Gradient Boosting

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

import 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']].values
y = df['T'].values
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.ensemble import GradientBoostingRegressor
# 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=0.2, random_state=i)
#     gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=1)
#     gbr.fit(x_train, y_train)
#     yhat_test = gbr.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

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

fit the model

```
### K-fold cross validation

# from sklearn.ensemble import GradientBoostingRegressor
# from sklearn.model_selection import GridSearchCV

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

# gb = GradientBoostingRegressor(random_state=1)
# gs = GridSearchCV(estimator=gb, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

# best_params = gs.best_params_
# print(best_params)


# def param
# loss='squared_error', learning_rate=0.1, n_estimators=100, subsample=1.0,
# criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
# min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0,
# init=None, random_state=None, max_features=None, alpha=0.9, verbose=0,
# max_leaf_nodes=None, warm_start=False, validation_fraction=0.1,
# n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0

from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
gbr.fit(x_train, y_train)
```

```
▾ GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

predict test data

```
yhat_test = gbr.predict(X_test)
```

Evaluate the model

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

```
r2-score (train data): 0.8758
r2-score (test data): 0.3618
```

```
print(f"MSE (train data): {mean_squared_error(y_train, gbr.predict(x_train))}")
print(f"MAE (train data): {mean_absolute_error(y_train, gbr.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): 23.444831592628613
MAE (train data): 3.7765383463895206
MSE (test data): 93.44204851804126
MAE (test data): 7.943147131116684
```

save the model

```
# import joblib
# joblib.dump(gbr, 'gbr_model.pkl')
```

load the model

```
# import joblib
# gbr = joblib.load('gbr_model.pkl')
```