LightGBM Regression

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

load dataset

```
import pandas as pd
df = pd.read_csv('df.csv')
df.head(3)
```

	f1	f2	f3	f4	T
0	16.5	202.0	865.500000	1880.0	50.000000
1	18.0	204.0	688.000000	1738.5	44.000000
2	18.0	203.0	583.666667	1470.0	66.666667

cleaning

```
# clean the data
```

encoding

```
# encode the data
```

define x , y

```
import numpy as np
x = df[['f1', 'f2', 'f3', 'f4']].values
y = df['T'].values
```

```
x[:5]
```

```
array([[ 16.5      ,  202.       ,  865.5      , 1880.       ],
       [ 18.       ,  204.       ,  688.       , 1738.5      ],
       [ 18.       ,  203.       ,  583.6666667, 1470.       ],
       [ 17.       ,  201.5      ,  892.5      , 1484.       ],
       [ 31.5      ,  218.       , 1059.5      , 2065.       ]])
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from lightgbm import LGBMRegressor
# from sklearn.metrics import r2_score

# 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)
#     lgb = LGBMRegressor(random_state=1)
#     lgb.fit(x_train, y_train)
#     yhat_test = lgb.predict(x_test)
#     r2 = r2_score(y_test, yhat_test)
#     lst.append(r2)
# t2 = time.time()
```

```
# print(f"run time: {round((t2 - t1) / 60 , 0)} min")
# print(f"R2_score = {round(max(lst),2)}")
# print(f"random_state = {np.argmax(lst) + 1}")

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

```
# lightGBM Regression doesn't need scaling
```

fit train data

```
### k-fold cross validation

# from lightgbm import LGBMRegressor
# from sklearn.model_selection import GridSearchCV

# parameters = {
#     '': [],
#     '': []
# }

# lg = LGBMRegressor(random_state=1)
# gs = GridSearchCV(estimator=lg, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

# best_params = gs.best_params_
# print(best_params)


# def param
# num_leaves=31, max_depth=-1, learning_rate=0.1, boosting_type='gbdt'
# n_estimators=100, colsample_bytree=1.0, subsample_for_bin=200000
# class_weight=None, min_split_gain=0.0, min_child_weight=0.001,
# min_child_samples=20, subsample=1.0, subsample_freq=0, objective=None
# reg_alpha=0.0, reg_lambda=0.0, random_state=None, n_jobs=None
# importance_type='split', **kwargs

from lightgbm import LGBMRegressor
lgb = LGBMRegressor(random_state=1)
lgb.fit(x_train, y_train)
```

Show hidden output

predict test data

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

```
C:\Users\javad\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid featur
  warnings.warn(
```

evaluate

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

```
r2-score (train data): 0.7516
r2-score (test data): 0.3450
C:\Users\javad\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid featur
  warnings.warn(
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(f"MSE (train data): {mean_squared_error(y_train, lgb.predict(x_train))}")
print(f"MAE (train data): {mean_absolute_error(y_train, lgb.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): 46.89568438466197
MAE (train data): 5.199545361211445
MSE (test data): 95.91032944708616
MAE (test data): 7.893493740963585
C:\Users\javad\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid featur
  warnings.warn(
C:\Users\javad\AppData\Local\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid featur
  warnings.warn(
```

save the model

```
# import joblib
# joblib.dump(lgb, 'lgb_model.pkl')
```

load the model

```
# import joblib
# lgb = joblib.load('lgb_model.pkl')
```