HW 2

September 26, 2022

1 Matthias Rathbun

1.1 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.tools.tools import maybe_unwrap_results
from statsmodels.graphics.gofplots import ProbPlot
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels
from typing import Type
import math
from sklearn.model_selection import train_test_split
import plotly.express as px
from sklearn.metrics import mean_squared_error
```

1.2 Diagnostic Class

[2]:
```
class Linear_Reg_Diagnostic():
    """
    Diagnostic plots to identify potential problems in a linear regression fit.
    Mainly,
        a. non-linearity of data
        b. Correlation of error terms
        c. non-constant variance
        d. outliers
        e. high-leverage points
        f. collinearity

"""

    def __init__(self,
                 results: Type[statsmodels.regression.linear_model.
 ↪RegressionResultsWrapper]) -> None:
```

```
        """
        For a linear regression model, generates following diagnostic plots:

        a. residual
        b. qq
        c. scale location and
        d. leverage

        and a table

        e. vif

        Args:
            results (Type[statsmodels.regression.linear_model.
↪RegressionResultsWrapper]):
                must be instance of statsmodels.regression.linear_model object

        Raises:
            TypeError: if instance does not belong to above object

        Example:
        >>> import numpy as np
        >>> import pandas as pd
        >>> import statsmodels.formula.api as smf
        >>> x = np.linspace(-np.pi, np.pi, 100)
        >>> y = 3*x + 8 + np.random.normal(0,1, 100)
        >>> df = pd.DataFrame({'x':x, 'y':y})
        >>> res = smf.ols(formula= "y ~ x", data=df).fit()
        >>> cls = Linear_Reg_Diagnostic(res)
        >>> cls(plot_context="seaborn-paper")

        In case you do not need all plots you can also independently make an␣
↪individual plot/table
        in following ways

        >>> cls = Linear_Reg_Diagnostic(res)
        >>> cls.residual_plot()
        >>> cls.qq_plot()
        >>> cls.scale_location_plot()
        >>> cls.leverage_plot()
        >>> cls.vif_table()
        """

        if isinstance(results, statsmodels.regression.linear_model.
↪RegressionResultsWrapper) is False:
            raise TypeError("result must be instance of statsmodels.regression.
↪linear_model.RegressionResultsWrapper object")
```

```python
        self.results = maybe_unwrap_results(results)

        self.y_true = self.results.model.endog
        self.y_predict = self.results.fittedvalues
        self.xvar = self.results.model.exog
        self.xvar_names = self.results.model.exog_names

        self.residual = np.array(self.results.resid)
        influence = self.results.get_influence()
        self.residual_norm = influence.resid_studentized_internal
        self.leverage = influence.hat_matrix_diag
        self.cooks_distance = influence.cooks_distance[0]
        self.nparams = len(self.results.params)

    def __call__(self, plot_context='seaborn-paper'):
        # print(plt.style.available)
        with plt.style.context(plot_context):
            fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10,10))
            self.residual_plot(ax=ax[0,0])
            self.qq_plot(ax=ax[0,1])
            self.scale_location_plot(ax=ax[1,0])
            self.leverage_plot(ax=ax[1,1])
            plt.show()

        self.vif_table()
        return fig, ax


    def residual_plot(self, ax=None):
        """
        Residual vs Fitted Plot

        Graphical tool to identify non-linearity.
        (Roughly) Horizontal red line is an indicator that the residual has a␣
↪linear pattern
        """
        if ax is None:
            fig, ax = plt.subplots()

        sns.residplot(
            x=self.y_predict,
            y=self.residual,
            lowess=True,
            scatter_kws={'alpha': 0.5},
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8},
            ax=ax)
```

```
        # annotations
        residual_abs = np.abs(self.residual)
        abs_resid = np.flip(np.sort(residual_abs))
        abs_resid_top_3 = abs_resid[:3]
        for i, _ in enumerate(abs_resid_top_3):
            ax.annotate(
                i,
                xy=(self.y_predict[i], self.residual[i]),
                color='C3')

        ax.set_title('Residuals vs Fitted', fontweight="bold")
        ax.set_xlabel('Fitted values')
        ax.set_ylabel('Residuals')
        return ax

    def qq_plot(self, ax=None):
        """
        Standarized Residual vs Theoretical Quantile plot

        Used to visually check if residuals are normally distributed.
        Points spread along the diagonal line will suggest so.
        """
        if ax is None:
            fig, ax = plt.subplots()

        QQ = ProbPlot(self.residual_norm)
        QQ.qqplot(line='45', alpha=0.5, lw=1, ax=ax)

        # annotations
        abs_norm_resid = np.flip(np.argsort(np.abs(self.residual_norm)), 0)
        abs_norm_resid_top_3 = abs_norm_resid[:3]
        for r, i in enumerate(abs_norm_resid_top_3):
            ax.annotate(
                i,
                xy=(np.flip(QQ.theoretical_quantiles, 0)[r], self.
↪residual_norm[i]),
                ha='right', color='C3')

        ax.set_title('Normal Q-Q', fontweight="bold")
        ax.set_xlabel('Theoretical Quantiles')
        ax.set_ylabel('Standardized Residuals')
        return ax

    def scale_location_plot(self, ax=None):
        """
        Sqrt(Standarized Residual) vs Fitted values plot
```

```
        Used to check homoscedasticity of the residuals.
        Horizontal line will suggest so.
        """
        if ax is None:
            fig, ax = plt.subplots()

        residual_norm_abs_sqrt = np.sqrt(np.abs(self.residual_norm))

        ax.scatter(self.y_predict, residual_norm_abs_sqrt, alpha=0.5);
        sns.regplot(
            x=self.y_predict,
            y=residual_norm_abs_sqrt,
            scatter=False, ci=False,
            lowess=True,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8},
            ax=ax)

        # annotations
        abs_sq_norm_resid = np.flip(np.argsort(residual_norm_abs_sqrt), 0)
        abs_sq_norm_resid_top_3 = abs_sq_norm_resid[:3]
        for i in abs_sq_norm_resid_top_3:
            ax.annotate(
                i,
                xy=(self.y_predict[i], residual_norm_abs_sqrt[i]),
                color='C3')
        ax.set_title('Scale-Location', fontweight="bold")
        ax.set_xlabel('Fitted values')
        ax.set_ylabel(r'$\sqrt{|\mathrm{Standardized\ Residuals}|}$');
        return ax

    def leverage_plot(self, ax=None):
        """
        Residual vs Leverage plot

        Points falling outside Cook's distance curves are considered␣
↪observation that can sway the fit
        aka are influential.
        Good to have none outside the curves.
        """
        if ax is None:
            fig, ax = plt.subplots()

        ax.scatter(
            self.leverage,
            self.residual_norm,
            alpha=0.5);
```

```
        sns.regplot(
            x=self.leverage,
            y=self.residual_norm,
            scatter=False,
            ci=False,
            lowess=True,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8},
            ax=ax)

        # annotations
        leverage_top_3 = np.flip(np.argsort(self.cooks_distance), 0)[:3]
        for i in leverage_top_3:
            ax.annotate(
                i,
                xy=(self.leverage[i], self.residual_norm[i]),
                color = 'C3')

        xtemp, ytemp = self.__cooks_dist_line(0.5) # 0.5 line
        ax.plot(xtemp, ytemp, label="Cook's distance", lw=1, ls='--', color='red')
        xtemp, ytemp = self.__cooks_dist_line(1) # 1 line
        ax.plot(xtemp, ytemp, lw=1, ls='--', color='red')

        ax.set_xlim(0, max(self.leverage)+0.01)
        ax.set_title('Residuals vs Leverage', fontweight="bold")
        ax.set_xlabel('Leverage')
        ax.set_ylabel('Standardized Residuals')
        ax.legend(loc='upper right')
        return ax

    def vif_table(self):
        """
        VIF table

        VIF, the variance inflation factor, is a measure of multicollinearity.
        VIF > 5 for a variable indicates that it is highly collinear with the
        other input variables.
        """
        vif_df = pd.DataFrame()
        vif_df["Features"] = self.xvar_names
        vif_df["VIF Factor"] = [variance_inflation_factor(self.xvar, i) for i in range(self.xvar.shape[1])]

        print(vif_df
                .sort_values("VIF Factor")
                .round(2))
```

```python


    def __cooks_dist_line(self, factor):
        """
        Helper function for plotting Cook's distance curves
        """
        p = self.nparams
        formula = lambda x: np.sqrt((factor * p * (1 - x)) / x)
        x = np.linspace(0.001, max(self.leverage), 50)
        y = formula(x)
        return x, y
```

1.3 Problem 1

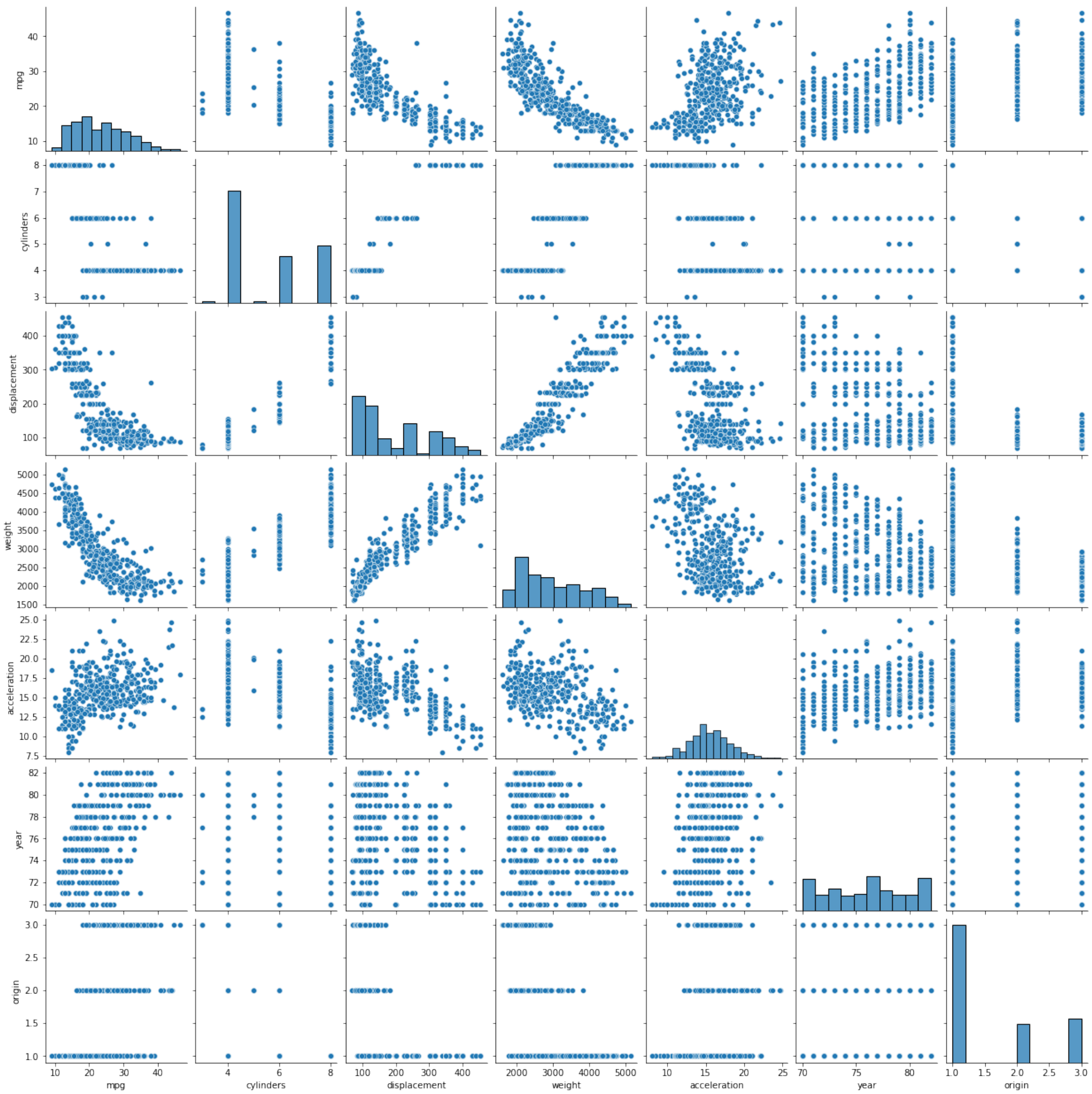
[3]:
```python
auto = pd.read_csv("auto.csv")
auto.head()
```

[3]:
```
    mpg  cylinders  displacement horsepower  weight  acceleration  year  \
0  18.0          8         307.0        130    3504          12.0    70
1  15.0          8         350.0        165    3693          11.5    70
2  18.0          8         318.0        150    3436          11.0    70
3  16.0          8         304.0        150    3433          12.0    70
4  17.0          8         302.0        140    3449          10.5    70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

1.3.1 Produce Scatter Matrix of Auto Dataset

[4]:
```python
sns.pairplot(auto)
```

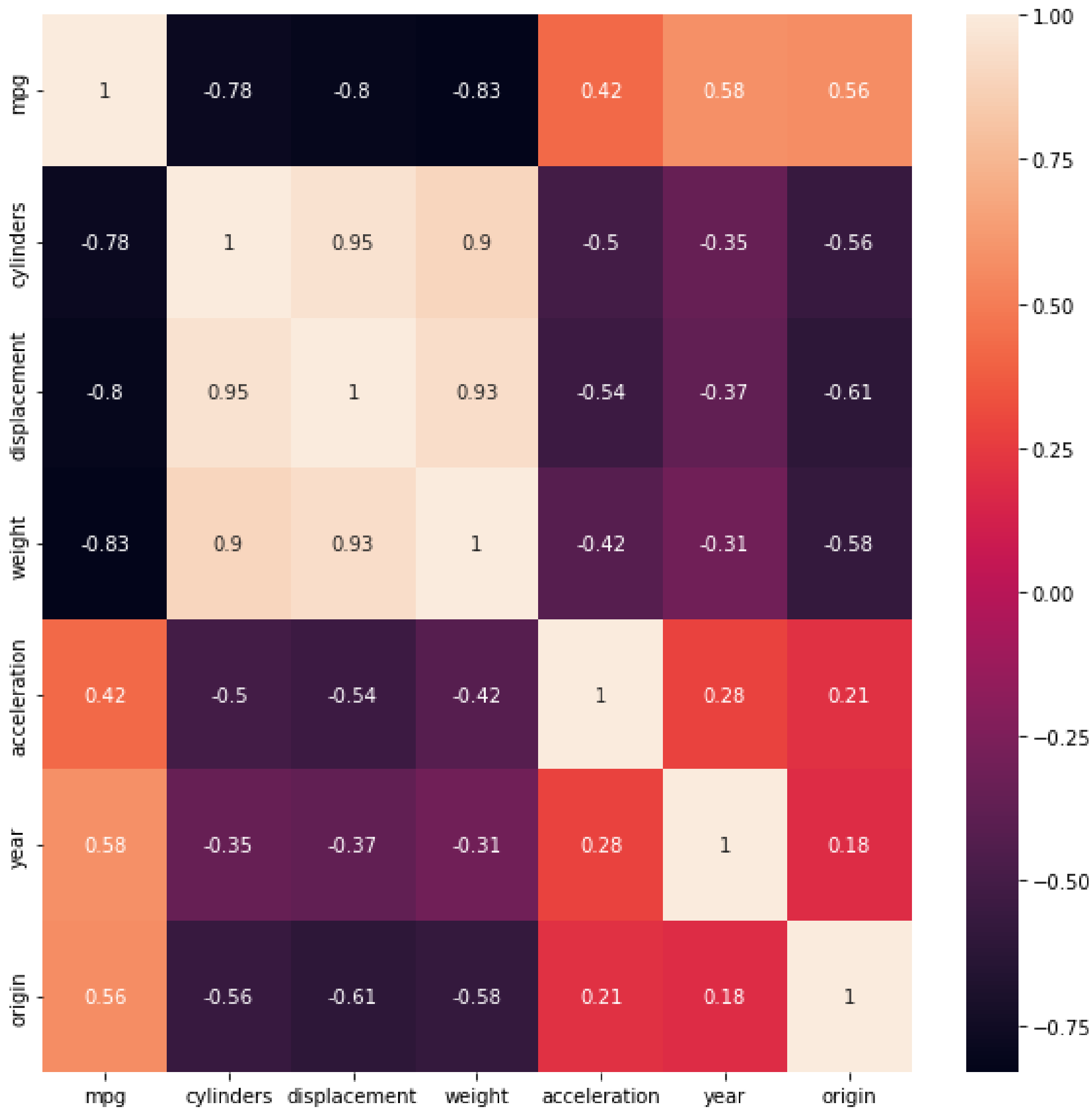
[4]:
```
<seaborn.axisgrid.PairGrid at 0x17ba293ccd0>
```



1.3.2 Correlation Matrix of Auto Dataset

[5]:
```
plt.figure(figsize = (10,10))
sns.heatmap(auto.corr(),annot = True)
```

[5]: <AxesSubplot:>



1.3.3 Multiple Regression Model

[6]:
```
mod = smf.ols(formula =␣
↪"mpg~cylinders+displacement+weight+acceleration+year+origin", data = auto).
↪fit()
mod.summary()
```

[6]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.821
```

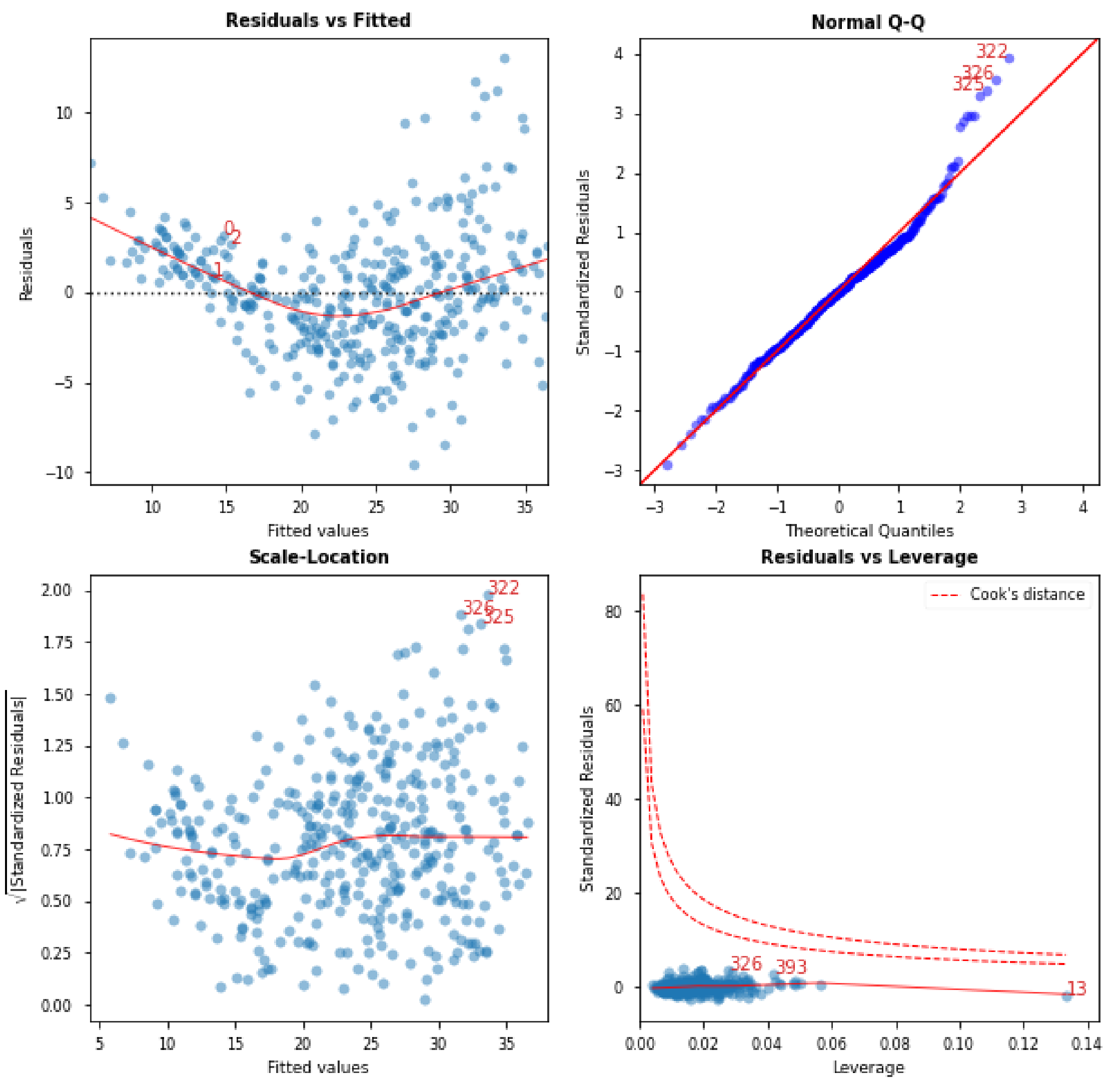
```
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     298.9
Date:                Mon, 26 Sep 2022   Prob (F-statistic):          1.72e-142
Time:                        19:55:14   Log-Likelihood:                -1037.7
No. Observations:                 397   AIC:                             2089.
Df Residuals:                     390   BIC:                             2117.
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept      -20.1358      4.145     -4.858      0.000     -28.286     -11.986
cylinders       -0.4198      0.320     -1.311      0.191      -1.049       0.210
displacement     0.0174      0.007      2.423      0.016       0.003       0.032
weight          -0.0069      0.001    -11.983      0.000      -0.008      -0.006
acceleration     0.1591      0.077      2.055      0.041       0.007       0.311
year             0.7703      0.049     15.613      0.000       0.673       0.867
origin           1.3560      0.269      5.040      0.000       0.827       1.885
==============================================================================
Omnibus:                       29.082   Durbin-Watson:                   1.289
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.906
Skew:                           0.494   Prob(JB):                     6.52e-11
Kurtosis:                       4.363   Cond. No.                     7.68e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 7.68e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

There exists a relationship between the predictors and response. There is a significant relationship between all predictor variables except cylinders and the response MPG. That for each year newer, the MPG increases by 0.7703.

1.3.4 Diagnostic Plots

[7]:
```
cls = Linear_Reg_Diagnostic(mod)
fig, ax = cls()
```

```
C:\Users\matth\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:993:
UserWarning: marker is redundantly defined by the 'marker' keyword argument and
the fmt string "bo" (-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



	Features	VIF Factor
5	year	1.18
4	acceleration	1.62
6	origin	1.66
3	weight	8.57
1	cylinders	10.59
2	displacement	20.08
0	Intercept	614.21

There seem to be no major outliers. There is a problem with collinearity as the VIF scores for Cylinders and Displacement are above 10. (Hint, displacement and Weight probably are very correlated). There also seems to be an issue with failing the homoscedasticity assumption of regression. No observations have unusually high leverage.

[8]:
```
mod2 = smf.ols(formula =␣
↪"mpg~displacement+acceleration+origin+displacement*weight+acceleration*year+acceleration*or
↪data = auto).fit()
mod2.summary()
```

[8]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.875
Model:                            OLS   Adj. R-squared:                  0.872
Method:                 Least Squares   F-statistic:                     301.9
Date:                Mon, 26 Sep 2022   Prob (F-statistic):          6.02e-169
Time:                        19:55:15   Log-Likelihood:                -966.31
No. Observations:                 397   AIC:                             1953.
Df Residuals:                     387   BIC:                             1992.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Intercept              97.6260     18.132      5.384      0.000      61.977
133.275
displacement           -0.0726      0.009     -8.360      0.000      -0.090
-0.056
acceleration           -5.4437      1.101     -4.943      0.000      -7.609
-3.279
origin                -17.2712      4.258     -4.056      0.000     -25.643
-8.900
weight                 -0.0097      0.001    -14.334      0.000      -0.011
-0.008
displacement:weight  1.902e-05   2.13e-06      8.933      0.000    1.48e-05
2.32e-05
year                   -0.5091      0.240     -2.120      0.035      -0.981
-0.037
acceleration:year       0.0671      0.015      4.580      0.000       0.038
0.096
acceleration:origin     0.3186      0.090      3.521      0.000       0.141
0.496
year:origin             0.1604      0.051      3.123      0.002       0.059
0.261
==============================================================================
Omnibus:                       52.266   Durbin-Watson:                   1.571
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              132.630
```

```
Skew:                           0.648   Prob(JB):                     1.58e-29
Kurtosis:                       5.518   Cond. No.                     1.09e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.09e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

These interactions selected are statistically significant

1.3.5 Transformations

[9]:
```python
mod3 = smf.ols(formula = "np.
↪log(mpg)~displacement+acceleration+origin+displacement*weight+acceleration*year+acceleratio
↪data = auto).fit()
mod3.summary()
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            np.log(mpg)   R-squared:                       0.896
Model:                            OLS   Adj. R-squared:                  0.894
Method:                 Least Squares   F-statistic:                     371.6
Date:                Mon, 26 Sep 2022   Prob (F-statistic):          2.24e-184
Time:                        19:55:15   Log-Likelihood:                 315.21
No. Observations:                 397   AIC:                            -610.4
Df Residuals:                     387   BIC:                            -570.6
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Intercept               4.8946      0.719      6.811      0.000       3.482
6.308
displacement           -0.0020      0.000     -5.755      0.000      -0.003
-0.001
acceleration           -0.1872      0.044     -4.288      0.000      -0.273
-0.101
origin                 -0.2509      0.169     -1.487      0.138      -0.583
0.081
weight                 -0.0004   2.68e-05    -13.122      0.000      -0.000
```

```
-0.000
displacement:weight  4.502e-07   8.44e-08      5.334      0.000    2.84e-07
6.16e-07
year                   -0.0066      0.010     -0.695      0.487      -0.025
0.012
acceleration:year       0.0023      0.001      3.994      0.000       0.001
0.003
acceleration:origin     0.0100      0.004      2.775      0.006       0.003
0.017
year:origin             0.0012      0.002      0.613      0.540      -0.003
0.005
==============================================================================
Omnibus:                       12.484   Durbin-Watson:                   1.506
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               25.414
Skew:                          -0.043   Prob(JB):                     3.03e-06
Kurtosis:                       4.237   Cond. No.                     1.09e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.09e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[10]:
```
mod4 = smf.ols(formula = "np.
↪sqrt(mpg)~displacement+acceleration+origin+displacement*weight+acceleration*year+accelerati
↪data = auto).fit()
mod4.summary()
```

[10]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            np.sqrt(mpg)   R-squared:                       0.890
Model:                             OLS   Adj. R-squared:                  0.887
Method:                  Least Squares   F-statistic:                     347.7
Date:                 Mon, 26 Sep 2022   Prob (F-statistic):         2.14e-179
Time:                         19:55:15   Log-Likelihood:                -38.231
No. Observations:                  397   AIC:                             96.46
Df Residuals:                      387   BIC:                             136.3
Df Model:                            9
Covariance Type:             nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
```

```
--------------------------------------------------------------------------------
-------
Intercept              10.7152      1.751      6.121      0.000       7.273
14.157
displacement           -0.0062      0.001     -7.407      0.000      -0.008
-0.005
acceleration           -0.5003      0.106     -4.706      0.000      -0.709
-0.291
origin                 -1.1895      0.411     -2.893      0.004      -1.998
-0.381
weight                 -0.0009   6.53e-05    -14.095      0.000      -0.001
-0.001
displacement:weight  1.541e-06   2.06e-07      7.495      0.000    1.14e-06
1.94e-06
year                   -0.0337      0.023     -1.452      0.147      -0.079
0.012
acceleration:year       0.0062      0.001      4.368      0.000       0.003
0.009
acceleration:origin     0.0281      0.009      3.222      0.001       0.011
0.045
year:origin             0.0098      0.005      1.972      0.049    2.76e-05
0.020
==============================================================================
Omnibus:                       24.247   Durbin-Watson:                   1.542
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.261
Skew:                           0.308   Prob(JB):                     2.72e-12
Kurtosis:                       4.685   Cond. No.                     1.09e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.09e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[11]:
```python
mod4 = smf.ols(formula = "np.
 ↪square(mpg)~displacement+acceleration+origin+displacement*weight+acceleration*year+accelera
 ↪data = auto).fit()
mod4.summary()
```

[11]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:          np.square(mpg)   R-squared:                       0.824
Model:                             OLS   Adj. R-squared:                  0.820
```

```
Method:                 Least Squares   F-statistic:                     202.0
Date:                Mon, 26 Sep 2022   Prob (F-statistic):          2.70e-140
Time:                        19:55:15   Log-Likelihood:                -2597.8
No. Observations:                 397   AIC:                             5216.
Df Residuals:                     387   BIC:                             5255.
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept            5991.0733   1104.550      5.424      0.000    3819.404
8162.742
displacement           -4.5318      0.529     -8.569      0.000      -5.572
-3.492
acceleration         -336.6730     67.084     -5.019      0.000    -468.568
-204.778
origin              -1413.9771    259.382     -5.451      0.000   -1923.951
-904.003
weight                 -0.5496      0.041    -13.340      0.000      -0.631
-0.469
displacement:weight     0.0013      0.000      9.850      0.000       0.001
0.002
year                  -44.7338     14.626     -3.058      0.002     -73.491
-15.977
acceleration:year       4.1572      0.893      4.655      0.000       2.401
5.913
acceleration:origin    20.4312      5.512      3.707      0.000       9.594
31.268
year:origin            14.3082      3.128      4.574      0.000       8.157
20.459
==============================================================================
Omnibus:                      127.171   Durbin-Watson:                   1.611
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              547.287
Skew:                           1.338   Prob(JB):                    1.44e-119
Kurtosis:                       8.091   Cond. No.                     1.09e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.09e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Square root and log transform worked the best

1.4 Problem 2

1.4.1 Prepare Dataset

[12]:
```
df = pd.read_csv("AirQuality.csv", index_col = "No")
df.head()
```

[12]:

	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
No												
1	2010	1	1	0	NaN	-21	-11.0	1021.0	NW	1.79	0	0
2	2010	1	1	1	NaN	-21	-12.0	1020.0	NW	4.92	0	0
3	2010	1	1	2	NaN	-21	-11.0	1019.0	NW	6.71	0	0
4	2010	1	1	3	NaN	-21	-14.0	1019.0	NW	9.84	0	0
5	2010	1	1	4	NaN	-20	-12.0	1018.0	NW	12.97	0	0

[13]:
```
df = df.dropna()
df.head()
```

[13]:

	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
No												
25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0
26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	2.68	0	0
27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	3.57	0	0
28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	5.36	1	0
29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	6.25	2	0

[14]:
```
df = df[df["pm2.5"] != 0]
df.head()
```

[14]:

	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
No												
25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0
26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	2.68	0	0
27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	3.57	0	0
28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	5.36	1	0
29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	6.25	2	0

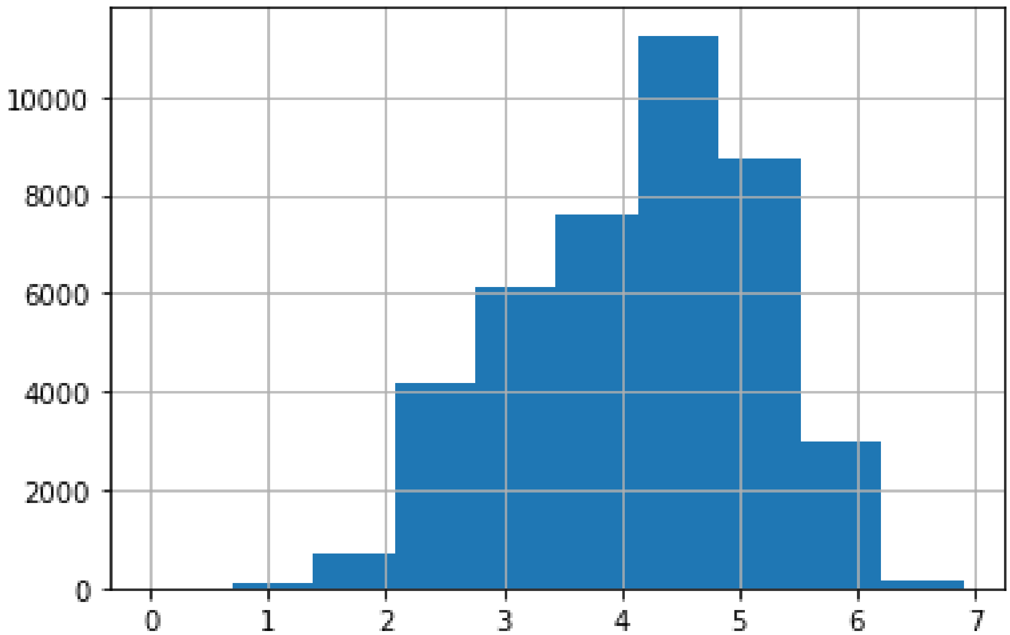
[15]:
```
df['log pm2.5'] = np.log(df['pm2.5'])
```

[16]:
```
df['date'] = pd.to_datetime(df[['year', 'month', 'day', 'hour']], format = '%Y/
↪%M/%D %H')
```

1.4.2 Historgram of Log transformed pm2.5

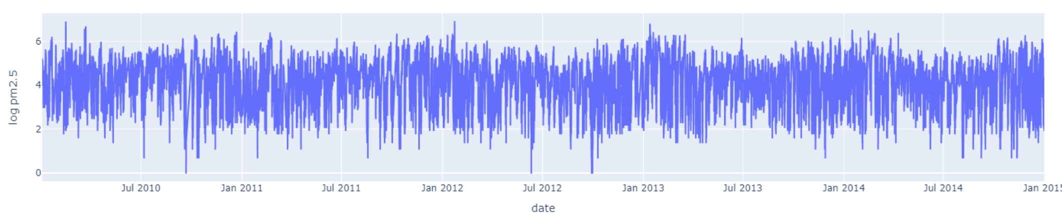
[17]:
```
df['log pm2.5'].hist()
```

[17]: <AxesSubplot:>



1.4.3 Timeseries of Polution

[18]:
```
fig = px.line(df, x='date', y="log pm2.5")
fig.show()
```

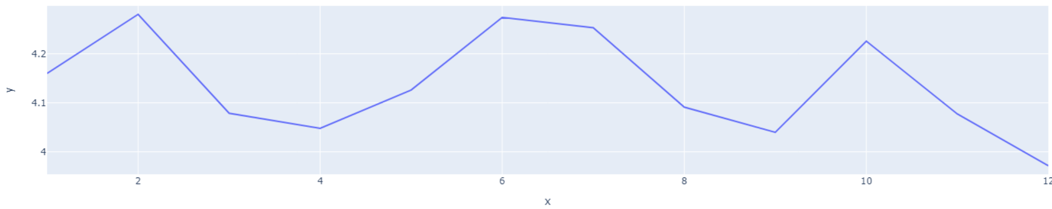


[19]:
```
h = df['date'].dt.hour
d = df['date'].dt.day
m = df['date'].dt.month
y = df['date'].dt.year
```

Polution seems to be not increase over time. It hovers around the same area of values.

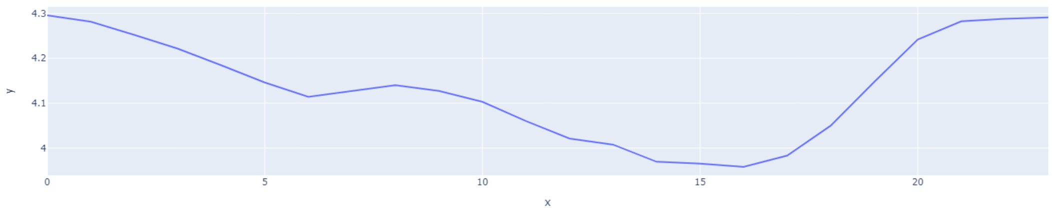
1.4.4 Polution Average Per Month

[20]:
```
result = df["log pm2.5"].groupby(m).mean()
fig = px.line(x=result.index, y=result)
fig.show()
```



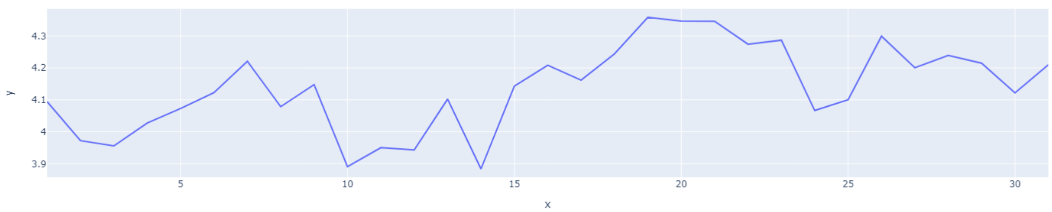
1.4.5 Plot Average per Hour

[21]:
```
result = df["log pm2.5"].groupby(h).mean()
fig = px.line(x=result.index, y=result)
fig.show()
```



1.4.6 Plot average per day

[22]:
```
result = df["log pm2.5"].groupby(d).mean()
fig = px.line(x=result.index, y=result)
fig.show()
```

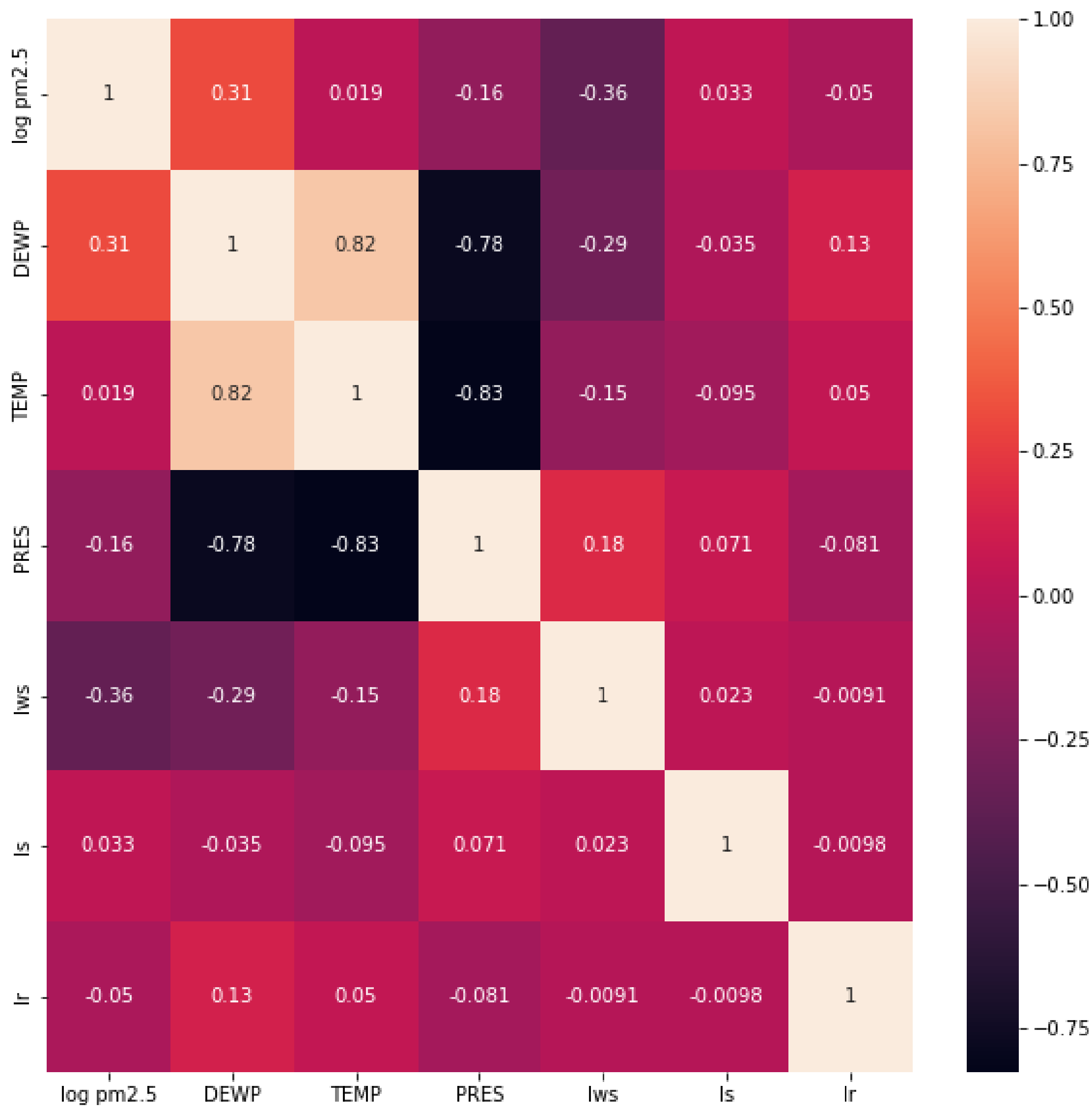


So for Day and Month I do not think it is a periodic trend, and are more stochastic in nature, as the end parts of the graphs are at different values. For the hour graph, there deffinetly is periodity as the ends of the graphs are pretty much continuous. If this was Z time, this graph would make sense with peaks near the morning (Add 8 to each hour to adjust) and minima late at night.

1.4.7 Relations Between Environment and Polution and Environmental Factors among themselves

[23]:
```
plt.figure(figsize = (10,10))
sns.heatmap(df[["log pm2.5", "DEWP", "TEMP", "PRES", "cbwd", "Iws", "Is",␣
↪"Ir"]].corr(),annot = True)
```

[23]:
```
<AxesSubplot:>
```



There seems to be some relationship between Dew Point, Pressure, and windspeed with polution. Dewpoint is correlated heavilly with temperature and pressure (this makes sense), Temperature and pressure are correlated (also makes sense), Windspeed might also have relationships with dewpoint, temperature, and pressure, but no as strong as the aforementioned relationships.

1.4.8 Cyclical Transformation

[24]:
```
def encode(data, col, max_val):
    data[col + '_sin'] = np.sin(2 * np.pi * data[col]/max_val)
    data[col + '_cos'] = np.cos(2 * np.pi * data[col]/max_val)
    return data
df = encode(df, 'month', 12)
df = encode(df, 'day', 31)
df = encode(df, 'hour', 23)
```

[25]:
```
df = df.drop(["month","day","hour","date"],axis = 1)
```

[26]:
```
df.head()
```

[26]:
```
    year  pm2.5  DEWP  TEMP    PRES cbwd   Iws  Is  Ir  log pm2.5  month_sin  \
No
25  2010  129.0   -16  -4.0  1020.0   SE  1.79   0   0   4.859812        0.5
26  2010  148.0   -15  -4.0  1020.0   SE  2.68   0   0   4.997212        0.5
27  2010  159.0   -11  -5.0  1021.0   SE  3.57   0   0   5.068904        0.5
28  2010  181.0    -7  -5.0  1022.0   SE  5.36   1   0   5.198497        0.5
29  2010  138.0    -7  -5.0  1022.0   SE  6.25   2   0   4.927254        0.5

    month_cos   day_sin   day_cos  hour_sin  hour_cos
No
25   0.866025  0.394356  0.918958  0.000000  1.000000
26   0.866025  0.394356  0.918958  0.269797  0.962917
27   0.866025  0.394356  0.918958  0.519584  0.854419
28   0.866025  0.394356  0.918958  0.730836  0.682553
29   0.866025  0.394356  0.918958  0.887885  0.460065
```

1.4.9 Train Test Split

[27]:
```
X = df[["year", "DEWP", "TEMP", "PRES", "Iws", "Is", "Ir", "month_sin",␣
 ↪"month_cos", "day_sin", "day_cos", "hour_sin", "hour_cos"]]
y = df["log pm2.5"]
```

[28]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=101)
```

1.4.10 Multivariate Model

[29]:
```
mod = sm.OLS(y_train, X_train).fit()
mod.summary()
```

[29]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:             log pm2.5   R-squared (uncentered):
0.970
Model:                           OLS   Adj. R-squared (uncentered):
0.970
Method:                Least Squares   F-statistic:
8.407e+04
Date:               Mon, 26 Sep 2022   Prob (F-statistic):
0.00
Time:                       19:55:17   Log-Likelihood:
-37066.
No. Observations:              33404   AIC:
7.416e+04
Df Residuals:                  33391   BIC:
7.427e+04
Df Model:                         13
Covariance Type:           nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
year           0.0106      0.000     26.401      0.000       0.010       0.011
DEWP           0.1017      0.001    140.896      0.000       0.100       0.103
TEMP          -0.0367      0.001    -32.536      0.000      -0.039      -0.035
PRES          -0.0166      0.001    -20.963      0.000      -0.018      -0.015
Iws           -0.0030   8.84e-05    -34.359      0.000      -0.003      -0.003
Is            -0.0425      0.005     -7.943      0.000      -0.053      -0.032
Ir            -0.0843      0.003    -29.571      0.000      -0.090      -0.079
month_sin      0.9534      0.011     84.063      0.000       0.931       0.976
month_cos      1.1313      0.016     68.695      0.000       1.099       1.164
day_sin       -0.0585      0.006    -10.320      0.000      -0.070      -0.047
day_cos       -0.0066      0.006     -1.144      0.253      -0.018       0.005
hour_sin      -0.1511      0.007    -21.193      0.000      -0.165      -0.137
hour_cos      -0.0321      0.006     -5.037      0.000      -0.045      -0.020
==============================================================================
Omnibus:                      880.713   Durbin-Watson:                   2.023
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1006.548
Skew:                          -0.367   Prob(JB):                    2.70e-219
Kurtosis:                       3.430   Cond. No.                     1.08e+04
==============================================================================
```

```
Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 1.08e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

1.4.11 Multivariate Model on Smaller Set of Predictors

[30]:
```python
mod_r = sm.OLS(y_train, X_train[["DEWP", "TEMP", "PRES", "Iws",␣
↪"Ir","month_sin","month_cos"]]).fit()
mod_r.summary()
```

[30]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:              log pm2.5   R-squared (uncentered):
0.969
Model:                            OLS   Adj. R-squared (uncentered):
0.969
Method:                 Least Squares   F-statistic:
1.504e+05
Date:                Mon, 26 Sep 2022   Prob (F-statistic):
0.00
Time:                        19:55:17   Log-Likelihood:
-37673.
No. Observations:               33404   AIC:
7.536e+04
Df Residuals:                   33397   BIC:
7.542e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
DEWP           0.1037      0.001    150.501      0.000       0.102       0.105
TEMP          -0.0153      0.001    -18.896      0.000      -0.017      -0.014
PRES           0.0042   1.09e-05    383.080      0.000       0.004       0.004
Iws           -0.0031   8.96e-05    -34.078      0.000      -0.003      -0.003
Ir            -0.0779      0.003    -26.949      0.000      -0.084      -0.072
month_sin      1.0550      0.011     97.548      0.000       1.034       1.076
month_cos      1.2137      0.015     80.737      0.000       1.184       1.243
```

```
==============================================================================
Omnibus:                      779.901   Durbin-Watson:                   2.025
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              872.752
Skew:                          -0.348   Prob(JB):                    3.05e-190
Kurtosis:                       3.377   Cond. No.                     4.36e+03
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 4.36e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

R Squared of the Reduced model is .001 less than the full one.

1.4.12 MSE for Full Model

[31]:
```python
y_pred = mod.predict(X_test)
mean_squared_error(y_test, y_pred)
```

[31]: 0.5256825001719991

1.4.13 MSE for Reduced Model

[32]:
```python
y_pred = mod_r.predict(X_test[["DEWP", "TEMP", "PRES", "Iws",␣
 ↪"Ir","month_sin","month_cos"]])
mean_squared_error(y_test, y_pred)
```

[32]: 0.5527035771591737

1.4.14 VIF for multicollinearity

[33]:
```python
vif_data = pd.DataFrame()
vif_data["feature"] = X_train.columns
vif_data["VIF"] = [variance_inflation_factor(X_train.values, i)
                          for i in range(len(X_train.columns))]
vif_data
```

[33]:

	feature	VIF
0	year	40463.527960
1	DEWP	6.828170
2	TEMP	23.872343
3	PRES	39939.700018
4	Iws	1.445356
5	Is	1.036855

```
6           Ir     1.057519
7    month_sin     3.962700
8    month_cos     8.455731
9      day_sin     1.009055
10     day_cos     1.004239
11    hour_sin     1.511662
12    hour_cos     1.305718
```

1.4.15 Final Interpretation of the Model

Polutions relation with time is strongly dependent on the month. The coefficients are significant and are higher than the coefficients for year, hour, and day. Certain Months are more associated with greater polution than others. Polution seems to peak when the sine and cosine of the month are higher. All the environmental factors except ls are well correlated with Polution and carry most of the information. Pressure has an extremely high VIF meaning other variables can predict it (most likely Temperature and Dewpoint.)