Multiple Linear Regression

Problem Statment:

So remember a venture capital is fund hired you as a data scientist to train a machine learning model and actually a multiple regression model to understand the correlations between these features which are the spend in R&D administration and marketing as well as the state and the profit of what of 50 startups.

So in this dataset it's very important to understand that each row corresponds to a certain startup and for each startup.

Data scientists collected the following data:

- **R&D spend**
- **Administration spend**
- **Marketing spend**
- **State of these trips**
- **Profit**

because the goal for this V.C. fund is to figure out in which startup to invest based on these informations.

Example which investment brings more profit eg. is R&D in New York better ? etc

So these are all information that we already know from 50 startups.

And therefore if you managed to train a machinery model that can understand well these correlations well for the next door app you'll be able to deploy this model on these features to predict what sort of profit this new star might generate.

Importing the libraries

In [0]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [0]:

```
dataset= pd.read_csv('50_Startups.csv')
X = dataset.iloc[:,:-1].values #x1, x2 etc
y = dataset.iloc[:,-1].values  #Y dependednt variable
```

In [0]:

```
print(X)
```

Encoding categorical data

In [0]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='pass
through')
X = np.array(ct.fit_transform(X))
```

Splitting the dataset into the Training set and Test set

In [0]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0)
```

In [0]:

```
print(X)
```

Training the Multiple Linear Regression model on the Training set

In [0]:

```
print(y_train)
```

In [43]:

```
#the feature selection part(features with highest P vals) and the dummy variable part is
done automatically by the library modules
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[43]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Predicting the Test set results

We dont directly plot cuz we dont want a 4D graph

In [44]:

```
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2) #display numerical value with only 2 decimals
#concatenate 2 vertical vectors
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1))
#y_pred is predicted profit  || y_test = real profit
#we want to reshape y_pred into an aray having len(y_pred) rows i.e number of rows = numb
er of startups and 1=one column
#1 means horitzonal concatenation
```

```
[[103015.2  103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1   77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

Making a single prediction

(for example the profit of a startup with R&D Spend = 160000, Administration Spend = 130000, Marketing Spend = 300000 and State = 'California')

In [45]:

```
print(regressor.predict([[1, 0, 0, 160000, 130000, 300000]]))
```

```
[181566.92]
```

Therefore, our model predicts that the profit of a Californian startup which spent 160000 in R&D, 130000 in Administration and 300000 in Marketing is $ 181566,92.

Important note 1: Notice that the values of the features were all input in a double pair of square brackets. That's because the "predict" method always expects a 2D array as the format of its inputs. And putting our values into a double pair of square brackets makes the input exactly a 2D array. Simply put:

$$1, 0, 0, 160000, 130000, 300000 \rightarrow \text{scalars}$$

$$[1, 0, 0, 160000, 130000, 300000] \rightarrow \text{1D array}$$

$$[[1, 0, 0, 160000, 130000, 300000]] \rightarrow \text{2D array}$$

Important note 2: Notice also that the "California" state was not input as a string in the last column but as "1, 0, 0" in the first three columns. That's because of course the predict method expects the one-hot-encoded values of the state, and as we see in the second row of the matrix of features X, "California" was encoded as "1, 0, 0". And be careful to include these values in the first three columns, not the last three ones, because the dummy variables are always created in the first columns.

Getting the final linear regression equation with the values of the coefficients

In [46]:

```
print(regressor.coef_)
print(regressor.intercept_)
```

```
[ 8.66e+01 -8.73e+02  7.86e+02  7.73e-01  3.29e-02  3.66e-02]
42467.52924853204
```

Therefore, the equation of our multiple linear regression model is:

$$\begin{aligned} &\text{Profit} \\ &= 86.6 \times \text{Dummy State 1} - 873 \times \text{Dummy State 2} + 786 \times \text{Dummy State 3} - 0.773 \times \text{R\&D Spend} + 0.0329 \times A \end{aligned}$$

Important Note: To get these coefficients we called the "coef_" and "intercept_" attributes from our regressor object. Attributes in Python are different than methods and usually return a simple value or an array of values.