Chapter 2 - Demo Linear Regression

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter2_Linear_Reg
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df = pd.read_csv('location_rev.csv')
df.head()
```

Out[4]:

	revenue	num_competitors	median_income	num_loyalty_members	population_density	location_
0	42247.80	3.0	30527.57	1407.0	3302.0	
1	38628.37	3.0	30185.49	1025.0	4422.0	
2	39715.16	1.0	32182.24	1498.0	3260.0	
3	35593.30	5.0	29728.65	2340.0	4325.0	
4	35128.18	4.0	30691.17	847.0	3774.0	

In [5]:
```
df.describe()
```

Out[5]:

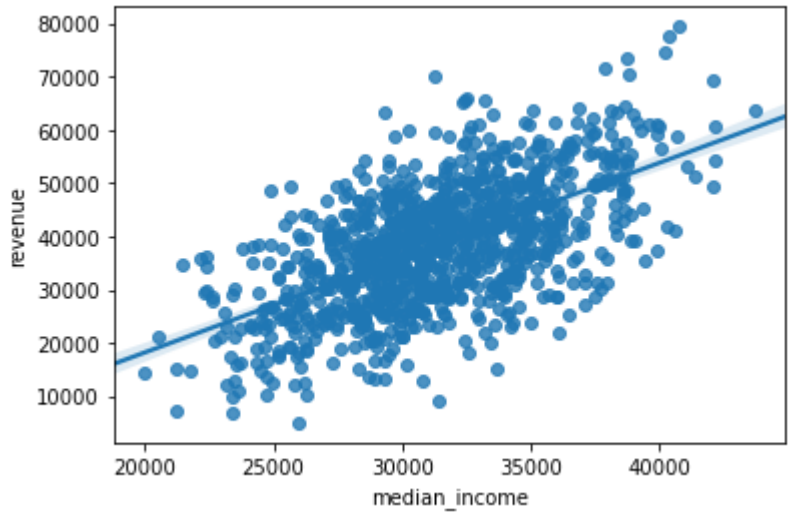
	revenue	num_competitors	median_income	num_loyalty_members	population_density
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	38433.469330	3.397000	31360.668500	1597.200000	3351.199000
std	11665.825242	1.016082	3943.278358	496.874663	975.664263
min	5000.000000	0.000000	20000.000000	0.000000	0.000000
25%	30277.897500	3.000000	28792.592500	1253.000000	2689.250000
50%	38323.095000	3.000000	31134.555000	1605.000000	3353.000000
75%	45894.670000	4.000000	34050.992500	1925.250000	4017.000000
max	79342.070000	7.000000	43676.900000	3280.000000	6489.000000

In [6]:
```
df.corr()
```

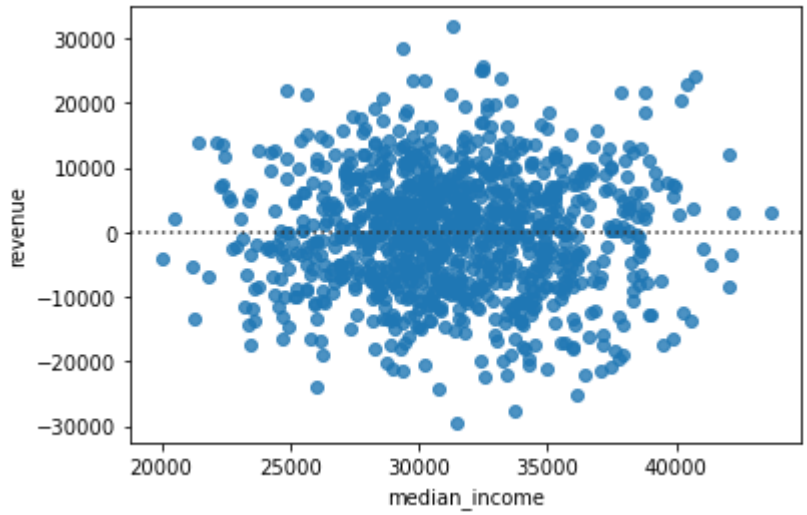
Out[6]:

	revenue	num_competitors	median_income	num_loyalty_members	populatio
revenue	1.000000	-0.156685	0.601888	0.173432	
num_competitors	-0.156685	1.000000	-0.018398	-0.027283	
median_income	0.601888	-0.018398	1.000000	0.011891	
num_loyalty_members	0.173432	-0.027283	0.011891	1.000000	
population_density	0.311653	0.035768	-0.041697	-0.028611	
location_age	0.552773	0.053796	0.045621	0.036016	

In [7]:
```
sns.regplot(data=df, x= 'median_income', y='revenue')
plt.show()
```



In [8]:
```
sns.residplot(df.median_income, df.revenue)
plt.show()
```



In [9]:
```
# Chuẩn bị dữ liệu training data/ test data
```

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
X1 = df[['median_income']]
y1 = df['revenue']
```

In [12]:
```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,
                                                        y1,
                                                        random_state = 42)
```

In [13]:
```
# Load module, tạo đối tượng linear regression
```

In [14]:
```
from sklearn.linear_model import LinearRegression
```

In [15]:
```
model = LinearRegression()
model.fit(X1_train,y1_train)
```

Out[15]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [16]:
```
y1_pred = model.predict(X1_test)
```

In [17]:
```
intercept = model.intercept_
slope = model.coef_[0]
```

In [18]:
```
print(intercept, slope)
```

```
-17944.393212607967 1.8044776691444901
```

In [19]:
```
# Đánh giá model
```

In [20]:
```
# R^2 cho toàn bộ dữ liệu
model.score(X1, y1)
```

Out[20]:
```
0.3618738533973396
```

In [21]:
```
# R^2 khi train
model.score(X1_train, y1_train)
```

Out[21]:
```
0.37376263809946475
```

In [22]:
```
# R^2 khi test
model.score(X1_test, y1_test)
```

Out[22]:
```
0.3246804918433005
```

In [23]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [24]:
```
mse = mean_squared_error(y1_pred, y1_test)
print(mse)
```

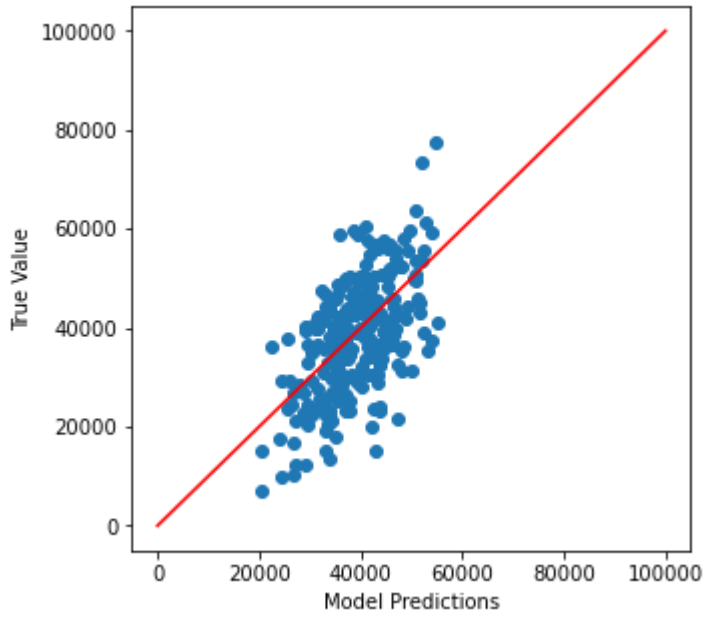
```
90952579.47704405
```

In [25]:
```
mae = mean_absolute_error(y1_pred, y1_test)
print(mae)
```

```
7751.936386800821
```

- Đặt giới hạn trục x và y là 10.000 để chúng ta có được cái nhìn rõ hơn về vị trí của hầu hết các điểm dữ liệu.
- Thêm line có độ dốc 1 đóng vai trò là tham chiếu. Nếu tất cả các điểm nằm trên line này, điều đó có nghĩa là có một mối quan hệ hoàn hảo giữa thực tế và dự đoán.

In [26]:
```
plt.figure(figsize=(5,5))
plt.scatter(model.predict(X1_test),y1_test)
plt.xlabel('Model Predictions')
plt.ylabel('True Value')
plt.plot([0, 100000], [0, 100000], 'k-', color = 'r')
plt.show()
```



In [27]:
```
# Huấn luyện model
```

In [28]:
```
# Multiple Linear Regression
```

In [29]:
```
from sklearn.model_selection import train_test_split
```

In [30]:
```
X = df[['num_competitors',
        'median_income',
        'num_loyalty_members',
        'population_density',
        'location_age'
        ]]
y = df['revenue']
```

In [31]:
```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state = 100)
```

In [32]:
```
from sklearn.linear_model import LinearRegression
```

In [33]:
```
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[33]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [34]:
```
y_pred = model.predict(X_test)
```

In [35]:
```
model.intercept_
```

Out[35]:
```
-51068.63644236374
```

In [36]:
```
model.coef_
```

Out[36]:
```
array([-2.14765128e+03,  1.71903196e+00,  3.50665069e+00,  4.31777912e+00,
        2.06703103e+03])
```

In [37]:
```
# Đánh giá model
```

In [38]:
```
model.score(X, y)
```

Out[38]:
```
0.8132062485664424
```

In [39]:
```
model.score(X_train, y_train)
```

Out[39]:
```
0.8107126447396588
```

In [40]:
```
model.score(X_test, y_test)
```

Out[40]:
```
0.8210733500078611
```

In [41]:
```
mse = mean_squared_error(y_pred, y_test)
print(mse)
```

```
23181637.891943764
```

In [42]:
```
mae = mean_absolute_error(y_pred, y_test)
print(mae)
```

```
3930.4052290167588
```

In [43]:
```
X_new = pd.DataFrame({
    'num_competitors': [3],
    'median_income': [30000],
    'num_loyalty_members': [1200],
    'population_density': [2000],
    'location_age': [10]
})
```

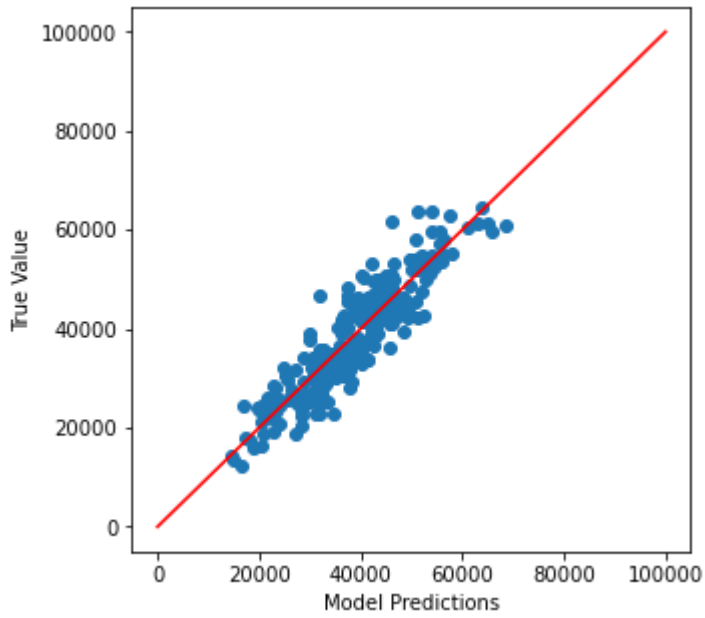
In [44]:
```
y_new = model.predict(X_new)
print(y_new)
```

```
[27573.21782447]
```

In [45]:
```
X.mean()
```

Out[45]:
```
num_competitors            3.3970
median_income          31360.6685
num_loyalty_members     1597.2000
population_density      3351.1990
location_age              11.0410
dtype: float64
```

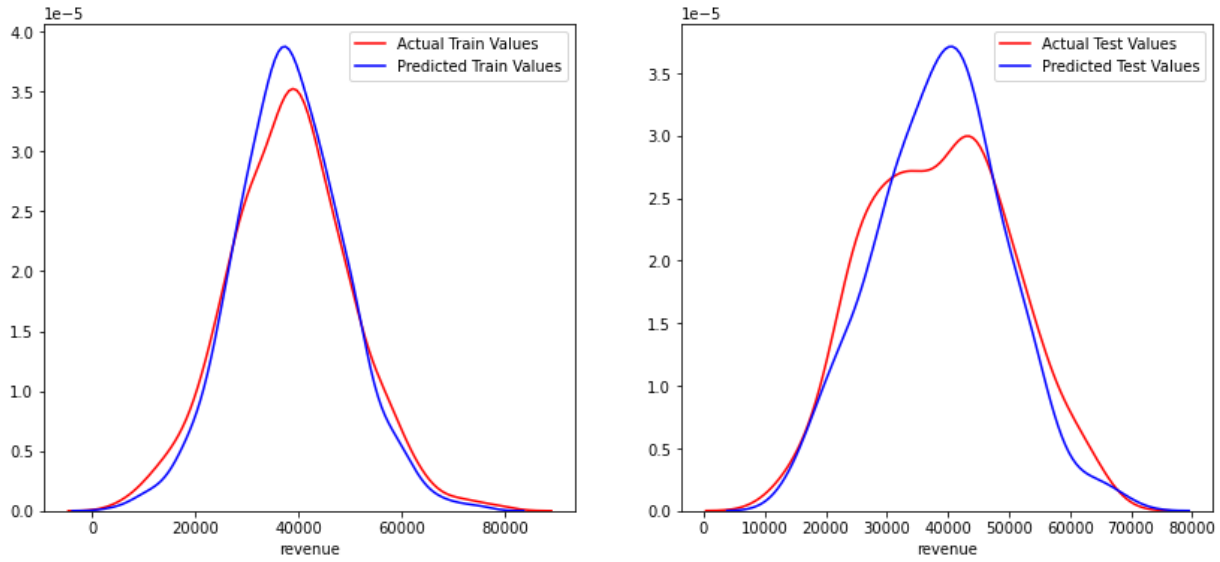
In [46]:
```
plt.figure(figsize=(5,5))
plt.scatter(model.predict(X_test),y_test)
plt.xlabel('Model Predictions')
plt.ylabel('True Value')
plt.plot([0, 100000], [0, 100000], 'k-', color = 'r')
plt.show()
```



In [47]:
```
plt.figure(figsize=(14,6))
plt.subplot(1,2,1)
ax1 = sns.distplot(y_train, hist=False, color="r",
                   label="Actual Train Values")
sns.distplot(model.predict(X_train), hist=False, color="b",
             label="Predicted Train Values", ax=ax1)

plt.subplot(1,2,2)
ax2 = sns.distplot(y_test, hist=False, color="r",
                   label="Actual Test Values")
sns.distplot(model.predict(X_test), hist=False, color="b",
             label="Predicted Test Values" , ax=ax2)

plt.show()
```



In [48]:
```
from scipy.stats.stats import pearsonr
```

In [49]:
```
pearsonr(model.predict(X_test),y_test)
```

Out[49]:
```
(0.9061597827907564, 1.1552714895195607e-94)
```

Lựa chọn thuộc tính

In [50]:
```
# Univariate Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
```

In [51]:
```
# Apply SelectKBest class to extract all best features
bestfeatures = SelectKBest(score_func=f_regression, k='all')
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

In [52]:
```
# Concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
# Sorting in descending order
featureScores.sort_values("Score", ascending = False, inplace = True)
print(featureScores)
```

```
                Specs       Score
1       median_income  566.922357
4        location_age  439.125397
3  population_density  107.360798
2  num_loyalty_members   30.949544
0     num_competitors   25.117590
```

In [53]:
```
# Correlation Matrix with Heatmap
corrmat = df.corr()
top_corr_features = corrmat.index
```

In [54]:
```
corrmat
```

Out[54]:

	revenue	num_competitors	median_income	num_loyalty_members	populatio
revenue	1.000000	-0.156685	0.601888	0.173432	
num_competitors	-0.156685	1.000000	-0.018398	-0.027283	
median_income	0.601888	-0.018398	1.000000	0.011891	
num_loyalty_members	0.173432	-0.027283	0.011891	1.000000	
population_density	0.311653	0.035768	-0.041697	-0.028611	
location_age	0.552773	0.053796	0.045621	0.036016	

In [55]:
```
import matplotlib
matplotlib.__version__
```

Out[55]:
```
'3.3.0'
```

In [56]:
```
plt.figure(figsize=(15,7))
# plot heat map
g=sns.heatmap(df[top_corr_features].corr(),cmap="RdYlGn", annot=True)
# annot=True: nếu muốn in cả giá trị
plt.show()
```