操作報告

> DataSet using: duke_gpa.csv

First I load my data using pandas.

In [ ]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.linear_model import LogisticRegression

# Load your data
df = pd.read_csv('duke_gpa.csv')
print(df)
```

```
      gpa  studyweek  sleepnight  out  gender
0   3.890         50         6.0  3.0  female
1   3.900         15         6.0  1.0  female
2   3.750         15         7.0  1.0  female
3   3.600         10         6.0  4.0    male
4   4.000         25         7.0  3.0  female
5   3.150         20         7.0  3.0    male
6   3.250         15         6.0  1.0  female
7   3.925         10         8.0  3.0  female
8   3.428         12         8.0  2.0  female
9   3.800          2         8.0  4.0    male
10  3.900         10         8.0  1.0  female
11  2.900         30         6.0  2.0  female
12  3.925         30         7.0  2.0  female
13  3.650         21         9.0  3.0  female
14  3.750         10         8.5  3.5  female
15  4.670         14         6.5  3.0    male
16  3.100         12         7.5  3.5    male
17  3.800         12         8.0  1.0  female
18  3.400          4         9.0  3.0  female
19  3.575         45         6.5  1.5  female
20  3.850          6         7.0  2.5  female
21  3.400         10         7.0  3.0  female
22  3.500         12         8.0  2.0    male
23  3.600         13         6.0  3.5  female
24  3.825         35         8.0  4.0  female
25  3.925         10         8.0  3.0  female
26  4.000         40         8.0  3.0  female
27  3.425         14         9.0  3.0  female
28  3.750         30         6.0  0.0  female
29  3.150          8         6.0  0.0  female
30  3.400          8         6.5  2.0  female
31  3.700         20         7.0  1.0  female
32  3.360         40         7.0  1.0  female
33  3.700         15         7.0  1.5    male
34  3.700         25         5.0  1.0  female
35  3.600         10         7.0  2.0  female
36  3.825         18         7.0  1.5  female
37  3.200         15         6.0  1.0  female
38  3.500         30         8.0  3.0    male
39  3.500         11         7.0  1.5  female
40  3.000         28         6.0  1.5  female
41  3.980          4         7.0  1.5  female
42  3.700          4         5.0  1.0    male
43  3.810         25         7.5  2.5  female
44  4.000         42         5.0  1.0  female
45  3.100          3         7.0  2.0    male
46  3.400         42         9.0  2.0    male
47  3.500         25         8.0  2.0    male
48  3.650         20         6.0  2.0  female
49  3.700          7         8.0  2.0  female
50  3.100          6         8.0  1.0  female
51  4.000         20         7.0  3.0  female
52  3.350         45         6.0  2.0  female
53  3.541         30         7.5  1.5  female
54  2.900         20         6.0  3.0  female
```

We can observe that there is a gender column which is nominal data. We convert it into binary data, 1 stands for male and 0 for female.

In [ ]:

```python
# Load the data
df = pd.read_csv('duke_gpa.csv')

df['gender'] = df['gender'].map({'male': 1, 'female': 0})

print(df)
```

	gpa	studyweek	sleepnight	out	gender
0	3.890	50	6.0	3.0	0
1	3.900	15	6.0	1.0	0
2	3.750	15	7.0	1.0	0
3	3.600	10	6.0	4.0	1
4	4.000	25	7.0	3.0	0
5	3.150	20	7.0	3.0	1
6	3.250	15	6.0	1.0	0
7	3.925	10	8.0	3.0	0
8	3.428	12	8.0	2.0	0
9	3.800	2	8.0	4.0	1
10	3.900	10	8.0	1.0	0
11	2.900	30	6.0	2.0	0
12	3.925	30	7.0	2.0	0
13	3.650	21	9.0	3.0	0
14	3.750	10	8.5	3.5	0
15	4.670	14	6.5	3.0	1
16	3.100	12	7.5	3.5	1
17	3.800	12	8.0	1.0	0
18	3.400	4	9.0	3.0	0
19	3.575	45	6.5	1.5	0
20	3.850	6	7.0	2.5	0
21	3.400	10	7.0	3.0	0
22	3.500	12	8.0	2.0	1
23	3.600	13	6.0	3.5	0
24	3.825	35	8.0	4.0	0
25	3.925	10	8.0	3.0	0
26	4.000	40	8.0	3.0	0
27	3.425	14	9.0	3.0	0
28	3.750	30	6.0	0.0	0
29	3.150	8	6.0	0.0	0
30	3.400	8	6.5	2.0	0
31	3.700	20	7.0	1.0	0
32	3.360	40	7.0	1.0	0
33	3.700	15	7.0	1.5	1
34	3.700	25	5.0	1.0	0
35	3.600	10	7.0	2.0	0
36	3.825	18	7.0	1.5	0
37	3.200	15	6.0	1.0	0
38	3.500	30	8.0	3.0	1
39	3.500	11	7.0	1.5	0
40	3.000	28	6.0	1.5	0
41	3.980	4	7.0	1.5	0
42	3.700	4	5.0	1.0	1
43	3.810	25	7.5	2.5	0
44	4.000	42	5.0	1.0	0
45	3.100	3	7.0	2.0	1
46	3.400	42	9.0	2.0	1
47	3.500	25	8.0	2.0	1
48	3.650	20	6.0	2.0	0
49	3.700	7	8.0	2.0	0
50	3.100	6	8.0	1.0	0
51	4.000	20	7.0	3.0	0
52	3.350	45	6.0	2.0	0
53	3.541	30	7.5	1.5	0
54	2.900	20	6.0	3.0	0

Next, I want to know what other features are related to gpa, so I will do feature selection using SelectKBest method.

I noticed that different features have different scales, like gpa may be using 4.0 scaling. Studyweek should be hours but is the accumalate study hours of a week. Sleep night should be the hours a student sleep at night...

They are all on different scales, so I would be doing Standarizing first. The method I'm using is MinMaxScaler.

In [ ]:

```
from sklearn.preprocessing import MinMaxScaler

# Initialize the scaler
scaler = MinMaxScaler()
```

```
# Fit the scaler to the data and transform the data
df[['gpa', 'studyweek', 'sleepnight', 'out']] = scaler.fit_transform(df[['gpa', 'st

print(df)
```

```
         gpa  studyweek  sleepnight    out  gender
0   0.559322   1.000000       0.250  0.750       0
1   0.564972   0.270833       0.250  0.250       0
2   0.480226   0.270833       0.500  0.250       0
3   0.395480   0.166667       0.250  1.000       1
4   0.621469   0.479167       0.500  0.750       0
5   0.141243   0.375000       0.500  0.750       1
6   0.197740   0.270833       0.250  0.250       0
7   0.579096   0.166667       0.750  0.750       0
8   0.298305   0.208333       0.750  0.500       0
9   0.508475   0.000000       0.750  1.000       1
10  0.564972   0.166667       0.750  0.250       0
11  0.000000   0.583333       0.250  0.500       0
12  0.579096   0.583333       0.500  0.500       0
13  0.423729   0.395833       1.000  0.750       0
14  0.480226   0.166667       0.875  0.875       0
15  1.000000   0.250000       0.375  0.750       1
16  0.112994   0.208333       0.625  0.875       1
17  0.508475   0.208333       0.750  0.250       0
18  0.282486   0.041667       1.000  0.750       0
19  0.381356   0.895833       0.375  0.375       0
20  0.536723   0.083333       0.500  0.625       0
21  0.282486   0.166667       0.500  0.750       0
22  0.338983   0.208333       0.750  0.500       1
23  0.395480   0.229167       0.250  0.875       0
24  0.522599   0.687500       0.750  1.000       0
25  0.579096   0.166667       0.750  0.750       0
26  0.621469   0.791667       0.750  0.750       0
27  0.296610   0.250000       1.000  0.750       0
28  0.480226   0.583333       0.250  0.000       0
29  0.141243   0.125000       0.250  0.000       0
30  0.282486   0.125000       0.375  0.500       0
31  0.451977   0.375000       0.500  0.250       0
32  0.259887   0.791667       0.500  0.250       0
33  0.451977   0.270833       0.500  0.375       1
34  0.451977   0.479167       0.000  0.250       0
35  0.395480   0.166667       0.500  0.500       0
36  0.522599   0.333333       0.500  0.375       0
37  0.169492   0.270833       0.250  0.250       0
38  0.338983   0.583333       0.750  0.750       1
39  0.338983   0.187500       0.500  0.375       0
40  0.056497   0.541667       0.250  0.375       0
41  0.610169   0.041667       0.500  0.375       0
42  0.451977   0.041667       0.000  0.250       1
43  0.514124   0.479167       0.625  0.625       0
44  0.621469   0.833333       0.000  0.250       0
45  0.112994   0.020833       0.500  0.500       1
46  0.282486   0.833333       1.000  0.500       1
47  0.338983   0.479167       0.750  0.500       1
48  0.423729   0.375000       0.250  0.500       0
49  0.451977   0.104167       0.750  0.500       0
50  0.112994   0.083333       0.750  0.250       0
51  0.621469   0.375000       0.500  0.750       0
52  0.254237   0.895833       0.250  0.500       0
53  0.362147   0.583333       0.625  0.375       0
54  0.000000   0.375000       0.250  0.750       0
```

Now we can do Feature Selection to the dataset. The scoring function I'm using is f_regression(suitable for continous target varriables and the predictors are binary or continuous)

In [ ]:

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression

# Separate input features and target
X = df.drop('gpa', axis=1)
```

```
y = df['gpa'] # Target variable

# feature selection
selector = SelectKBest(score_func=f_regression, k=3)
fit = selector.fit(X, y)

# feature ranking
print('Top 3 features: ')
for i in range(3):
    print("%d. %s (%f)" % (i + 1, X.columns[indices[i]], fit.scores_[indices[i]]))
```

```
Top 3 features:
1. out (0.995810)
2. gender (0.215639)
3. sleepnight (0.197839)
```

From the above result we can observe that 'out' (nights going out per week) feature got a high score. Which may represent this feature has a strong relationship with gpa (target).

Next I will split the dataset for testing/training using train_test_split. For observing whether doing feature selection makes a difference, I'll do the training for only 'out' feature and all the features except 'gpa' respectively.

In [ ]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Separate input features and target
X = df[['out', 'gender', 'sleepnight', 'studyweek']]
y = df['gpa']

# Split the data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

print("Training set size:", len(X_train))
print("Test set size:", len(X_test))

model = LinearRegression()
model.fit(X_train, y_train)
```

```
Training set size: 38
Test set size: 17
```

Out[ ]:

```
▾ LinearRegression
LinearRegression()
```

In [ ]:

```
from sklearn.metrics import mean_squared_error

# predictions
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean squared error: ", mse)
```

```
Mean squared error:  0.021974176126192076
```

I trained a Linear regression model to do the prediction. The accuracy is mse = 0.021974

Now I'll be training the model with only 'out' feature.

In [ ]:

```
X = df[['out']]
y = df['gpa']

# Split the data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

print("Training set size:", len(X_train))
print("Test set size:", len(X_test))
```

```python
model = LinearRegression()
model.fit(X_train, y_train)
```

```
Training set size: 38
Test set size: 17
```

Out[ ]:

```
▾ LinearRegression

LinearRegression()
```

In [ ]:

```python
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean squared error: ", mse)


df_results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df_results)
```

```
Mean squared error:  0.02041822296962972
      Actual  Predicted
31  0.451977   0.347523
5   0.141243   0.416615
32  0.259887   0.347523
13  0.423729   0.416615
19  0.381356   0.364796
49  0.451977   0.382069
41  0.610169   0.364796
26  0.621469   0.416615
43  0.514124   0.399342
12  0.579096   0.382069
52  0.254237   0.382069
3   0.395480   0.451162
33  0.451977   0.364796
34  0.451977   0.347523
8   0.298305   0.382069
17  0.508475   0.347523
6   0.197740   0.347523
```

The mse = 0.0204. The prediction accuracy did improved a little, but don't think it's due to feature selection.