DASC 500: Introduction to Data Analytics

Assignment #4

21 Jun 2020

Each of the questions for this assignment involve the `mtcars` data set, which has been provided as part of this assignment in Canvas. For many, `mtcars` is known as a "hello world" data set for students learning linear regression.

The data were extracted from the 1974 Motor Trend US magazine, and contains one response variable `mpg` (fuel consumption) and 10 predictor variables that represent different aspects of automobile design and performance for 32 automobiles.

Problem 1

Partition the `mtcars` data into a training set and a test set. Use a 70/30 split wherein 70% of the data is contained in the training set and 30% is contained in the test set.

As shown in the multiple linear regression lecture slides you can generate this sample using the `sample()` and `drop()` functions that are included in a `pandas.DataFrame` object. When using this method you're able to include the argument `random_state` in the call to `sample()`. By specifying a value for `random_state` you ensure that your random sample is repeatable and if everyone chooses the same value for `random_state` it becomes easier to compare results. For this assignment you should set `random_state = 42`. Why 42?

First, let's load the libraries we'll need to complete this assignment

```
import random
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import scipy as sp
import statsmodels.api as sm
import statistics as st
import lxml
```

Next, let's read in the data from the CSV file using `read_csv()` from the `pandas` library

```
df = pd.read_csv("../regression/mtcars.csv")
```

With the data read in as an object named `df`, we can now partition the data set into a training set a test set. As stated above we use the argument `random_state = 42` to ensure that the data in my training data is the same as yours. Rather than printing out all of the observations in `train_data`, I use `head()` and `tail()` to show the first and last 6 rows of the data.

```
train_data = df.sample(frac = 0.7, random_state = 42)
test_data = df.drop(index = train_data.index)

train_data.head()
```

```
     mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  carb
29  19.7    6  145.0  175  3.62  2.770  15.50   0   1     5     6
15  10.4    8  460.0  215  3.00  5.424  17.82   0   0     3     4
24  19.2    8  400.0  175  3.08  3.845  17.05   0   0     3     2
17  32.4    4   78.7   66  4.08  2.200  19.47   1   1     4     1
8   22.8    4  140.8   95  3.92  3.150  22.90   1   0     4     2
```

```python
train_data.tail()
```

```
     mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  carb
2   22.8    4  108.0   93  3.85  2.320  18.61   1   1     4     1
26  26.0    4  120.3   91  4.43  2.140  16.70   0   1     5     2
3   21.4    6  258.0  110  3.08  3.215  19.44   1   0     3     1
21  15.5    8  318.0  150  2.76  3.520  16.87   0   0     3     2
27  30.4    4   95.1  113  3.77  1.513  16.90   1   1     5     2
```

Problem 2

Suppose you were asked to build a linear regression model but were only allowed to use a single predictor. Build 10 different simple linear models where `mpg` is the response variable and the predictor variable is one of the other 10 variables.

Your answer to this problem should include a table containing the following columns:

- **Predictor: The name of the predictor**
- **beta_1: The value of the β_1 parameter for each model**
- **t-stat: The value of the t_statistic**
- **$\underline{\beta_1}$: The lower limit of the confidence interval for β_1**
- **$\overline{\beta_1}$: The upper limit of the confidence interval for β_1**

For this question we're asked to evaluate 10 different simple linear regression models, build a table containing the specified results from each model and then choose a model, from the 10, that you believe to be the best at describing the uncertainty in the `mpg` responses.

First, let's create a `DataFrame` to contain the results from each of the models. Note that I use `zeros()` from `numpy` to create an object with 6 columns and 11 rows. The reason my `DataFrame` has 11 rows is because I will also include the model in which `mpg` is regressed against itself.

```python
result_df = pd.DataFrame(data = np.zeros([len(df.columns),6]),
                         index = df.columns,
                         columns = ["coef",  "std err", "t", "P>|t|", "[0.025", "0.975]"])
```

With the `DataFrame` create the loop defined below can be used to fill each row with the results obtained from each model.

```python
for i in range(len(df.columns)):

  y  = train_data["mpg"]
  X1 = train_data[{result_df.index[i]}]
  X1 = sm.add_constant(X1)
  model = sm.OLS(y, X1)
  model_fit = model.fit()

  # create HTML table from model_fit.summary()
  # note that this requires that the lxml library is installed
  out = pd.read_html(model_fit.summary().tables[1].as_html(),header=0,index_col=0)[0]
  # Now store the output from the object out
  # as the ith row of the object result_df
  result_df[i:] = out[df.columns[i]:]
```

	coef	std err	t	P>|t|	[0.025	0.975]
mpg	1.0000	0.000	7.030e+15	0.000	1.000	1.000
cyl	-2.8024	0.352	-7.966e+00	0.000	-3.536	-2.069
disp	-0.0383	0.006	-6.618e+00	0.000	-0.050	-0.026
hp	-0.0629	0.012	-5.167e+00	0.000	-0.088	-0.037
drat	7.4349	1.909	3.894e+00	0.001	3.452	11.417
wt	-4.8949	0.616	-7.943e+00	0.000	-6.180	-3.609
qsec	0.9745	0.649	1.502e+00	0.149	-0.379	2.328
vs	6.9143	1.951	3.544e+00	0.002	2.844	10.984
am	6.8632	1.891	3.629e+00	0.002	2.918	10.808
gear	3.9302	1.303	3.016e+00	0.007	1.212	6.648
carb	-1.5664	0.597	-2.623e+00	0.016	-2.812	-0.321

Problem 3

Consider the results you obtained in Problem #2. If you're only allowed to use a single predictor - which one would you choose? There is no one right answer to this question, I'm just looking for your reasoning behind choosing one model over the others.

Based on the results shown in the table in the previous question I would choose the regression model in which the predictor was `wt`. I chose this predictor as the model results indicate that the weight of the car has the greatest combination of a large β_1 coefficient (albeit negative) and a small t-statistic

Problem 4

Consider the model you chose in Problem # 3. How would you explain what this model is saying?

The model is saying that every unit increase in a car's weight (pounds, kilograms, tons, etc.) results in a -4.8949 change in the car's fuel efficiency (measured in miles per gallon).

Problem 5

Now build and fit a single model that includes all 10 predictors. Show a summary of results from this model.

```
# select mpg as the output (response)
y  = train_data["mpg"]

# select all the other columns as inputs (predictors)
# Note that the column names must be converted to an array
X1 = train_data[df.columns[1:].array]
X1 = sm.add_constant(X1)
model = sm.OLS(y, X1)
model_fit = model.fit()

out = pd.read_html(model_fit.summary().tables[1].as_html(),header=0,index_col=0)[0]
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-47.1135	40.319	-1.169	0.267	-135.855	41.628
cyl	3.7036	2.480	1.493	0.164	-1.756	9.163
disp	0.0059	0.020	0.299	0.770	-0.038	0.049
hp	-0.0212	0.026	-0.806	0.437	-0.079	0.037
drat	4.4426	2.571	1.728	0.112	-1.215	10.101
wt	-1.1331	2.302	-0.492	0.632	-6.199	3.933
qsec	0.9279	0.806	1.151	0.274	-0.847	2.703
vs	3.4352	2.523	1.362	0.201	-2.118	8.988
am	4.6487	2.841	1.636	0.130	-1.604	10.902
gear	5.6231	3.097	1.816	0.097	-1.193	12.439
carb	-2.3622	1.259	-1.877	0.087	-5.132	0.408

Problem 6

Compare the results you obtained in Problem #5 to those you obtained in Problem #1. Do these results of the larger model align with those seen in the 10 individual models? If not, comment on why you think this is the case.

No the results are not the same. This is mainly due the `OLS` algorithm attempting to fit model to each predictor in the presence of the other predictors. Note that in the results table obtained from the multiple linear regression model the coefficient for `cyl` has changed signs.