week09

October 27, 2024

1 STOR 320 Introduction to Data Science

1.1 LINEAR REGRESSION IN PYTHON

This lecture is comprised of two main parts. We will see two ways of running Multiple Linear Regression: - 1. MLR with only numerical variables - 2. MLR with numerical + categorical variables

We will use the Linear Regression package from the `statsmodels` library. This library includes a variety of functions that are helpful for data exploration and statiscal models, its documentation can be found here: https://www.statsmodels.org/stable/index.html.

In the terminal, type the following commands to install `statsmodels` (change the package name to install other packages): - 1. `conda install -c conda-forge statsmodels`, or - 2. `pip install statsmodels`

More specifically, we will use Linear Regression to predict the quality of the wines as measured by their 'Auction Index'

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
import statsmodels.formula.api as smf
```

[ ]:
```
wine = pd.read_csv('wine_agg.csv')
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Year             46 non-null     int64
 1   LogAuctionIndex  46 non-null     float64
 2   WinterRain       46 non-null     float64
 3   HarvestRain      46 non-null     float64
 4   GrowTemp         46 non-null     float64
 5   HarvestTemp      46 non-null     float64
 6   Age              46 non-null     int64
 7   FrancePop        46 non-null     float64
```

```
 8   USAlcConsump     46 non-null     float64
dtypes: float64(7), int64(2)
memory usage: 3.4 KB
```

[ ]:
```
wine.describe()
```

[ ]:
```
              Year  LogAuctionIndex  WinterRain  HarvestRain   GrowTemp  \
count    46.000000        46.000000   46.000000    46.000000  46.000000
mean   1977.195652         7.137596  547.141304   150.341304  17.294348
std      13.889436         0.656632  123.979187    76.963817   0.894670
min    1952.000000         5.868500  282.400000    37.200000  15.270000
25%    1966.250000         6.671825  482.025000    87.675000  16.650000
50%    1977.500000         7.150400  529.300000   126.050000  17.280000
75%    1988.750000         7.584275  645.025000   186.525000  17.840000
max    2000.000000         8.493700  755.200000   341.600000  19.100000

       HarvestTemp        Age   FrancePop  USAlcConsump
count    46.000000  46.000000  46.000000     46.000000
mean     17.951304  37.804348  52.384783      9.150870
std       1.525705  13.889436   4.808421      1.074219
min      14.390000  15.000000  42.460000      7.740000
25%      16.800000  26.250000  49.257500      8.135000
50%      17.845000  37.500000  53.265000      9.055000
75%      19.030000  48.750000  56.345000     10.190000
max      21.050000  63.000000  59.050000     10.950000
```

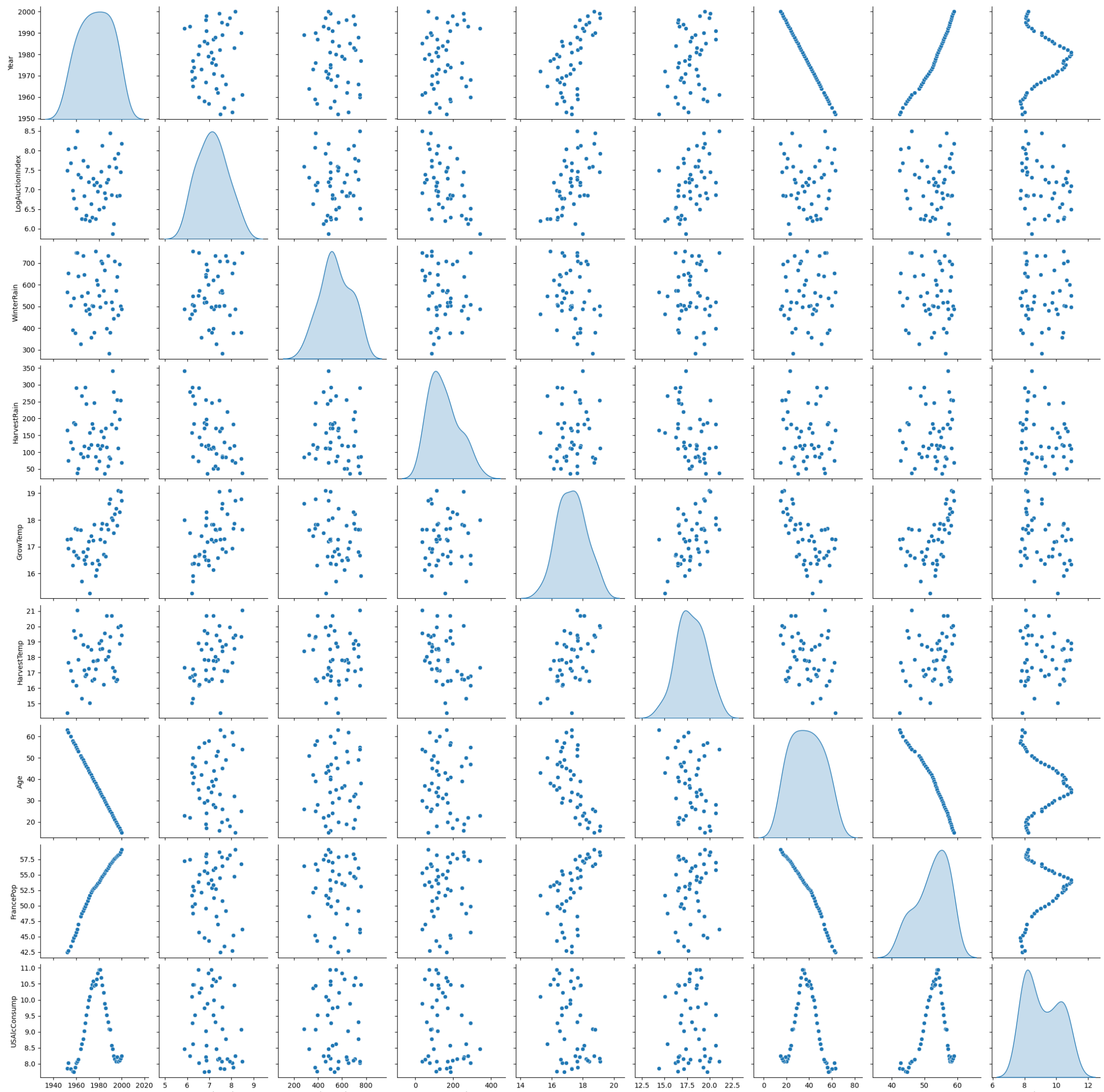
[ ]:
```
wine.head(5)
```

[ ]:
```
   Year  LogAuctionIndex  WinterRain  HarvestRain  GrowTemp  HarvestTemp  Age  \
0  1952           7.4950       566.4        165.5     17.28        14.39   63
1  1953           8.0393       653.3         75.6     16.94        17.64   62
2  1955           7.6858       504.3        129.5     17.30        17.13   60
3  1957           6.9845       390.8        110.4     16.31        16.47   58
4  1958           6.7772       538.8        187.0     16.82        19.72   57

   FrancePop  USAlcConsump
0      42.46          7.85
1      42.75          8.03
2      43.43          7.84
3      44.31          7.77
4      44.79          7.74
```

[ ]:
```
# Plot scatter matrix for each pair of variables off diagonal and the
 ↪histograms (or density plots) on the diagonal
# In ggplot2 in R, one can use ggscatmat, which also prints the correlation in
 ↪the upper triangle.
import seaborn as sns
```

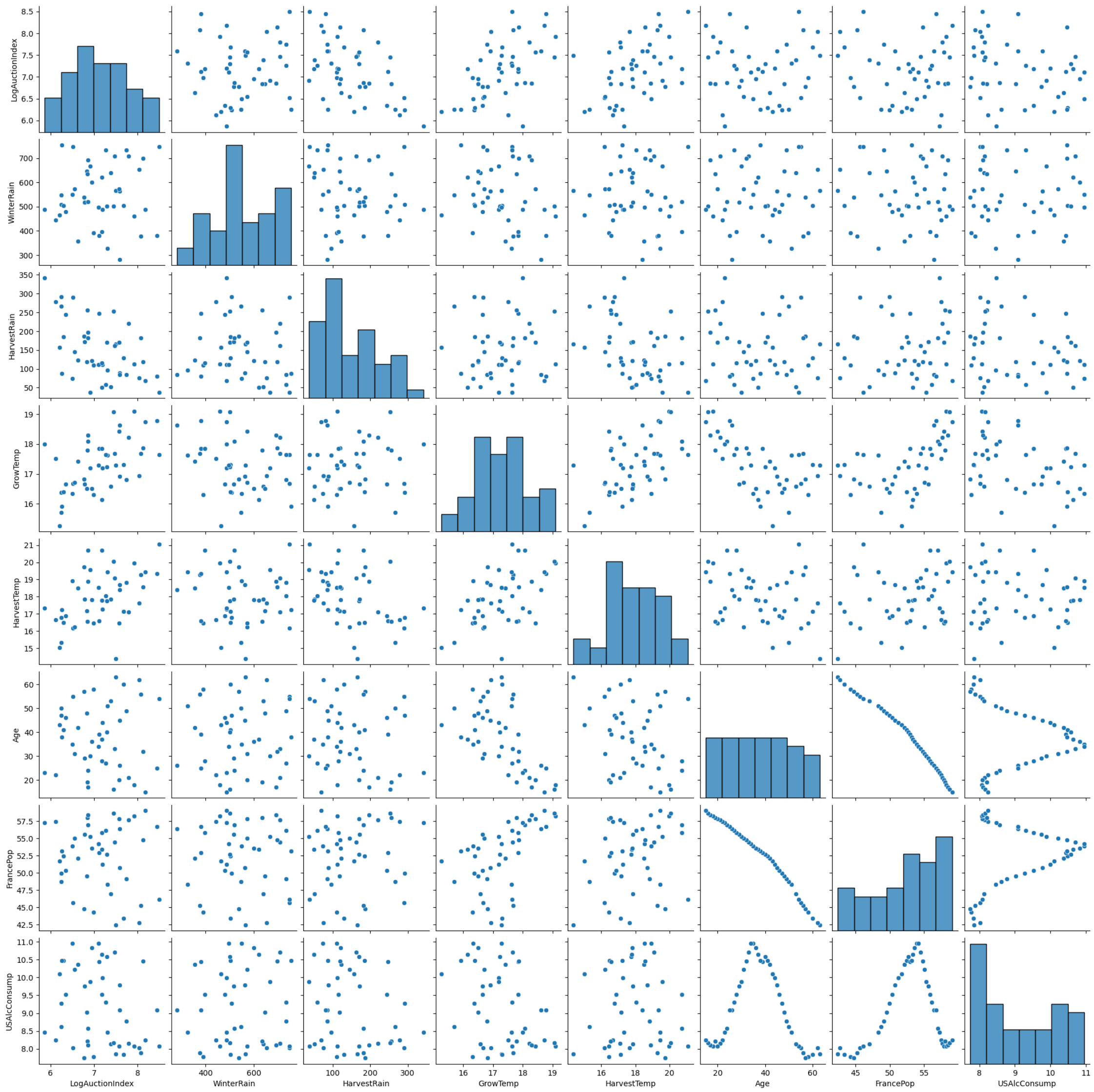
```
sns.pairplot(wine.iloc[:,0:9],diag_kind='kde') # try diag_kind='hist' for␣
↪histograms
```

[ ]: <seaborn.axisgrid.PairGrid at 0x14bfdb080>



[ ]:
```
sns.pairplot(wine.iloc[:,1:9],diag_kind='hist')
```

[ ]: <seaborn.axisgrid.PairGrid at 0x150ae7e30>



[ ]:
```python
# If you apply .corr() directly to your dataframe,
# it will return all pairwise correlations between your columns;
wine.corr()
```

[ ]:
```
                     Year  LogAuctionIndex  WinterRain  HarvestRain  GrowTemp  \
Year             1.000000        -0.011346   -0.032424     0.128569  0.600924
LogAuctionIndex -0.011346         1.000000    0.058326    -0.525882  0.559727
WinterRain      -0.032424         0.058326    1.000000    -0.120258 -0.214181
HarvestRain      0.128569        -0.525882   -0.120258     1.000000  0.036437
GrowTemp         0.600924         0.559727   -0.214181     0.036437  1.000000
HarvestTemp      0.277000         0.469832   -0.046874    -0.410439  0.513076
Age             -1.000000         0.011346    0.032424    -0.128569 -0.600924
FrancePop        0.986137        -0.076993   -0.045196     0.112394  0.516918
```

```
USAlcConsump     0.131949        -0.271188    0.004174    -0.220185 -0.352116

                 HarvestTemp        Age  FrancePop  USAlcConsump
Year                0.277000 -1.000000   0.986137      0.131949
LogAuctionIndex     0.469832  0.011346  -0.076993     -0.271188
WinterRain         -0.046874  0.032424  -0.045196      0.004174
HarvestRain        -0.410439 -0.128569   0.112394     -0.220185
GrowTemp            0.513076 -0.600924   0.516918     -0.352116
HarvestTemp         1.000000 -0.277000   0.250442     -0.035964
Age                -0.277000  1.000000  -0.986137     -0.131949
FrancePop           0.250442 -0.986137   1.000000      0.269647
USAlcConsump       -0.035964 -0.131949   0.269647      1.000000
```

2 Weekly class activity 1: What do you observe from the correlation table?

Age and Year have a negative correlation score of -1. FrancePop and Year(/Age) also are correlated.

2.1 1. MLR (with only numerical variables)

2.1.1 Dataset train-test splitting

Next, we will split the dataset into a training set and a test set. There are various ways of splitting the dataset, we will first do an example of chronological separation. Eventually we'll cover randomized splitting as well.

We can do chronological splitting sing Boolean predicates.

[ ]:
```
wine_train = wine[wine['Year'] <= 1985]
wine_test = wine[wine['Year'] > 1985]

len(wine_train), len(wine_test) # aim for 70% train, 30% test
```

[ ]: (31, 15)

[ ]:
```

```

3 Weekly class activity 2: In the training set, how many wineries are older than 40?

[ ]:
```
count = wine_train[wine_train["Age"] > 40].count()
count.iloc[0]
```

[ ]: 20

[ ]:
```
len(wine_train[wine_train["Age"] > 40])
```

[ ]: 20

3.0.1 Training the model

We will show two ways of passing the data into the model:

- We can select the columns of interest that will constitute our matrices
- Or we can use syntaxis that follows R-style formulas

3.0.2 1) Training the model (matrix style)

https://www.statsmodels.org/stable/examples/notebooks/generated/ols.html

[ ]:
```
# Import the library that contains all the functions/modules related to the␣
↪regression model
import statsmodels.api as sm

# Choose the features to be used
cols = ['WinterRain', 'HarvestRain', 'GrowTemp', 'HarvestTemp', 'Age',␣
↪'FrancePop', 'USAlcConsump']
X_train = wine_train[cols]
y_train = wine_train['LogAuctionIndex']

# We must add an intercept as the standard model doesn't automatically fit one
X_train = sm.add_constant(X_train)

# Fit the data to the model
model1 = sm.OLS(y_train, X_train).fit() #ordinary least square
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        LogAuctionIndex   R-squared:                       0.789
Model:                            OLS   Adj. R-squared:                  0.725
Method:                 Least Squares   F-statistic:                     12.31
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           1.86e-06
Time:                        16:17:32   Log-Likelihood:                -5.0600
No. Observations:                  31   AIC:                             26.12
Df Residuals:                      23   BIC:                             37.59
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const           -4.9663      9.382     -0.529      0.602     -24.375      14.443
WinterRain       0.0012      0.001      2.108      0.046     2.2e-05       0.002
HarvestRain     -0.0033      0.001     -3.112      0.005      -0.006      -0.001
GrowTemp         0.6583      0.122      5.387      0.000       0.405       0.911
```

```
HarvestTemp      0.0044      0.060      0.074      0.942      -0.120       0.129
Age              0.0240      0.051      0.473      0.641      -0.081       0.129
FrancePop       -0.0290      0.137     -0.212      0.834      -0.312       0.254
USAlcConsump     0.1093      0.168      0.651      0.522      -0.238       0.457
==============================================================================
Omnibus:                        0.539   Durbin-Watson:                   2.719
Prob(Omnibus):                  0.764   Jarque-Bera (JB):                0.618
Skew:                           0.042   Prob(JB):                        0.734
Kurtosis:                       2.313   Cond. No.                     9.35e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.35e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

3.0.3 Plot the coefficients and the confidence intervals

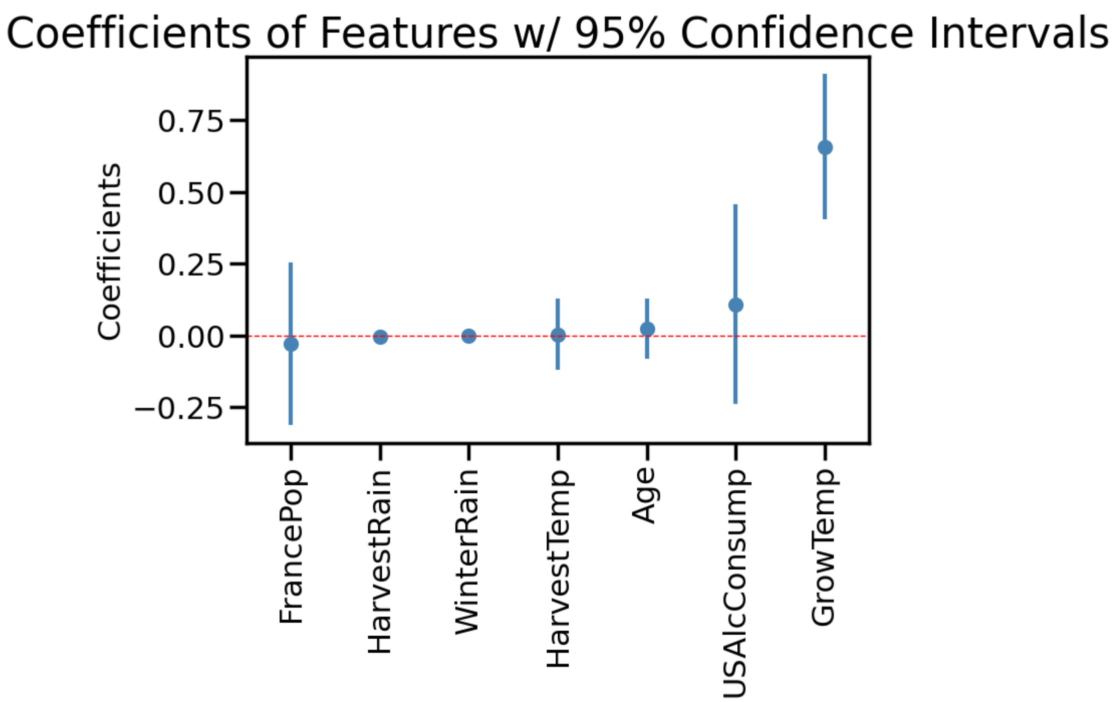
In `R` and `ggplot2`, one can use function `ggcoef` to create such graph easily. However, there is no simple solution to this task in Python. The code below is a user-defined code credit to Jessica Forrest-Baldini (https://medium.com/analytics-vidhya/create-your-own-coefficient-plot-function-in-python-aadb9fe27a77). It involves some work with `Matplotlib`. You are encouraged to figure out why the code works.

[ ]:
```
import matplotlib.pyplot as plt
def coefplot(results):
    '''
    Takes in results of OLS model and returns a plot of
    the coefficients with 95% confidence intervals.

    Removes intercept, so if uncentered will return error.
    '''
    # Create dataframe of results summary
    coef_df = pd.DataFrame(results.summary().tables[1].data)

    # Add column names
    coef_df.columns = coef_df.iloc[0]

    # Drop the extra row with column labels
    coef_df=coef_df.drop(0)

    # Set index to variable names
    coef_df = coef_df.set_index(coef_df.columns[0])

    # Change datatype from object to float
    coef_df = coef_df.astype(float)
```

```python
    # Get errors; (coef - lower bound of conf interval)
    errors = coef_df['coef'] - coef_df['[0.025']

    # Append errors column to dataframe
    coef_df['errors'] = errors

    # Drop the constant for plotting
    coef_df = coef_df.drop(['const'])

    # Sort values by coef ascending
    coef_df = coef_df.sort_values(by=['coef'])

    ### Plot Coefficients ###

    # x-labels
    variables = list(coef_df.index.values)

    # Add variables column to dataframe
    coef_df['variables'] = variables

    # Set sns plot style back to 'poster'
    # This will make bars wide on plot
    sns.set_context("poster")

    # Define figure, axes, and plot
    fig, ax = plt.subplots(figsize=(8, 5))

    # Error bars for 95% confidence interval
    # Can increase capsize to add whiskers
    coef_df.plot(x='variables', y='coef', kind='bar',
                 ax=ax, color='none', fontsize=22,
                 ecolor='steelblue',capsize=0,
                 yerr='errors', legend=False)

    # Set title & labels
    plt.title('Coefficients of Features w/ 95% Confidence␣
↪Intervals',fontsize=30)
    ax.set_ylabel('Coefficients',fontsize=22)
    ax.set_xlabel('',fontsize=22)

    # Coefficients
    ax.scatter(x=np.arange(coef_df.shape[0]),
               marker='o', s=80,
               y=coef_df['coef'], color='steelblue')

    # Line to define zero on the y-axis
    ax.axhline(y=0, linestyle='--', color='red', linewidth=1)
```

```
    return plt.show()
```

[ ]:
```
coefplot(model1)
```



3.0.4 2) Training the model (using R-style formulas)

https://www.statsmodels.org/stable/example_formulas.html#categorical-variables

One of the main advantages of using this type of notation is the fact that categorical variables are handled automatically.

Furthermore, the constant representing the intercept is generated by default in `smf.ols`.

For the remaineder of this lab, we will stick with the formula style notation as presented below:

[ ]:
```
import statsmodels.formula.api as smf

ols = smf.ols(formula='LogAuctionIndex ~ WinterRain + HarvestRain + GrowTemp +␣
 ↪HarvestTemp + Age + FrancePop + USAlcConsump',
                  data=wine_train)
model1 =ols.fit()
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        LogAuctionIndex   R-squared:                       0.789
```

```
Model:                            OLS   Adj. R-squared:                  0.725
Method:                 Least Squares   F-statistic:                     12.31
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           1.86e-06
Time:                        16:17:32   Log-Likelihood:                -5.0600
No. Observations:                  31   AIC:                             26.12
Df Residuals:                      23   BIC:                             37.59
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept     -4.9663      9.382     -0.529      0.602     -24.375      14.443
WinterRain     0.0012      0.001      2.108      0.046     2.2e-05       0.002
HarvestRain   -0.0033      0.001     -3.112      0.005      -0.006      -0.001
GrowTemp       0.6583      0.122      5.387      0.000       0.405       0.911
HarvestTemp    0.0044      0.060      0.074      0.942      -0.120       0.129
Age            0.0240      0.051      0.473      0.641      -0.081       0.129
FrancePop     -0.0290      0.137     -0.212      0.834      -0.312       0.254
USAlcConsump   0.1093      0.168      0.651      0.522      -0.238       0.457
==============================================================================
Omnibus:                        0.539   Durbin-Watson:                   2.719
Prob(Omnibus):                  0.764   Jarque-Bera (JB):                0.618
Skew:                           0.042   Prob(JB):                        0.734
Kurtosis:                       2.313   Cond. No.                     9.35e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.35e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

3.0.5 Evaluating the model

If we want to use evaluation metrics that are not contained in the standard package, such as Out-of-sample R^2, we can manipulate our dataframe's columns and take advantage of `numpy's` matrix operation functions

[ ]:
```
# compute out-of-sample R-squared using the test set
def OSR2(model, df_train, df_test, dependent_var):
    y_test = df_test[dependent_var]
    y_pred = model.predict(df_test)
    SSE = np.sum((y_test - y_pred)**2)
    SST = np.sum((y_test - np.mean(df_train[dependent_var]))**2)
    return 1 - SSE/SST
```

[ ]:
```
OSR2(model1, wine_train, wine_test, 'LogAuctionIndex')
```

[ ]: 0.5377506779455627

Question: Since out-of-sample R-squared is the performance on the test set, why do we need training set as one of the inputs in `OSR2` function?

It's needed to calculate the SST.

3.0.6 Variance Inflation Factor (VIF) measures multicollinearity

[ ]:
```
# calculate Variance Inflation Factor for each explanatory variable
from statsmodels.stats.outliers_influence import variance_inflation_factor

# The dataframe passed to VIF must include the intercept term. We add it the␣
↪same way we did before.
def VIF(df, columns):
    values = sm.add_constant(df[columns]).values
    num_columns = len(columns)+1
    vif = [variance_inflation_factor(values, i) for i in range(num_columns)]
    return pd.Series(vif[1:], index=columns)

cols = ['WinterRain', 'HarvestRain', 'GrowTemp', 'HarvestTemp', 'Age',␣
↪'FrancePop', 'USAlcConsump']
VIF(wine_train, cols)
```

[ ]:
```
WinterRain       1.295370
HarvestRain      1.578682
GrowTemp         1.700079
HarvestTemp      2.198191
Age             66.936256
FrancePop       81.792302
USAlcConsump    10.441217
dtype: float64
```

3.0.7 Refine the model

4 In-class acitivity 3: To remove variables, which should we look at first, VIF or p-value?

Multicollinearity could artificially inflate/deflate p-values, should deal with that first.

[ ]:
```
# remove FrancePop because of its high VIF
model2 = smf.ols(formula='LogAuctionIndex ~ WinterRain + HarvestRain + GrowTemp␣
↪+ HarvestTemp + Age + USAlcConsump',
                 data=wine_train).fit()
print(model2.summary())

cols = ['WinterRain', 'HarvestRain', 'GrowTemp', 'HarvestTemp', 'Age',␣
↪'USAlcConsump']
```

```
VIF(wine_train, cols) # r-squared is the same
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        LogAuctionIndex   R-squared:                       0.789
Model:                            OLS   Adj. R-squared:                  0.736
Method:                 Least Squares   F-statistic:                     14.95
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           4.60e-07
Time:                        16:17:32   Log-Likelihood:                -5.0902
No. Observations:                  31   AIC:                             24.18
Df Residuals:                      24   BIC:                             34.22
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept     -6.8405      3.071     -2.228      0.036     -13.178      -0.503
WinterRain     0.0012      0.001      2.266      0.033       0.000       0.002
HarvestRain   -0.0034      0.001     -3.294      0.003      -0.005      -0.001
GrowTemp       0.6671      0.113      5.930      0.000       0.435       0.899
HarvestTemp    0.0021      0.058      0.036      0.972      -0.117       0.121
Age            0.0341      0.018      1.912      0.068      -0.003       0.071
USAlcConsump   0.0933      0.147      0.634      0.532      -0.210       0.397
==============================================================================
Omnibus:                        0.388   Durbin-Watson:                   2.726
Prob(Omnibus):                  0.824   Jarque-Bera (JB):                0.534
Skew:                           0.040   Prob(JB):                        0.766
Kurtosis:                       2.362   Cond. No.                     3.11e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.11e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]:
```
WinterRain      1.223155
HarvestRain     1.509070
GrowTemp        1.500905
HarvestTemp     2.122007
Age             8.580764
USAlcConsump    8.350251
dtype: float64
```

[ ]:
```
# remove USAlcConsump
model3 = smf.ols(formula='LogAuctionIndex ~ WinterRain + HarvestRain + GrowTemp␣
 ↪+ HarvestTemp + Age',
                 data=wine_train).fit()
```

```python
print(model3.summary())

cols = ['WinterRain', 'HarvestRain', 'GrowTemp', 'HarvestTemp', 'Age']
VIF(wine_train, cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        LogAuctionIndex   R-squared:                       0.785
Model:                            OLS   Adj. R-squared:                  0.743
Method:                 Least Squares   F-statistic:                     18.30
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           1.21e-07
Time:                        16:17:32   Log-Likelihood:                -5.3480
No. Observations:                  31   AIC:                             22.70
Df Residuals:                      25   BIC:                             31.30
Df Model:                           5
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept      -5.2152      1.672     -3.119      0.005      -8.659      -1.771
WinterRain      0.0011      0.001      2.202      0.037    7.24e-05       0.002
HarvestRain    -0.0034      0.001     -3.433      0.002      -0.005      -0.001
GrowTemp        0.6643      0.111      5.981      0.000       0.436       0.893
HarvestTemp    -0.0067      0.055     -0.120      0.905      -0.121       0.108
Age             0.0235      0.006      3.820      0.001       0.011       0.036
==============================================================================
Omnibus:                        1.212   Durbin-Watson:                   2.663
Prob(Omnibus):                  0.546   Jarque-Bera (JB):                0.897
Skew:                           0.005   Prob(JB):                        0.639
Kurtosis:                       2.167   Cond. No.                     1.72e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.72e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]:
```
WinterRain     1.126094
HarvestRain    1.488511
GrowTemp       1.498590
HarvestTemp    2.002247
Age            1.045928
dtype: float64
```

[ ]:
```python
# remove HarvestTemp
model4 = smf.ols(formula='LogAuctionIndex ~ WinterRain + HarvestRain + GrowTemp␣
 ↪+ Age',
```

```
                 data=wine_train).fit()
print(model4.summary())

cols = ['WinterRain', 'HarvestRain', 'GrowTemp', 'Age']
VIF(wine_train, cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        LogAuctionIndex   R-squared:                       0.785
Model:                            OLS   Adj. R-squared:                  0.752
Method:                 Least Squares   F-statistic:                     23.78
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           2.31e-08
Time:                        16:17:32   Log-Likelihood:                -5.3569
No. Observations:                  31   AIC:                             20.71
Df Residuals:                      26   BIC:                             27.88
Df Model:                           4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept      -5.2164      1.640     -3.180      0.004      -8.588      -1.845
WinterRain      0.0011      0.000      2.246      0.033    9.43e-05       0.002
HarvestRain    -0.0034      0.001     -3.971      0.001      -0.005      -0.002
GrowTemp        0.6569      0.091      7.255      0.000       0.471       0.843
Age             0.0236      0.006      3.940      0.001       0.011       0.036
==============================================================================
Omnibus:                        1.208   Durbin-Watson:                   2.664
Prob(Omnibus):                  0.547   Jarque-Bera (JB):                0.896
Skew:                          -0.013   Prob(JB):                        0.639
Kurtosis:                       2.167   Cond. No.                     1.72e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.72e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]:
```
WinterRain     1.110834
HarvestRain    1.116268
GrowTemp       1.035367
Age            1.029810
dtype: float64
```

4.0.1 Feature Engineering

[ ]:
```
# Let us add an interaction term of GrowTemp and Age, and test its predictive␣
↪power
model5 = smf.ols(formula='LogAuctionIndex ~ WinterRain * HarvestRain + GrowTemp␣
↪+ Age',
                 data=wine_train).fit()
print(model5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        LogAuctionIndex   R-squared:                       0.803
Model:                            OLS   Adj. R-squared:                  0.763
Method:                 Least Squares   F-statistic:                     20.33
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           4.41e-08
Time:                        16:17:32   Log-Likelihood:                -4.0580
No. Observations:                  31   AIC:                             20.12
Df Residuals:                      25   BIC:                             28.72
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
Intercept                 -5.9560      1.680     -3.545      0.002      -9.417
-2.496
WinterRain                 0.0025      0.001      2.364      0.026       0.000
0.005
HarvestRain                0.0025      0.004      0.618      0.542      -0.006
0.011
WinterRain:HarvestRain -1.006e-05   6.81e-06     -1.478      0.152   -2.41e-05
3.96e-06
GrowTemp                   0.6486      0.089      7.309      0.000       0.466
0.831
Age                        0.0244      0.006      4.159      0.000       0.012
0.037
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   2.374
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.613
Skew:                          -0.040   Prob(JB):                        0.736
Kurtosis:                       2.316   Cond. No.                     2.63e+06
==============================================================================

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

[2] The condition number is large, 2.63e+06. This might indicate that there are strong multicollinearity or other numerical problems.

4.1 2. MLR (with numerical + categorical variables)

For this part, we will use a second dataset, `wine_disagg.csv`, which contains additional information related to the Wineries. The `Winery` variable is a `string object`, but we can do some transformations that will help us fit it into the continuous model

[ ]:
```
wine_new = pd.read_csv("wine_disagg.csv")
wine_new_train = wine_new[wine_new['Year'] <= 1985]
wine_new_test = wine_new[wine_new['Year'] > 1985]

wine_new.info()
wine_new.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 147 entries, 0 to 146
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Winery        147 non-null    object
 1   Year          147 non-null    int64
 2   Age           147 non-null    int64
 3   LogAuction    147 non-null    float64
 4   WinterRain    147 non-null    float64
 5   HarvestRain   147 non-null    float64
 6   GrowTemp      147 non-null    float64
 7   HarvestTemp   147 non-null    float64
 8   FrancePop     147 non-null    float64
 9   USAlcConsump  147 non-null    float64
dtypes: float64(7), int64(2), object(1)
memory usage: 11.6+ KB
```

[ ]:
```
             Winery  Year  Age  LogAuction  WinterRain  HarvestRain  \
0      Cheval Blanc  1952   63    6.653108       566.4        165.5
1  Lafite-Rothschild  1952   63    6.861502       566.4        165.5
2      Cheval Blanc  1953   62    6.664192       653.3         75.6
3      Cheval Blanc  1955   60    6.311426       504.3        129.5
4  Lafite-Rothschild  1955   60    6.550209       504.3        129.5

   GrowTemp  HarvestTemp  FrancePop  USAlcConsump
0     17.28        14.39      42.46          7.85
1     17.28        14.39      42.46          7.85
2     16.94        17.64      42.75          8.03
3     17.30        17.13      43.43          7.84
4     17.30        17.13      43.43          7.84
```

```
[ ]: # Simple regression using new data, not yet incorporating the Winery variable
modOld = smf.ols(formula='LogAuction ~ WinterRain + HarvestRain + GrowTemp +␣
 ↪Age',
                 data=wine_new_train).fit()
print(modOld.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             LogAuction   R-squared:                       0.222
Model:                            OLS   Adj. R-squared:                  0.182
Method:                 Least Squares   F-statistic:                     5.567
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           0.000539
Time:                        16:17:32   Log-Likelihood:                -112.14
No. Observations:                  83   AIC:                             234.3
Df Residuals:                      78   BIC:                             246.4
Df Model:                           4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept      -0.2184      3.431     -0.064      0.949      -7.050       6.613
WinterRain      0.0018      0.001      1.855      0.067      -0.000       0.004
HarvestRain     0.0025      0.002      1.261      0.211      -0.001       0.006
GrowTemp        0.1210      0.193      0.628      0.532      -0.263       0.505
Age             0.0529      0.012      4.470      0.000       0.029       0.076
==============================================================================
Omnibus:                        7.497   Durbin-Watson:                   2.634
Prob(Omnibus):                  0.024   Jarque-Bera (JB):                2.891
Skew:                          -0.027   Prob(JB):                        0.236
Kurtosis:                       2.087   Cond. No.                     1.94e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.94e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
[ ]: # compute the out-of-sample R squared
OSR2(modOld,wine_new_train, wine_new_test, 'LogAuction')
```

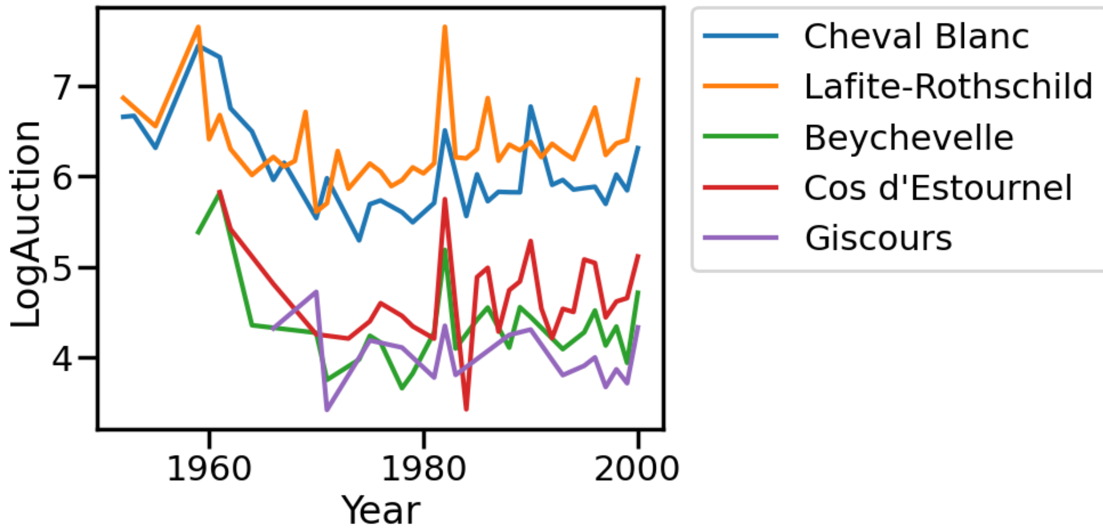
```
[ ]: -0.5558862240672262
```

Next, let us plot the price of the wine versus year, by different wineries

```
[ ]: # Plot the responses for different events and regions
g = sns.lineplot(x="Year", y="LogAuction", hue="Winery", data=wine_new)
```

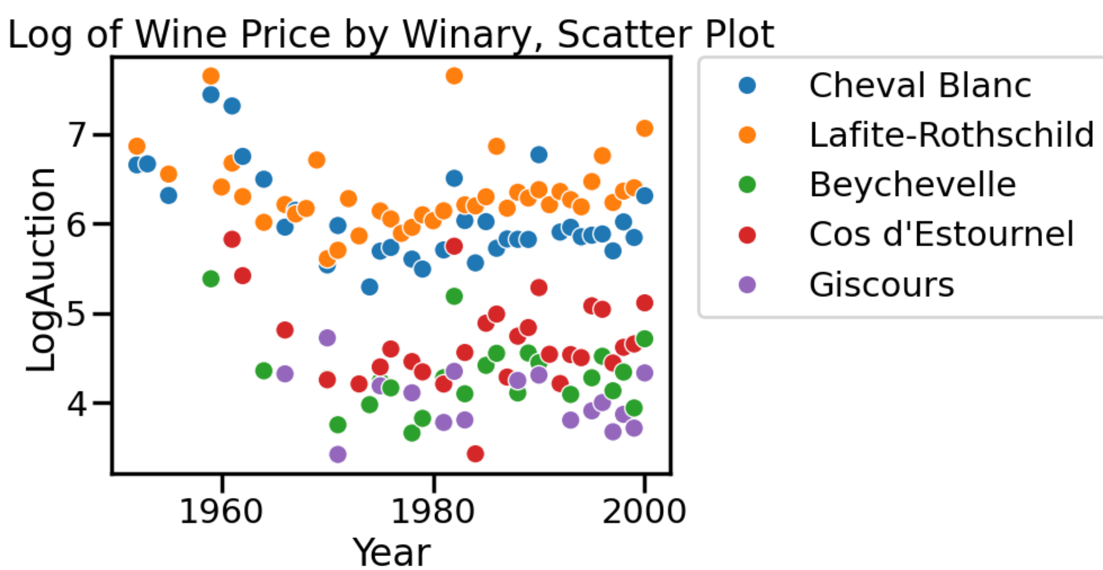
```
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.) # This line is to␣
↪display the lengend out of the graph region.
```

[ ]: <matplotlib.legend.Legend at 0x169787e60>



[ ]:
```
# Plot the responses for different events and regions
g = sns.scatterplot(x="Year", y="LogAuction", hue="Winery", data=wine_new).
↪set_title('Log of Wine Price by Winary, Scatter Plot')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

[ ]: <matplotlib.legend.Legend at 0x15682ab70>



4.1.1 Two Wineries

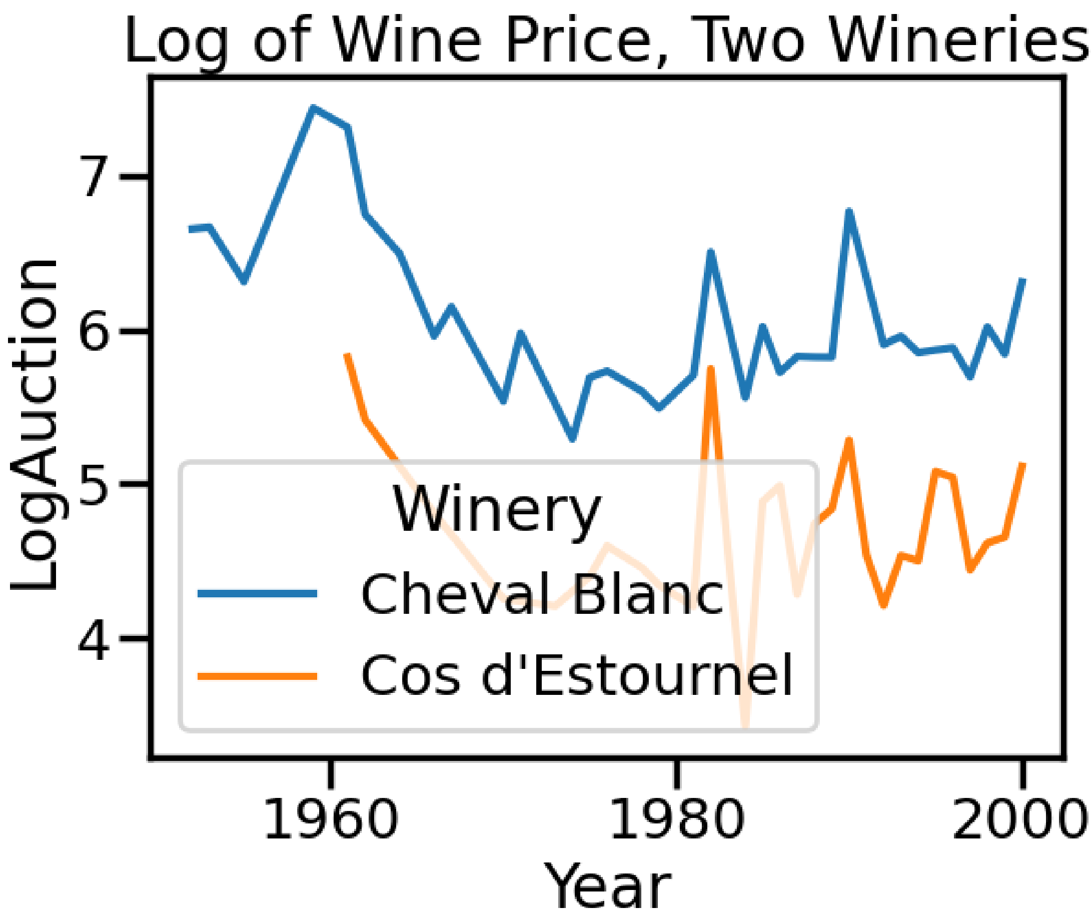
Before constructing a complete model for all the wineries, let's first attempt to regress on only 2 wineries. We pick Cheval Blanc and Cos d'Estournel as an example

[ ]:
```
wine_two = wine_new[(wine_new['Winery'] == 'Cheval Blanc') |␣
↪(wine_new['Winery'] == 'Cos d\'Estournel')]

g = sns.lineplot(x="Year", y="LogAuction", hue="Winery", data=wine_two).
↪set_title('Log of Wine Price, Two Wineries')
#plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

wine_two_train = wine_two[wine_two['Year'] <= 1985]
wine_two_test = wine_two[wine_two['Year'] > 1985]
wine_two_train.tail()
```

[ ]:

	Winery	Year	Age	LogAuction	WinterRain	HarvestRain	GrowTemp
73	Cos d'Estournel	1983	32	4.558498	698.3	119.3	17.87
76	Cheval Blanc	1984	31	5.558641	572.6	144.8	16.71
77	Cos d'Estournel	1984	31	3.426865	572.6	144.8	16.71
80	Cheval Blanc	1985	30	6.018934	667.1	37.2	17.19
81	Cos d'Estournel	1985	30	4.885072	667.1	37.2	17.19

	HarvestTemp	FrancePop	USAlcConsump
73	18.57	54.77	10.46
76	16.24	55.03	10.22
77	16.24	55.03	10.22
80	19.56	55.28	9.88
81	19.56	55.28	9.88



4.1.2 Passing a categorical variable

To use a categorical variables like `Winery`, we can simply pass it to the formula, and `smf.ols` will handle the variable.

```
[ ]: modTwo = smf.ols(formula='LogAuction ~ Winery + WinterRain + HarvestRain +␣
      ↪GrowTemp + Age',
                       data=wine_two_train).fit()
     print(modTwo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             LogAuction   R-squared:                       0.860
Model:                            OLS   Adj. R-squared:                  0.836
Method:                 Least Squares   F-statistic:                     35.55
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           1.62e-11
Time:                        16:17:33   Log-Likelihood:                -13.178
No. Observations:                  35   AIC:                             38.36
Df Residuals:                      29   BIC:                             47.69
```

```
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
=============
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------
Intercept                      -6.6292      2.391     -2.773      0.010
-11.519      -1.739
Winery[T.Cos d'Estournel]      -1.3617      0.139     -9.770      0.000
-1.647      -1.077
WinterRain                      0.0016      0.001      2.363      0.025
0.000       0.003
HarvestRain                    -0.0016      0.002     -1.005      0.323
-0.005       0.002
GrowTemp                        0.6093      0.140      4.359      0.000
0.323       0.895
Age                             0.0358      0.007      4.966      0.000
0.021       0.050
==============================================================================
Omnibus:                        5.334   Durbin-Watson:                   1.852
Prob(Omnibus):                  0.069   Jarque-Bera (JB):                3.865
Skew:                           0.747   Prob(JB):                        0.145
Kurtosis:                       3.648   Cond. No.                     2.21e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.21e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

4.1.3 More Wineries

Now let's expand to the complete set of values that `Winery` can take:

[ ]:
```python
modNew = smf.ols(formula='LogAuction ~ Winery + WinterRain + HarvestRain +␣
 ↪GrowTemp + Age',
                 data=wine_new_train).fit()
print(modNew.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             LogAuction   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.834
Method:                 Least Squares   F-statistic:                     52.68
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           1.73e-27
```

```
Time:                        16:17:33   Log-Likelihood:                -43.659
No. Observations:                  83   AIC:                             105.3
Df Residuals:                      74   BIC:                             127.1
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
===============
                                coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept                    -4.4946      1.576     -2.852      0.006
-7.634      -1.355
Winery[T.Cheval Blanc]        1.6425      0.152     10.819      0.000
1.340       1.945
Winery[T.Cos d'Estournel]     0.2754      0.165      1.669      0.099
-0.053       0.604
Winery[T.Giscours]           -0.2993      0.193     -1.547      0.126
-0.685       0.086
Winery[T.Lafite-Rothschild]   1.8941      0.148     12.788      0.000
1.599       2.189
WinterRain                    0.0016      0.000      3.665      0.000
0.001       0.003
HarvestRain                   0.0004      0.001      0.436      0.664
-0.001       0.002
GrowTemp                      0.3876      0.089      4.374      0.000
0.211       0.564
Age                           0.0308      0.005      5.617      0.000
0.020       0.042
==============================================================================
Omnibus:                        6.522   Durbin-Watson:                   1.725
Prob(Omnibus):                  0.038   Jarque-Bera (JB):                6.227
Skew:                           0.667   Prob(JB):                       0.0444
Kurtosis:                       3.136   Cond. No.                     1.98e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.98e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]:
```
wine_new_train["New_winery"] = wine_new_train["Winery"]
wine_new_train["New_winery"][(wine_new_train["Winery"] == "Giscours") |␣
 ↪(wine_new_train["Winery"] == "Beychevelle")] = "Baseline"
wine_new_train
```

```
/var/folders/nw/5zcrqdxs7c57b12ptv8284p80000gn/T/ipykernel_56530/687993426.py:1:
```

```
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  wine_new_train["New_winery"] = wine_new_train["Winery"]
/var/folders/nw/5zcrqdxs7c57b12ptv8284p80000gn/T/ipykernel_56530/687993426.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  wine_new_train["New_winery"][(wine_new_train["Winery"] == "Giscours") |
(wine_new_train["Winery"] == "Beychevelle")] = "Baseline"
```

[ ]:

```
                Winery  Year  Age  LogAuction  WinterRain  HarvestRain  \
0         Cheval Blanc  1952   63    6.653108       566.4        165.5
1    Lafite-Rothschild  1952   63    6.861502       566.4        165.5
2         Cheval Blanc  1953   62    6.664192       653.3         75.6
3         Cheval Blanc  1955   60    6.311426       504.3        129.5
4    Lafite-Rothschild  1955   60    6.550209       504.3        129.5
..                 …     …    …          …           …            …
78   Lafite-Rothschild  1984   31    6.194936       572.6        144.8
79         Beychevelle  1985   30    4.414736       667.1         37.2
80        Cheval Blanc  1985   30    6.018934       667.1         37.2
81      Cos d'Estournel  1985   30    4.885072       667.1         37.2
82   Lafite-Rothschild  1985   30    6.296612       667.1         37.2

    GrowTemp  HarvestTemp  FrancePop  USAlcConsump         New_winery
0      17.28        14.39      42.46          7.85       Cheval Blanc
1      17.28        14.39      42.46          7.85  Lafite-Rothschild
2      16.94        17.64      42.75          8.03       Cheval Blanc
3      17.30        17.13      43.43          7.84       Cheval Blanc
4      17.30        17.13      43.43          7.84  Lafite-Rothschild
..         …            …          …             …                  …
78     16.71        16.24      55.03         10.22  Lafite-Rothschild
79     17.19        19.56      55.28          9.88           Baseline
80     17.19        19.56      55.28          9.88       Cheval Blanc
81     17.19        19.56      55.28          9.88    Cos d'Estournel
82     17.19        19.56      55.28          9.88  Lafite-Rothschild

[83 rows x 11 columns]
```

[ ]:

```
modNew = smf.ols(formula='LogAuction ~ New_winery + WinterRain + HarvestRain +
 ↪GrowTemp + Age',
                 data=wine_new_train).fit()
```

```python
print(modNew.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:             LogAuction   R-squared:                       0.846
Model:                            OLS   Adj. R-squared:                  0.831
Method:                 Least Squares   F-statistic:                     58.77
Date:                Sun, 27 Oct 2024   Prob (F-statistic):           6.91e-28
Time:                        16:17:33   Log-Likelihood:                -44.980
No. Observations:                  83   AIC:                             106.0
Df Residuals:                      75   BIC:                             125.3
Df Model:                           7                                         
Covariance Type:            nonrobust                                         
================================================================================
===================
                                      coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------------
Intercept                          -4.6261      1.588     -2.913      0.005
-7.789      -1.463
New_winery[T.Cheval Blanc]          1.7507      0.136     12.873      0.000
1.480       2.022
New_winery[T.Cos d'Estournel]       0.3861      0.150      2.573      0.012
0.087       0.685
New_winery[T.Lafite-Rothschild]     2.0027      0.132     15.213      0.000
1.740       2.265
WinterRain                          0.0016      0.000      3.504      0.001
0.001       0.002
HarvestRain                         0.0004      0.001      0.425      0.672
-0.001       0.002
GrowTemp                            0.3904      0.089      4.367      0.000
0.212       0.569
Age                                 0.0310      0.006      5.615      0.000
0.020       0.042
==============================================================================
Omnibus:                        5.596   Durbin-Watson:                   1.697
Prob(Omnibus):                  0.061   Jarque-Bera (JB):                5.153
Skew:                           0.605   Prob(JB):                       0.0760
Kurtosis:                       3.157   Cond. No.                     1.98e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.98e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

4.1.4 Evaluate our final model

[ ]:
```python
# compute out-of-sample R squared
wine_new_test["New_winery"] = wine_new_test["Winery"]
wine_new_test["New_winery"][(wine_new_test["Winery"] == "Giscours") |
 ↪(wine_new_test["Winery"] == "Beychevelle")] = "Baseline"
OSR2(modNew, wine_new_train, wine_new_test, 'LogAuction')
```

```
/var/folders/nw/5zcrqdxs7c57b12ptv8284p80000gn/T/ipykernel_56530/478586887.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  wine_new_test["New_winery"] = wine_new_test["Winery"]
/var/folders/nw/5zcrqdxs7c57b12ptv8284p80000gn/T/ipykernel_56530/478586887.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  wine_new_test["New_winery"][(wine_new_test["Winery"] == "Giscours") |
(wine_new_test["Winery"] == "Beychevelle")] = "Baseline"
```

[ ]: 0.7992742850350536