Day3

January 8, 2019

Multiple linear regression. Instead of model relationship between single dimensional fearute X and Y, we inlcude multiple features, denoted as $x_1, x_2, \cdots, x_n$ - simple linear regression: $y = b_0 + b_1 x$ - multiple linear regression: $y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$

0.1 Step 1: Date preprocessing

- import modules

```
In [10]: import pandas as pd
         import numpy as np
```

- import dataset

```
In [11]: dataset = pd.read_csv('../datasets/50_Startups.csv')
         print(dataset.head())
         X = dataset.iloc[:, :-1].values
         Y = dataset.iloc[:, 4].values
         print("Original data shape X:{}, Y:{}".format(X.shape, Y.shape))
```

```
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
Original data shape X:(50, 4), Y:(50,)
```

- encoding categorical data

```
In [12]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
         labelencoder =LabelEncoder()
         X[:, 3] = labelencoder.fit_transform(X[:, 3])
         onehotencoder = OneHotEncoder(categorical_features=[3])
         X = onehotencoder.fit_transform(X).toarray() # convert to one-hot encoding
```

- remove redundancy features

```
In [13]: X = X[:, 1:]
```

- splitting dateset

```
In [14]: from sklearn.cross_validation import train_test_split
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state
```

0.2 Step 2: Fitting regression model

```
In [15]: from sklearn.linear_model import LinearRegression
         model = LinearRegression()
         model.fit(X_train, Y_train)
```

```
Out[15]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

0.3 Step 3: Predict

```
In [16]: y_pred = model.predict(X_test)
         print(y_pred)
```

```
[ 103015.20159796  132582.27760815  132447.73845175   71976.09851258
  178537.48221056  116161.24230166   67851.69209676   98791.73374687
  113969.43533013  167921.06569551]
```