The Engineering World #DataScience 30 & 31

May 31, 2018

AKKAL BAHADUR BIST
DATA SCIENTIST AT
KATHMANDU INSTITUTE OF APPLIED SCIENCES (KIAS)
Center for Conservation Biology (CCB)

1 LINEAR REGRESSION FOR MACHINE LEARNING

1.0.1 Linear Regression

```
In [1]: import numpy as np
        import pandas as pd
        from pylab import rcParams
        import seaborn as sb
        import matplotlib.pyplot as plt

        import sklearn
        from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import scale
        from collections import Counter
```

```
In [2]: %matplotlib inline
        rcParams ['figure.figsize'] = 5,4
        sb.set_style ('whitegrid')
```

1.0.2 (Multiple)Linear Regression on the enrollment data

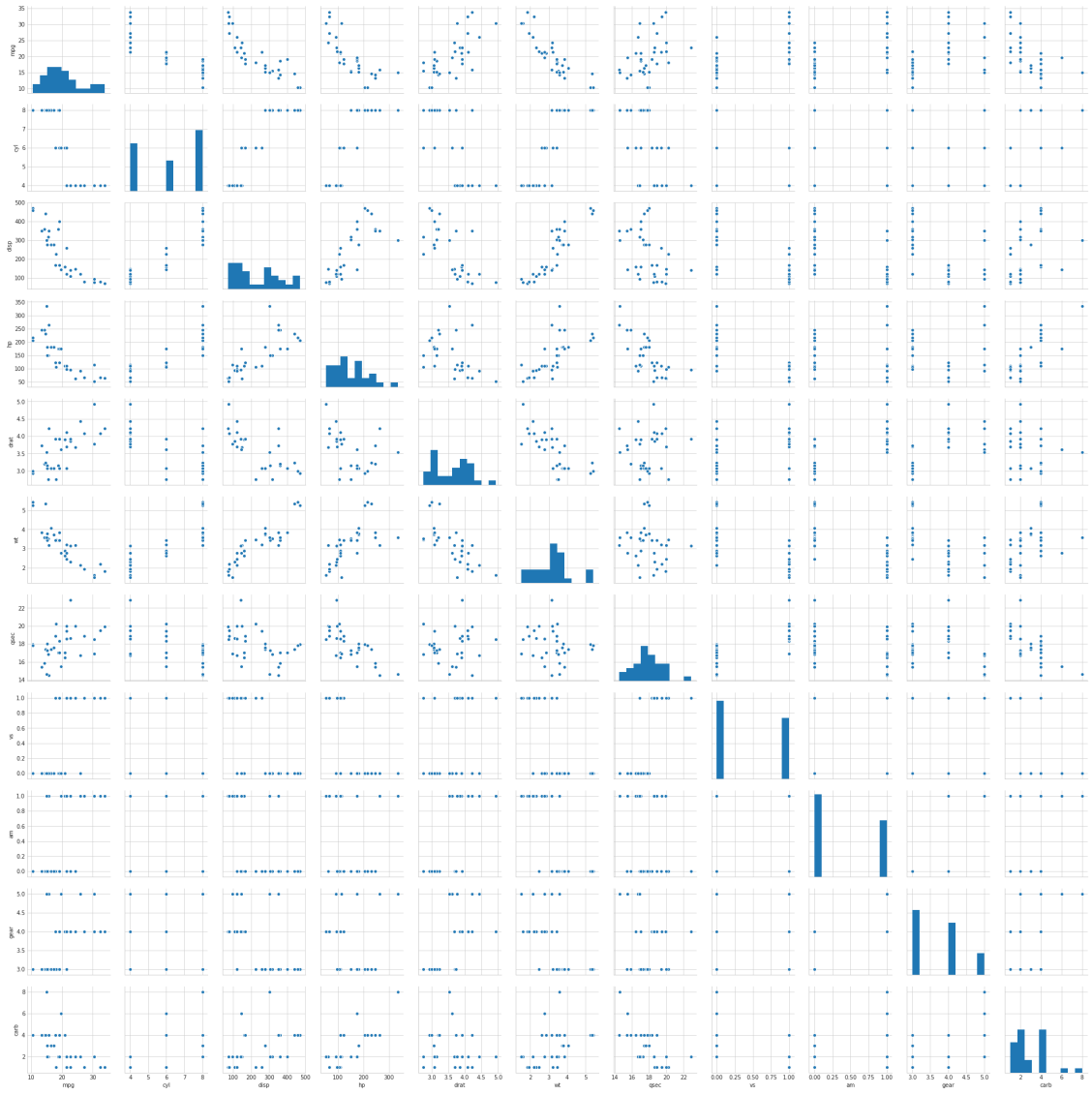
```
In [3]: address = 'mtcars.csv'
        cars = pd.read_csv(address)
        cars.columns = ['car_names', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am
        cars.head()
```

```
Out[3]:            car_names   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  \
        0          Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4
        1      Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4
        2         Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4
        3     Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3
        4  Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0     3
```

```
   carb
0     4
1     4
2     1
3     1
4     2
```

In [4]: `sb.pairplot(cars)`

Out[4]: `<seaborn.axisgrid.PairGrid at 0x7f1f98314c50>`



In [5]: `print(cars.corr())`

```
           mpg       cyl      disp        hp      drat        wt      qsec  \
mpg   1.000000 -0.852162 -0.847551 -0.776168  0.681172 -0.867659  0.418684
cyl  -0.852162  1.000000  0.902033  0.832447 -0.699938  0.782496 -0.591242
disp -0.847551  0.902033  1.000000  0.790949 -0.710214  0.887980 -0.433698
hp   -0.776168  0.832447  0.790949  1.000000 -0.448759  0.658748 -0.708223
drat  0.681172 -0.699938 -0.710214 -0.448759  1.000000 -0.712441  0.091205
wt   -0.867659  0.782496  0.887980  0.658748 -0.712441  1.000000 -0.174716
qsec  0.418684 -0.591242 -0.433698 -0.708223  0.091205 -0.174716  1.000000
vs    0.664039 -0.810812 -0.710416 -0.723097  0.440278 -0.554916  0.744535
am    0.599832 -0.522607 -0.591227 -0.243204  0.712711 -0.692495 -0.229861
gear  0.480285 -0.492687 -0.555569 -0.125704  0.699610 -0.583287 -0.212682
carb -0.550925  0.526988  0.394977  0.749812 -0.090790  0.427606 -0.656249

            vs        am      gear      carb
mpg   0.664039  0.599832  0.480285 -0.550925
cyl  -0.810812 -0.522607 -0.492687  0.526988
disp -0.710416 -0.591227 -0.555569  0.394977
hp   -0.723097 -0.243204 -0.125704  0.749812
drat  0.440278  0.712711  0.699610 -0.090790
wt   -0.554916 -0.692495 -0.583287  0.427606
qsec  0.744535 -0.229861 -0.212682 -0.656249
vs    1.000000  0.168345  0.206023 -0.569607
am    0.168345  1.000000  0.794059  0.057534
gear  0.206023  0.794059  1.000000  0.274073
carb -0.569607  0.057534  0.274073  1.000000
```

```
In [6]: cars_data = cars.ix[:,(2,3)].values
        cars_target = cars.ix[:,1].values
        cars_data_names = ['hp', 'am']
        X,Y = scale(cars_data), cars_target
```

```
/home/akkal/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
```

1.0.3 Checking for missing values

```
In [7]: missing_values = X == np.NAN
        X[missing_values == True]
```

```
Out[7]: array([], dtype=float64)
```

```
In [8]: LinReg = LinearRegression(normalize=True)
        LinReg.fit(X,Y)
        print(LinReg.score(X,Y))
```

```
0.7595657755568964
```

2 LOGESTIC REGRESSION FOR MACHINE LEARNING

2.0.1 Logestic Regression

```
In [9]: import numpy as np
        import pandas as pd
        from pandas import Series, DataFrame
        from pylab import rcParams
        import scipy
        from scipy.stats import spearmanr
        import seaborn as sb
        import matplotlib.pyplot as plt

        import sklearn
        from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import train_test_split
        from sklearn import metrics
        from sklearn.preprocessing import scale
        from sklearn import preprocessing
```

```
In [10]: %matplotlib inline
         rcParams ['figure.figsize'] = 5,4
         sb.set_style ('whitegrid')
```

2.0.2 Logestic regression on mtcars

```
In [11]: address = 'mtcars.csv'
         cars = pd.read_csv(address)
         cars.columns = ['car_names', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'a
         cars.head()
```

```
Out[11]:            car_names   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  \
         0          Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1     4
         1      Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1     4
         2         Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1     4
         3     Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0     3
         4  Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0     3

            carb
         0     4
         1     4
```

```
2     1
3     1
4     2
```

```
In [12]: cars_data = cars.ix[:,(5,11)].values
         cars_data_names = ['drat', 'carb']

         y = cars.ix[:,9].values
```

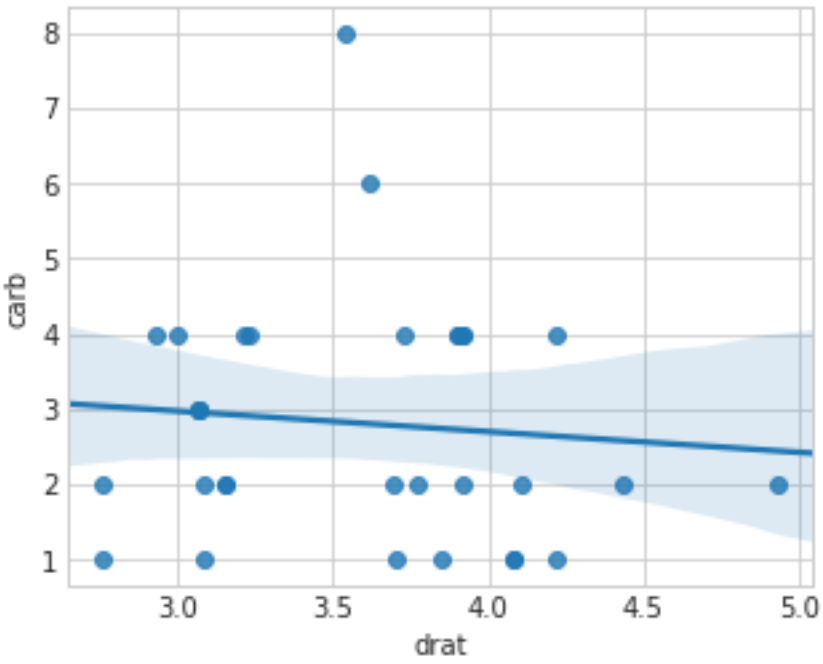
```
/home/akkal/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
```

2.0.3 Checking for independence between features

```
In [13]: sb.regplot(x = 'drat', y = 'carb', data = cars, scatter=True)
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f5a0eea20>
```



```
In [14]: drat = cars['drat']
         carb = cars['carb']
         spearmanr_coefficient, p_value = spearmanr(drat, carb)
         print('spearmanr Rand Correlation Coefficient %0.3f' % (spearmanr_coefficient))
```

```
spearmanr Rand Correlation Coefficient -0.125
```

2.0.4 Checking for missing values

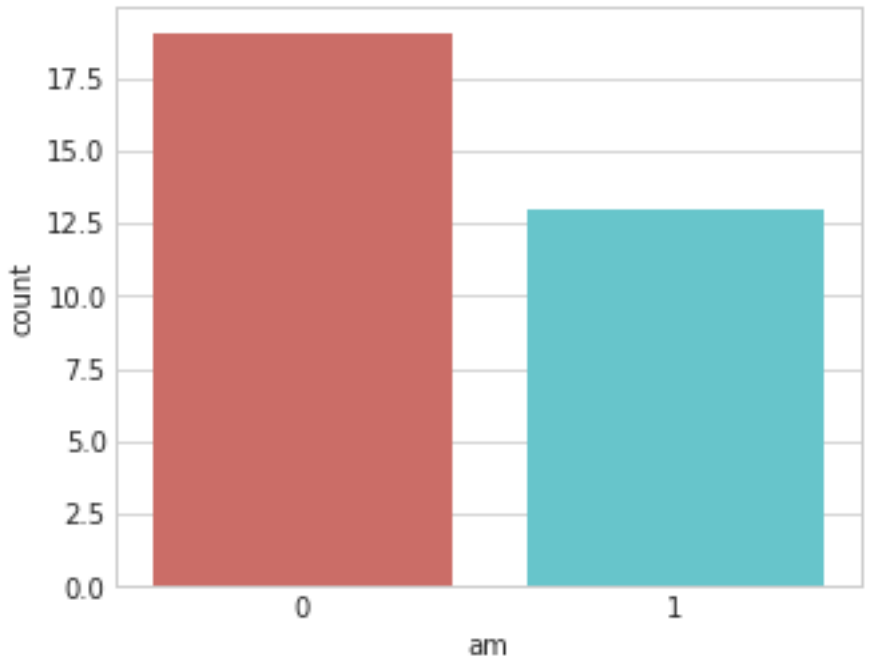
```
In [15]: cars.isnull().sum()
```

```
Out[15]: car_names    0
         mpg          0
         cyl          0
         disp         0
         hp           0
         drat         0
         wt           0
         qsec         0
         vs           0
         am           0
         gear         0
         carb         0
         dtype: int64
```

2.0.5 Checking that your binary or ordinal

```
In [16]: sb.countplot(x = 'am', data = cars, palette='hls')
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f59bf3828>
```



2.0.6 Checking that yur data size is sufficient

```
In [17]: cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
car_names    32 non-null object
mpg          32 non-null float64
cyl          32 non-null int64
disp         32 non-null float64
hp           32 non-null int64
drat         32 non-null float64
wt           32 non-null float64
qsec         32 non-null float64
vs           32 non-null int64
am           32 non-null int64
gear         32 non-null int64
carb         32 non-null int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```

2.0.7 Deploying and evaluating yur model

```
In [18]: X = scale(cars_data)
```

```
In [19]: LogReg = LogisticRegression()
         LogReg.fit(X,y)
         print(LogReg.score(X,y))
```

```
0.8125
```

```
In [20]: y_pred = LogReg.predict(X)
         from sklearn.metrics import classification_report
         print (classification_report(y,y_pred))
```

```
             precision    recall  f1-score   support

          0       0.88      0.79      0.83        19
          1       0.73      0.85      0.79        13

avg / total       0.82      0.81      0.81        32
```