Simple Linear Regression

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

import dataset

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

	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
3	3.5	6	11.1	255
4	3.5	6	10.6	244

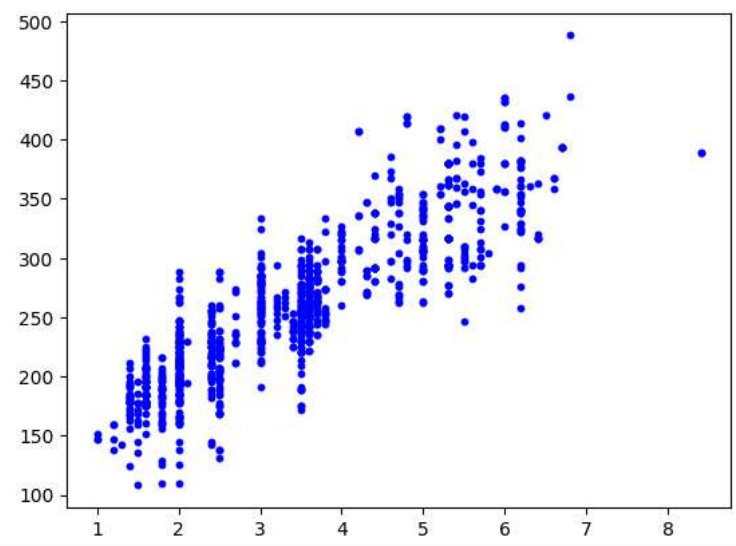
```
df = df[['A', 'T']]
df.head()
```

	A	T
0	2.0	196
1	2.4	221
2	1.5	136
3	3.5	255
4	3.5	244

```
# df.size
# df.shape
# df.info()
df.describe()
```

	A	T
count	1067.000000	1067.000000
mean	3.346298	256.228679
std	1.415895	63.372304
min	1.000000	108.000000
25%	2.000000	207.000000
50%	3.400000	251.000000
75%	4.300000	294.000000
max	8.400000	488.000000

```
import matplotlib.pyplot as plt
plt.scatter(df['A'], df['T'], s=10, color='blue')
plt.show()
```



cleaning

```
# clean the data
```

encoding

```
# encode the data
```

define x , y

```
import numpy as np
x = np.array(df[['A']])
y = np.array(df['T'])

# x = df[['A']].values
# y = df['T'].values
```

```
y[:5]
```

```
array([196, 221, 136, 255, 244])
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.linear_model import LinearRegression
# 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)
#     slr = LinearRegression()
#     slr.fit(x_train, y_train)
#     yhat_test = slr.predict(x_test)
#     r2 = r2_score(y_test, yhat_test)
#     lst.append(r2)
# t2 = time.time()
# print(f"run time: {round((t2 - t1)/60, 2)} min")
# print(f"r2_score: {round(max(lst), 2)}")
```

```
# rs = np.argmax(lst) + 1
```

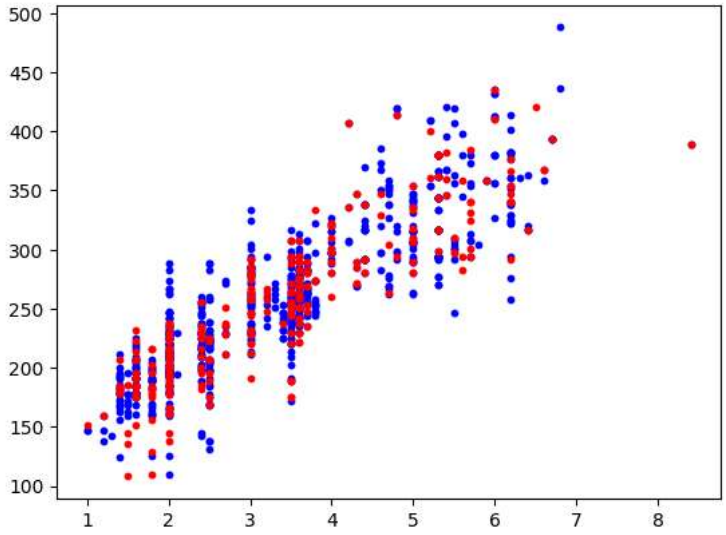
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

```
### spliting without sklearn

# msk = np.random.rand(len(df)) < 0.8
# train = df[msk]
# test = df[~msk]

# x_train = np.array(train[['ENGINESIZE']])
# x_test = np.array(test[['ENGINESIZE']])
# y_train = np.array(train[['CO2EMISSIONS']])
# y_test = np.array(test[['CO2EMISSIONS']])

plt.scatter(x_train, y_train, s=10, c='b')
plt.scatter(x_test, y_test, s=10, c='r')
plt.show()
```



scaling

```
# do not need for scaling in simple linear regression
```

fit train data

```
### K-fold cross validation

# from sklearn.linear_model import LinearRegression
# from sklearn.model_selection import GridSearchCV

# parameters = {
#     'fit_intercept': [True, False],
#     'copy_X': [True, False],
#     'n_jobs': [None],
#     'positive': [True, False]
# }

# lr = LinearRegression()
# gs = GridSearchCV(estimator=lr, param_grid=parameters, cv=5)

# gs.fit(x_train, y_train)

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

```
# def param
# copy_X=True, fit_intercept=True, n_jobs=None, positive=False
```

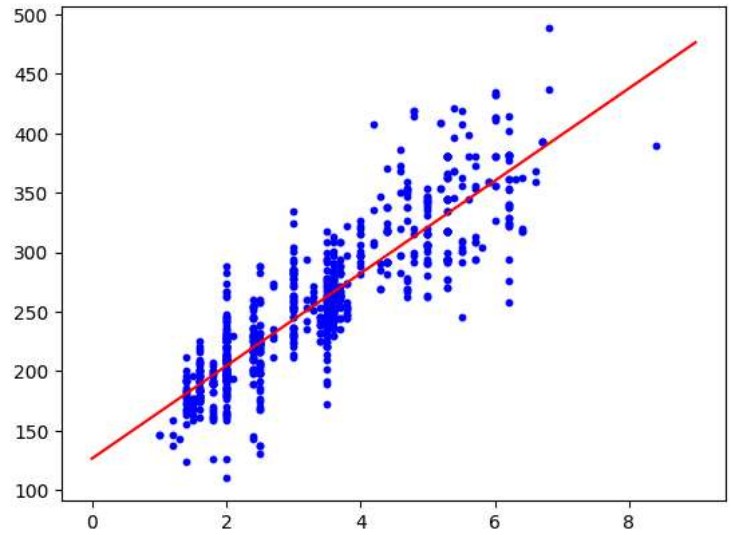
```
from sklearn.linear_model import LinearRegression
slr = LinearRegression()
slr.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

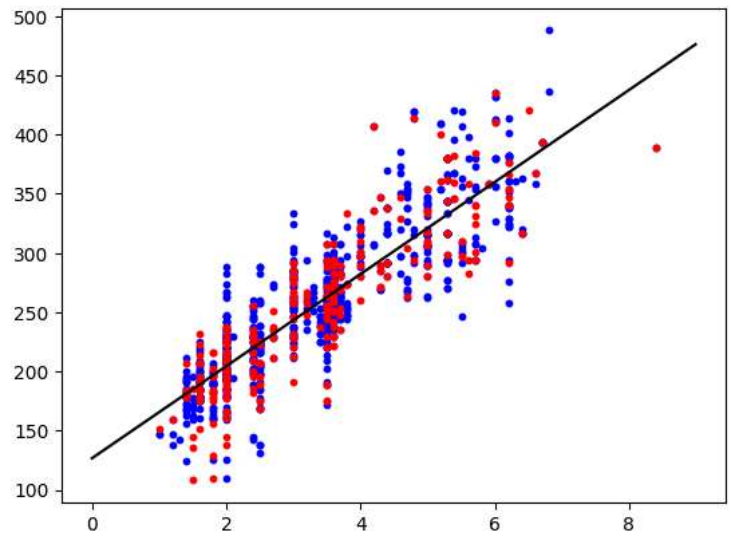
```
print(slr.intercept_)
print(slr.coef_)
```

```
126.62361300856665
[38.89375359]
```

```
xx = np.arange(0, 9, 0.01)
plt.scatter(x_train, y_train, s=10, c='b')
plt.plot(xx, slr.intercept_ + slr.coef_[0] * xx, c='r')
plt.show()
```




```
xx = np.arange(0,9,0.01)
plt.scatter(x_train, y_train, s=10, c='b')
plt.plot(xx, slr.intercept_ + slr.coef_[0] * xx, c='black')
plt.scatter(x_test, y_test, c='r', s=10)
plt.show()
```



predict test data

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

evaluate the model

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

```
r2-score (train data): 0.7635
r2-score (test data): 0.7656
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
print(f"MSE (train data): {mean_squared_error(y_train, slr.predict(x_train))}")
print(f"MAE (train data): {mean_absolute_error(y_train, slr.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): 938.2412248261459
MAE (train data): 23.238294619464767
MSE (test data): 972.1181539625654
MAE (test data): 23.804281919112945
```

```
### evaluate without sklearn

# print("MSE: %0.2f" % np.mean((y_test - yhat_test) ** 2))
# print("MAE: %0.2f" % np.mean(np.absolute(y_test - yhat_test)))
```

predict new data

```
slr.predict([[0]])
```

```
array([126.62361301])
```

save the model

```
# import joblib
# joblib.dump(slr, 'slr_model.pkl')
```

load the model

```
# import joblib
# slr = joblib.load('slr_model.pkl')
```