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.

```
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

```
import numpy.random

# we are just setting up an example data set of a relative complex relationship

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?

X1 and x2 are important

Which variable has no influence on y? x3 has no influence

Does y have some "error", or "noise" or "unexplained variance" which is not predicted by x1, x2 or x3? yes from np.random.normal(0,2,30) written after the x1 and x2

Put things into a pandas array

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

```
X.head()
```

	x1	x2	x3
0	-2.088312	0.849560	-0.953572
1	3.878657	-0.579104	4.095545
2	0.297128	0.205443	1.536150
3	-0.945617	1.502165	2.754302
4	3.566879	-1.256781	-3.519012

```
# add a constant column to the predictors, this results in a constant value in the linear model, in the approach used in statsmodels
X=sm.add_constant(X,prepend=False)
```

```
#gotta check matters...
X.head()
```

	x1	x2	x3	const
0	-2.088312	0.849560	-0.953572	1.0
1	3.878657	-0.579104	4.095545	1.0
2	0.297128	0.205443	1.536150	1.0
3	-0.945617	1.502165	2.754302	1.0
4	3.566879	-1.256781	-3.519012	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

```
# here is the linear regression model,   Ordinary Least Squares (OLS)
# this is from the statsmodels package

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.958
Model:                            OLS   Adj. R-squared:                  0.954
Method:                 Least Squares   F-statistic:                     199.3
Date:                Mon, 05 Feb 2024   Prob (F-statistic):           4.69e-18
Time:                        20:29:32   Log-Likelihood:                -64.966
No. Observations:                  30   AIC:                             137.9
Df Residuals:                      26   BIC:                             143.5
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             1.6697      0.144     11.557      0.000       1.373       1.967
x2            -3.4718      0.225    -15.450      0.000      -3.934      -3.010
x3            -0.1706      0.173     -0.986      0.333      -0.526       0.185
const          0.0438      0.426      0.103      0.919      -0.832       0.919
==============================================================================
Omnibus:                        0.126   Durbin-Watson:                   2.601
Prob(Omnibus):                  0.939   Jarque-Bera (JB):                0.258
Skew:                          -0.135   Prob(JB):                        0.879
Kurtosis:                       2.634   Cond. No.                         3.37
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix 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?

The p-value of the f statistic is 1.99e-15 < 0.05 so at the 5% the overall model is statistically significant.

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

X1 and X2 has p values of 0.000 so they are statistically significant at the 5% level

X3 with a p value of 0.884 is not statistically significant at the 5% level

Add your answer here

```
dir(results)
```

```
     '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']
```

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

```
0.9583246919863257
```

```
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()
```

```
0.2052373324356581
```

Question

1.Explain what is happening the the loop above.

It's going to loop a 100 times, it creates a copy of X then create permutaions of x1 and then fit the OLS model. Then compare it's R^2 diffrence from the intitial R^2. Do this a 100 times.

2.What is the value of x1_change.mean() telling you? At the end take the average of the 100 diffrences.

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

It implies that it's important.

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

implies that it is not important.

I wrote my ansers below the questions

Add your answer here

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.

```
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()
```

```
0.36684577460785867
```

```
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()
    x3_change[k]=abs(resx.rsquared-obs_r2)
```

```
x3_change.mean()
```

```
0.001343120421186601
```

Sure enough it agrees with our hypothesis testing that X1 and X2 are statistically significant in the model as there is a high x_change in x1 and x2 while x3 which was initially demmed not statistically significant was found to not be significant with this test.

```
Start coding or generate with AI.
```