Random Forest Regression

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

import dataset

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

```
     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
```

```
# 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'])
```

spliting

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=9)
```

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.ensemble import RandomForestRegressor
# 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)
#     rfr = RandomForestRegressor(random_state=1)
#     rfr.fit(x_train, y_train)
#     yhat_test = rfr.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}")
```

scaling

```
# Random Forest Regression doesn't need scaling
```

train the model

```
# def param
# n_estimators=100, max_depth=None,
# min_samples_split=2, min_samples_leaf=1, max_features=1.0
# criterion='squared_error', min_weight_fraction_leaf=0.0
# max_leaf_nodes=None, min_impurity_decrease=0.0
# bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0
# warm_start=False, ccp_alpha=0.0, max_samples=None, monotonic_cst=None

from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=300, max_depth=200, random_state=1)
rfr.fit(x_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(max_depth=200, n_estimators=300, random_state=1)
```

```
### K-fold cross validation

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

# parameters = {
#     'n_estimators': [50, 100, 150],
#     'max_depth': [50, 100, 150]
# }

# rf = RandomForestRegressor(random_state=183)
# gs = GridSearchCV(estimator=rf, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

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

predict test data

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

evaluating the moodel

```
from sklearn.metrics import r2_score
print(f"r2_score: {r2_score(y_test, yhat_test)}")
```

```
r2_score: 0.9842984781993429
```

predict new data

```
rfr.predict([[2, 4, 8.5]])
```

```
array([196.11333333])
```

feature importance

```
importances = rfr.feature_importances_
X = df[['A', 'B', 'C']]
feature_names = X.columns

print(importances)
print(feature_names)
```

```
[0.08299628 0.03870104 0.87830268]
Index(['A', 'B', 'C'], dtype='object')
```

```
fi = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
fi = fi.sort_values(by='Importance', ascending=False)
print(fi)
```

```
  Feature  Importance
2       C    0.878303
0       A    0.082996
1       B    0.038701
```

save the model

```
# import joblib
# joblib.dump(rfr, 'rfr_model.pkl')
```

load the model

```
# import joblib
# rfr = joblib.load('rfr_model.pkl')
```