Carrying out a Eli5 style permutation test of variable importance

This is a numerical approach to understanding which variables are most important in a predictive model we have built. Eli5 is a libary that does permutation testing of variable importance.

We are not going to use Eli5 today, that will be next time. We will create our own permutation test using Python code to see what effect randomizing each variable, one at a time, has on the predicted performance of the model.

We are going to use a linear model analyzed with a linear regression model, and see what the relative importance of three variables is.

Note that people have the tendency to identify the most important model in the model as being the most important model in the real world. But, for a variety of reasons (correlation among variables, missing variables, or oddities in the model structure), what is important in a model may not be what is important in the external world.

In many cases, you really do want to know what the model is doing in making predictions. You really don't want to see a proxy for age, gender or race being the primary factor in a model of loan eligibility for example.

In [1]:
```
import numpy as np
import pandas as pd
import statsmodels. api as sm
```

Generate predictors x1, x2, x2 and an output y of known form, then we will prredict the importance of each variable based on the Epi5 style model

Note this is a generative use of a model, or synthetic data, so we know what the structure is and can learn to use the method

In [2]:
```
import numpy.random

# we are just setting up an example data set of a r

x1=np.random.normal(0,3,30)
x2=np.random.normal(0,2,30)
x3=np.random.normal(0,2,30)

y=2*x1-3*x2+ np.random.normal(0,2,30)
```

Which two variables are important in predicting y?

Which variable has no influence on y?

Does y have some "error", or "noise" or "unexplained variance" which is not predicted by x1, x2 or x3?

Put things into a pandas array

In [3]:
```
X=pd.DataFrame(x1,columns=['x1'])
X['x2']=x2
X['x3']=x3
```

In [4]:
```
X.head()
```

Out[4]:

	x1	x2	x3
0	0.135349	-3.269062	-1.488947
1	1.968518	2.750372	2.186532
2	-1.485521	1.597218	0.254440
3	-1.158635	0.532057	1.484048
4	-3.487669	-0.045296	3.756608

In [5]:
```
# add a constant column to the predictors, this res
X=sm.add_constant(X,prepend=False)
```

In [6]:
```
#gotta check matters...
X.head()
```

Out[6]:

	x1	x2	x3	const
0	0.135349	-3.269062	-1.488947	1.0
1	1.968518	2.750372	2.186532	1.0
2	-1.485521	1.597218	0.254440	1.0
3	-1.158635	0.532057	1.484048	1.0
4	-3.487669	-0.045296	3.756608	1.0

Classical approaches to predictor importance

There is a set of classical statistical methods known as Analysis of Variance (ANOVA). It is meant as a way to determine the amound of variance explained by each term in a model

In [7]:
```
# here is the linear regression model,   Ordinary L
# this is from the statsmodels package

results = sm.OLS(y,X).fit()
print(results.summary())
```

```
                            OLS Regression Results
===================================================
===========================
Dep. Variable:                      y   R-squared:
0.950
Model:                            OLS   Adj. R-squa
red:                     0.944
Method:                 Least Squares   F-statisti
c:                        164.0
Date:                Fri, 26 Jan 2024   Prob (F-sta
tistic):             5.23e-17
Time:                        01:34:40   Log-Likelih
ood:                   -54.413
No. Observations:                  30   AIC:
116.8
Df Residuals:                      26   BIC:
122.4
Df Model:                           3
Covariance Type:            nonrobust
===================================================
===========================
                 coef    std err          t      P>
|t|      [0.025      0.975]
---------------------------------------------------
---------------------------
x1             1.8103      0.098     18.459      0.
000       1.609       2.012
x2            -3.0498      0.189    -16.132      0.
000      -3.438      -2.661
x3             0.1238      0.189      0.656      0.
518      -0.264       0.512
const          0.2482      0.326      0.761      0.
454      -0.423       0.919
===================================================
===========================
Omnibus:                        0.196   Durbin-Wats
on:                      2.264
Prob(Omnibus):                  0.906   Jarque-Bera
(JB):                   0.284
Skew:                           0.170   Prob(JB):
```

```
0.868
Kurtosis:                          2.665   Cond. No.
3.72
==================================================================
============================

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

What does this result mean?

Is the overall model, that y is predicted by the whole set (x1,x2,x3 and the constant) statistically significant? How do you know this?

Of the predictor variables, x1,x2,x3 which appear to be meaningful predictors? How do you know this?

Add your answer here

In [8]:
```
dir(results)
```

Out[8]:
```
['HC0_se',
 'HC1_se',
 'HC2_se',
 'HC3_se',
 '_HCCM',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_abat_diagonal',
 '_cache',
 '_data_attr',
 '_data_in_cache',
 '_get_robustcov_results',
 '_get_wald_nonlinear',
 '_is_nested',
 '_transform_predict_exog',
 '_use_t',
```

```
                '_wexog_singular_values',
                'aic',
                'bic',
                'bse',
                'centered_tss',
                'compare_f_test',
                'compare_lm_test',
                'compare_lr_test',
                'condition_number',
                'conf_int',
                'conf_int_el',
                'cov_HC0',
                'cov_HC1',
                'cov_HC2',
                'cov_HC3',
                'cov_kwds',
                'cov_params',
                'cov_type',
                'df_model',
                'df_resid',
                'diagn',
                'eigenvals',
                'el_test',
                'ess',
                'f_pvalue',
                'f_test',
                'fittedvalues',
                'fvalue',
                'get_influence',
                'get_prediction',
                'get_robustcov_results',
                'info_criteria',
                'initialize',
                'k_constant',
                'llf',
                'load',
                'model',
                'mse_model',
                'mse_resid',
                'mse_total',
```

```
 'nobs',
 'normalized_cov_params',
 'outlier_test',
 'params',
 'predict',
 'pvalues',
 'remove_data',
 'resid',
 'resid_pearson',
 'rsquared',
 'rsquared_adj',
 'save',
 'scale',
 'ssr',
 'summary',
 'summary2',
 't_test',
 't_test_pairwise',
 'tvalues',
 'uncentered_tss',
 'use_t',
 'wald_test',
 'wald_test_terms',
 'wresid']
```

In [9]:

```
# extract the R^2 value we will use it as our metri
obs_r2=results.rsquared
print(obs_r2)
```

```
0.9498159372549415
```

In [9]:

```
```

In [10]:

```
x1_change=np.empty(100)

for k in np.arange(0,100,1,dtype="int32"):
    Xtemp=X.copy()
    Xtemp['x1']=np.random.permutation(Xtemp['x1'])
    modelx=sm.OLS(y,Xtemp)
    resx=modelx.fit()
```

```
    x1_change[k]=abs(resx.rsquared-obs_r2)


x1_change.mean()
```

Out[10]:

```
0.6272855585816256
```

Question

Explain what is happening the the loop above.

What is the value of x1_change.mean() telling you?

If this value (x1_change.mean()) is large, what does that imply about x1?

What if this change.mean is small or even negative?

Add your answer here

Answer:

- Calculating the mean squared error from randomizing the variables
- If the value is large the varible is important and had a big impact on the model
- If the value is small the varible is important and had a small impact on the model

Question

Find the change in the R^2 produced when x2 and x3 are permuted

Use these values to produce a relative ranking of the importance of the 3 variables

Cut and paste my code above into cells below, and then modify my code to check whether or not x2 and x3 are useful as predictors.

In [12]:
```
x2_change=np.empty(100)

for k in np.arange(0,100,1,dtype="int32"):
    Xtemp=X.copy()
    Xtemp['x2']=np.random.permutation(Xtemp['x2'])
    modelx=sm.OLS(y,Xtemp)
    resx=modelx.fit()
    x2_change[k]=abs(resx.rsquared-obs_r2)


x2_change.mean()
```

Out[12]:
```
0.4799566108778029
```

In [13]:
```
x3_change=np.empty(100)

for k in np.arange(0,100,1,dtype="int32"):
    Xtemp=X.copy()
    Xtemp['x3']=np.random.permutation(Xtemp['x3'])
    modelx=sm.OLS(y,Xtemp)
    resx=modelx.fit()
    x1_change[k]=abs(resx.rsquared-obs_r2)


x1_change.mean()
```

Out[13]: 0.0015700616886188711

In [ ]:
```
# - Because X1 was the largest varible is important
```