Multiple Linear Regression

```
# from google.colab import files
# up = files.upload()
```

import dataset

```
import pandas as pd
df = pd.read_csv('dataset.csv')
df.head()
```

	A	B	C	T
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244

```
# df.info()
```

cleaning

```
# clean the data
```

encoding

```
# encode the data
```

define x , y

```
import numpy as np
x = np.array(df[['A', 'B', 'C']])
y = np.array(df['T'])

# x = df[['A', 'B', 'C']].values
# y = df['T'].values

y[:5]
```

```
array([196, 221, 136, 255, 244])
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.linear_model import LinearRegression
# from sklearn.metrics import r2_score
# from sklearn.preprocessing import StandardScaler

# import time
# t1 = time.time()
# lst = []
# for i in range(1,10):
```

```
#     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=i)
#     sc = StandardScaler().fit(x_train)
#     x_train = sc.transform(x_train)
#     x_test = sc.transform(x_test)
#     mlr = LinearRegression()
#     mlr.fit(x_train, y_train)
#     yhat_test = mlr.predict(x_test)
#     r2 = r2_score(y_test, yhat_test)
#     lst.append(r2)
# t2 = time.time()
# print(f"run time: {round((t2 - t1)/60, 2)} min")
# print(f"r2_score: {round(max(lst), 2)}")
# rs = np.argmax(lst) + 1
# print(f"random state: {rs}")
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler().fit(x_train)
```

```
x_train = sc.transform(x_train)
x_test = sc.transform(x_test)
```

fit train data

```
### K-fold cross validation

# from sklearn.linear_model import LinearRegression
# from sklearn.model_selection import GridSearchCV

# parameters = {
#     'fit_intercept': [True, False],
#     'copy_X': [True, False],
#     'n_jobs': [None],
#     'positive': [True, False]
# }

# lr = LinearRegression()
# gs = GridSearchCV(estimator=lr, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

# best_params = gs.best_params_
# print(best_params)
```

```
# def param
# fit_intercept=True, copy_X=True, n_jobs=None, positive=False
```

```
from sklearn.linear_model import LinearRegression
mlr = LinearRegression()
mlr.fit(x_train, y_train)
```

```
▾ LinearRegression ⓘ ?
LinearRegression()
```

```
print(mlr.intercept_)
print(mlr.coef_)
```

```
256.5287500000001
[16.32125112 12.15819912 33.2700625 ]
```

predict test data

```
yhat_test = mlr.predict(x_test)
```

evaluate the model

```
from sklearn.metrics import r2_score
print("r2-score (train data): %0.4f" % r2_score(y_train, mlr.predict(x_train)))
print("r2-score (test data): %0.4f" % r2_score(y_test, yhat_test))
```

```
r2-score (train data): 0.8608
r2-score (test data): 0.8725
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(f"MSE (train data): {mean_squared_error(y_train, mlr.predict(x_train))}")
print(f"MAE (train data): {mean_absolute_error(y_train, mlr.predict(x_train))}")
print(f"MSE (test data): {mean_squared_error(y_test, yhat_test)}")
print(f"MAE (test data): {mean_absolute_error(y_test, yhat_test)}")
```

```
MSE (train data): 552.1298107554162
MAE (train data): 16.85683912285988
MSE (test data): 528.8568781174732
MAE (test data): 17.128396139448324
```

predict new data

```
mlr.predict(sc.transform([[2, 4, 8.5]]))
```

```
array([199.52309944])
```

save the model

```
# import joblib
# joblib.dump(mlr, 'mlr_model.pkl')
```

load the model

```
# import joblib
# mlr = joblib.load('mlr_model.pkl')
```