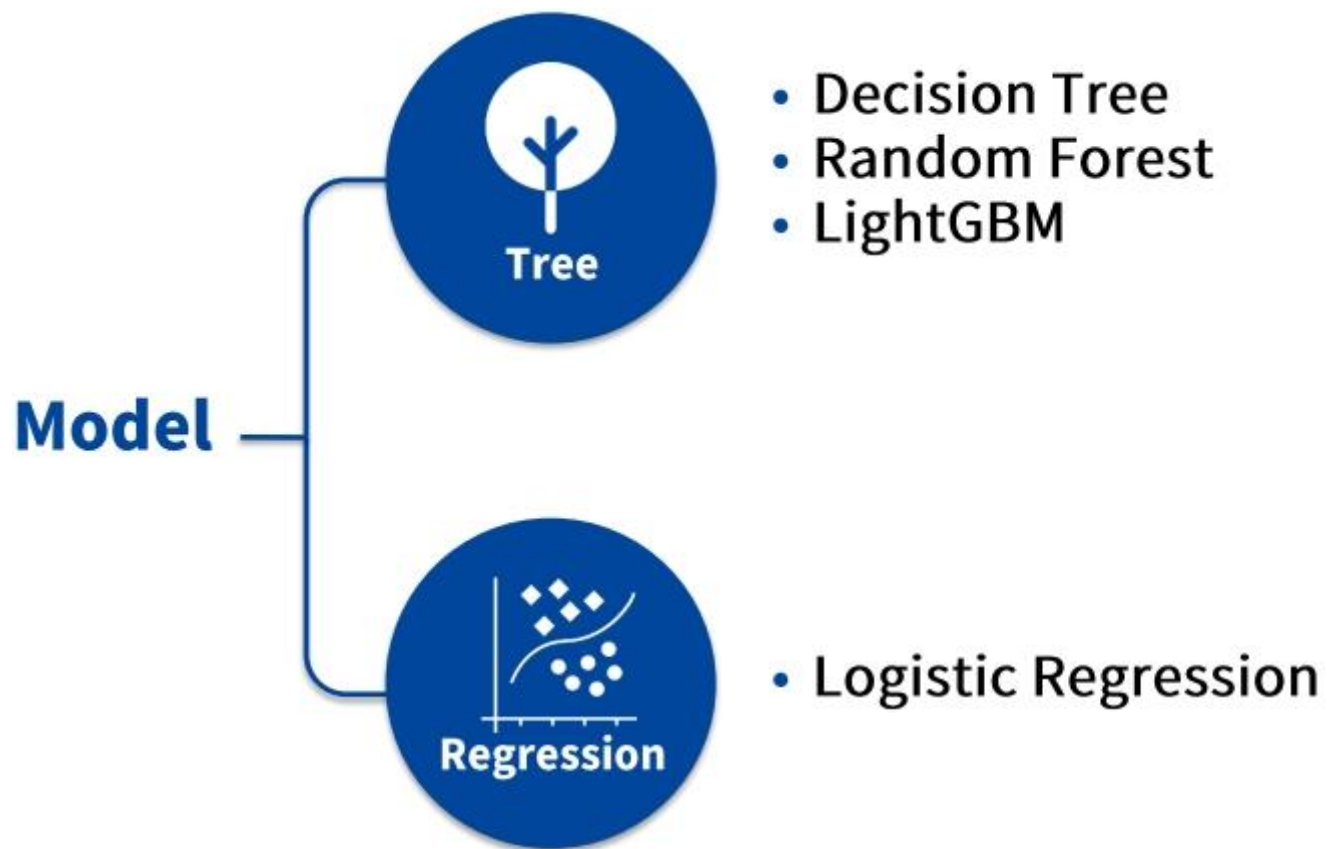
recall ↑ ▶ 사기적발 ↑ ▶ 손해 ↓



## 우리의 Baseline

	accuracy	precision	recall
Support Vector Machine	0.492791	0.003153	0.714286

Source : 14 기



평가기준

## 일단 모델을 돌려보자

전처리 전 과적합이 매우 심하다

### train

	accuracy	precision	recall
LogisticRegression	0.99736	0.000000	0.000000
DecisionTree	1.000000	1.000000	1.000000
RandomForest	1.000000	1.000000	1.000000
LightGBM	1.000000	1.000000	1.000000

avg 0.74

### test

	accuracy	precision	recall
LogisticRegression	0.997757	0.000000	0.000000
DecisionTree	0.990388	0.000000	0.000000
RandomForest	0.997757	0.000000	0.000000
LightGBM	0.997757	0.000000	0.000000

avg 0.00

## 4. 데이터 전처리

1

## Remove Data

---

- 결측치 처리
- 데이터 불균형 개선

2

## Data Handling

---

- Nominal Data 를  
One-Hot Encoding
- Ordinal Data 를 재분류

3

## Scaling

---

- 이상치 제어

4

## Oversampling

---

- 데이터 불균형 해소

## 우선데이터일부를 없애보기도하고

### Remove Data

#### 결측치 처리

미기록 데이터가 다수 포함된 컬럼

- insurance\_site\_aid\_YN
- police\_site\_aid\_YN
- total\_prsn\_cnt

insurance_site_aid_YN	police_site_aid_YN	total_prsn_cnt
0	0	-1
0	0	-1
0	0	-1
0	0	-1
0	0	-1
0	0	-1
0	0	-1
0	0	-1
0	0	-1

drop 3 columns

#### 데이터 불균형 개선

No Fraud에만 발생하는 데이터를 제거하여  
데이터 불균형을 개선

- accident\_hour (-1: 미기록)
- accident\_location(3:고속도로, 5: 확인불가)
- car\_model(4: 수입차, 5: 전기차)
- b2b(2:법인고객)
- duration(5:1시간 이하)
- accident\_ratio(0, 40, 100% 제외후 모두)
- Repair\_cnt(1, 2, 3 제외후 모두)

```
df[df['accident_hour']==-1]
df[df['accident_location'].isin([3,5])]
df[df['car_model'].isin([4,5])]
df[df['b2b']==2]
df[df['duration']==5]
df[df['accident_ratio'].isin([0,40,100,20,30,40,50,60,70,80,90,100])]
df[df['repair_cnt'].isin([0,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,524,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,598,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,623,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,672,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,697,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,993,994,995,996,997,998,999,1000,1001,1002,1003,1004,1005,1006,1007,1008,1009,1010,1011,1012,1013,1014,1015,1016,1017,1018,1019,1020,1021,1022,1023,1024,1025,1026,1027,1028,1029,1030,1031,1032,1033,1034,1035,1036,1037,1038,1039,1040,1041,1042,1043,1044,1045,1046,1047,1048,1049,1050,1051,1052,1053,1054,1055,1056,1057,1058,1059,1060,1061,1062,1063,1064,1065,1066,1067,1068,1069,1070,1071,1072,1073,1074,1075,1076,1077,1078,1079,1080,1081,1082,1083,1084,1085,1086,1087,1088,1089,1090,1091,1092,1093,1094,1095,1096,1097,1098,1099,1100,1101,1102,1103,1104,1105,1106,1107,1108,1109,1110,1111,1112,1113,1114,1115,1116,1117,1118,1119,1120,1121,1122,1123,1124,1125,1126,1127,1128,1129,1130,1131,1132,1133,1134,1135,1136,1137,1138,1139,1140,1141,1142,1143,1144,1145,1146,1147,1148,1149,1150,1151,1152,1153,1154,1155,1156,1157,1158,1159,1160,1161,1162,1163,1164,1165,1166,1167,1168,1169,1170,1171,1172,1173,1174,1175,1176,1177,1178,1179,1180,1181,1182,1183,1184,1185,1186,1187,1188,1189,1190,1191,1192,1193,1194,1195,1196,1197,1198,1199,1200,1201,1202,1203,1204,1205,1206,1207,1208,1209,1210,1211,1212,1213,1214,1215,1216,1217,1218,1219,1220,1221,1222,1223,1224,1225,1226,1227,1228,1229,1230,1231,1232,1233,1234,1235,1236,1237,1238,1239,1240,1241,1242,1243,1244,1245,1246,1247,1248,1249,1250,1251,1252,1253,1254,1255,1256,1257,1258,1259,1260,1261,1262,1263,1264,1265,1266,1267,1268,1269,1270,1271,1272,1273,1274,1275,1276,1277,1278,1279,1280,1281,1282,1283,1284,1285,1286,1287,1288,1289,1290,1291,1292,1293,1294,1295,1296,1297,1298,1299,1300,1301,1302,1303,1304,1305,1306,1307,1308,1309,1310,1311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321,1322,1323,1324,1325,1326,1327,1328,1329,1330,1331,1332,1333,1334,1335,1336,1337,1338,1339,1340,1341,1342,1343,1344,1345,1346,1347,1348,1349,1350,1351,1352,1353,1354,1355,1356,1357,1358,1359,1360,1361,1362,1363,1364,1365,1366,1367,1368,1369,1370,1371,1372,1373,1374,1375,1376,1377,1378,1379,1380,1381,1382,1383,1384,1385,1386,1387,1388,1389,1390,1391,1392,1393,1394,1395,1396,1397,1398,1399,1400,1401,1402,1403,1404,1405,1406,1407,1408,1409,1410,1411,1412,1413,1414,1415,1416,1417,1418,1419,1420,1421,1422,1423,1424,1425,1426,1427,1428,1429,1430,1431,1432,1433,1434,1435,1436,1437,1438,1439,1440,1441,1442,1443,1444,1445,1446,1447,1448,1449,1450,1451,1452,1453,1454,1455,1456,1457,1458,1459,1460,1461,1462,1463,1464,1465,1466,1467,1468,1469,1470,1471,1472,1473,1474,1475,1476,1477,1478,1479,1480,1481,1482,1483,1484,1485,1486,1487,1488,1489,1490,1491,1492,1493,1494,1495,1496,1497,1498,1499,1500,1501,1502,1503,1504,1505,1506,1507,1508,1509,1510,1511,1512,1513,1514,1515,1516,1517,1518,1519,1520,1521,1522,1523,1524,1525,1526,1527,1528,1529,1530,1531,1532,1533,1534,1535,1536,1537,1538,1539,1540,1541,1542,1543,1544,1545,1546,1547,1548,1549,1550,1551,1552,1553,1554,1555,1556,1557,1558,1559,1560,1561,1562,1563,1564,1565,1566,1567,1568,1569,1570,1571,1572,1573,1574,1575,1576,1577,1578,1579,1580,1581,1582,1583,1584,1585,1586,1587,1588,1589,1590,1591,1592,1593,1594,1595,1596,1597,1598,1599,1600,1601,1602,1603,1604,1605,1606,1607,1608,1609,1610,1611,1612,1613,1614,1615,1616,1617,1618,1619,1620,1621,1622,1623,1624,1625,1626,1627,1628,1629,1630,1631,1632,1633,1634,1635,1636,1637,1638,1639,1640,1641,1642,1643,1644,1645,1646,1647,1648,1649,1650,1651,1652,1653,1654,1655,1656,1657,1658,1659,1660,1661,1662,1663,1664,1665,1666,1667,1668,1669,1670,1671,1672,1673,1674,1675,1676,1677,1678,1679,1680,1681,1682,1683,1684,1685,1686,1687,1688,1689,1690,1691,1692,1693,1694,1695,1696,1697,1698,1699,1700,1701,1702,1703,1704,1705,1706,1707,1708,1709,1710,1711,1712,1713,1714,1715,1716,1717,1718,1719,1720,1721,1722,1723,1724,1725,1726,1727,1728,1729,1730,1731,1732,1733,1734,1735,1736,1737,1738,1739,1740,1741,1742,1743,1744,1745,1746,1747,1748,1749,1750,1751,1752,1753,1754,1755,1756,1757,1758,1759,1760,1761,1762,1763,1764,1765,1766,1767,1768,1769,1770,1771,1772,1773,1774,1775,1776,1777,1778,1779,1780,1781,1782,1783,1784,1785,1786,1787,1788,1789,1790,1791,1792,1793,1794,1795,1796,1797,1798,1799,1800,1801,1802,1803,1804,1805,1806,1807,1808,1809,1810,1811,1812,1813,1814,1815,1816,1817,1818,1819,1820,1821,1822,1823,1824,1825,1826,1827,1828,1829,1830,1831,1832,1833,1834,1835,1836,1837,1838,1839,1840,1841,1842,1843,1844,1845,1846,1847,1848,1849,1850,1851,1852,1853,1854,1855,1856,1857,1858,1859,1860,1861,1862,1863,1864,1865,1866,1867,1868,1869,1870,1871,1872,1873,1874,1875,1876,1877,1878,1879,1880,1881,1882,1883,1884,1885,1886,1887,1888,1889,1890,1891,1892,1893,1894,1895,1896,1897,1898,1899,1900,1901,1902,1903,1904,1905,1906,1907,1908,1909,1910,1911,1912,1913,1914,1915,1916,1917,1918,1919,1920,1921,1922,1923,1924,1925,1926,1927,1928,1929,1930,1931,1932,1933,1934,1935,1936,1937,1938,1939,1940,1941,1942,1943,1944,1945,1946,1947,1948,1949,1950,1951,1952,1953,1954,1955,1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,2028,2029,2030,2031,2032,2033,2034,2035,2036,2037,2038,2039,2040,2041,2042,2043,2044,2045,2046,2047,2048,2049,2050,2051,2052,2053,2054,2055,2056,2057,2058,2059,2060,2061,2062,2063,2064,2065,2066,2067,2068,2069,2070,2071,2072,2073,2074,2075,2076,2077,2078,2079,2080,2081,2082,2083,2084,2085,2086,2087,2088,2089,2090,2091,2092,2093,2094,2095,2096,2097,2098,2099,2100,2101,2102,2103,2104,2105,2106,2107,2108,2109,2110,2111,2112,2113,2114,2115,2116,2117,2118,2119,2120,2121,2122,2123,2124,2125,2126,2127,2128,2129,2130,2131,2132,2133,2134,2135,2136,2137,2138,2139,2140,2141,2142,2143,2144,2145,2146,2147,2148,2149,2150,2151,2152,2153,2154,2155,2156,2157,2158,2159,2160,2161,2162,2163,2164,2165,2166,2167,2168,2169,2170,2171,2172,2173,2174,2175,2176,2177,2178,2179,2180,2181,2182,2183,2184,2185,2186,2187,2188,2189,2190,2191,2192,2193,2194,2195,2196,2197,2198,2199,2200,2201,2202,2203,2204,2205,2206,2207,2208,2209,2210,2211,2212,2213,2214,2215,2216,2217,2218,2219,2220,2221,2222,2223,2224,2225,2226,2227,2228,2229,2230,2231,2232,2233,2234,2235,2236,2237,2238,2239,2240,2241,2242,2243,2244,2245,2246,2247,2248,2249,2250,2251,2252,2253,2254,2255,2256,2257,2258,2259,2260,2261,2262,2263,2264,2265,2266,2267,2268,2269,2270,2271,2272,2273,2274,2275,2276,2277,2278,2279,2280,2281,2282,2283,2284,2285,2286,2287,2288,2289,2290,2291,2292,2293,2294,2295,2296,2297,2298,2299,2300,2301,2302,2303,2304,2305,2306,2307,2308,2309,2310,2311,2312,2313,2314,2315,2316,2317,2318,2319,2320,2321,2322,2323,2324,2325,2326,2327,2328,2329,2330,2331,2332,2333,2334,2335,2336,2337,2338,2339,2340,2341,2342,2343,2344,2345,2346,2347,2348,2349,2350,2351,2352,2353,2354,2355,2356,2357,2358,2359,2360,2361,2362,2363,2364,2365,2366,2367,2368,2369,2370,2371,2372,2373,2374,2375,2376,2377,2378,2379,2380,2381,2382,2383,2384,2385,2386,2387,2388,2389,2390,2391,2392,2393,2394,2395,2396,2397,2398,2399,2400,2401,2402,2403,2404,2405,2406,2407,2408,2409,2410,2411,2412,2413,2414,2415,2416,2417,2418,2419,2420,2421,2422,2423,2424,2425,2426,2427,2428,2429,2430,2431,2432,2433,2434,2435,2436,2437,2438,2439,2440,2441,2442,2443,2444,2445,2446,2447,2448,2449,2450,2451,2452,2453,2454,2455,2456,2457,2458,2459,2460,2461,2462,2463,2464,2465,2466,2467,2468,2469,2470,2471,2472,2473,2474,2475,2476,2477,2478,2479,2480,2481,2482,2483,2484,2485,2486,2487,2488,2489,2490,2491,2492,2493,2494,2
```

## N개의 단어를 각각 N차원의 벡터로 표현하는 방식

## 단어가 포함되는 자리엔 1을 넣고 나머지에는 0을 넣는다

Nominal Data에만 적용

- car\_model
- accident\_location
- acc\_type1

	car_model
0	2
1	1
2	1
3	3
4	1
...	...
15995	2
15996	2
15997	2
15998	2
15999	2

pd.get\_dummies( )



	car_model_1	car_model_2	car_model_3	car_model_4	car_model_5
0	0	1	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	0	0	1	0	0
4	1	0	0	0	0
...	...	...	...	...	...
15995	0	1	0	0	0
15996	0	1	0	0	0
15997	0	1	0	0	0
15998	0	1	0	0	0
15999	0	1	0	0	0

new feature

```
np.unique(df['car_model'])
array([1, 2, 3, 4, 5], dtype=int64)
```

## Ordinal Data를 재분류해보고

### 순차적이지 않은 값을 수정

Ordinal Data에만 적용

- start\_hour
- accident\_hour

	Before	After
21~04시	1	<b>1</b>
05~07시	3	<b>2</b>
08~10시	4	<b>3</b>
11~13시	5	<b>4</b>
14~16시	6	<b>5</b>
17~20시	2	<b>6</b>





## 스케일링을 적용해보기도 하였지만..

### 스케일링 기법

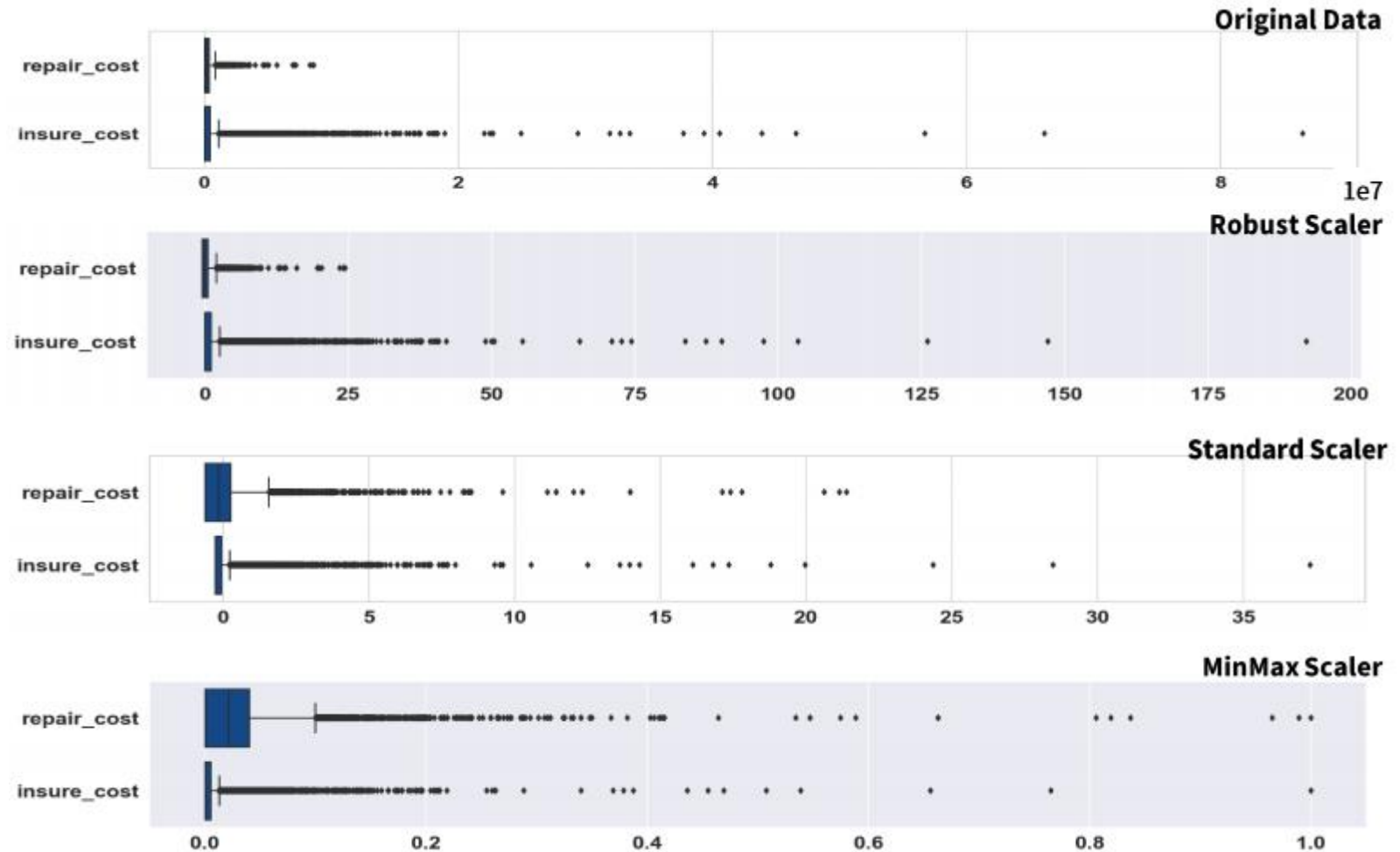
- MinMaxScaler
- StandardScaler
- RobustScaler

Outlier가 중요한 역할을 할때 사  
용하면 안됨

### 스케일링 영향을 받는 모델

- Logistic Regression(O)
- Tree계열 모델(X)

## Scaling

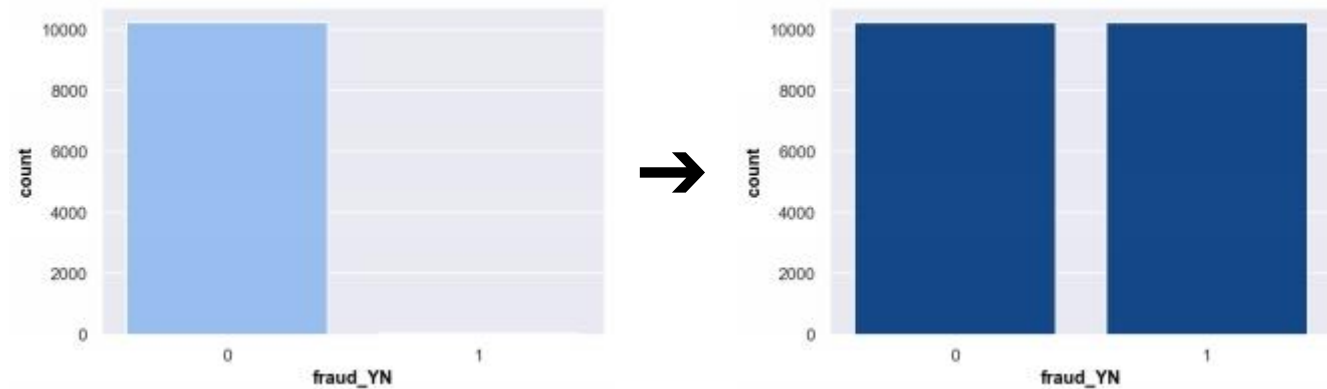


## 그나마샘플링이 도움이되는듯하다

오버샘플링으로데이터 불균형을 해소해보자.

### 오버샘플링 기법

- SMOTE
- BorderlineSMOTE
- RandomOverSampler
- ADASYN



### 언더샘플링을 지양한 이유

- 언더샘플링을 진행하기엔 데이터가 부족..  
train 기준, 34개의 데이터로 진행해야 한다

## 그러나 만족할만한 성능은 나오지 않았다

### 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

Source : 14 기

### 전처리 후 모델별 성능

	accuracy	precision	recall
LogisticRegression	0.078167	0.002501	0.002501
DecisionTree	0.993454	0.083333	0.142857

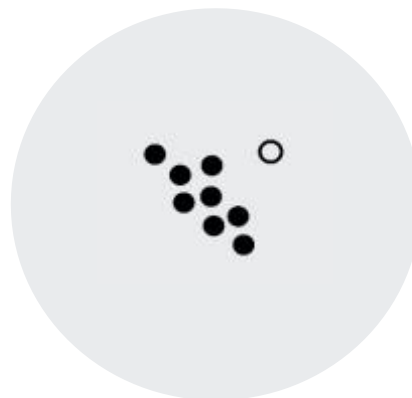
## 5. 모델링

## 우리가무언가놓치고있지는않을까?

혹시튜닝이 답이 되지 않을까?



결측치



이상치



스케일링



피쳐 생성

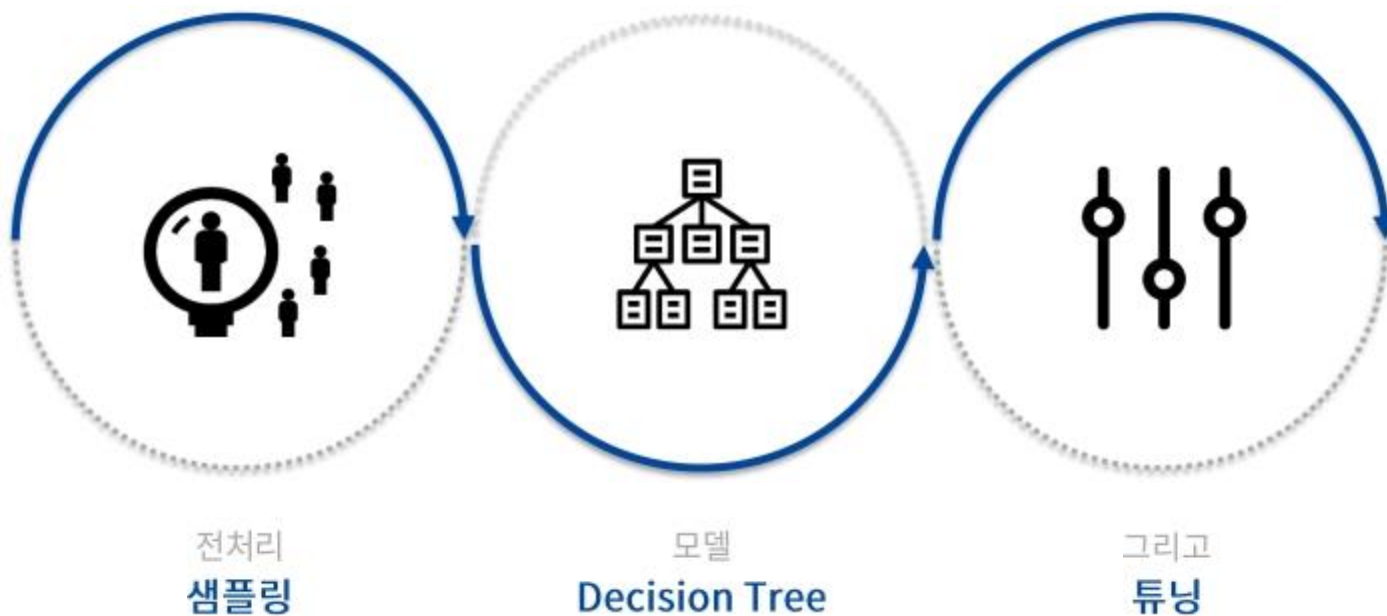


샘플링



튜닝

우리가무언가  
놓치고있지는  
않을까?



# 우리가무언가 놓치고있지는 않을까?

과적합을 방지하려면  
pruning이 필요하다.

수십 개의 parameter, 우리가 눈여겨본 것은?

## sklearn.tree.DecisionTreeClassifier

```
class sklearn.tree._classes.DecisionTreeClassifier(
    criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
    min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0) [source]
```

A decision tree classifier.

Read more in the User Guide.

<b>Parameters:</b>	<p><b>criterion</b> : {'gini', 'entropy'}, default='gini' The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.</p> <p><b>splitter</b> : {'best', 'random'}, default='best' The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.</p> <p><b>max_depth</b> : int, default=None The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.</p> <p><b>min_samples_split</b> : int or float, default=2 The minimum number of samples required to split an internal node:</p> <ul style="list-style-type: none"> <li>• If int, then consider min_samples_split as the minimum number.</li> <li>• If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.</li> </ul> <p>Changed in version 0.18: Added float values for fractions.</p> <p><b>min_samples_leaf</b> : int or float, default=1 The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.</p> <ul style="list-style-type: none"> <li>• If int, then consider min_samples_leaf as the minimum number.</li> <li>• If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.</li> </ul> <p>Changed in version 0.18: Added float values for fractions.</p> <p><b>min_weight_fraction_leaf</b> : float, default=0.0 The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.</p> <p><b>max_features</b> : int, float or {'auto', 'sqrt', 'log2'}, default=None The number of features to consider when looking for the best split:</p> <ul style="list-style-type: none"> <li>• If int, then consider max_features features at each split.</li> <li>• If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.</li> <li>• If "auto", then max_features=sqrt(n_features).</li> <li>• If "sqrt", then max_features=sqrt(n_features).</li> <li>• If "log2", then max_features=log2(n_features).</li> <li>• If None, then max_features=n_features.</li> </ul> <p>Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features.</p>
--------------------	--

## pruning 을 위한 후보군

- **max\_depth**  
tree 최대 깊이
- **class\_weight**  
예측하고자 하는 class에  
가중치를 주어서 학습
- **max\_features**  
노드 분리 시 고려할 속성 개수
- **criterion**  
분할 기준이 되는 불순도
- **min\_samples\_leaf**  
leaf 노드가 되기 위한  
최소한의 샘플 데이터 개수





# **Model 1 – Decision Tree 1**

## 답은그리 멀지않은곳에있다

model 1 은 다음과 같은 절차를 거쳤다.

- BorderlineSMOTE
- Decision Tree

```
max_depth
= 4
max_features
= 'sqrt'
class_weight
= {0:0.01, 1:1.0}
```

### model 1 – DecisionTree

#### 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

Source : 147]

#### train

	accuracy	precision	recall
LogisticRegression	0.823745	0.739890	0.998521
DecisionTree	0.923355	0.868079	0.998443
RandomForest	1.000000	1.000000	1.000000
LightGBM	1.000000	1.000000	1.000000

#### test

	accuracy	precision	recall
LogisticRegression	0.645626	0.005401	0.857143
<b>DecisionTree</b>	<b>0.777635</b>	<b>0.009986</b>	<b>1.000000</b>
RandomForest	0.997757	0.000000	0.000000
LightGBM	0.997437	0.000000	0.000000

## **Model 2 – Decision Tree 2**

## 답은그리 멀지않은곳에있다

model 2 는 다음과 같은 절차를 거쳤다.

- BorderlineSMOTE
- Decision Tree
  - max\_depth  
= 4
  - max\_features  
= 'sqrt'
  - class\_weight  
= 'balanced'

### model 2 – Decision Tree

#### 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

Source : 147]

#### train

	accuracy	precision	recall
LogisticRegression	0.823745	0.739890	0.998521
DecisionTree	0.949358	0.90971	0.997742
RandomForest	1.000000	1.000000	1.000000
LightGBM	1.000000	1.000000	1.000000

#### test

	accuracy	precision	recall
LogisticRegression	0.645626	0.005401	0.857143
<b>DecisionTree</b>	<b>0.868952</b>	<b>0.014493</b>	<b>0.857143</b>
RandomForest	0.997757	0.000000	0.000000
LightGBM	0.997437	0.000000	0.000000

**Model 3 ~ 5**

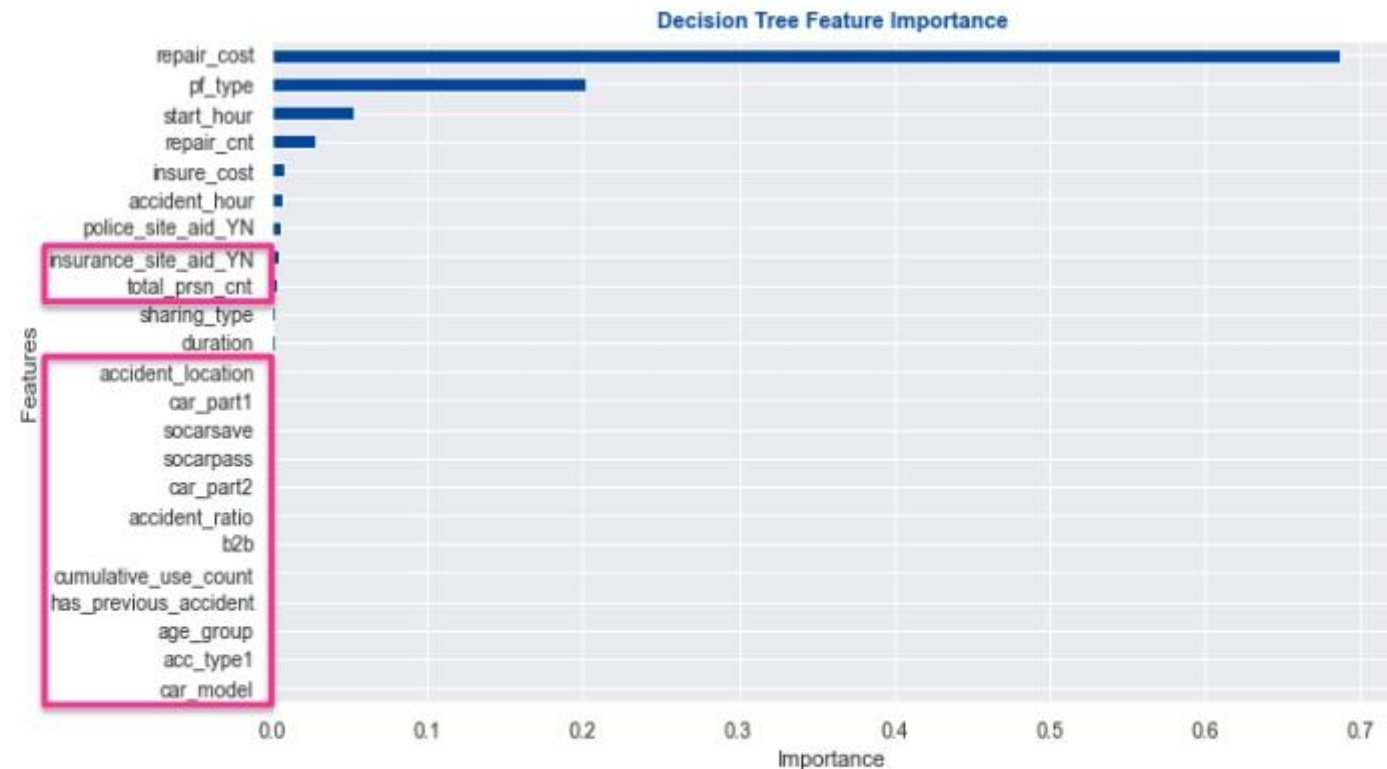
## 답은 그리 멀지않은 곳에있다

### model 1 - Feature Importances

## 각 피쳐 중요도 (불순도 감소분) 측정

모델 1 에 따르면..

- repair\_cost 가 가장 중요한 피쳐
- accident\_location, car\_part1 등 중요도가 0 인 피쳐는 사용하지 않는 것이 성능 향상에 도움이 될 것 같다
- 중요도가 낮은 피쳐 중에서 결측치가 많은 피쳐도 사용하지 않는 것이 성능 향상에 도움이 될 것 같다



## 그래서 컬럼을 버렸다 ^^

```
# Remove Features
isnull = ['accident_location', 'car_part1', 'car_part2',
          'acc_type1', 'socarsave', 'socarpass',
          'insurance_site_aid_YN', 'accident_ratio',
          'b2b', 'cumulative_use_count', 'has_previous_accident',
          'age_group', 'total_prsn_cnt']
df = df.drop(isnull, axis=1)
```

그리고 또 다시 즐거운  
파라미터 튜닝 여행 🏃🏃🏃

DecisionTree외 남은 3 Classifier 모두 성능이 좋아졌다.

## 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

## train

	accuracy	precision	recall
LogisticRegression	0.786999	0.765349	0.827793
DecisionTree	0.820670	0.869881	0.754146
RandomForest	0.964344	0.935143	0.997898
LightGBM	0.977540	0.958857	0.997898

## test

	accuracy	precision	recall
LogisticRegression	0.742711	0.006203	0.714286
DecisionTree	0.891381	0.008876	0.428571
RandomForest	0.934957	0.024272	0.714286
LightGBM	0.964114	0.043478	0.714286



## **Model 3 – Logistic Regression**

model 3 은 다음과 같은 절차를 거쳤다.

- drop columns
- BorderlineSMOTE
- Logistic Regression  
     class\_weight  
     = 'balanced'

## 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

## train

	accuracy	precision	recall
LogisticRegression	0.786999	0.765349	0.827793
DecisionTree	0.820670	0.869881	0.754146
RandomForest	0.964344	0.935143	0.997898
LightGBM	0.977540	0.958857	0.997898

## test

	accuracy	precision	recall
<b>LogisticRegression</b>	<b>0.742711</b>	<b>0.006203</b>	<b>0.714286</b>
DecisionTree	0.891381	0.008876	0.428571
RandomForest	0.934957	0.024272	0.714286
LightGBM	0.964114	0.043478	0.714286

## **Model 4 – Random Forest**

model 4 는 다음과 같은 절차를 거쳤다.

- drop columns
- BorderlineSMOTE
- Random Forest
  - max\_depth = 7
  - class\_weight = 'balanced'

## 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

## train

	accuracy	precision	recall
LogisticRegression	0.786999	0.765349	0.827793
DecisionTree	0.820670	0.869881	0.754146
RandomForest	0.964344	0.935143	0.997898
LightGBM	0.977540	0.958857	0.997898

## test

	accuracy	precision	recall
LogisticRegression	0.742711	0.006203	0.714286
DecisionTree	0.891381	0.008876	0.428571
<b>RandomForest</b>	<b>0.934957</b>	<b>0.024272</b>	<b>0.714286</b>
LightGBM	0.964114	0.043478	0.714286

## **Model 5 – LightGBM**

model 5 는 다음과 같은 절차를 거쳤다.

- drop columns
- BorderlineSMOTE
- LightGBM

max\_depth

= 4

learning rate

= 0.02

num\_iterations

= 330

## 우리의 Baseline

	accuracy	precision	recall
SVM	0.492791	0.003153	0.714286

## train

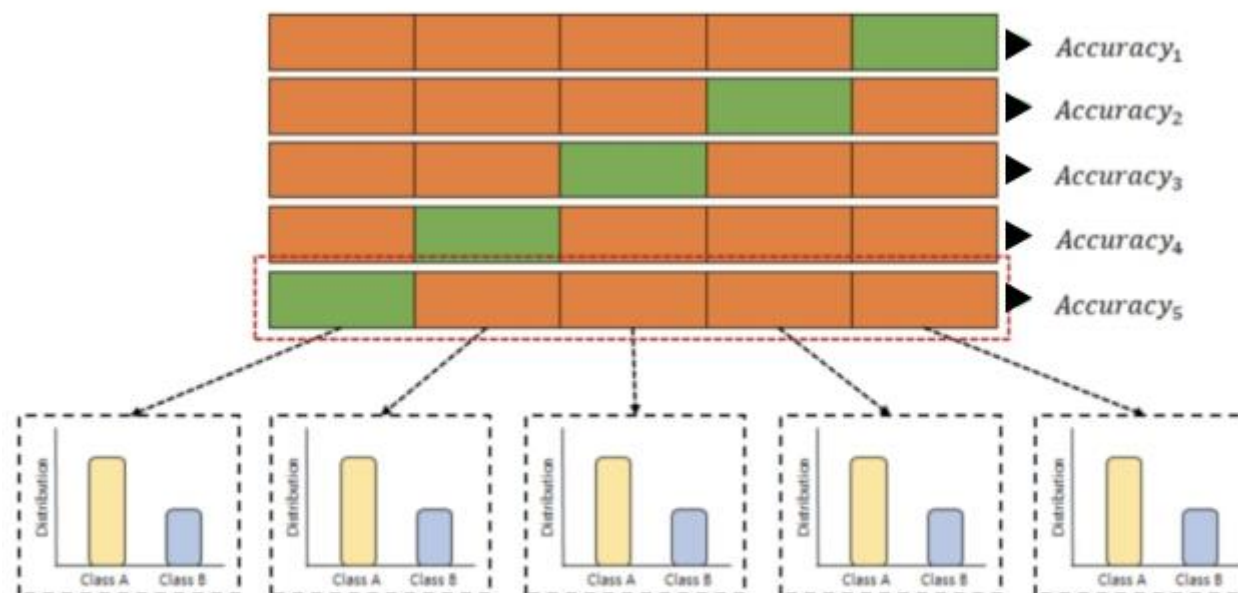
	accuracy	precision	recall
LogisticRegression	0.786999	0.765349	0.827793
DecisionTree	0.820670	0.869881	0.754146
RandomForest	0.964344	0.935143	0.997898
LightGBM	0.977540	0.958857	0.997898

## test

	accuracy	precision	recall
LogisticRegression	0.742711	0.006203	0.714286
DecisionTree	0.891381	0.008876	0.428571
RandomForest	0.934957	0.024272	0.714286
<b>LightGBM</b>	<b>0.964114</b>	<b>0.043478</b>	<b>0.714286</b>

## 6. 모델 검증

## Stratified K-fold



$$Accuracy = Average(Accuracy_1, \dots, Accuracy_k)$$

- 데이터 별 분포를 고려해서 데이터 fold 셋을 만드는 방법
- Class별 데이터가 아주 불균형하다면 Stratified K-fold를 사용해야 한다



## Stratified K-fold 성능 & 표준편차

Stratified K-fold  
n\_splits  
= 5

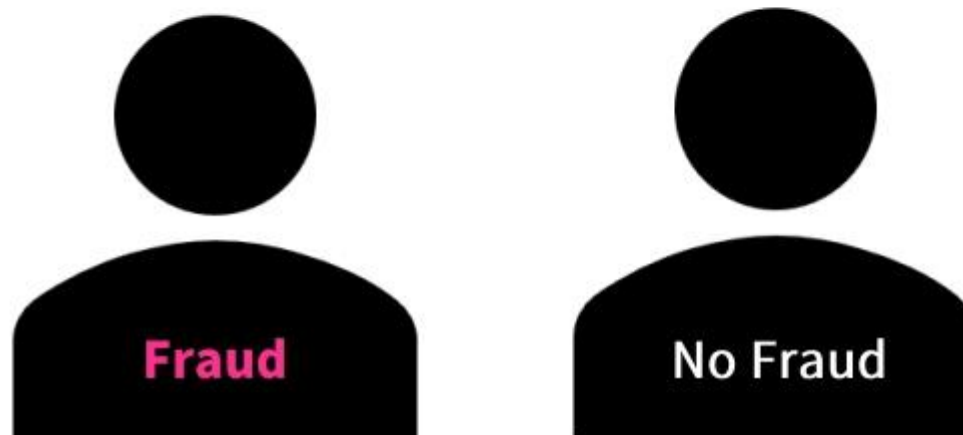
model	accuracy	std	precision	std	recall	std
1	0.94	0.01	0.89	0.01	1.00	0.00
2	0.95	0.00	0.91	0.00	1.00	0.01
3	0.85	0.03	0.77	0.01	1.00	0.07
4	0.96	0.00	0.92	1.00	1.00	0.00
5	0.98	0.00	0.96	0.01	1.00	0.00

## 테스트 성능과 비교

model	accuracy			pre			recall		
	CV	Diff	test	CV	Diff	test	CV	Diff	test
1	0.94	0.16	0.78	0.89	0.88	0.78	1.00	0.00	1.00
2	0.95	0.06	0.89	0.91	0.90	0.89	1.00	0.14	0.86
3	0.85	0.06	0.74	0.77	0.76	0.74	1.00	0.15	0.71
4	0.96	0.03	0.93	0.92	0.91	0.93	1.00	0.29	0.71
5	0.98	0.02	0.96	0.96	0.92	0.96	1.00	0.29	0.71

## 7. 결론

## Model 1



수리비용

자기부담금

대여 시작 시간

대여 유형

파손부위

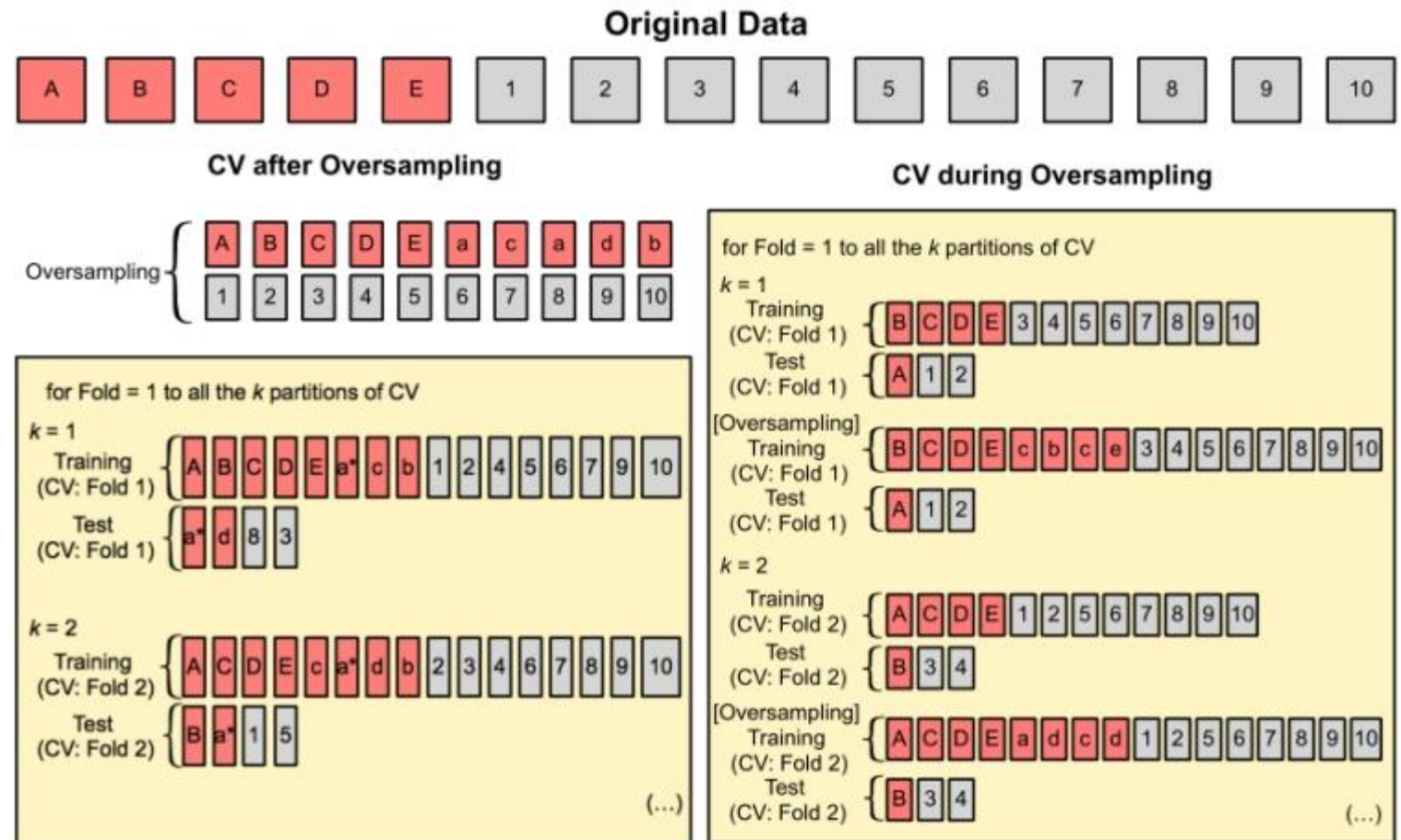
보험사 현장출동

해당 정보는 SOCAR 기업정보 보호를 위하여  
숨김처리 되었음.

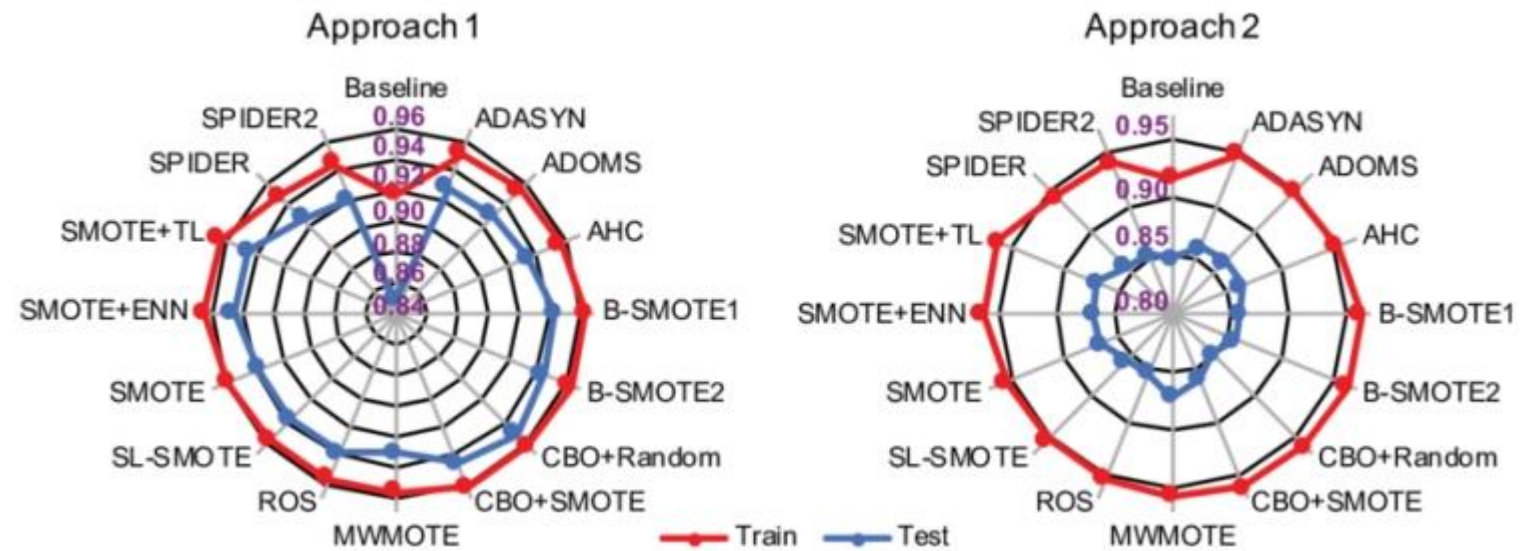
## Stratified K-fold 성능 & 표준편차

	CV after sampling			Diff			CV during sampling		
Model	accuracy	precision	recall	accuracy	precision	recall	accuracy	precision	recall
1	0.94	0.89	1.00	0.53	0.47	0.25	0.41	0.42	0.75
2	0.95	0.91	1.00	0.13	0.09	0.75	0.82	0.82	0.25
3	0.85	0.77	1.00	0.06	0.77	0.55	0.72	0.00	0.33
4	0.96	0.92	1.00	0.01	0.92	0.89	0.94	0.01	0.16
5	0.98	0.96	1.00	0.06	0.95	0.84	0.92	0.01	0.18

일반화할 수 있는가?



Overoptimism 발생



Santos, Miriam & Soares, Jastin & Henriques Abreu, Pedro & Araujo, Helder & Santos, Joao. (2018). Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches. IEEE Computational Intelligence Magazine. 13. 59-76. 10.1109/MCI.2018.2866730.



## 마무리

- **크게 개선되지 않은 precision**

데이터 라벨링이 잘못되었을 가능성 ?

- 피해자에게 불리한 법적 시스템 ⑦ 미신고 사례 多

(<http://asq.kr/QhrdCTCuNHV11o>)

- 사기를 밝히는 것이 쉽지 않다

- **모델 1에 의하면 결측값을 가진 피처가 중요한 피처**

e.g. 보험사/경찰 출동 유무, 탑승 인원 수

- **Scaling 이 tree계열 모델 성능에 영향을 줄 수도 있다**

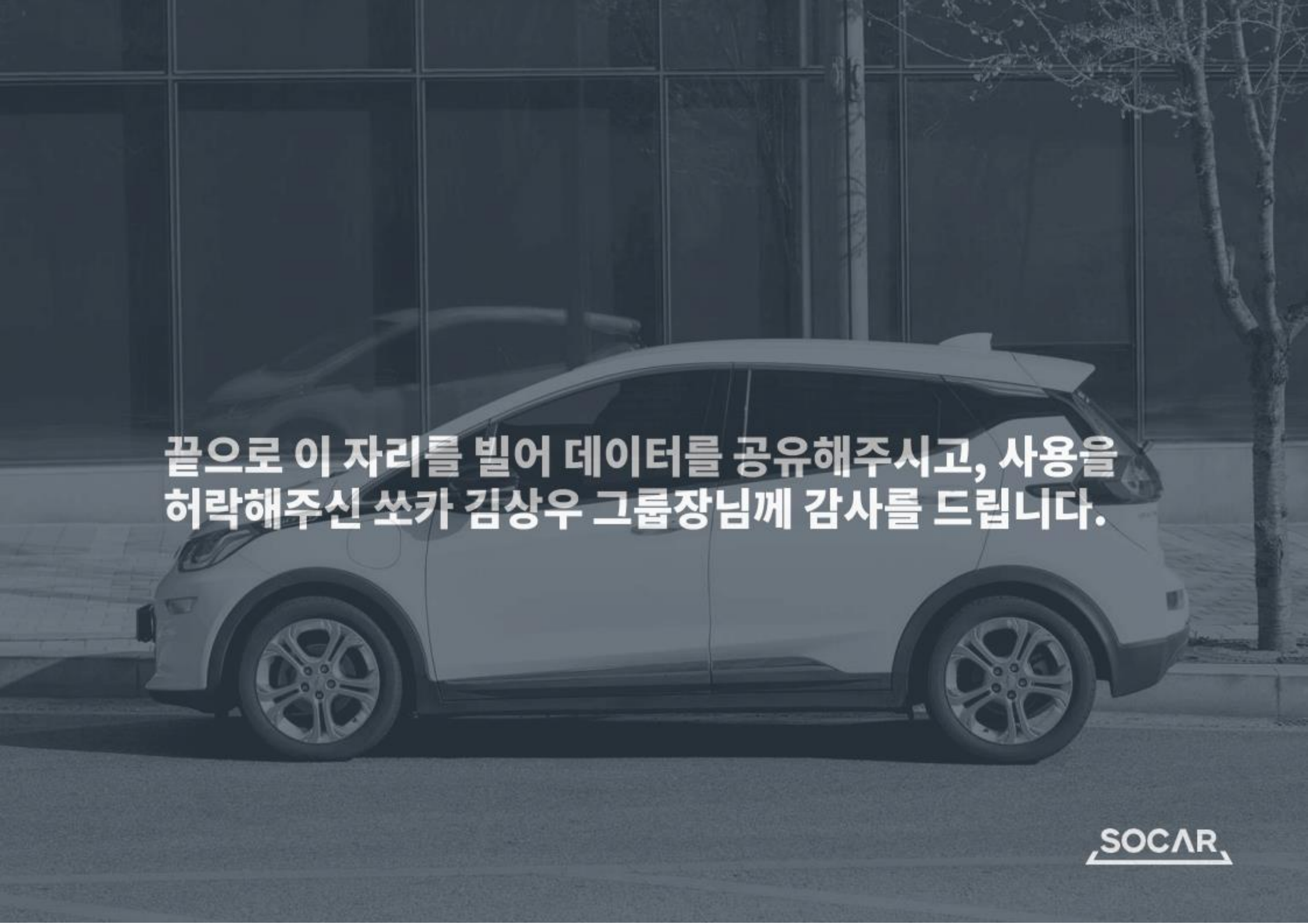
Scaling 은 Oversampling 에 영향을 준다 ⑦ 결과적으로 모델 학습에 영향을 주게 된다

- **과적합엔 튜닝이 답이 될 수 있다**

우리 모델의 경우 max\_features 가 성능향상에 결정적인 도움이 되었다

- **작지만 모이면 큰 힘!**

의견을 조율하고 협력하는 과정을 통하여 좋은 결과를 낼 수 있었다

A white SUV is parked on a paved surface in front of a modern building with large glass windows. The car is shown from a side profile. The background includes a tree on the right and a reflection of the car in the glass. The overall image has a dark, monochromatic tint.

**끝으로 이 자리를 빌어 데이터를 공유해주시고, 사용을  
허락해주신 쏘카 김상우 그룹장님께 감사를 드립니다.**





김준성

E : joon90s@gmail.com  
G : github.com/whistle-boy



방희란

E : bluebluedawn@gmail.com  
G : github.com/Heeran-cloud



정혜주

E : hjjung193@gmail.com  
G : github.com/hjung53