32
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
%matplotlib inline
```

3
```
data_path = os.path.join(os.getcwd(), 'data', 'regression_part1.csv')
regression_dataset = pd.read_csv(data_path, delimiter = ',')
regression_dataset.head(5)
```

3

	revision_time	exam_score
0	2.723	27.682
1	2.844	22.998
2	3.303	19.765
3	3.588	24.481
4	4.050	22.974

4
```
regression_dataset.describe()
```

4

	revision_time	exam_score
count	50.000000	50.000000
mean	22.220020	49.919860
std	13.986112	20.925594
min	2.723000	14.731000
25%	8.570500	32.125000
50%	21.688000	47.830500
75%	32.257500	65.069750
max	48.011000	94.945000

6
```
regression_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   revision_time  50 non-null     float64
 1   exam_score     50 non-null     float64
dtypes: float64(2)
memory usage: 928.0 bytes
```

84
```
model_x = regression_dataset['revision_time']
```

```
model_y = regression_dataset['exam_score']

# x_train, x_test, y_train, y_test = train_test_split(model_x, model_y, train_size=0.75, test_size=0.

x_train = np.array([[1,x] for x in model_x])
# x_test = np.array([[1,x] for x in x_test])

y_train = np.array(model_y)
```

```
lm = LinearRegression(fit_intercept=False)
lm.fit(x_train, y_train)
print('Training accuracy: {:.3f}'.format(lm.score(x_train, y_train)))
```

```
Training accuracy: 0.928
```

```
# print('Testing accuracy by using score function: {:.3f}'.format(lm.score(x_test, y_test)))
# print('Testing accuracy by using r2_score meric: {:.3f}'.format(r2_score(y_test, lm.predict(x_test)
```

```
Testing accuracy by using score function: 0.941
Testing accuracy by using r2_score meric: 0.941
```

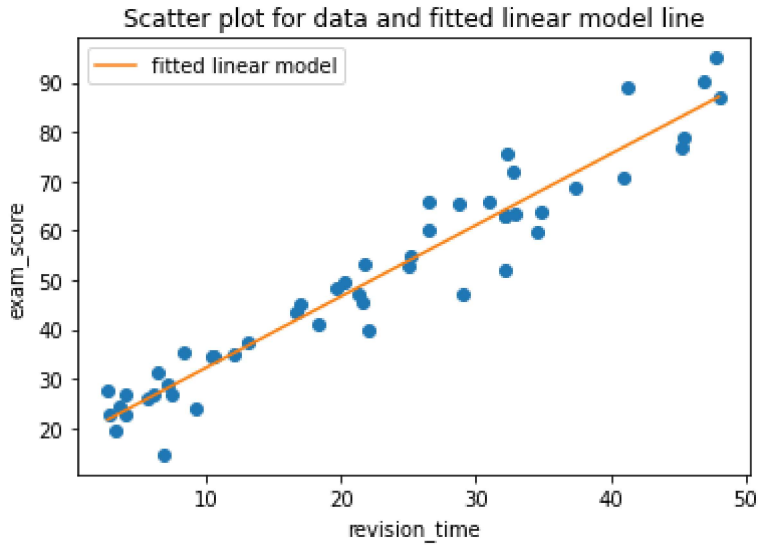
```
lm.coef_
```

```
array([17.89768026,  1.44114091])
```

```
prediction_y = lm.predict(np.array([[1,x] for x in model_x]))
```

```
fig, ax = plt.subplots()
ax.scatter(model_x, model_y)
ax.plot([model_x.min(),model_x.max()], [prediction_y.min(), prediction_y.max()], color='tab:orange',
ax.legend()
plt.title('Scatter plot for data and fitted linear model line')
ax.set_xlabel('revision_time')
ax.set_ylabel('exam_score')
```

```
Text(0, 0.5, 'exam_score')
```




```
phi_w = np.dot(np.linalg.pinv(x_train) , y_train)
```

```
phi_w
```

```
array([17.89768026,  1.44114091])
```

```
pseudo_inverse = np.linalg.inv(np.dot(x_train.T,x_train)),
pseudo_inverse = np.dot(pseudo_inverse, x_train.T)
phi_w = np.dot(pseudo_inverse, y_train)
phi_w
```

```
array([[17.89768026,  1.44114091]])
```