Introduction:

The goal of this final technical report is to discover whether the LVX-INV fine wine index is a store of value against US inflation and the vagaries of selected US stock market indices. Our findings will determine the direction of the Bordeaux Wine Council's (CIVB) marketing in the US market. Did the LVX-INV fine wine index retain its value during upheavals in US inflation and stock market? Our target variable is the LVX-INV.

Business Understanding:

CIVB's website bordeaux.com has the following information:
"The Conseil Interprofessionnel du Vin de Bordeaux (CIVB or Bordeaux Wine Council), founded in 1948, represents the three families of the Bordeaux wine industry: winegrowers, wine merchants and brokers.
As such, the CIVB has 3 missions:
Marketing: develop the notoriety of Bordeaux wines in France and abroad through advertising campaigns, digital communications, public and press relations, and training.
Economic: acquiring data and improving knowledge relating to the production, the markets and the sale of Bordeaux wines throughout the world.
Technical: improve the industry's understanding of various technical issues relating to the production and quality of Bordeaux wines and anticipate new environment – and health-related requirements."

We will be addressing the marketing mission of CIVB. The Bordeaux Wine Council (CIVB) has a limited budget for marketing to the US market. They have hired our consulting group to determine the possibility of marketing their wine as a possible hedge against the vagaries of US inflation and the S&P 500.

Data Understanding:

We utilized the following data sources from Quandl:

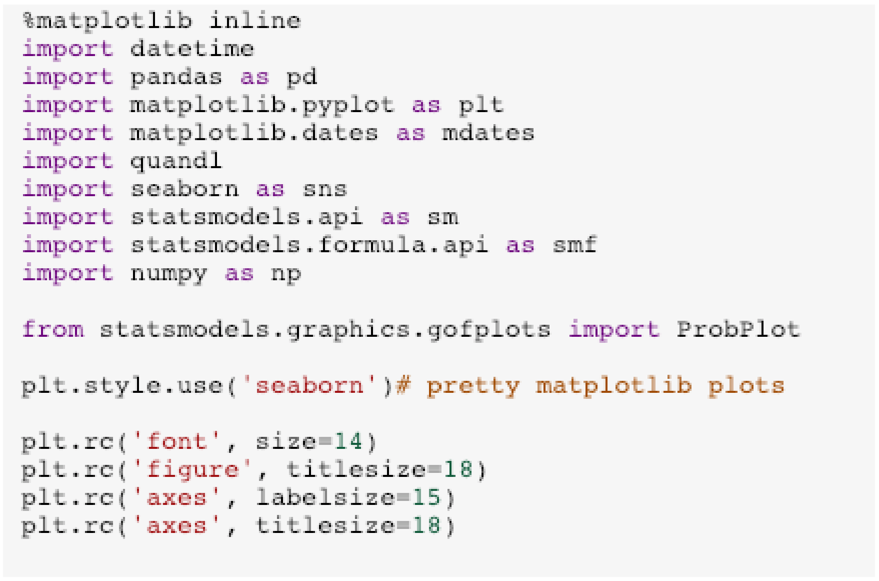
LVX INV Fine Wine Index The Liv-ex Fine Wine Investables Index tracks the most 'investable' wines in the market - around 200 wines from 24 top Bordeaux chateaux. In essence it aims to mirror the performance of a typical wine investment portfolio. The index data starts in 1988, further back than any other Liv-ex index. It has monthly data from 1998 -2017 which is formatted as a date with a value.
Consumer Price Index for All Urban Consumers: All Items This CPI Index tracks the index for all items. This data is provided by the research department of the Federal Reserve Bank of St. Louis. It has monthly data from 1947-2018 which is formatted as a date with a value.

S&P 500 Dividend Yield by Month This S&P 500 database is a ratio of the dividend yield (12 month dividend per share)/price. The source of this data is Standard & Poor's for current S&P 500 Dividend Yield. It has monthly data from 1871-2018 which is formatted as a date with a value.

These three data sets will give us the ability to measure the performance of the LVX vs the US CPI and S&P 500.

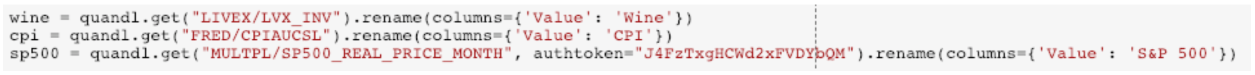
<u>Data Exploration/Preparation:</u>

All data exploration and modeling was conducted in the Python language utilizing the following libraries/modules:
Datetime - Python module for manipulating dates and times
Matplotlib - Python 2D plotting library
Seaborn - Python data visualization library based on matplotlib.
Statsmodels - Python module for conducting statistical tests, and statistical data exploration
NumPy - Part of Python SciPy library for scientific computing
Pandas - Python library for data analysis



```
%matplotlib inline
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import quandl
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
import numpy as np

from statsmodels.graphics.gofplots import ProbPlot

plt.style.use('seaborn')# pretty matplotlib plots

plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
```

The three datasets were imported from Quandl and the Value columns were labeled respectively Wine, CPI, and S&P 500. The monthly dates were already included in the datasets.



```
wine = quandl.get("LIVEX/LVX_INV").rename(columns={'Value': 'Wine'})
cpi = quandl.get("FRED/CPIAUCSL").rename(columns={'Value': 'CPI'})
sp500 = quandl.get("MULTPL/SP500_REAL_PRICE_MONTH", authtoken="J4FzTxgHCWd2xFVDYbQM").rename(columns={'Value': 'S&P 500'})
```

```
cpi.head()
```

Date	CPI
1947-01-01	21.48
1947-02-01	21.62
1947-03-01	22.00
1947-04-01	22.00
1947-05-01	21.95

```
wine.tail()
```

Date	Wine
2017-10-31	337.91
2017-11-30	341.13
2017-12-31	341.28
2018-01-31	341.57
2018-02-28	341.22

```
sp500.tail()
```

Date	S&P 500
2018-03-01	2702.77
2018-04-01	2653.63
2018-05-01	2701.49
2018-06-01	2754.35
2018-07-01	2736.61

The first observation on our data set was that the dates were not aligned. The data either begins at the first calendar day (CPI, S&P 500) or the last calendar day (Wine) of the month.

```
print("How many different days of the month?")
print("CPI: ", cpi.index.day.unique())
print("Wine ", wine.index.day.unique())
print("S&P 500 ", sp500.index.day.unique())
```

```
How many different days of the month?
CPI:  Int64Index([1], dtype='int64', name='Date')
Wine  Int64Index([1, 31, 30, 29, 28], dtype='int64', name='Date')
S&P 500  Int64Index([1], dtype='int64', name='Date')
```

The function “nearest_month” was constructed to align the indices. Pandas offsets.Day was used to align the indices by changing the end-of-month date to the first of the following month.

```
# Need to align the indices
def nearest_month(dt):
    #either first or last day of month
    return dt if dt.is_month_start else dt + pd.offsets.Day(1)
```

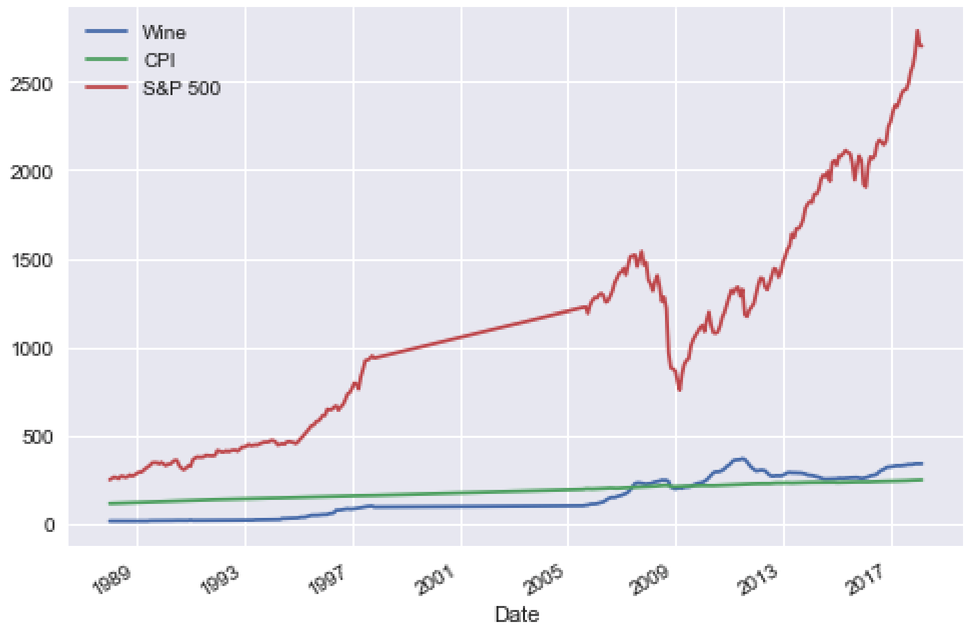
A copy of each of the datasets was constructed (cpimod, winemod, sp500mod). Each new dataset had the .index inbuilt function with the nearest_month function applied to modify each dataset to be reset to the nearest month. The Wine dataset was successfully modified to start at the first of the month in alignment with CPI and S&P 500.

```
cpimod = cpi.copy()
winemod = wine.copy()
sp500mod = sp500.copy()
cpimod.index = cpi.index.map(nearest_month)
winemod.index = wine.index.map(nearest_month)
sp500mod.index = sp500.index.map(nearest_month)
# Check is wine is modified
winemod.tail()
```

	Wine
Date	
2017-11-01	337.91
2017-12-01	341.13
2018-01-01	341.28
2018-02-01	341.57
2018-03-01	341.22

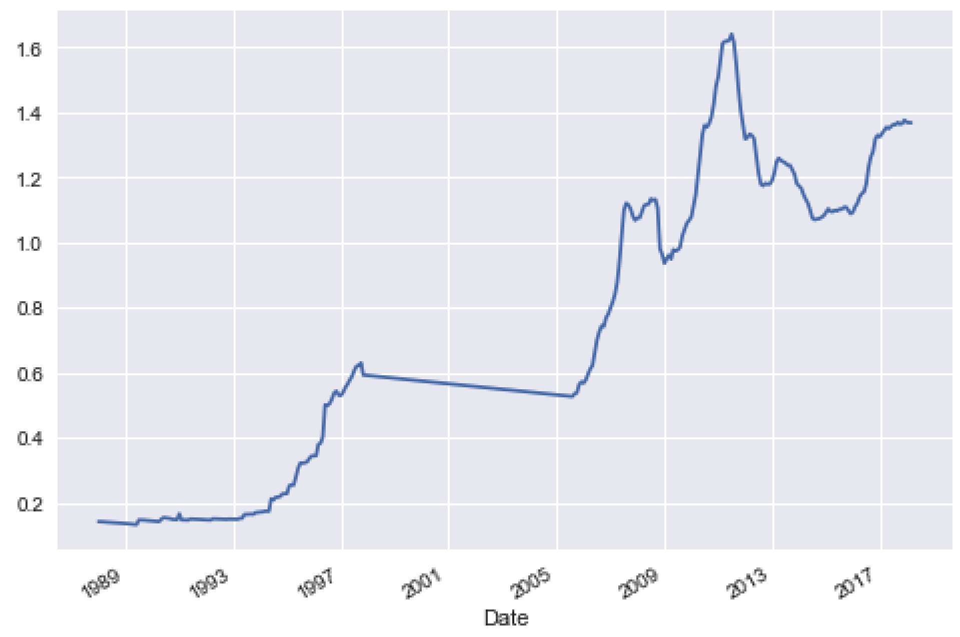
Our data exploration continued with merging all three datasets into one chart by using a dataframe(df):

```
df = winemod.join(cpimod).join(sp500mod)
df.plot();
```



We noticed that the chart was flat between 1998-2005. We charted a Wine Over CPI ratio to clarify the scale because the previous chart Wine and CPI were so flat in comparison to S&P 500.

```
df["Wine_Over_CPI"]= df.Wine/df.CPI
df.Wine_Over_CPI.plot();
```



But the same flatness occurs again in this chart. This is because there is missing data in the Wine database 1998-2005. All further analysis will have to exclude that date range.

Model building and Model evaluation

Building of various models and evaluating them was challenging. Basic charting showed some initial interesting trends.

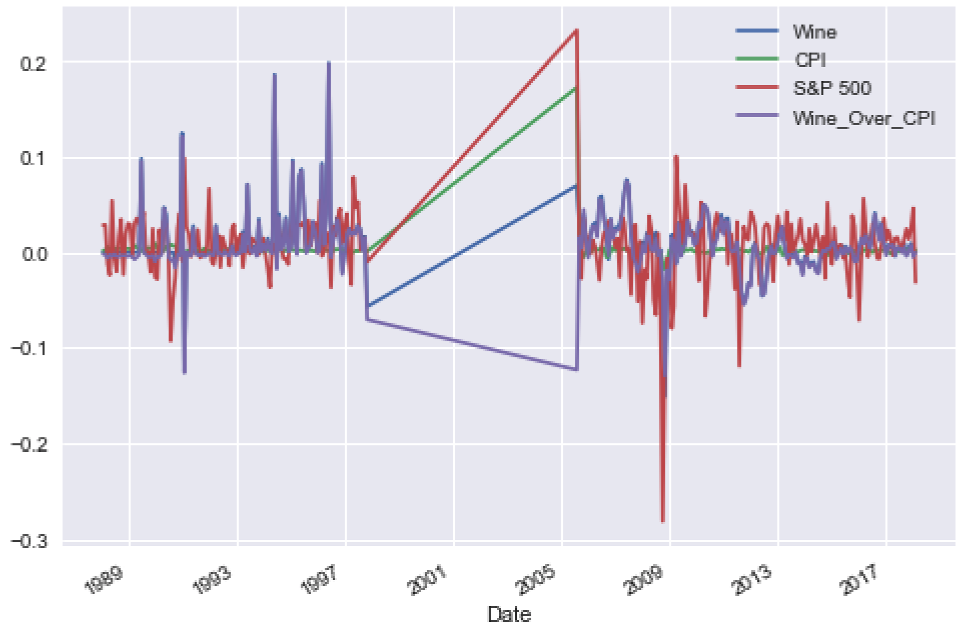
The contents of our dataframes:

```
df.head()
```

	Wine	CPI	S&P 500	Wine_Over_CPI
Date				
1988-01-01	16.54	116.0	250.5	0.142586
1988-02-01	16.54	116.2	258.1	0.142341
1988-03-01	16.54	116.5	265.7	0.141974
1988-04-01	16.54	117.2	262.6	0.141126
1988-05-01	16.54	117.5	256.1	0.140766

A comparison of all four dataframes was plotted. Notice that Wine strengthens over CPI after 2009. (note the missing data between 1998-2005)

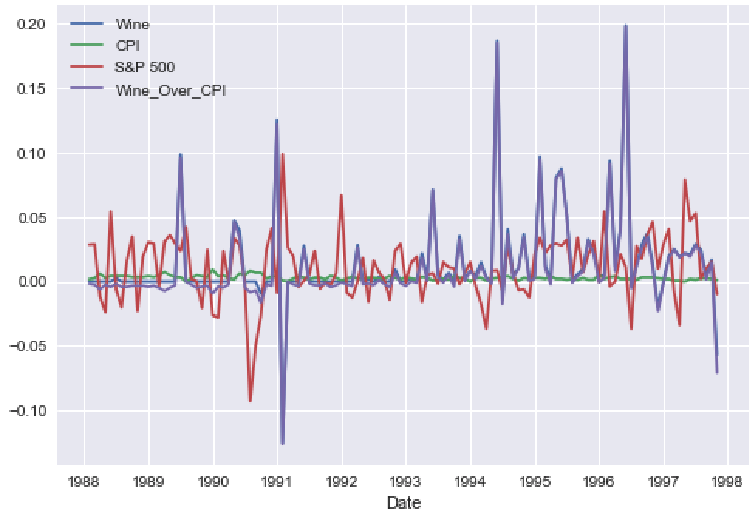
```
# Return = (value(now)-value(prev)) / value(prev)
df_returns = (df.diff()/df.shift(-1)).dropna()
df_returns.plot();
```



The same dataframes were charted before 1998:

```
df_returns[:'1998'].plot()
```

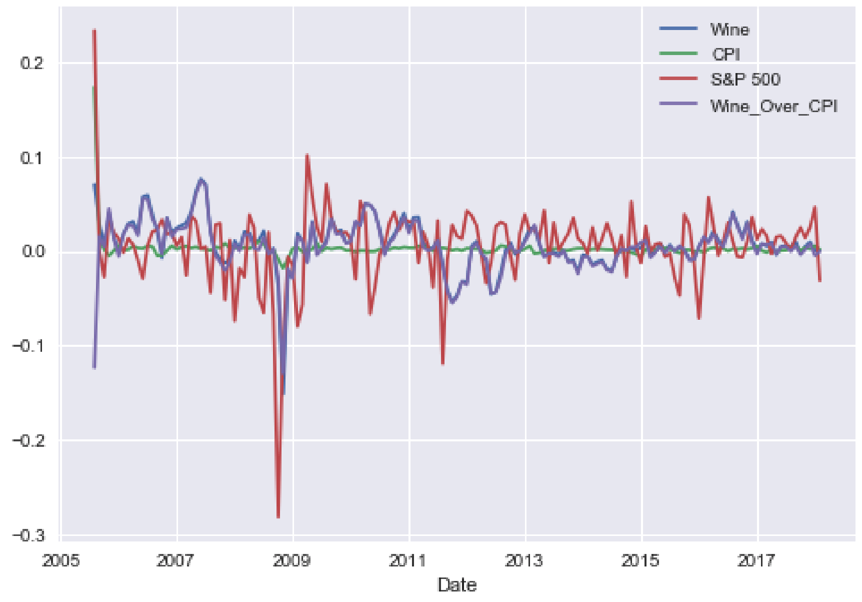
```
<matplotlib.axes._subplots.AxesSubplot at 0x21c5a32aa58>
```



The same dataframes were charted after 2005:

```
df_returns['2005':].plot()
```

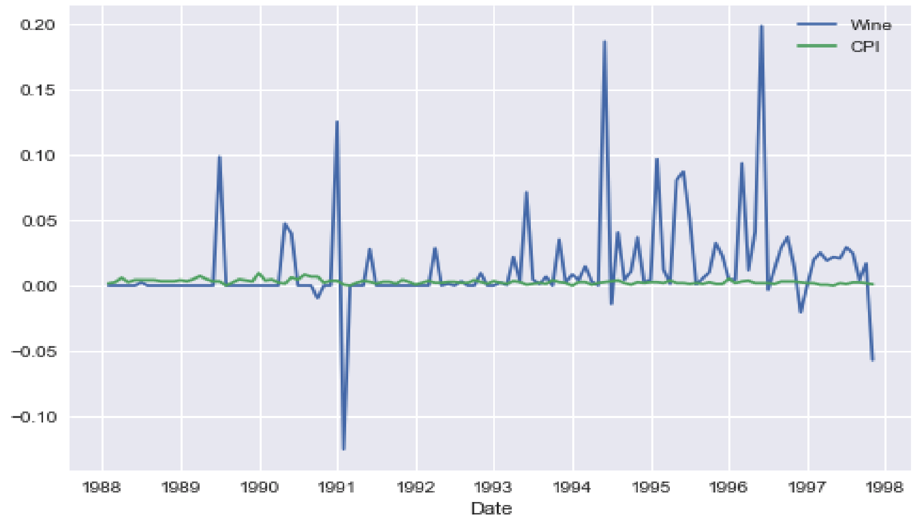
```
<matplotlib.axes._subplots.AxesSubplot at 0x21c5a288ac8>
```



Only Wine and CPI dataframes were plotted 1988-1998:

```
df_returns[['Wine','CPI']][:'1998'].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x21c5a418e48>
```



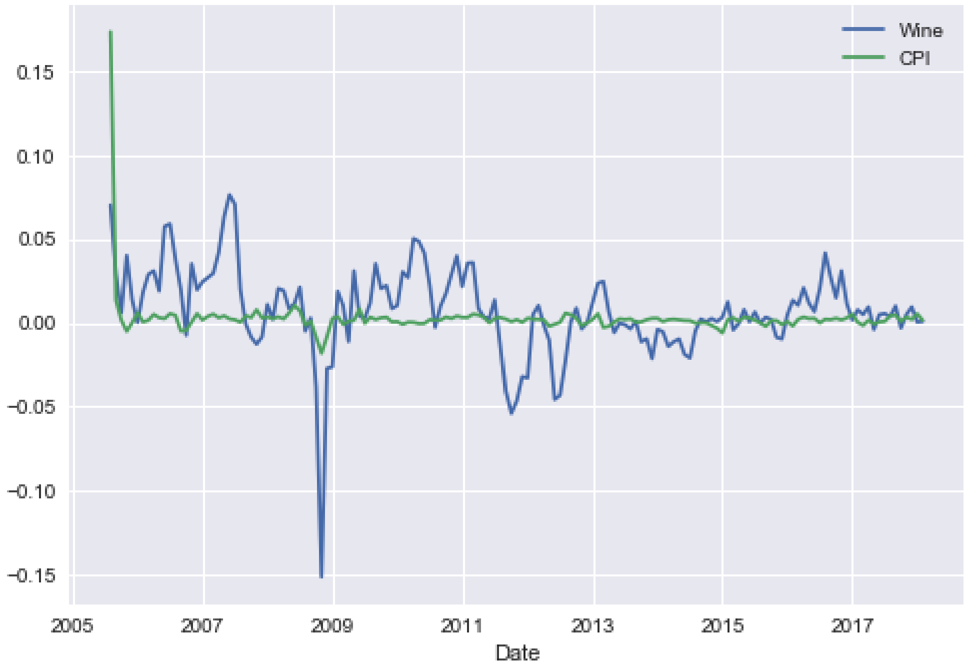
<u>Interesting points are early 1991 and late 1997 on this chart.</u>
According to the winecellarinsider.com, "1991 Bordeaux wine is atrocious. The vintage, the product of the second coldest vintage in recorded history was a complete disaster in Pomerol and St. Emilion

and not much better in the Left Bank or Pessac Leognan. Numerous estates declassified their 1991 Bordeaux wine. How did this happen? Below freezing temperatures struck the vineyards April 21." In another online article by winecellarinsider.com, "1997 Bordeaux wine was another in a series of moderate to poor vintages ending in the dreaded number 7. 1957, 1967, 1977, 1987 and 1997. That pattern continued with 2007."

Only Wine and CPI dataframes were plotted 2005-2018:

```
df_returns[['Wine','CPI']]['2005':].plot()
```

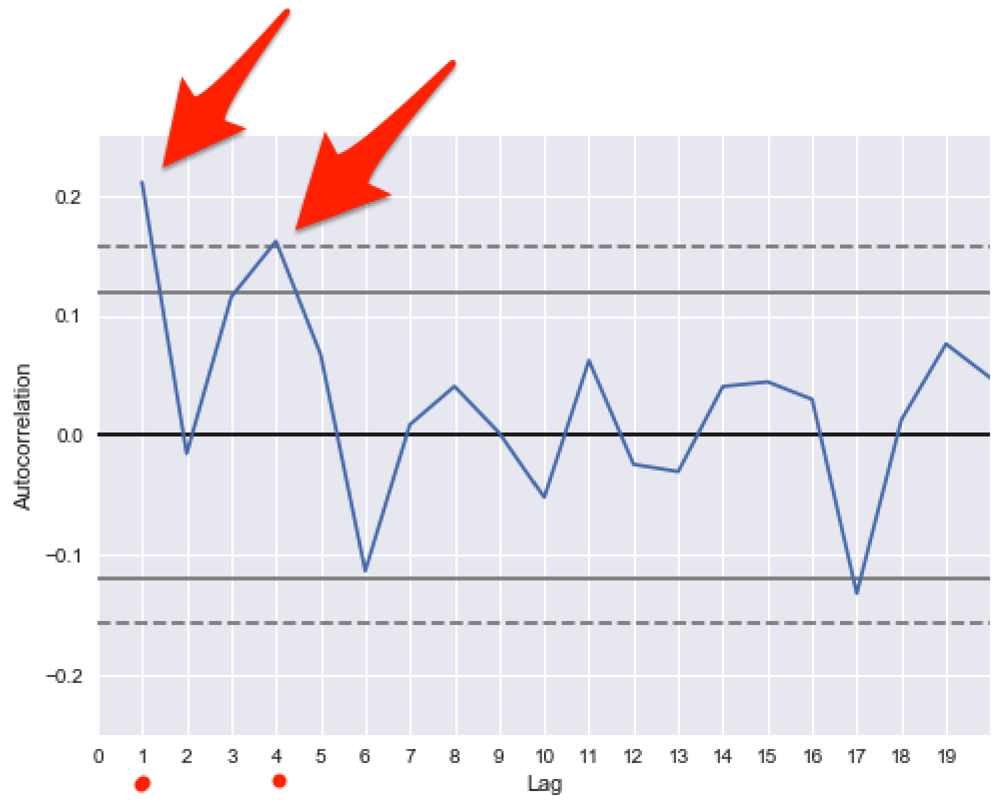
```
<matplotlib.axes._subplots.AxesSubplot at 0x21c5a32af28>
```



<u>Interesting points on this chart are late 2008 and 2012.</u>
The winecellarinsider.com offers an explanation for 2008's dramatically different index, "2008 Bordeaux wine enjoys the fact that is was the last of the well priced Bordeaux wine vintages. Due to the world-wide economic crisis, and because Robert Parker was not going to review or rate 2008 Bordeaux wine until later that year, the wines were first offered at very low prices. The wines languished. There was very little demand until Robert Parker published his notes later that year. Keep in mind, the First Growths were being offered at $200 per bottle."
According to vinous.com, 2012 was a poor year for bordeaux wines, "Simply put, the lousy weather of 2012 did not allow for great wines."

The next step in model building and evaluation was investigating if there were any time series dependencies between the datasets.
The first step was developing an autocorrelation plot to determine how many lags to include in the regression of the S&P 500 using the Pandas library.

```
# PANDAS
ax = pd.plotting.autocorrelation_plot(
    df_returns['S&P 500'],
    #grid=True,
    #xlim=(0,20),
    #xticks=range(20),
)
ax.set_xlim(0,20)# set parameters lag 1
ax.set_ylim(-.25,.25)
ax.set_xticks(range(20))
ax.set_xticklabels(range(20))
plt.grid(True);
```



Lag 1 and 4 are the only points outside the 99% confidence level. This justifies using them for the regression analysis to address time dependency.

The next step is to make a dataframe that has the lags (1 & 4) and construct a regression model. This model will show CPI and LXV regressed against S&P 500.

As noted before, dates for CPI are 1/1/2018 and dates for WINE are 1/31/2018. For this model we shifted CPI (from 1/1/2018 to 2/1/2018) by a month so they are only one day apart.

CPI:

```
cpimod = cpi.reset_index()
cpimod.columns = "Date","CPI"
cpimod["Year"]= cpimod.Date.dt.year
cpimod["Month"]= cpimod.Date.dt.month
cpimod["Day"] = 1
cpimod["Rounded_Date"] = pd.to_datetime(cpimod[["Year", "Month", "Day"]])
cpimod["Joined_Column"] = 100*cpimod.Year + cpimod.Month
cpimod.head()
```

```
         Date    CPI  Year  Month  Day  Rounded_Date  Joined_Column
0  1947-01-01  21.48  1947      1    1    1947-01-01         194701
1  1947-02-01  21.62  1947      2    1    1947-02-01         194702
2  1947-03-01  22.00  1947      3    1    1947-03-01         194703
3  1947-04-01  22.00  1947      4    1    1947-04-01         194704
4  1947-05-01  21.95  1947      5    1    1947-05-01         194705
```

S&P 500:

```
sp500mod = sp500.reset_index()
sp500mod.columns = "Date","S&P 500"
sp500mod["Year"]= sp500mod.Date.dt.year
sp500mod["Month"]= sp500mod.Date.dt.month
sp500mod["Day"] = 1
sp500mod["Rounded_Date"] = pd.to_datetime(sp500mod[["Year", "Month", "Day"]])
sp500mod["Joined_Column"] = 100*sp500mod.Year + sp500mod.Month
sp500mod.head()
```

```
         Date  S&P 500  Year  Month  Day  Rounded_Date  Joined_Column
0  1871-01-01     4.44  1871      1    1    1871-01-01         187101
1  1871-02-01     4.50  1871      2    1    1871-02-01         187102
2  1871-03-01     4.61  1871      3    1    1871-03-01         187103
3  1871-04-01     4.74  1871      4    1    1871-04-01         187104
4  1871-05-01     4.86  1871      5    1    1871-05-01         187105
```

Wine:

```
winemod = wine.reset_index()
winemod.columns = "Date", "LVX"
winemod["Year"]= winemod.Date.dt.year
winemod["Month"]= winemod.Date.dt.month
winemod["Day"] = 1
winemod["Rounded_Date"] =pd.to_datetime(winemod[["Year", "Month", "Day"]])
winemod["Joined_Column"] = 100*winemod.Year + winemod.Month
#winemod["Shift_Date"]= winemod.Date datetime.timedelta(days = 1)
winemod.head()
```

	Date	LVX	Year	Month	Day	Rounded_Date	Joined_Column
0	1988-01-01	16.54	1988	1	1	1988-01-01	198801
1	1988-02-01	16.54	1988	2	1	1988-02-01	198802
2	1988-03-01	16.54	1988	3	1	1988-03-01	198803
3	1988-04-01	16.54	1988	4	1	1988-04-01	198804
4	1988-05-01	16.54	1988	5	1	1988-05-01	198805

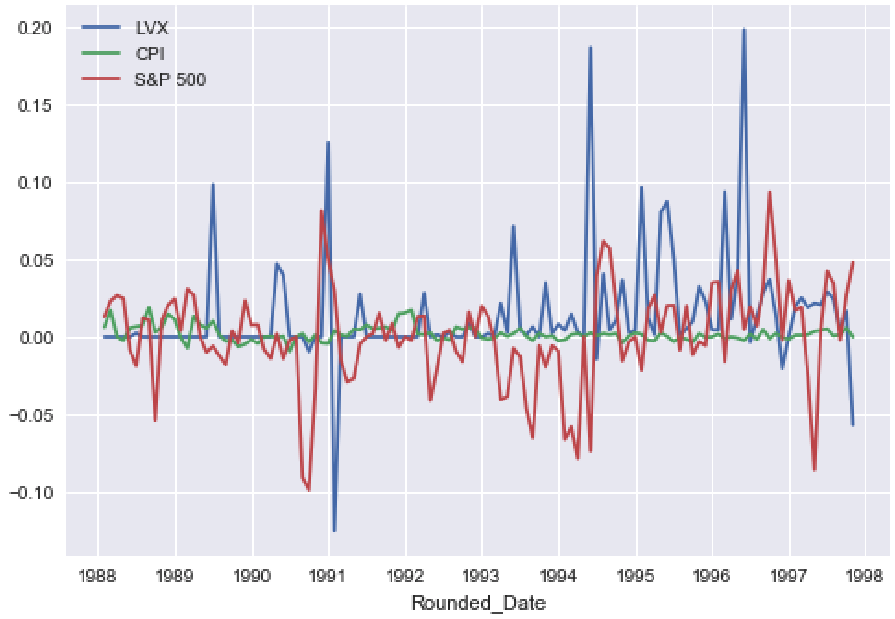
Then we joined all the modified dataframes into one by the date:

```
#df = winemod[["Rounded_Date", "Joined_Column","LVX"]].join(cpimod[["Joined_Column", "CPI"]],on=["Joined_Column"])
df = pd.concat([winemod[["Rounded_Date", "Joined_Column","LVX"]],
                cpimod[["Joined_Column", "CPI"]],
                sp500mod[["Joined_Column", "S&P 500"]]],
              axis=1, join='inner', ignore_index=True)
df.columns = ["Rounded_Date", "Joined_Column_1","LVX",
              "Joined_Column_2", "CPI",
             "Joined_Column_3", "S&P 500"]
df.drop(labels=["Joined_Column_1", "Joined_Column_2", "Joined_Column_3"], inplace=True, axis=1)
df.set_index("Rounded_Date", inplace=True)
df.head()
df.tail()
```

	LVX	CPI	S&P 500
Rounded_Date			
2017-10-01	337.91	36.1	5.31
2017-11-01	341.13	36.3	5.31
2017-12-01	341.28	36.4	4.84
2018-01-01	341.57	36.6	4.61
2018-02-01	341.22	36.8	4.18

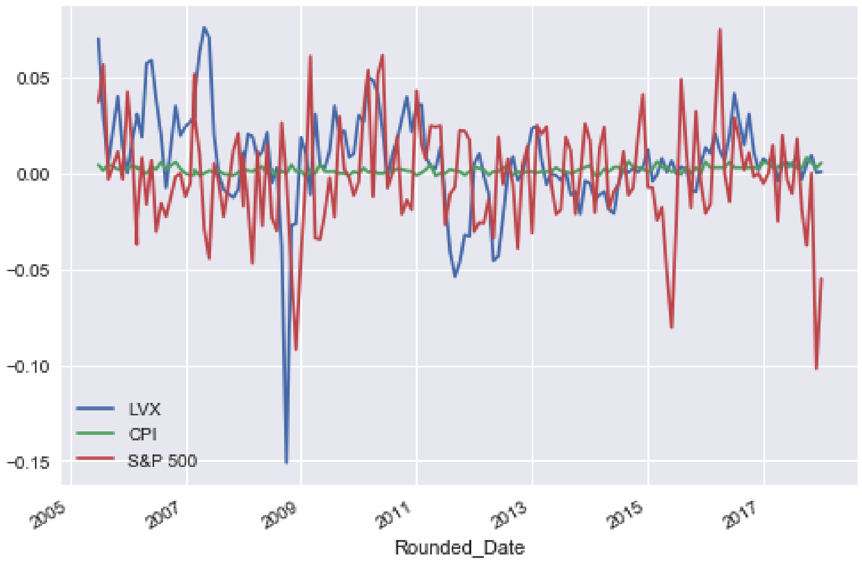
The dataframe of all three indexes from 1988-1998:

```
df_newreturns = (df.diff()/df.shift(-1)).dropna()
df_newreturns[:'1998'].plot();
```



The dataframe of all three indexes from 2005-2017:

```
df_newreturns['2005':].plot();
```



Now we are at a point to use the lag 1 & 4 we identified in the auto-correlation chart. We will make a regression of the S&P 500, CPI, LVX against lags 1 & 4 of S&P 500.

```
df = df.assign(SP500_lag1=df.SP500.shift(1))
df = df.assign(SP500_lag4=df.SP500.shift(4))
df.head()
```

	LVX	CPI	SP500	SP500_lag1	SP500_lag4
Rounded_Date					
1988-01-01	16.54	21.48	4.44	NaN	NaN
1988-02-01	16.54	21.62	4.50	4.44	NaN
1988-03-01	16.54	22.00	4.61	4.50	NaN
1988-04-01	16.54	22.00	4.74	4.61	NaN
1988-05-01	16.54	21.95	4.86	4.74	4.44

Initial regression analysis results on CPI versus S&P 500 lag1, lag4 and intercept.

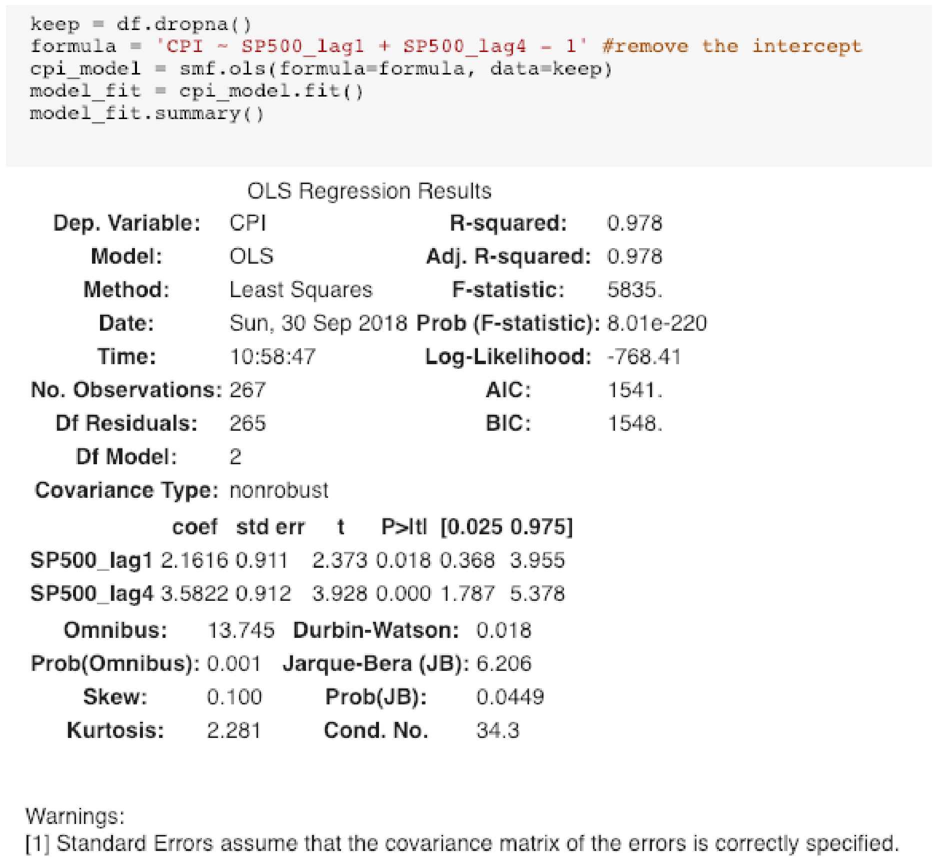
```
keep = df.dropna()
formula = 'CPI ~ SP500_lag1 + SP500_lag4'
cpi_model = smf.ols(formula=formula, data=keep).fit()
cpi_model.summary()
```

```
                        OLS Regression Results
   Dep. Variable:    CPI                  R-squared:         0.174
          Model:     OLS              Adj. R-squared:        0.168
         Method:     Least Squares        F-statistic:       27.85
           Date:     Sun, 30 Sep 2018 Prob (F-statistic):    1.06e-11
           Time:     10:58:39         Log-Likelihood:        -674.62
No. Observations:    267                        AIC:         1355.
    Df Residuals:    264                        BIC:         1366.
       Df Model:     2
Covariance Type:     nonrobust

                coef     std err   t       P>|t|  [0.025  0.975]
   Intercept  19.8618  1.211     16.400  0.000  17.477  22.246
SP500_lag1    0.1420   0.654     0.217   0.828  -1.146  1.430
SP500_lag4    1.6532   0.654     2.529   0.012  0.366   2.940

       Omnibus:     4.707   Durbin-Watson:      0.007
Prob(Omnibus):     0.095   Jarque-Bera (JB):   2.984
          Skew:     0.012   Prob(JB):           0.225
      Kurtosis:     2.483   Cond. No.           46.6


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

Interpretation: R-squared isn't close to 1, actual is .174

We then ran the model without the intercept.

CPI:



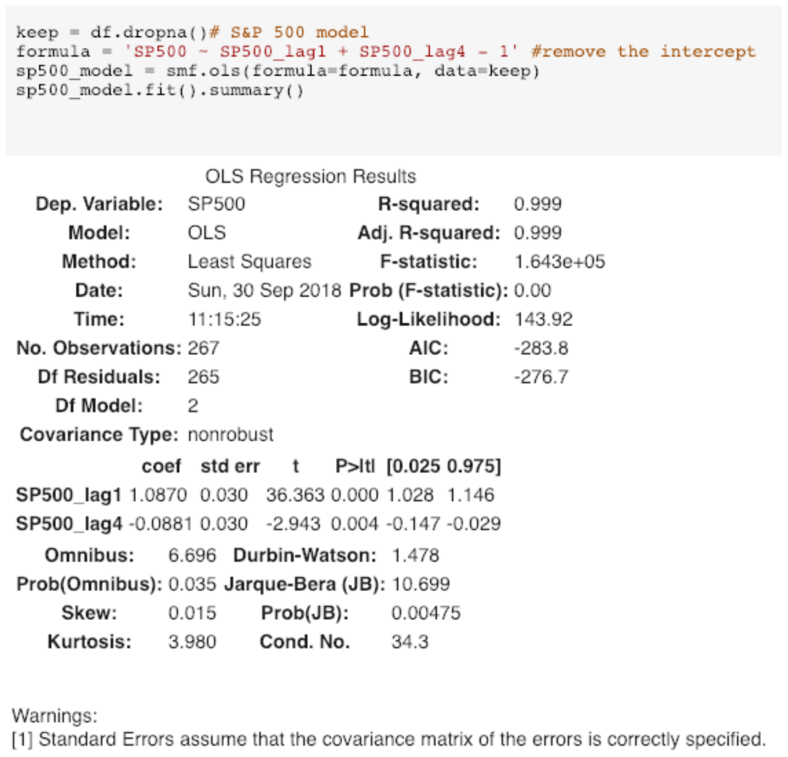
```
keep = df.dropna()
formula = 'CPI ~ SP500_lag1 + SP500_lag4 - 1' #remove the intercept
cpi_model = smf.ols(formula=formula, data=keep)
model_fit = cpi_model.fit()
model_fit.summary()
```

```
                         OLS Regression Results
   Dep. Variable:    CPI                   R-squared:         0.978
         Model:      OLS                   Adj. R-squared:    0.978
        Method:      Least Squares         F-statistic:       5835.
          Date:      Sun, 30 Sep 2018  Prob (F-statistic):  8.01e-220
          Time:      10:58:47              Log-Likelihood:    -768.41
No. Observations:    267                        AIC:          1541.
    Df Residuals:    265                        BIC:          1548.
        Df Model:    2
 Covariance Type:    nonrobust

                  coef   std err    t     P>|t|  [0.025 0.975]
SP500_lag1 2.1616 0.911    2.373 0.018 0.368   3.955
SP500_lag4 3.5822 0.912    3.928 0.000 1.787   5.378

     Omnibus:       13.745  Durbin-Watson:    0.018
Prob(Omnibus):  0.001   Jarque-Bera (JB): 6.206
        Skew:       0.100        Prob(JB):        0.0449
     Kurtosis:      2.281        Cond. No.        34.3


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

This model had a R-squared closer to 1 and a P value is less than .05 (p = 0.018). This means it is likely that both coefficients are relevant in the model, both positive.

We proceeded to run the same model on S&P 500 and the Wine indexes.

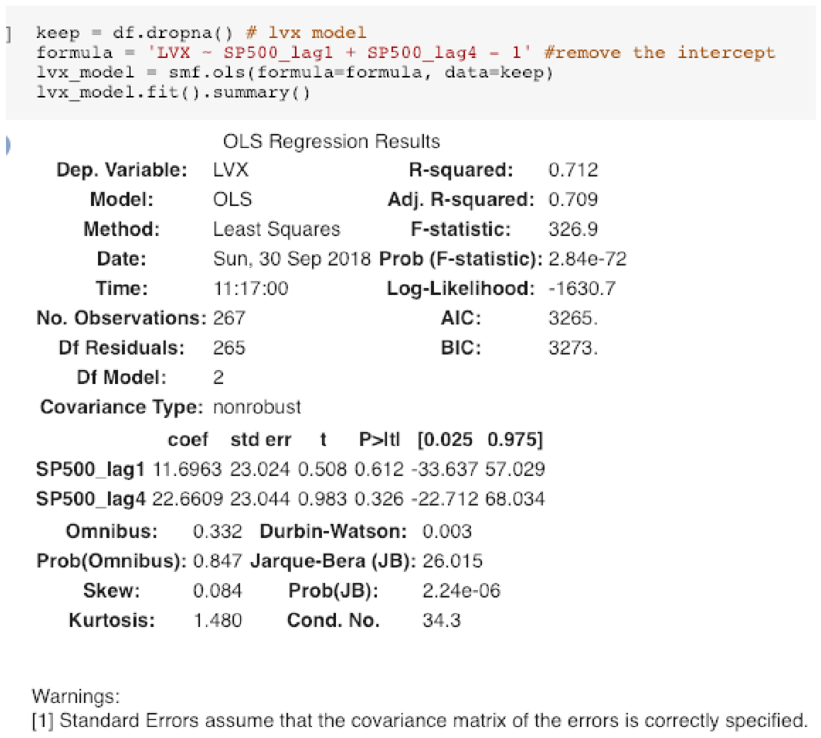
S&P 500:



```
keep = df.dropna()# S&P 500 model
formula = 'SP500 ~ SP500_lag1 + SP500_lag4 - 1' #remove the intercept
sp500_model = smf.ols(formula=formula, data=keep)
sp500_model.fit().summary()
```

```
                    OLS Regression Results
  Dep. Variable:    SP500               R-squared:          0.999
         Model:     OLS              Adj. R-squared:        0.999
        Method:     Least Squares       F-statistic:        1.643e+05
          Date:     Sun, 30 Sep 2018 Prob (F-statistic):    0.00
          Time:     11:15:25         Log-Likelihood:        143.92
No. Observations:   267                      AIC:           -283.8
   Df Residuals:    265                      BIC:           -276.7
      Df Model:     2
Covariance Type:    nonrobust

              coef    std err    t      P>|t|  [0.025  0.975]
SP500_lag1  1.0870   0.030    36.363  0.000  1.028   1.146
SP500_lag4  -0.0881  0.030    -2.943  0.004  -0.147  -0.029

    Omnibus:      6.696  Durbin-Watson:     1.478
Prob(Omnibus):    0.035  Jarque-Bera (JB):  10.699
      Skew:       0.015     Prob(JB):       0.00475
    Kurtosis:     3.980     Cond. No.       34.3


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

Wine:



```
keep = df.dropna() # lvx model
formula = 'LVX ~ SP500_lag1 + SP500_lag4 - 1' #remove the intercept
lvx_model = smf.ols(formula=formula, data=keep)
lvx_model.fit().summary()
```

```
                    OLS Regression Results
  Dep. Variable:    LVX                 R-squared:          0.712
         Model:     OLS              Adj. R-squared:        0.709
        Method:     Least Squares       F-statistic:        326.9
          Date:     Sun, 30 Sep 2018 Prob (F-statistic):    2.84e-72
          Time:     11:17:00         Log-Likelihood:        -1630.7
No. Observations:   267                      AIC:           3265.
   Df Residuals:    265                      BIC:           3273.
      Df Model:     2
Covariance Type:    nonrobust

              coef     std err   t      P>|t|  [0.025   0.975]
SP500_lag1  11.6963  23.024  0.508  0.612  -33.637  57.029
SP500_lag4  22.6609  23.044  0.983  0.326  -22.712  68.034

    Omnibus:      0.332  Durbin-Watson:     0.003
Prob(Omnibus):    0.847  Jarque-Bera (JB):  26.015
      Skew:       0.084     Prob(JB):       2.24e-06
    Kurtosis:     1.480     Cond. No.       34.3


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

The following is a table summarizing the results of all three regressions together for comparison:

Linear Regression Table (OLS) of the CPI, S&P 500, LVX

	Coefficient:		
Dependent Variable	**CPI**	**S&P 500**	**LVX**
S&P 500 lag 1	2.1616 (0.911)**	1.0870 (0.030)***	11.6963 (23.024)
S&P 500 lag 4	3.5822 (0.912)***	-0.0881 (0.030)**	22.6609 (23.044)
Number	267	267	267
Adj. R-squared	0.978	0.999	0.709
Note: Standard Error in parenthesis P-values: no star denotes no statistical significance, * = p < 0.1, **= p<0.05, ***=p<0.001			

R-squared is the coefficient of determination. A statistical measure of how well the regression line approximates the real data points.
Adj. R-squared: adjusted based on the number of observations and the degrees-of-freedom of the residuals.
From the Adjusted R-squared numbers the variation inside the CPI and S&P 500 dependent variables is mostly explained by the S&P 500 lag variables that we selected. This is not as true for the wine index which has an Adjusted R-squared that is 0.709.
The standard errors for the wine index are large. The LVX P-values are S&P 500 lag 1 (p = 0.612) and S&P 500 lag 4 (p = 0.326). Large P-values indicate that there is not statistically significant relationship between the term and the response. However, the lagged S&P 500 variables have a statistically significant relationship for both the CPI and S&P 500.

- CPI: P-values S&P 500 lag 1 (p = 0.018), S&P 500 lag 4 (p = 0.000)
- S&P 500: P-values S&P 500 lag 1 (p = 0.000), S&P 500 lag 4 (p = 0.004)
- LVX: P-values S&P 500 lag 1 (p = 0.612), S&P 500 lag 4 (p = 0.326)

LVX has large coefficients, and P-values, Confidence Intervals that contain 0 and large standard error numbers relative to the values.
This indicates that LVX is not correlated with S&P 500 and CPI and thus a potential hedge for investors.

Conclusion:

Outside of the vagaries of the weather, the wine index seems to be historically stronger versus the CPI which means fine wines could potentially be marketed as a hedge against inflation.
Historically, wine has been an excellent investment during the following time periods:
1993-1996, 2009-2011, and 2015-2017.

References:

Quandl 3.4.5
Pandas 0.22.0
NumPy 1.14.6
Matplotlib 2.1.2
Statsmodels 0.8.0
Seaborn 0.7.1
Python 3.6.7

Conseil interprofessionnel du vin de bordeaux (CIVB), https://www.bordeaux.com/us/Contact

Leve, Jeff. “1991 Bordeaux Wine Vintage Report and Buying Guide.” The Wine Cellar Insider, 2010, www.thewinecellarinsider.com/wine-topics/bordeaux-wine-buying-guide-tasting-notes-ratings/bordeaux-wine-detailed-vintage-summary-1945-today/1991-bordeaux-wine-vintage-report-buying-tips/.

Leve, Jeff. “1997 Bordeaux Wine Vintage Report and Buying Guide.” The Wine Cellar Insider, 2010, www.thewinecellarinsider.com/wine-topics/bordeaux-wine-buying-guide-tasting-notes-ratings/bordeaux-wine-detailed-vintage-summary-1945-today/1997-bordeaux-wine-vintage-report-buying-tips/.

Leve, Jeff. “2008 Bordeaux Wine Vintage Report and Buying Guide.” The Wine Cellar Insider, 2010, www.thewinecellarinsider.com/wine-topics/bordeaux-wine-buying-guide-tasting-notes-ratings/bordeaux-wine-detailed-vintage-summary-1945-today/2008-bordeaux-wine-vintage-report-buying-tips/.

“Bordeaux 2012: The Good, the Bad and the Ugly.” Edited by Antonio Galloni, The Wine Cellar Insider, Vinous Media LLC, May 2013, www.vinous.com/articles/bordeaux-2012-the-good-the-bad-and-the-ugly-may-2013.