Chapter 3 - ML Multivariate Linear Regression

March 30, 2020

1 Multvariate Linear Regression

1.1 import the dataset

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```python
dataset = pd.read_csv('ds/50_Startups.csv')
dataset.head()
```

[2]:

	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

1.1.1 dataset size

[3]:
```python
dataset.shape
```

[3]:
```
(50, 5)
```

1.1.2 about the dataset

- The data set comprises details of 50 startups, each with their location, expendature in R&D, Admin, marketting and net profit. the job is to advice a venture capitalist fund to which type of startups are more feasible to invest.
- create a ML model f from the dataset.
- $f : \{RnD, admin, mark, state\} \rightarrow profit$

1.2 Multivariate Linear Regression

1.2.1 Basics

- Genereal form : $y = b_0 + b_1x_1 + ... + b_nx_n = \sum_0^n b_ix_i$
- A straight line in n dimentinal space
- b_i : slope at i^{th} dimention , b_0 : y intercept

1.2.2 Assumptions

make sure the following assumptions hold for the model you're building, before start buildng 1. Linearity 2. Homoscedasticity 3. Multivariate normality 4. Independence of Error 5. Lac of Multicollinearity

1.2.3 Dummy Variables

- while building the model. **State** attribute in our case, happens to be be non-numeric.
- thus the linear regressor can't learn from it.
- therefore we need to encode them into some numeric expression
- now, a simple numeric map may create confusion to the regressor as higher numbers would be dominent and impose bias to the model, since there are no ordinal correaltion exists.
- therefore each distinct values from the target colum is presented as individual column as each occurence of the subjected item is presented as 1, otherwise 0.
- this new columns are called **Dummy Attributes** (d_i)
- this encodes into numeric attributes without impossing ordinality
- new Expression is $y = \sum_0^n b_ix_i + \sum_0^{k-1} b_jd_j$
- Although you souldn't include all your dummy varibale in training (**Dummy Variable Trap**)

1.2.4 Dummy variable Trap

- no two coherent dummy variables (sourced from a single attribute) for a given sample can't be 1 at the same time. $D_1 = 1 - D_2$
- this feature is called multi-collinearity
- including all dummy variable may confuse the model with the bias, called **Dummy variable Trap**
- *Rule of Thumb : for every set of dummy variables exclude 1 variable for training*

1.2.5 Optimise attribute set

- Dont include all the dependent variables
- select the right variables
- too many variable may incure noise (**Curse of dimentionality**)
- slows down the traing process
- hards to explain the dataset
- methods of building model

1. **All-in : include all variables** > * Domain / Priore knowledge (you're sure about it) > * Requirements (Company, client etc.) > * Preparing for backward elemination

2. **Backward Elimination** > * step 1 : select the **significance level (SL)** to stay in the model > * step 2 : **fit the full** model will all possible predictors > * step 3 : Consider the predictor with **Highest P-value**. ; if P > SL goto Step 4, else **Fin** (Model is ready). > * step 4 : **Remove** the predictor with highest P-value > * step 5 : **Rebuild** the model without the eleminated predictor ; goto step 3

3. **Forwards Selection** > * Step 1 : select the **SL to Enter the model** (5%) > * step 2 : fit all sample regression model $y = f(X_n)$; select one with **lowest P-value** > * step 3 : build all possible linear regression with **adding a new variable and keeping** the selected variable > * step 4 : among all the newly added variables, choose the one that gives model with **lowest p value** ; if P < SL goto step 3 otherwise **Fin**

4. **Bidirectional Elimination** > * step 1 : select $SL_{stey} = 0.05$ and $SL_{enter} = 0.05$ > * step 2 : perform the next step of Fotward selection, for new variable $ (P < SL_{enter}) $ > * step 3 : perform all steps from backward elemination, for old variables $ (P < SL_{stey}) $; goto step 2 > * step 4 : no new variables can enter, no old variables can exit : **Fin**

5. **Score Comparison (Most resource consuming model)** > * Step 1 : select a goodness criteria of fit (Akaike Criterion) > * Step 2 : Construct all possible regression models 2^{N-1} total combination > * step 3 : Select the best model ; **Fin**
6. Step wise regression - 2,3,**4** (More general)

1.3 Implementing Multivariate Linear Regression

1.3.1 Data Preprocessing

[4]:
```python
# separating dependent and independent variables
X = dataset.iloc[:,:-1].values
Y = dataset.iloc[:,-1].values

# Encode categorical variable (State)
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
le_X = LabelEncoder()
X[:,3] = le_X.fit_transform(X[:,3])
ohe = OneHotEncoder(categorical_features = [3])
X = ohe.fit_transform(X).toarray()
```

1.3.2 after one hot encoding

[5]:
```
pd.DataFrame(X).head()
```

[5]:

	0	1	2	3	4	5
0	0.0	0.0	1.0	165349.20	136897.80	471784.10
1	1.0	0.0	0.0	162597.70	151377.59	443898.53
2	0.0	1.0	0.0	153441.51	101145.55	407934.54
3	0.0	0.0	1.0	144372.41	118671.85	383199.62
4	0.0	1.0	0.0	142107.34	91391.77	366168.42

1.3.3 Avoiding Dummy variable Trap

[6]:
```
X = X [:, 1:] # remove the first column
pd.DataFrame(X).head()
```

[6]:

	0	1	2	3	4
0	0.0	1.0	165349.20	136897.80	471784.10
1	0.0	0.0	162597.70	151377.59	443898.53
2	1.0	0.0	153441.51	101145.55	407934.54
3	0.0	1.0	144372.41	118671.85	383199.62
4	1.0	0.0	142107.34	91391.77	366168.42

1.3.4 Train Test Split

[11]:
```
# train test split
from sklearn.cross_validation import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=.2,␣
↪random_state=40)
```

1.3.5 Run Multple LR

[8]:
```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression() # create regressor
regressor.fit(X_train, Y_train) # build model


Y_pred = regressor.predict(X_test) # make prediction
```

[10]:
```
train_plot = plt.subplot(211)
test_plot = plt.subplot(212)
```

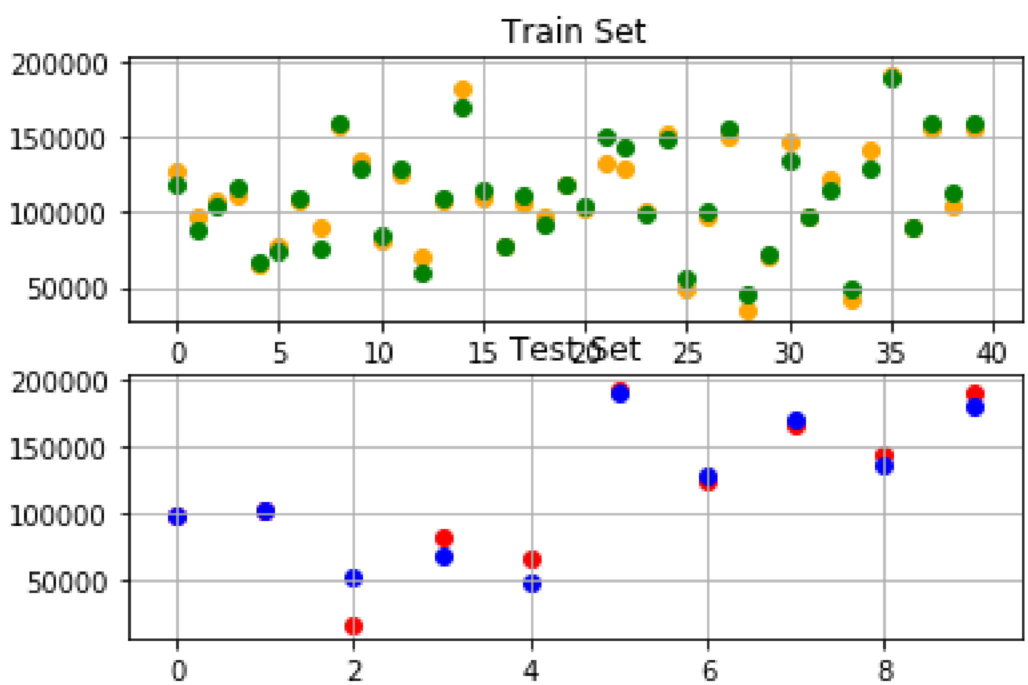
```
train_plot.grid(True)
train_plot.set_title('Train Set')
train_plot.scatter(np.arange(X_train.shape[0]),Y_train,
                   color='Orange',
                   label = 'actual train set')

train_plot.scatter(np.arange(X_train.shape[0]),regressor.predict(X_train),
                   color='green',
                   label = 'predicted train set')
train_plot.legend=True

test_plot.grid(True)
test_plot.set_title('Test Set')
test_plot.scatter(np.arange(X_test.shape[0]),Y_test,
                  color='red',
                  label = 'actual test set')

test_plot.scatter(np.arange(X_test.shape[0]),Y_pred,
                  color='blue',
                  label = 'Predicted test set')

test_plot.legend=True
plt.show()
```



```
[ ]:
```