303-2 Assignment 1 solutions

January 10, 2022

1 STAT303-2: Assignment 1 solutions

47 points possible (with bonus questions answered correctly) - 42 points possible for code & answer
5 points for anonymity and proper formatting

1.1 Part 1

(20+5 points possible if bonus questions answered correctly)
Read the dataset petrol_consumption_train.csv. It contains the following five columns:
Petrol_tax: Petrol tax (cents per gallon)
Per_capita_income: Average income (dollars)
Paved_highways: Paved Highways (miles)
Prop_license: Proportion of population with driver's licenses
Petrol_consumption: Consumption of petrol (millions of gallons)

1.1.1 a.

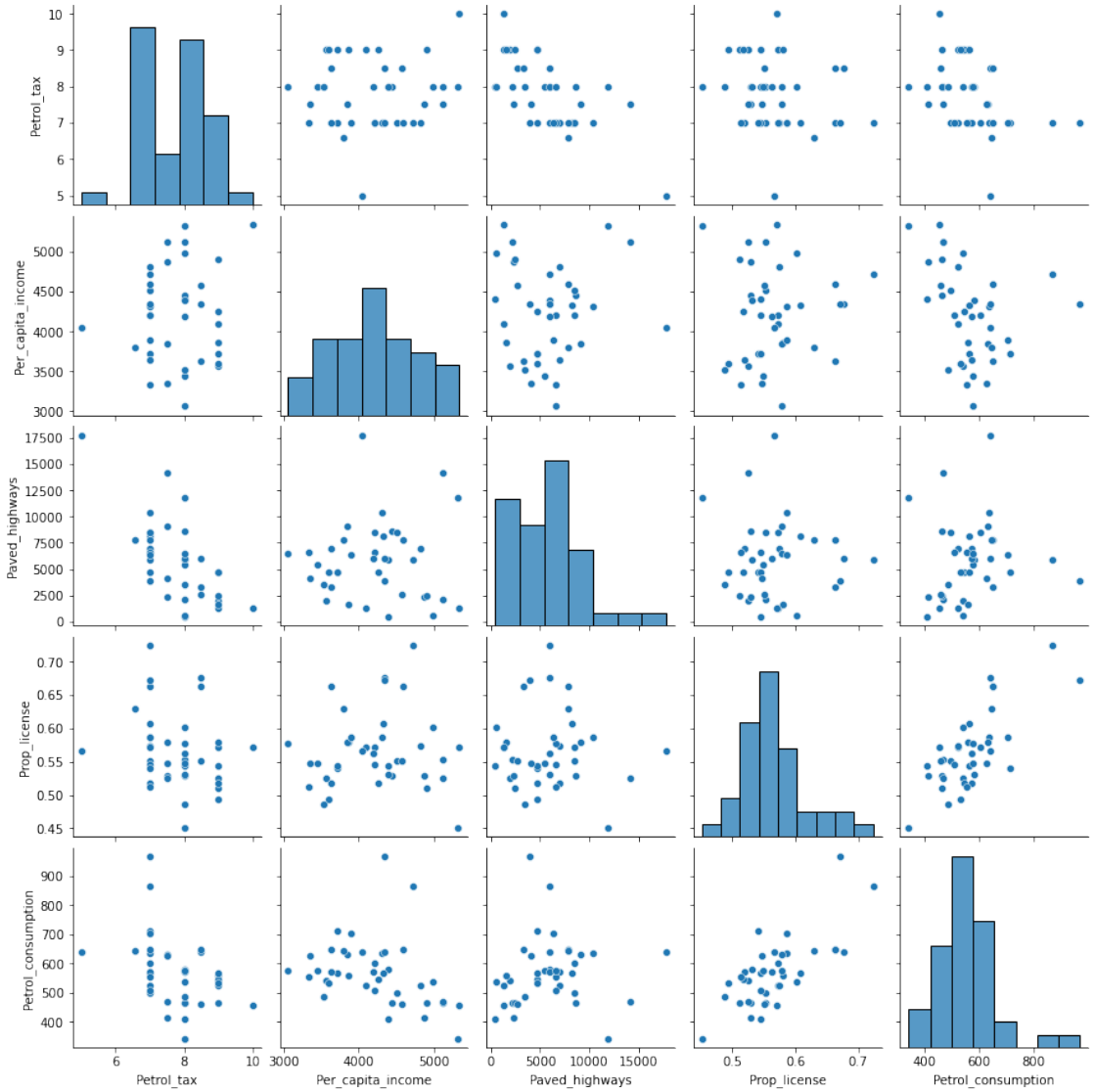
Make a pairwise plot of all the variables in the dataset. Which variable seems to have the highest linear correlation with petrol_consumption? Let this variable be predictor *P*. *Note: If you cannot figure out P by looking at the visualization, you may find the pairwise correlation coefficient to identify P.* (3 points for visualization, 1 point for answer)

[1]:
```
import pandas as pd
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
```

[2]:
```
train = pd.read_csv('petrol_consumption_train.csv')
```

[3]:
```
sns.pairplot(train)
```

[3]:
```
<seaborn.axisgrid.PairGrid at 0x7fd57a623880>
```



[4]:
```
train.corr()
```

[4]:
```
                    Petrol_tax  Per_capita_income  Paved_highways  \
Petrol_tax            1.000000           0.082359       -0.660022
Per_capita_income     0.082359           1.000000        0.040256
Paved_highways       -0.660022           0.040256        1.000000
Prop_license         -0.223920           0.048153       -0.037998
Petrol_consumption   -0.393415          -0.314039        0.098117

                    Prop_license  Petrol_consumption
Petrol_tax              -0.223920           -0.393415
Per_capita_income        0.048153           -0.314039
Paved_highways          -0.037998            0.098117
```

```
Prop_license            1.000000            0.718303
Petrol_consumption      0.718303            1.000000
```

Prop_license has the highest linear correlation with *Petrol_consumption*

1.1.2 b.

Fit a simple linear regression model to predict petrol_consumption based on predictor P (identified in the previous part). Print the model summary. (2 points for code)

[5]:
```
model = smf.ols(formula='Petrol_consumption~Prop_license',data = train).fit()
model.summary()
```

[5]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:     Petrol_consumption   R-squared:                       0.516
Model:                            OLS   Adj. R-squared:                  0.503
Method:                 Least Squares   F-statistic:                     40.51
Date:                Mon, 10 Jan 2022   Prob (F-statistic):           1.80e-07
Time:                        17:47:44   Log-Likelihood:                -231.59
No. Observations:                  40   AIC:                             467.2
Df Residuals:                      38   BIC:                             470.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -267.6155    132.038     -2.027      0.050    -534.912      -0.319
Prop_license  1479.1803    232.414      6.364      0.000    1008.682    1949.678
==============================================================================
Omnibus:                        5.963   Durbin-Watson:                   1.280
Prob(Omnibus):                  0.051   Jarque-Bera (JB):                4.620
Skew:                           0.762   Prob(JB):                       0.0993
Kurtosis:                       3.670   Cond. No.                         23.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

1.1.3 c.

What is the increase in petrol consumption for an increase of 0.01 in P? (1 point for answer)

[6]:
```
1479.1803*0.01
```

[6]: 14.791803

For an increase in 0.01 in proportion of people with driving license, the consumption of petrol increase by 14.79 million gallons.

1.1.4 d.

Does petrol consumption have a statistically significant relationship with the predictor P? (1 point for answer)

Yes, as the p-value of the coeffecient of *Prop_license* is very small («0.05), petrol, we conclude that *Prop_license* has a statistically significant relationship with *Petrol_consumption*.

1.1.5 e.

What is the R-squared? Interpret its value. (1 point for answer, 1 point for interpretation)

The R-squared = 51.6%. This means that 51.6% of variation in the petrol consumption can be explained using the linear relationship with the proportion of people having driving license.

1.1.6 f.

Estimate the petrol consumption for a state in which 50% of the population has a driver's license. What are the confidence and prediction intervals for your estimate? (2 points for code, 1.5 points for answer)

[7]:
```
intervals = model.get_prediction(pd.DataFrame({'Prop_license':[0.5]}))
intervals.summary_frame(alpha=0.05)
```

[7]:

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	471.974627	19.896237	431.6968	512.252454	302.822725	641.126528

The estimate of the petrol consumption is 472 million gallons. The confidence interval of the estimate is [432,512]. The prediction interval of the estimate is [303,641].

1.1.7 g.

Estimate the petrol consumption for a state in which 10% of the population has a driver's license. Are you getting a reasonable estimate? Why or why not? (2 points for code, 1.5 points for answer)

[8]:
```
model.predict(pd.DataFrame({'Prop_license':[0.10]}))
```

[8]:
```
0   -119.697506
dtype: float64
```

[9]:
```
print(train.Prop_license.min(), train.Prop_license.max())
```

```
0.451 0.724
```

The estimate obtained is -119.7 million gallons. This is not a reasonable estimate as petrol consumption cannot be negative. The model has been trained for *Prop_license* in [0.45, 0.72]. *Prop_license = 0.1* is far away from the domain space on which the model has been trained. The model cannot be trusted to predict accurately at a point far away from its training domain space.

1.1.8 h.

Estimate the petrol consumption for the observations in *petrol_consumption_train.csv*. Find the RMSE (Root mean squared error). (2 points for code, 1 point for answer)

[10]:
```
test = pd.read_csv('petrol_consumption_test.csv')
```

[11]:
```
pred = model.predict(test)
```

[12]:
```
np.sqrt(((pred-test.Petrol_consumption)**2).mean())
```

[12]:
```
80.13903941152402
```

The RMSE is around 80 million gallons.

Bonus point questions (5 points, please only give credit if all parts are answered correctly, no partial credit should be given)

1.1.9 i.

Fit a simple linear regression model to predict *petrol_consumption* based on predictor *P*, but without an intercept term.

[13]:
```
model = smf.ols(formula='Petrol_consumption~Prop_license-1',data = train).fit()
```

1.1.10 j.

Estimate the petrol consumption for the observations in petrol_consumption_test.csv using the model in developed in (i). Find the RMSE

[14]:
```
pred = model.predict(test)
np.sqrt(((pred-test.Petrol_consumption)**2).mean())
```

[14]:
```
76.398744492552
```

The RMSE is around 76 million gallons.

1.1.11 k.

The RMSE for the models in (i) and (b) are similar, which indicates that both models are almost equally good. However, the R-squared for the model in (j) is much higher than the R-squared for the model in (b). Why?

Both models are equally good. However, their R-squared values cannot be compared. This is because the presence or absence of intercept changes the formulation of R-squared.

For the model with intercept, the developed model is compared against an only-intercept model. For an only-intercept model, $\hat{\beta}_0 = \bar{y}$ and so the R-squared is given by:

$$R^2 = 1 - \frac{\Sigma_i (y_i - \hat{y}_i)^2}{\Sigma_i (y_i - \bar{y})^2} \tag{1}$$

s For the model with no intercept, the developed model is compared against the model with only noise, and the so R-squared is given by:

$$R_0^2 = 1 - \frac{\Sigma_i (y_i - \hat{y}_i)^2}{\Sigma_i y_i^2} \tag{2}$$

As $\bar{y}$ in our dataset is actually not equal to zero, the denominator in the latter equation is much larger than the former one, which leads to a much higher value of R_0^2 as compared to R^2.

1.2 Part 2

(17 points total)

A study was conducted on 97 men with prostate cancer who were due to receive a radical prostatectomy. The dataset prostate.csv contains data on 9 measurements made on these 97 men. The description of variables can be found here: https://rafalab.github.io/pages/649/prostate.html

1.2.1 a.

Fit a linear regression model with lpsa as the response and the other variables as the predictors. Write down the equation to predict *lpsa* based on the other eight variables. (2 points for code, 1 point for answer)

[15]:
```
data = pd.read_csv('prostate.csv')
```

[16]:
```
predictors=list(data.columns)
predictors.remove('lpsa')
predictors = "+".join(predictors)
```

[17]:
```
model = smf.ols(formula='lpsa~'+predictors, data=data).fit()
model.summary()
```

[17]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   lpsa   R-squared:                       0.655
Model:                            OLS   Adj. R-squared:                  0.623
Method:                 Least Squares   F-statistic:                     20.86
Date:                Mon, 10 Jan 2022   Prob (F-statistic):           2.24e-17
Time:                        17:47:44   Log-Likelihood:                -99.476
No. Observations:                  97   AIC:                             217.0
Df Residuals:                      88   BIC:                             240.1
```

```
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.6693      1.296      0.516      0.607      -1.907       3.246
lcavol         0.5870      0.088      6.677      0.000       0.412       0.762
lweight        0.4545      0.170      2.673      0.009       0.117       0.792
age           -0.0196      0.011     -1.758      0.082      -0.042       0.003
lbph           0.1071      0.058      1.832      0.070      -0.009       0.223
svi            0.7662      0.244      3.136      0.002       0.281       1.252
lcp           -0.1055      0.091     -1.159      0.250      -0.286       0.075
gleason        0.0451      0.157      0.287      0.775      -0.268       0.358
pgg45          0.0045      0.004      1.024      0.309      -0.004       0.013
==============================================================================
Omnibus:                        0.235   Durbin-Watson:                   1.507
Prob(Omnibus):                  0.889   Jarque-Bera (JB):                0.026
Skew:                          -0.017   Prob(JB):                        0.987
Kurtosis:                       3.073   Cond. No.                     1.28e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.28e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

The prediction equation is lpsa = 0.669+0.587 * lcavol + 0.454l * weight − 0.020 * age + 0.107l * bph + 0.766 * svi − 0.105 * lcp + 0.045 * gleason + 0.005 * pgg45

1.2.2 b.

Is the relationship between lpsa and the predictor variables significant at 0.05? (1 point for answer)

For the overall significance of the regression, we get F = 20.86 with 8 and 88 degrees of freedom. The p − value = 2.2 × 10−17 is extremely small indicating that the regression is highly significant.

1.2.3 c.

Report the p-value for age. What do you conclude about the significance of this variable? (1 point for answer, 1 point for conclusion)

The p-value of the age variable is 0.08229. This variable is not significant at 5% level, but is significant at 10% level. So, overall, it is a marginally significant variable.

1.2.4 d.

What is the 95% confidence interval for the coefficient of age? Can you conclude anything about its significance based on the confidence interval? (1 point for answer, 1 point for conclusion)

The 95% confidence interval for the coefficient of age is [-0.042 0.003]. We can see that the 95% confidence interval includes 0 and therefore, this variable is not significant at 5% level.

1.2.5 e.

Fit a simple linear regression on lpsa against age. What is the p-value for age? (2 points for code, 1 point for answer)

[18]:
```
model = smf.ols(formula='lpsa~age', data=data).fit()
model.summary()
```

[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   lpsa   R-squared:                       0.029
Model:                            OLS   Adj. R-squared:                  0.019
Method:                 Least Squares   F-statistic:                     2.813
Date:                Mon, 10 Jan 2022   Prob (F-statistic):             0.0968
Time:                        17:47:44   Log-Likelihood:                -149.64
No. Observations:                  97   AIC:                             303.3
Df Residuals:                      95   BIC:                             308.4
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.7991      1.008      0.793      0.430      -1.202       2.800
age            0.0263      0.016      1.677      0.097      -0.005       0.057
==============================================================================
Omnibus:                        2.538   Durbin-Watson:                   0.067
Prob(Omnibus):                  0.281   Jarque-Bera (JB):                2.096
Skew:                           0.152   Prob(JB):                        0.351
Kurtosis:                       3.653   Cond. No.                         558.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

The p-value for age is 0.097

1.2.6 f.

Explain why this p-value in (e) is different from the p-value in (b). (3 points for answer)

The p-value for the coefficient of age is 0.097. This is not the same as the p-value=0.082 obtained earlier. This is because the models under consideration are different among the two tests. For the simple linear regression we are testing:

$H_0 : E(y) = \beta_0$ against $H_1 : E(y) = \beta_0 + \beta_1 age$,

whereas in the multiple linear regression, we are testing: $H_0 : E(y) = \beta_0 + \beta_1 lcavol + \beta_2 lweight + \beta 4 lbph + \cdots$ against $H_1 : E(y) = \beta_0 + \beta_1 lcavol + \beta_2 lweight + \beta_3 age + \beta 4 lbph + \cdots$

1.2.7 g.

Predict lpsa of a 65-year old man with *lcavol = 1.35, lweight = 3.65, lbph = 0.1, svi = 0.22, lcp = -0.18, gleason = 6.75, and pgg45 = 25* and find 95% prediction intervals. (2 points for code, 1 points for answer)

[19]:
```
predictor_vals=pd.DataFrame({'lcavol':[1.35], 'lweight': 3.65, 'lbph': 0.
↪1,'svi': [0.22],'lcp': [-0.18],'gleason': [6.75],'pgg45': [25],'age':65})
model.predict(predictor_vals)
```

[19]:
```
0    2.508206
dtype: float64
```

[20]:
```
intervals = model.get_prediction(predictor_vals)
intervals.summary_frame(alpha=0.05)
```

[20]:

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	2.508206	0.117466	2.275006	2.741405	0.225969	4.790443

Predicted value for lpsa = 2.51, and prediction interval is [0.23, 4.79]