Proyek UAS Expert System

Data set yang digunakan The Cars : https://www.kaggle.com/datasets/qusaybtoush1990/the-cars

Import Library

In [1]:
```
import sys
import numpy as np
import pandas as pd
```

Pengecekan Dataset

In [2]:
```
data = pd.read_csv("car1.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	buying	maint	doors	persons	lug_boot	safety	rating
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   rating    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

In [5]:
```
data.describe()
```

Out[5]:

	buying	maint	doors	persons	lug_boot	safety	rating
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	vhigh	vhigh	2	2	small	low	unacc
freq	432	432	432	576	576	576	1210

Metode yang digunakan dalam perhitungan adalah TOPIS, maka kita akan menggantikan prioritas / yang dicari dalam pembelian mobil.

Value yang diinginkan adalah; buying: vhigh, maint: vlow, doors: 5more, persons: more, lug_boot: high, safety: high, rating: vgood.
Weight value yang digunakan adalah default 1 untuk perhitungan ini.

Note: Value yang diambil merupakan hasil pendapat dari kelompok kami.

Proses Defuzzyfikasi

In [6]:
```
# defuzzifikasi untuk kolom 'buying'
def defuzzify_buying(value):
    if value == 'low':
        return (1+2+3)/3
    elif value == 'med':
        return (3+4+5)/3
    elif value == 'high':
        return (5+6+7)/3
    elif value == 'vhigh':
        return (7+8+9)/3
    else:
        return None
```

In [7]:
```
# defuzzifikasi untuk kolom 'maint'
def defuzzify_maint(value):
    if value == 'vhigh':
        return (1+2+3)/3
    elif value == 'high':
        return (3+4+5)/3
    elif value == 'med':
        return (5+6+7)/3
    elif value == 'low':
        return (7+8+9)/3
    else:
        return None
```

In [8]:
```
# defuzzifikasi untuk kolom 'doors'
def defuzzify_doors(value):
    if value == '2':
        return (1+2+3)/3
    elif value == '3':
        return (2+3+4)/3
    elif value == '4':
        return (3+4+5)/3
    elif value == '5more':
        return (4+5+6)/3
    else:
        return None
```

In [9]:
```
# defuzzifikasi untuk kolom 'persons'
def defuzzify_persons(value):
    if value == '2':
        return (1+2+3)/3
    elif value == '4':
        return (3+4+5)/3
    elif value == 'more':
        return (5+6+7)/3
    else:
        return None
```

In [10]:
```
# defuzzifikasi untuk kolom 'lug_boot'
def defuzzify_lug_boot(value):
    if value == 'small':
        return (1+2+3)/3
    elif value == 'med':
        return (3+4+5)/3
    elif value == 'high':
        return (5+6+7)/3
    else:
        return None
```

In [11]:
```
# defuzzifikasi untuk kolom 'safety'
def defuzzify_safety(value):
    if value == 'low':
        return (1+2+3)/3
    elif value == 'med':
        return (3+4+5)/3
    elif value == 'high':
        return (5+6+7)/3
    else:
        return None
```

In [12]:
```
# defuzzifikasi untuk kolom 'rating'
def defuzzify_rating(value):
    if value == 'unacc':
        return (1+2+3)/3
    elif value == 'acc':
        return (3+4+5)/3
    elif value == 'good':
        return (5+6+7)/3
    elif value == 'vgood':
        return (7+8+9)/3
    else:
        return None
```

In [13]:
```
data['buying'] = data['buying'].apply(defuzzify_buying)
data['maint'] = data['maint'].apply(defuzzify_maint)
data['doors'] = data['doors'].apply(defuzzify_doors)
data['persons'] = data['persons'].apply(defuzzify_persons)
data['lug_boot'] = data['lug_boot'].apply(defuzzify_lug_boot)
data['safety'] = data['safety'].apply(defuzzify_safety)
data['rating'] = data['rating'].apply(defuzzify_rating)
```

In [14]:
```
data.head()
```

Out[14]:

	buying	maint	doors	persons	lug_boot	safety	rating
0	8.0	2.0	2.0	2.0	2.0	2.0	2.0
1	8.0	2.0	2.0	2.0	2.0	4.0	2.0
2	8.0	2.0	2.0	2.0	2.0	6.0	2.0
3	8.0	2.0	2.0	2.0	4.0	2.0	2.0
4	8.0	2.0	2.0	2.0	4.0	4.0	2.0

In [15]:
```
data = data.dropna()
```

In [16]:
```
df = pd.DataFrame(data)

# ubah dataframe menjadi matriks numpy
decision_matrix = df.to_numpy()

print("Matriks hasil konversi:")
print(decision_matrix)
```

```
Matriks hasil konversi:
[[8. 2. 2. ... 2. 2. 2.]
 [8. 2. 2. ... 2. 4. 2.]
 [8. 2. 2. ... 2. 6. 2.]
 ...
 [2. 8. 5. ... 4. 2. 2.]
 [2. 8. 5. ... 4. 4. 6.]
 [2. 8. 5. ... 4. 6. 8.]]
```

Proses Algoritma TOPSIS

In [17]:
```
# normalisasi matriks keputusan
def normalize_decision_matrix(matrix):
    normalized_matrix = np.zeros(matrix.shape)
    for j in range(matrix.shape[1]):
        col_sum_of_squares = np.sqrt(np.sum(matrix[:, j] ** 2))
        normalized_matrix[:, j] = matrix[:, j] / col_sum_of_squares
    return normalized_matrix

normalized_decision_matrix = normalize_decision_matrix(decision_matrix)
print("Matriks keputusan yang telah dinormalisasi:")
print(normalized_decision_matrix)
```

```
Matriks keputusan yang telah dinormalisasi:
[[0.04303315 0.01075829 0.01603751 ... 0.0186339  0.01363862 0.01958527]
 [0.04303315 0.01075829 0.01603751 ... 0.0186339  0.02727724 0.01958527]
 [0.04303315 0.01075829 0.01603751 ... 0.0186339  0.04091585 0.01958527]
 ...
 [0.01075829 0.04303315 0.04009377 ... 0.0372678  0.01363862 0.01958527]
 [0.01075829 0.04303315 0.04009377 ... 0.0372678  0.02727724 0.0587558 ]
 [0.01075829 0.04303315 0.04009377 ... 0.0372678  0.04091585 0.07834107]]
```

In [18]:
```
weighted_value = [1, 1, 1, 1, 1, 1, 1]
```

In [19]:
```
# karena nilai Weight = 1 maka tidak ada perubahan nilai pada matriks
weighted_matrix = normalized_decision_matrix * weighted_value
```

In [20]:
```
print(weighted_matrix)
```

```
[[0.04303315 0.01075829 0.01603751 ... 0.0186339  0.01363862 0.01958527]
 [0.04303315 0.01075829 0.01603751 ... 0.0186339  0.02727724 0.01958527]
 [0.04303315 0.01075829 0.01603751 ... 0.0186339  0.04091585 0.01958527]
 ...
 [0.01075829 0.04303315 0.04009377 ... 0.0372678  0.01363862 0.01958527]
 [0.01075829 0.04303315 0.04009377 ... 0.0372678  0.02727724 0.0587558 ]
 [0.01075829 0.04303315 0.04009377 ... 0.0372678  0.04091585 0.07834107]]
```

In [21]:
```
# solusi ideal positif dan negatif
ideal_positive = np.max(weighted_matrix, axis=0)
ideal_negative = np.min(weighted_matrix, axis=0)
```

In [22]:
```
# nilai ideal positif
ideal_positive
```

Out[22]:
```
array([0.04303315, 0.04303315, 0.04009377, 0.04091585, 0.0372678 ,
       0.04091585, 0.07834107])
```

In [23]:
```
# nilai ideal negatif
ideal_negative
```

Out[23]:
```
array([0.01075829, 0.01075829, 0.01603751, 0.01363862, 0.0186339 ,
       0.01363862, 0.01958527])
```

In [24]:
```
# jarak relatif
positive_distance = np.sum((weighted_matrix - ideal_positive)**2, axis=1)**0.5
negative_distance = np.sum((weighted_matrix - ideal_negative)**2, axis=1)**0.5
```

In [25]:
```
# jarak ideal positif
positive_distance
```

Out[25]:
```
array([0.08311397, 0.07968624, 0.07851041, ..., 0.07237374, 0.04014052,
       0.03227486])
```

In [26]:
```
# jarak ideal negative
negative_distance
```

Out[26]:
```
array([0.03227486, 0.03503824, 0.04225771, ..., 0.05207341, 0.06657314,
       0.08311397])
```

In [27]:
```
# skor preferensi
preference_score = negative_distance / (positive_distance + negative_distance)
print(preference_score)
```

```
[0.27870525 0.3054121  0.34990784 ... 0.41843796 0.62384835 0.72029475]
```

In [28]:
```
# peringkat
ranking = np.argsort(preference_score)
ranking
```

Out[28]:
```
array([864, 882, 936, ..., 791, 845, 863], dtype=int64)
```

Hasil Ranking

In [29]:
```
# membuat dataframe baru untuk menampilkan hasil ranking
result_df = pd.DataFrame({
    'Index': np.arange(1, len(preference_score) + 1),
    'Ranking': ranking
})

print("Hasil Ranking:")
print(result_df)
```

```
Hasil Ranking:
      Index  Ranking
0         1      864
1         2      882
2         3      936
3         4      576
4         5      865
...     ...      ...
1147   1148      773
1148   1149      857
1149   1150      791
1150   1151      845
1151   1152      863

[1152 rows x 2 columns]
```

In [30]:
```
result_df = pd.DataFrame({
    'Ranking': ranking + 1,
    'Preference Score': preference_score,
    'Negative Distance': negative_distance,
    'Positive Distance': positive_distance
})

# urutin dataframe berdasarkan kolom 'Ranking'
result_df = result_df.sort_values(by='Ranking')

print("Hasil Ranking dan Skor Preferensi:")
print(result_df)
```

```
Hasil Ranking dan Skor Preferensi:
      Ranking  Preference Score  Negative Distance  Positive Distance
137         1          0.401549           0.054894           0.064041
211         2          0.461549           0.054894           0.064041
398         3          0.355641           0.039749           0.072017
246         4          0.436971           0.053774           0.069287
353         5          0.410161           0.048155           0.069251
...       ...               ...                ...                ...
991      1148          0.288605           0.030103           0.077144
1111     1149          0.458813           0.049190           0.058021
751      1150          0.450825           0.044234           0.053884
1113     1151          0.387381           0.046875           0.074129
1141     1152          0.459207           0.048744           0.057404

[1152 rows x 4 columns]
```

In [31]:
```
!jupyter nbconvert --to html "./00000056883_NayashaClarisaDwisutrisna_Demo_UAS_IF541.ipynb" --output-dir="./"
```

```
[NbConvertApp] Converting notebook ./00000056883_NayashaClarisaDwisutrisna_Demo_UAS_IF541.ipynb to html
[NbConvertApp] Writing 633080 bytes to 00000056883_NayashaClarisaDwisutrisna_Demo_UAS_IF541.html
```