XGBoost Regression

```
# 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 xgboost import XGBRegressor
# 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)
#     xgbr = XGBRegressor(random_state=1)
#     xgbr.fit(x_train, y_train)
#     yhat_test = xgbr.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 xgboost import XGBRegressor
# from sklearn.model_selection import GridSearchCV

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

# xg = XGBRegressor(random_state=42)
# gs = GridSearchCV(estimator=xg, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

# best_params = gs.best_params_
# print(best_params)

from xgboost import XGBRegressor
xgbr = XGBRegressor(random_state=559)
xgbr.fit(x_train, y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=None, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=None,
             n_jobs=None, num_parallel_tree=None, ...)
```

predict test data

```
yhat_test = xgbr.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, xgbr.predict(x_train)))
print("r2-score (test data): %0.4f" % r2_score(y_test, yhat_test))
```

```
r2-score (train data): 1.0000
r2-score (test data): 0.3266
```

```
print(f"MSE (train data): {mean_squared_error(y_train, xgbr.predict(x_train))}")
print(f"MAE (train data): {mean_absolute_error(y_train, xgbr.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): 0.0020019264347165237
MAE (train data): 0.03151735096080728
MSE (test data): 98.5982820128015
MAE (test data): 7.646535949707031
```

save the model

```
# import joblib
# joblib.dump(xgbr, 'xgbr_model.pkl')
```

load the model

```
# import joblib
# xgbr = joblib.load('xgbr_model.pkl')
```