CS3-end1-p2

2-1

2-1-1 (1) 大きく / large
2-1-2 (1) ジニ不純度 / Gini impurity (2) 小さくなる / small
2-1-3 (1) 分散 / variance (2) 小さくなる / small

2-2

Import libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

Read CSV file

In [2]:
```
csv_in = 'end1-p2.csv'
df = pd.read_csv(csv_in, sep=',', skiprows=0, header=0)
print(df.shape)
print(df.info())
display(df.head())
```

```
(240, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   y       240 non-null    float64
 1   e1      240 non-null    object 
 2   e2      240 non-null    object 
 3   e3      240 non-null    float64
```

```
 4   e4      240 non-null    float64
dtypes: float64(3), object(2)
memory usage: 9.5+ KB
None
```

	y	e1	e2	e3	e4
0	12.32	40-60	N	0.17	1.30
1	5.17	60-	N	1.47	-1.67
2	16.45	-20	N	-0.52	-1.59
3	10.52	40-60	H	-1.07	-0.22
4	9.52	40-60	N	0.58	0.24

Separate features and true values

特徴量と正解の数値を分ける

In [3]:
```
X = df.drop(columns='y')  # features
y = df['y']  # true values
print('X:', X.shape)
display(X.head())
print('y:', y.shape)
print(y.head())
```

```
X: (240, 4)
```

	e1	e2	e3	e4
0	40-60	N	0.17	1.30
1	60-	N	1.47	-1.67
2	-20	N	-0.52	-1.59
3	40-60	H	-1.07	-0.22
4	40-60	N	0.58	0.24

```
y: (240,)
0    12.32
```

```
1      5.17
2     16.45
3     10.52
4      9.52
Name: y, dtype: float64
```

Encoding of categorical variables

Assign integers

整数を割り当てる

In [4]:
```
print(X['e1'].value_counts())
```

```
40-60    72
60-      61
20-40    56
-20      51
Name: e1, dtype: int64
```

In [5]:
```
X['e1'] = X['e1'].replace(
    {'-20':1, '20-40':2,
     '40-60':3, '60-':4,
    }
)
display(X.head())
```

	e1	e2	e3	e4
0	3	N	0.17	1.30
1	4	N	1.47	-1.67
2	1	N	-0.52	-1.59
3	3	H	-1.07	-0.22
4	3	N	0.58	0.24

Apply get_dummies()

ダミー変数化

In [6]:
```
X_dumm = pd.get_dummies(X, drop_first=True)
print('X_dumm:', X_dumm.shape)
display(X_dumm.head())
```

X_dumm: (240, 5)

	e1	e3	e4	e2_H	e2_N
0	3	0.17	1.30	0	1
1	4	1.47	-1.67	0	1
2	1	-0.52	-1.59	0	1
3	3	-1.07	-0.22	1	0
4	3	0.58	0.24	0	1

Split X and y for train and test

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X_dumm, y, test_size=0.33, random_state=28)
```

In [8]:
```
print(X_train.shape)
print(y_train.shape)
```

(160, 5)
(160,)

In [9]:
```
print(X_test.shape)
print(y_test.shape)
```

(80, 5)
(80,)

Ans 4. 80

Training of RFR

In [10]:
```
rfr=RandomForestRegressor(n_estimators=300, max_depth=None,
                          random_state=43)
```

In [11]:
```
%%time
rfr.fit(X_train, y_train)
```

```
CPU times: total: 438 ms
Wall time: 431 ms
```

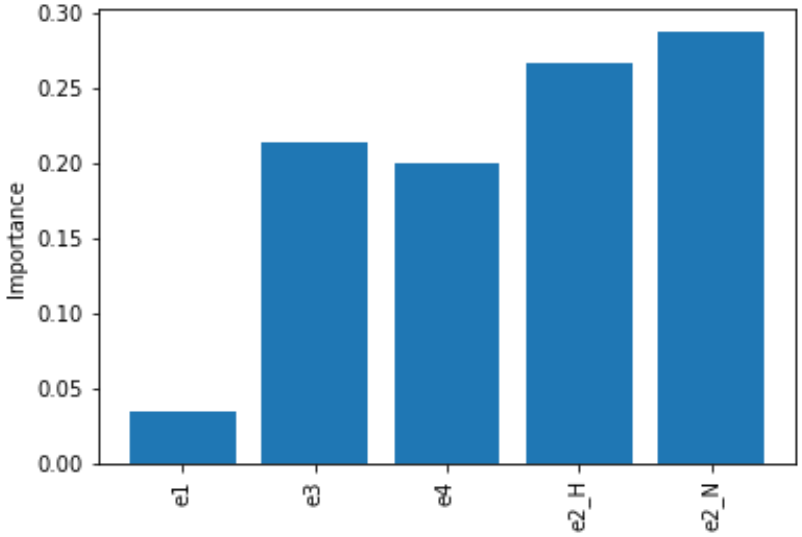
Out[11]:
```
RandomForestRegressor(n_estimators=300, random_state=43)
```

Show feature importances of the model

In [12]:
```
ser_fi = pd.Series(rfr.feature_importances_, index=X_train.columns)
print(ser_fi.sort_values(ascending=False))
```

```
e2_N    0.287392
e2_H    0.266108
e3      0.213313
e4      0.199019
e1      0.034167
dtype: float64
```

In [13]:
```
plt.bar(X_train.columns, rfr.feature_importances_)
plt.ylabel('Importance')
plt.xticks(rotation=90)
plt.show()
```



Ans 8. e2_N, 0.287

Do prediction for train data

In [14]:
```
y_train_pred = rfr.predict(X_train)
```

In [15]:
```
mse = mean_squared_error(y_train, y_train_pred)
print('MSE, RMSE for train data:', mse, np.sqrt(mse))
```

```
MSE, RMSE for train data: 3.715420605624993 1.9275426339318653
```

Ans 6. 1.928

Do prediction for test data

In [16]:
```
y_test_pred = rfr.predict(X_test)
```

In [17]:
```
mse = mean_squared_error(y_test, y_test_pred)
print('MSE, RMSE for test data:', mse, np.sqrt(mse))
```

```
MSE, RMSE for test data: 17.844390100736128 4.2242620776576025
```

Ans 7. 4.224

Option problems

In [18]:

```
csv_pred = 'end1-p2-pred.csv'
df_pred = pd.read_csv(csv_pred, sep=',', skiprows=0, header=0)
print(df_pred.shape)
print(df_pred.info())
display(df_pred.head())
```

```
(50, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   e1      50 non-null     object 
 1   e2      50 non-null     object 
 2   e3      50 non-null     float64
 3   e4      50 non-null     float64
dtypes: float64(2), object(2)
memory usage: 1.7+ KB
None
```

	e1	e2	e3	e4
0	-20	C	0.50	-0.15
1	-20	H	-1.42	1.72
2	60-	N	-0.78	0.56
3	40-60	C	0.67	0.05
4	20-40	H	-1.18	-1.14

In [19]:

```
print(df_pred['e1'].value_counts())
```

```
-20      13
60-      13
40-60    13
20-40    11
Name: e1, dtype: int64
```

In [20]:

```
df_pred['e1'] = df_pred['e1'].replace(
    {'-20':1, '20-40':2,
     '40-60':3, '60-':4,
    }
)
display(df_pred.head())
```

	e1	e2	e3	e4
0	1	C	0.50	-0.15
1	1	H	-1.42	1.72
2	4	N	-0.78	0.56
3	3	C	0.67	0.05
4	2	H	-1.18	-1.14

In [21]:

```
df_pred_dumm = pd.get_dummies(df_pred, drop_first=True)
print('df_pred_dumm:', df_pred_dumm.shape)
display(df_pred_dumm.head())
```

```
df_pred_dumm: (50, 5)
```

	e1	e3	e4	e2_H	e2_N
0	1	0.50	-0.15	0	0
1	1	-1.42	1.72	1	0
2	4	-0.78	0.56	0	1
3	3	0.67	0.05	0	0
4	2	-1.18	-1.14	1	0

In [22]:

```
y_pred = rfr.predict(df_pred_dumm)
print(y_pred.max())
```

```
17.28346666666667
```

Ans 9. 17.283

In [ ]: