Adaboost 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

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

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
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.ensemble import AdaBoostRegressor
# from sklearn.tree import DecisionTreeRegressor
# 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)
#     base_estimator = DecisionTreeRegressor(max_depth=3)
#     abr = AdaBoostRegressor(estimator=base_estimator, n_estimators=100, random_state=42)
#     abr.fit(x_train, y_train)
#     yhat_test = abr.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

```
# decision tree based model do not need scaling
```

fit the model

```
### K-fold cross validation

# from sklearn.ensemble import AdaBoostRegressor
# from sklearn.tree import DecisionTreeRegressor
# from sklearn.model_selection import GridSearchCV

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

# ab = AdaBoostRegressor(random_state=42)
# gs = GridSearchCV(estimator=ab, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

# best_params = gs.best_params_
# print(best_params)# from sklearn.model_selection import GridSearchCV
```

```
# def param
# estimator=None, n_estimators=50, random_state=None
# loss='linear', learning_rate=1.0
```

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor

base_estimator = DecisionTreeRegressor(max_depth=3)

abr = AdaBoostRegressor(estimator=base_estimator, n_estimators=100, random_state=42)
abr.fit(x_train, y_train)
```

```
▸ AdaBoostRegressor
                              i ?
▸ estimator:
  DecisionTreeRegressor
    ▸ DecisionTreeRegressor  ?
```

predict test data

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

evaluate the model

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

```
r2-score (train data): 0.6443
r2-score (test data): 0.3732
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(f"MSE (train data): {mean_squared_error(y_train, abr.predict(x_train))}")
print(f"MAE (train data): {mean_absolute_error(y_train, abr.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): 67.16895591769736
MAE (train data): 7.084704250947143
MSE (test data): 91.77392597007267
MAE (test data): 7.828614789194458
```

save the model

```
# import joblib
# joblib.dump(adr, 'abr_model.pkl')
```

load the model

```
# import joblib
# abr = joblib.load('abr_model.pkl')
```