# Chapter 2 - Exercise 6: Petrol consumption

## Cho dữ liệu petrol_consumption.csv

## Hãy áp dụng Linear Regression để dự đoán Petrol_Consumption dựa trên Petrol_tax, Average_income, Paved_Highways, Population_Driver_licence(%)

1. Đọc dữ liệu. Xem thông tin data về dữ liệu. Chuẩn hóa dữ liệu nếu cần.
2. Vẽ viểu đồ quan sát mối liên hệ giữa các biến Petrol_Consumption, Petrol_tax, Average_income, Paved_Highways, Population_Driver_licence(%)
3. Tạo inputs data và outputs data => Tạo X_train, X_test, y_train, y_test với tỷ lệ 80:20
4. Thực hiện Linenear Regression với X_train, y_train
5. Dự đoán y từ X_test => so sánh với y_test
6. Tính Coefficients, Intercept và Variance score, MSE
7. Vẽ hình và xem kết quả
8. Nhận xét dựa trên kết quả, có giải pháp nào để kết quả tốt hơn không?

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter2_Linear_Reg
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
data = pd.read_csv("petrol_consumption.csv", sep=",")
```

In [4]:
```
print(data.shape)
type(data)
```

```
(48, 5)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

In [5]:
```
data.dtypes
```

Out[5]:
```
Petrol_tax                    float64
Average_income                  int64
Paved_Highways                  int64
Population_Driver_licence(%)  float64
Petrol_Consumption              int64
dtype: object
```

In [6]:
```
data.head()
```

Out[6]:

| | Petrol_tax | Average_income | Paved_Highways | Population_Driver_licence(%) | Petrol_Consumption |
|---|---|---|---|---|---|
| 0 | 9.0 | 3571 | 1976 | 0.525 | 541 |
| 1 | 9.0 | 4092 | 1250 | 0.572 | 524 |
| 2 | 9.0 | 3865 | 1586 | 0.580 | 561 |
| 3 | 7.5 | 4870 | 2351 | 0.529 | 414 |
| 4 | 8.0 | 4399 | 431 | 0.544 | 410 |

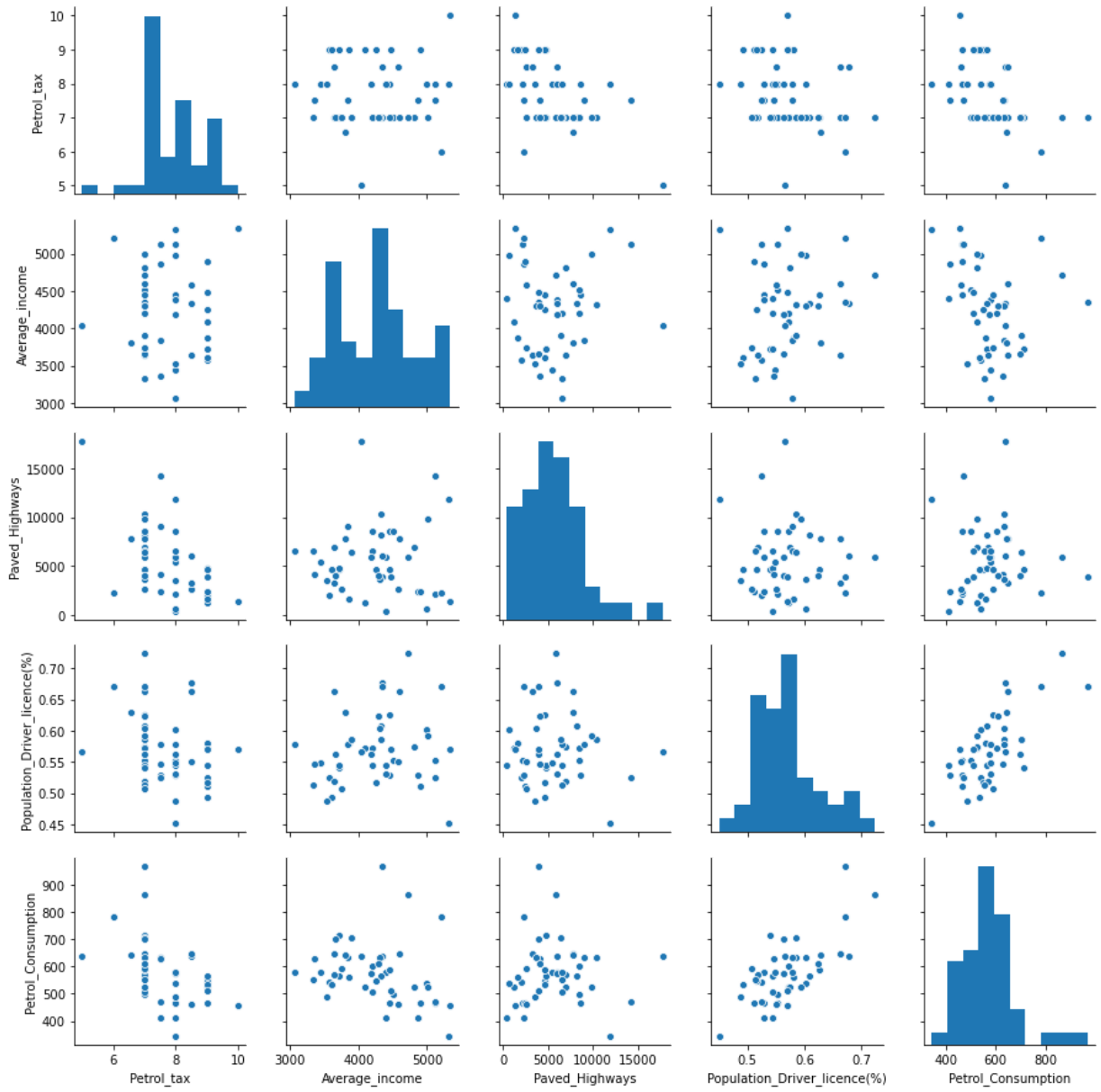
In [7]:
```
data.describe()
```

Out[7]:

| | Petrol_tax | Average_income | Paved_Highways | Population_Driver_licence(%) | Petrol_Consumpt |
|---|---|---|---|---|---|
| count | 48.000000 | 48.000000 | 48.000000 | 48.000000 | 48.000 |
| mean | 7.668333 | 4241.833333 | 5565.416667 | 0.570333 | 576.770 |
| std | 0.950770 | 573.623768 | 3491.507166 | 0.055470 | 111.885 |
| min | 5.000000 | 3063.000000 | 431.000000 | 0.451000 | 344.000 |
| 25% | 7.000000 | 3739.000000 | 3110.250000 | 0.529750 | 509.500 |
| 50% | 7.500000 | 4298.000000 | 4735.500000 | 0.564500 | 568.500 |
| 75% | 8.125000 | 4578.750000 | 7156.000000 | 0.595250 | 632.750 |
| max | 10.000000 | 5342.000000 | 17782.000000 | 0.724000 | 968.000 |

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
sns.pairplot(data)
plt.show()
```



In [10]:
```
inputs = data[['Petrol_tax', 'Average_income', 'Paved_Highways',
       'Population_Driver_licence(%)']]
inputs.head()
```

Out[10]:

| | Petrol_tax | Average_income | Paved_Highways | Population_Driver_licence(%) |
|---|---|---|---|---|
| 0 | 9.0 | 3571 | 1976 | 0.525 |
| 1 | 9.0 | 4092 | 1250 | 0.572 |
| 2 | 9.0 | 3865 | 1586 | 0.580 |
| 3 | 7.5 | 4870 | 2351 | 0.529 |
| 4 | 8.0 | 4399 | 431 | 0.544 |

In [11]:
```
outputs = data['Petrol_Consumption']
outputs.head()
```

Out[11]:
```
0    541
1    524
2    561
3    414
4    410
Name: Petrol_Consumption, dtype: int64
```

In [12]:
```
from sklearn.model_selection import train_test_split
from sklearn import datasets, linear_model, metrics
from sklearn.metrics import mean_squared_error, r2_score
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
                                                    test_size=0.20,
                                                    random_state = 42)
regr = linear_model.LinearRegression()
regr = regr.fit(X_train, y_train)
```

In [14]:
```
y_pred = regr.predict(X_test)
```

In [15]:
```
df = pd.DataFrame({'Actual': pd.Series(y_test.values),
                   'Prediction': pd.DataFrame(y_pred)[0].values})
df.head()
```

Out[15]:

| | Actual | Prediction |
|---|---|---|
| 0 | 631 | 606.692665 |
| 1 | 587 | 673.779442 |
| 2 | 577 | 584.991490 |
| 3 | 591 | 563.536910 |
| 4 | 460 | 519.058672 |

In [16]:
```
# The coefficients
print('Coefficients: \n', regr.coef_)
print('Intercept: \n', regr.intercept_)
```

```
Coefficients:
 [-3.69937459e+01 -5.65355145e-02 -4.38217137e-03  1.34686930e+03]
Intercept:
 361.45087906653225
```

In [17]:
```
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(outputs, regr.predict(inputs)))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(inputs, outputs))
```

```
Mean squared error: 4029.43
Variance score: 0.67
```

''' Ta có thể thấy rằng mô hình chỉ khái quát được ~67% dữ liệu, MSE lớn => Điều này có nghĩa là thuật toán không chính xác lắm (nhưng vẫn có thể đưa ra những dự đoán hợp lý) Một số yếu tố có thể đã góp phần vào sự không chính xác này, đó là:

- Cần thêm dữ liệu: Chỉ có một năm giá trị của dữ liệu nên chưa đủ lớn
- Giả định xấu: khi đưa giả thiết rằng dữ liệu này có mối quan hệ tuyến tính => nhưng điều đó có thể không đúng. (Trực quan hóa dữ liệu có thể giúp ta xác định điều này)
- Tính năng kém: Các tính năng sử dụng có thể không có tương quan đủ cao với các giá trị ta đang cố gắng dự đoán. '''

In [18]:
```
print('Variance score: %.2f' % metrics.r2_score(y_test, y_pred))
```

```
Variance score: 0.39
```

In [19]:
```
# Check the score of train and test
regr.score(X_train, y_train)
```

Out[19]:
```
0.7068781342155135
```

In [20]:
```
regr.score(X_test, y_test)
```

Out[20]:
```
0.3913664001428886
```

Chúng ta thấy R^2 cho dữ liệu huấn luyện là 0.70 trong khi R^2 trên dữ liệu thử nghiệm là 0.39. (Chú ý: R^2 càng thấp, mô hình càng tệ, R^2 âm là dấu hiệu của overfitting).
Cần suy nghĩ về các thuộc tính áp dụng để huấn luyện mô hình, lựa chọn lại nếu cần!

In [21]:
```
# Trực quan hóa kết quả
```

In [22]:
```
# Có giải pháp nào tốt hơn không???
```

# Polynominal

In [23]:
```
from sklearn.preprocessing import PolynomialFeatures
```

In [24]:
```
pr=PolynomialFeatures(degree=2)
pr
```

Out[24]:
```
PolynomialFeatures(degree=2, include_bias=True, interaction_only=False,
                   order='C')
```

In [25]:
```
X_pr=pr.fit_transform(inputs)
```

In [26]:
```
inputs.shape, X_pr.shape
```

Out[26]:
```
((48, 4), (48, 15))
```

In [27]:
```
X_train_n, X_test_n, y_train_n, y_test_n = train_test_split(X_pr,
                                                            outputs,
                                                            test_size=0.20,
                                                            random_state = 42)
regr_n = linear_model.LinearRegression()
regr_n = regr.fit(X_train_n, y_train_n)
```

In [28]:
```
y_pred_n = regr_n.predict(X_test_n)
```

In [29]:
```
df_n = pd.DataFrame({'Actual': pd.Series(y_test_n.values),
                     'Prediction': pd.DataFrame(y_pred_n)[0].values})
df_n.head()
```

Out[29]:

| | Actual | Prediction |
|---|---|---|
| 0 | 631 | 598.219346 |
| 1 | 587 | 684.951003 |
| 2 | 577 | 572.904670 |
| 3 | 591 | 580.529789 |
| 4 | 460 | 516.369765 |

In [30]:
```
# The coefficients
print('Coefficients: \n', regr_n.coef_)
print('Intercept: \n', regr_n.intercept_)
```

```
Coefficients:
 [ 0.00000000e+00  1.20673536e+02 -4.13821517e-01 -1.20872791e-02
  3.10230319e+03 -8.21479991e-01  5.97669721e-02  4.85835272e-03
 -7.28338629e+02 -6.11311084e-05  4.90434144e-06  6.13752963e-01
  1.37119613e-06 -1.19102673e-01  1.62695637e+03]
Intercept:
 86.34273742943338
```

In [31]:
```
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(outputs, regr_n.predict(X_pr)))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr_n.score(X_pr, outputs))
```

```
Mean squared error: 2694.08
Variance score: 0.78
```

''' Ta có thể thấy rằng mô hình chỉ khái quát được ~78% dữ liệu (tăng so với model cũ, MSE đã giảm gần 1/2 => Dùng Polynorminal đã tốt hơn '''

In [32]:
```
# Check the score of train and test
regr_n.score(X_train_n, y_train_n)
```

Out[32]:
```
0.8344530112544698
```

In [33]:
```
regr_n.score(X_test_n, y_test_n)
```

Out[33]:
```
0.35693145973406426
```

Chúng ta thấy R^2 cho dữ liệu huấn luyện là 0.83 trong khi R^2 trên dữ liệu thử nghiệm là 0.35.
=> Overfitting (Chú ý: R^2 càng thấp, mô hình càng tệ, R^2 âm là dấu hiệu của overfitting).
Cần suy nghĩ về các thuộc tính áp dụng để huấn luyện mô hình, lựa chọn lại nếu cần!

In [34]:
```
y_train_hat_n = regr_n.predict(X_train_n)
y_test_hat_n = regr_n.predict(X_test_n)
```

In [35]:
```
plt.figure(figsize=(10,5))
plt.subplot(1, 2, 1)
ax1 = sns.distplot(y_train_n, hist=False, color="b", label='Train Actual')
sns.distplot(y_train_hat_n, hist=False, color="r", label='Train Predict', ax=ax1)
plt.subplot(1,2,2)
ax2 = sns.distplot(y_test_n, hist=False, color="b", label='Test Actual')
sns.distplot(y_test_hat_n, hist=False, color="r", label='Test Predict', ax=ax2)
plt.show()
```