

## ✓ Exercise - Linear Regression

### DATA 3300

Name: Chance Wiese

#### ✓ Q1

**Begin by loading the required libraries and packages - including pandas , numpy , matplotlib.pyplot , statsmodels.api , sklearn.model\_selection , and sklearn.linear\_model . Then import the dataset as a pandas dataframe and remove any leading or trailing spaces from the columns.**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metric

df = pd.read_csv("/content/carsalesS19.csv")
df = df.rename(columns=lambda x: x.strip()) # let's remove any leading/trailing
# pull up a heading
df.head()
```

	SPID	Age	Gender	Ex	Su	Brand	HSGrad	CollegeGrad	Married	IQ	WritExam	
0	1	54	female	3.0	4.5	Honda	Y	N	Y	95	0.727	
1	2	35	male	7.0	5.5	Subaru	Y	Y	Y	99	0.809	
2	3	40	male	3.6	3.5	Honda	Y	N	Y	112	0.945	
3	4	34	female	6.2	1.5	Honda	Y	N	N	109	0.844	
4	5	40	male	6.4	1.5	Subaru	N	N	N	86	0.703	



Next steps:

 [View recommended plots](#)

Now let's create a new dataframe object that contains only the numerical variables so we can FIRST run a CORRELATION analysis. Let's call the new object `df_cor`.

Generate a CORRELATION MATRIX to evaluate the relationships amongst the IVs and the IVs with the DV. Be sure to only include those variables that should be included for Pearson correlation analysis (e.g., should categorical variables be included or excluded?).

```
df_cor = df.drop(["SPID", "Gender", "Brand", "HSGrad", "CollegeGrad", "Married"], a:
df_cor.head()
```

	Age	Ex	Su	IQ	WritExam	Months	
0	54	3.0	4.5	95	0.727	24	
1	35	7.0	5.5	99	0.809	16	
2	40	3.6	3.5	112	0.945	9	
3	34	6.2	1.5	109	0.844	11	
4	40	6.4	1.5	86	0.703	24	

Next steps: [View recommended plots](#)

```
# create correlation matrix
cor_matrix = df_cor.corr()
cor_matrix.style.background_gradient(cmap='seismic', axis=None, vmin=-1, vmax=1)
```

	Age	Ex	Su	IQ	WritExam	Months
Age	1.000000	0.087810	-0.019577	0.037683	0.083308	0.100242
Ex	0.087810	1.000000	0.017798	0.098405	0.063983	0.555805
Su	-0.019577	0.017798	1.000000	0.075746	0.081838	0.057962
IQ	0.037683	0.098405	0.075746	1.000000	0.823414	-0.300422
WritExam	0.083308	0.063983	0.081838	0.823414	1.000000	-0.246135
Months	0.100242	0.555805	0.057962	-0.300422	-0.246135	1.000000

✓ 1A

Because the dealership is interested in predicting Months (how long an employee will stay in a position), consider Months to be the DV.

**Based on the correlation matrix, given a minimum correlation coefficient ( $r$ ) value of 0.5 to determine collinearity, are there any collinear *independent variables*? If so, report the variable pair and their correlation coefficient.**

Written Exam and IQ are collinear because they have a  $r = 0.83$ , which passes our threshold

✓ 1B

**Based on the correlation matrix, which two independent variables appear to have the strongest relationship to the dependent variable? For each, report the variable, the correlation coefficient, and the  $R^2$  value.**

- EX - Months,  $r = 0.555805$  |  $r^2 = 0.30$
- IQ - Months,  $r = -0.300422$  |  $r^2 = 0.09$

✓ 1C

**Within this correlation context, what does the  $R^2$  value tell us?**

The  $R^2$  is variance explained, 30% of the variance or change in how long a salesperson stays can be explained by their extraversion score

✓ 1D

**For the collinear relationship reported in 1A, provide a hypothesis that could explain how the one variable value may be causing the other. How should we address this issue?**

Written Exam and IQ are likely collinear because they are both measuring some form of intelligence

✓ Q2

**Run a LINEAR REGRESSION ANALYSIS to create a model predicting the number of months a newly hired sales associate at an Innergystic dealership will remain on the job. If you have two variables**

that appear to be collinear, only include one or the other (retaining the more broadly applicable of the two variables).

Split the data into a training and a test set, use the training set to fit the model, and then evaluate its performance on the test set.

```
df.columns
```

```
Index(['SPID', 'Age', 'Gender', 'Ex', 'Su', 'Brand', 'HSGrad', 'CollegeGrad',
      'Married', 'IQ', 'WritExam', 'Months'],
      dtype='object')
```

```
x = df.drop(['SPID', 'WritExam', 'Months'], axis=1) # select the IVs
x = pd.get_dummies(data = x, drop_first = True)      # input parameters for creating
x.head()
```

	Age	Ex	Su	IQ	Gender_male	Brand_Honda	Brand_Subaru	HSGrad_Y	CollegeGrad
0	54	3.0	4.5	95	0	1	0	1	
1	35	7.0	5.5	99	1	0	1	1	
2	40	3.6	3.5	112	1	1	0	1	
3	34	6.2	1.5	109	0	1	0	1	
4	40	6.4	1.5	86	1	0	1	0	

Next steps: [View recommended plots](#)

```
y = df['Months'] # set DV
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_
```

```
print(x_train.shape) # print the shape
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(238, 10)
(60, 10)
(238,)
(60,)
```

✓ 2A

**Generate a table of the summary statistics of your regression model**

```
x_train_Sm = sm.add_constant(x_train)
ls = sm.OLS(y_train, x_train_Sm).fit()
# produce the summary statistics table
print(ls.summary())
```

### OLS Regression Results

Dep. Variable:	Months	R-squared:	0.800
Model:	OLS	Adj. R-squared:	0.792
Method:	Least Squares	F-statistic:	91.06
Date:	Tue, 05 Mar 2024	Prob (F-statistic):	1.12e-73
Time:	20:43:51	Log-Likelihood:	-625.95
No. Observations:	238	AIC:	1274.
Df Residuals:	227	BIC:	1312.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	29.4153	3.421	8.599	0.000	22.675	36.155
Age	0.0406	0.032	1.289	0.199	-0.021	0.102
Ex	2.6562	0.137	19.430	0.000	2.387	2.925
Su	0.4382	0.127	3.443	0.001	0.187	0.689
IQ	-0.3968	0.032	-12.547	0.000	-0.459	-0.334
Gender_male	0.2629	0.527	0.499	0.618	-0.775	1.301
Brand_Honda	5.5074	0.560	9.840	0.000	4.405	6.610
Brand_Subaru	5.4110	0.551	9.813	0.000	4.324	6.498
HSGrad_Y	6.3004	0.583	10.810	0.000	5.152	7.449
CollegeGrad_Y	-7.2430	0.536	-13.504	0.000	-8.300	-6.186
Married_Y	-1.0877	0.462	-2.352	0.020	-1.999	-0.176

Omnibus:	23.057	Durbin-Watson:	1.911
Prob(Omnibus):	0.000	Jarque-Bera (JB):	34.419
Skew:	0.604	Prob(JB):	3.36e-08
Kurtosis:	4.419	Cond. No.	1.65e+03

#### Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly
- [2] The condition number is large, 1.65e+03. This might indicate that there are strong multicollinearity or other numerical problems.

## 2B

Which if any IV are not significant at the  $\alpha = 0.05$  level? How do you know this?

Age and Gender\_male are not statistically significant, we know this because they have p values greater than 0.05

## ✓ 2C

**What is the value of the intercept coefficient? What does this value tell us?**

The intercept is the average value of y when all Xs are set to 0, so in this case we'd expect a 0 year old, female, etc. to stay 29 months

## ✓ Q3

**Based on your answer from 2B, formulate and apply the model developed above by responding to each of the following:**

## ✓ 3A

**Remove the IV with the highest p-value, rerun the model, then generate the summary statistics table. Repeat as necessary until all non-significant IVs have been removed. With each iteration, state which IV has been removed.**

```
x_train_Sm = x_train_Sm.drop(['Gender_male'], axis=1)
ls=sm.OLS(y_train,x_train_Sm).fit()
# print the summary table
print(ls.summary())
```

OLS Regression Results						
=====						
Dep. Variable:	Months	R-squared:	0.800			
Model:	OLS	Adj. R-squared:	0.792			
Method:	Least Squares	F-statistic:	101.5			
Date:	Tue, 05 Mar 2024	Prob (F-statistic):	1.22e-74			
Time:	20:43:51	Log-Likelihood:	-626.08			
No. Observations:	238	AIC:	1272.			
Df Residuals:	228	BIC:	1307.			
Df Model:	9					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975
-----						
const	29.7028	3.367	8.823	0.000	23.069	36.336
Age	0.0407	0.031	1.292	0.198	-0.021	0.102
Ex	2.6601	0.136	19.523	0.000	2.392	2.928
Su	0.4294	0.126	3.412	0.001	0.181	0.677
IQ	-0.3975	0.032	-12.605	0.000	-0.460	-0.335

Brand_Honda	5.4968	0.558	9.844	0.000	4.397	6.59
Brand_Subaru	5.3883	0.549	9.822	0.000	4.307	6.46
HSGrad_Y	6.2977	0.582	10.823	0.000	5.151	7.44
CollegeGrad_Y	-7.2472	0.535	-13.536	0.000	-8.302	-6.19
Married_Y	-1.0663	0.460	-2.320	0.021	-1.972	-0.16

Omnibus:	20.303	Durbin-Watson:	1.904
Prob(Omnibus):	0.000	Jarque-Bera (JB):	29.697
Skew:	0.548	Prob(JB):	3.56e-07
Kurtosis:	4.339	Cond. No.	1.62e+03

## Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly  
 [2] The condition number is large, 1.62e+03. This might indicate that there are strong multicollinearity or other numerical problems.

```
x_train_Sm = x_train_Sm.drop(['Age'], axis=1)           # drop tl
ls=sm.OLS(y_train,x_train_Sm).fit()                   # retrain tl
print(ls.summary())                                   # produce tl
```

## OLS Regression Results

Dep. Variable:	Months	R-squared:	0.799
Model:	OLS	Adj. R-squared:	0.792
Method:	Least Squares	F-statistic:	113.6
Date:	Tue, 05 Mar 2024	Prob (F-statistic):	2.53e-75
Time:	20:43:51	Log-Likelihood:	-626.95
No. Observations:	238	AIC:	1272.
Df Residuals:	229	BIC:	1303.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975
const	31.1721	3.173	9.823	0.000	24.919	37.42
Ex	2.6784	0.136	19.735	0.000	2.411	2.94
Su	0.4316	0.126	3.425	0.001	0.183	0.68
IQ	-0.3987	0.032	-12.630	0.000	-0.461	-0.33
Brand_Honda	5.5672	0.557	10.004	0.000	4.471	6.66
Brand_Subaru	5.4733	0.545	10.034	0.000	4.399	6.54
HSGrad_Y	6.2509	0.582	10.748	0.000	5.105	7.39
CollegeGrad_Y	-7.2494	0.536	-13.520	0.000	-8.306	-6.19
Married_Y	-0.9823	0.456	-2.155	0.032	-1.880	-0.08

Omnibus:	20.675	Durbin-Watson:	1.912
Prob(Omnibus):	0.000	Jarque-Bera (JB):	29.494
Skew:	0.567	Prob(JB):	3.94e-07
Kurtosis:	4.299	Cond. No.	1.43e+03

## Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly  
 [2] The condition number is large, 1.43e+03. This might indicate that there are strong multicollinearity or other numerical problems.

### ✓ 3B

Once you've run a regression in which all IVs are significant ( $\alpha = 0.05$ ), write out the full model in equation form.

$$\text{Months} = 31.1721 + 2.6784 * \text{Ex} + 0.4316 * \text{Su} - 0.3987 * \text{IQ} + 5.4733 * \text{Brand\_Hond} \\ * \text{HSGrad}_Y - 7.2494 * \text{CollegeGrad}_Y - 0.9823 * \text{Married}_Y$$

### ✓ 3C

Let's say that a Suburu dealership owned by the company is considering hiring a certain job applicant into a sales position. Using the formula above, how long would you expect the applicant to stay in the position if they were a single, 20 year-old female high school grad with no college experience, a 100 IQ and WritExam score of 80%, an Extraverison score of 5, and an SuMind Score of 1? Show your math.

```
months = 31.1721 + (2.6784*5) + (0.4316*1) - (0.3987*100) + (5.5672*0) + (5.4733*1)
print("predicted_months =", months)

predicted_months = 16.8499
```

### ✓ 3D

Does HSGrad\_Y have a postive or negative relationship with Months? What about CollegeGrad\_Y? What does it mean in relation to the reference groups (HSGrad\_N and CollegeGrad\_N)?

HSGrad\_Y has a positive relationship with months, meaning HS grads stay longer than non-HS grads on average. Specifically about 6 months longer.

CollegeGrad\_Y has a negative relationship with months, meaning college grads stay less time than non-college grads on average. About 7 months shorter.

### ✓ Q4



**Now let's evaluate the model performance on your test set to examine how well our model might generalize to new data:**

#### ✓ 4A

**Using the `LinearRegression` method in Scikit learn, fit the final model to the training data and then apply the model to the test data.**

```
model = LinearRegression()
model.fit(x_train_Sm, y_train) # train the regression model

x_test = x_test.drop(['Age', 'Gender_male'], axis=1) # drop the two variables
x_test_Sm = sm.add_constant(x_test) # add a constant (intercept) value

# make predictions onto the test set
predictions = model.predict(x_test_Sm)
```

#### ✓ 4B

**Using the metric method, produce the mean squared error and  $R^2$  for the model, how does this  $R^2$  compare to the value generated on the training data? What does this indicate?**

```
print("Mean squared error =", round(metric.mean_squared_error(y_test, predictions), 2))
print("R2 score =", round(metric.r2_score(y_test, predictions), 2))
```

```
Mean squared error = 8.66
R2 score = 0.85
```

```
print("RMSE = ", np.sqrt(8.66))
```

```
RMSE = 2.9427877939124323
```

We have an R-squared that is on par (if not a bit higher) for our test data, as compared to our training data, indicating that our model should generalize well when applied to new data

#### ✓ 4C

**Produce a visualization of the relationship between the observed (actual) values of months in the test set and the predicted values of months**

```
import seaborn as sns
sns.regplot(x = y_test, y = predictions)
plt.ylabel("Predicted Months")
plt.xlabel("Actual Months")
plt.show()
```

# insert

