SIMPLE LINEAR REGRESSION

RAZA BAKIR SHAH

PGD DATA SCIENCES & AI (BATCH 6)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import random_state
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```
# Load the training data
df_income = pd.read_csv('canada_per_capita_income.csv')
```

In [3]:
```
# Lets understand the what is available in the data

df_income.head()
```

Out[3]:

	year	per capita income (US$)
0	1970	3399.299037
1	1971	3768.297935
2	1972	4251.175484
3	1973	4804.463248
4	1974	5576.514583

In [4]:
```
# Splitting variables

X = df_income.iloc[:, :1]  # independent
y = df_income.iloc[:, 1:]  # dependent
```

In [5]:
```
X[:5]
```

Out[5]:

	year
0	1970
1	1971
2	1972
3	1973
4	1974

In [6]:
```
y[:5]
```

Out[6]:

	per capita income (US$)
0	3399.299037
1	3768.297935
2	4251.175484
3	4804.463248
4	5576.514583

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, randc
```

In [8]:
```
# Create and train the linear regression model

model = LinearRegression()
model.fit(X_train, y_train)
```

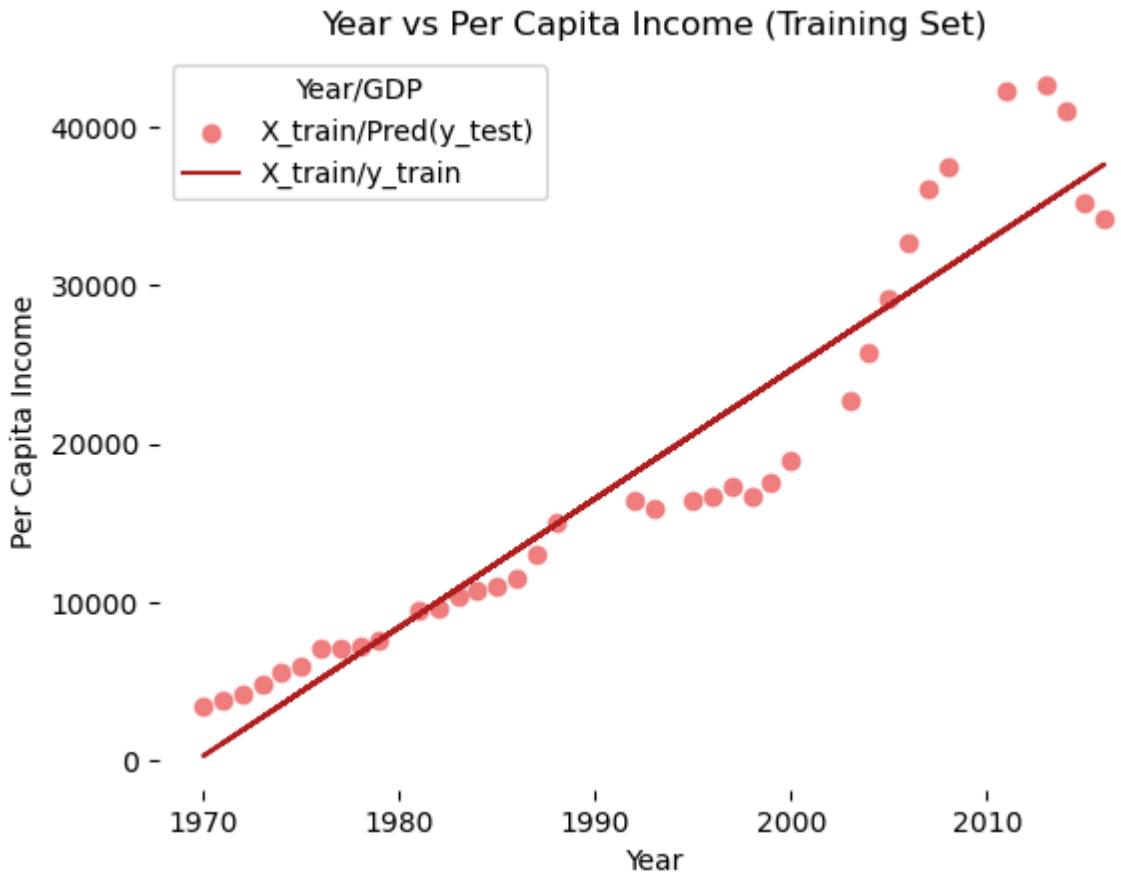
Out[8]:

LinearRegression

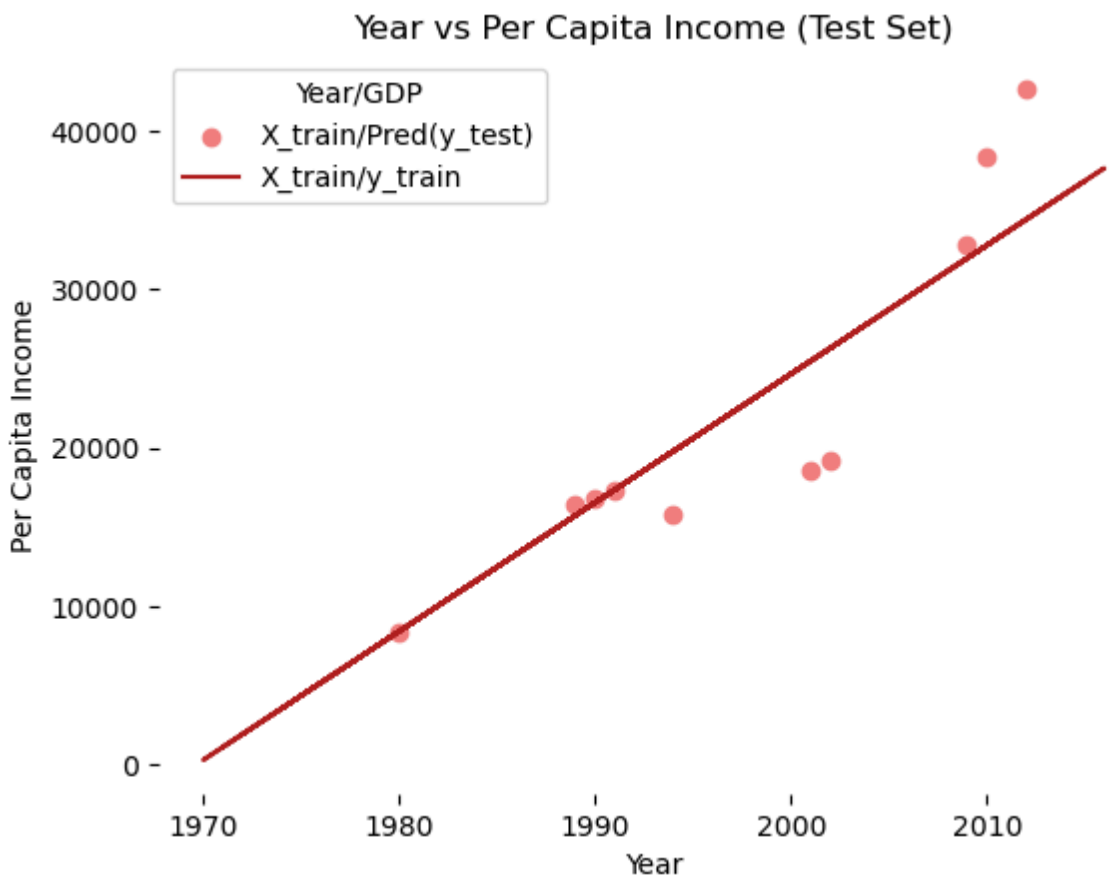
```
LinearRegression()
```

In [9]:
```
# Prediction result
y_pred_test = model.predict(X_test)     # predicted value of y_test
y_pred_train = model.predict(X_train)   # predicted value of y_train
```

In [10]:
```
# Prediction on training set
plt.scatter(X_train, y_train, color = 'lightcoral')
plt.plot(X_train, y_pred_train, color = 'firebrick')
plt.title('Year vs Per Capita Income (Training Set)')
plt.xlabel('Year')
plt.ylabel('Per Capita Income')
plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title = 'Year/GDP', loc=
plt.box(False)
plt.show()
```



In [11]:
```
# Prediction on test set
plt.scatter(X_test, y_test, color = 'lightcoral')
plt.plot(X_train, y_pred_train, color = 'firebrick')
plt.title('Year vs Per Capita Income (Test Set)')
plt.xlabel('Year')
plt.ylabel('Per Capita Income')
plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title = 'Year/GDP', loc=
plt.box(False)
plt.show()
```



In [12]:
```
# take out model intercept and slop, make an equation
print(model.intercept_)
print(model.coef_)
print('y = ', model.intercept_, '+', model.coef_, '* X')
```

```
[-1599840.69535437]
[[812.24857662]]
y =  [-1599840.69535437] + [[812.24857662]] * X
```

In [13]:
```
# evaluate the model
print('MSE = ', mean_squared_error(y_test, y_pred_test))
print('R2 = ', r2_score(y_test, y_pred_test))
print('RMSE = ', np.sqrt(mean_squared_error(y_test, y_pred_test)))
```

```
MSE =  21436921.628820017
R2 =  0.8107683170479373
RMSE =  4630.002335725115
```

Training Model from Separate File

In [14]:
```
# Load the data for which predictions are needed

df_years = pd.read_csv('years.csv')
```

In [15]:
```
# Lets understand the data we will use for predictions

df_years.head()
```

Out[15]:

	year
0	1978
1	1979
2	1980
3	1981
4	1982

In [16]:
```
# Lets define the test data we will use for predictions

X2_test = df_years.iloc[:, :1]
```

In [17]:
```
X2_test.head()
```

Out[17]:

	year
0	1978
1	1979
2	1980
3	1981
4	1982

In [18]:
```
predictions = model.predict(X2_test)
```

In [19]:
```
predictions
```

Out[19]:
```
array([[ 6786.98919983],
       [ 7599.23777645],
       [ 8411.48635307],
       [ 9223.73492969],
       [10035.98350631],
       [18158.46927251],
       [18970.71784913],
       [19782.96642575],
       [20595.21500237],
       [21407.46357899],
       [22219.71215561],
       [23031.96073223],
       [23844.20930885],
       [24656.45788547],
       [25468.70646209],
       [26280.95503871],
       [33591.19222829],
       [34403.44080491],
       [35215.68938153],
       [36027.93795815],
       [36840.18653477],
       [37652.43511139],
       [38464.68368801],
       [39276.93226463],
       [40089.18084125],
       [40901.42941787],
       [41713.67799449]])
```

In [20]:
```
df_predictions = pd.DataFrame({
    'year': df_years['year'].tolist(),
    'predicted per capita income (US$)': predictions.tolist()
})
```

In [21]:
```
df_predictions.head()
```

Out[21]:

	year	predicted per capita income (US$)
0	1978	[6786.9891998297535]
1	1979	[7599.237776449649]
2	1980	[8411.486353069544]
3	1981	[9223.734929689439]
4	1982	[10035.983506309334]

In [22]:
```
# Predictions are saved in a separate file "prediction.csv"

df_predictions.to_csv('prediction.csv', index=False)
```